Python团队项目

团队组别：第七组

团队项目：网络聊天室

2019年11月8日

## 一、需求分析

网上聊天是从互联网建立初期就一直在发展的一项技术，现如今他的功能应用并不仅限于在线聊天了。我们可以轻而易举的列出它在网上的各种应用，如大家经常使用QQ微信等聊天工具进行的实时通信，在工作时通过github等进行的文件共享，甚至是云端储存，只要你愿意，这些云端文件也可以共享给各个用户。所以网上聊天实际就是信息的共享，储存和传输。

在本次实验中，我们将实现更为具体的功能，即网络聊天室。但是其功能不应该仅仅停止在网络聊天，我们想让他更具有拓展性，能够轻易的改造成其他应用。或者说，网络聊天室本来就应该是各个模块的整合，而这些模块也应该能在其他应用中被利用起来

本次开发我们设计的网络聊天室定位为私人的，小规模应用的聊天工具。这种工具的特点是，功能设计上有针对性，面向特定用户群体，群体规模可预测，安全性能要求较低。外界应用的需求主要分为两个方面，用户端和聊天室界面，与此同时我们也需要搭建一个服务端来相应用户端需求，以及与聊天室界面进行实时交换消息

## 二、设计要求

基本设计要求分为三个主要方面：用户端，聊天室，和服务端。其中用户端和聊天室在设计上相互独立又相互联系，二者均可与服务端进行双向信息交流，有权修改服务端中的信息。

### 1、用户端

设计主要内容：

用户可以按照一定的格式设计用户名和密码，并与服务器通信，将用户相关信息储存在服务器中，之后可以通过所设定的用户名和密码登陆聊天室。

基本需求与流程：

1） 用户注册，用户设置用户名以及密码和密码确认，服务端处理并将用户信息储存在服务器中，服务器返回一个应答供用户端判断是否注册成功，若注册成功则提示“注册成功，请登录聊天室”，系统自动返回登录界面，用户重新输入用户名与密码登录聊天室，若输入的用户名已经存在或者注册密码与确认密码不匹配则提示注册失败原因，要求用户重新注册。

2） 用户登录，服务端将用户输入的信息，包括用户名以及用户密码与服务器中的信息进行比对，返回一个应答供客户端判断是否输入正确信息，若登录成功则进入聊天室界面并且收到已成功登录的提示，若登录失败则提示登录失败的原因，系统返回登录界面让用户重新登录。

3） 成功登陆的时候在用户端中打开聊天室主界面。

4） 用户可随时退出聊天室，当用户退出聊天室时，系统自动弹出窗口提示成功退出，并提示是否重新登录。

### 2、聊天室

设计主要内容：

用户通过系统验证后进入公共聊天室的在线用户聊天界面，用户可以对当前的聊天系统发送消息，同时在线的用户可以即时收到消息。

基本需求与描述：

1） 用户端在验证成功后与服务器建立并保持连接状态以便实时接收服务端讯息。

2）用户成功登录后，聊天室界面向登录成功者发送欢迎加入聊天室消息，向已

经在聊天室成员发送新用户加入聊天室消息。

3）当用户输入信息时， 判别用户输入的信息是否符合规范，是否存在无法传输的字符等情况，并进行相应的提示。

4） 在发送信息时，用户通过用户端输入的信息将被传输给服务端，服务端收到后向所有用户端广播信息，并保存信息记录。

5） 其他用户端接收到服务器的广播信息并显示。

6）聊天界面：背景界面为相关的符合聊天室主题的图片，可设为动态变化，追求较高的视觉体验；聊天室上方显示聊天室名称，系统当前时间，在线人数等聊天室基本状况；主界面显示各个用户发送的消息，包括用户头像，用户姓名，用户群头衔，用户发送消息的时间，用户发送的消息，支持表情包的发送，界面上滑查看历史消息；界面下方显示用户输入消息框，消息框中加入提示词，同时有发送按钮，用户输入消息点击发送则成功发送消息；界面右侧显示聊天室成员的相关信息，包括头像以及姓名。

