非凡聊天室概要设计

目录

1. 引言

1.1. 目的

本篇概要设计主要目的是为了清楚地介绍聊天室的功能、操作方法以及使用规则，从而帮助用户能够安全、有效地使用这一工具。以下是编写聊天室详细说明书时预期的读者群体：

(1) 开发人员：他们需要了解系统的详细功能需求、技术架构、接口定义以及实现细节，以便正确且高效地构建系统。

(2) 聊天室用户：他们需要了解如何注册、登录、使用聊天室的各项功能（如创建聊天室、加入聊天室、发送消息、设置个人偏好等）。

1.2. 背景

a. 非凡聊天室

b. 本项目的任务提出者：卓伊杰

开发者：卓伊杰，邵振峰，冯博，卢亚晗，袁硕，代小雨

用户：所有人员

运行该程序系统的计算中心：Windows 11 家庭中文版

1.3. 术语与缩写解释

缩写、术语	解 释
Go	Go 是一个开源的编程语言，它能让构造简单、可靠且高效的软件变得容易。
Vue	Vue.js 是一套构建用户界面的渐进式框架
NSQ	NSQ 是 Go 语言编写的一个开源的实时分布式内存消息队列，其性能十分优异
NATS	NATS 是一种允许以消息形式进行数据交换的基础设施。我们称之为“面向消息的中间件”。
…	

1.4. 参考资料

[1]《go 语言之路》，电子工业出版社出版，李文周

[2]Vue 官网：https://cn.vuejs.org/guide/quick-start.html

[3]go 语言教程：https://www.runoob.com/go/go-tutorial.html

[4]聊天室设计:https://golang2.eddycjy.com/posts/ch4/03-requirement-and-design/

2. 总体设计

2.1. 需求规定

2.1.1. 系统功能

允许多个用户聊天，文件上传下载，发送表情以及登录等功能

2.1.2. 系统性能

2.1.2.1. 精度

微秒级的定时精度

2.1.2.2. 时间特性要求

最大延迟为 100-200ms

2.1.2.3. 可靠性

错误处理和恢复能力:Go 语言内置了强大的错误处理机制,可以优雅地处理各种运行时错误。通过合理的错误处理策略,可以最大程度地减少系统崩溃的风险,提高系统的健壮性。

并发控制和同步机制:

Go 语言的并发编程模型基于 Goroutine 和通道的,可以非常方便地控制并发访问和共享资源。通过合理的并发控制和同步机制,可以避免死锁、数据竞争等并发问题,保证系统的正确性。

自动垃圾回收:

Go 语言内置了高效的自动垃圾回收机制,无需手动管理内存,降低了内存泄漏的风险。这有助于系统长期稳定运行,避免因内存问题而导致的崩溃。

可靠的网络通信:

Go 标准库提供了稳定且高性能的网络通信库,如 net 和 http。

结合 WebSocket 库,可以实现高可靠的实时双向通信,减少网络问题导致的系统故障。

可靠的消息队列:

使用可靠的消息队列系统,如 NSQ 或 NATS,可以确保消息的可靠投递,降低消息丢失的风险。消息队列还可以提供负载均衡和故障转移等功能,提高系统的可用性。

完善的监控和日志:

结合日志记录库,如 Logrus,可以全面记录系统运行时的各种事件和错误信息。配合监控系统,可以快速发现并诊断系统问题,有利于提高系统的可靠性。

2.1.2.4. 灵活性

部署灵活性:

Go 编译出的二进制文件是独立可执行的,可以在不同的操作系统和硬件平台上运行。

可配置性:

Go 语言支持通过配置文件或环境变量等方式动态设置系统参数,如监听端口、日志级别等。这使得我们的聊天室系统可以轻松地适应不同的运行环境,提高系统的适应性。

高性能可扩展:

Go 语言天生具有高并发性和高性能的特点,可以支撑大规模用户的实时通信需求。结合负载均衡和水平扩展技术,可以轻松地提升系统的处理能力和吞吐量。

良好的可维护性:

