Instant Message Demo 项目文档

即时通讯系统

OverFlowLab

毛爽然 占海川 聂子豪 邹瀚元 周晋

软件工程 2023 春

目录 Contents

第一部分 综述

1.1 系统概述

Instant Message(即时通讯) 是指即时通讯系统，它是一种通过互联网或其他网络实现即时消息交流的技术。即时通讯允许用户发送和接收实时文本消息、多媒体文件、语音消息、视频聊天等内容。

即时通讯系统在个人和商业领域都广泛应用。个人用户可以使用即时通讯应用与朋友、家人和同事进行实时交流，而企业可以利用即时通讯系统提高团队协作效率，进行项目管理和远程办公等。

1.2 项目说明

本文档用于详细说明 OverFlowLab 小组开发的即时通讯系统，项目部署于 https://se-im-frontend-overflowlab.app.secoder.net。

本项目开发时间为 2023 年春季学期。

1.3 项目分工

前端：占海川、邹瀚元

后端：毛爽然、聂子豪、周晋

数据库：周晋

第二部分 需求分析

2.1 用户故事

开发过程中，我们听取用户故事，针对需求进行开发。

具体而言，作为 IM 系统的用户和作为会话的成员，存在着不同的需求。对于会话的成员，在私聊和群聊中有着不同的额外需求，而群聊中的成员存在着群主、管理员和普通用户三种身份，不同的身份需求也不相同。

具体的用户故事如下。

作为 IM 系统的用户，我想要：

1）注册、编辑我的账号，以便管理我的用户信息

2）登陆、登出我的账号，以便正常使用 IM 系统

3）搜索用户并浏览信息，以便了解其他的用户

4）添加好友和删除好友，以便管理我的好友

5）查看并管理好友列表和分组信息，以便查看和管理好友信息

6）创建群聊，以便与一组好友进行交流

作为会话的成员，我想要：

1）正常发送和接收信息，以便能与其他人进行实时通讯

2）获取消息的相关信息，以便了解消息的已读等属性

3）传递文件等多媒体信息，以便多维度传递和获取消息

4）获取和跳转 URL 链接，以便浏览所提及的页面

5）撤回消息，以便清除错误或不该发送的消息

6）删除消息，以便清除不想看见的消息

7）回复消息，以便对某个具体消息进行回复

8）转发消息，以便扩散一组消息

9）提及成员，以便更加鲜明的提示指定的成员

10）筛选聊天记录，以便准确获取所想要的信息

11）获取未读消息数，以便获取未了解的消息规模

12）将语音转成文字和翻译外语，以便准确和快速地获取信息

13）设置消息免打扰，以便防止受到不关心会话的影响

14）设置置顶会话，以便更容易看到我重视的会话

作为私聊成员，我想要:

1）进行语音通话和视频通话，以便和好友进行面对面交流

作为群聊成员，我想要:

1）浏览群聊基本信息，以便了解群聊的基本情况

2）邀请好友加入群聊，以便邀请好友入群

3）退出群聊，以便离开不关心的群聊

作为群聊管理员和群主，我想要:

1）撤销群消息，以便管理群聊消息

2）发布群公告，以便公布重要信息

3）移除群员，以便管理群成员

4）审核进群邀请，以便管理群成员

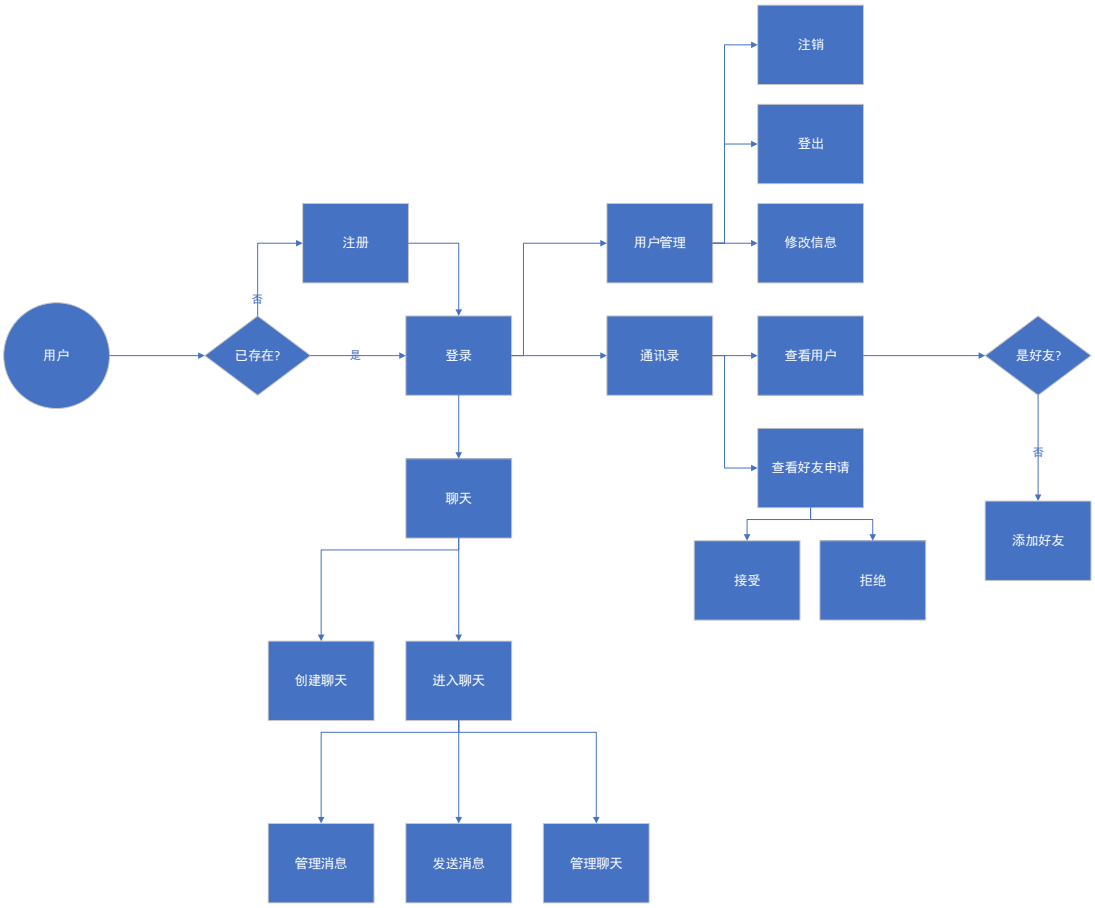
作为群聊群主，我想要:

