汕頭大學
SHANTOU UNIVERSITY

网络工程项目·即时通讯系统
个人报告

院（系）名 称：数学与计算机学院

专 业 名 称：计算机科学与技术

姓　　名：林泓宇 2022631024

汕头大学

2025 年 8 月 31 日

1 实践目的

本实验目标是设计并实现一个 SRT 字幕解析器，主要功能包括：解析字幕文件，提取每一条字幕的时间戳、字幕内 容，并将这些信息存储到内存数据结构中。同时，还支持对字幕内容进行平移，并更新字幕文件。

2 实践内容

· 解析 SRT 字幕文件：分析字幕文件中的每条字幕，包括字幕的起止时间和对应的文本内容。

· 词法分析器与语法分析器：根据字幕内容定义相应的文法规则，实现字幕内容的词法和语法分析。

· 内存数据结构设计：设计适合的内存数据结构存储解析出的字幕信息。

· 字幕平移功能：支持字幕的平移操作，如将每条字幕的时间推迟 2 秒。

· 更新字幕文件：实现字幕内容的修改并保存为新的字幕文件。

3 实践过程

3.1 调试环境

· 编译器：Trae IDE

· 编程语言：Python、HTML、CSS、JavaScript 等

3.2 设计过程

本项目作为一个实时聊天应用，其设计理念和实施方案旨在提供一个高性能、高可用且易于扩展的通信平台。我们深入考量了现代 Web 应用的需求，将整个系统划分为清晰的前后端架构，并在此基础上融合了多种前沿技术，以确保最终产品的卓越表现。

3.2.1 前端设计方案

前端部分作为用户与系统交互的门户，其设计核心在于提供流畅、直观且响应迅速的用户体验。我们选择将前端代码库放置于 frontend 目录下，并采纳了行业领先的 Vue.js 框架，结合 TypeScript 进行开发。Vue.js 的组件化开发模式极大地提高了代码的复用性和可维护性，而 TypeScript 则在编译阶段就提供了强大的类型检查，有效减少了运行时错误，提升了开发效率和代码质量。

在项目结构上，我们进行了精心规划。src/vue/pages 目录汇集了所有顶层页面组件，例如 UserProfile.vue，每个组件都独立封装了特定页面的 UI 和交互逻辑。这种模块化的设计使得页面开发更为聚焦，便于团队成员并行开发。前端路由管理通过 src/ts/instances/routerInstance.ts 实现，其中利用 Vue Router 定义了所有应用路由，并实现了声明式导航、嵌套路由、路由守卫等高级功能，确保了用户在应用内部的无缝跳转和状态管理。

src/ts 目录下包含了一系列通用工具类和实例，如 API 服务封装、设备信息检测、消息格式化工具等，这些通用模块遵循单一职责原则，为前端业务逻辑提供了坚实的基础支撑。

构建系统方面，我们选用了当下热门的 Vite 作为开发和生产构建工具。Vite 以其闪电般的冷启动速度和按需编译能力，极大地提升了开发体验。同时，它集成了 Rollup 进行生产环境打包优化，实现了代码分割、Tree Shaking 等高级优化，确保了最终部署包的体积最小化和加载速度最大化。考虑到安全传输的重

要性，构建流程还包含了自动化 SSL 证书生成和管理，以及静态资源的优化处理，为应用提供了 HTTPS 支持和高效资源加载。

3.2.1 后端设计方案

后端系统作为整个应用的数据大脑和业务逻辑核心，位于 backend 目录下，并基于稳健且功能丰富的 Python Django 框架构建。Django 以其“约定优于配置”的原则和完善的生态系统，为快速开发和安全部署提供了强大的保障。

chat 目录是 Django 核心应用所在，其中 models.py 定义了应用的所有数据模型，如用户、消息、会话等，通过 Django ORM 提供了强大的数据库抽象层，使得数据操作更为简洁和安全。urls.py 负责定义 API 接口的路由，将不同的请求映射到相应的视图函数。settings.py 则集中管理着数据库连接、安全密钥、中间件配置等所有环境相关的设置，支持不同部署环境下的灵活配置。