Go 语言的静态类型检查和自带的测试框架,可以帮助开发者编写高质量的可维护代码。

良好的代码组织和注释规范,使得系统的扩展和维护变得更加简单高效。

2.1.3. 输入输出要求

要求用户输入的文件必须是 pdf 或者 txt 格式的文件，取决于系统自定义要求

2.1.4. 数据管理能力要求

高效的数据缓存:

对于频繁访问的数据,如在线用户列表、最近聊天记录等,可以使用缓存系统进行缓存。

可靠的数据备份与恢复:

对于重要的聊天记录和用户信息,需要定期进行备份,以防止数据丢失。

安全可控的数据访问:

聊天室系统需要严格控制数据的访问权限,确保只有授权用户才能查看和操作相关数据。利用 Go 语言的安全编程特性,如输入验证和加密操作,可以有效防范各种数据安全隐患。

2.1.5. 故障处理要求

1.聊天室系统要确保关键数据,如聊天记录和用户状态等,在故障时不会丢失或不一致。

2.聊天室系统需要对各种异常情况做出合理的容错处理,如网络中断、服务器崩溃、资源耗尽等。

2.1.6. 其他专门要求

2.2. 运行环境

2.2.1. 设备

正常可联网的计算机设备、win10 及以上操作系统，台式以及笔记本

2.2.2. 支持软件

VMware、kaliLinux、vscode

2.2.3. 接口

用户接口：系统需要提供方便、易懂、易操作的用户界面，用户可以发消息。 界面必须响应迅速，操作简单直观，发送的消息不泄露。

文件接口： 系统提供的可供用户上传 txt、pdf 文件的接口，以及用户可以下载系统的文件。 文件上传类型限制，限制为 txt 和 pdf，下载穿透过滤保证用户下载的数据没有安全问题。

2.2.4. 控制

配置文件和环境变量:

聊天室系统的许多配置信息,如服务器地址、端口号、数据库连接等,通常会存储在配置文件或环境变量中。配置信息可以在系统初始化时读取,并用于控制系统的运行。

命令行参数和标志:

在启动聊天室服务时,可以通过命令行参数或标志来传递一些控制信号,如是否开启调试模式、日志级别等。这些参数可以在程序启动时解析,并用于控制系统的行为。

HTTP API 和 WebSocket 事件:

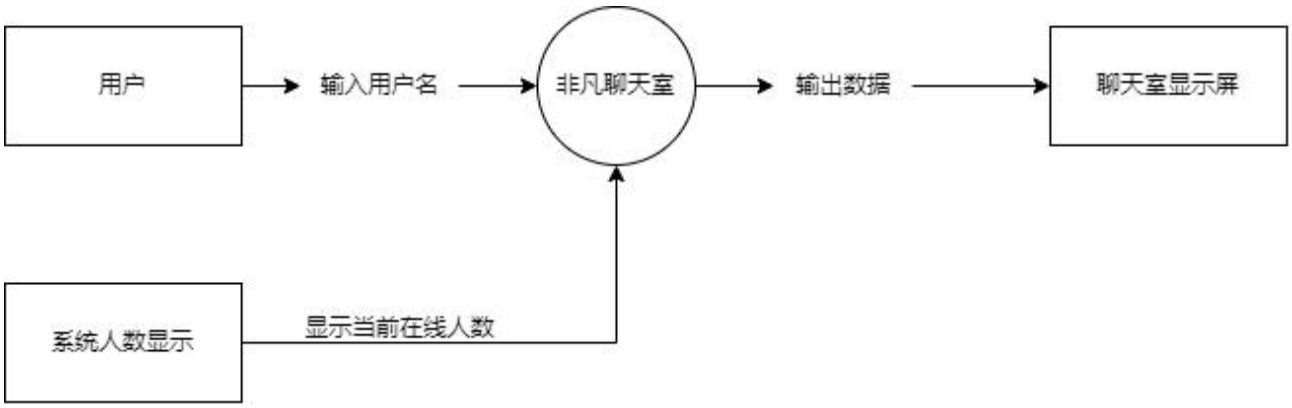
聊天室系统使用 WebSocket 接口,用于与客户端进行交互。这些接口可以用于接收来自客户端的各种控制信号,如添加新用户、修改用户权限、发送消息等。这些信号通常来自客户端应用程序的用户操作。

