WeBlog 测试分析报告

1 引言

1.1 编写目的

本文档是在项目具体代码完成过程中和完成后，项目中负责测试的人员记录对测试用例的设计、运行情况，并对出现的问题提出改进的建议。

此文档将供本系统的开发和维护人员查阅和使用，方便进一步开发以及正确地部署维护项目，并可指导下一次测试的展开。

1.2 背景

1. 被测软件系统名称：WeBlog 个人博客系统
2. 软件开发者：徐海洲、张佳琪、汪思涵、王云琨、张宇、李景涛、李国昌；
3. 用户：按角色分为访客和注册用户；用户特征为计算机技术人员,、计算机专业的学生、从业者、爱好者，且懂得基础的 markdown 标记语言使用，愿意分享技术。
4. 测试环境与实际运行环境之间可能存在的差异以及这些差异对测试结果的影响：
   (1) 实际运行环境：位于公网的 Ubuntu22.04 server 版，安装了 Docker 20.10.12；
   (2) 测试环境：Ubuntu 20.04 desktop 版，安装了 Docker 20.10.12、MySQL 5.7 以及 Redis 5.0.7；

   项目在测试和运行中均被打包为 Docker 镜像部署，可以尽最大可能保证测试环境和实际运行环境在操作系统层面上的一致性；但由于两个环境使用的计算机硬件配置和网络环境有细微差异，因此性能测试等可能存在一些偏差，但总体而言可以参考。

1.3 参考材料

1. WeBlog 软件需求说明书
2. WeBlog 面向对象分析设计文档

2 测试概述

2.1 测试采用的标准及技术

1. 测试采用的标准及技术

表 1 开始/中断/完成测试

开始/中断/完成测试	标准说明

开始测试标准	硬件环境可用，且软件已经安装部署完成
中断测试标准	安装无法正确完成；测试机异常；文档出现过多错误；单元测试出现过多错误
完成测试标准	完成测试计划中的所有计划项，达到需求分析文档中的功能和性能等需求指标。

表 2 测试类型

测试类型	说明
功能测试	根据《WeBlog 个人博客系统需求规格说明书》和《WeBlog 个人博客系统设计说明书》，检查产品是否正确实现了个能
边界值测试	选择边界数据测试，确保系统功能正常，程序无异常
界面测试	检查界面是否美观合理，跳转逻辑是否正常
文档测试	检查文档是否足够，描述是否合理，是否便于开发者和用户阅读
性能测试	检查软件性能是否达到需求规格说明书中的要求

表 3 测试技术

测试类型	说明
单元测试	对具体的类的函数，以及前端的界面进行测试
集成测试	检测模块集成后的系统是否达到需求
确认测试和系统测试	对业务流程以及数据流的处理是否符合标准，系统对业务流的处理是否存在逻辑不严谨和错误以及是否存在不合理的标准及要求

一般情况下，确认测试和系统测试不是一个概念。确认测试（有效性测试）的目的是发现软件实现的功能与需求规格说明书不一致的错误，以确保软件符合所有功能、行为、性能的需求；而系统测试是集中检查系统包括硬件和信息在内的所有元素是否协作合适，整个系统的性能和功能是否达标。对于有外部系统的软件（比如工业控制软件），确认测试和系统测试的区别是明确的；但网页应用作为相对独立的系统，不需要在多样的外部环境下进行复杂的系统测试，因此这里将两者合二为一。

2. 单元测试(各个类)

项目的后端采用 Java 语言中最常用的测试组件 JUnit 进行单元测试，测试分模块进行。后端中每个被测类都有一个与之对应的测试类，测试类包含了和被测类相同的方法，用于测试原类，每个方法为一个最小单元。运行测试时将执行这些测试方法，判断是否满足提前编写的断言。如果不满足断言，则单元测试结果将显示失败以及失败原因；若成功则会显示通过。后端涉及到的被测类以及测试结果概要如表 4 所示。

