1．实习目的

本课程要求学生能够综合应用 Java 基础知识和面向对象程序设计方法，根据问题需求设计数据结构和算法，独立完成一个中、小型软件的设计与开发，提高数据结构和算法设计能力、运用 Java 语言编程解决实际问题的能力。通过本课程设计的训练，让学生体会 “OOA→OOD→OOP”的全过程，增强学生理论与实践相结合的能力，同时提升学生撰写技术文档的规范化水平，为今后从事软件项目的研发打下坚实的基础，促进学生成长为既有扎实的理论知识又有较强动手能力的软件技术人才。

2．实习任务与要求：

本课程要求学生通过资料查阅和学习，了解包括海洋领域知识在内的相关领域知识，参考教材《Java 语音实验与课程设计指导》中的案例，结合《面向对象程序设计》和《数据结构》课程中所学知识，根据问题需求设计数据结构和算法，独立完成一个中、小型软件的设计与开发，提高数据结构和算法设计的能力以及运用 Java 语言编程解决实际问题的能力。实习可选用 NetBeans、IntelliJ IDEA、Eclipse 等集成开发环境为工具，以提高开发效率。建议采用 UML 建模技术进行系统的分析设计，***在 Microsoft Visio 中画出系统用例图和类图，并将 UML 图复制到设计报告中。***

通过这次实习，要求掌握以下内容：

1）面向对象技术中的继承与多态（重载和覆盖）机制、各种修饰符的使用；

2）类、包、接口的定义与使用；

3）常用数据结构相关的工具类与算法实现（数组、向量、字符串、链表、栈、队列、哈希表等）；

4）Java 常用标准 GUI 组件及其事件处理；

5）Java 的异常处理机制；

6）Java 的数据库连接技术；

7）Java 的多线程技术与多媒体技术；

8）涉海管理系统的数据结构和算法的设计方法及实现技术；课程设计可选用 NetBeans、Eclipse 等作为开发平台以提高开发效率，尽可能熟练掌握其中一种集成开发环境。建议采用 UML 建模技术进行系统的分析设计，在 Visio 中***画出系统用例图和类图，并将 UML 图复制到设计报告中***。

选题：局域网聊天程序的设计与实现

选用的开发工具：IntelliJ IDEA Ultimate 2023.1.3

3．实习说明书

3.1 需求分析

需求分析：

为了满足用户的实时聊天需求，我们计划开发一个 Web 聊天室项目。该项目将提供一个用户友好的界面，让用户能够方便地进行在线聊天，并且可以与多个用户同时进行聊天。为了保障隐私和贯彻“少即是多”的设计理念，只有在获得授权的情况下后台才会有限的查看聊天历史。同时只有在获得授权的情况下后台将会限制个别用户发言。

1. 用户界面：我们将设计一个直观、简洁且易于使用的用户界面。用户可以在界面上看到在线用户数量，查看其他用户消息并发送消息。

2. 多用户聊天：该聊天室将支持多个用户同时在线进行聊天。用户可以创建自己的账号或以匿名方式进入聊天室，与其他在线用户进行实时的文字交流。

3. 即时通信：我们将使用 Websockets 技术实现实时通信功能。这意味着当有新消息发送时，用户会立即收到，并且聊天界面会自动更新以显示最新的聊天内容。

4.隐私安全：为了增强用户隐私和安全性，登录后的用户将无法查看之前的聊天历史消息。这样可以确保用户只能查看与他们登录时的聊天内容，限制了对历史消息的访问权限。当用户离开时所有消息都会清空。

5.管理控制：为了配合官方需要，有限提供禁言和导出历史消息功能，该功能需要管理员授权，历史消息会已加密方式存储部分副本。

选题意义：

这个项目的选题意义在于提供一个实践的机会，让我能够熟悉和掌握使用 Spring Boot、Websockets 和 Maven 构建 Web 聊天室的技术。通过这个项目，我将学习如何使用 Spring Boot 框架快速构建一个现代化的 Web 应用程序，并了解实时通信的原理和实现方式。同时，我还能够掌握 Maven 的使用和依赖管理，以及如何进行前后端的协同开发。

对于用户来说，该项目的选题意义在于为用户提供一个简洁、干净、自由和具有隐私保护的交流空间。在当今数字化时代，人们越来越依赖于在线交流，出现的聊天软件也越来越繁琐和复杂。负载的软件似乎让我们忘记了一些事情，或许我就想纯粹的与他人交流，如昙花一现般的交流。而一个优质的 Web 聊天室可以满足用户的需求，少即是多。

通过完成这个项目，我将达到以下目标：

- 掌握 Spring Boot 框架的基本概念和使用方法，包括配置、路由、控制器等。
- 理解 Websockets 技术在实时通信中的应用，以及如何处理客户端和服务器之间的双向通信。
- 学习使用 Maven 进行项目构建和依赖管理，以提高开发效率和代码质量。
- 培养团队合作和协调能力，通过前后端的协同开发，将项目顺利推进并最终完成。

