聊天室设计文档

目录

[1 绪论 1](#_Toc15331)

[1.1 选题目的及意义 1](#_Toc26484)

[1.2 系统概述 2](#_Toc7813)

[2 需求分析 2](#_Toc22899)

[2.1功能需求 2](#_Toc8179)

[2.2其它需求 3](#_Toc24455)

[2.3系统逻辑模型 4](#_Toc15552)

[3 总体设计 6](#_Toc4519)

[3.1 系统架构 6](#_Toc19441)

[3.2 模块设计 7](#_Toc13826)

[3.3 软件结构设计 7](#_Toc8496)

[3.4 数据库设计 8](#_Toc18160)

[4 详细设计 10](#_Toc13700)

[4.1总体设计 10](#_Toc18750)

[4.2类的设计 10](#_Toc13020)

[5 系统实现 21](#_Toc32273)

[5.1 开发环境 21](#_Toc24819)

[5.2 关键技术 21](#_Toc5305)

[5.3 运行结果（主要界面展示） 22](#_Toc9815)

[6软件测试与调试 25](#_Toc25630)

[6.1 测试的目的与方法 25](#_Toc9877)

[6.2 测试方案设计 25](#_Toc12475)

[6.3 测试结果 27](#_Toc3802)

[6.3.1 功能测试 27](#_Toc27452)

[6.3.2 性能测试 29](#_Toc10406)

[6.3.3 界面测试 30](#_Toc23367)

[6.3.4 安全测试 30](#_Toc31739)

[参考文献 30](#_Toc16053)

# 1 绪论

## 1.1 **选题目的及意义**

### 1.1.1目的

在这个信息高速发展的新时代，目前人们用来信息交流的方法有非常多，比如电话、电子邮件、甚至信件等通讯手段，但是这些不方便而且有费用问题，比如电话就是属于花费较大的一种方式，而且一般情况下不能进行多人会谈，在大多数情况下都是一对一的交流；电子邮件则不是一种及时联系的手段，需要一定时间延迟才能到达，只能作为一种辅助交流的方式。聊天室应用就可以很好的解决这些问题，现阶段，网上流量通信的花销也相较于电话通信会小很多。自然会引起通信行业的大变革。人们一贯追求的都是较少的资源得到最大化的利用，“物尽其用”就是这个道理。

聊天室的经典意义在于“交流”。方便快捷的方式使得聊天室软件快速发展。目前，流行于市面的聊天类软件主要有腾讯QQ、微信；其他类型软件也附加了聊天功能，比如淘宝、支付宝、美团等，虽然主要业务不是聊天，但附加聊天功能后可以提高用户体验度，使得获得更多的用户好感；在大多数软件中，C/S结构模式占据大部分市场，在C/S结构中，其只适用于局域网，对远程访问有一定的限制，并且客户需要安装专用的客户端软件以及对客户端的操作系统一般也会有限制；相比于C/S模式，B/S结构最大的优点是可以在任何地方进行操作而不用安装任何专门的软件，只要有一台能上网的电脑就能使用，客户端零维护，维护工作量大大减少，降低了总体成本。所以本次项目实践选择B/S结构来完成一个简单的网页版聊天室系统。

### 1.1.2意义

近年来计算机技术的快速发展，特别是计算机网络的发展，越来越深刻的改变了人们生活的方方面面。使得人们能以更低廉的价格，开发出更方便、更实用的网络工具。各种在线服务系统，更是深刻的影响了人们的联系和交流方式，使得人们可以在远隔千里之遥随时通讯。过去的种种陈旧的联系方式，比如信件等 已经不能满足现代生活的需要。网上聊天系统作为一种方便消费者与人们之间联系的实用系统便应运而生。网上聊天系统是为人们进行交流和联系提供的一个平台。通过提供完善的网上聊天系统的管理，可以达到增进人们之间的交流和联系的目的。

## **1.2 系统概述**

基于WebSocket的网页版聊天室系统使用到的主要技术栈有Servlet、Tomcat、DruidDataSource、Junit、JQuery、Apache commons I/O、html,css,javascript该系统主要实现了四大功能，即常见的登录、注册、私聊、群聊；当启动tomcat后，跳转至用户登录界面，若用户注册过，即可以直接登录，输入用户名和密码后，系统自动抓取信息与数据库中的用户表数据进行匹配，匹配成功，页面跳转，用户成功进入聊天室页面；若用户未注册，则点击网页下方注册链接，此时页面跳转至注册页面按照规则输入正确的用户名和密码，系统将信息存至数据库中，即用户注册成功，页面跳转至登录界面；用户登录成功之后，进入聊天室页面，此时可进行勾选好友进行群聊或是私聊，系统默认勾选当前聊天室在线的全部用户，勾选好友结束后，就可以开始聊天了。这就是一个简单的网页版聊天室实现的基本功能。

# 2 需求分析

## **2.1功能需求**

* 用户注册

可以完成新用户注册，将注册信息添加到数据库中，用于用户的登录验证，经登录验证的用户即可进入聊天室进行聊天

* 用户登录

已经注册过的用户，输入注册时的用户名和密码。如果与数据库中的用户名和密码对应成功，则登录成功，进入聊天室。若是对应失败，则跳转至登录页面，提示用户重新输入登录信息

* 用户私聊

可用于两个用户之间的聊天

* 用户群聊

可用于多个用户之间的聊天

## **2.2其它需求**

### 2.2.1性能需求

1. **数据精确度**

基本要求就是数据精确度高，在进行用户信息管理、聊天室好友信息更新、私聊群聊好友勾选、用户接收/发送消息时，要保证数据记录的准确性，数据的精确度是网页版聊天系统中很重要的一部分内容，比如用户登录从数据库表中匹配数据，用户发送给好友信息，好友接收该信息等这就要求高精度的数据。

1. **时间特性**

* 响应时间：响应时间应该在一定时间范围内（最多不超过3秒），避免出 现长时间等待甚至无响应；比如用户在点击注册按钮时，系统出现注册 页面时的响应间隔时间等
* 传输时间：最好能在3秒钟内完成，数据传输时间应该符合软件系统开 发的标准；用户发送消息给好友，好友收到消息的时间间隔，信息传输 时间
* 运行时间：由系统运行状态、算法处理能力、计算机处理时间等多个因 素来确定

1. **适应性**

当需求发生变化时，系统必须有其适应性，运行环境应该基本不发生变化，系统会根据变化进行相应调整，并且系统在设计时，必须有可拓展性，以适应将来系统功能的拓展；还必须有兼容性，确保在不同系统，不同型号的客户端可以正常运行。