为了满足聊天应用的实时通信需求，我们创新性地引入了高性能的异步 Web 服务器 Tornado，并将其集成到 Django 项目中，具体实现位于 backend/chat/tornado 目录下。Tornado 以其非阻塞 I/O 和事件驱动的特性，非常适合处理大量的并发 WebSocket 连接。message_handler.py 负责处理所有接收到的 WebSocket 消息，进行解析、验证和分发，确保消息能够准确无误地送达目标客户端。http_handler.py 则作为 Tornado 的 HTTP 入口，协同 Django 处理部分轻量级的 HTTP 请求，或者作为 WebSocket 连接的握手端点。这种 Django 与 Tornado 的混合架构，既利用了 Django 在传统 Web 开发上的优势，又发挥了 Tornado 在实时通信上的专长，实现了性能的最优化。

数据库层面，我们选择了 SQLite 作为开发环境的轻量级数据库，其易于部署和管理的特性加速了开发迭代周期。在生产环境中，可平滑迁移至 PostgreSQL 或 MySQL 等更强大的关系型数据库。migrations 目录则通过 Django 的迁移系统，版本化管理着数据库模式的每次变更，确保了数据库结构与模型定义的一致性。

为了进一步提升系统的响应速度和处理高并发消息的能力，我们深度集成了 Redis 作为缓存系统和消息队列。global_redis.py 模块封装了与 Redis 交互的逻辑。Redis 在消息分发中扮演了关键角色，新消息首先写入 Redis，再由 Tornado 或其他后台任务从 Redis 中获取并推送给订阅的客户端，这有效解耦了消息生产者和消费者，减轻数据库的 I/O 压力，并提高了消息处理的吞吐量。

最后，我们严格遵循了 Python 项目的最佳实践，通过 venv38 目录下的 Python 虚拟环境来管理项目的所有依赖。这确保了项目在一个独立且干净的环境中运行，避免了不同项目之间的依赖冲突，并提供了环境的可重现性。

