知你社交——基于 Springboot 的 AI 个性化社交平台

开题报告

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 13 日

1.项目概述

1.1 项目背景

随着社交媒体和内容平台的迅速普及，用户越来越依赖在线平台分享生活、思想和经验，对个性化、实时性及高可用性的需求不断上升。数字化转型推动了智能化和高效管理的需求，互联网技术、人工智能、大数据和云计算的发展带来了新的商业机会和技术挑战。

在这样的背景下，用户基数的急剧增长要求平台具备处理高并发读写请求的能力，确保用户操作的实时响应和数据一致性。同时，构建集成化、智能化的系统平台已成为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，可以提升运营效率，优化客户体验。

本项目旨在开发全面、智能化的系统平台，提供社交互动、内容管理、AI 推荐、智能搜索和活动管理等功能，为用户和企业提供智能化、高效的服务。

1.2 项目范围

本项目旨在构建一个内容分享与社交互动平台，支持用户发布笔记、点赞、收藏、关注等功能，满足用户对社交和内容分享的高标准需求。项目采用微服务架构，以应对海量用户的高并发需求，确保平台的高可用性和高扩展性。通过分布式架构实现数据一致性和实时响应，结合创新的社区互动方式，提升用户体验。项目的范围定义清晰，确保小组成员在开发过程中保持一致的理解，并明确项目的边界和交付物。

1.2.1 项目目标

系统的核心目标是构建一个稳定、高效、可扩展的内容分享与社交互动平台，满足用户在高并发环境下对个性化内容推荐、实时互动和数据一致性的需求。该平台的具体目标包括:

- **高效的内容分享平台**：提供便捷的内容创建、分发和管理工具，让用户能够快速传播信息并提升互动体验。
- **多层级社交网络构建**：支持用户建立不同层级的社交关系，促进信息共享与社群互动，提高社交网络的黏性。
- **AI 驱动的个性化服务**：利用人工智能算法分析用户偏好，提供精准推荐、智能搜索和自动化交互，提升个性化体验。
- **笔记管理与互动**：支持云端笔记存储、标签分类和多人协作，使用户能够轻松记录想法并与他人交流。
- **系统稳定性**：通过高并发处理、故障恢复机制和安全防护措施，确保平台在各种复杂场景下稳定运行。

1.2.2 项目边界

为了确保项目的聚焦性，平台的开发将专注于上述功能模块。以下内容不包括在项目范围内:

- 第三方社交平台的整合。
- 视频直播及其他媒体形式的支持。
- 电商、广告管理等业务扩展功能。

1.2.3 项目约束与假设

- **约束条件**：平台需具备高可用性和数据一致性，在面对大量用户并发操作时，必须保持稳定。数据隐私和安全是项目的首要考虑，所有用户交互和数据存储必须符合相关的安全规范。
- **假设条件**：假设用户希望通过个性化推荐和实时通知功能提升平台的互动性，同

时假设系统能够通过分布式架构有效应对高并发需求。

1.2.4 验收标准

为了确保项目的成功交付，项目的验收标准包括：

- **功能完整性**：平台的核心功能能够正常使用，并满足用户需求。
- **高并发支持**：平台在高并发环境下能够稳定运行，数据一致性和用户操作的实时性得到保障。
- **性能测试通过**：系统通过压力测试，满足平台高可用和高扩展性要求。
- **后台管理功能**：管理员可以通过后台管理系统有效管理用户内容并确保平台合规运行。

1.2.5 范围变更管理

任何范围变更必须通过小组讨论并获得批准后方可实施。项目的范围变更需要对项目时间表、资源分配及其他相关文档进行相应的更新，以确保变更对项目的影响最小化。

3.项目功能

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**：
   - 实现用户之间的关注与取关操作，支持用户查看关注者与粉丝列表。
   - 用户关系的变化将同步更新推荐服务，用于个性化推荐调整。
   - 用户可以对笔记进行点赞、取消点赞，以及收藏笔记功能。
   - 收藏功能支持分类管理，用户可以对收藏的笔记进行分类保存，方便管理收藏内容。
3. **内容管理**：
   - 用户可以发布、编辑、删除笔记，支持设置笔记的可见权限（公开、仅好友可见、私密）。
   - 支持笔记置顶功能，便于用户展示重要内容。
   - 评论服务支持嵌套评论与多层回复，用户可以通过评论与其他用户互动。
4. **智能推荐系统**：
   - 基于用户的行为数据（浏览历史、点赞记录、收藏情况等），通过机器学习算法

