课程：专业方向综合项目

指导老师：唐剑锋

小组成员：

2253551 李沅衡

2250763 李俊旻

2254272 赵子毅

2251760 黄志栋

2251225 王铭乾

完成日期：2025年3月3日

组队文档

Social That Gets You

目录

[1.项目背景 3](#_Toc6101)

[2.项目范围 3](#_Toc31754)

[3.项目功能 3](#_Toc28067)

[4.逻辑架构 5](#_Toc13084)

[4.1 逻辑架构图 5](#_Toc29966)

[4.2 架构层次(Architecture Layers) 6](#_Toc5267)

[4.3 技术特点 (Technical Features) 8](#_Toc16871)

[5.候选开发技术栈 8](#_Toc10801)

[5.1 后端技术栈 8](#_Toc8659)

[5.2 前端技术栈 9](#_Toc22441)

[5.3 数据库技术栈 10](#_Toc7036)

[6.涉及到的中间件 10](#_Toc10237)

[7.涉及到的平台和工具 10](#_Toc14052)

[7.1 开发工具 10](#_Toc19351)

[7.2 测试与部署 11](#_Toc27800)

[7.3 监控与运维 11](#_Toc22354)

[7.4 AI相关平台和工具 11](#_Toc12189)

[8.项目管理和系统测试方面的设想 11](#_Toc11354)

[8.1 项目管理 11](#_Toc30363)

[8.2 系统测试 12](#_Toc29445)

[9.AI部分的设想 13](#_Toc12734)

[9.1 ai对话系统（智能聊天助手） 13](#_Toc6683)

[9.2 智能搜索 14](#_Toc28802)

[9.3 智能推荐 14](#_Toc8046)

[10.团队分工 14](#_Toc23100)

[11.项目日程详细安排 15](#_Toc18796)

[13.附加部分 15](#_Toc21749)

[13.1 参考文献与数字资源 15](#_Toc5806)

[13.2 相关课程 16](#_Toc27668)

[13.3 相关知识 16](#_Toc8537)

[13.4 项目挑战 16](#_Toc9334)

[13.5 课程建议 17](#_Toc10785)

## 1.项目背景

随着数字化转型的不断推进，各行各业对于智能化和高效管理的需求日益增加。特别是在互联网技术、人工智能、大数据和云计算等领域的飞速发展，带来了新的商业机会和技术挑战。在这种背景下，构建一个集成化、智能化的系统平台，已成为许多企业和机构的核心需求。通过集成前端应用、后端服务、AI技术和微服务架构，企业能够提升其运营效率，优化客户体验，同时满足用户多元化的需求。

本项目旨在开发一套全面、智能的系统平台，提供集成的社交互动、内容管理、AI推荐、智能搜索以及活动管理等功能，旨在为用户和企业提供更为智能化、高效的服务。通过本平台，用户不仅能够获得个性化内容推荐和高效信息检索，还能享受到与其他用户的互动和丰富的社交体验。

## 2.项目范围

本项目的主要目标是开发一个集成多种功能的智能平台。平台将由前端、后端、AI模块、数据库以及服务治理等多个部分构成。前端部分采用响应式设计，确保用户无论使用移动设备还是桌面设备，都能获得良好的操作体验；后端则通过微服务架构和分布式数据库来支撑系统的高并发和高可用性；AI模块将实现智能推荐、搜索和对话机器人等功能，提升用户体验；服务治理部分则关注系统的稳定性、流量管理和安全性，确保在高负载下系统能够稳定运行。此外，平台还将支持对接外部API，方便集成其他系统或服务。