表 4 后端被测类及测试结果概要

序号	测试类名	类功能	是否成功	是否更正
1	PublicController	处理来自前端的、向访客公开的功能的请求，如查看博文列表、搜索博文、查看热门标签等。	是	/

2	LoginRegisterController	处理来自前端的登录和注册请求	是	/
3	BloggerController	处理来自前端的、关于博主信息的请求，如获取博主的信息、修改博主的个人信息、更改头像等功能	是	/
4	BlogController	处理来自前端的、和博主个人信息无关但和博客相关的信息的请求，如获取某博客下的所有博文等	是	/
5	SubscribeController	处理来自前端关于用户订阅相关的请求	是	/
6	TagController	处理来自前端的和标签相关的请求	是	/
7	AttachmentController	处理来自前端的和附件相关的请求	是	/
8	PostController	处理来自前端的和博文相关的请求	是	/
9	CommentController	处理来自前端的和博文下的评论相关的请求，如添加评论、获取所有回复等	是	/
10	BloggerServiceImpl	实际处理处理博主信息的服务类。	否	是
11	PermissionServiceImpl	用于检测用户是否有访问某接口的权限的服务类。大多数 controller 类需要调用此类功能以判断当前登录的用户是否有访问接口的权限	是	/
12	BlogServiceImpl	处理和博客信息相关的服务类	是	/
13	SubscribeServiceImpl	处理和订阅功能相关的服务类	是	/
14	TagServiceImpl	处理和标签相关的服务类	是	/
15	AttachmentServiceImpl	处理和附件相关的服务类	否	是
16	PostServiceImpl	处理和博文相关的服务类	是	/
17	CommentServiceImpl	处理和博文下的评论相关的服务类	是	/
18	PermissionMapper	和实现权限认证相关的数据访问类	否	是
19	BloggerMapper	访问数据库中博主信息的类	是	/
20	BlogMapper	访问数据库中博客信息的类	是	/
21	SubscribeMapper	实现订阅功能的数据访问类	是	/
22	TagMapper	访问数据库中标签信息的类	是	/
23	AttachmentMapper	访问数据库中附件信息的类	否	是

24	PostMapper	访问数据库中帖子信息的类	否	是
25	CommentMapper	访问数据库中评论信息的类	是	/

前端单元测试主要使用 vue 自带的 Vue Test Utils，测试内容主要为项目中的各部分组件，测试过程考虑了合法数据及不合法数据，并检测组件运行结果，具体测试内容在第二节中阐述。

3. 集成测试

对于后端，本系统的集成测试的重点是控制层、服务层、持久层的组合是否能够实现预期的要求。对此我们依靠 SpringBoot 的依赖注入机制，复用了后端单元测试的代码，将其中的桩模块和驱动模块分层逐步替换成实际的类和模块。

对于前端，本系统的集成测试重点是页面间的跳转是否正确，以及前后端的通信是否正常。前后端通信部分，结合浏览器的 Network 输出，根据不同的数据样例测试了所有设计前后端交互的接口，主要关注返回状态码、返回数据及 cookie 设置；面跳转部分，主要通过人工检测，在不同的场景下测试页面跳转和页面参数传奇是否正常。

4. 系统测试

我们对 WeBlog 个人博客系统进行了功能测试、界面测试、性能测试等系统测试。

对于功能测试，按照《WeBlog 软件需求说明书》中的数据流图，为系统的各个功能设计了测试方案。其中，对于对数据需求不高或功能逻辑复杂度不高的功能，我们仅提出了大致的测试要求；对于几项对数据正确性要求比较高的功能，我们按照等价类划分的方法设计了测试用例。对于每个测试用例，均以清晰的表格形式记录了测试结果以及发现的问题。

对于界面测试，我们设计了关于界面的测试标准，并在测试中对照测试标准给予相应的评价；

对于性能测试，按照《WeBlog 个人博客系统需求规格书》中对性能的要求，设计了性能测试表格，按照该表格对系统中比较重要的几项性能需求进行了测试。每项测试都记录了期望的性能和实际性能。