### 3、服务端

设计主要内容：

建立用户与用户之间的连接；储存新用户信息与管理老用户登陆状态；作为用户之间通信的中介；暂存用户通信记录（聊天记录）；在收到信息后返回应答，协助用户判断服务器此时状态。

基本需求与描述：

1）服务器的基本应答需求已经包含在了用户端和聊天室需求中。

2）服务端应该有自己的数据库或者数据储存方式

3）提供用户端所需的视图函数，便于用户端对页面进行实时渲染

## 三、设计工具

### 1、Flask框架

Flask是一个使用Python编写的轻量级的Web应用框架，它是依赖jinja2模板和Werkzeug WSGI服务的一个微型框架。Flask使用BSD授权。Flask被称为“microframework”，是一个典型的”微”框架。首先它使用简单的核心，默认情况下，Flask 不包含数据库抽象层、表单验证，但是这意味着诸如使用何种数据库一类的抉择可以由你自己决定。当然，Flask保留了扩增的弹性，可以用Flask-extension加入这些功能：ORM、表单验证工具、档案上传、各种开放式身份验证技术。

### Flask\_Login

Flask\_Login模块主要用来进行会话管理，处理我们通常所说的登入登出功能。其主要功能如下：

1.在会话中存储当前活跃的用户 ID，让你能够自由地登入和登出。

2.让你限制登入(或者登出)用户可以访问的视图。

3.处理让人棘手的 “记住我” 功能。

4.帮助你保护用户会话免遭 cookie 被盗的牵连。

5.可以与以后可能使用的 Flask-Principal 或其它认证扩展集成。

同时它的使用也有一定的优点和限制，如它不会：

1.限制你使用特定的数据库或其它存储方法。如何加载用户完全由你决定。

2.限制你使用用户名和密码，OpenIDs，或者其它的认证方法。

3.处理超越 “登入或者登出” 之外的权限。

4.处理用户注册或者账号恢复。

### Flask\_socketio

Flask\_socketio模块主要用于处理客户端和服务端之间双向通信的问题，客户端可以使用Socketio库或任何支持与服务端建立长链接的兼容库。

该软件包所依赖的异步服务可以从以下三种选择中选择：

·eventlet是最好的高性能选项，支持长轮询和WebSocket传输。  
·gevent支持多种不同的配置。 long-polling传输完全由gevent包支持，但与 eventlet不同，gevent没有原生的WebSocket支持。为了添加对WebSocket的支持，目前有两种选择。

·基于Werkzeug的Flask开发服务器，但缺少其他两个选项的性能，因此只能用于简化开发流程。该选项仅支持长轮询传输。

本次实验中由于我们使用WebSocket传输，采用高性能的eventlet。

### 2、WebSocket

WebSocket 是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通讯的协议，websocket为一次HTTP握手后，后续通讯为tcp协议的通讯方式。特点为在一次握手之后，浏览器和服务器之间就能创建持久的连接，而不需要被轮询浪费大量的资源。但是也由于他旨在打开连接，尽可能快地处理请求，然后关闭连接，所以对于服务器而言，任何实际的 WebSockets 服务器端实现都需要一个异步服务器。而对于客户端而言在，则没有必要为 WebSockets 使用 JavaScript 库，因为实现 WebSockets 的 Web 浏览器将通过 WebSockets 对象公开所有必需的客户端功能

WebSocket 使用一种被称作“Upgrade handshake（升级握手）”的机制将标准的 HTTP 或HTTPS 协议转为 WebSocket。这种升级的时间是可以选择的，具体而言在我们的应用中，仅当 URL 请求以“/ws”结束时，我们才升级协议为WebSocket。否则，服务器将使用基本的 HTTP/S。一旦连接升级，之后的数据传输都将使用 WebSocket。

当然，和HTTP一样，websocket也有一些约定的通讯方式，http通讯方式为http开头的方式,e.g. http://xxx.com/path ,websocket通讯方式则为ws开头的方式,e.g. ws://xxx.com/path

