专业实习与实训课程报告

Chat 即时聊天网站

学　　院：计算机与信息技术学院

组　　号：　Z20281106

项目简要介绍			
Chat 即时聊天网站是一个基于 vue.js 和 socket.io 的聊天网站，实现了用户注册登录，申请添加好友和删除好友，即时消息推送，自定义个人信息等功能。			
成员工作及贡献			
姓名	**学号**	**主要工作**	**贡献**
邓明	20281106	后端开发	50%
曹步云	20281104	前端开发	50%

目　　录

1 概述

Chat 即时聊天网站实现了简单的用户聊天以及即时聊天程序相关的功能需要，比如用户的注册注销，好友添加与删除，修改个人信息，即时消息推送等。

程序包括三层架构：数据库端，后端和前端，由 socket.io 在后端和前端之间进行通信，再由后端完成对数据库的操作完成业务逻辑，并将结果返回给前端进行显示。

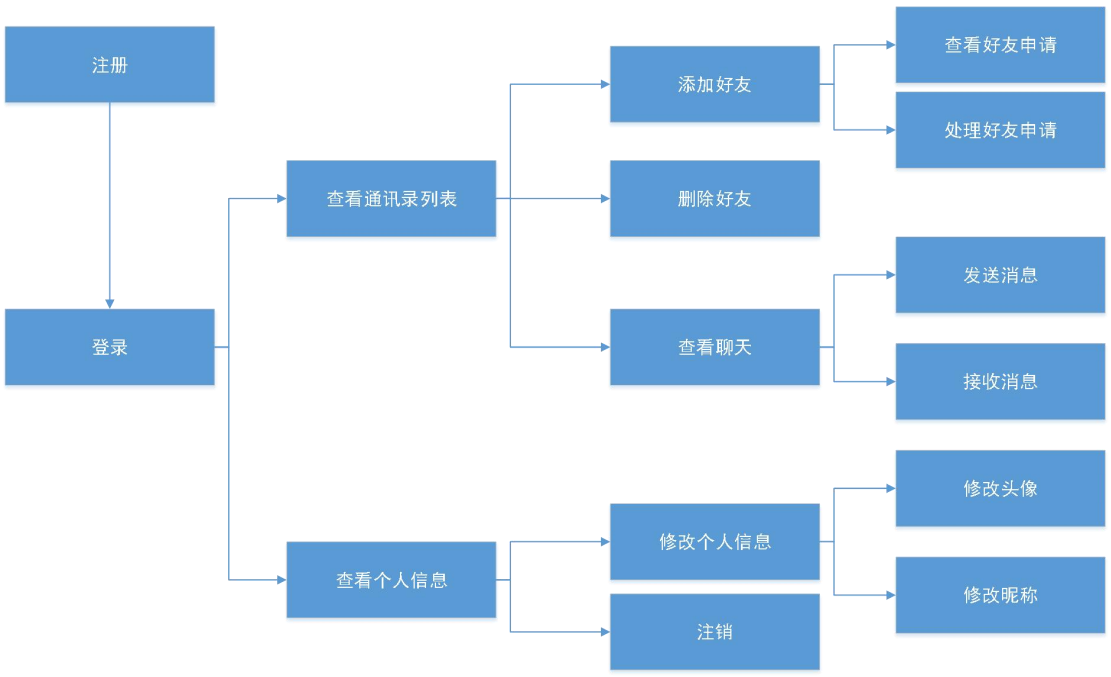
项目的数据库端使用 mysql，后端使用 nodejs+express，前端使用 vue，通过 socket.io 完成前端和后端的通信。

项目的特点有：

1. 实用性：Chat 基本可以满足用户需求，完成即时通信及相关业务。
2. 完整性：项目包括了数据库端，后端和前端三层架构，基本符合实际即时聊天程序开发结构，项目构成完整。
3. 用户友好性：Chat 聊天网站通过简洁而直观的用户界面，提供了良好的用户体验。用户可以轻松进行注册、注销、好友添加和删除等操作，并且可以自由编辑个人信息。
4. 即时消息推送：Chat 使用 socket.io 实现了即时消息的推送功能。用户可以收到实时的消息通知，不需要手动刷新页面即可看到最新的聊天信息。
5. 界面美观：Chat 即时聊天网站界面设置简洁精致，可吸引用户，且便于用户使用。
6. 可扩展性：Chat 的三层架构使项目具备良好的可扩展性。通过后端的业务逻辑处理和数据库操作的抽象，可以方便地增加新的功能模块，满足未来的扩展需求。
7. 响应式设计：前端使用 Vue 框架，采用响应式设计，使得 Chat 能够适应不同设备的屏幕尺寸，包括桌面端和移动端，用户可以在不同终端上流畅地使用。
8. 交互性：Chat 通过实时通信和即时消息推送功能，提供了良好的交互体验。用户可以与好友进行实时对话，并及时收到消息的提醒，增加了用户间的互动性。
9. 代码结构清晰：项目采用三层架构，使得代码结构清晰明了，易于维护和扩展。每一层都有明确的职责和功能划分，便于团队成员协作开发和代码模块的复用。