内部事件和信号:

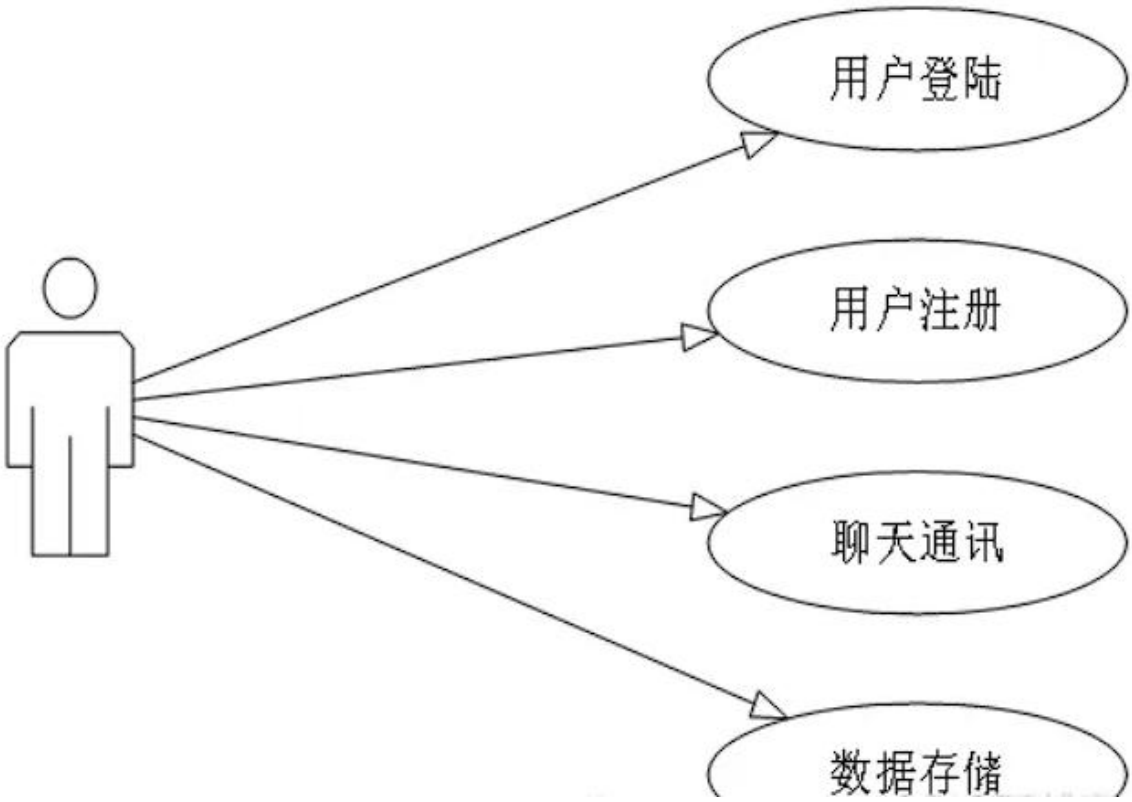
在聊天室系统内部,也可能存在各种事件和信号,用于控制系统的运行。例如,当有新用户加入或退出聊天室时,系统会产生相应的事件,以触发对应的处理逻辑。这些事件通常由系统内部的组件或模块产生。