提供个性化推荐。

- 推荐服务与用户关系、点赞、收藏等操作数据联动，确保推荐内容的精准性和时效性。

5. **智能搜索功能**：

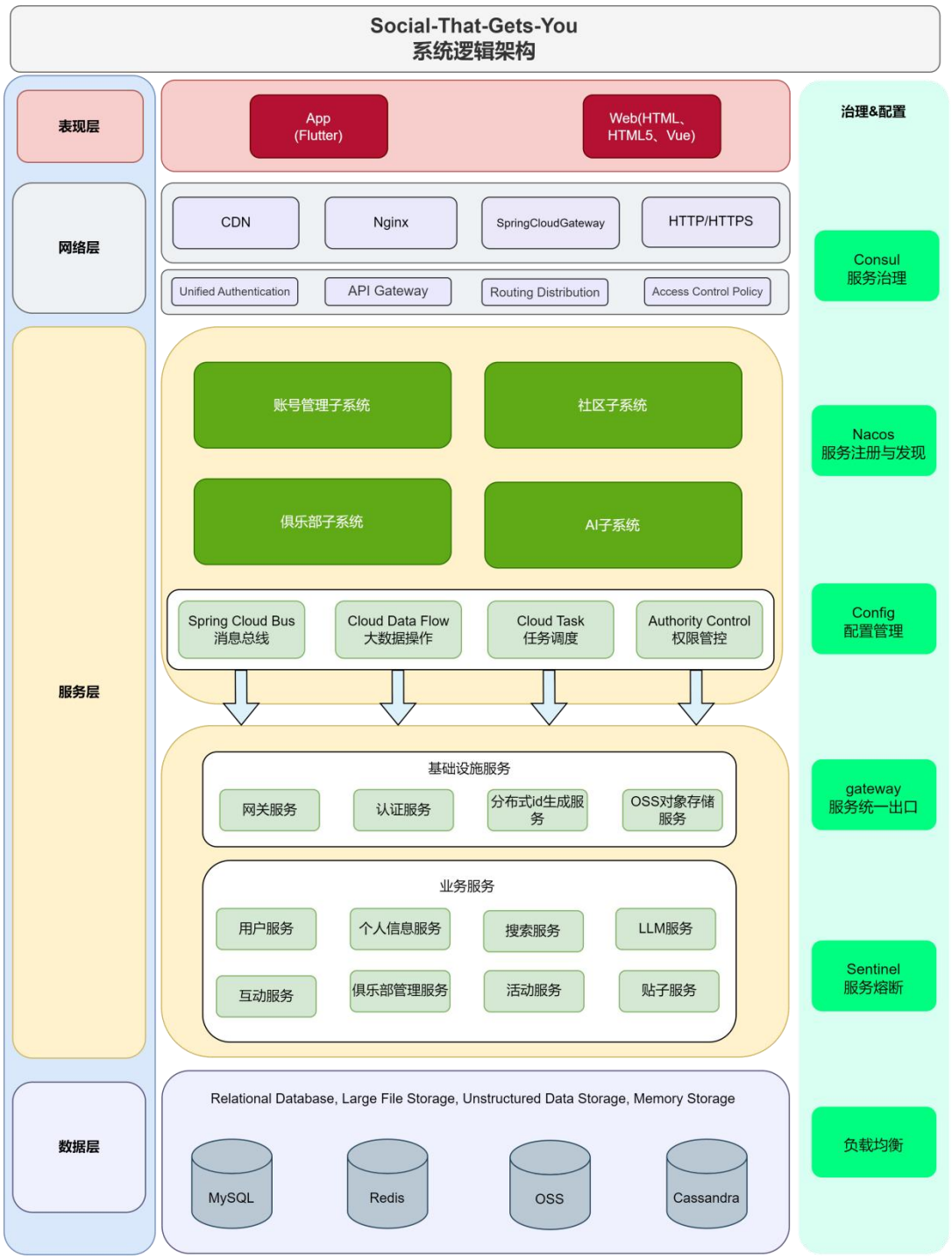
- 用户可以通过关键词搜索笔记、用户、标签、话题，实现高效全文检索。
- 集成智能搜索引擎，提供高效、准确的内容搜索服务。

6. **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）

- **Web 用户端**：基于 Vue 框架开发，提供桌面端用户界面，支持用户发布笔记、查看内容、互动等操作。
- **移动端（如果有）**：基于 uni-app 开发，提供移动端用户界面，确保用户能够方

便地通过移动设备访问平台。

- **反向代理与负载均衡**：表现层通过 **Nginx** 反向代理服务器将用户请求分发至后端微服务，提供前后端解耦和负载均衡能力，保证用户请求的可靠分发和访问速度。