通过这个项目，我将能够提升自己的技术水平，为将来的 Web 开发工作打下坚实的基础，并且为用户提供一个方便、实用且有趣的 Web 聊天室应用。

根据上面的分析，系统设计如图 3-1 所示。

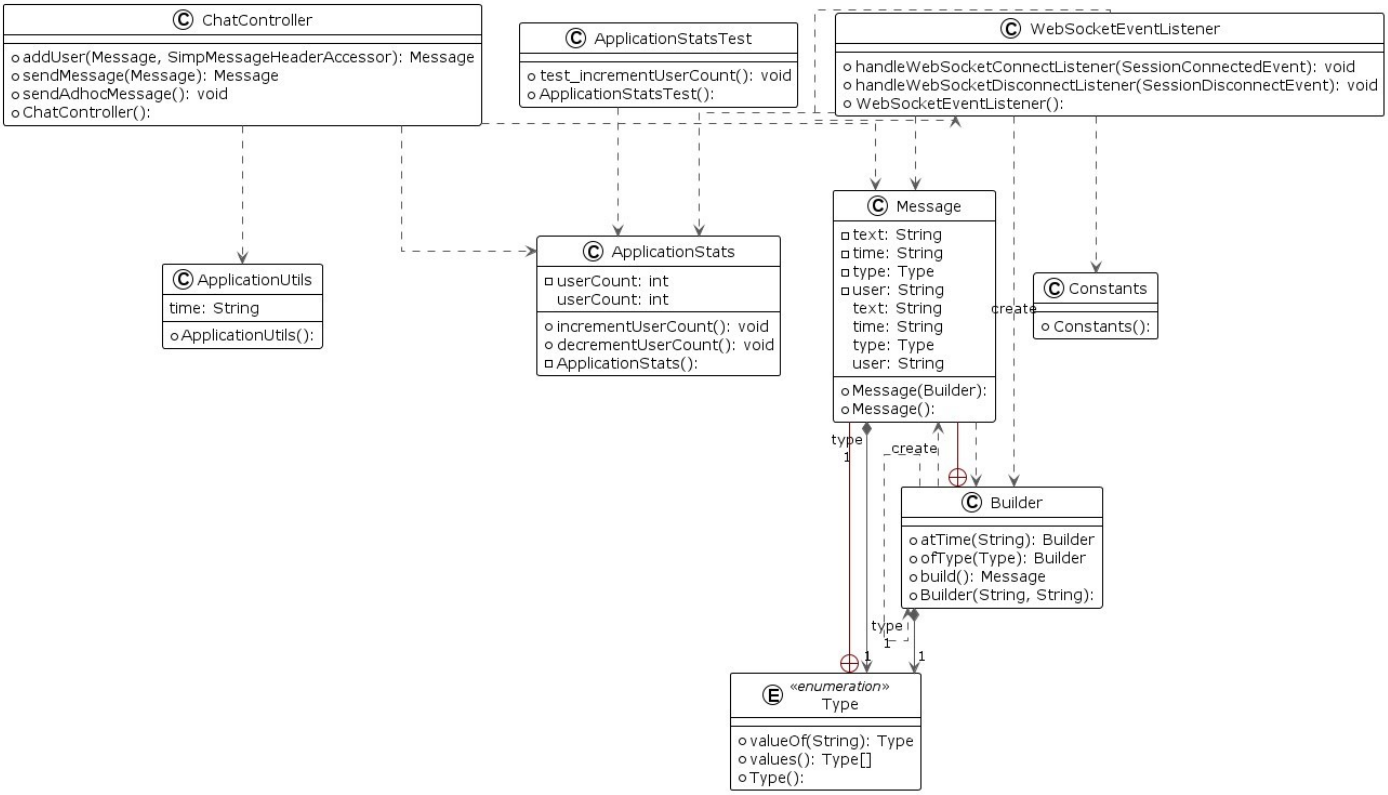

图 3-1-1　UML 类图_1

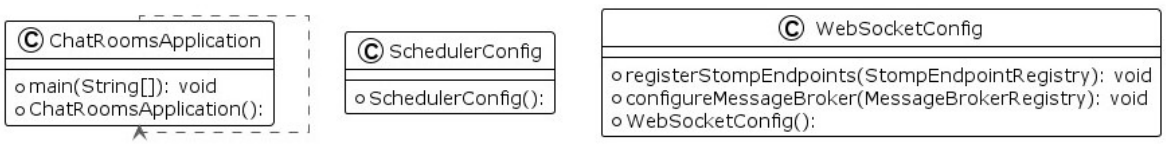

图 3-1-2　UML 类图_2

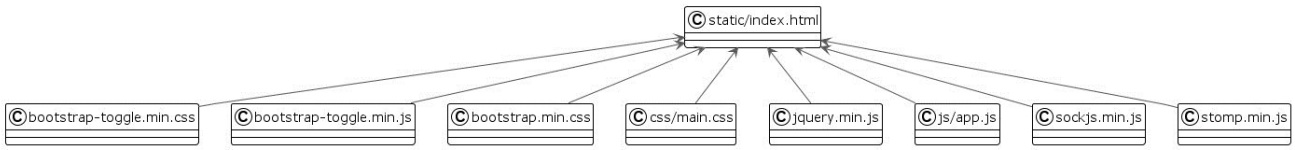

图 3-1-3　JavaScrip 模块依赖图

图 3-1-4　JavaScrip 类依赖图

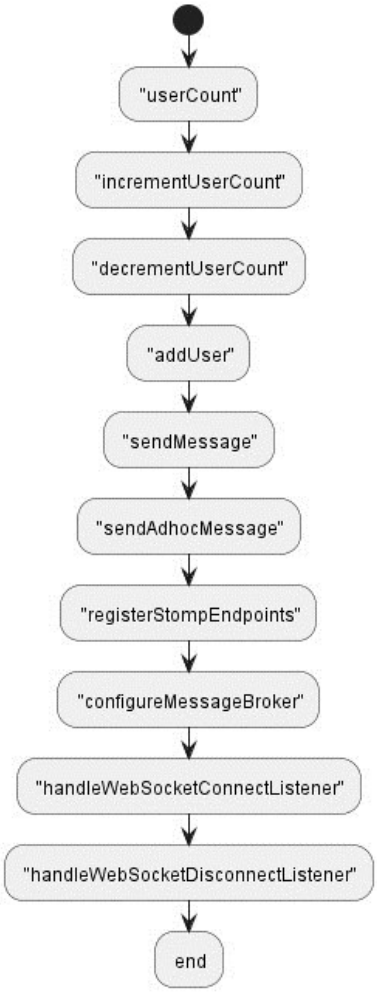

图 3-1-5 简易业务流程图

3.2 概要设计

Java 后端代码概要设计：

ApplicationStats: 应用程序统计类，用于记录用户数量并提供增加和减少用户数量的方法。

ApplicationStatsTest: 应用程序统计测试类，用于测试增加用户数量的方法。

ApplicationUtils: 应用程序工具类，用于获取当前时间。

Builder: 构建器类，用于构建消息对象。

ChatController: 聊天控制器类，处理用户添加、发送消息和发送即时消息的操作。

ChatRoomsApplication: 主应用程序类，包含主方法用于启动应用程序。

Constants: 常量类，包含应用程序中使用的常量值。

Message: 消息类，包含消息的文本、时间、类型和用户信息。

SchedulerConfig: 调度器配置类，用于配置定时任务。

Type: 枚举类型，表示消息的类型。

WebSocketConfig: WebSocket 配置类，配置 Stomp 端点和消息代理。

WebSocketEventListener: WebSocket 事件监听器类，处理 WebSocket 连接和断开事件。

JavaScript 前端代码概要设计：、

bootstrap-toggle.min.css：Bootstrap 插件的 CSS 文件。

bootstrap-toggle.min.js：Bootstrap 插件的 JavaScript 文件。

bootstrap.min.css：Bootstrap 框架的 CSS 文件。

css/main.css：自定义的主要 CSS 文件。

jquery.min.js：jQuery 库的 JavaScript 文件。

js/app.js：应用程序的 JavaScript 文件。

sockjs.min.js：SockJS 库的 JavaScript 文件。

static/index.html：静态 HTML 文件，整个应用程序的入口文件。

stomp.min.js：STOMP 库的 JavaScript 文件。

3.3 详细设计及实现

Java 后端代码设计：、

ApplicationStats 类：

userCount：私有整数类型变量，表示用户数量。

incrementUserCount()：增加用户数量的方法。

decrementUserCount()：减少用户数量的方法。

ApplicationStats()：构造函数，初始化用户数量为 0。

ApplicationStatsTest 类：

test_incrementUserCount()：测试增加用户数量的方法。

ApplicationStatsTest()：构造函数。

ApplicationUtils 类：

time：私有字符串类型变量，表示时间。

ApplicationUtils()：构造函数。

Builder 类：

atTime(String)：设置消息时间的方法，返回 Builder 对象。

ofType(Type)：设置消息类型的方法，返回 Builder 对象。

build()：构建消息对象的方法，返回 Message 对象。

Builder(String, String)：构造函数，接收消息文本和用户名称作为参数。

ChatController 类：

addUser(Message, SimpMessageHeaderAccessor)：添加用户的方法，接收消息和消息头访问器作为参数，返回消息对象。

sendMessage(Message)：发送消息的方法，接收消息作为参数，返回消息对象。

sendAdhocMessage()：发送自定义消息的方法，无返回值。

ChatController()：构造函数。

ChatRoomsApplication 类：

main(String[])：应用程序的入口方法，接收命令行参数作为参数。

ChatRoomsApplication()：构造函数。

Constants 类：

Constants()：构造函数。

Message 类：

text：私有字符串类型变量，表示消息文本。

time：私有字符串类型变量，表示消息时间。