2.3. 基本设计概念和处理流程

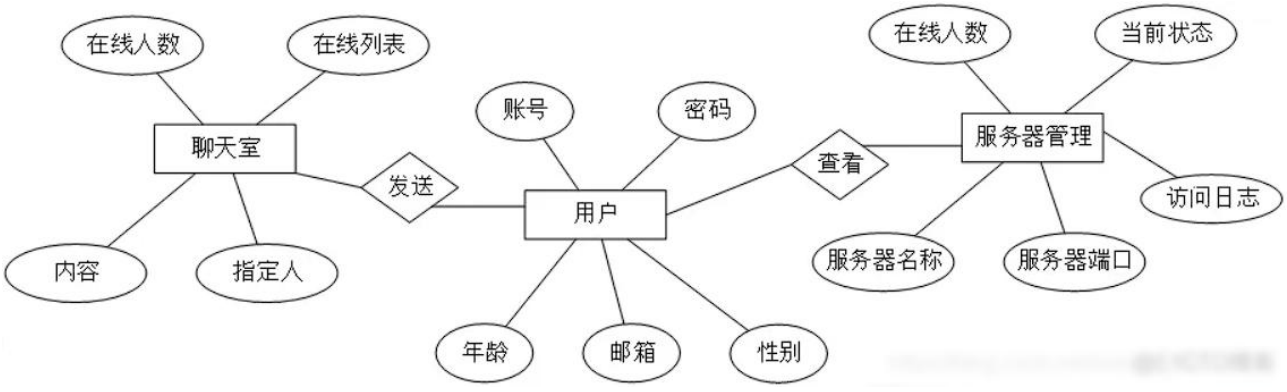
顶流设计图：



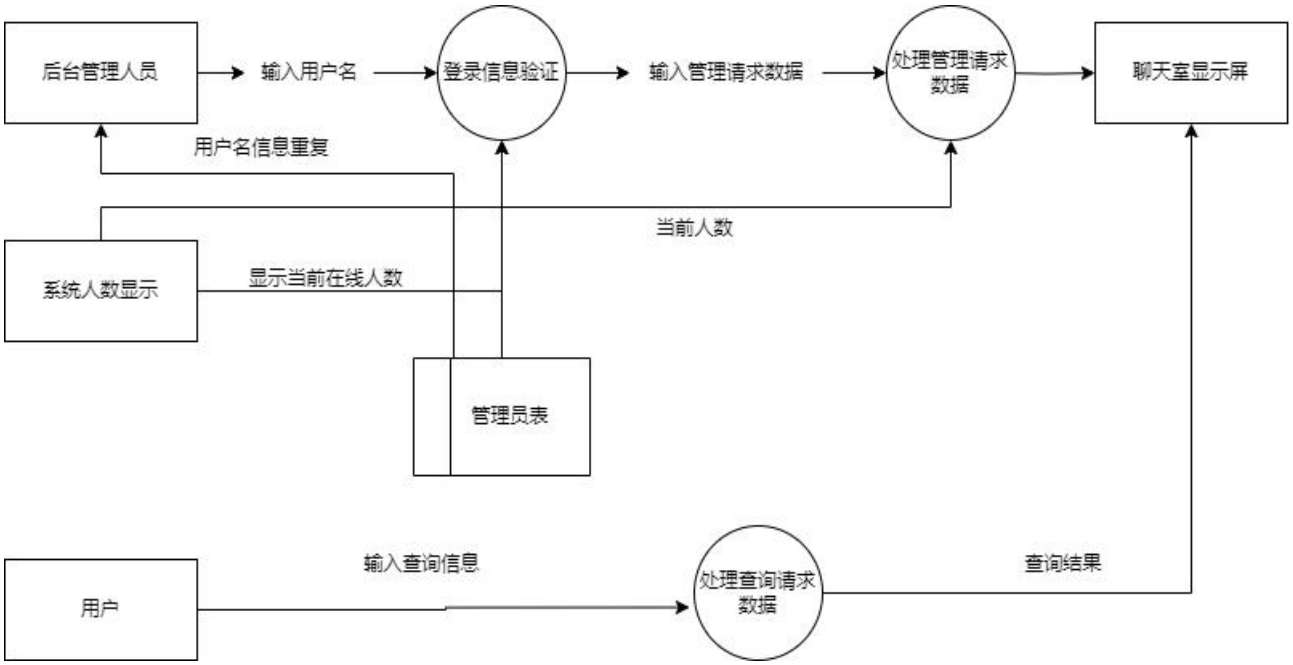
聊天室管理系统主要满足用户与用户之间的通讯，实现数据的交互



E-R 图

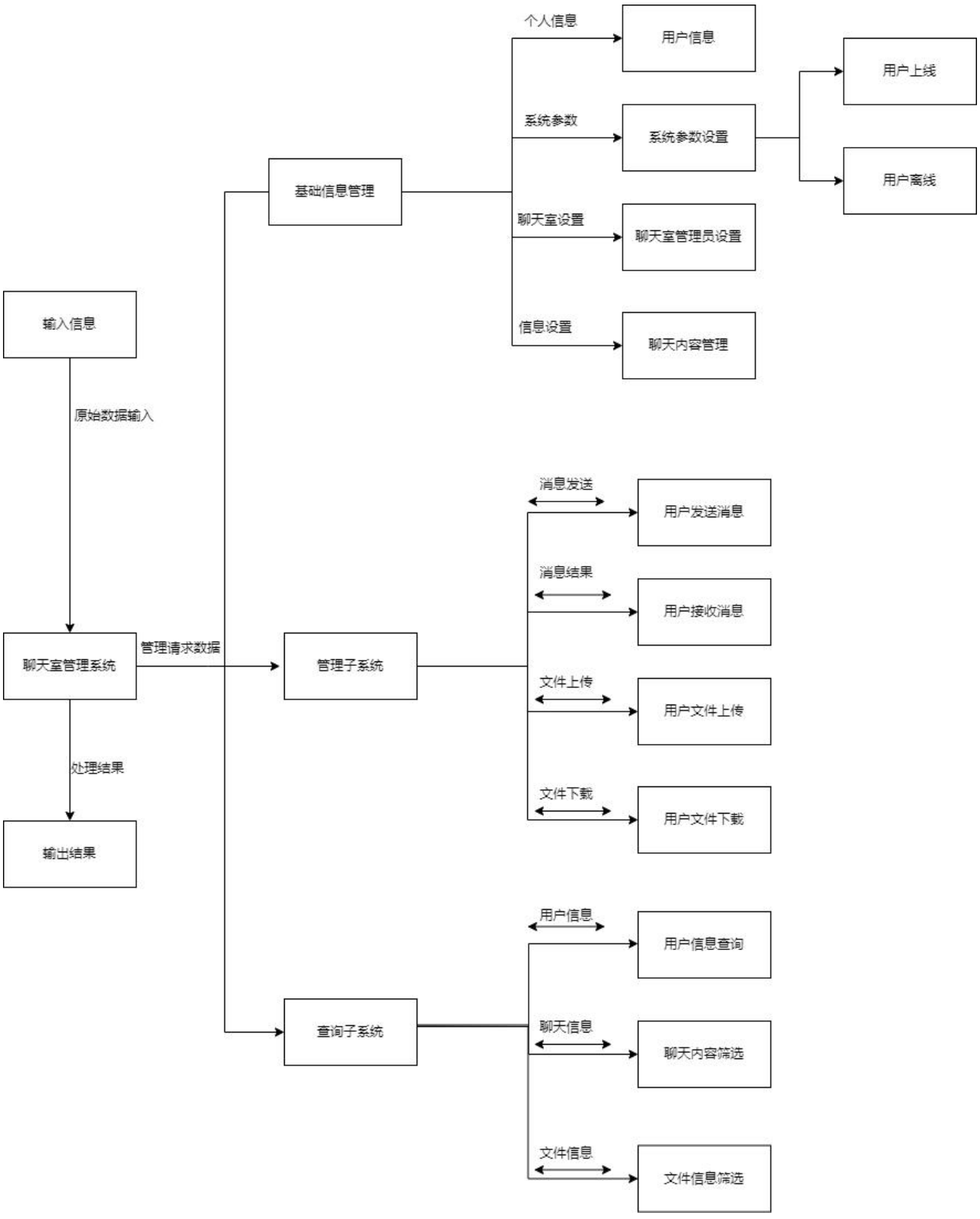


交流主要逻辑流程图



2.4. 结构

主要的功能模块分为四个分别是 broadcast.go,message.go,offline.go,user.go。broadcast.go 是广播模块，主要功能是对用户的一些操作进行广播，比如用户上线，用户断开连接等。message.go 模块定义和实现了消息的类型结构体和创建不同类型消息的方法。offline.go 模块实现了离线消息管理器，用来保存和发送离线消息。user.go 模块定义了 user 结构体，创建新用户以及接受消息和发送消息。


个人信息
用户信息
用户上线
系统参数
系统参数设置
用户离线
基础信息管理
聊天室设置
聊天室管理员设置
输入信息
信息设置
聊天内容管理
原始数据输入