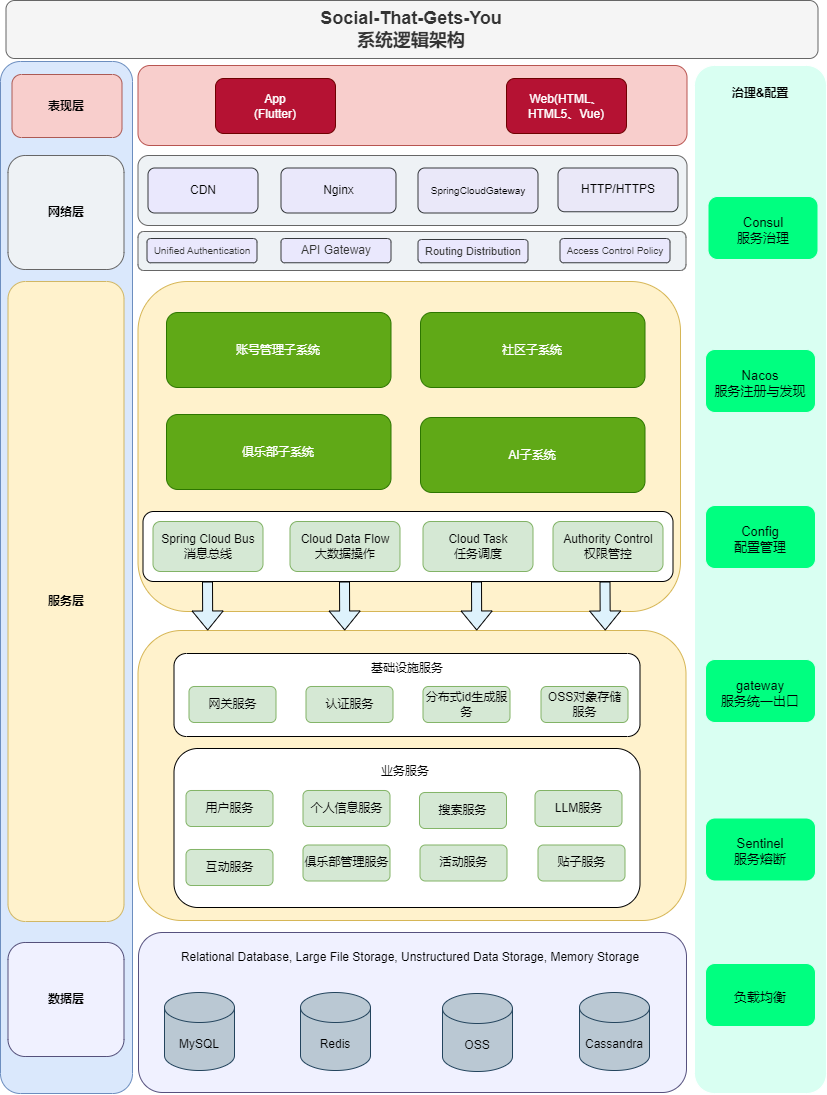
## 3.项目功能

项目将具备以下主要功能：

1. **用户管理系统**：用户注册、登录、信息维护、权限管理等。
2. **社交功能模块**：包括用户之间的互动、评论、点赞等社交功能。
3. **内容管理**：用户发布的帖子、图片、视频等内容的管理、搜索和展示功能。
4. **智能推荐系统**：基于用户行为数据和AI算法，为用户推荐个性化内容。
5. **智能搜索功能**：集成智能搜索引擎，提供高效、准确的内容搜索服务。
6. **活动管理系统**：用户可以参与各种线上活动，支持活动创建、报名和管理。
7. **数据存储与管理**：通过数据库和对象存储系统管理用户数据和大文件。
8. **高可用性服务**：通过微服务架构、负载均衡和服务治理，确保系统的稳定运行。
9. **AI对话系统**：集成智能聊天机器人，为用户提供便捷的问答和互动体验。

## 4.逻辑架构

### 4.1 逻辑架构图

****

### 4.2 架构层次(Architecture Layers)

#### 4.2.1 表现层 (Presentation Layer)

* **客户端 (Client)**
* **应用 (App, Flutter)**：考虑使用 Flutter 框架开发移动端应用，提供用户友好的界面和跨平台支持。
* **网页 (Web, HTML5, Vue)**：基于 HTML5 和 Vue.js 构建的网页前端，支持桌面和移动浏览器的访问，提供动态交互体验。

#### 4.2.2 网络层(Network Layer)

* **CDN**：内容分发网络，用于加速静态资源（如图片、视频）的分发，提升全球用户访问速度。
* **Nginx**：高性能 Web 服务器和反向代理，用于处理 HTTP/HTTPS 请求，负载均衡和静态资源服务。
* **Spring Cloud Gateway**：微服务网关，负责请求路由、API 聚合和协议转换。
* **HTTP/HTTPS**：采用安全加密协议，确保数据传输的安全性。