type：私有枚举类型变量，表示消息类型。

user：私有字符串类型变量，表示消息发送者。

Message(Builder)：构造函数，接收 Builder 对象作为参数。

Message()：构造函数。

SchedulerConfig 类：

SchedulerConfig()：构造函数。

Type 枚举：

valueOf(String)：根据字符串值获取对应的枚举类型。

values()：获取所有枚举类型的数组。

Type()：构造函数。

WebSocketConfig 类：

registerStompEndpoints(StompEndpointRegistry)：注册 WebSocket 端点的方法，接收 StompEndpointRegistry 对象作为参数。

configureMessageBroker(MessageBrokerRegistry)：配置消息代理的方法，接收 MessageBrokerRegistry 对象作为参数。

WebSocketConfig()：构造函数。

WebSocketEventListener 类：

handleWebSocketConnectListener(SessionConnectedEvent)：处理 WebSocket 连接事件的方法，接收 SessionConnectedEvent 对象作为参数。

handleWebSocketDisconnectListener(SessionDisconnectEvent)：处理 WebSocket 断开连接事件的方法，接收 SessionDisconnectEvent 对象作为参数。

WebSocketEventListener()：构造函数。

应用过程的说明如下：

1. 应用程序启动时，会执行`ChatRoomsApplication`类的`main`方法。
2. 在`main`方法中，会创建一个 Spring Boot 应用程序，并开始初始化过程。
3. 在初始化过程中，会根据配置信息加载并注册相应的 Bean。
4. `WebSocketConfig`类会被自动扫描，并注册为 WebSocket 的配置类，用于配置 WebSocket 端点和消息代理。
5. `ChatController`类会被自动扫描，并注册为一个控制器，处理用户的添加、消息发送等操作。
6. 在应用程序启动时，`ApplicationStats`类会被创建，用于统计和管理用户数量。
7. `SchedulerConfig`类会被自动扫描，并配置定时任务。

8. 当应用程序完全初始化后，应用程序会等待客户端的连接请求。

9. 客户端可以使用 WebSocket 协议连接到应用程序，并进行聊天操作。

10. 当有新用户连接或断开连接时，`WebSocketEventListener`类会监听相应的 WebSocket 事件，并执行相应的处理逻辑。

11. 当有用户发送消息时，`ChatController`类会接收到消息并进行处理，包括广播消息给其他在线用户。

12. 应用程序会持续运行，直到被关闭或终止。

JavaScript 前端代码设计：

bootstrap-toggle.min.css：这是 Bootstrap 插件的 CSS 文件。它定义了开关按钮（toggle button）的样式，使其具有吸引力和交互性。

bootstrap-toggle.min.js：这是 Bootstrap 插件的 JavaScript 文件。它为开关按钮提供了交互功能，例如切换状态、处理事件等。

bootstrap.min.css：这是 Bootstrap 框架的 CSS 文件。它包含了大量的 CSS 样式规则，用于快速构建响应式和现代化的网页设计。

css/main.css：这是自定义的主要 CSS 文件。它包含了应用程序的特定样式规则，用于自定义页面的外观和布局。

jquery.min.js：这是 jQuery 库的 JavaScript 文件。jQuery 是一个功能强大且流行的 JavaScript 库，提供了简化 DOM 操作、事件处理、动画效果等功能。

js/app.js：这是应用程序的主要 JavaScript 文件。它可能包含了应用程序的逻辑和交互代码，处理用户输入、发送和接收数据等操作。

sockjs.min.js：这是 SockJS 库的 JavaScript 文件。SockJS 是一个 JavaScript 库，提供了对 WebSocket 协议的封装和跨浏览器支持，用于实现实时通信功能。

static/index.html：这是静态 HTML 文件，可能是整个应用程序的入口文件。它定义了网页的结构、布局和内容，并加载其他的 CSS 和 JavaScript 文件。

stomp.min.js：这是 STOMP 库的 JavaScript 文件。STOMP（Simple Text Oriented Messaging Protocol）是一种消息传递协议，用于在客户端和服务器之间进行异步通信。