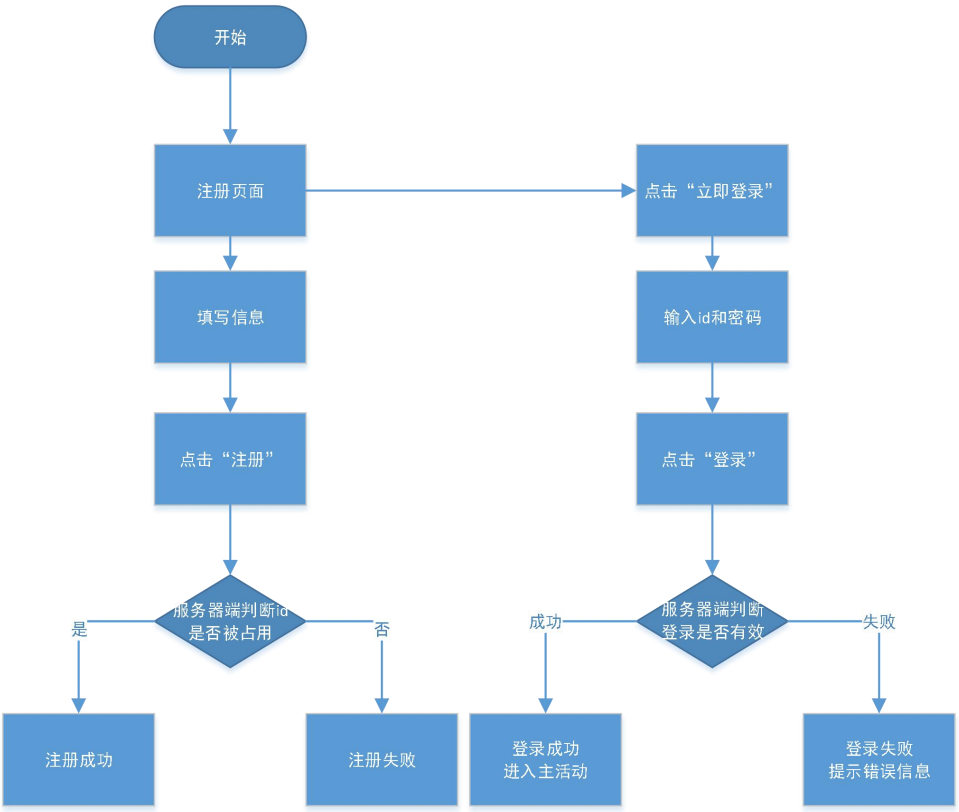
2 项目说明

2.1 功能模块

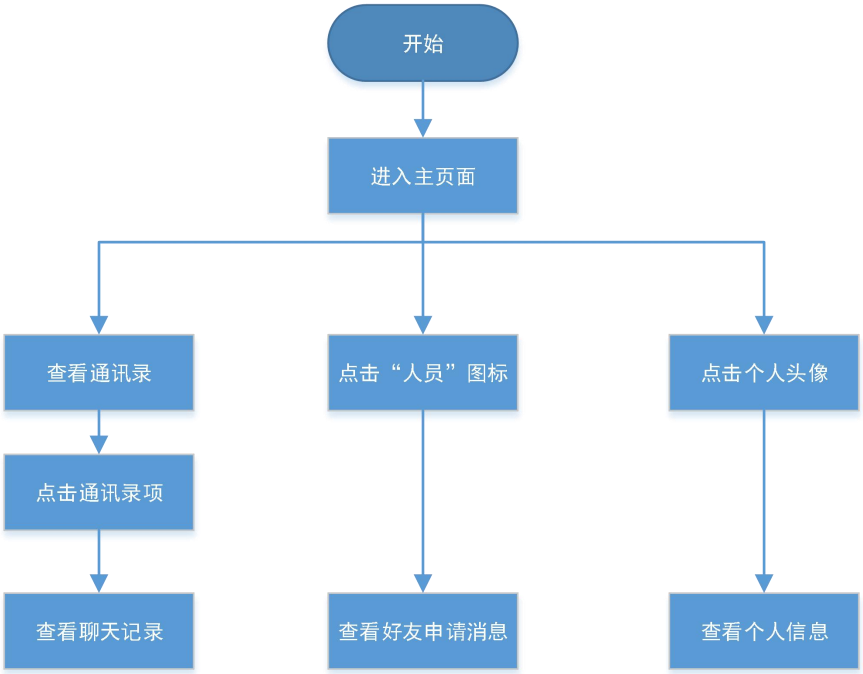


2.2 逻辑流程

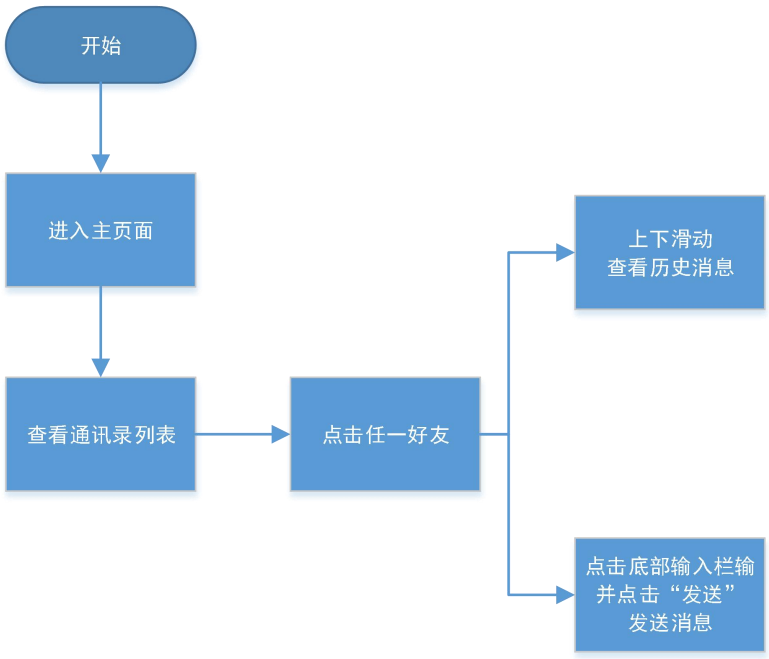
1) 登录/注册



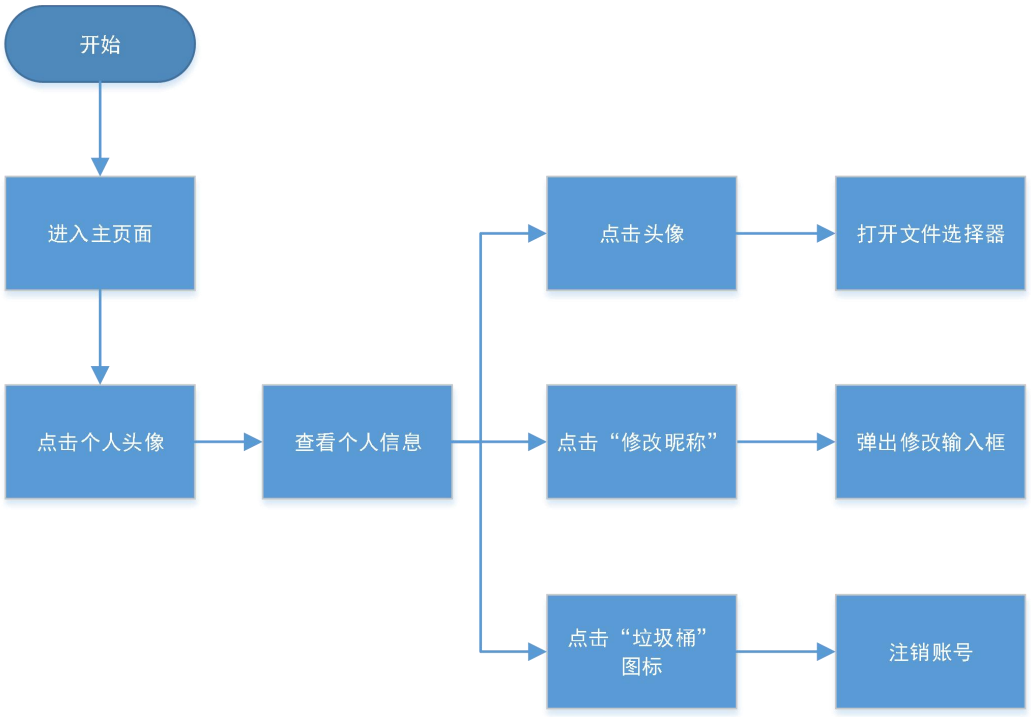
2) 查看通讯录列表/个人信息/聊天记录/好友申请消息



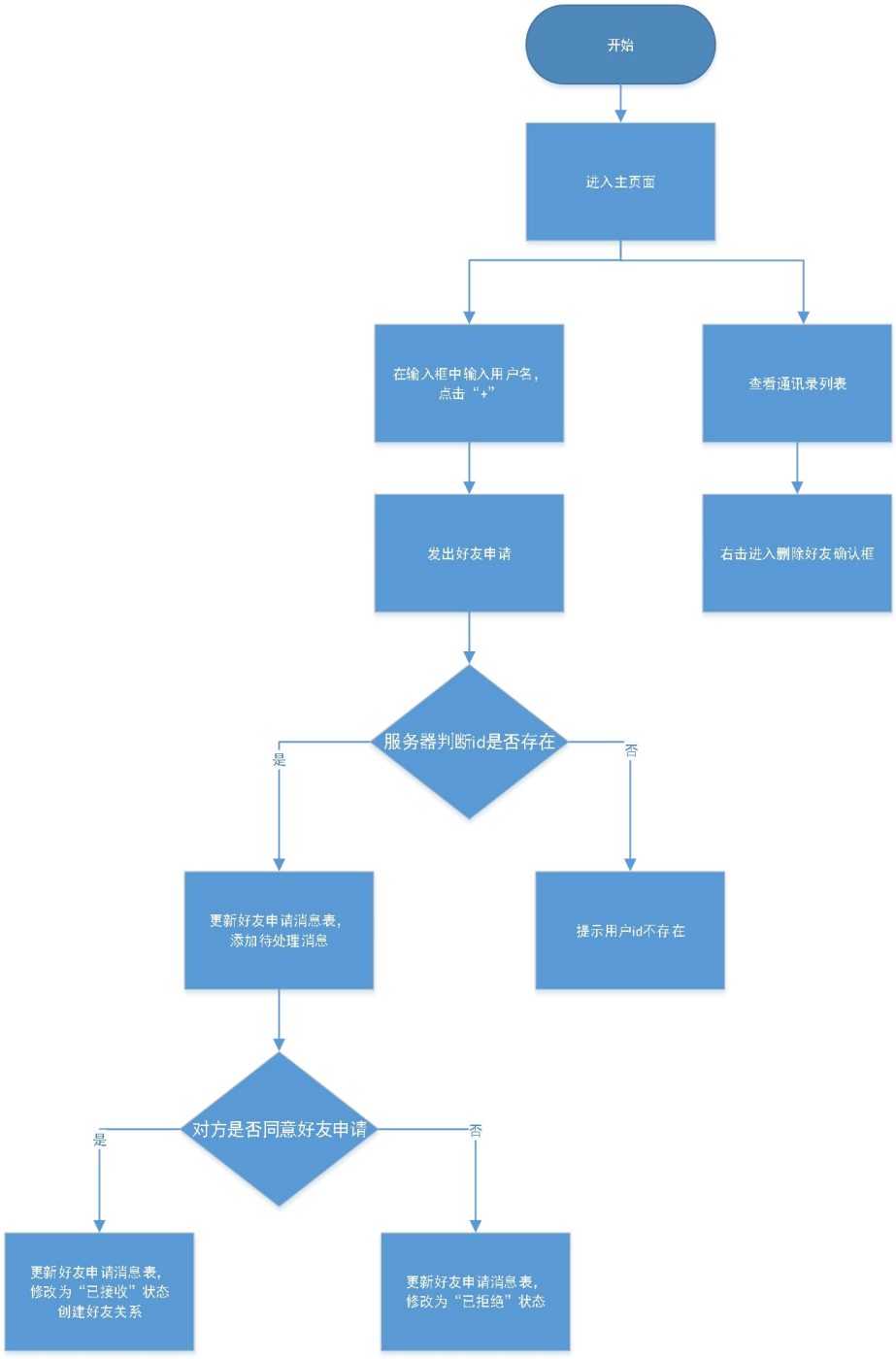
3) 聊天


开始
进入主页面
查看通讯录列表
点击任一好友
上下滑动
查看历史消息