1. **故障处理**

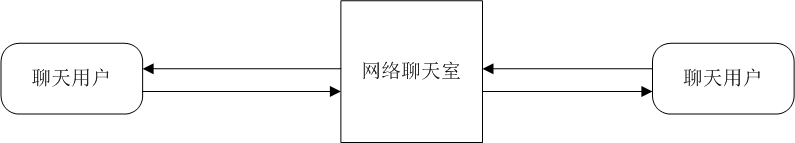
设备故障分为软件故障和硬件故障。

* 设备的硬件故障可能造成这个软件系统不能运行或不能正常进行对应功 能的操作等后果，系统的资源不足及网络传输效率低下可能造成软件系 统不能运行，或者造成“死机”，硬件故障需要用户进行处理；
* 设备自身的软件故障，比如数据库错误、软件功能存在缺陷等现象，需 要软件开发者和软件维护者协助系统管理员进行处理；
* 系统在正常运行时，不会出现错误，对于用户的错误操作，系统应给予 用户提醒以及给系统反馈，保证一些经常出现的错误操作的解决方法提 供至该软件系统的帮助部分，方便用户查询处理问题。

## **2.3系统逻辑模型**

### 2.3.1数据流图

1. 顶层数据流图：（如图1）

 图1 顶层数据流图

1. 一层数据流图：（如图2）

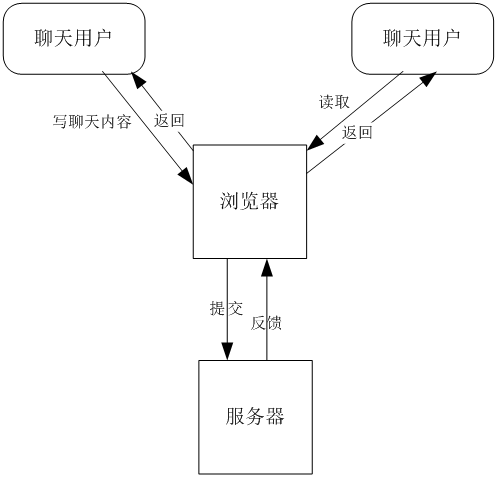


图2 一层数据流图

1. 二层数据流图：（如图3）

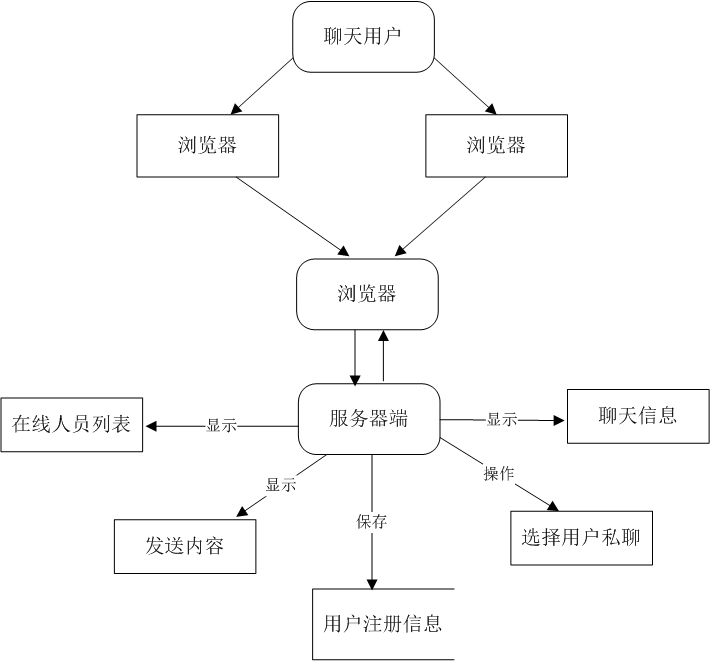


图3 二层数据流图

### 2.3.2 数据字典

|  |
| --- |
| 数据项名称：ID名  数据项别名：id  说明：用户连接标识  类型：int  长度：11  取值范围及意义：用户在数据库中用户表user中的序号 |

|  |
| --- |
| 数据项名称：用户名  数据项别名：username  说明：用户唯一标识  类型：varchar  长度：20  取值范围及意义：用户用以登录聊天室的用户名长度在1-20之内不论数字、字母还是符号。 |

|  |
| --- |
| 数据项名称：密码  数据项别名：password  类型：varchar  长度：100  取值范围及意义：用户用以登录聊天室的密码长度在1-100之内不论数字、字母还是符号。 |

# 3 总体设计

## **3.1 系统架构**

本系统采用目前流行的B/S结构，即浏览器(Browser) 和服务器( Server)结构，这种结构可以认为是对C/S结构，即客户端(Client) 和服务器( Server)结构的发展，在B/S结构下，用户只需简单通过IE、 谷歌、火狐等浏览器来实现与服务器的连接。这种结构最主要的优点在于，这种结构中的客户机几乎没什么工作量，主要承担了浏览的功能;而后端服务器承担了系统中的绝大部分的业务逻辑处理，接到用户请求后，访问数据进行数据加工处理、执行应用程序并将信息传送到前端浏览器。这也就是我们常说的三层结构，这三部分分布在系统中的不同位置：

表示层:位于最外层，离用户最近。用于显示，给用户传输直接的信息，不做任何数据处理。

功能层:它处于数据访问层与表示层中间，起到了数据交换中承上启下的作用。业务逻辑层的设计对于一个支持可扩展的架构尤为关键。

数据层:主要是负责数据库的访问，就是实现对数据表的操作。

## **3.2 模块设计**

### 3.2.1 用户注册模块

用户注册是将用户的输入正确的相关信息存储入到数据库当中，以便后续登录。

### 3.2.2 用户登录模块

将用户的输入信息与数据库中的信息对比，如果用户名和密码对应成功，则登录成功，进入聊天室。若对应失败，则跳转至登录页面，提示用户重新输入登录信息。

### 3.2.3 用户群聊模块

当前用户勾选将要群聊的几个目标用户，与多个目标用户开启群聊模式。在群聊模式下，群聊中在线的所有人都可以看到聊天的内容。该系统登录后默认进入群聊模式。

### 3.2.4 用户私聊模块

当前用户勾选将要私聊的目标用户，与目标用户开启私聊模式。在私聊模式下，只有私聊的两个人才可以看到聊天内容。

## **3.3 软件结构设计**

该聊天室系统软件结构分为前端应用服务和后端管理功能；前台应用服务包括用户注册、用户登录、勾选好友三个部分，勾选好友可分为用户私聊和用户群聊两个部分，私聊、群聊均可接受消息和发送消息；后台管理功能包括用户信息管理、信息比较、信息存储三个部分。（如图4）