1）任命和撤回群管理员，以便允许其他认可的成员管理群聊

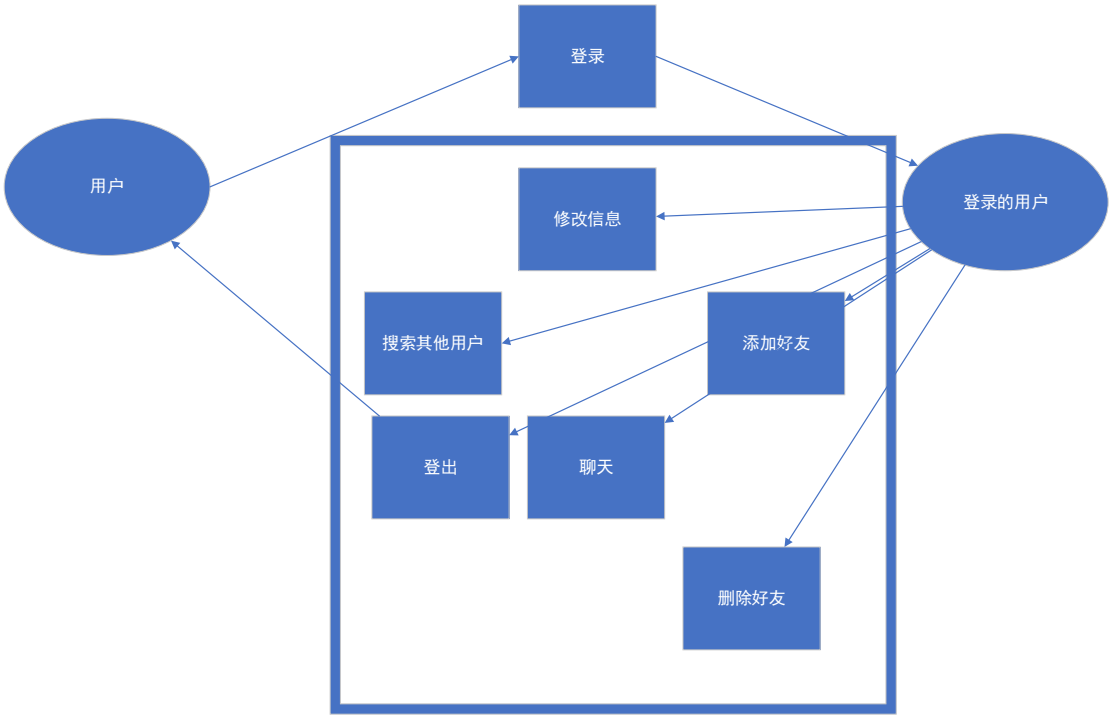
2）转让群主，以便实现群管理的交替

2.2 项目流程

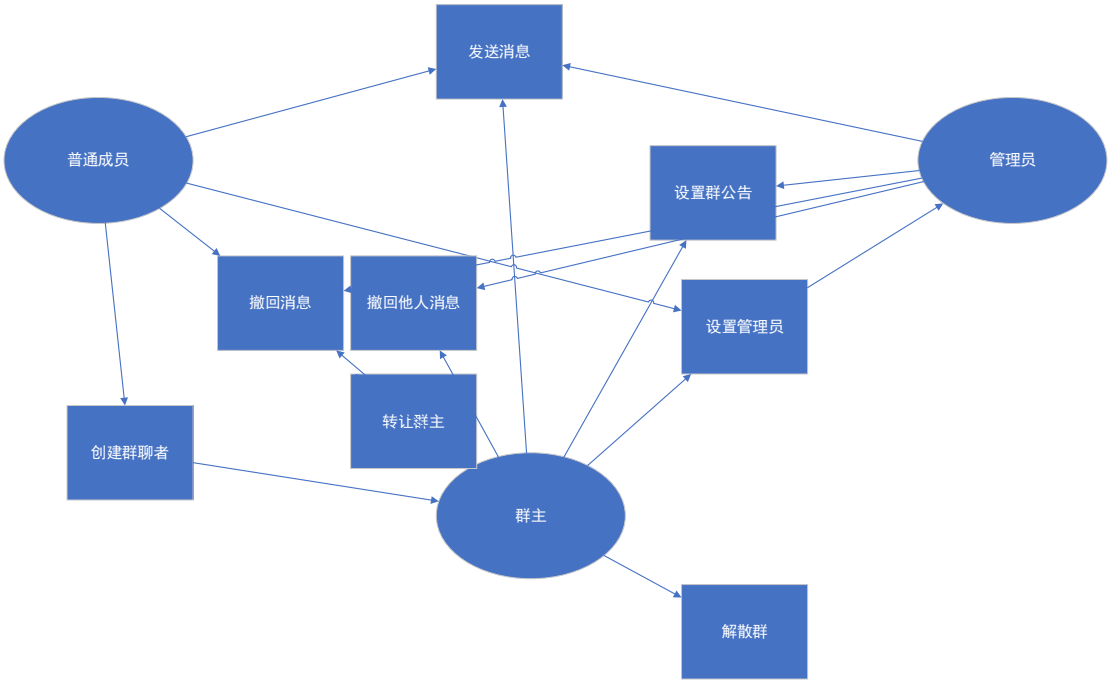
项目的整体流程如下图所示



项目的用户用例如下图所示



项目的会话用例如下图所示

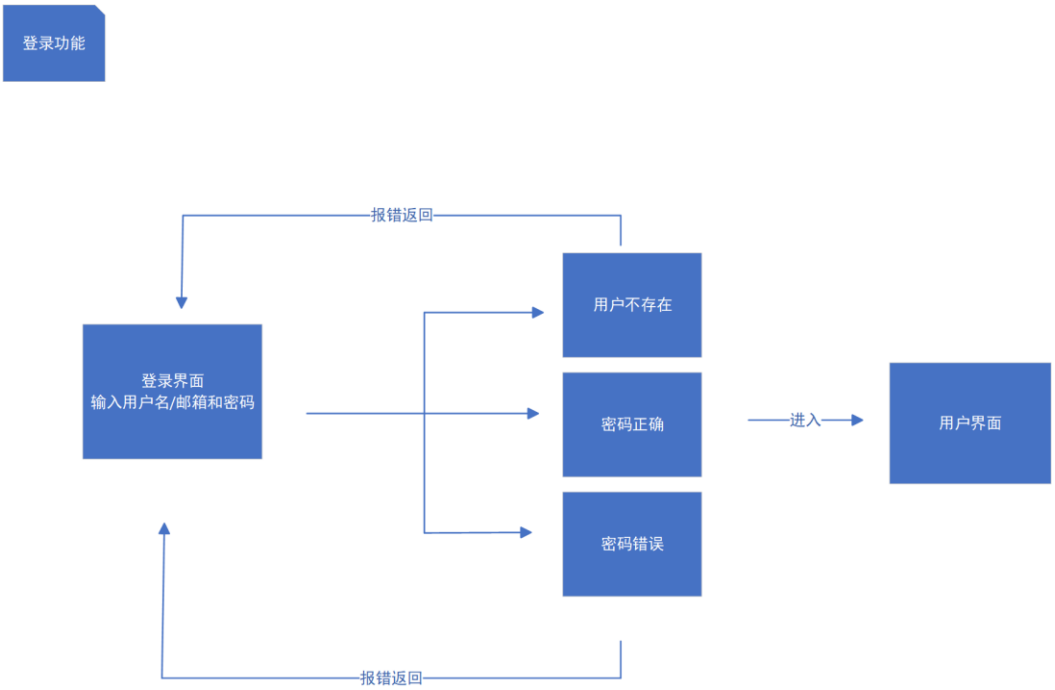


IM 系统的项目流程主要由用户管理、好友关系、通用绘画、群聊会话和视频聊天组成，具体流程如下所示：

a) 用户登录

1. 用户进入登录界面，填写用户名或邮箱和密码
2. 点击提交，如果该用户不存在，则提示错误并返回
3. 用户存在则检查密码是否正确，如果不正确就提示错误并返回
4. 密码正确，登录成功

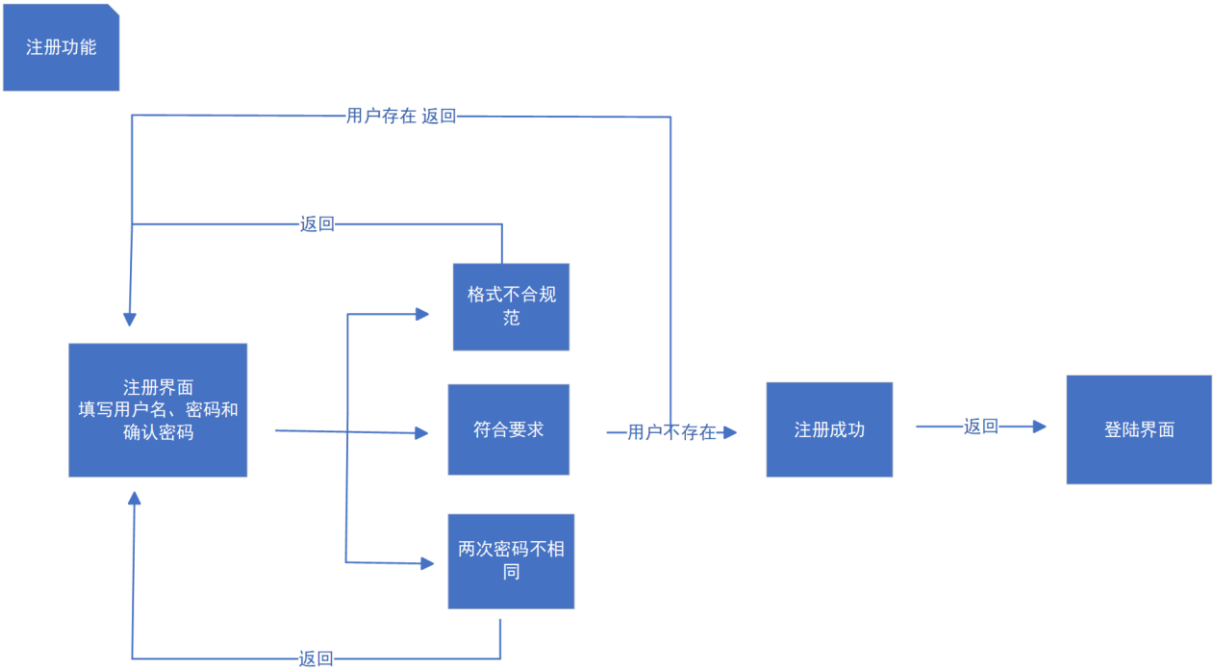
用户登录的具体流程如下图所示



b) 用户注册

1. 用户进入注册界面，填写用户名，两次填写密码
2. 点击提交，如果格式错误，则提示错误并返回
3. 格式正确则判断两次输入密码是否相同，不相同则提示错误并返回
4. 两次密码相同，检查输入的用户名是否已存在
5. 用户名不存在，创建新用户成功