点击底部输入栏输
并点击“发送”
发送消息

4) 个人信息管理

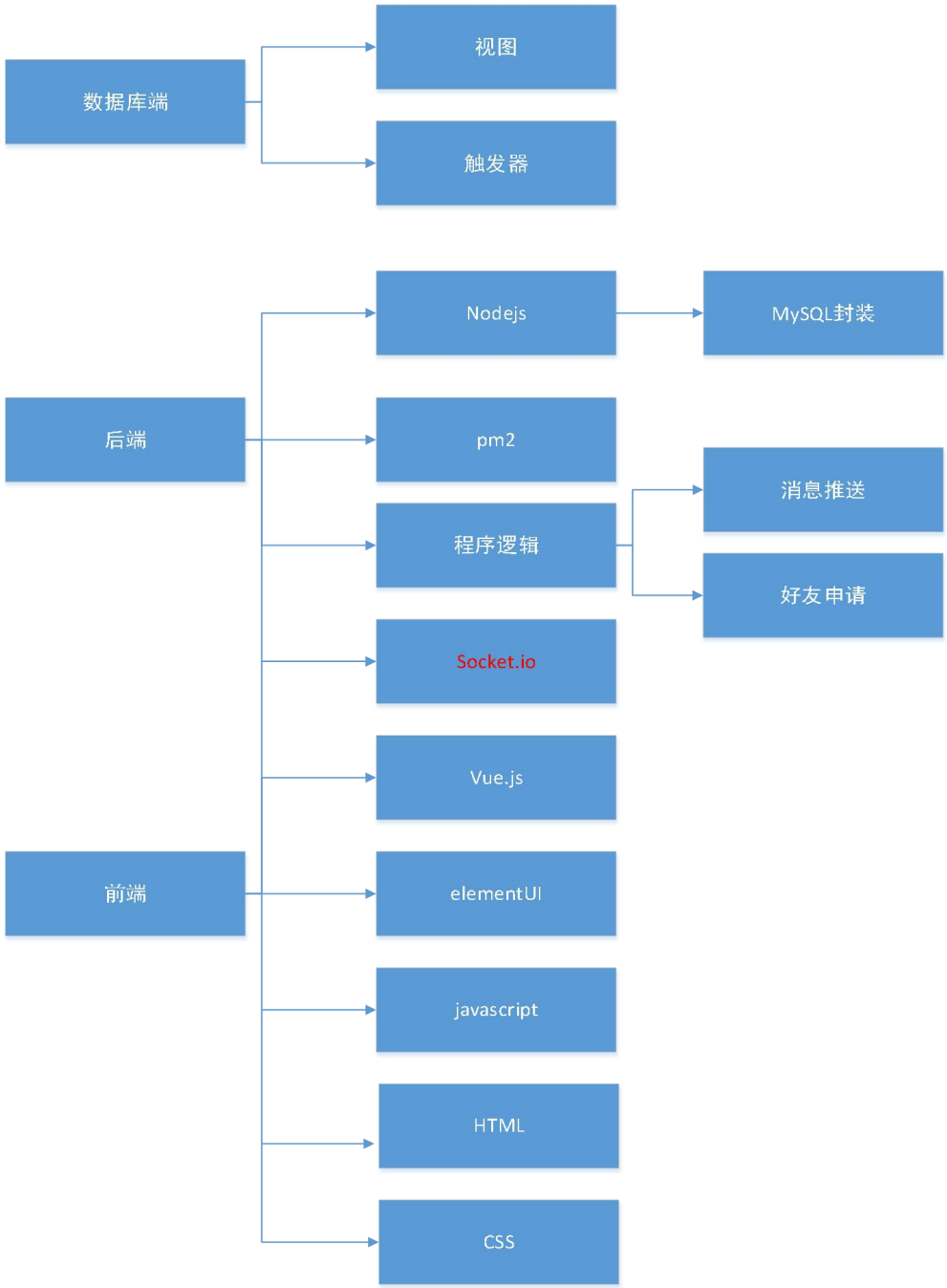

开始
进入主页面
点击个人头像
查看个人信息
点击头像
打开文件选择器
点击“修改昵称”
弹出修改输入框
点击“垃圾桶”
图标
注销账号

5) 好友添加和删除


开始
进入主页面
在输入框中输入用户名，点击“+”
查看通讯录列表
发出好友申请
右击进入删除好友确认框
服务器判断id是否存在
是
否
更新好友申请消息表，添加待处理消息
提示用户id不存在
对方是否同意好友申请
是
否
更新好友申请消息表，修改为“已接收”状态 创建好友关系
更新好友申请消息表，修改为“已拒绝”状态