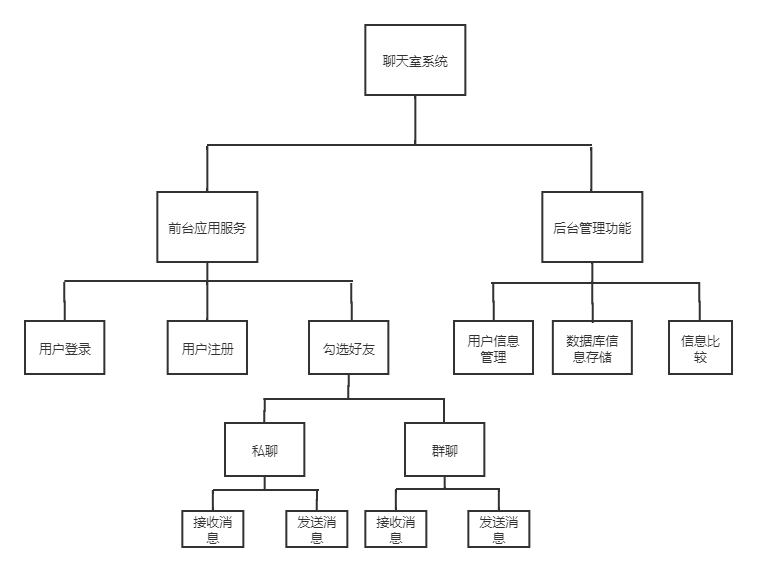


图4 聊天室软件结构图

## **3.4 数据库设计**

### 3.4.1 概念设计

E-R图：该系统中只存在一个实体，即用户，其属性有：id，用户名，密码（如图5）

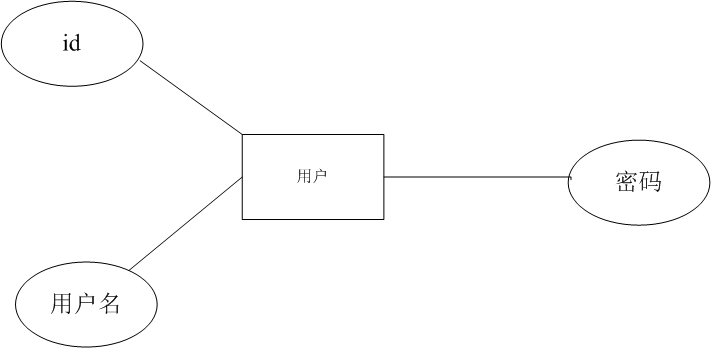


图5 用户E-R图

### 3.4.2 逻辑设计

用户表（表2）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用户表user | | | | |
| 字段名 | 数据类型 | 长度 | 说明 | 描述 |
| id | int | 11 | 主键 | 用户id |
| username | varchar | 20 | 唯一键 | 用户名 |
| password | varchar | 100 | 不为空 | 用户密码 |

### 3.4.3 物理设计

数据库类型使用mysql，建立用户表user，主键为id。

# 4 详细设计

## **4.1总体设计**

**该聊天室项目分为四层进行设计，分别为DAO、Entity、Service和Controller**

* **DAO层**：持久层
* 功能：与数据库进行交互
* 描述：DAO层包含两个类，BaseDao类和AccountDao类；其中，BaseDao中主要封装了一些基础操作,数据源加载、获取连接、关闭资源；AccountDao类中主要封装了用户模块的一些基本操作，用户注册、用户登录、将数据表信息封装到User类中；

* **Entity层**：实体层
* 功能：数据库在项目中的类
* 描述：Entity层包含三个类，分别封装了用户实体、后端发给前端的消息实体、前端发送给后端的消息实体；
* **Service层**：业务层
* 功能：控制业务
* 描述：Service层包含两个类，分别是AccountService类和WebSocket类;Service主要负责业务模块的逻辑应用设计。和DAO层一样都是先设计接口，再创建要实现的类，然后在配置文件中进行配置其实现的关联。接下来就可以在service层调用接口进行业务逻辑应用的处理。封装Service层的业务逻辑有利于业务逻辑的独立性和重复利用性。
* **Controller层**:控制层
* 功能：控制业务逻辑
* 描述：Controller层包含两个类，分别是AccountController类和LoginController类；主要负责前后端交互以及具体的业务模块流程的控制，controller层主要调用Service层里面的接口控制具体的业务流程，控制的配置也需要在配置文件中进行。

## **4.2类的设计**

聊天室后端类的结构图（如图6）

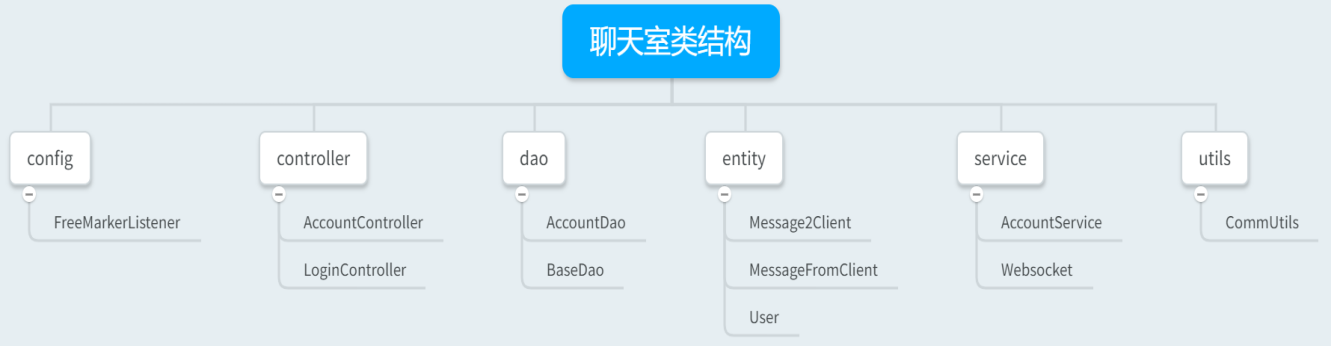
****

图6 聊天室类结构图

1. config

* FreeMarkerListener

**类的功能**：模版引擎，基于模板和要改变的数据， 并用来生成输出文本(HTML网页，电子邮件，配置文件，源代码等)的通用工具。

**类的属性**：

public static final String TEMPLATE\_KEY

**类的方法**：

@Override

public void contextInitialized(ServletContextEvent sce) {

// 配置版本

...

// 配置加载ftl的路径

...

// 配置页面编码

...

}

2. controller

* AccountController

**类的功能**：控制用户注册成功与失败相应的响应操作

**类的属性**：

private AccountService accountService

**类的方法**：

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

// 用户注册成功

// 弹框提示，返回登陆界面

writer.println("<script>\n" +

" alert(\"注册成功\");\n" +

" window.location.href = \"/index.html\";\n" +

"</script>");

}else {

// 弹框提示失败，保留原页面

writer.println("<script>\n" +

" alert(\"注册失败\");\n" +

" window.location.href = \"/registration.html\";\n" +

"</script>");

}

}

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

doGet(req,resp);

}

* LoginController

**类的功能**：控制用户登录成功与失败相应的响应操作

**类的属性**：

private AccountService accountService

**类的方法**：

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

...