2.2 测试环境与配置

资源名称/类型	配　　置
测试 PC	前端： 处理器 Intel i7-10510U CPU @ 1.80GHz 机带 RAM DDR4 16.0GB 操作系统 windows11 后端： 处理器 Intel i7-10510U (8) @ 4.900GHz 机带 RAM 16.0GB 操作系统 Ubuntu 22.04.5 LTS x86_64
浏览器	Chrome 111.0.5563.64（64 位）
数据库管理系统	MySQL 5.6 (Docker 镜像)、Redis（Docker 镜像）
单元测试框架	JUnit4, SpringMockMvc

负载性能测试工具	JMeter

2.3 测试机构和人员

测试阶段	测试机构名称	负责人	参与人员	所充当角色
后端单元测试	后端开发组	徐海洲	徐海洲、王云琨、张宇、李景涛	初步测试，保证业务逻辑、权限控制和数据库访问的正确性
前端单元测试	前端开发组	张佳琪	张佳琪、汪思涵	初步测试，保证前端页面跳转和各组件功能的正确性
集成测试	全组	徐海洲	全组成员	检查前端页面跳转、后端模块组合，以及前后端通信交互是否正常
系统测试	全组	徐海洲	全组成员	检查系统的性能、功能等是否符合需求规格说明书中的要求

2.4 测试问题小结

所有测试项目都已执行。其中：
1. 单元测试相对完善。对于后端，除了部分涉及到文件传输的功能，其他的类和方法均得到了充分的测试；
2. 前端测试和界面测试充分考虑到用户各种操作的情况。在后续开发中，可以考虑利用 selenium 等框架提高前端测试的自动化程度；
3. 功能测试和性能测试参照需求规格说明书，筛查并弥补了不足。

3 单元测试

此部分将给出单元测试的测试用例设计以及测试执行的结果。由于涉及到的用例过多，这里仅列出部分用例，完整的测试用例将在其他文档中另外描述。

3.1 后端-控制层（Controller Layer）

控制层负责处理客户端发来的 HTTP 请求，并将请求转发到相应的服务层。

表 1 PostController 中方法的测试用例

编号	函数声明	输入数据	期望结果	实际结果	测试结果
1	getBloggerPosts(uid: long, page: int, pageSize: int)	uid = 1, page = 0, perpage = 10	访问正常	访问正常	通过
2	getBloggerPosts(uid: long, page: int, pageSize: int)	未登录, uid = 1, page = 0, perpage = 10	访问正常	访问正常	通过
3	addPost(uid: long, post: PostInfo)	uid = 1, post = PostInfo(title="理塘", content="理塘简介", detail="到达世界最高层——理塘。")	访问正常	访问正常	通过
4	addPost(uid: long, post: PostInfo)	未登录, uid = 1, post = ...	返回错误码 1	返回错误码 1	通过
5	addPost(uid: long, post: PostInfo)	登录用户 uid=2, uid=1, post=...	返回错误码 2	返回错误码 2	通过
6	getPostInfo(uid: long, pid: long)	uid = 1, pid = 2	访问正常	访问正常	通过
7	getPostDetail(uid: long, pid: long)	uid = 1, pid = 2	访问正常	访问正常	通过
8	updatePostInfo(uid: long, pid: long, post: PostInfo)	uid = 1, pid = 2, post = PostInfo(title="理塘", content="理塘简介", detail="到达世界最高层——理塘。")	访问正常	访问正常	通过
9	updatePostDetail(uid: long, pid: long, detail: String)	uid = 1, pid = 2, detail = "详情"	访问正常	访问正常	通过
10	deletePost(uid: long, pid: long)	uid = 1, pid = 2	访问正常	访问正常	通过