#### 4.2.3 接入层 (Access Layer)

* **Unified Authentication (统一认证)**：提供单点登录（SSO）和用户身份验证功能，确保用户访问的安全性。
* **API Gateway (API 网关)**：集中管理外部和内部 API 请求，包含路由分发、限流、认证等功能。
* **Routing Distribution (路由分发)**：动态路由策略，将请求分发到合适的微服务实例。
* **Access Control Policy (访问控制策略)**：定义和执行权限管理，保障系统安全性。

#### 4.2.4 服务层 (Service Layer)

* **基础设施服务**: 提供系统运行所需的基础设施支持。
* **分布式Id生i成服务**: 分布式服务，支持系统的扩展和高可用性。
* **OSS对象存储服务**: 提供对象存储服务，用于存储和管理大文件。
* **业务处理服务**
* **用户服务**: 管理用户相关的信息和服务。
* **个人信息服务**: 提供用户个人信息的维护和查询功能。
* **搜索服务**: 提供系统内的搜索功能。
* **LLM服务**: 提供与大型语言模型相关的服务。
* **互动服务**: 管理用户之间的互动功能。
* **俱乐部管理服务**: 提供俱乐部相关的管理功能。
* **活动服务**: 管理系统的活动功能。
* **贴子服务**: 管理用户发布的贴子内容。
* **服务治理 (Service Governance)**
* **Spring Cloud Bus (消息总线)**：用于服务间的事件广播和配置刷新。
* **Cloud Data Flow (云数据流)**：处理大规模数据流转和实时数据处理。
* **Cloud Task (云任务)**：管理定时任务和批量任务的执行。
* **Authority Control (权限控制)**：细粒度权限管理，确保服务间的安全访问。

#### 4.2.5 分布式服务治理 (Distributed Service Governance)

* **Consul (服务注册与发现)**：提供服务注册、发现和健康检查功能，支持分布式系统的动态扩展。
* **Nacos (配置与服务管理)**：集中式配置管理和服务注册中心，支持动态配置和服务的灰度发布。
* **Config (配置管理)**：统一管理系统配置，支撑多环境下的配置切换。
* **gateway (网关扩展)**：扩展网关功能，支持更复杂的流量管理和安全策略。
* **Sentinel (流量控制)**：提供流量控制、熔断降级和系统保护，保障服务稳定性。

#### 4.2.6 数据层 (Data Layer)

数据层负责系统的数据存储和管理，支持多种数据存储方式。该层包括以下组件

* **关系型数据库 (Relational Database, MySQL)**：存储结构化数据，如用户数据和交易记录。
* **内存存储 (Memory Storage, Redis)**：用于高性能缓存和会话管理。
* **大文件存储 (Large File Storage, OSS)**：存储非结构化数据，如图片、视频等。
* **非结构化数据存储 (Unstructured Data Storage, Cassandra)**：处理大规模非结构化数据，支持高并发读写。

#### 4.2.7 基础设施 (Infrastructure)

* 提供底层支持，包括服务器、云服务和网络资源，确保系统的稳定性和可扩展性。

### 4.3 技术特点 (Technical Features)

* **高可用性 (High Availability)**：通过负载均衡、服务治理和分布式架构，确保系统在高并发场景下的稳定运行。
* **可扩展性 (Scalability)**：支持微服务架构和分布式存储，方便根据业务需求扩展服务和资源。
* **安全性 (Security)**：采用统一认证、访问控制和加密协议，保护用户数据和系统安全。
* **智能化 (Intelligence)**：集成 AI 服务，提升推荐精度和用户体验。