应用过程的说明如下：

static/index.html 作为主要的入口文件，依赖于其他文件。其中，bootstrap-toggle.min.css 和 bootstrap-toggle.min.js 提供了 Bootstrap 插件的样式和交互功能，bootstrap.min.css 提供了 Bootstrap 框架的

样式，css/main.css 提供了自定义的样式，jquery.min.js 提供了 jQuery 库的支持，js/app.js 是应用程序的主要 JavaScript 文件，sockjs.min.js 和 stomp.min.js 提供了与 SockJS 和 STOMP 协议相关的功能。

4. 实习成果

4.1 源程序清单

表 4-1　源程序清单

源程序名	程序功能描述	代码行数
main/config/SchedulerConfig.java	调度配置类	10
main/config/WebSocketConfig.java	WebSocket 配置类	30
main/controller/ChatController.java	聊天控制器类	40
main/event/WebSocketEventListener.java	WebSocket 事件监听器类	50
main/model/ApplicationStats.java	应用统计模型类	30
main/model/Message.java	消息模型类	80
main/utils/ApplicationUtils.java	实用工具类	20
main/utils/Constants.java	常量类	10
main/ChatRoomsApplication.java	主类	10
main/ApplicationStatsTest.java	应用统计测试类	40
static/index.html	静态 HTML 文件	110
static/js/app.js	前端 JavaScript 文件	120
static/css/main.css	前端 CSS 文件	200
application.properties	配置文件	0

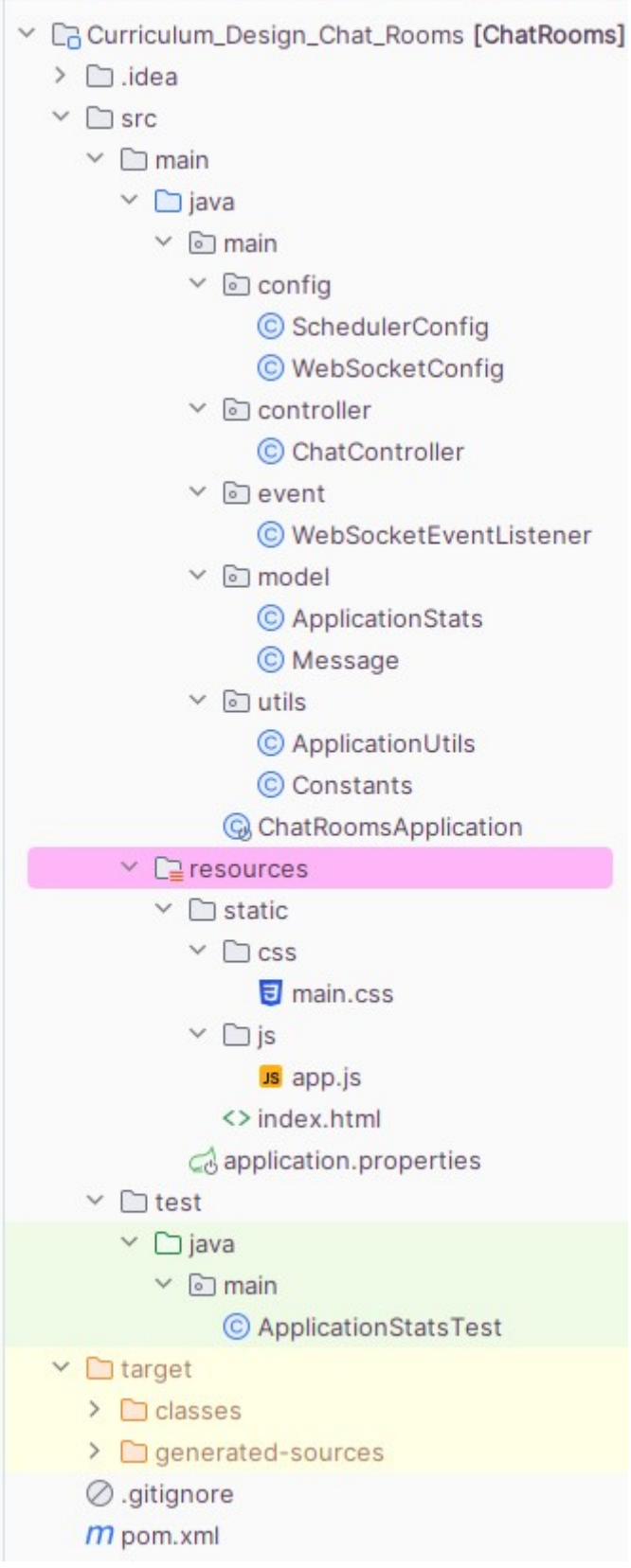

图 4-1 项目所有文件截图

Maven 项目 POM 文件

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>me.amarpandey</groupId>
    <artifactId>ChatRooms</artifactId>
    <version>0.0.1</version>
    <packaging>jar</packaging>

    <name>chat-rooms</name>
    <description>Real time public/private chat application using spring boot web-sockets</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.5.RELEASE</version>
        <relativePath/>
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-websocket</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
```