4.2.2 网络层（Network Layer）

- **服务网络层** 使用 **Spring Cloud Gateway** 作为平台的统一入口，负责处理外部请求与后端微服务之间的通信，提供灵活的路由、负载均衡和安全控制等核心功能。其主要职责包括：
  - **请求路由与负载均衡**：Spring Cloud Gateway 作为 API 网关，负责将用户的 HTTP 请求路由至合适的微服务，确保请求能够根据规则被高效地分发到不同的后端服务。系统支持动态路由调整和负载均衡机制，确保在高并发情况下，系统的可用性和响应速度。特别地，通过 **Nginx** 结合 Gateway 提供的负载均衡能力，支持后端微服务的动态扩展和高效分配流量。
  - **统一鉴权与用户信息透传**：通过集成 **SaToken** 实现统一的用户认证和授权机制。每次请求都会通过网关进行 Token 解析，验证用户身份和权限，并将解析出的用户信息（如用户 ID、角色等）透传至下游服务，确保后端微服务无需重复进行身份认证，从而简化微服务内部的鉴权逻辑。通过 RBAC（基于角色的访问控制）模型，网关可以对请求进行权限校验，确保用户只能访问被授权的资源。
  - **限流与熔断机制**：为保护后端微服务不被大量请求冲击导致崩溃，网关层集成了 **Sentinel** 进行限流和熔断处理。通过动态限流规则，可以在流量激增时控制用户请求的速率，防止系统超负荷运转。熔断机制则确保在后端服务异常时，网关能够迅速返回默认响应，避免影响到用户体验。
  - **全局过滤器与请求增强**：Spring Cloud Gateway 支持全局过滤器，在所有请求经过网关时，可以通过这些过滤器实现日志记录、参数校验、Header 增加或修改等功能，增强请求处理能力。例如，可以在请求进入网关时自动注入用户的权限信息，或在响应阶段对返回的结果进行统一格式化处理。

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**：作为服务注册与配置中心，Nacos 负责管理微服务的注册、发现和配置。Nacos 支持动态配置管理，确保服务实例能够自动注册与发现，有效应对系统扩展或缩容时的服务治理需求，并在运行时动态调整配置。
- **Sentinel**：流量控制、熔断降级和系统保护
- **Redis**：作为内存级缓存系统，用于存储热点数据和高频访问的数据，如用户信息、笔记详情、计数服务等。Redis 提供快速的读写操作，有效减轻了数据库的负载，提高了系统的响应速度。通过 **分布式缓存设计** 和 **持久化机制**，确保数据的高可用性和可靠性，避免因缓存失效导致的系统性能问题。
- **RocketMQ**：作为高性能、分布式消息中间件，RocketMQ 负责微服务之间的异步消息传递。它通过消息队列实现微服务的解耦和异步处理，特别适合高并发场景下的事件驱动架构。RocketMQ 通过持久化和重试机制，确保消息传递的可靠性和数据一致性，保障数据最终一致性。
- **Zookeeper**：分布式协调服务（Cassandra 依赖）
- **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

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

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。

2. **项目分阶段规划**

见项目日程详细安排部分

3. **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 80% 以上（大学项目推荐一个实际目标）。
2. 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。
3. 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。
4. 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。

5. 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）。
6. 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**

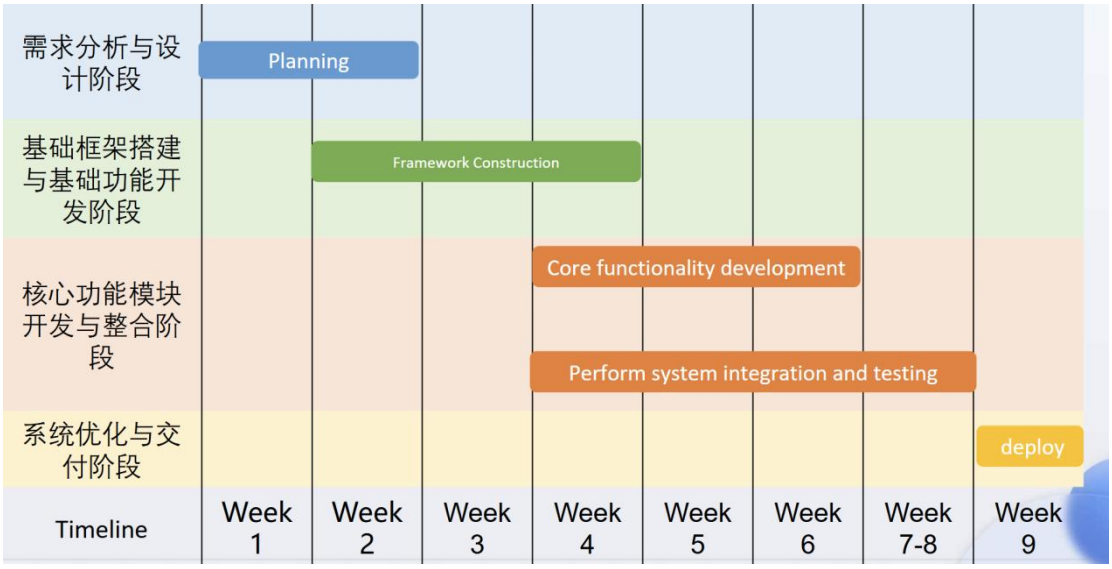
- **图神经网络**：PyTorch Geometric（用户-圈子-帖子异构图）。
- **实时特征**：Flink 计算用户行为热度值。
- **特征存储**：Hopsworks 特征库

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。
2. 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。
3. 核心功能模块开发与整合阶段（4-5 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

4. 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