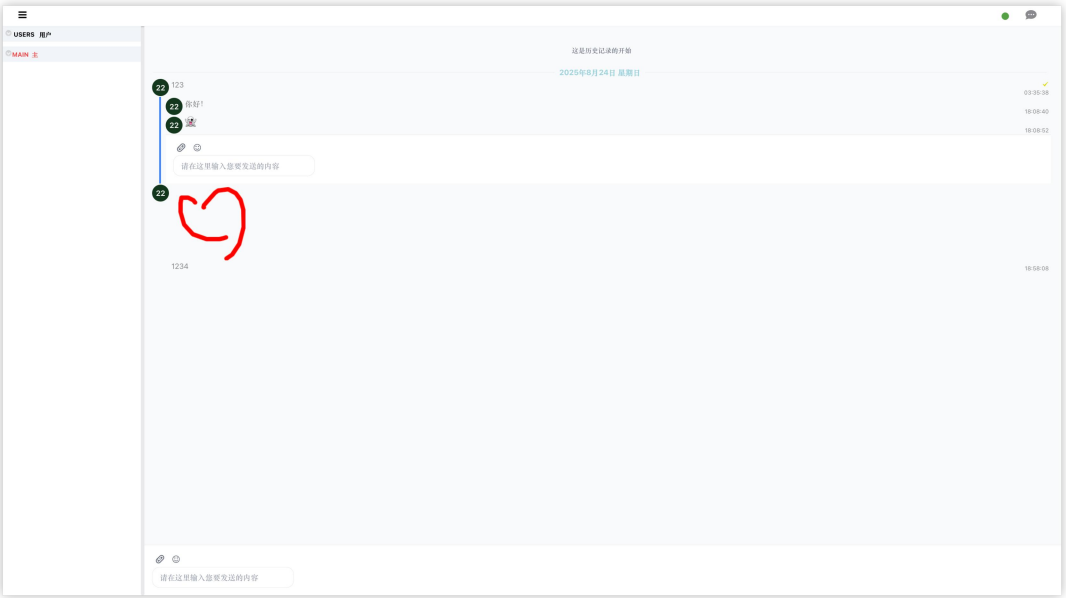
4 重点界面展示

4.1 首页

4.2 注册和登录

4.3 主页面（USER+MAIN 界面）

USER 用于用户一对一私聊，MAIN 是主群组（均不可删除）

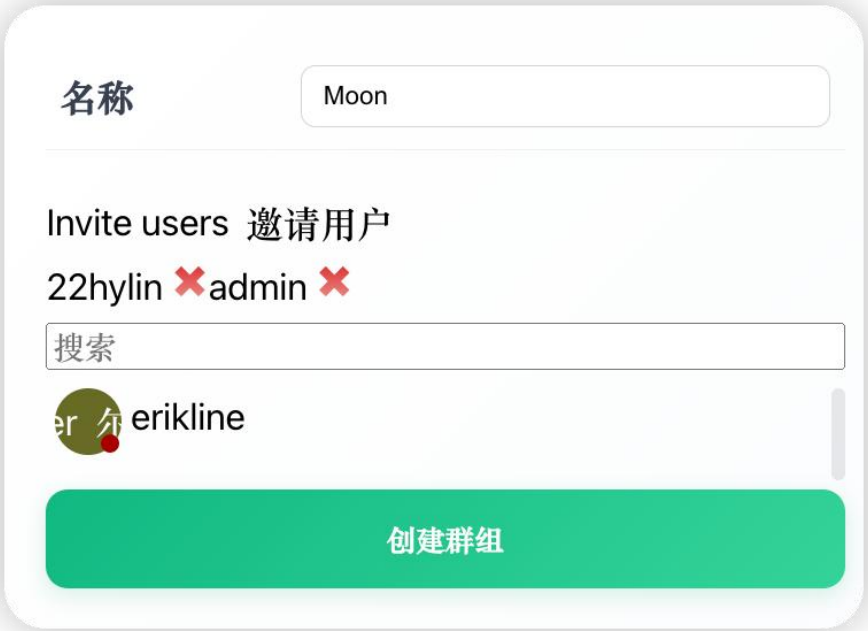


4.4 左右侧边栏

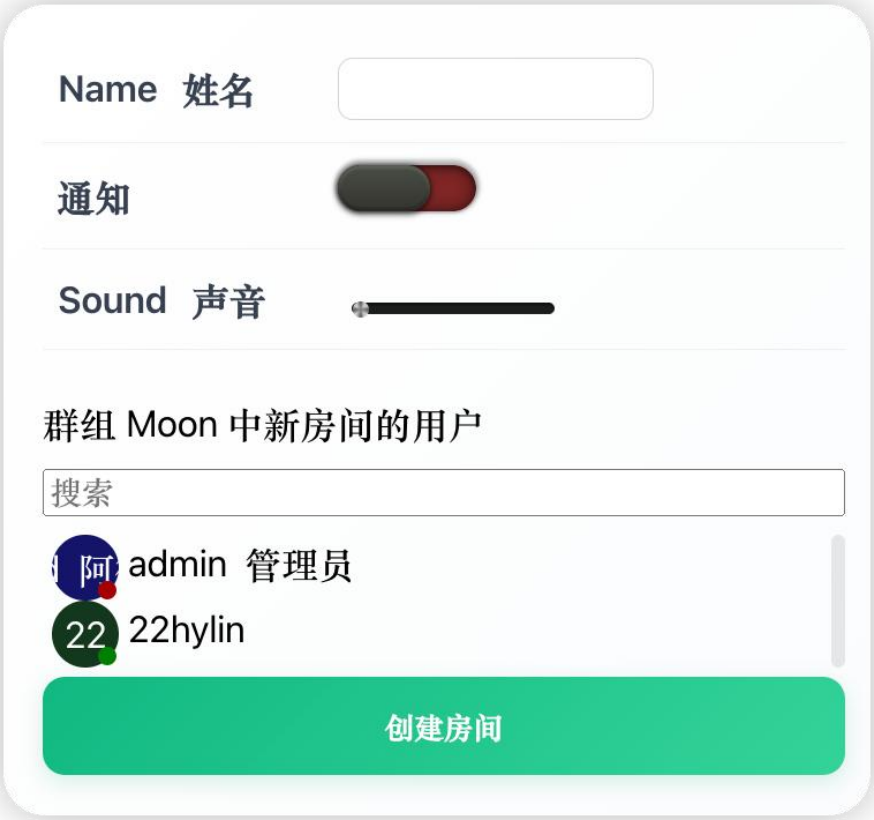
4.5 创建私聊用户频道/创建新群组频道/在某群组创建新房间