// 登录失败,停留登录页面

out.println(" <script>\n" +

" alert(\"用户名或密码为空!\");\n" +

" window.location.href = \"/index.html\";\n" +

" </script>");

// 登录成功,跳转到聊天页面

// 加载chat.ftl

...

// 登录失败,停留在登录页面

out.println(" <script>\n" +

" alert(\"用户名或密码不正确!\");\n" +

" window.location.href = \"/index.html\";\n" +

" </script>");

}

}

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

doGet(req,resp);

}

private Template getTemplate(HttpServletRequest req,String fileName) {

...

}

3. service

* AccountService

**类的功能**：与AccountDao类进行数据交互，完成用户注册登录的逻辑操作

**类的属性**：

private AccountDao accountDao;

**类的方法**：

// 用户登陆

public boolean userLogin(String userName,String password) {

User user = accountDao.userLogin(userName,password);

if (user == null) {

return false;

}

return true;

}

// 用户注册

public boolean userRegister(String userName,String password) {

...

}

* Websocket

**类的功能**：封装WebSocket相关的方法操作，包括WebSocket连接、关闭、发送消息等

**类的属性**：

// 存储所有连接到后端的websocket

private static CopyOnWriteArraySet<WebSocket> clients

// 缓存所有的用户列表

private static Map<String,String> names

// 绑定当前websocket会话

private Session session

// 当前客户端的用户名

private String userName

**类的方法**：

@OnOpen

public void onOpen(Session session) {

this.session = session;//绑定当前session

// username=' + '${username}

String userName = session.getQueryString().split("=")[1];

this.userName = userName;

// 将客户端聊天实体保存到clients

clients.add(this);

// 将当前用户以及SessionID保存到用户列表

names.put(session.getId(),userName);

System.out.println("有新的连接,SessionID为"+session.getId() +

",用户名为"+userName);

// 发送给所有在线用户一个上线通知

Message2Client message2Client = new Message2Client();

message2Client.setContent(userName+"上线了!");

message2Client.setNames(names);

// 发送信息

String jsonStr = CommUtils.object2Json(message2Client);

for (WebSocket webSocket : clients) {

webSocket.sendMsg(jsonStr);

}

}

@OnError

public void onError(Throwable e) {

...

}

@OnMessage

//群聊:{"msg":"777","type":1}

//私聊:{"to":"0-","msg":"33333","type":2}

public void onMessage(String msg) {

...

}

@OnClose

public void onClose() {

...

}

public void sendMsg(String msg) {

...

}

4. entity

* Message2Client

**类的功能**：封装后端发送给前端的信息实体

**类的属性**：

// 聊天内容

private String content;

// 服务端登录的所有用户列表

private Map<String, String> names;

**类的方法**：

public void setContent(String msg) {

this.content = msg;

}

public void setContent(String userName,String msg) {

this.content = userName + "说:" + msg;

}

public String getContent() {

return content;

}

public Map<String, String> getNames() {

return names;

}

public void setNames(Map<String, String> names) {

this.names = names;

}

* MessageFromClient

**类的功能**：封装前端发送给后端的信息实体

**类的属性**：

// 聊天信息

private String msg;

// 聊天类别: 1表示群聊 2表示私聊

private String type;

// 私聊的对象SessionID

private String to;

**类的方法**：

public String getMsg() {

return msg;

}

public void setMsg(String msg) {

this.msg = msg;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getTo() {

return to;

}

public void setTo(String to) {

this.to = to;

}

* User

**类的功能**：封装用户信息实体

**类的属性**：

private Integer id;

private String userName;

private String password;

**类的方法**：

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

5. dao

* AccountDao

**类的功能**：封装用户登录、注册对数据库的具体操作以及将数据库中的用户信息封装至user对象

**类的属性**：None

**类的方法**：

//用户登录

public User userLogin(String userName,String password) {

Connection connection = null;

PreparedStatement statement = null;

ResultSet resultSet = null;

User user = null;

try {

connection = getConnection();

String sql = "SELECT \* FROM user WHERE username = ? AND " +

" password = ?";

statement = connection.prepareStatement(sql);

statement.setString(1,userName);

statement.setString(2, DigestUtils.md5Hex(password));

resultSet = statement.executeQuery();

if (resultSet.next()) {

user = getUserInfo(resultSet);

}

}catch (Exception e) {

System.err.println("查询用户信息出错");

e.printStackTrace();

}finally {

closeResources(connection,statement,resultSet);

}

return user;

}

// 用户注册

public boolean userRegister(User user) {

...

}

// 将数据表信息封装到User类中

public User getUserInfo(ResultSet resultSet) throws SQLException {

...

}

* BaseDao

**类的功能**：封装数据库相关操作，包括加载数据源、获取数据库连接以及关闭数据库相关资源

**类的属性**：

private static DataSource dataSource;

**类的方法**：

//加载数据源

static {

Properties properties = CommUtils.

loadProperties("datasource.properties");

try {

dataSource = DruidDataSourceFactory.createDataSource(properties);

} catch (Exception e) {

System.err.println("数据源加载失败");

}

}

// 获取数据库连接

protected Connection getConnection() {

...

}

// 关闭资源

protected void closeResources(Connection connection,Statement statement) {

...

}

6. utils

* Commutils

**类的功能**：封装系统中所用到的一些基础工具方法，包括加载配置文件、json序列化等

**类的属性**：

private static final Gson gson

**类的方法：**

//加载配置文件

public static Properties loadProperties(String fileName) {

Properties properties = new Properties();

// 获取当前配置文件夹下的文件输入流

InputStream in = CommUtils.class.getClassLoader()

.getResourceAsStream(fileName);

// 加载配置文件中的所有内容

try {

properties.load(in);

} catch (IOException e) {

e.printStackTrace();

}

return properties;

}

//对象->JSON字符串

public static String object2Json(Object obj) {

return gson.toJson(obj);

}

//JSON字符串->对象

public static Object json2Object(String jsonStr,Class objClass) {

return gson.fromJson(jsonStr,objClass);

}

//判断字符串是否为空

public static boolean strIsNull(String str) {

return str == null || str.equals("");