## 5.候选开发技术栈

### 5.1 后端技术栈

* **编程语言**：Java17
* **Web框架**：Spring Boot，后端开发的核心框架选择了 **Spring Boot 3.x**。它通过自动配置、内嵌服务器和丰富的 Starter 组件大幅提升了开发效率。Spring Boot 3.x 还支持响应式编程（WebFlux），适合实时动态刷新等高并发场景，并与 Spring Cloud、Spring Security 无缝集成，提供微服务、安全和数据访问的全栈支持。
* **微服务框架**：Spring Cloud Alibaba，它集成了 Nacos（服务注册与发现、动态配置）、Sentinel（流量控制、熔断降级）和 Seata（分布式事务）等组件，提供高可用性和高并发支持。相比 Spring Cloud Netflix 的 Eureka 和 Hystrix（后者已停止维护），Spring Cloud Alibaba 的组件更现代化且功能更强。
* **API网关**：Spring Cloud Gateway，基于 Netty 的异步非阻塞架构，性能远超传统阻塞式网关（如 Zuul），支持每秒数万请求，提供了动态路由、请求过滤（如认证、限流）和路径重写等功能。它与 Spring Cloud 无缝集成，支持服务发现和负载均衡，并允许开发者编写自定义过滤器实现复杂逻辑。
* **服务间通信**：OpenFeign（简化HTTP调用）
* **认证与授权**：Sa-Token（轻量级，支持JWT和RBAC）
* **数据访问**：MyBatis-Plus（简化数据库操作，支持代码生成）
* **缓存**：Redis，Redis 是一款高性能的内存键值存储，支持每秒数十万次读写操作，适合缓存热门帖子、用户信息等。它提供多种数据结构（如哈希、列表、有序集合），可用于实现排行榜或消息队列，并通过主从复制和 Sentinel 模式确保高可用性。Redis Cluster 支持线性扩展，能应对流量激增，且其过期策略（TTL）可自动清理数据，减少内存占用。
* **消息队列**：RocketMQ，对于异步处理，我们选用了 **RocketMQ** 作为消息队列。它单机支持每秒数十万消息，适合处理发帖、评论等操作，并通过事务消息功能确保消息投递和业务操作的一致性（如发帖后积分增加失败可回滚）。RocketMQ 还支持顺序消息和延迟消息，适用于日志处理或定时通知。
* **分布式ID**：Leaf（美团分布式ID生成器，支持雪花算法）
* **对象存储**：Minio（开源、兼容S3 API）
* **键值存储**：Cassandra（高可用，适合大规模数据）
* **搜索与分析**：Elasticsearch（全文搜索、支持复杂查询）
* **AI功能**：
* **聊天机器人**：Spring AI（集成OpenAI API或其他大语言模型服务）
* **智能搜索**：Elasticsearch ML（机器学习插件）或自定义搜索模型
* **智能推荐**：Mahout（推荐算法库）或TensorFlow Serving（部署推荐模型）

### 5.2 前端技术栈

* **框架**：Vue.js 3（响应式、高性能、易维护）
* **UI组件库**：Element Plus（与Vue 3兼容，提供丰富组件）
* **状态管理**：Pinia（轻量级、类型安全）
* **路由**：Vue Router（官方路由管理）
* **HTTP客户端**：Axios（支持Promise，易于拦截请求）
* **打包工具**：Vite（快速构建，支持热更新）

### 5.3 数据库技术栈

* **关系型数据库**：MySQL 8.0（支持JSON和窗口函数）
* **NoSQL数据库**：MongoDB（文档存储，适合灵活schema）
* **图数据库**：Neo4j（用于社交关系查询）

## 6.涉及到的中间件

* **Nacos**：服务发现、配置管理和动态DNS
* **Sentinel**：流量控制、熔断降级和系统保护
* **Redis**：缓存、会话管理和分布式锁
* **RocketMQ**：消息队列，支持事件驱动架构
* **Zookeeper**：分布式协调服务（Cassandra依赖）
* **Elasticsearch**：全文搜索和日志分析
* **Logstash**：数据采集、处理和传输
* **Canal**：MySQL Binlog解析，实时数据同步
* **Minio**：对象存储服务
* **Cassandra**：分布式NoSQL数据库
* **Nginx**：反向代理和负载均衡（可选，用于网关前置）

## 7.涉及到的平台和工具

### 7.1 开发工具

* **IDE**：IntelliJ IDEA（Java开发）、VS Code（前端开发）
* **版本控制**：Git（代码管理）、GitLab/Gitee（私有仓库）
* **项目管理**：Maven（Java依赖管理）、NPM/Yarn（前端依赖管理）
* **API文档**：APIPost（接口文档生成）

### 7.2 测试与部署

* **单元测试**：JUnit 5（Java）
* **接口测试**：Postman、JMeter（性能测试）
* **容器化**：Docker（应用打包）、Docker Compose（本地环境）
* **CI/CD**：Jenkins、GitLab CI（自动化构建和部署）

### 7.3 监控与运维

* **应用监控**：Spring Boot Admin（微服务监控）
* **日志管理**：ELK Stack（Elasticsearch、Logstash、Kibana）
* **链路追踪**：SkyWalking（分布式链路追踪）
* **指标监控**：Prometheus + Grafana（时序数据监控）
* **告警系统**：Alertmanager（集成Prometheus）