```
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.4.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>ChatRooms</finalName>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.0.0-M5</version>
                <dependencies>
                    <dependency>
                        <groupId>org.junit.jupiter</groupId>
                        <artifactId>junit-jupiter-engine</artifactId>
                        <version>5.3.2</version>
                    </dependency>
                </dependencies>
            </plugin>
        </plugins>
    </build>
</project>
```

4.2 系统调试分析

后端代码和前端代码没有出现影响开发的大问题，代码相对比较简单，在使用 Maven 构建项目时出现过很多次问题，图 4-2-1、图 4-2-2 为当时情况的复现，当前镜像 Maven 的库中缺少需要的版本，所以需要更换镜像，使用代理并使用 Maven clean 和重新构建。

前端设计因为要贯彻我本人 “少即是多”的设计理念出现过美观问题，但是简单修改即可。

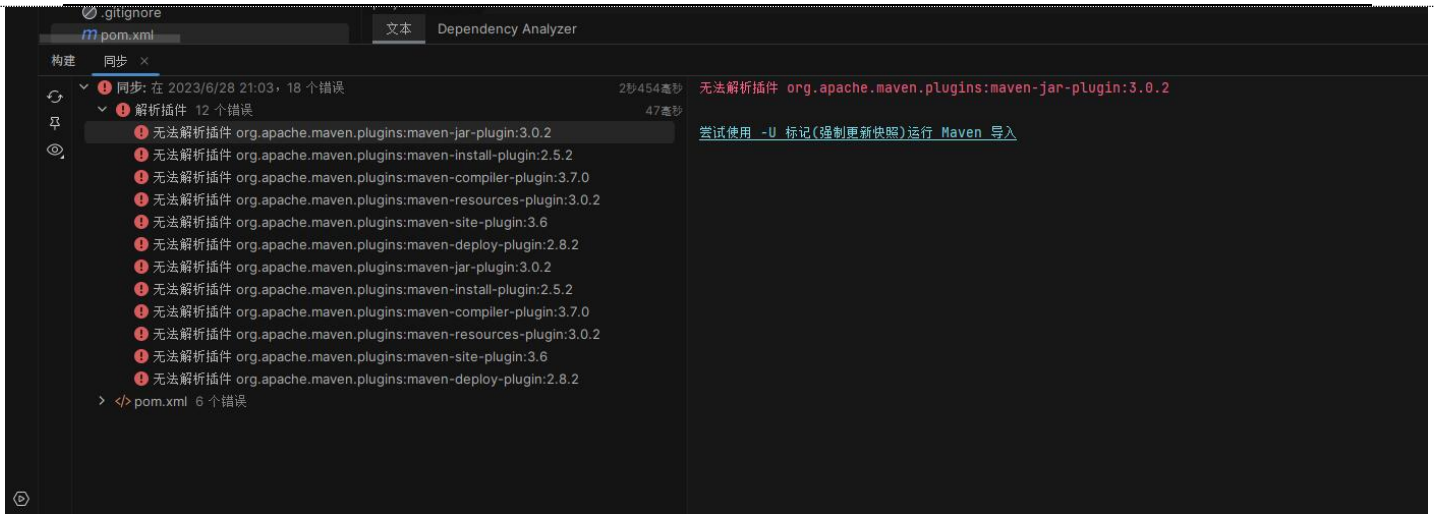

图 4-2-1

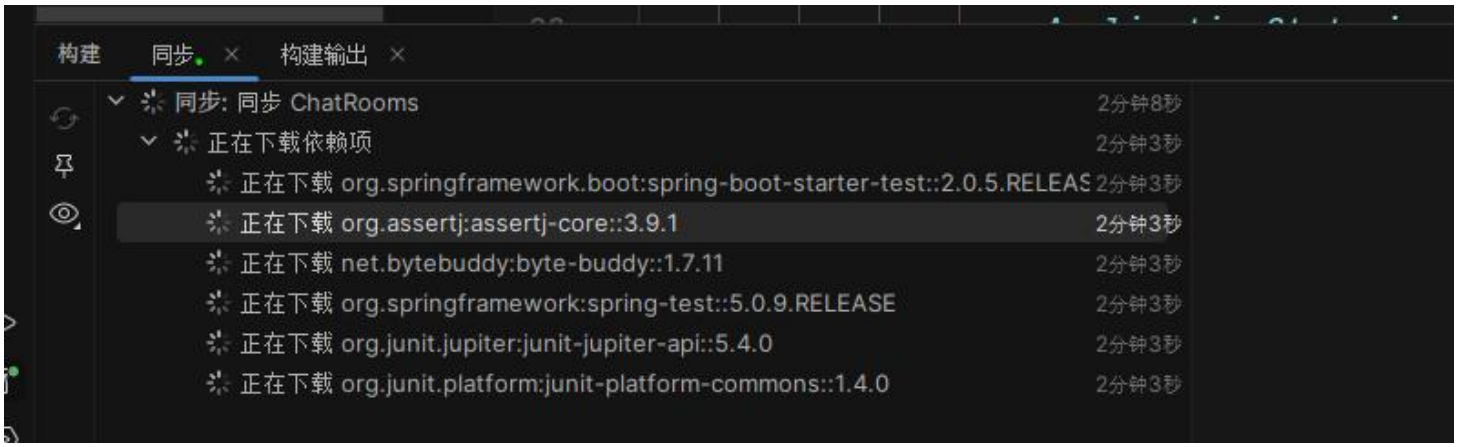

图 4-2-2

修改的时候遇到个调试了几个小时的 bug，想要当前用户气泡为绿色，他人为蓝色。问题图如下：

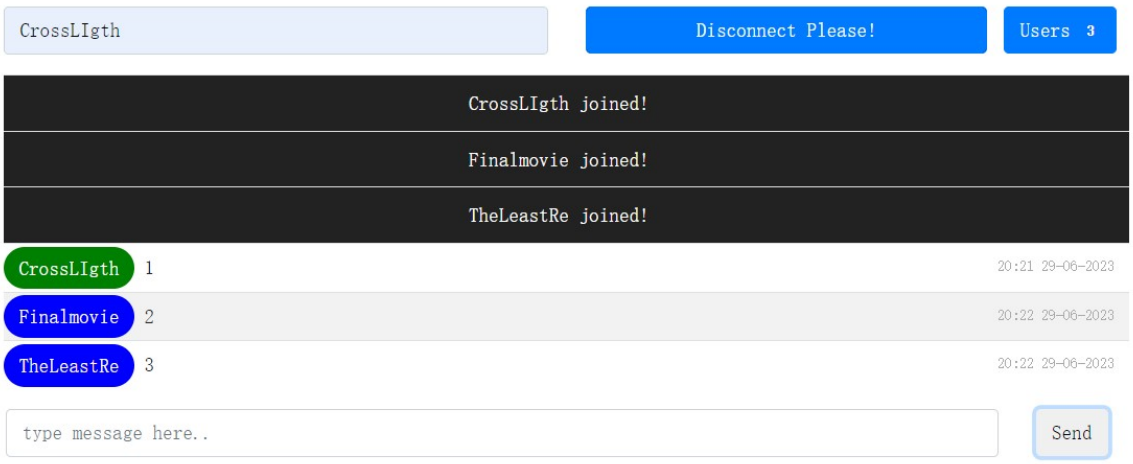

图 4-2-3

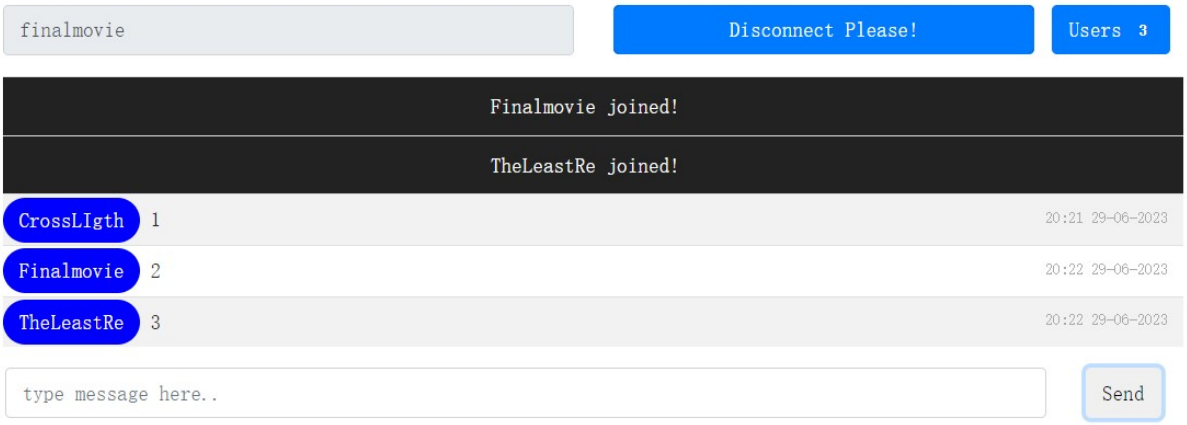

图 4-2-4

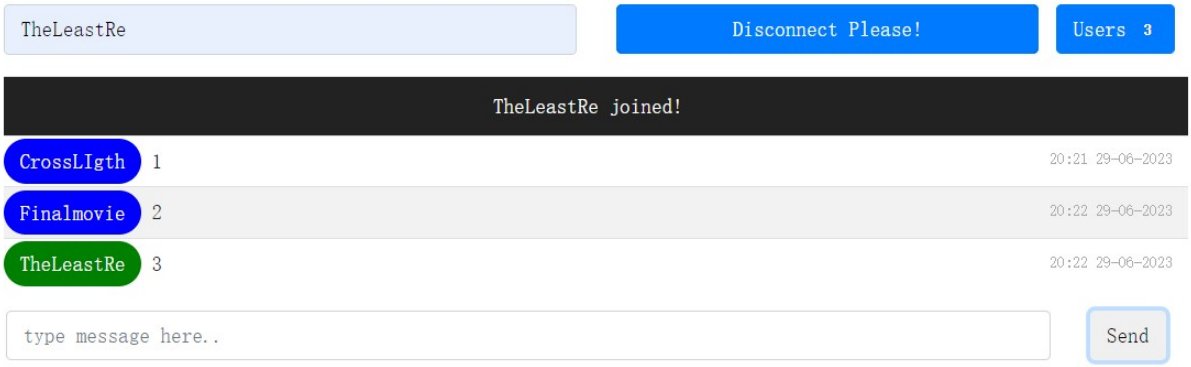

图 4-2-5

问题描述：会间歇性的出现某个用户界面气泡没有变成绿色，反复调试后观察源代码：