## 四、设计流程

在实际设计中，我们将聊天室主界面和用户端的需求统一为前端需求进行设计，对于服务器的需求则被划到后端需求，在前端后端之间，对信息交互的格式与储存格式（表单包含的信息格式）也进行设计，用于正确和规范的信息传输。

### 1、前端方面：

为了简化工作量，我们直接采用了Boostrap模版来编写HTML界面。在所有界面的底部，我们加入了包含程序的版权标识和编写者、源码等相关信息。  
我们的前端模板由  
1） 聊天主页面  
2） 登录页面  
3） 注册页面  
4） 基础页面  
几大模块构成。  
 模板template文件夹中包含了以上这几个基本的html。这是我们的前端代码文件的骨架。由于前端采用的是Boostrap，而不是Vue，React等动态前端框架，代码的编写的难度在于框架使用和资源的整合与划分。  
 我们将装饰使用到的静态文件放入到static中，并将我们具体的编写代码放入了chat中，其他的按照资源类型做了相应的文件目录划分，是项目结构清晰具体，降低前端代码的耦合性，提高前端代码的鲁棒性和可维护性。  
 前端使用模板的，在整个项目MVC的架构中，提供了VIEW 视图的功能，是整个项目的功能的展示区，但我们并没有因为前端使用的是静态模板就放弃代码的可维护度，相反是做了以上更加清晰的划分，方便我们复用，测试，修改。

### 2、信息交互方面：

### 1）登陆/注册表单

登陆以及注册表单，这个表单包含用户名和密码两个字段，其中用户名字段和密码字段是普通的文本字段，为了获得良好的样式效果，对这两个字段的输入值进行长度上的限制，用户名最长为20个字符，而密码字段也设置为最长为32个字符。

用户提交发布表单之后：

1）如果验证出错，错误消息以flash消息的形式显示在页面上方

2）如果通过验证，则在程序标题小猫显示一个提示消息，用户可以通过提示消息判断自己消息是否通过认证

### 2）消息表单

消息表单，这个表单包含用户名、头像、时间和消息内容四个字段，其中用户名字段是普通的文本字段，头像则是通过用户名字段从网络获取头像网站获取的字段，时间和消息内容字段也是普通的文本字段

### 3、后端方面

后端程序开发主要包含了几个方面：数据库建模、创建表单类、编写视图函数以及编写模版几部分。

### 1）数据库建模

我们在编写完功能规格书之后，我们对于各种数据的长度范围有了基本的了解，可以确定用哪些表来储存数据，表中有哪些字段，以及各个表之间的相互联系。对于复杂的数据库结果，我们可以使用建模工具来辅助建立数据关系。

### 2）创建表单类

表单包括了用户名密码表单和消息接收发送表单，我们使用了文本区域字段TextAreaField，表示HTML中的标签。

### 3) 编写视图函数

错误处理的函数比较简单，我们重点介绍一下index视图。index视图有两个作用：

1）处理GET请求，从数据库中查询所有的消息记录，返回渲染后的包括消息列表的主页模版index.html。

2）处理POST请求，问候表单提交之后，验证表单数据，通过验证后将数据保存到数据库中，使用flash()函数显示一条提示，然后重定向到index视图，渲染页面。

在获取消息记录是，我们可以使用order\_by()过滤器对数据库记录进行过滤和排序，参数是排序的规则。我们根据Message模型的timestamp字段值排序，字段上附加的排序方法为desc()，代表降序（descending），同样还有一个asc()方法表示升序（ascending）。

