项目总结报告

日期：2021年1月7日

组号	6	项目名称	交大说说
编程语言	HTML+JavaScript+Java +Shell+Python	开发平台和框架	OpenStack、React、SpringBoot

软件需求特性

1.是否实现了项目立项时的所有需求？列出实现的新增需求和未实现的需求。

对于基础功能基本高级功能外基本已完成。

未有新的需求变更。

未实现的需求：

（1）推荐系统未实现

2.采用哪种架构风格？哪些设计模式?

架构风格：后端使用了微服务架构，运用 Kubernetes 平台，使所有的服务拥有极高的可用性与横向扩展能力。

设计模式：MVC。我们的设计模式遵循面向对象原则，后端功能按照 Controller, Service, Dao, Repository 进行结构分层；运用接口与实现分离的思想，整个开发面向接口编程，拔高模块化和开发效率；后端的开发遵循依赖注入，确保了控制反转的效果。

3.技术方案有哪些亮点?

（1）违禁词系统使用了一层cache，在每次启动后加载在内存里，免去磁盘的访问，便于发表问题和答案、评论的时候检查违禁词。

（2）专家徽章系统应用启发式算法，为最近回答的问题的theme和被关注人数赋予权重，在每个人发表问题时会计算一次，这样他的专家徽章就是最近几次发表最多相关问题的徽章。

（3）语义搜索使用了ABCNN模型，实现了通过语义检索相似内容

（4）对图片的处理上，使用了云原生的对象存储服务进行存储管理，使用了MinIO框架。使得图片存储服务拥有了极高的可扩展性和可用性，同时拥有很高的吞吐量。

（5）我们还使用 Istio 进行服务治理，引导服务流量分配，配合 Kubernetes 负载均衡，服务的请求数据有了监控。后端使用了 MiniKube 和 LazyKube 进行监控，每一个服务的状态、流量、资源消耗、错误输出都能有及时的、自动化的反馈，使得运维的工作大为减少。后端还部署了 Jenkins，运用 DevOps 的思想，执行持续集成、持续部署、持续发布（CI/CD），监听 GitHub release 分支的 commit 和 merge，自动化执行编译、测试、打包、部署上线的流水线流程，简化运维成本，关注开发效能。

（5）为了提高 Docker 的利用效率，我们不使用 DockerHub（国内对 DockerHub 的连接十分缓慢）。我们采用自建的 Docker Registry 进行镜像管理、金丝雀发布和私有部署，解决 CI/CD 的性能瓶颈，让发布上线镜像的过程从 数十分钟 缩减为 一分半，大大提高开发效率。

4.是否做了单元测试？是否做了系统功能测试？是否做了性能测试？是否做了兼容性等其他非功能测试?

进行了单元测试、性能测试、压力测试，基本达到了要求，测试基本通过

项目组成员对项目的贡献度（%）	
姓名（按拼音排序） 许正霖（组长） 龙泓栈 斯金泽 张淇 周义天 贡献度（%） 20 20 20 20 20	
软件度量	
软件代码行数（不包括注解行、空行和复用代码）：	12647
复用他人代码行数：	266
类的个数：	94

经验、教训和建议

这次项目基本实现目标，但对于高级功能的实现缺乏经验所以有部分没有完成。

对于前端，还需要加强部分组员的代码结构规范和鲁棒性。在写测试时覆盖率更能体现这一点。另外，对于设计某些界面缺乏美观性和协调性。

这次项目的经验教训：

（1）计划分配和制定十分重要，合理的计划能加快迭代进度和效率；

（2）组员之间要积极沟通，明确分工与合作；

（3）接口、需求要尽早明确，避免无用功；

（4）工作要尽早开展，避免进度风险；

（5）在轮子的选择上，要选择足够健壮、有人维护的，避免遇到问题无法解决；

（6）代码与文件的层级结构要足够合理，避免多人 merge 时出现的混乱；

（7）每次迭代的工作量需要合理；

（8）每次工作的分配必须尽量明确，减少重复的工作；

（9）技术栈的选择要尽早制定，避免因为技术变更而出现的技术风险。