}

# 5 系统实现

## **5.1 开发环境**

操作系统：Windows10

代码开发环境：JDK8

代码运行软件：IntelliJ idea

数据库：MySQL

服务器：Tomcat

## **5.2 关键技术**

### 5.2.1 前端技术

* **html**：是一种标识性的语言。它包括一系列标签．通过这些标签可以将网络上的文档格式统一。HTML文本是由HTML命令组成的描述性文本，HTML命令可以说明文字，图形、动画、声音、表格、链接等。利用html搭建聊天室登录、注册、主界面等的页面主框架结构。
* **css**：层叠样式表，是一种用来表现HTML等文件样式的计算机语言，CSS 能够对网页中元素位置的排版进行像素级精确控制，支持几乎所有的字体字号样式，拥有对网页对象和模型样式编辑的能力。利用css渲染页面效果，优化界面的美观程度及显示效果。
* **javascript、jquery**：JavaScript是一种属于网络的脚本语言,常用来为网页添加各式各样的动态功能,为用户提供更流畅美观的浏览效果；jQuery是一个快速、简洁的JavaScript框架。利用javascript和jquery完成用户登录注册的表单信息的正则验证，页面跳转等逻辑功能。

### 5.2.2 后端技术

* **WebSocket**:WebSocket协议的⽬标是在⼀个独⽴的持久连接上提供全双⼯双向通信。客户端和服务器可以向对⽅主动发送和接受数据。在JS中创建WebSocket后，会有⼀个HTTP请求发向浏览器以发起请求。在取得服务器响应后，建⽴的连接会使⽤HTTP升级将HTTP协议转换为WebSocket协议。也就是说，使⽤标准的HTTP协议⽆法实现WebSocket，只有⽀持那些协议的专⻔浏览器才能正常⼯作。
* **Servlet**:提供请求/响应的Web服务模式
* **Tomcat**：是一个Web应用服务器，Servlet容器，通过调用Servlet的方法（doGet()或doPost()）来响应客户端请求
* **DruidDataSource**：阿里巴巴开源平台上一个数据库连接池实现，用于用户连接数据库
* **Junit**：单元测试框架，用于测试代码中各个模块功能的实现
* **Apache commons I/O**：是由Apache基金会维护创建的Java函数库，关于I/O操作的工具类库
* **JDBC**：是Java和数据库之间的一个桥梁，是一个规范而不是一个实现，能够执行SQL语句。它由一组用Java语言编写的类和接口组成。各种不同类型的数据库都有相应的实现，本项目中的代码都是采用MySQL数据库实现的。