用户注册的具体流程如下图所示



c) 信息管理

1. 用户进入个人信息设置界面，选择修改用户名/修改密码/修改邮箱/设置头像
   （1） 修改用户名
   填写修改的用户名和密码

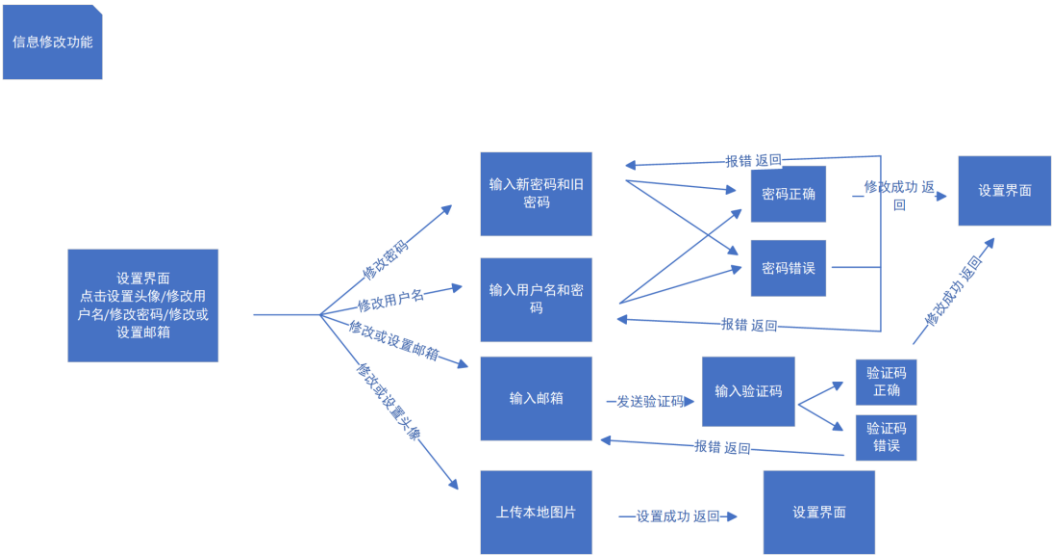
验证密码是否正确
验证成功后完成修改
完成设置后返回设置界面

(2) 修改密码
填写旧的密码并两次填写新的密码
验证旧密码是否正确，不正确则提示错误
验证成功后完成修改
完成设置后返回设置界面

(3) 修改邮箱
输入邮箱后点击发送验证码
给邮箱发送验证码
填写收到的验证码
验证码正确后完成邮箱修改
完成设置后返回设置界面

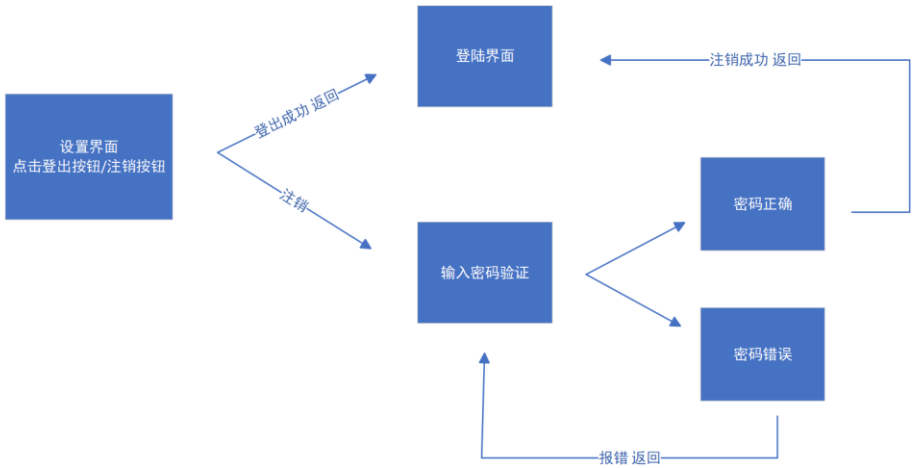
(4) 设置头像
选择本地的一份图片上传
完成设置后返回设置界面

信息修改的具体流程如下图所示



2. 用户登出
点击登出按钮
返回登录页面

3. 用户注销
输入密码
点击注销按钮
验证成功后注销账号
返回登录页面
登录和注销的具体流程如下图所示

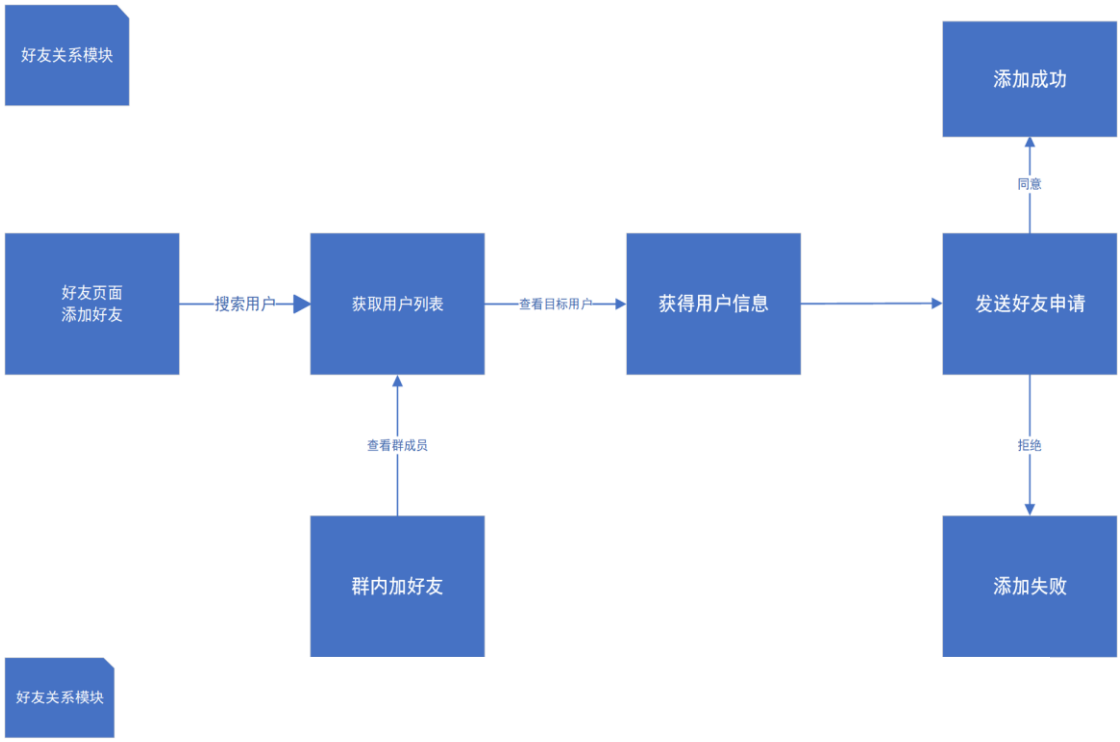


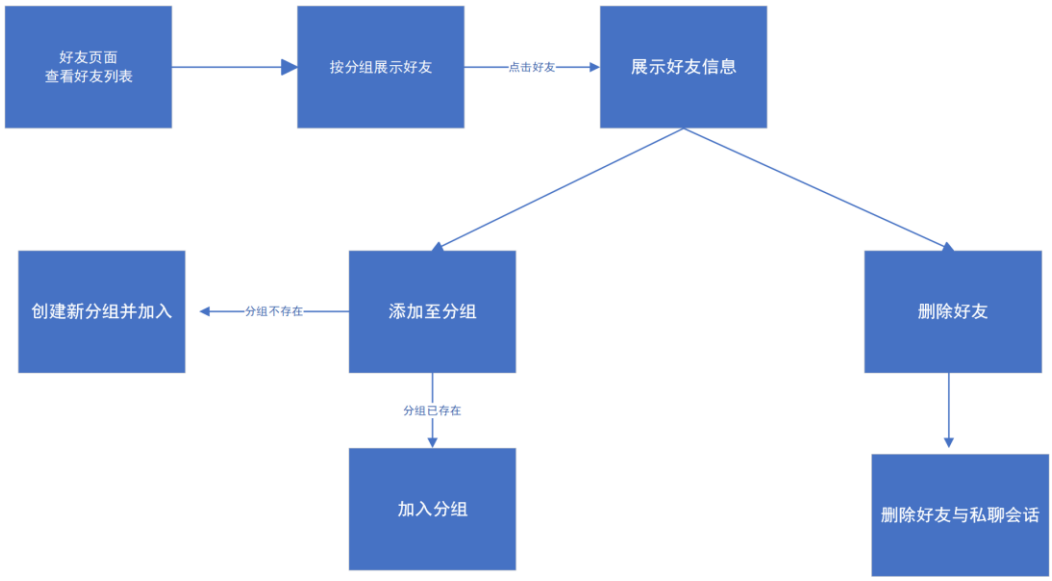
d) 好友关系

好友关系的功能梗概如下:

1. 用户进入通讯录页面选择搜索/查看好友申请
   (1) 选择搜索，输入非空的搜索关键字界面显示搜索到的若干用户
   点击进入该用户的公开信息界面
   点击添加好友，发送好友申请
   删除好友或选择添加至群组
   选择添加至群组，填写群组名并提交
   将好友添加至群组
   创建新群组
   (2) 选择查看好友申请，会列出所有自己发出的和收到的好友申请
   选择点击接受或拒绝好友申请
   查看对方是否已经接受/拒绝好友申请
   (3) 在通讯录中点击好友进入其公开信息界面
   删除空的分组

好友关系的具体流程如下图所示





e) 会话

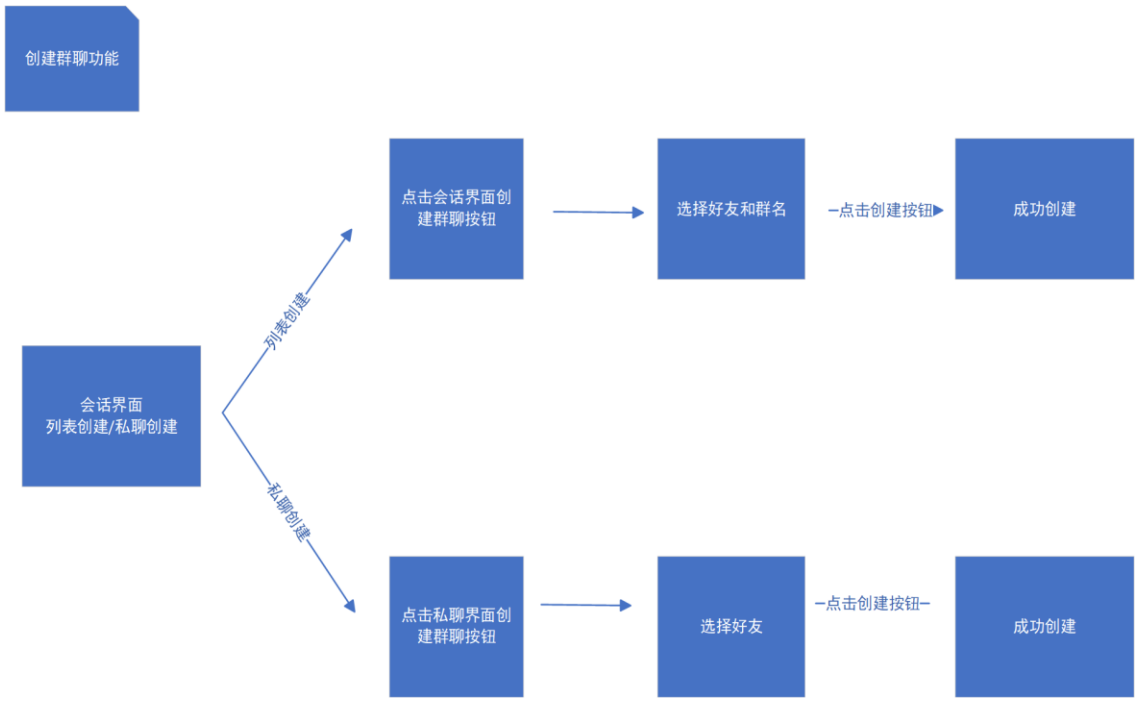
会话部分的功能梗概如下：

1. 点击会话页面，进入已有的会话或创建新会话
   选择创建新会话，输入会话名称并选择若干好友，创建新会话
2. 点击一个会话，显示会话的消息，在消息上右键对消息进行一系列操作
   进行消息的转发，删除，回复，撤回，翻译，语音转文字
3. 点击底部发送框，输入消息后点击发送
   点击表情图标，在弹出菜单中点击表情可输入

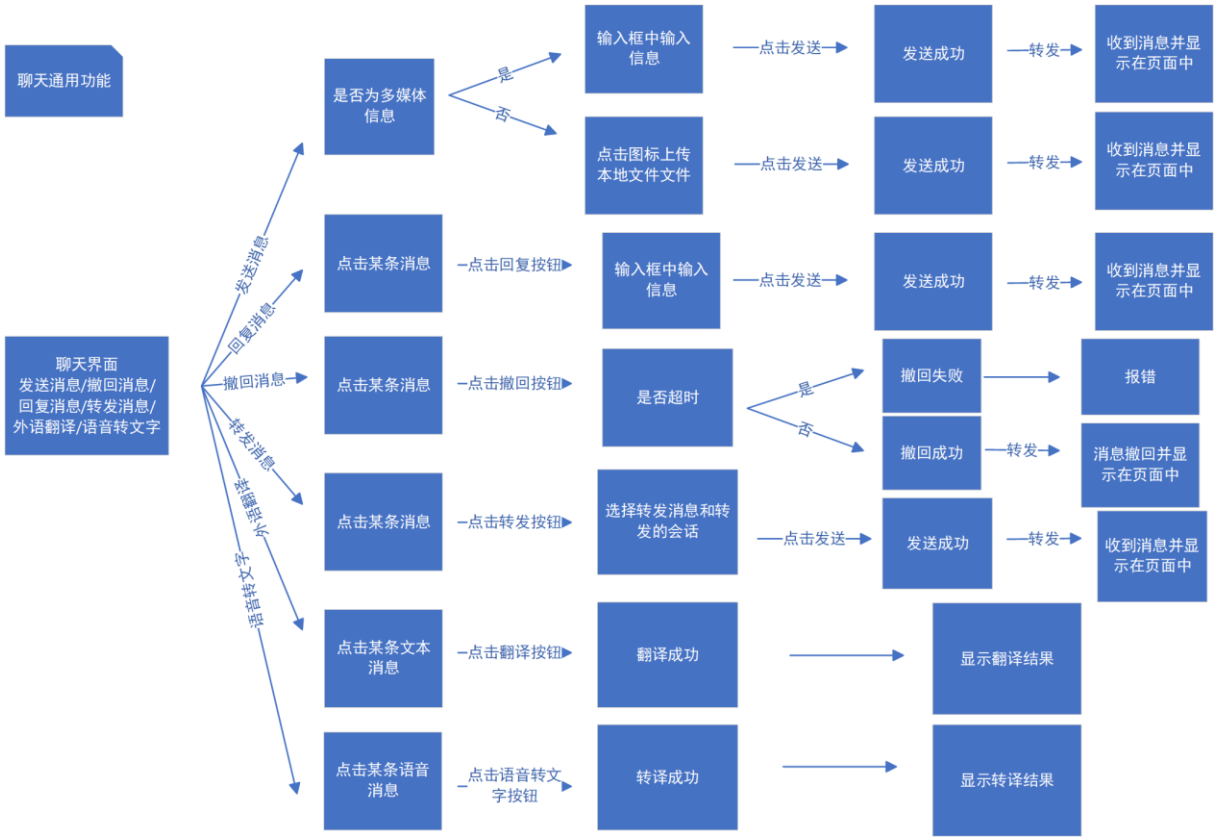
点击图片/音频/视频/文件按钮在弹出窗口中选择对应类型的文件，点击确认发送文件
点击视频聊天按钮，发起视频聊天
点击历史按钮，查看消息历史，查看所有历史/按照成员/时间/类型搜索历史
选择消息记录进行删除

4. 点击会话信息按钮，打开会话信息页面
设置会话置顶/免打扰/需要输入密码才能进入
作为群主将其他成员设为管理员/解除管理员，踢出群聊，将群主身份转让，编辑/查看群公告，解散群聊，撤回群消息
作为管理员将普通成员踢出群聊，编辑/查看群公告，撤回群消息
作为普通成员查看群公告，邀请新的成员，退出群聊