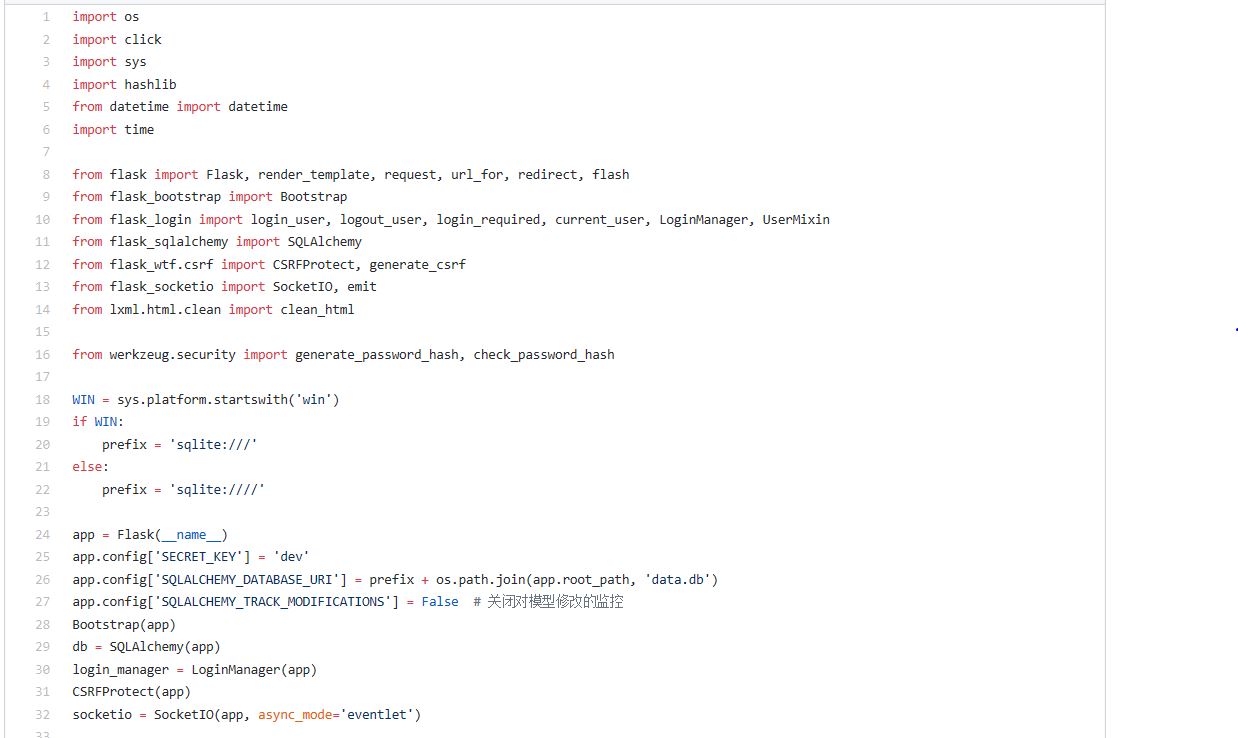
### 4) 编写模版

我们将index.html和404.html，以及500.html中的共有部分抽出合并为基模版base.html。基模版包含一个完整的HTML结构，我们在其中创建了几个块：title、content和footer。

我们可以在head标签和body标签内引入Boostrap所需的CSS和JavaScript文件，以及Boostrap所依赖的jQuery和Popper.js。另外，我们还可以引入我们自己所编写的CSS样式定义和JavaScript代码。利用模版，我们还可以直接编写出我们的login.html登录界面和注册界面。