```
function showMessage(message) : void {
    var user = capitalizeFirstLetter(message.user);
    var text = message.text;
    var time : … = message.time;
    var username = $("#name").val();

    // 根据发送者设置用户名的背景颜色
    var nameBackgroundColor : string = (user === username) ? 'green' : 'blue';

    // 构建消息的 HTML
    var html : string = '<tr><td><span class="name-info" style="background-color: ' + nameBackgroundColor + '">' + user + '</span> ' + text + ' <span class="time-info">' + time + '</span></td></tr>';

    // 将消息添加到聊天记录表格中
    $("#userinfo").append(html);
```

图 4-2-6

代码是将

var username = $("#name").val();与

var user = capitalizeFirstLetter(message.user);进行比较，而后者被我优化过使其首字母大写！

因此比较结果不同，没有更改 Style，最后修正处理一下即可：

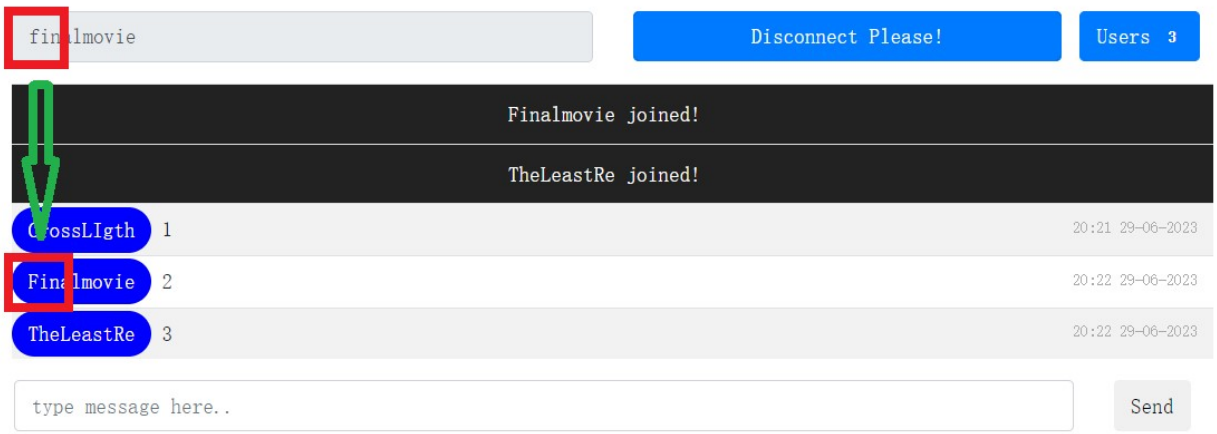

图 4-2-7

4.3 系统运行测试

图 4-2-1 编译运行

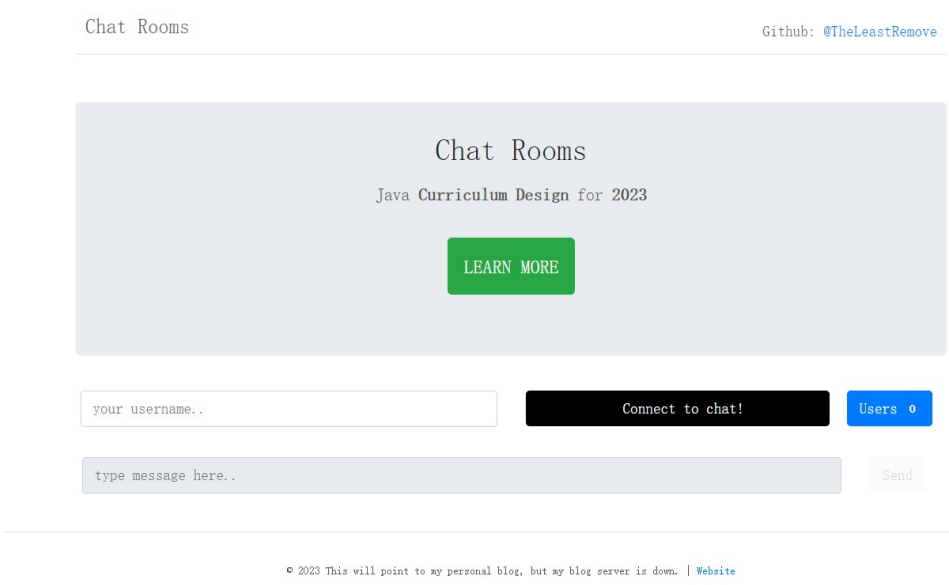

图 4-2-2 Locolhost:8080 主界面

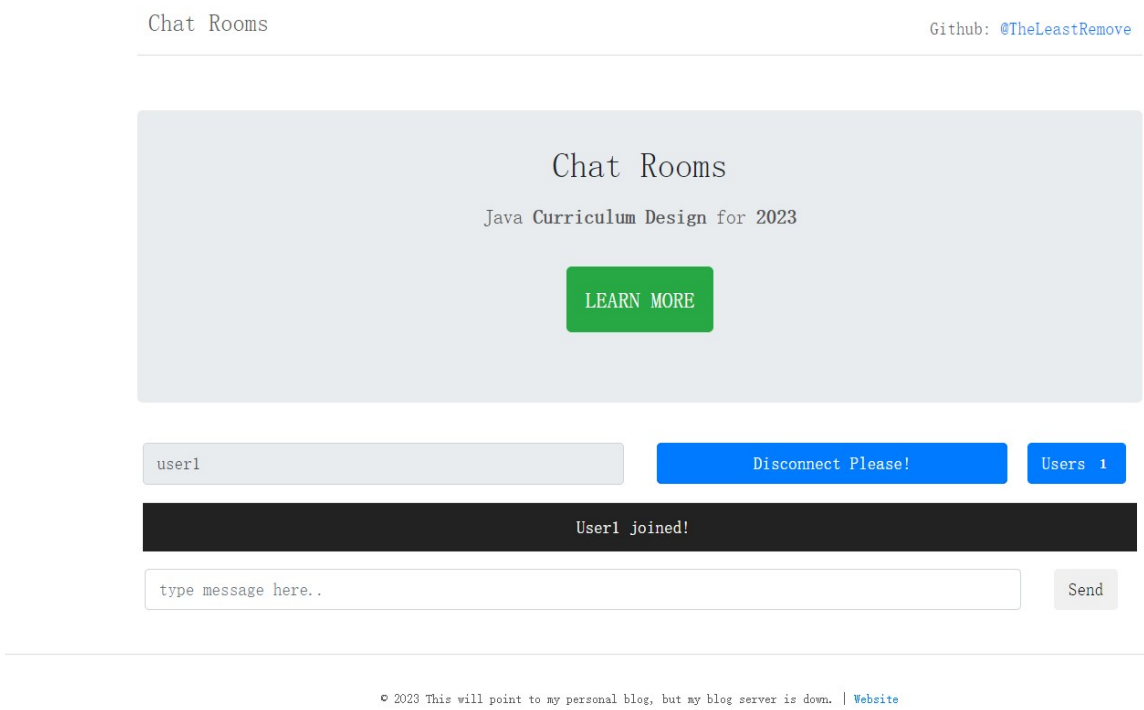

图 4-2-3 单用户登录

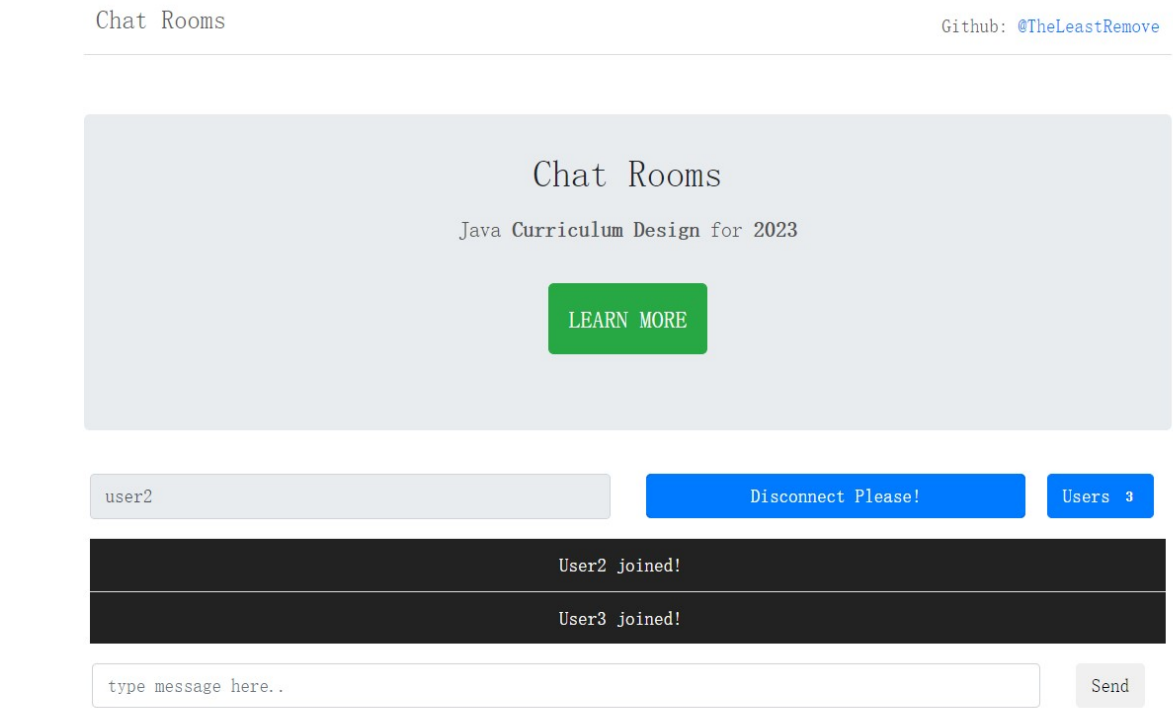

图 4-2-4 多用户（并发）登录

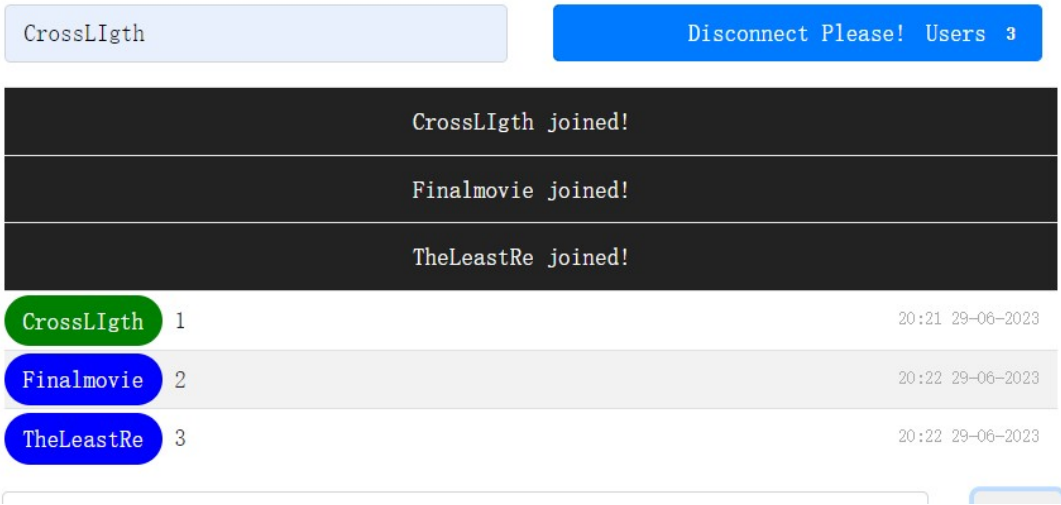