2.3 技术说明

数据库端，后端，前端重点技术结构图：



2.3.1 数据库端

1) 视图

使用视图从通讯录表、私聊表和私聊信息表中提取每个聊天的最后一条消息，构成聊天列表视图，简化对聊天列表的查询过程。

```
CREATE VIEW view_pc_last_msg_info(chat_id, last_msg_id, last_msg_time, last_msg)
...
FROM private_chat_message
...

CREATE VIEW view_pc_outline(user_id, chat_with_id, remark, chat_id, last_msg_time, last_msg)
...
FROM private_chat_info, contacts, view_pc_last_msg_info
...
```

2) 触发器

在数据库端使用触发器完成添加好友的同时插入私聊信息表信息和通讯录信息。

```
...
CREATE TRIGGER create_contacts_private_chat_info
AFTER INSERT ON private_chat
FOR EACH ROW
BEGIN
    ...
    INSERT INTO contacts (user_id, friend_id, remark, chat_id) VALUES (user1_id, user2_id, NULL, chat_id);
    INSERT INTO contacts (user_id, friend_id, remark, chat_id) VALUES (user2_id, user1_id, NULL, chat_id);
    INSERT INTO private_chat_info (user_id, chat_with_id, chat_id, do_not_disturb) VALUES ( user1_id, user2_id, chat_id, FALSE);
    INSERT INTO private_chat_info (user_id, chat_with_id, chat_id, do_not_disturb) VALUES ( user2_id, user1_id, chat_id, FALSE);
...
```

2.3.2 WEB 服务器端

1） nodejs

在 WEB 服务器端使用 nodejs+express 搭建服务器，连接 mysql 数据库，并对数据库的增删改查操作进行封装，使其便于使用。

封装代码：插入操作

```
// 插入一条数据
let insertData = (table,datas,callback)=>{
    var fields='';
    var values='';
    for( var k in datas){
        fields+=k+',';
        values=values+"'"+datas[k]+"',"
    }
    fields=fields.slice(0,-1);
    values=values.slice(0,-1);
    var sql="INSERT INTO "+table+'('+fields+') VALUES('+values+')';
    // console.log(sql);
    connection.query(sql,callback);
}
```

2） pm2

在 WEB 服务器端使用 pm2 部署服务器程序，同时部署数据库备份程序，定时备份数据库，保证用户数据安全。

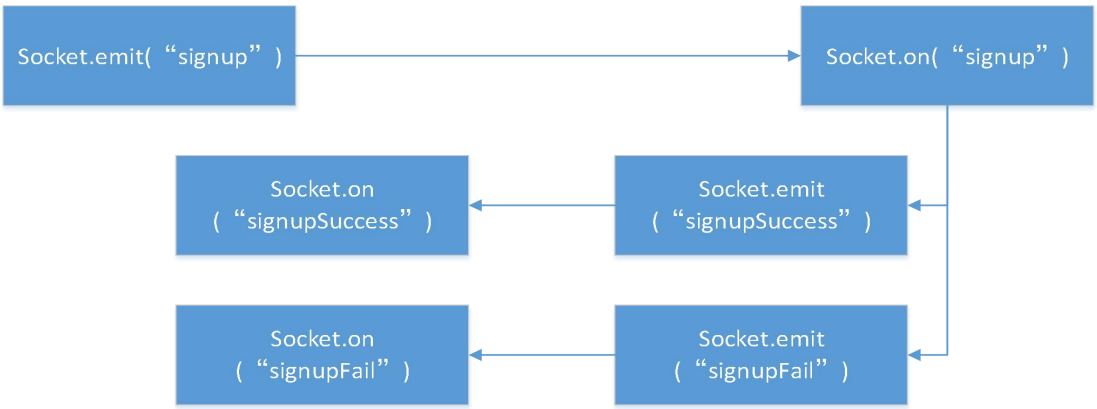
```
raki@raki-virtual-machine:~/Desktop/Chat1$ pm2 list
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	app	default	N/A	fork	43726	23s	0	online	0%	56.3mb	raki	disabled
1	backup	default	N/A	fork	0	0	15	errored	0%	0b	raki	disabled

图：使用 pm2 部署服务

3） socket.io

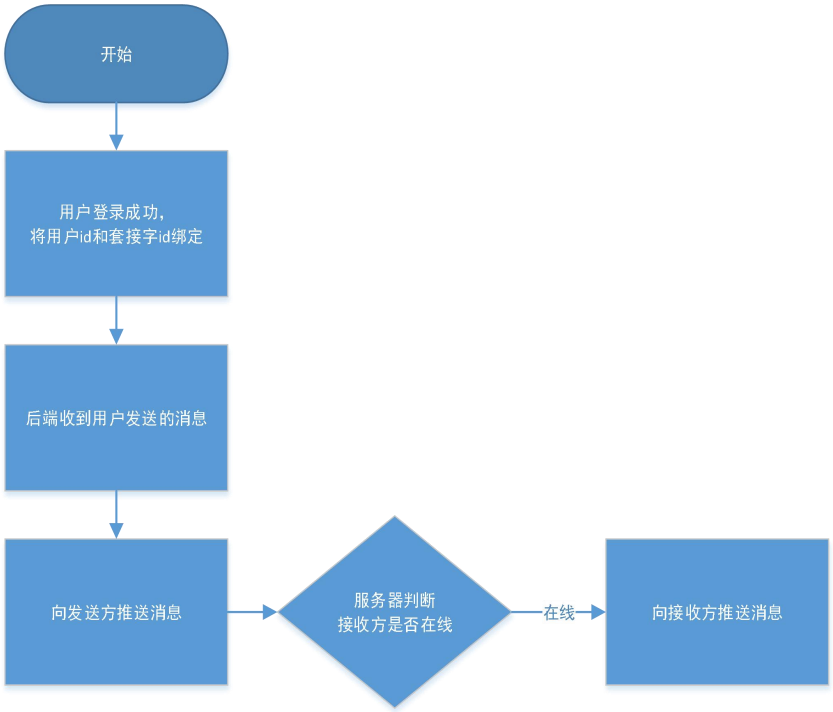
在 WEB 服务器端使用 socket.io 响应移动端发送的事件（登录，注册，发送消息，添加和删除好友等），完成对应业务逻辑。



图：登录事件交互示意

4） 消息推送

即时通信程序服务器需要在接收到用户消息时向接收方推送消息，在此过程中，后端需决定向哪个套接字发送消息。本程序在用户登录时维护用户 id 和连接 socket 的映射图，在收到消息后，检查消息接收方是否在线，如果在线，则向对应 socket 推送消息，完成消息推送。

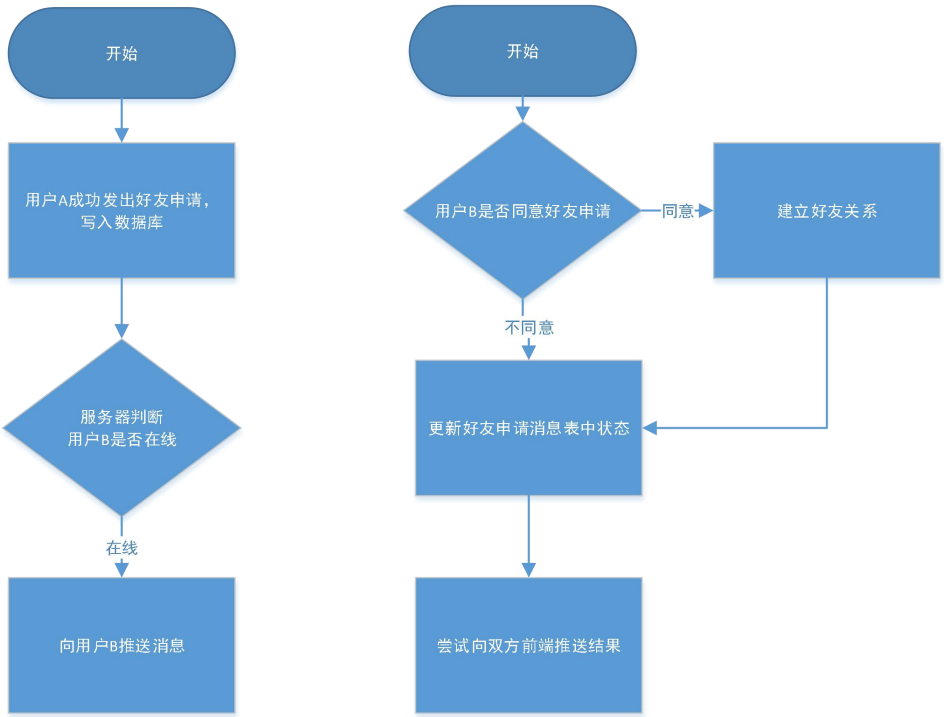


使用 socket.xxx 只能操作当前连接的 socket，使用 io.xxx 则可以操作所有的连接。