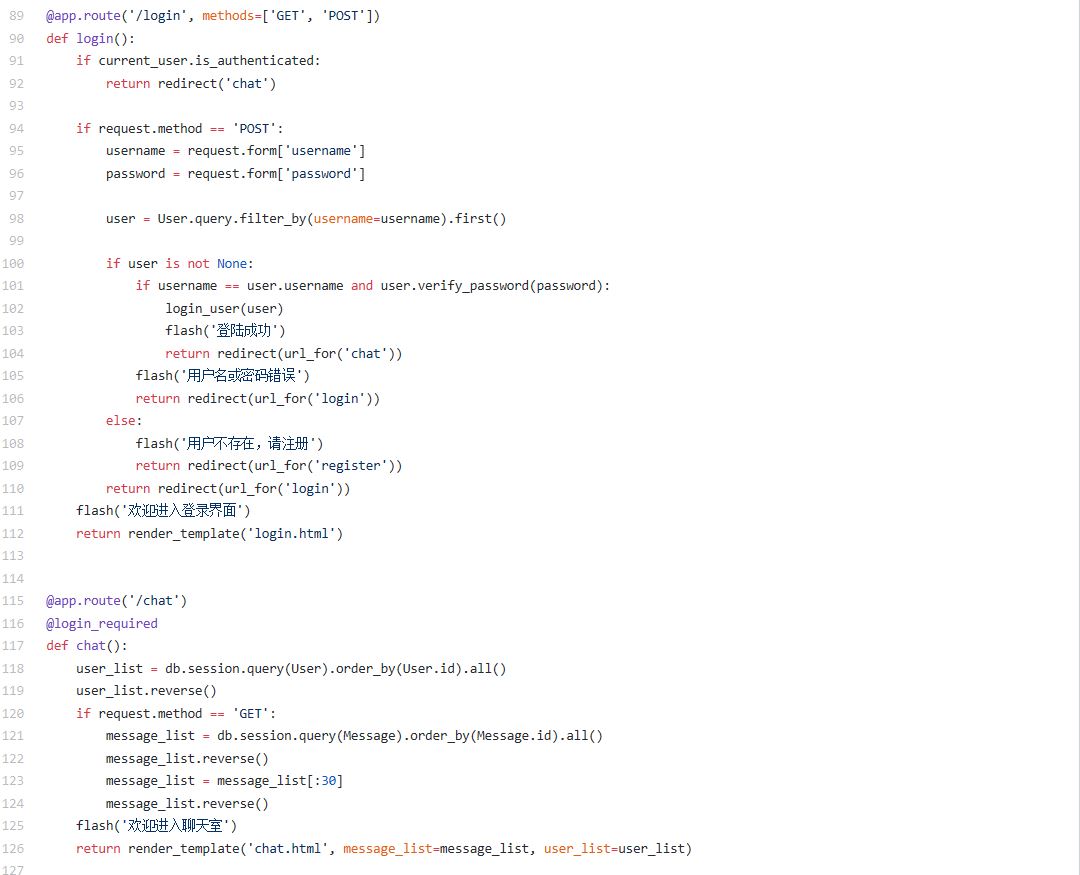
## 五、设计程序

后端代码









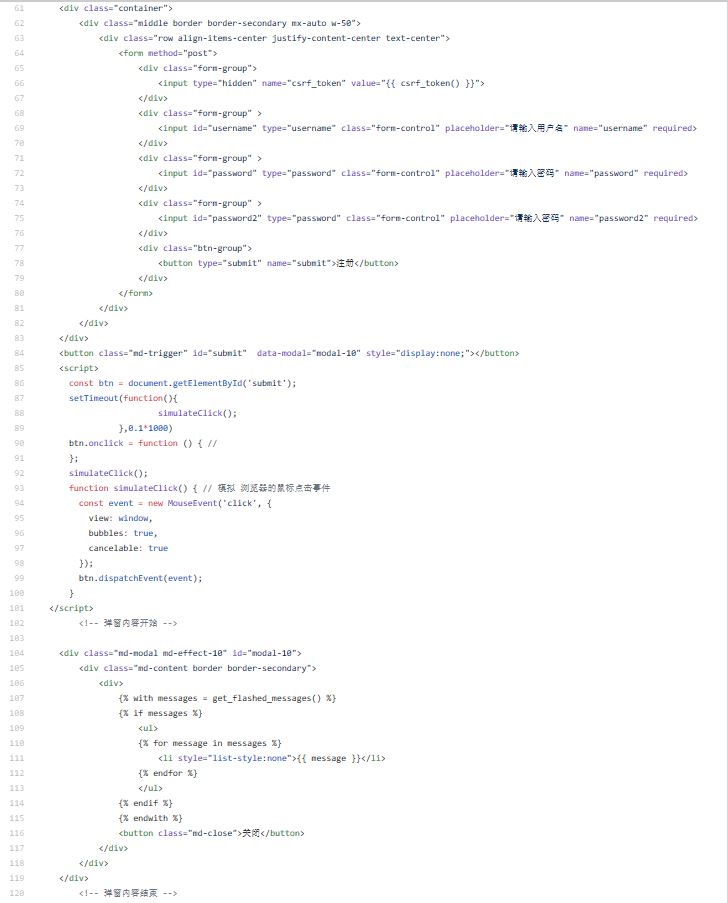




前端界面代码

1. 注册界面 register.html







1. 登录界面 login.html





1. 聊天室界面 chat.html









1. 基础界面 base.html

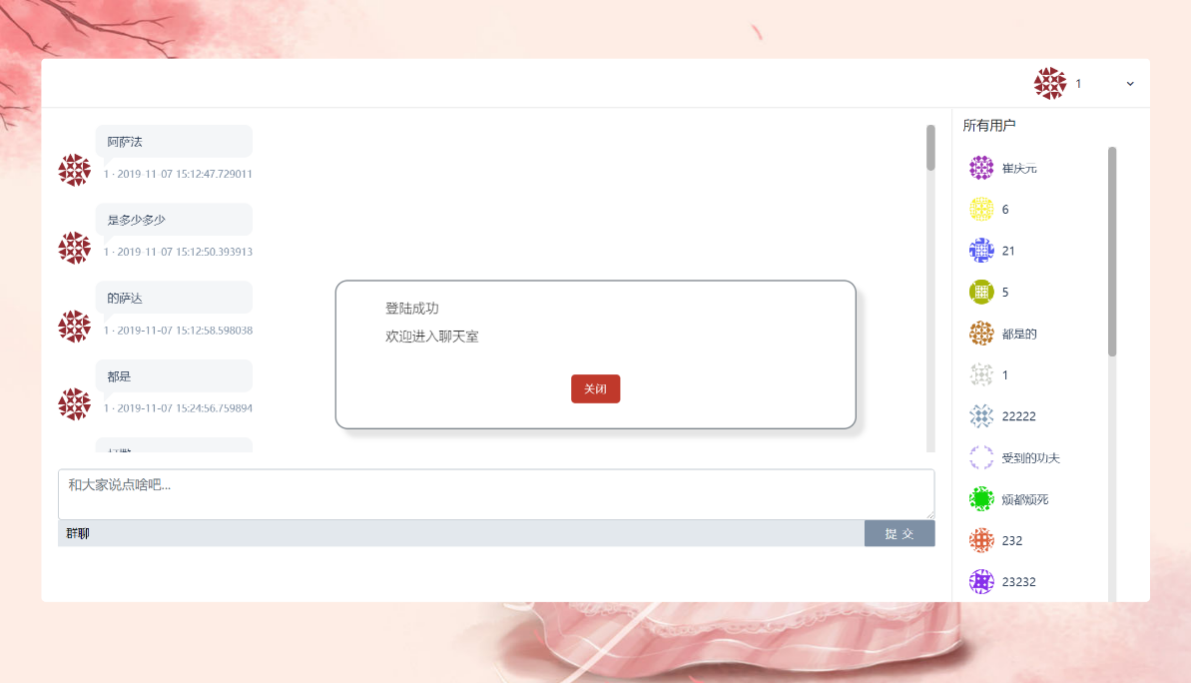


1. 消息界面 message.html

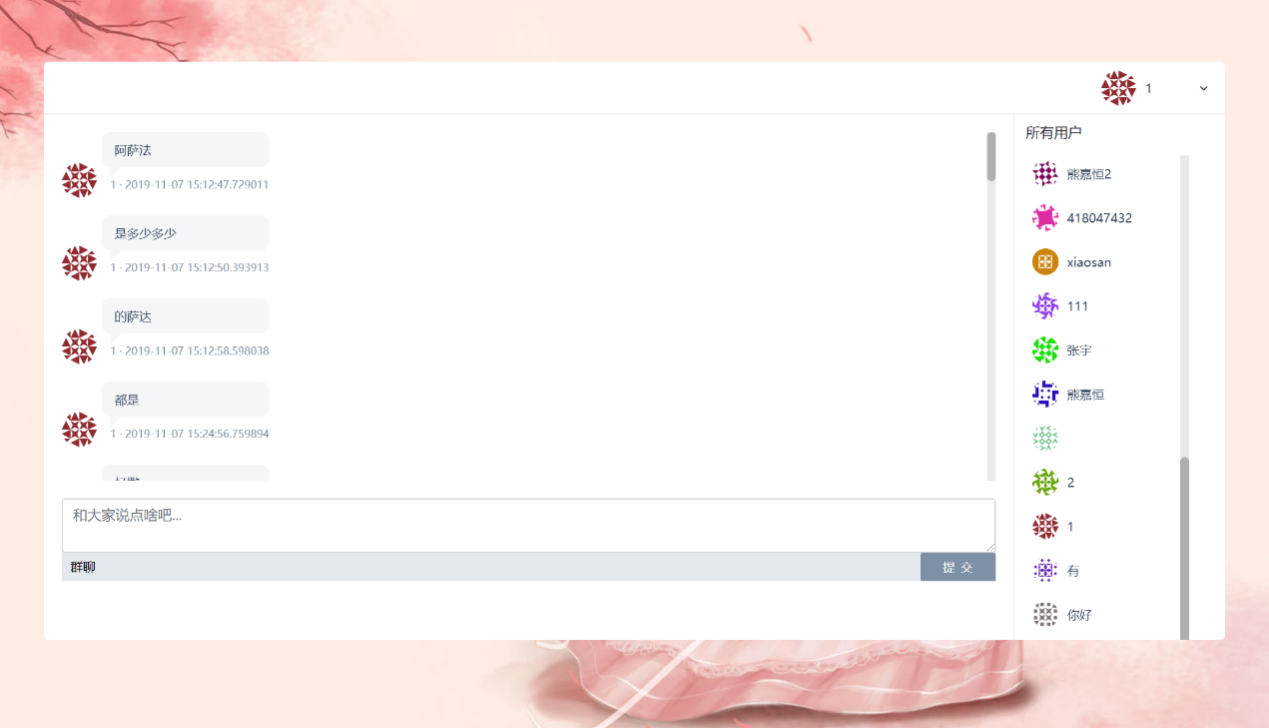