4.5.1 创建私聊用户频道

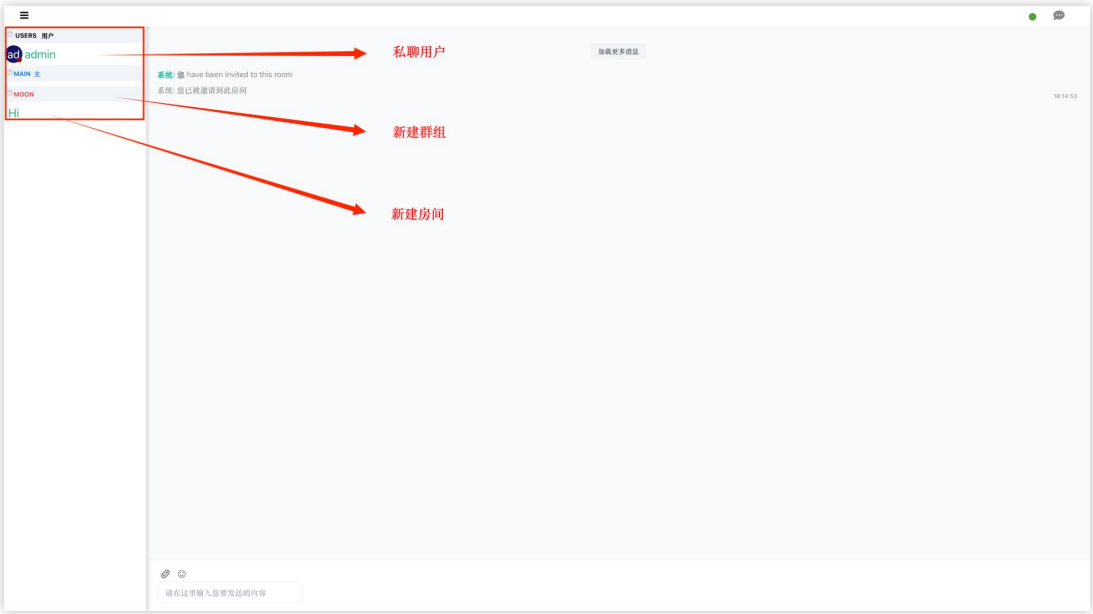
4.5.2 创建新群组频道



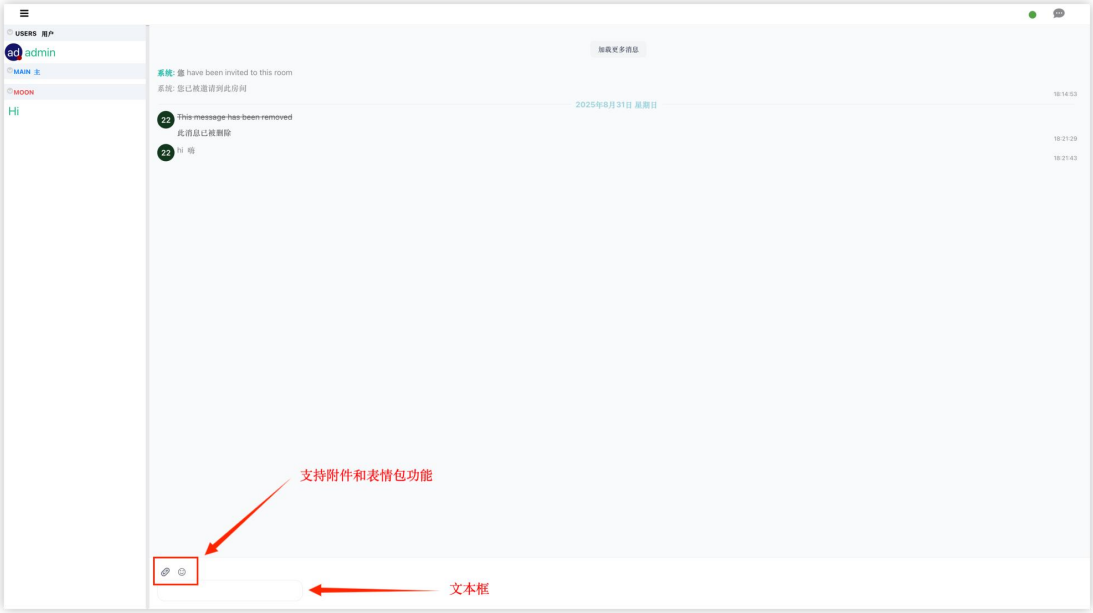
4.5.3 在某群组创建新房间



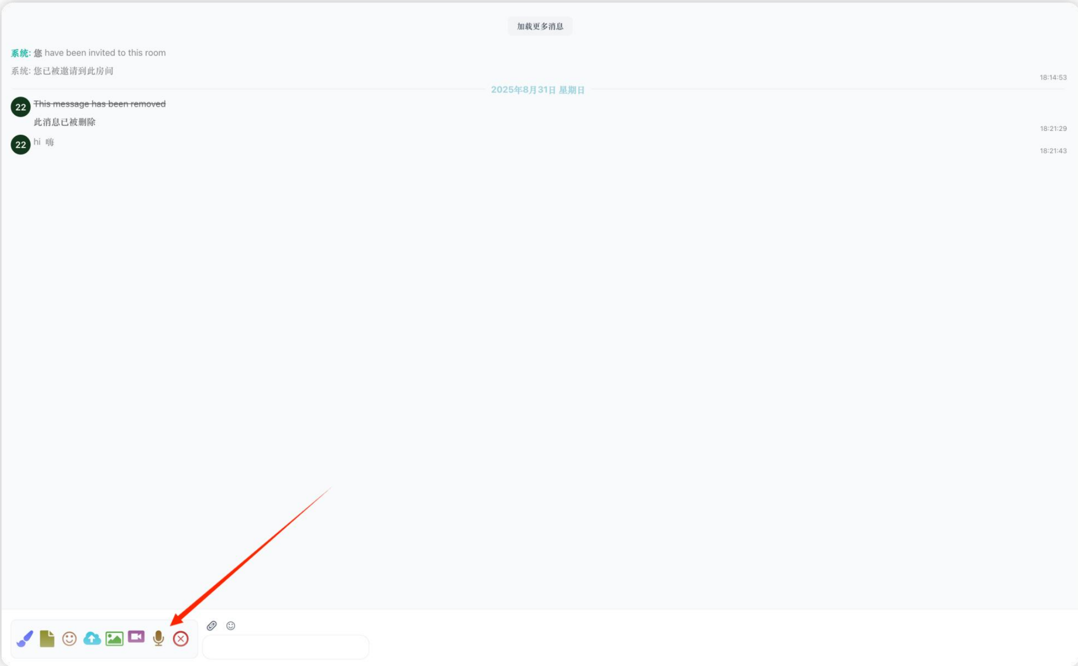
4.5.4 主页面显示

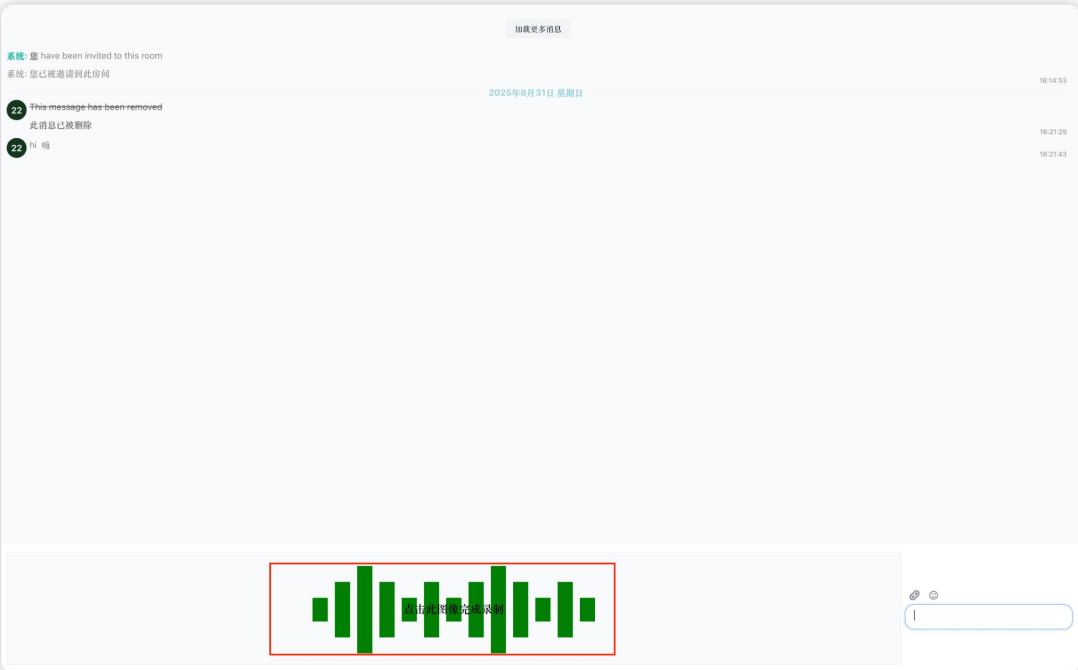


4.6 文本对话功能