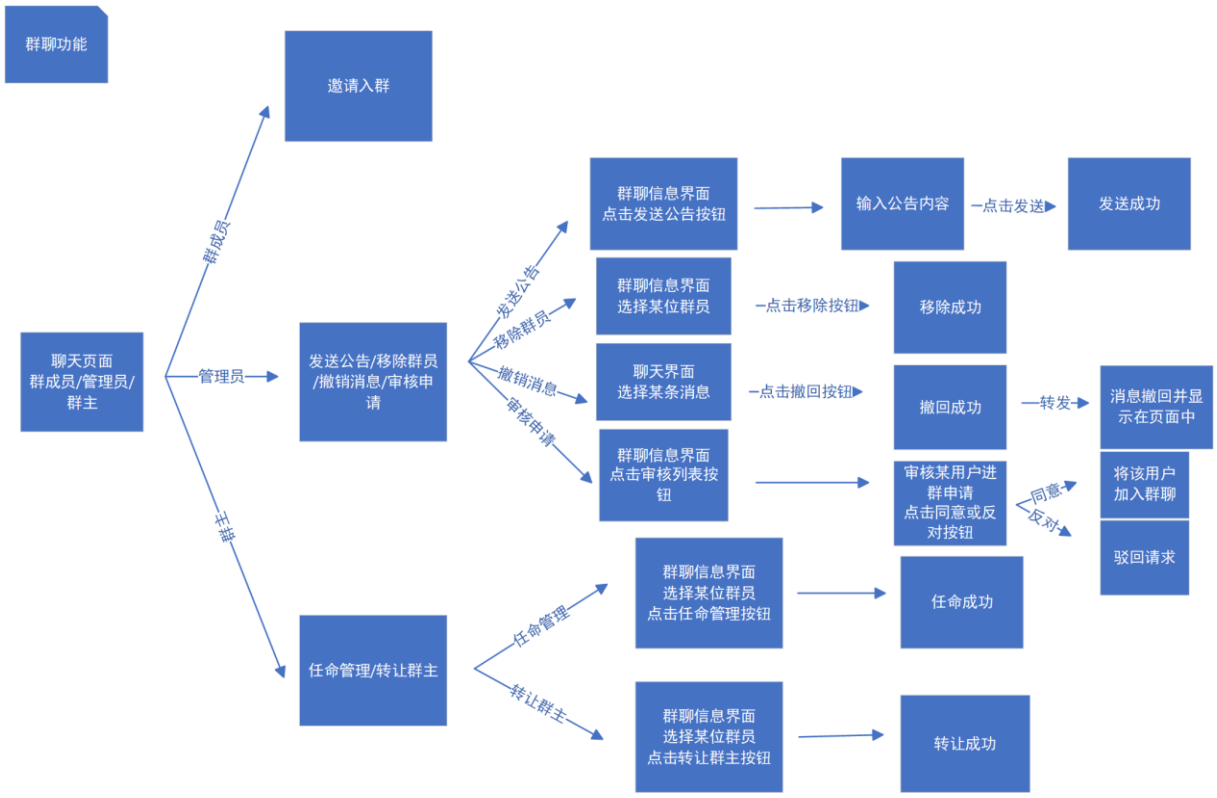
创建群聊的具体流程如下所示



聊天功能的具体流程如下所示

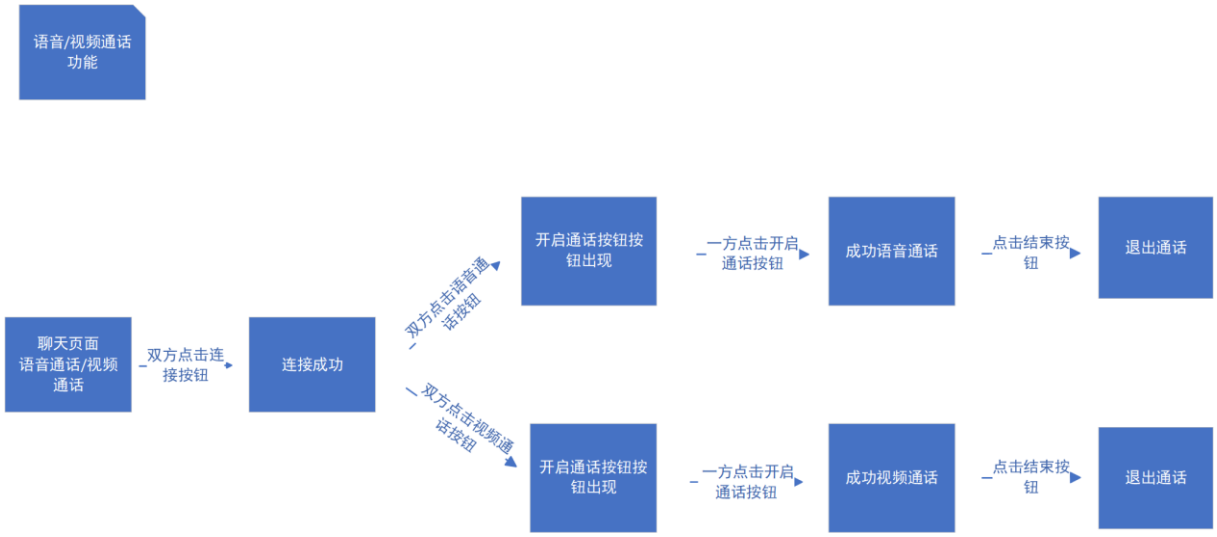


群聊功能的具体流程如下所示



f)视频通话

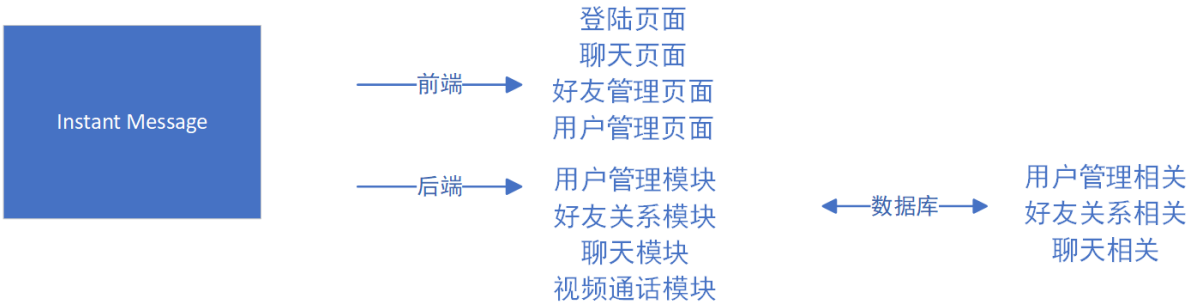
语音/视频通话的具体流程如下图所示



第三部分 模块与架构

3.1 整体架构设计

依照项目需求和开发实际情况，我们将项目进行如下划分：



根据功能划分，可以将 IM 划分为用户管理、好友关系、聊天、视频通话四个模块：

1）用户管理：支持基本鉴权和信息修改等用户相关的功能

用户管理模块主要负责的功能点如下：

用户注册
用户注销
登录登出
验证手段
信息编辑

2）好友关系：支持用户之间的好友添加、删除和添加小组等相关的功能

好友关系模块主要负责的功能点如下：

用户查找
好友申请
好友删除
好友列表

3）聊天：支持私聊、群聊发送消息、成员管理等基本功能及转发文件、翻译外语等附加功能

聊天模块主要负责的功能点如下：

二次验证
界面显示
发送信息
辅助信息
未读计数
多媒体
链接格式化
提及
提交反馈
基本要求
消息撤回

回复消息
外语翻译
消息转发
语音转文字
查看记录
筛选记录
删除记录
消息免打扰
置顶会话
列表创建
私聊创建
信息展示
群管理员
群主转让
群公告

4）视频通话：支持私聊用户之间的语音通话、视频通话功能

视频通话部分主要负责的功能点如下：

语音通话
视频通话

项目运行环境为 Web 环境，支持所有主流浏览器访问。

由于即时通讯系统的需要，项目协议采用 HTTP 和 Websocket 协议混合的方式。具体而言，用户管理模块采用 HTTP 的形式进行通讯；好友关系模块采用 HTTP 和 Websocket 混合的方式进行通讯：对于搜索用户等不需要即时反应的功能采用 HTTP 协议进行通讯，而对于添加好友等需要即时双方转发的功能采用 Websocket 协议通讯；聊天模块和视频通话模块采用 Websocket 协议通讯。

3.2 后端模块设计

后端主要使用 Django 进行开发，由于 Django 不支持 Websocket 协议，故采用 channels 建立 Websocket 连接。

后端主要分为用户管理（UserManage）、好友关系（FriendRelation）、聊天（Chat）和视频通话（Call）四个模块，均注册为 Django 的 INSTALLED_APPS 作为独立模块。

具体而言，用户管理模块和好友关系模块采用 HTTP 协议进行通讯，采用 Django 提供的默认方法实现，聊天模块和视频通话模块采用 Websocket 协议进行通讯，采用 channels 提供的 Consumer 方法实现。

用户管理模块的 urls 映射和大致功能如下所示：

```
urlpatterns = [
    # 用户登录
    path('login', views.user_login_pre_treat),

    # 用户注册
    path('register', views.user_register),

    # 绑定邮箱
    path('email', views.bind_email),

    # 发送邮箱
    path('send_email', views.send_email),

    # 注销用户
    path('cancel', views.user_cancel),

    # 用户登出
    path('logout', views.user_logout),

    # 修改用户信息
    path('revise', views.user_revise),

    # 上传头像
    path('upload', views.upload_avatar),

    # 上传文件
    path('uploadfile', views.upload),
```

```
    # 语音转文字
    path('audio', views.audio_to_text),

    # 返回转发信息
    path('message', views.fetch_message)
]
```