## 六、运行结果

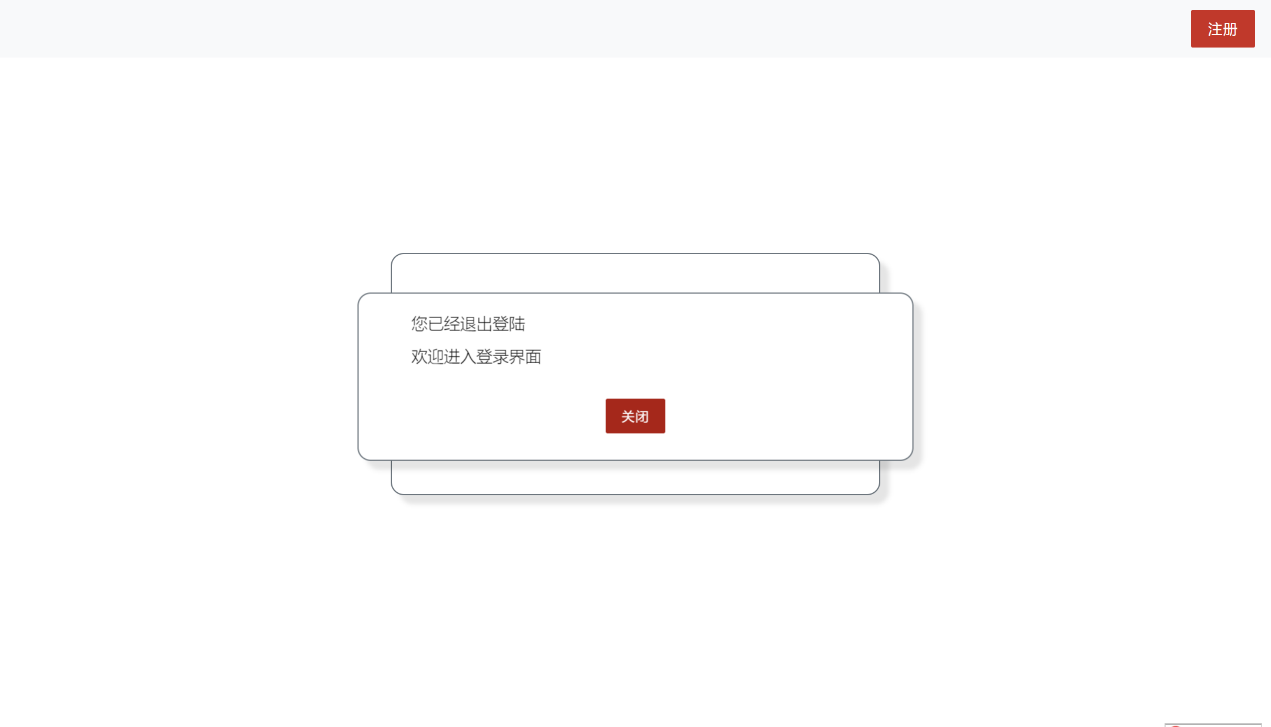
登录成功



聊天室



退出登录



## 七、结果分析

从本次项目的执行结果来看，可以完成项目的全部基本要求成功达到了实验效果。本次程序实现聊天基本功能，多个用户可进行在线实时聊天，同时可以查看历史消息。在信息交互方面，用户注册时可提示注册成功、注册不成功原因；用户登录时可弹出登录成功窗口；用户退出时，可提示成功退出，并说明“欢迎进入登录界面”。在聊天界面方面，背景界面为相关的符合聊天室主题的图片，比较符合视觉美学；主界面可以显示各个用户发送的消息，包括用户头像，用户姓名，用户发送消息的时间，用户