图 4-2-5 相互聊天

图 4-2-6 后台反馈

附加功能（管理员控制）

在授权情况下可以控制用户和导出聊天记录。

图 4-2-6 管理员密码验证

图 4-2-7 功能菜单

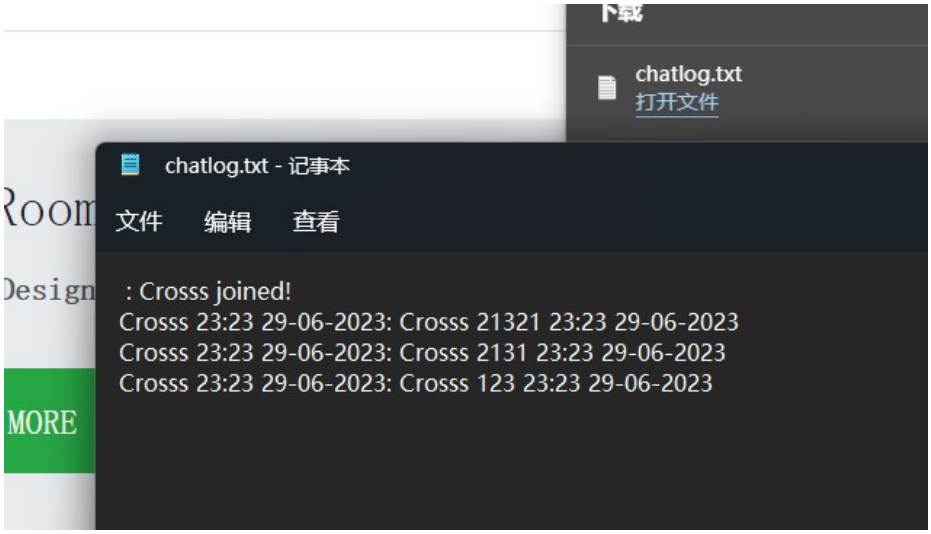

图 4-2-8 导出聊天记录

5.实习心得

在这个项目中，我使用了 Spring Boot、Websockets 和 Maven 构建了一个 Web 聊天室。这些技术的组合可以快速构建一实时通信应用程序，其中 Websockes 是我非常想尝试的新东西，Maven 提供了方便的库管理但是也很浪费时间。接下来就聊聊都用到了什么。

Spring Boot 是一个基于 Spring 框架的工具，它提供了许多便利功能，例如自动配置、自动装配和嵌入式 Web 服务器。这使得我可以更快地构建和部署应用程序，而不必关注各种配置细节。同时，Spring Boot 还提供了许多扩展和插件，可以轻松地集成其他框架和技术。用起来很方便和愉快。

Websockets 是一种在 Web 浏览器和服务器之间建立实时双向通信的协议。它允许服务器主动向客户端推送消息，而不需要客户端发起请求。这对于实时通信应用程序非常有用，例如聊天室、游戏等等。它和之前学过的 http 协议有很多不同，全双工和通过网络传输的任何实时更新或连续数据流真的非常强大和有趣。

Maven 是一个项目管理工具，它可以自动下载和管理项目依赖项。使用 Maven 可以轻松地管理项目的构建过程，包括编译、测试、打包和部署。Maven 还提供了许多插件，可以帮助开发人员完成各种任务，例如代码检查、文档生成等等，但是实际使用中会出现很多网络问题和版本问题，要是有 python 的 pip 一样方便就好了。

因为时间问题，就局限于本地了，网页的 LearnMore 按钮会跳转到 github 库，之后我会把服务迁移到服务器上。 同时从项目一开始我其实打算使用 Rust 重写 TCP（基于原有 UDP 协议），Rust 作为系统级编程语言，非常适合编写高性能和高可靠性的网络程序。但是最后没有完成，要考虑和做的事情太多，时间太少，在未来，我会继续努力学习并完成更多 Rust 和 Java 项目。

至于前端和 js 了，我的评价是有手就行。Js 的使用过程中非常愉快，有点喜欢了。第一次验收后增加了管理员控制，优化了面板，顺便修改了一些交互逻辑。