### 7.4 AI相关平台和工具

* **模型训练**：TensorFlow、PyTorch（视需求选择）
* **模型部署**：TensorFlow Serving、TorchServe
* **自然语言处理**：Hugging Face Transformers（预训练模型）

## 8.项目管理和系统测试方面的设想

### 8.1 项目管理

1. **项目管理方法：Scrum**

Scrum 是一种常用的轻量级敏捷开发框架，整个项目可以划分为 3-4 个 Sprint，每个 Sprint 为期 2-3 周。具体过程如下：

* 每日站会：每天召开晨会（线下/在线），持续 10-15 分钟。汇报工作进展（完成了什么），遇到的问题，以及今天要做的事情。
* Sprint 规划会议：每个 Sprint 的开始阶段，明确当前 Sprint 的任务目标和工作分配（如完成某些模块开发或实现模版功能）。
* Sprint 回顾：一个 Sprint 结束后，总结开发中问题和经验，改进团队协作。

1. **项目分阶段规划**

见项目日程详细安排部分

1. **使用的管理工具**

* 任务管理工具：使用 GitHub 的项目管理功能，划分和分配任务。
* 版本控制工具：Git。定期合并分支，主分支（master）代码需通过测试。
* 文档管理工具：Notion、Typora、Office系列等，用于记录需求、设计文档、日常会议记录。
* 即时沟通工具：微信讨论组、飞书。

### 8.2 系统测试

**测试的分类与方法：**

1. 单元测试：

* 目标：验证各模块（如用户管理、内容发布等）的代码逻辑正确性。
* 工具：JUnit（针对 Java 后端）或 Jest（针对 Vue 前端）。
* 方案：
* 每个功能模块的逻辑单元都需编写测试用例，例如验证用户登录中的账号密码校验逻辑是否正确。
* 覆盖率需达到 90% 以上。

1. 集成测试：

* 目标：检验各模块之间是否能够正确交互。
* 工具：Postman （调用 API 测试后端接口）、Spring Test。
* 方案：
* 测试用户从前端提交请求，后端 API 返回数据和数据库交互的正确性。
* 检查 AI 推荐模块对其他服务调用的兼容性和响应时间。

1. 功能测试：

* 目标：确保所有功能在指定场景下按预期运行。
* 工具：Selenium（针对前端 UI 测试）。
* 方案：
* 模拟用户操作场景，如注册-登录-发布内容-评论-点赞。

1. 性能测试：

* 目标：验证系统在高并发、高负载条件下的表现。
* 工具：JMeter。
* 方案：
* 模拟多线程用户访问热门文章页面或搜索功能的请求。
* 目标性能：在 1000 用户并发访问时，响应时间低于 2 秒。

1. 安全测试：

* 目标：确保系统数据的安全性，防止常见漏洞。
* 方案：
* 测试越权访问（例如，低权限用户是否能访问管理员模块）。
* 测试 SQL 注入（通过特殊格式输入内容测试系统处理不当的情况）。

1. 用户验收测试 (UAT)：

* 目标：确保最小化产品能满足用户需求。
* 方法：邀请外部同学成为测试用户，进行试用并提交反馈。

**测试流程：**

1. 在每个开发阶段后的功能完成后，进行单元和集成测试。
2. 在基础功能开发完成后（第2阶段末），进行第一次功能整体测试。
3. 核心模块开发完成后开始性能测试与安全测试。
4. 项目结束时，邀请外部测试用户进行 UAT，收集改进意见。

## 9.AI部分的设想

### 9.1 ai对话系统（智能聊天助手）

1. **目标**：实现自然语言的交互，以及文本、图像等多模态对话，并实现多轮对话上下文，为用户提供良好互动体验，实现社交陪伴以及活动管理辅助。该机器人作为虚拟好友可基于用户社交动态生成个性化问候与互动，作为助手可根据用户输入，帮助用户完成帖子创作，也可完成基本的俱乐部和活动安排。
2. **技术栈**：

* **对话引擎**：文本生成可通过Spring AI调用现有大模型API，并通过HuggingFace Transformers（BERT微调）实现意识识别，NLTK Vader + 自研情感词典进行情感分析，并通过Neo4j存储用户兴趣图谱；图像交互通过tensorflow或pytorch等深度学习框架去训练对应的图像识别模型。
* **对话状态管理：**使用状态机管理多轮对话上下文，Redis存储对话Session（Key: user:123:dialog，Value: JSON结构化的历史记录）。