消息发送
用户发送消息
消息结果
用户接收消息
聊天室管理系统
管理请求数据
管理子系统
文件上传
用户文件上传
处理结果
文件下载
用户文件下载
输出结果
用户信息
用户信息查询
查询子系统
聊天信息
聊天内容筛选
文件信息
文件信息筛选

2.5. 功能需求与系统模块的关系

[本条用一张矩阵图说明各项功能需求的实现同各模块的分配关系。]

	[broadcast]	[message]	[offline]	[user]
[用户的登入和等出]	✓		✓	✓
[用户共享和发送消息]	✓	✓		
[文件上传]	✓			✓
[文件下载]	✓			√

2.6. 人工处理过程

聊天室系统需要对用户发送的消息进行内容审核,以识别和屏蔽违规内容,这种内容审核通常需要我们人工参与。

2.7. 尚未解决的问题

1. 私人聊天功能
2. 创建群聊功能
3. 上传 zip 等形式的文件

3. 接口设计

3.1. 用户接口

系统需要提供方便、易懂、易操作的用户界面，可以进入聊天室，编写昵称，之后编写消息（文本、表情）。主屏幕看到其他用户发的消息，系统反应迅速，用户的交流无延迟。

3.2. 内部接口

系统提供的可供用户上传 txt、pdf 文件的接口，并在右下角展示，以及用户可以下载上传至系统的文件。通过前端和后端的双重过滤，限制文件传输的类型，在传输文件的同时确保安全性。

3.3. 外部接口

系统可以通过电脑的外接口，接收到来自外界（usb 等）的文件。

4. 运行设计

4.1. 运行模块组合

4.1.1 服务器端模块

（1）连接管理模块

- **功能**：负责处理客户端的连接请求，建立并维护客户端与服务器之间的连接状态。
- **技术实现**：使用 TCP/UDP 套接字进行网络连接，通过监听特定端口接收客户端的连接请求，并管理连接池中的连接状态。

（2）消息处理模块

- **功能**：接收客户端发送的消息，进行解析、处理和转发。
- **技术实现**：解析消息格式（如 JSON、XML 等），根据消息类型（如文本消息、图片消息、系统通知等）调用相应的处理函数，并将处理结果或转发消息发送给目标客户端。

（3）用户管理模块

- **功能**：管理用户的验证、状态更新（如在线/离线）和权限控制。
- **技术实现**：使用数据库或内存数据结构存储用户信息，通过身份验证机制确保用户身份的真实性，并根据用户权限控制其访问和操作