其中，path 的第一个参数为所对应的 Url 地址，第二个参数为所对应的处理函数。

这些函数包括了用户注册、登录登出、修改信息、用户注销的全部功能，同时链接到好友和聊天模块，在相关信息发生变动时修改其他模块的数据，这些功能在开发过程中时最先实现。

为了便于区分 HTTP 和 Websocket，一些需要用到 HTTP 请求的聊天功能也在这里实现，包括语音转文字、抓取转发信息、上传多媒体文件。

好友关系模块的 urls 映射和大致功能如下所示：

```
urlpatterns = [
    # 创建好友分组
    path('createfgroup', views.create_friend_group),

    # 添加好友到分组
    path('addfgroup', views.add_friend_group),

    # 搜索用户信息
    path('searchuser', views.search_user),

    # 查看用户信息
    path('checkuser', views.check_user),

    # 删除好友
    path('deletefriend', views.delete_friend),

    # 删除好友分组
    path('deletefgroup', views.delete_friend_group)
]
```

其中，path 的第一个参数为所对应的 Url 地址，第二个参数为所对应的处理函数。

这些函数包含好友管理的大部分功能，同时在删除好友时也相应地修改了 ChatRoom 部分的数据，这些在开发过程中于实现用户功能后完成。其中重要的添加好友部分，为了保持实时性，

放在了 WebSocket 部分实现。

聊天部分的 Websocket 后端处理函数映射和大致功能设置如下所示:

```
# 维持 Websocket 连接
if function == 'heartbeat':
    await self.heart_beat()

# 发送好友申请
elif function == 'apply':
    await self.apply_friend(json_info)

# 同意好友申请
elif function == 'confirm':
    await self.confirm_friend(json_info)

# 拒绝好友申请
elif function == 'decline':
    await self.decline_friend(json_info)

# 刷新页面信息
elif function == 'refresh':
    await self.refresh(json_info)

# 获取申请列表
elif function == 'fetchapplylist':
    await self.fetch_apply_list(json_info)

# 获取接受列表
elif function == 'fetchreceivelist':
    await self.fetch_reply_list(json_info)

# 获取好友列表
elif function == 'fetchfriendlist':
    await self.fetch_friend_list(json_info)

# 初始化所有私聊/群聊
elif function == 'add_channel':
    await self.add_channel(json_info)

# 连接某个私聊/群聊
elif function == 'add_chat':
    await self.add_chat(json_info)

# 离开某个私聊/群聊
```

```
elif function == 'leave_chat':
    await self.leave_chat(json_info)

# 发送各种消息
elif function == 'send_message':
    await self.send_message(json_info)

# 撤回消息
elif function == 'withdraw_message':
    await self.withdraw_message(json_info)

# 创建群聊
elif function == 'create_group':
    await self.create_group(json_info)

# 群主删除群聊
elif function == 'delete_chat_group':
    await self.delete_chat_group(json_info)

# 群主/管理员处理申请用户加入群聊信息
elif function == 'reply_add_group':
    await self.reply_add_group(json_info)

# 用户自己退出群聊
elif function == 'leave_group':
    await self.leave_group(json_info)

# 群主任命管理员
elif function == 'appoint_manager':
    await self.appoint_manager(json_info)

# 群主卸任管理员
elif function == 'remove_manager':
    await self.remove_manager(json_info)

# 群主转让给他人
elif function == 'transfer_master':
    await self.transfer_master(json_info)

# 群主/管理员直接添加用户到群聊
elif function == 'add_group_member':
    await self.add_group_member(json_info)

# 群主/管理员直接移除群成员
elif function == 'remove_group_member':
    await self.remove_group_member(json_info)

# 获取用户的所有私聊/群聊信息
elif function == "fetch_room":
```

```
    await self.fetch_room(json_info)

# 获取用户作为群主/管理对应群的入群申请
elif function == "fetch_invite_list":
    await self.fetch_invite_list(json_info)

# 获取某一个私聊/群聊的具体信息
elif function == "fetch_roominfo":
    await self.fetch_roominfo(json_info)

# 修改私聊/群聊免打扰属性
elif function == "revise_is_notice":
    await self.revise_is_notice(json_info)

# 修改私聊/群聊置顶属性
elif function == "revise_is_top":
    await self.revise_is_top(json_info)

# 修改特殊私聊/群聊属性
elif function == 'revise_is_specific':
    await self.revise_is_specific(json_info)

# 设置已读消息
elif function == 'read_message':
    await self.read_message(json_info)

# 删除消息
elif function == 'delete_message':
    await self.delete_message(json_info)

# 二次检验密码
elif function == 'examine_password_twice':
    await self.examine_password_twice(json_info)

# 获取消息具体内容
elif function == "fetch_message":
    await self.fetch_message(json_info)
```

其中 json_info 存储前端传递的信息，function 字段为区别前端请求的特殊字段，后端以此作为区分。

这些函数使用 WebSocket 实现，包含聊天功能和群组管理功能，确保实时性，同时维护数据库，可以实时发送数据/从数据库中抓取数据。

视频通话模块的 Websocket 后端处理函数映射和大致功能设置如下所示：

```
# 连接 Websocket
if rtc_type == 'login':
    await self.login()

# 发送通话请求
elif rtc_type == 'call':
    await self.call_received()

# 接受通话请求
elif rtc_type == 'answer_call':
    await self.answer_call()

# 结束通话
elif rtc_type == 'stop_call':
    await self.stop_call()

# 添加通话用户
elif rtc_type == 'ICEcandidate':
    await self.ICEcandidate()
```

视频通话采用 WebRTC 协议实现，该协议建立在原有的 Websocket 协议上进行通讯。其中，rtc_type 用来区分视频通话的申请功能，用以返回不同的数据信息。

3.3 前端模块设计

前端主要采用 React 进行开发, 并使用 Ant Design 与 Material UI 组件库

前端系统被划分为用户、好友、会话三大模块

用户模块包括用户注册/登录/注销/登出/信息管理等功能

好友模块包括用户搜索/查看用户信息/添加好友/删除好友/管理好友分组等功能

会话模块包含会话管理以及消息分发的四次握手模型等功能

其具体实现可参考 API 文档

前端各数据类型接口如下:

```
interface messageListData {
    msg_id: number;
    msg_type: string;
    msg_body: string;
    msg_time: string;
    sender: string;
    read_list: boolean[];
    avatar: string;
    is_delete: boolean;
    msg_answer?: number;
    reply_id?: number;
    combine_list?: number[];
}

interface friendListData {
    groupname: string;
    username: string[];
}

interface userData {
    username: string;
}

interface receiveData {
    username: string;
    is_confirmed: boolean;
    make_sure: boolean;
}

interface roomListData {
    roomname: string;
    roomid: number;
    is_notice: boolean;
```

```
    is_top: boolean;
    is_private: boolean;
    message_list: messageListData[];
    index: number;
    is_delete: boolean;
    is_specific: boolean;
}

interface roomInfoData {
    mem_list: string[];
    manager_list: string[];
    master: string;
    mem_count: number;
    is_private: boolean;
}
```

第四部分 API 接口

4.1 API 概述

API 接口分为 HTTP 协议与 Websocket 协议两类，前者更加易用，而后者主要用于保障信息在 Web 页面上的即时渲染。

4.2 用户管理模块

1. 登录 **/user/login POST**

   用户登录（用户名与邮箱选其一填入）

键 (key)		参数类型	说明
account	username	string	用户名
	email	string	邮箱
password		string	密码

成功代码: 200

返回体:

code	0
username	username
token	token
info	Login Success

失败代码: 400

返回体:

错误原因	提示信息
用户不存在	User not exists -4
密码错误	Wrong password -2
其他错误	Unexpected error -1

2. 注册 **/user/register POST**

   用户注册

键 (key)	参数类型	说明
username	string	用户名
password	string	密码

成功代码: 200

返回体:

code	0
info	Success

失败代码: 400

返回体:

错误原因	提示信息
用户已经存在	User already exists -3
数据不合格式	Invalid userdata -2

3. 用户登出 **/user/logout DELETE**

登出账号

键 (key)	参数类型	说明
token	string	令牌
username	string	用户名

成功代码: 200

返回体:

code	0
info	Logout Suceed

失败代码: 400

返回体:

错误原因	提示信息
令牌错误（鉴权未通过）	Token error -1

4. 用户注销 **/user/cancel DELETE**

注销账号

键 (key)	参数类型	说明
username	string	用户名
password	string	密码

成功代码: 200

返回体:

code	0
info	User Canceled

失败代码: 400

返回体:

错误原因	提示信息
用户不存在	User not exists -1
密码错误	Wrong password -1

5. 发送验证邮件 **/user/send_email POST**

作为修改邮箱时的验证手段

键 (key)	参数类型	说明
email	string	新邮箱

成功代码: 200

返回体:

code	0
info	验证码已发送

失败代码: 400

返回体:

错误原因	提示信息
发送失败	发送失败 -1

6. 修改用户信息 **/user/revise PUT**

修改用户基本信息

键 (key)	参数类型	说明
revise_field	string	修改项
revise_content	string	修改值
username	string	用户名
password	string	密码
token	string	令牌

成功代码: 200

返回体:

code	0
info	Revise Succeed

失败代码: 400

返回体:

错误原因	提示信息
找不到用户	User Not exists or Login State Error -1
令牌错误（鉴权未通过）	Token error -2
传入数据类型错误	Revise_field Error -3

7. 上传头像 **/user/upload POST**

请求修改头像

键 (key)	参数类型	说明
username	string	用户名
avatar	file	头像

成功代码: 200

返回体:

code	0
info	Upload Success

失败代码: 400

返回体:

错误原因	提示信息
上传失败	Unexpected error -1

4.3 好友关系模块

1. 搜索用户 **/friend/searchuser POST**

   搜索存在的用户

键 (key)	参数类型	说明
my_username	string	己方用户名
friend_username	string	对方用户名

成功代码: 200

返回体:

code	0
info	Search Succeed
search_user_list	userdata[]

失败代码: 400

返回体:

错误原因	提示信息
预期外异常	Unexpected error -1

2. 查看用户信息 **/friend/checkuser POST**

   查看某个用户的信息

键 (key)	参数类型	说明
my_username	string	己方用户名
check_username	string	对方用户名
token	string	令牌

成功代码: 200

返回体:

code	0
info	Search Succeed
is_friend	boolean

失败代码: 400

返回体:

错误原因	提示信息
check_username 不存在	User not found -20
my_username 不存在	User not found -3
username 相同	The Query User Is The Same As User -4
令牌错误	Token Error -2
预期外错误	Unexpected error -1

3. 发送好友申请 **/wsconnect**

键 (key)	参数类型	说明
function: apply	string	ws 中区分功能的字段
from	string	发送申请方
to	string	接收申请方
username	string	用户名

4. 同意好友申请 **/wsconnect**

键 (key)	参数类型	说明
function: confirm	string	ws 中区分功能的字段
from	string	发送申请方
to	string	接收申请方
username	string	用户名

5. 拒绝好友申请 **/wsconnect**

键 (key)	参数类型	说明
function: decline	string	ws 中区分功能的字段
from	string	发送申请方
to	string	接收申请方
username	string	用户名

6. 删除好友 **/friend/deletefriend DELETE**

   删除某个好友

键 (key)	参数类型	说明
username	string	用户名
token	string	令牌
friend_name	stirng	好友用户名

7. 建立好友分组 **/friend/createfgroup POST**

   建立新好友分组

键 (key)	参数类型	说明
username	string	用户名
fgroup_name	string	分组名称
token	string	令牌

成功代码: 200

返回体:

code	0
info	CreateGroup Succeed

失败代码: 400

返回体:

错误原因	提示信息
分组已经存在	Group Already Exists -1

令牌错误	Token Error -2

8. 移动好友到指定分组 **/friend/addfgroup PUT**

键 (key)	参数类型	说明
username	string	用户名
fgroup_name	string	分组名称
token	string	令牌
friend_name	string	好友用户名

成功代码: 200

返回体:

code	0
info	AddGroup Succeed

失败代码: 400

返回体:

错误原因	提示信息
令牌错误	Token Error -2

9. 删除好友分组 **/friend/deletefgroup DELETE**

键 (key)	参数类型	说明
username	string	用户名
fgroup_name	string	分组名称
token	string	令牌

成功代码: 200

返回体:

code	0

info	Delete FriendGroup Succeed

失败代码: 400

返回体:

错误原因	提示信息
令牌错误	Token Error -2
分组不存在	Group Not Exists -1

10. 抓取申请列表 **/wsconnect**

请求体:

键 (key)	参数类型	说明
function: fetchapplylist	string	ws 中区分功能的字段
username	string	用户名

返回体:

键 (key)	参数类型	说明
function: applylist	string	ws 中区分功能的字段
applylist	applyListData[]	好友申请发送列表

11. 抓取接收列表 **/wsconnect**

请求体:

键 (key)	参数类型	说明
function: fetchreceivelist	string	ws 中区分功能的字段
username	string	用户名

返回体:

键 (key)	参数类型	说明
function: receivelist	string	ws 中区分功能的字段
receivelist	receiveListData[]	好友申请接收列表

12. 抓取好友列表 **/wsconnect**

请求体:

键 (key)	参数类型	说明
function: fetchfriendlist	string	ws 中区分功能的字段
username	string	用户名

返回体:

键 (key)	参数类型	说明
function: friendlist	string	ws 中区分功能的字段
friendlist	friendListData[]	好友申请发送列表

4.4 会话模块

1. websocket 私聊/群聊连接建立 **/wsconnect**

键 (key)	参数类型	说明
function: add_channel	string	ws 中区分功能的字段
username	string	用户名

2. websocket 心跳机制 **/wsconnect**

键 (key)	参数类型	说明
function: heartbeat	string	ws 中区分功能的字段

3. websocket 心跳确认 **/wsconnect**

键 (key)	参数类型	说明
function: heartbeatconfirm	string	ws 中区分功能的字段

4. 抓取会话列表 **/wsconnect**

请求体:

键 (key)	参数类型	说明
function: fetchRoomList	string	ws 中区分功能的字段

username	string	用户名

返回体:

键 (key)	参数类型	说明
function: fetchroom	string	ws 中区分功能的字段
roomlist	roomListData[]	会话列表

5. 抓取聊天室具体信息 **/wsconnect**

请求体:

键 (key)	参数类型	说明
function: fetch_roominfo	string	ws 中区分功能的字段
roomid	number	房间号

返回体:

键 (key)	参数类型	说明
function: fetchroominfo	string	ws 中区分功能的字段
mem_list	string[]	成员列表
manager_list	string[]	管理员列表
master	string	群主
mem_count	number	成员数量
is_private	boolean	是否为私聊
index	number	当前用户在成员列表中的下标

6. 消息推送流程 **/wsconnect**

0）消息推送模型

A -> Server -> B(s)

1）A 发送消息

键 (key)	参数类型	说明
function: send_message	string	ws 中区分功能的字段

msg_type	string	消息类型
msg_body	string	消息体
reply_id	number ?	回复消息 id (可选)

2) 后端向 A 返回 ack, A 更新 msg_id

键 (key)id	参数类型	说明
function: Ack2	string	ws 中区分功能的字段
msg_id	number	消息 id

3) 后端向 B 发送 msg, B 更新消息列表

键 (key)id	参数类型	说明
function: Msg	string	ws 中区分功能的字段
msg_id	number	消息 id
msg_type	string	消息类型
msg_body	string	消息体
reply_id	number	回复消息 id
combine_list	number[]	转发的消息 id
msg_time	string	发送时间
sender	string	发送方用户名
read_list	string[]	已读列表
avatar	string	头像
is_delete	boolean	是否被本用户删除

4) B 返回 ack

键 (key)id	参数类型	说明
function: acknowledge_message	string	ws 中区分功能的字段
is_back	boolean	是否为重新登录
room_id	number	房间 id
count	number	收到消息计数

7. 设置置顶 **/wsconnect**

键 (key)	参数类型	说明
function: revise_is_top	string	ws 中区分功能的字段
chatroom_id	number	会话 id
is_top	booleean	是否置顶

8. 设置免打扰 **/wsconnect**

键 (key)	参数类型	说明
function: revise_is_notice	string	ws 中区分功能的字段
chatroom_id	number	会话 id
is_notice	booleean	是否免打扰

9. 上传文件 **/user/uploadfile POST**

请求修改头像

键 (key)	参数类型	说明
username	string	用户名
avatar	file	头像

成功代码: 200

返回体:

code	0
info	Upload Success

失败代码: 400

返回体:

错误原因	提示信息
上传失败	Unexpected error -1

10. 视频通话 **/wsconnect**

建立 Websocket 连接

请求体:

键 (key)	参数类型	说明
type: login	string	ws 中区分功能的字段
name	string	用户名

发送通话请求

请求体:

键 (key)	参数类型	说明
type: call	string	ws 中区分功能的字段
name	string	对方用户名

返回体:

键 (key)	参数类型	说明
type: call_received	string	ws 中区分功能的字段
caller	string	请求方用户名
rtcMessage	string	WebRTC 信息

接受通话请求

请求体:

键 (key)	参数类型	说明
type: answer_call	string	ws 中区分功能的字段
name	string	对方用户名

返回体:

键 (key)	参数类型	说明
type: call_answered	string	ws 中区分功能的字段
rtcMessage	string	WebRTC 信息

结束通话

请求体:

键 (key)	参数类型	说明
type: stop_call	string	ws 中区分功能的字段
name	string	对方用户名

返回体:

键 (key)	参数类型	说明
type: call_stopped	string	ws 中区分功能的字段

添加成员

请求体:

键 (key)	参数类型	说明
type: ICEcandidate	string	ws 中区分功能的字段
user	string	添加的成员用户名

返回体:

键 (key)	参数类型	说明
type: ICEcandidate	string	ws 中区分功能的字段
rtcMessage	string	WebRTC 信息

11. 创建群聊 **/wsconnect**

请求体:

键 (key)	参数类型	说明
function: create_group	string	ws 中区分功能的字段
roomname	string	群组名称
member_list	array	群成员列表

12. 解散群聊 **/wsconnect**

请求体:

键 (key)	参数类型	说明

function: delete_chat_group	string	ws 中区分功能的字段
chatroom_id	number	会话 id

返回体:

键 (key)	参数类型	说明
function: delete_chat_group	string	ws 中区分功能的字段
message: Delete Group Success	string	删除成功信息

13. 任命管理员 **/wsconnect**

请求体:

键 (key)	参数类型	说明
function: appoint_manager	string	ws 中区分功能的字段
chatroom_id	number	会话 id
manager_name	string	任命对象

返回体:

键 (key)	参数类型	说明
function: appoint_manager	string	ws 中区分功能的字段
message	string	任命成功/对方已经是管理员

14. 移除管理员 **/wsconnect**

请求体:

键 (key)	参数类型	说明
function: remove_manager	string	ws 中区分功能的字段
chatroom_id	number	会话 id
manager_name	string	任命对象

返回体:

键 (key)	参数类型	说明

function: remove_manager	string	ws 中区分功能的字段
message	string	移除成功/对方不是管理员

15. 转让群主 **/wsconnect**

请求体:

键 (key)	参数类型	说明
function: transfer_master	string	ws 中区分功能的字段
chatroom_id	number	会话 id
new_master_name	string	转让对象

返回体:

键 (key)	参数类型	说明
function: transfer_master	string	ws 中区分功能的字段
message：Transfer Master Success	string	转让成功

16. 踢出成员 **/wsconnect**

请求体:

键 (key)	参数类型	说明
function: remove_group_member	string	ws 中区分功能的字段
chatroom_id	number	会话 id
member_name	string	踢出对象

返回体:

键 (key)	参数类型	说明
function: remove_group_member	string	ws 中区分功能的字段
message	string	踢出成功/失败

17. 退出群聊 **/wsconnect**

键 (key)	参数类型	说明
function: leave_group	string	ws 中区分功能的字段
chatroom_id	number	会话 id

18. 批准入群 **/wsconnect**

键 (key)	参数类型	说明
function: reply_add_group	string	ws 中区分功能的字段
chatroom_id	number	会话 id
message_id	number	消息 id
answer	number	是否同意

第五部分 数据库设计

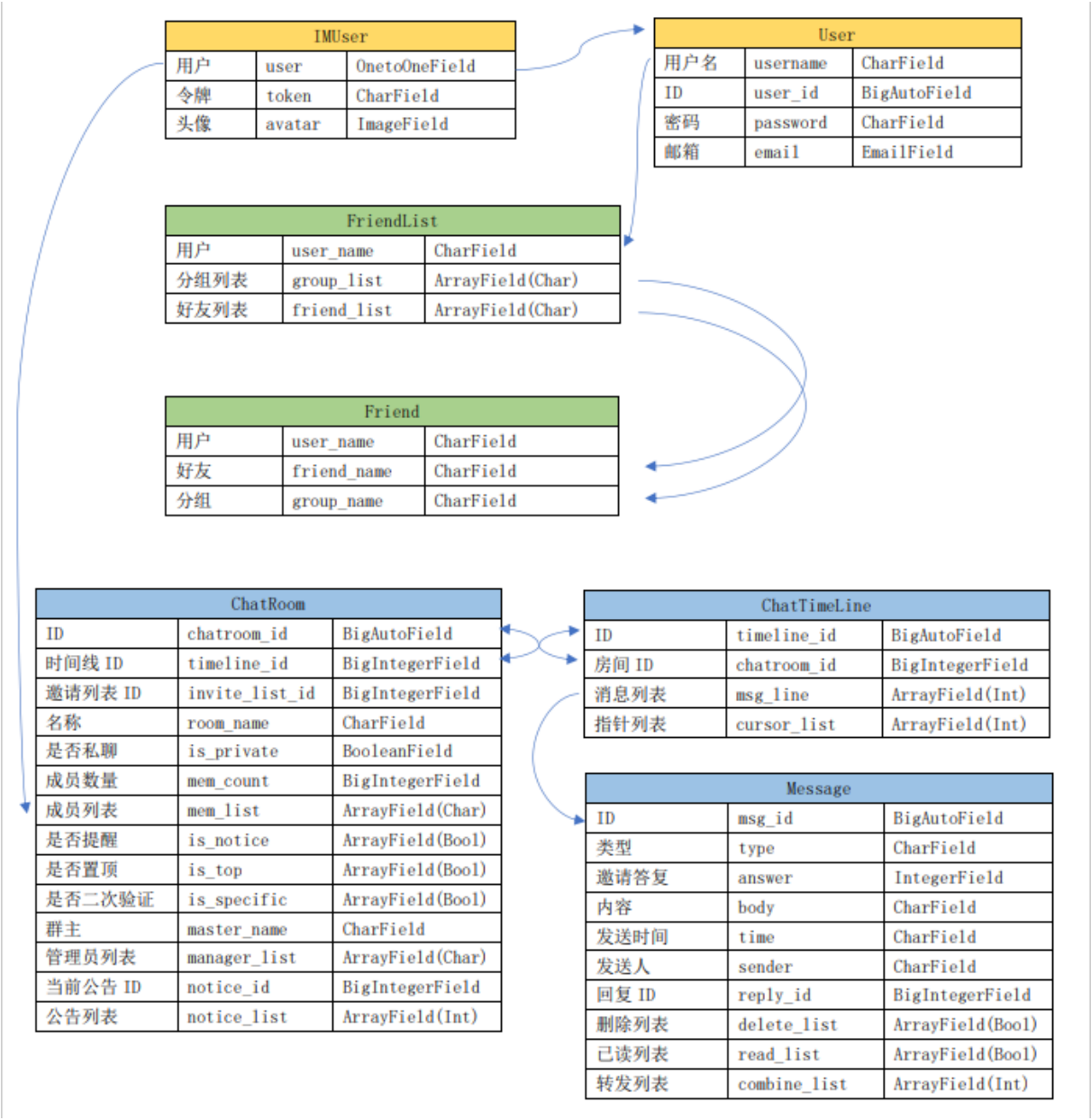
5.1 数据库概述

数据库采用与 Django 兼容的 PostGreSQL，该数据库支持使用 Arrayfield 列表类型，便于存储隶属于同一对象的大量信息。

数据库模型如下:

模型	作用
user	auth 库自带的 user 模型，用于存储用户信息
IMUser	对 user 模型的扩展，用于存储登录 token、头像
TokenPoll	用于存储所有 token
Fileload	用于存储多媒体文件
EmailCode	用于存储邮箱验证码
Friend	用于存储一对一的好友关系
FriendList	用于存储某用户的好友列表
AddList	用于存储好友申请列表
Message	用于存储单条信息
ChatTimeLine	用于以时间线形式存储信息列表
ChatRoom	用于存储会话信息，包括私聊和群聊
InviteList	用于存储邀请列表

核心模型关系如下:



数据库依功能分为三个模块：UserManage（用户），FriendRelation（好友），Chat（聊天）。其中以用户为核心，在此基础上实现好友和聊天。用户部分使用了 auth 库自带的 user 模型，并在此基础上拓展出 IMUser，用于添加令牌和头像功能；好友部分建立了一对一的好友关系为基础，并维护所在分组，进而对每个用户维护一个好友列表，同时存储了好友申请列表；聊天部分每条消息独立存储，并通过 base64 加密，在时间线中维护信息 id 列表用于找到信息，同时时间线和会话房间一一绑定，会话房间负责实现群组管理功能。

5.2 数据库部署

数据库部署在额外开启的数据库容器内，名称为 database-postgresql。该容器拉取了小组在 DockerHub 上传的数据库配置镜像，并添加了持久存储，挂载点为/var/lib/postgresql/data。同时，多媒体文件使用 Django 提供的 ImageField 字段，存储在后端容器的静态文件/collect_static/内，这些文件对于容器重启来说具有易失性。