## **5.3 运行结果（主要界面展示）**

* **登录界面**

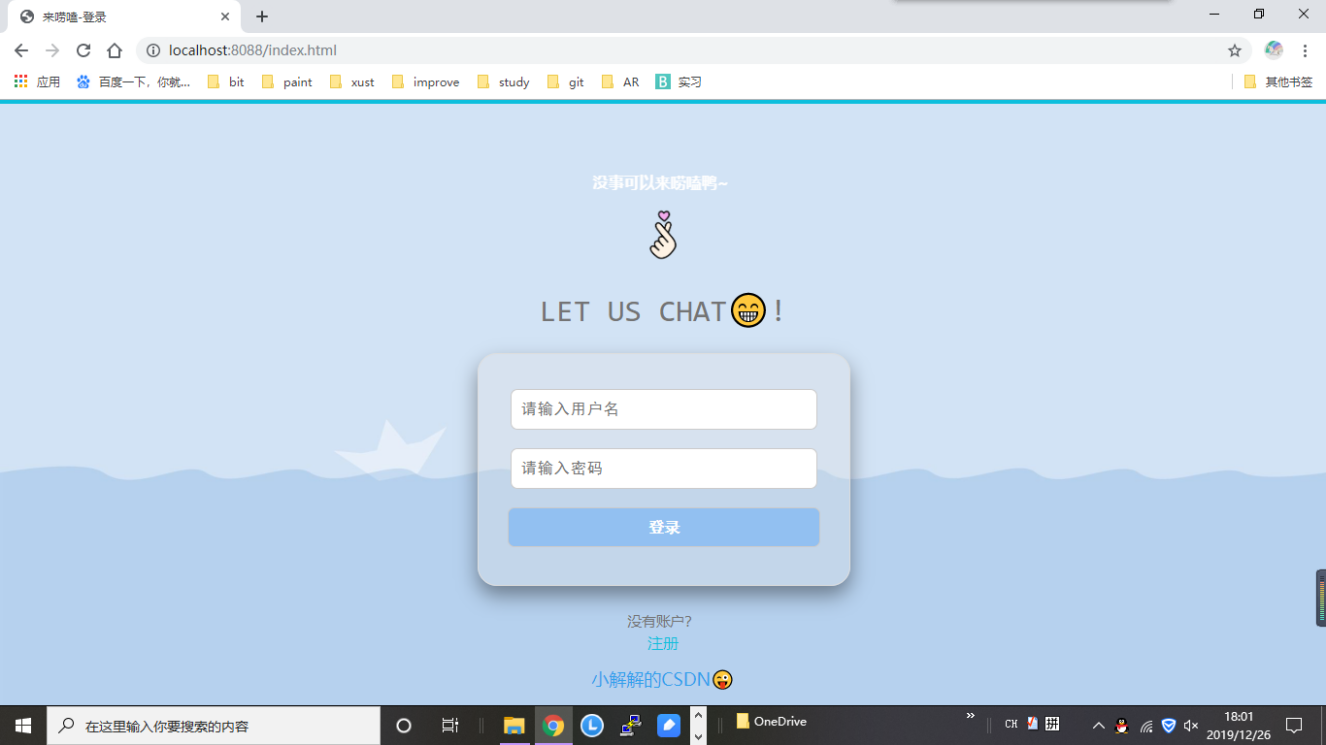


图7 登录界面图

* **注册界面**

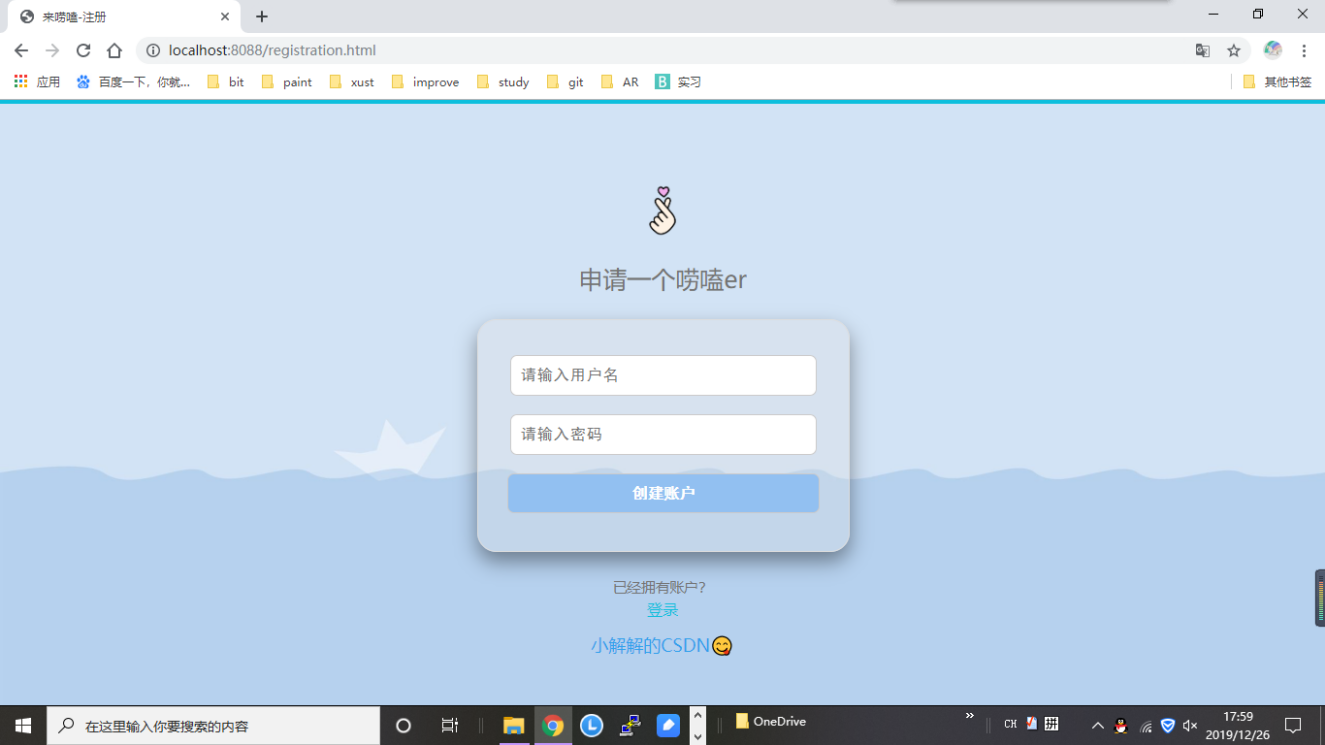


图8 注册界面图

* **发送消息界面**

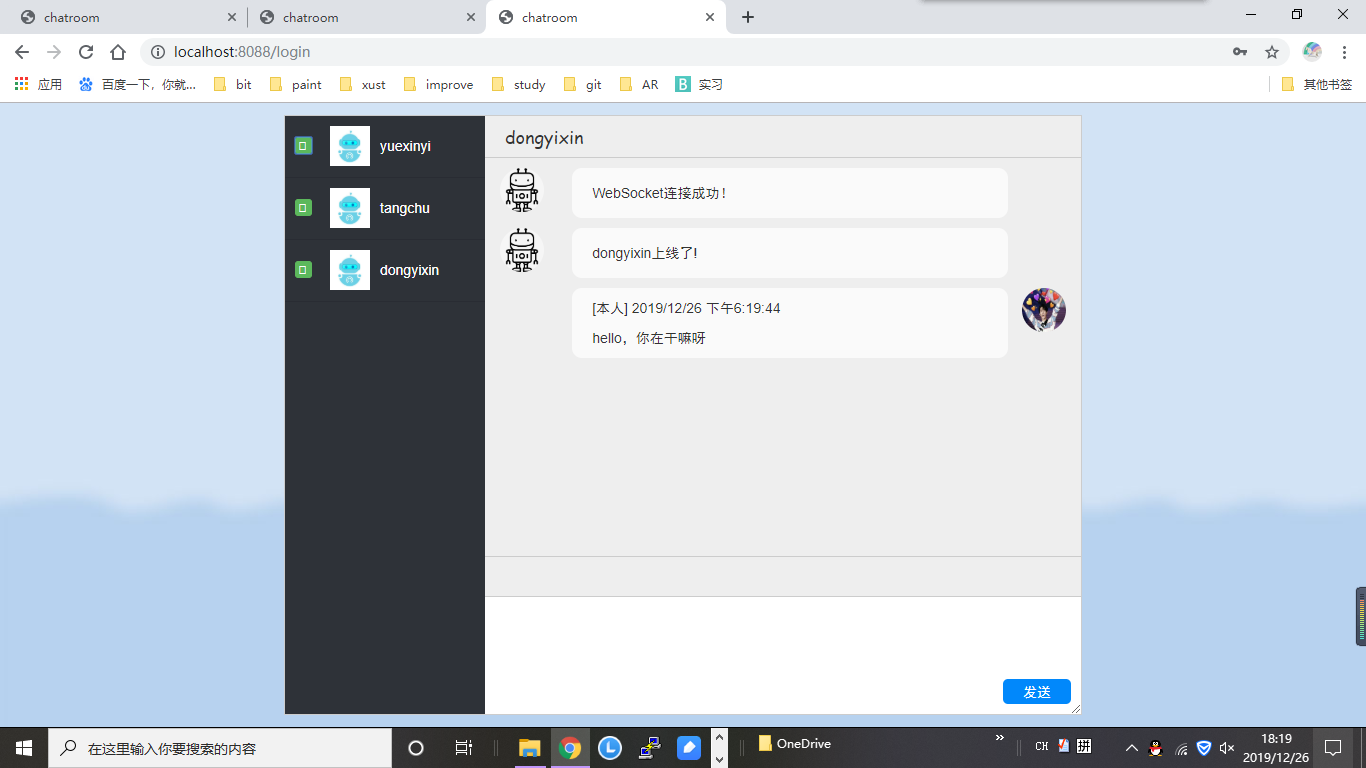


图9 发送消息界面图

* **接收消息界面**

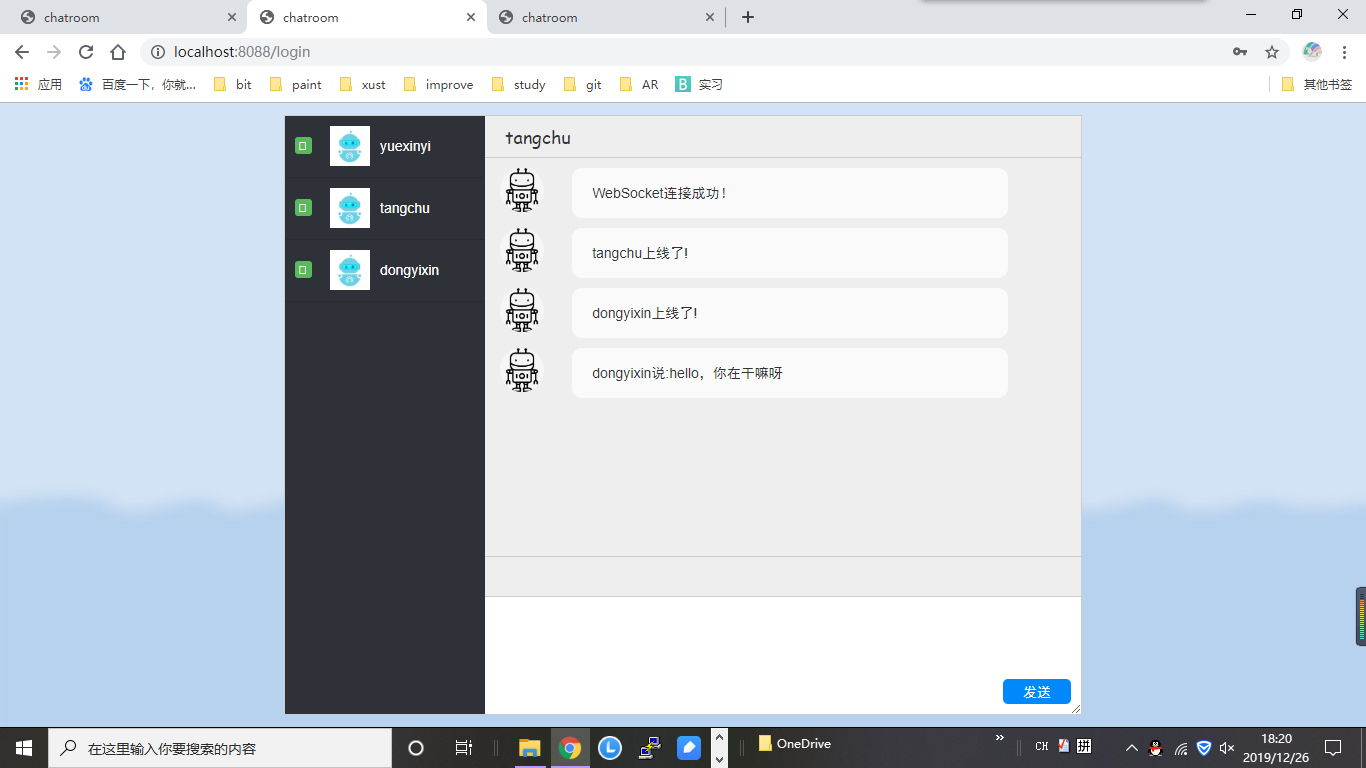


图10 接收消息界面图

# 6软件测试与调试

## **6.1 测试的目的与方法**

**测试目的**：为了保证软件产品的最终质量，在软件开发的过程中，对软件产品进行质量控制，目的是发现程序中的错误；

**测试方法**：采用黑盒测试技术和白盒测试技术对聊天室系统进行测试；

* 采用等价类结合边界值对登录注册界面的输入文本框进行了测试；
* 采用边界值和场景设计发对用户注册登录时支持的用户数、信息收发长度进行了测试；
* 采用错误推测法对密码校验中大小写、密码存储是否加密、消息收发的正确性进行了测试；
* 采用JUnit进行了单元测试，测试的模块主要包括数据库连接、用户注册模块、用户登录模块等；