发送的消息，支持表情包的发送，界面上滑查看历史消息；界面下方显示用户输入消息框，消息框中加入提示词“和大家说点啥吧……”，同时有提交按钮，用户输入消息点击提交则成功发送消息；界面右侧上方显示当前登录个人头像以及用户名；界面右侧显示所有聊天室成员的相关信息，包括头像以及姓名。

另外在很多方面依然具有改进空间，比如以下几点：  
1.未实现文件的共享功能。  
 在网络聊天室中，用户可能会想分享自己创建的文件，特别是在在线会议的时候，此时FTP服务就显得极为关键。由于时间紧张，我们没有添加文件共享功能，可以考虑在以后的改进中去实现  
2.网页较为简单  
 项目中的HTML代码部分，由于经验不足，编写的不够精美，可能会在一定程度上影响用户体验。  
3.个人用户主页未创建  
 虽然项目中有注册功能，但是仅仅是作为显示当前用户名称来使用的。我们还可以将存储用户名的数据库和存储消息的数据库关联起来，做到可以查询每个用户单独的历史发送消息，甚至可以在此主页上添加更多功能，使用户能够更好的展示自己  
4、用户权限未设置  
 在一般的聊天室中，通常具有不同的权限，例如禁言，同意陌生用户加入，踢人等等。我们可以通过在用户名数据库中增添权限字段名，用以标明不同用户的权限，在操作时查询此字段，来使不同的用户获得不同的使用体验。

## 八、分工情况

后端程序逻辑：熊嘉恒，张宇

前端用户界面：崔庆元，李锦洋

答辩PPT及报告：林逸凡，杨一帆

技术指导：文杰