4.7 语音对话功能（在附件标志按键中）

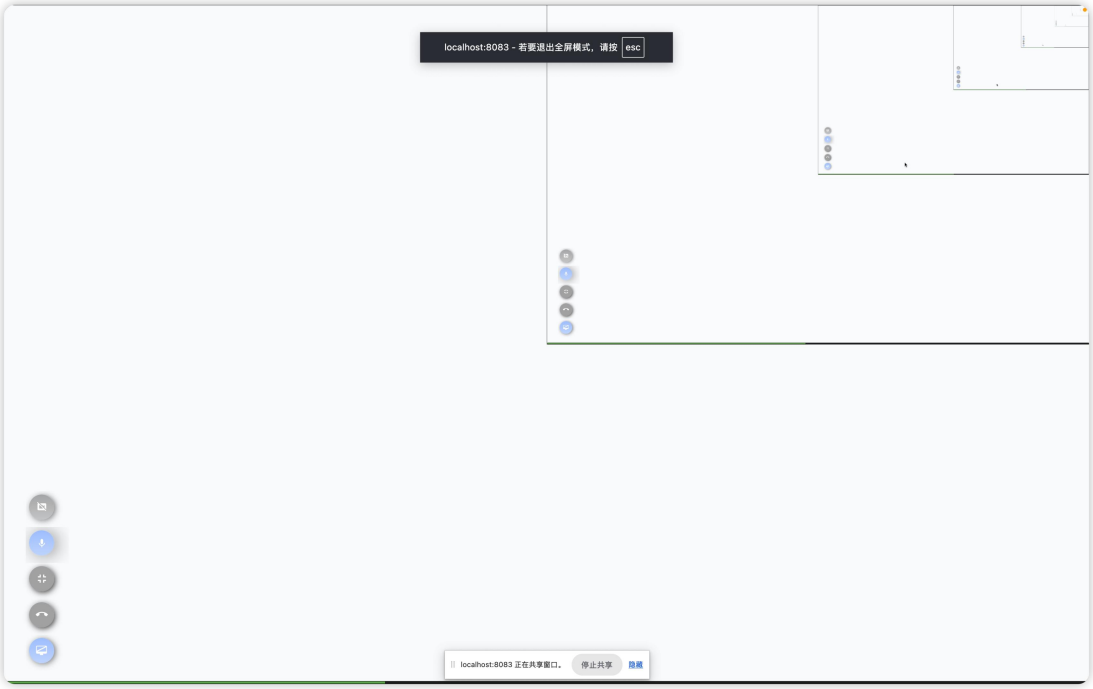




4.8 音视频协同对话功能（在通话标志中进入）

4.8.1 功能简述

4.8.2 窗口放大效果展示



5 心得感想

在本次网络即时通讯系统应用的开发过程中，我深入体会到了前后端分离架构在提升开发效率和系统可维护性方面的优势。前端基于 Vue.js 与 TypeScript 构建，结合 Vite 实现了高效的模块化开发与构建优化，让我认识到现代前端工程化工具的重要性；后端则采用 Django 处理用户认证与业务逻辑，同时借助 Tornado 的异步 I/O 能力实现 WebSocket 实时通信，并通过 Redis 缓存和消息队列提升了系统在高并发场景下的响应性能。在实际开发中，我逐渐理解到 RESTful API 与 WebSocket 在数据交互中各自适用的场景，也体会到存储与缓存相结合的设计理念对系统稳定性和性能的保障作用。整个项目让我不仅掌握了多种技术的组合应用，更培养了架构思维与工程化意识，对我后续的学习与科研工作具有重要的启发意义。

2022631024 林泓宇