## **6.2 测试方案设计**

本项目分别设计了功能测试，性能测试，界面测试和安全测试，其中功能测试中对正常和异常的情况进行了测试。设计的测试方案如下。（如图11）

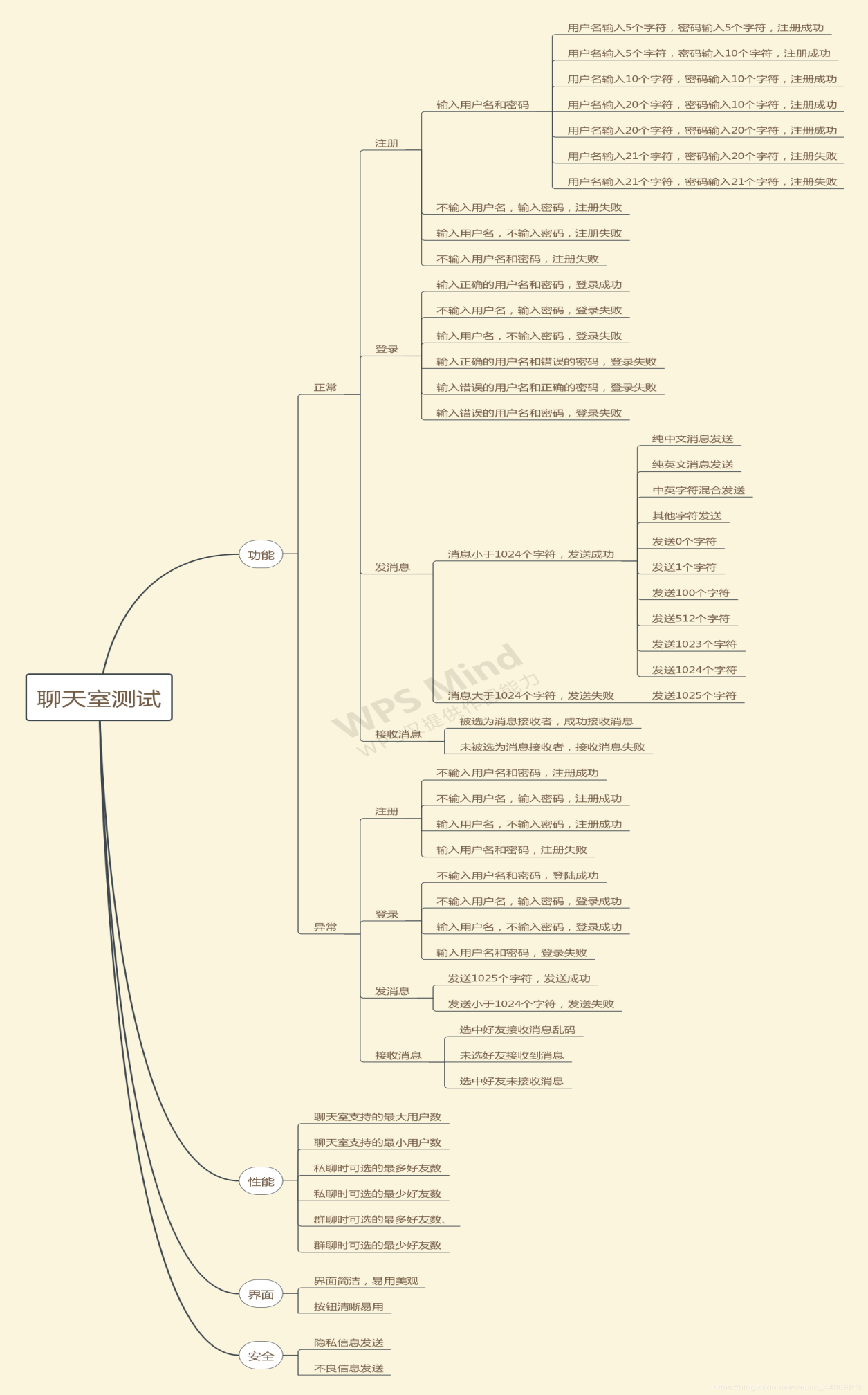


图10 测试方案设计思维导图

## **6.3 测试结果**

### 6.3.1 功能测试

#### 6.3.1.1 正常测试

测试方案设计表（表3）

|  |  |
| --- | --- |
| 测试项目 | 测试用例 |
| 注册 | 1、用户名输入5个字符，密码输入5个字符  2、用户名输入5个字符，密码输入10个字符  3、用户名输入10个字符，密码输入10个字符  4、用户名输入20个字符，密码输入10个字符  5、用户名输入20个字符，密码输入20个字符  6、用户名输入21个字符，密码输入20个字符  7、用户名输入21个字符，密码输入21个字符  8、不输入用户名，输入密码  9、输入用户名，不输入密码  10、不输入用户名和密码 |
| 登录 | 11、输入正确的用户名和密码  12、不输入用户名，输入密码  13、输入用户名，不输入密码  14、输入正确的用户名和错误的密码  15、输入错误的用户名和正确的密码  16、输入错误的用户名和密码 |
| 接收消息 | 17、被选为消息接收者  18、未被选为消息接收者 |
| 发送消息 | 19、消息小于1024个字符且纯中文消息发送  20、消息小于1024个字符且纯英文发送  21、消息小于1024个字符且中英文字符混合发送  22、消息小于1024个字符且其他字符发送  23、发送0个字符  24、发送1个字符  25、发送100个字符  26、发送512个字符  27、发送1023个字符  28、发送1024个字符  29、发送1025个字符 |

具体测试用例表（表4）

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 测试输入 | 测试结果 | 覆盖用例 |
| 1 | 用户名：stacy  密码：12345 | 注册成功，页面跳转 | 1 |
| 2 | 用户名：alice  密码：1234567890 | 注册成功，页面跳转 | 2 |
| 3 | 用户名：testaaaaaa  密码：1234567890 | 注册成功，页面跳转 | 3 |
| 4 | 用户名：testbbbbbbbbbbbbbbbb  密码：1234567890 | 注册成功，页面跳转 | 4 |
| 5 | 用户名：testcccccccccccccccc  密码：12345678901234567890 | 注册成功，页面跳转 | 5 |
| 6 | 用户名：testdddddddddddddddd  密码：  12345678901234567890 | 注册成功，页面跳转 | 6 |
| 7 | 用户名：testeeeeeeeeeeeeeeeeee  密码：  123456789012345678901 | 注册成功，页面跳转 | 7 |
| 8 | 用户名：None  密码：12345 | 注册失败，弹框提醒 | 8 |
| 9 | 用户名：stacy  密码：None | 注册失败，弹框提醒 | 9 |
| 10 | 用户名：None  密码：None | 注册失败，弹框提醒 | 10 |
| 11 | 用户名：stacy  密码：12345 | 登录成功 | 11 |
| 12 | 用户名：None  密码：12345 | 登录失败，弹框提醒 | 12 |
| 13 | 用户名：stacy  密码：None | 登录失败，弹框提醒 | 13 |
| 14 | 用户名：stacy  密码：1234 | 登录失败，弹框提醒 | 14 |
| 15 | 用户名：stac  密码：12345 | 登录失败，弹框提醒 | 15 |
| 16 | 用户名：stac  密码：1234 | 登录失败，弹框提醒 | 16 |
| 17 | 不勾选该好友 | 该好友未收到消息 | 17 |
| 18 | 勾选该好友 | 该好友收到消息 | 18 |
| 19 | 发送消息：你好呀 | 消息发送成功 | 19 |
| 20 | 发送消息：hello | 消息发送成功 | 20 |
| 21 | 发送消息：hello,世界 | 消息发送成功 | 21 |
| 22 | 发送消息：@hi | 消息发送成功 | 22 |
| 23 | 发送消息：None | 消息发送失败 | 23 |
| 24 | 发送消息：a | 消息发送成功 | 24 |
| 25 | 发送消息：a...(100个字符) | 消息发送成功 | 25 |
| 26 | 发送消息：a...(512个字符) | 消息发送成功 | 26 |
| 27 | 发送消息：a...(1023个字符) | 消息发送成功 | 27 |
| 28 | 发送消息：a...(1024个字符) | 消息发送成功 | 28 |
| 29 | 发送消息：a...(1025个字符) | 消息发送失败 | 29 |

#### 6.3.1.2 异常点测试

测试方案设计表（表5）

|  |  |
| --- | --- |
| 测试项目 | 测试用例 |
| 注册 | 1、不输入用户名，输入密码，注册成功  2、输入用户名，不输入密码，注册成功  3、不输入用户名，不输入密码，注册成功  4、输入用户名和密码，注册失败 |
| 登录 | 5、不输入用户名，输入密码，登录成功  6、输入用户名，不输入密码，登录成功  7、不输入用户名和密码，登录成功  8、输入正确的用户名和错误的密码，登录成功  9、输入错误的用户名和正确的密码，登录成功  10、输入正确的用户名和密码，登录失败 |
| 接收消息 | 11、被选中好友接受消息乱码  12、未被选中好友接收到消息  13、被选中好友未接收到消息 |
| 发送消息 | 14、发送小于1024个字符，发送失败  15、发送大于1024个字符发送成功 |

### 6.3.2 性能测试

测试方案设计表（表6）

|  |  |  |
| --- | --- | --- |
| 序号 | 测试用例 | 测试结果 |
| 1 | 聊天室支持的最大用户数 | 最多支持30个用户 |
| 2 | 聊天室支持的最小用户数 | 最少支持1个用户 |
| 3 | 私聊时可选的最多好友数 | 最多可选1个好友 |
| 4 | 私聊时可选的最少好友数 | 最少可选0个好友 |
| 5 | 群聊时可选的最多好友数 | 最多可选当前在线的所有好友数 |
| 6 | 群聊时可选的最少好友数 | 最少可选1个好友 |
| 7 | 发送消息的最大字符数 | 最多可发送1024个字符 |
| 8 | 发送消息的最小字符数 | 最少可发送1个字符 |

### 6.3.3 界面测试

* 界面简洁，易用美观
* 按钮清晰易用

### 6.3.4 安全测试

* 隐私消息发送
* 不良信息发送

# 参考文献

[1]软件工程导论第六版 张海藩、牟永敏 清华大学出版社 2013.8.1 52~139

[2]UML系统建模与分析设计 刁成嘉 机械工业出版社 2008.1 32~67

[3]数据库系统概论 王珊、萨师煊 高等教育出版社 2014.9.16 65~160

[4]计算机网络应用技术教程第四版 吴功宜、吴英 清华大学出版社 2014.1.1 35~147