```
//登录时维护在线用户图
socket.on('login', function(data) {
                } else { //  登录验证成功
                    ...
                    onlineUsers.set(resultData.id, socket.id)
...
//接收消息时检查在线用户并完成消息推送
socket.on('sendMessage', (data)=>{
   db.insertData('private_chat_message', data, (e, r) => {
   ...
   if(io.sockets.connected[onlineUsers.get(data.receiver_id)]){
       io.sockets.connected[onlineUsers.get(data.receiver_id)].emit('receiveMessage',   [data])
   }
```

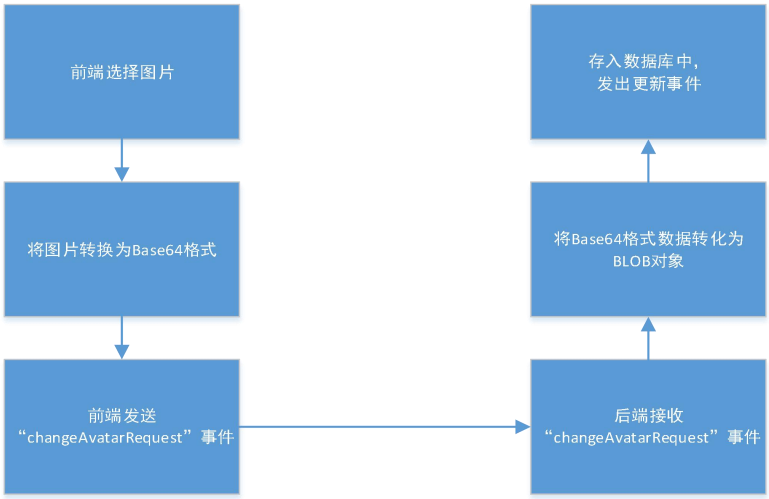
5） 好友申请

当用户 A 向用户 B 发送好友申请后，如果用户 B 存在，则将该消息写入数据库，如果用户 B 当前在线，还会向用户 B 发送提示；当用户 B 同意好友申请后，建立双方好友关系，如果用户 A 在线，将自动刷新用户 A 的通讯录；如果用户 B 拒绝，将更新数据库信息，并将新的状态尝试推送给双方前端。



6） 修改头像

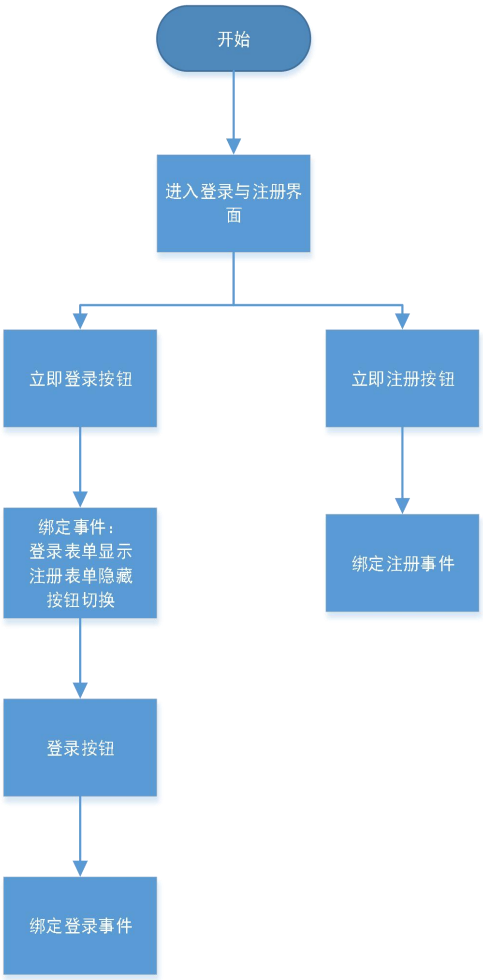
在前端打开文件选择器选择的新头像将会被转换为Base64格式通过socket.io进行传输，在后端，需要对 Base64 格式的图片进行格式转换，构建一个 BLOB 对象存入数据库中。



2.3.3 前端

1） 登录与注册界面

界面包括登录表单和注册表单两个部分，并通过切换按钮进行转换显示。

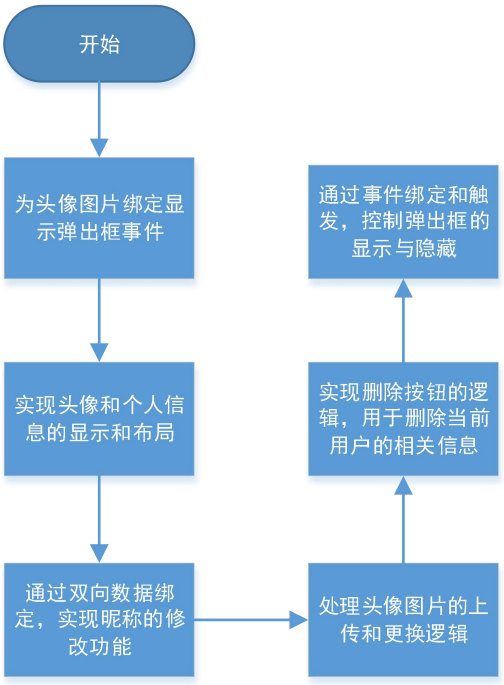


具体实现步骤如下：