### 9.2 智能搜索

1. **目标**：根据用户搜索关键词或描述信息，匹配相关帖子和活动，同时结合用户画像优化搜索权重，并结合热点事件综合展示结果，同时支持自然语言查询的模糊匹配与意图识别，例如用户输入“周末能带孩子玩的地方”需解析为“亲子活动推荐”+“地理位置筛选”。
2. **技术栈**

* **搜索算法**：LightGBM（多特征融合） + DNN（深度语义匹配），并通过ONNX加速推理。
* **语义模型**：使用预训练模型（如BERT）进行意图识别，HuggingFace Transformers进行意图分类。

### 9.3 智能推荐

1. **目标**：基于用户行为、社交关系、内容特征，实现实时个性化推荐。系统根据用户的历史搜索、浏览记录以及社交动态等行为来推荐相关个性化内容，包括帖子、俱乐部和活动。
2. **技术栈**

* **图神经网络**：PyTorch Geometric（用户-圈子-帖子异构图）。
* **实时特征**：Flink计算用户行为热度值。
* **特征存储**：Hopsworks特征库

## 10.团队分工

|  |  |  |
| --- | --- | --- |
| 学号 | 姓名 | 分工 |
| 2253551 | 李沅衡 | 后端设计和开发 |
| 2251225 | 王铭乾 | 后端设计和开发 |
| 2250763 | 李俊旻 | 前端设计和开发 |
| 2254272 | 赵子毅 | AI模块设计和开发 |
| 2251760 | 黄志栋 | AI模块设计和开发 |

## 11.项目日程详细安排

将项目分为4个主要阶段，每个阶段的目标明确：

1. 需求分析与设计阶段（1-2周）：

* 明确需求，绘制业务流程图、功能模块图。
* 确定前端技术选型（Vue）、后端框架（Spring Boot 或 Spring Cloud）、数据库类型（MySQL）、AI模块功能设想。
* 输出成果：需求文档、系统架构图、初步 ER 图。

1. 基础框架搭建与基础功能开发阶段（3-4周）：

* 前端开发：Vue 框架搭建，完成首页、用户登录注册界面。
* 后端开发：完成用户管理功能（注册、登录 API），搭建数据库存储用户信息。
* AI模块：确定 AI 模块的具体功能、选择模型。
* 输出成果：一个可以运行的最小化功能系统。

1. 核心功能模块开发与整合阶段（4-6周）：

* 实现核心功能
* 前端完善并美化界面
* 后端完成项目核心功能并与前端对接
* AI模块接入整合

1. 系统优化与交付阶段（2周）：

* 进行性能优化（后端服务的负载均衡）。
* 实现服务治理（如 Spring Cloud Gateway 实现简单的流量控制）。
* 编写用户操作手册和系统文档。

## 13.附加部分

### 13.1 参考文献与数字资源

* **微服务架构**：《微服务架构设计模式》--Chris Richardson
* **前端开发**：《Vue.js 3 By Example》--Giuseppe Cali
* **数据库管理**：《MySQL 8 Query Performance Tuning》--Jesper Wisborg Krogh
* **AI与机器学习**：《Deep Learning with Python, Second Edition》--Francois Chollet
* **项目管理**：《软件工程》--罗杰·S.普莱斯曼，布鲁斯·R.马克西姆
* **测试**：《软件测试技术》--杜庆峰

### 13.2 相关课程

* **软件设计模式**：范鸿飞
* **微服务架构**：刘岩
* **系统分析与设计**：孙萍
* **分布式系统**：饶卫雄
* **数据结构**：张颖
* **算法设计与分析**：罗烨
* **数据库原理与应用**：袁时金
* **软件工程**：杜庆峰
* **人工智能导论**：邓浩
* **软件测试**：杜庆峰
* **软件项目管理**：黄杰

### 13.3 相关知识

* 分布式系统的数据一致性和事物处理机制
* AI模型的训练与部署
* 前端性能优化策略
* 高并发场景下的优化技巧

### 13.4 项目挑战

* 技术栈的整合，确保前后端技术的无缝衔接
* 未来流量增长时系统的高可用性和可扩展性
* 数据安全和个人隐私保护
* 持续集成的工作流程，包括自动化构建、测试与部署

### 13.5 课程建议

无