（4）聊天室管理模块

- **功能**：创建、维护和管理多个聊天室，包括聊天室的基本信息（如名称、描述、成员列表等）和动态信息（如消息历史、文件共享、时间等）。
- **技术实现**：使用数据库或内存数据结构存储聊天室信息，提供创建聊天室、加入/退出聊天室、设置聊天室权限等功能的 API 接口。

4.1.2 客户端模块

（1） 消息发送模块

- **功能**：提供用户输入消息并发送给服务器的功能。
- **技术实现**：接收用户输入的文本、表情包等信息，封装成特定格式的消息包，通过连接模块发送给服务器。

（2） 消息接收模块

- **功能**：接收服务器转发的消息，并展示给用户。
- **技术实现**：监听来自服务器的消息推送，解析消息包，将解析后的消息内容展示在用户界面上（如聊天窗口、通知栏等）。

（3）用户界面模块

- **功能**：提供用户与聊天室交互的界面。
- **技术实现**：设计并实现用户界面，包括登录登出页面、聊天室列表页面、聊天页面等，确保用户能够方便地浏览和操作聊天室内容。

（4）文件上传和下载模块

- **功能**：提供用户上传和下载文件。
- **技术实现**：前端文件上传只允许客户上传 txt 和 pdf 文件，文件下载通过超链接将服务器上的文件下载到本地。

4.1.3 辅助模块

（1）安全模块

- **功能**：保障聊天室的数据安全和用户隐私。
- **技术实现**：采用加密技术（如 jwt）对传输的数据进行加密，实施身份验证和授权机制，防止恶意攻击和数据泄露。

（2） 日志模块

- **功能**：记录聊天室的运行日志，便于问题排查和性能优化。
- **技术实现**：将关键操作（如用户登录、消息发送/接收等）记录到日志文件中，提供日志查询和导出功能。

（3） 通知模块

- **功能**：向用户发送系统通知和聊天室内的消息提醒。
- **技术实现**：通过弹窗、声音、震动等方式向用户发送通知，确保用户能够及时接收到重要信息。

4.2. 运行控制

4.2.1. 聊天室创建与删除

方法：

- **管理员操作**：通过管理界面或 API 接口进行。

操作步骤：

1. 登录到聊天室管理后台。
2. 进入聊天室管理页面，找到“创建聊天室”的选项。
3. 填写聊天室的名称、描述、权限设置等信息，提交创建请求。
4. 聊天室创建成功后，可以在列表中看到新创建的聊天室，并进行进一步的管理。
5. 如需删除聊天室，选中目标聊天室，点击“删除”按钮，确认删除操作。

4.2.2. 消息发送

方法：

- **实时消息传输**：用户发送的消息通过 WebSocket 等实时通信技术传输到服务器。

操作步骤：

1. 用户在聊天室输入框中输入消息并发送。
2. 消息通过实时通信技术传输到服务器。
3. 消息被广播到聊天室中，所有成员都能看到。

4.2.3 聊天室活跃度监控

方法：

- **统计在线人数和消息量：**通过聊天室 API 接口获取实时在线人数和消息量数据。

操作步骤：

1. 定期（如每分钟）调用聊天室 API 接口获取在线人数和消息量数据。
2. 将数据存储在数据库或内存中，以便后续分析和展示。
3. 根据数据分析结果，评估聊天室活跃度，并采取相应的运营策略。