a. 使用条件渲染和动态绑定类名的方式，根据变量的值决定显示登录表单还是注册表单；
b. 在登录和注册表单中，使用 v-model 实现表单数据的双向绑定；
c. 定义登录和注册的方法来处理登录注册的逻辑；
d. 使用条件渲染和动态绑定类名，根据登录状态和注册状态控制切换按钮的显示和样式。

2） 个人信息弹出框

该组件由包含头像、昵称、用户名以及其他操作按钮的弹出框构成，用户可以点击头像打开弹出框，并进行修改头像、修改昵称、注销用户等操作。

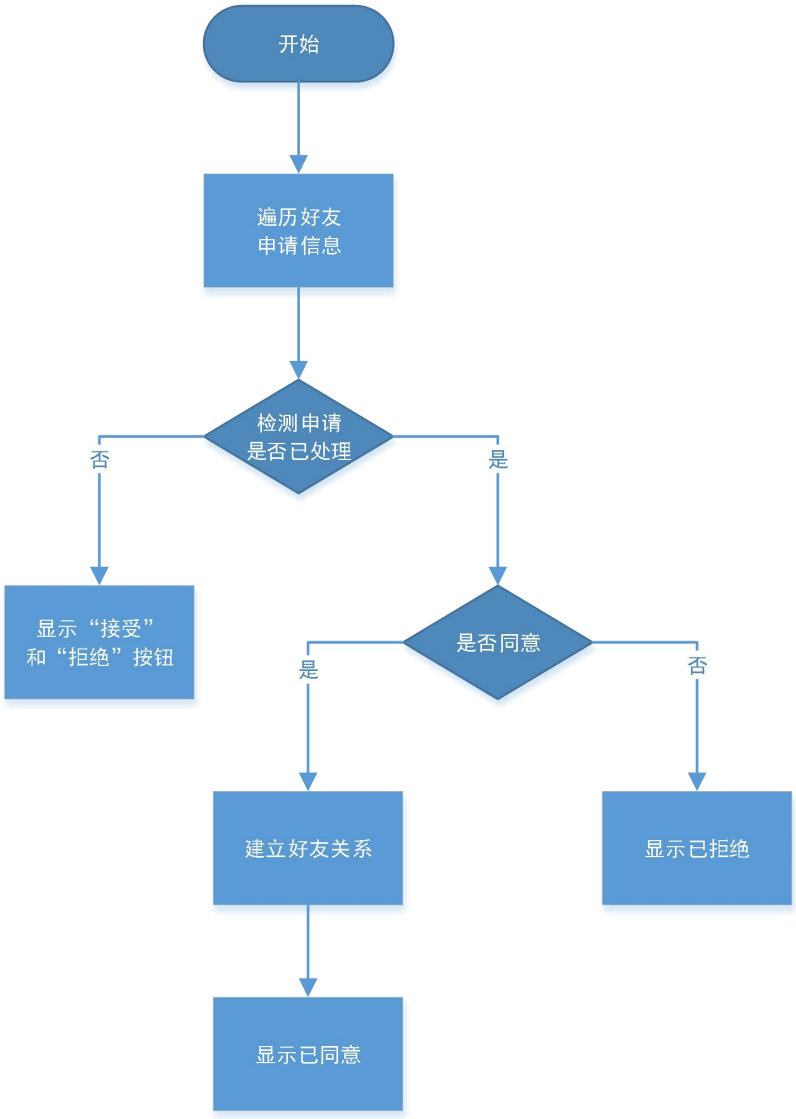


具体实现步骤如下：

a. 实现头像和个人信息的显示和布局，包括头像图片的展示、昵称的显示、用户名的显示等；
b. 使用动态绑定类名的方式，在不同的状态下控制表单的显示与隐藏；
c. 通过双向数据绑定，实现昵称的修改功能；
d. 处理头像图片的上传和更换逻辑，包括打开文件选择框、上传新头像图片等；
e. 实现删除按钮的逻辑，用于删除当前用户的相关信息；
f. 通过事件绑定和触发，控制弹出框的显示与隐藏。

3） 好友申请列表弹出框

该组件会展示用户收到的好友申请信息，并提供相应的操作按钮来处理申请。

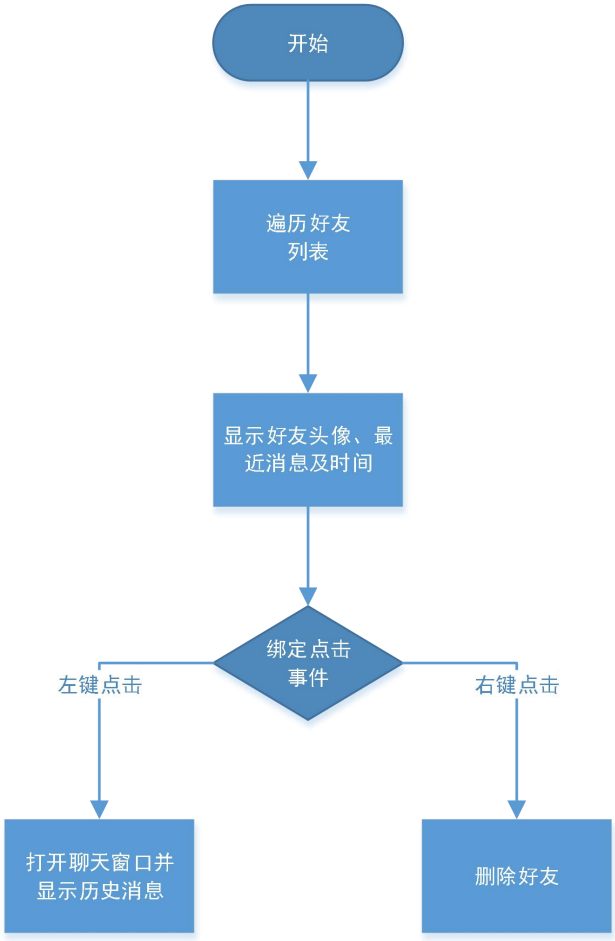


具体步骤如下：

a. 使用动态绑定类名和条件渲染，根据好友申请的状态显示不同的界面；
b. 使用 v-for 循环遍历好友申请信息数组，并展示每条申请的相关信息；
c. 根据申请状态动态显示不同的操作按钮，如果申请未处理，则显示“接受”和“拒绝”按钮；
d. 为操作按钮绑定对应的点击事件，以便处理好友申请；
e. 设置对话框的样式和交互行为，包括对话框的宽度、位置等；
f. 使用局部变量和方法来获取申请用户的信息、申请时间和申请状态，用于在界面上显示相应的信息。

4） 好友列表

该组件会展示用户的好友头像、好友名以及最近一条聊天记录的内容和时间，并添加了点击事件来处理发送消息和右键点击事件来删除好友。

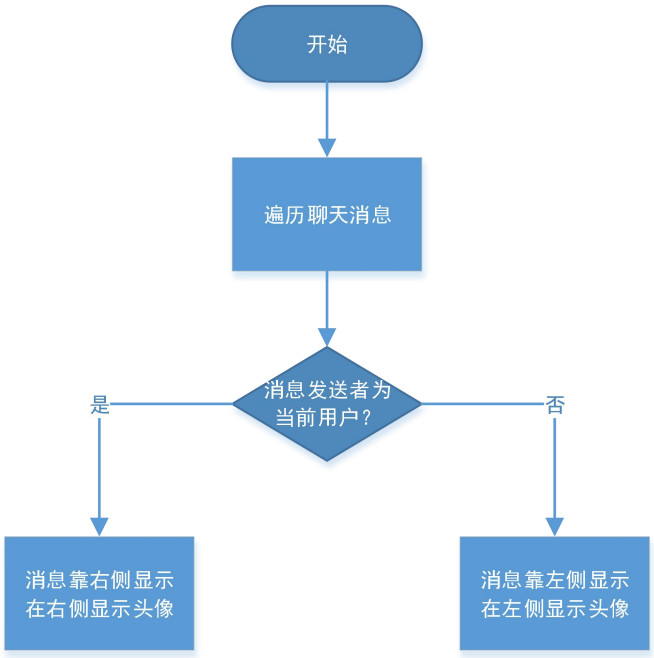


具体实现步骤如下：

a. 使用 v-for 循环遍历好友列表数据，并将每个好友的信息显示在界面上；
b. 使用动态绑定类名和条件渲染，根据界面的不同状态进行样式控制；
c. 为好友列表中的每个好友项绑定点击事件，当用户点击某个好友时触发对应的逻辑，如打开聊天窗口发送消息；
d. 为好友列表中的每个好友项绑定右键点击事件，当用户右键点击某个好友时触发删除好友的逻辑；
e. 定义方法来处理点击事件；
f. 使用局部变量和方法来获取最近一条聊天记录的内容和时间，并在界面上进行展示。

5） 聊天窗口

该组件会根据用户与好友之间的消息内容和发送者进行动态渲染，左侧显示好友的消息，右侧显示用户自己的消息。



具体实现步骤如下：

a. 使用 v-for 循环遍历聊天消息数据，并将每条消息及其发送者的信息显示在界面上；
b. 根据消息的发送者动态绑定类名，设置不同的样式，左侧为好友的消息，右侧为用户自己的消息；
c. 通过条件渲染，当收到的消息为好友所发时，在消息左侧显示好友的头像；
d. 通过条件渲染，当发送的消息为用户自己所发时，在消息右侧显示用户自己的头像；
e. 使用局部变量和方法来获取消息的发送者、消息内容，并根据发送者来设置不同的样式和展示逻辑。

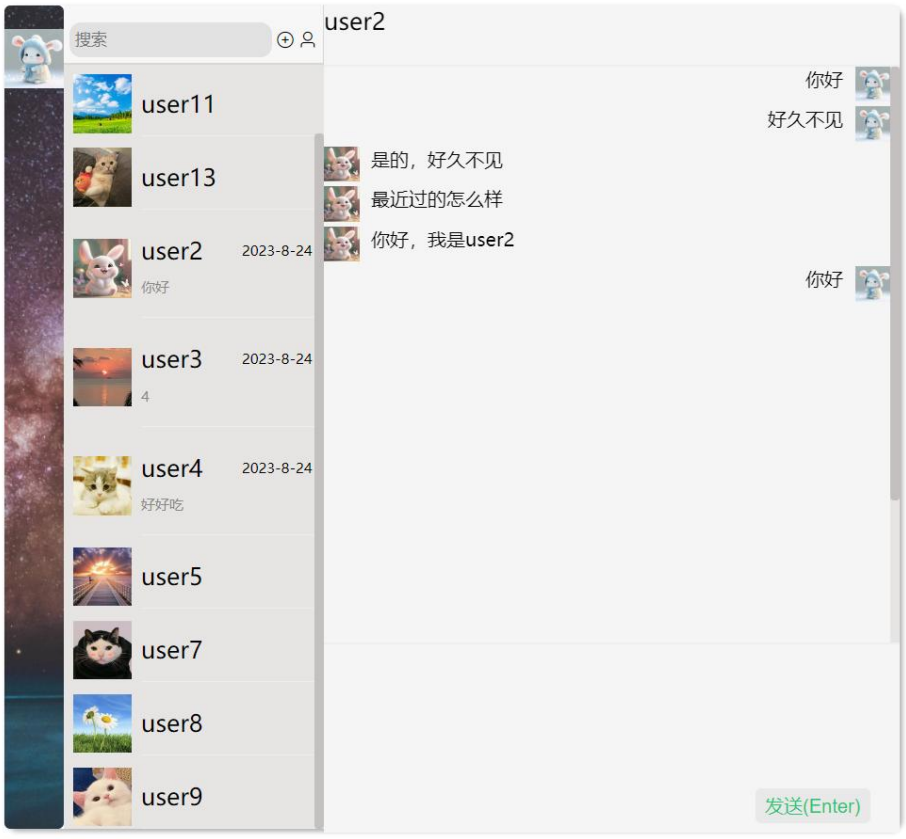
3 应用效果

1）登录与注册

图：用户登录

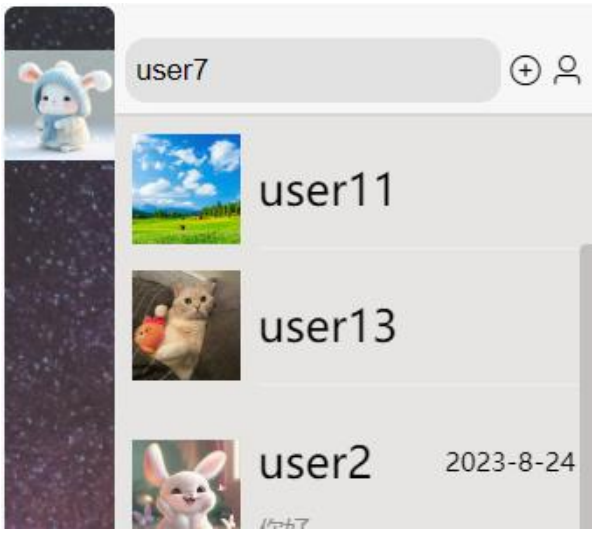
图：用户注册

2）主界面

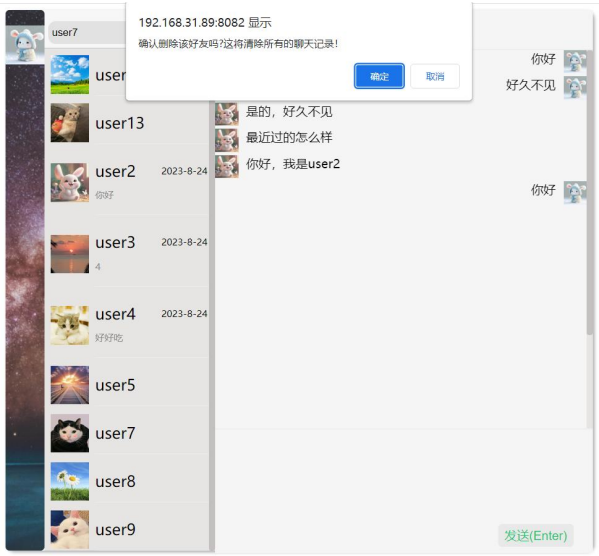


图：主界面

3）添加和删除好友

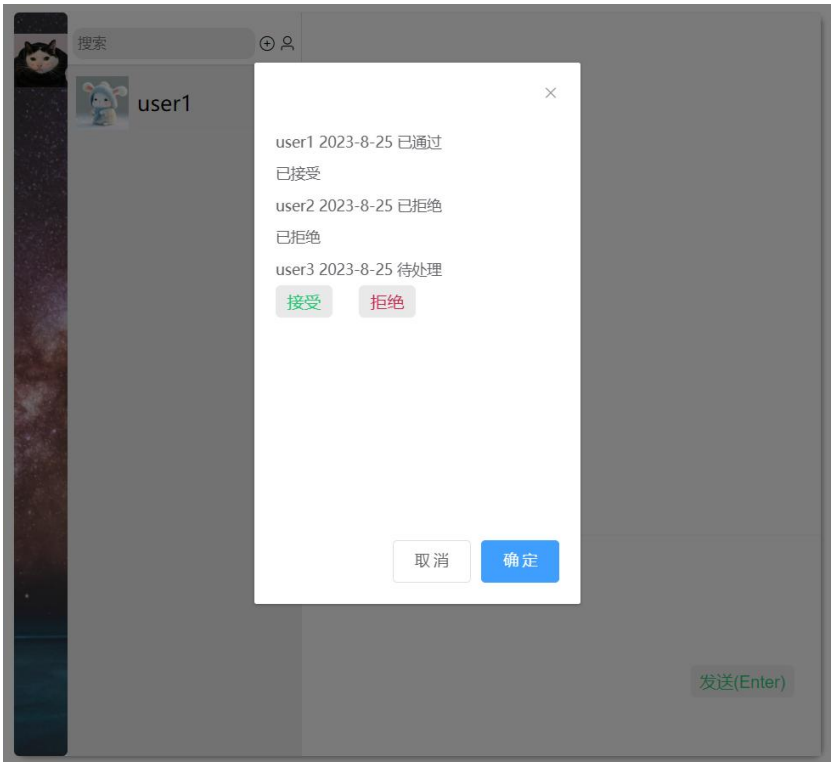


图：添加好友



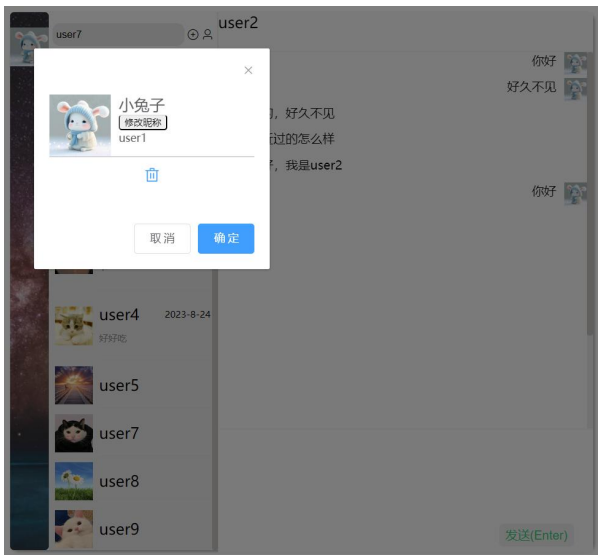
图：删除好友

4）好友申请

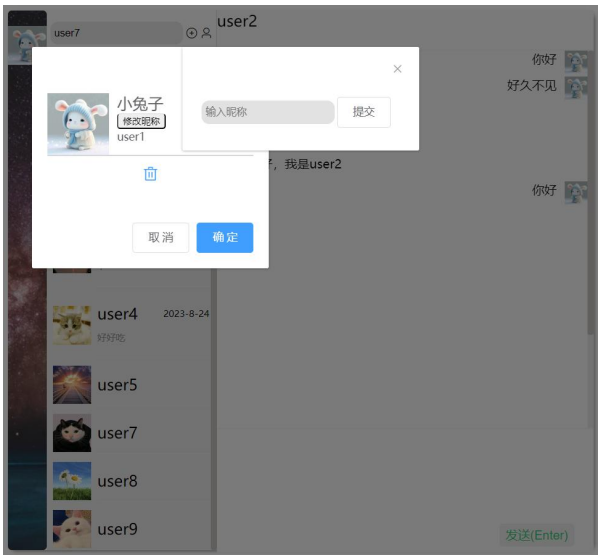


图：好友申请列表

5）修改个人信息

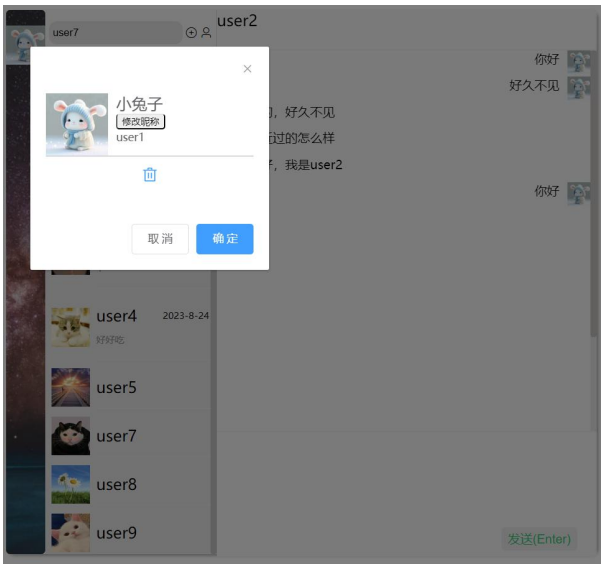


图：个人信息



图：修改昵称