5. 系统数据结构设计

5.1. 逻辑结构设计要点

5.1.1. 用户表

用户（**用户 ID**，姓名，性别，联系电话，登录时间，离开时间，在线时间）

5.1.2. 聊天记录表

聊天记录（记录 ID，**用户 ID**，聊天内容，时间戳）

5.1.3. 聊天室表

聊天室（**聊天室 ID**，聊天室名称，创建时间，**管理员 ID**）

5.1.4. 用户聊天室关系表

用户聊天室关系（关系 ID，**用户 ID**，聊天室 ID，加入时间，离开时间）

5.1.5. 消息表

消息（消息 ID，发送者 ID，聊天室 ID，消息内容，发送时间）

5.1.6. 文件表

文件（**文件 ID**，上传者 ID，聊天室 ID，文件名称，文件路径，上传时间）

5.1.7. 好友关系表

好友关系（关系 ID，**用户 ID1**，用户 ID2，建立时间）

5.1.8. 用户设置表

用户设置（**用户 ID**，设置项，设置值）

5.1.9. 群组表

群组（**群组 ID**，群组名称，创建时间，管理员 ID）

5.2. 物理结构设计要点

一、用户表

用户 ID int(11) NO PRI auto_increment
姓名 varchar(50) NO
性别 varchar(10) YES
联系电话 varchar(20) YES
登录时间 datetime YES
离开时间 datetime YES
在线时间 time YES

二、聊天记录表

记录 ID int(11) NO PRI auto_increment
用户 ID int(11) YES MUL
聊天内容 text NO
时间戳 datetime NO

三、聊天室表

聊天室 ID int(11) NO PRI auto_increment
聊天室名称 varchar(100) NO
创建时间 datetime NO
管理员 ID int(11) YES MUL

四、用户聊天室关系表

关系 ID int(11) NO PRI auto_increment
用户 ID int(11) YES MUL
聊天室 ID int(11) YES MUL
加入时间 datetime NO
离开时间 datetime YES

五、消息表

消息 ID int(11) NO PRI auto_increment
发送者 ID int(11) YES MUL
聊天室 ID int(11) YES MUL

消息内容 text NO
发送时间 datetime NO

六、文件表

文件 ID int(11) NO PRI auto_increment
上传者 ID int(11) YES MUL
聊天室 ID int(11) YES MUL
文件名称 varchar(255) NO
文件路径 varchar(255) NO
上传时间 datetime NO

七、好友关系表

关系 ID int(11) NO PRI auto_increment
用户 ID1 int(11) YES MUL
用户 ID2 int(11) YES MUL
建立时间 datetime NO

八、用户设置表

用户 ID int(11) NO PRI
设置项 varchar(50) NO PRI
设置值 varchar(255) NO

九、群组表

群组 ID int(11) NO PRI auto_increment
群组名称 varchar(100) NO
创建时间 datetime NO
管理员 ID int(11) YES MUL

5.3. 数据结构与程序的关系

数据结构为关系型数据库，所以，在程序中可以通过标准的 SQL 语句与数据结构进行交互，交互过程中采用通用的数据访问接口。为了保持良好的程序架构，对数据库访问采用 DAO 设计模式实现，提高维护性以及扩展性。

[说明各个数据结构与访问这些数据结构的各个程序之间的对应关系。]

6. 系统出错处理设计

6.1. 出错信息

当出现错误时，系统会自动检测并记录错误信息，并尝试自动修复。如果无法自动修复，系统会向管理员发送通知，并提供错误码和建议的解决方案。同时，为了提高用户体验，系统还可以在界面上显示的错误提示，帮助用户快速解决问题。

6.2. 补救措施

首先，建立一个备份系统，确保在出现严重错误时能够快速切换到备份系统，保证聊天室的正常运行。

其次，系统设置有回滚机制，当出现严重错误时，可以将系统恢复到之前的稳定状态。另外，建议定期进行系统检测和更新，及时发现并修复潜在的错误，避免出现大规模的系统故障。

最后，本系统建立有用户反馈系统，让用户可以及时报告错误，并根据用户反馈改进系统，提高系统的稳定性和可靠性。这些补救措施能够有效地减少系统出错的影响，保障聊天室系统的稳定运行。

6.3. 系统维护设计

我们会定期进行维护。在维护期间，会向用户提前通知维护时间，并提供联系方式，让用户可以及时反馈问题。维护结束后，会发布维护结果和更新内容，让用户可以及时了解系统的最新情况。另外，我们会定期备份系统数据，防止因维护过程中出现意外导致数据丢失。通过合理的维护设计，可以保证系统的稳定运行，提高用户满意度。