6）注销帐号



图：注销帐号

4 思考及感想

完成项目之后，我们有以下思考和感想：

项目理解和设计：在开始项目之前，对聊天室的需求和功能进行深入理解，并进行合理的项目设计是非常重要的。需要考虑用户界面的友好性、实时通信的需求、数据存储和处理等方面的问题。

前端技术栈的应用：通过这个项目，我们有机会应用和巩固了各种前端技术，如HTML、CSS 和 JavaScript，以及 Vue.js 框架。我们学会了如何使用组件、路由、状态管理等前端技术来构建用户界面并与后端进行数据交互。

后端技术栈的应用：在聊天室项目中，后端起到了连接前端和数据库的关键作用。我们使用了一种或多种后端技术，如 Node.js、Express 框架和 Socket.IO 库等。通过编写后端代码，我们能够实现用户认证、消息传递以及与数据库的交互等功能。

实时通信技术的应用：聊天室项目要求实时传输消息和实时更新用户界面。我们探索了实时通信技术，例如 WebSockets，以实现即时通信功能。这让我们对实时通信的原理和实现有了更深入的了解。

数据库的应用和设计：在聊天室项目中，数据的存储和读取是至关重要的。我们使用了数据库来存储用户信息、聊天记录等数据。我们学习了如何设计和创建适合聊天室项目的数据库结构，并使用数据库操作语言（如 SQL）进行数据的增删改查。

用户体验和界面设计：一个好的聊天室项目不仅要有功能强大的前后端，还需要考虑用户体验和界面设计。我们通过设计友好的用户界面、实时消息提示、用户交互反馈等方式，提升用户在聊天室中的体验。

项目开发流程和团队协作：完成一个项目需要良好的项目管理和团队协作能力。我们学会了如何划分任务、制定开发计划、进行代码版本控制以及进行团队成员之间的沟通和协作。

错误处理和调试能力：在项目开发过程中，我们可能会遇到各种问题和错误。通过解决这些问题，我们增强了调试能力，并学会了如何处理错误和优化代码。

综上所述，完成一个聊天室项目是一次综合性的学习和实践过程。通过项目的开发，我们巩固了前端和后端技术栈的应用，学会了如何综合运用各种技术来实现一个功能完善的应用。我们也从中获得了许多经验和教训，为未来的项目开发打下了坚实的基础。