表 2 BloggerController 中方法的测试用例

编号	函数声明	输入数据	期望结果	实际结果	测试结果

11	getSelfBloggerInfo()	-	访问正常	访问正常	通过
12	getSelfBloggerInfo()	未登录	返回错误码 1	返回错误码 1	通过
13	getBloggerInfo(uid: long)	uid = 1	访问正常	访问正常	通过
14	updateBloggerInfo(uid: long, info: BloggerInfo)	uid = 1, info = BloggerInfo(...)	访问正常	访问正常	通过
15	updateBloggerInfo(uid: long, info: BloggerInfo)	未登录, uid = 1, info = BloggerInfo(...)	返回错误码 1	返回错误码 1	通过
16	updateBloggerInfo(uid: long, info: BloggerInfo)	登录用户 uid=2, uid = 1, info = BloggerInfo(...)	返回错误码 2	返回错误码 2	通过
17	updateBloggerAvatar(uid : long, file: MultipartFile)	uid = 1, file = 本地图片(≤2MB)	访问正常	访问正常	通过

其他被测类测试用例从略.

3.2 后端-服务层（Service Layer）

服务层是业务逻辑的核心部分，它主要负责业务逻辑和数据的处理。

表 3 AttachmentServiceImpl 中方法的测试用例

编号	函数声明	输入数据	期望结果	实际结果	测试结果
93	getBloggerAttachments(uid: long, page: int, perpage: int)	uid = 1, page = 0, perpage = 10	用户 1 的所有附件	用户 1 的所有附件	通过
94	getAttachmentInfo(aid: long)	aid = 1	附件 1 的信息	附件 1 的信息	通过
95	saveAttachment(uid: long, filename: String, file: MultipartFile)	uid = 1, filename = "压缩包.gz", file = ...	4 (新插入的附件 ID)	4	通过
96	deleteAttachment(aid: long)	aid = 1	-	-	通过
97	getAttachmentMd5sum(aid: long)	aid = 1	2d1bbde2acac0afd07646d98154f402e	2d1bbde2acac0afd07646d98154f402e	通过
98	getFileMD5(file: MultipartFile)	file = ...	2d1bbde2acac0afd07646d98154f402e	2d1bbde2acac0afd07646d98154f402e	通过

3.3 后端-持久层（Persistence Layer）

持久层主要负责数据的持久化，即将数据存储到数据库或其他持久化存储介质中。在后端使用的框架中，实现数据增删改查的类又被称为 Mapper。

表 4　CommentMapper 中方法的测试用例

编号	函数声明	输入数据	期望结果	实际结果	测试结果
176	getComment(pid: long, start: int, count: int)	pid = 1, start = 0, count = 10	1 号帖子的所有评论	1 号帖子的所有评论	通过
177	getCommentOnlyUnreviewed(id: long, start: int, count: int)	pid = 1, start = 0, count = 10	1 号帖子的所有未审核的评论	1 号帖子的所有未审核的评论	通过
178	addComment(pid: long, bid: long, content: String, replyTo: long)	pid = 1, bid = 1, content = '评论', replyTo = null	数据库中为 1 号帖子添加一条记录	数据库中为 1 号帖子添加一条记录	通过
179	addComment(pid: long, bid: long, content: String, replyTo: long)	pid = 1, bid = 1, content = '评论', replyTo = 4	数据库中为 1 号帖子添加一条回复评论 5 的记录	数据库中为 1 号帖子添加一条回复评论 5 的记录	通过
180	deleteComment(cid: long)	cid = 1	评论从数据中删除	评论从数据中删除	通过
181	getReplyComment(replyTo: long)	replyTo = 4	4 号评论的所有回复	4 号评论的所有回复	通过
182	reviewComment(id: long, pass: boolean)	id = 2, pass = true	2 号评论的状态更新为已审核	2 号评论的状态更新为已审核	通过
183	reviewComment(id: long, pass: boolean)	id = 2, pass = false	2 号评论的状态更新为审核不通过	2 号评论的状态更新为审核不通过	通过
184	getCommentById(id: long)	id = 1	1 号评论的记录	1 号评论的记录	通过

3.4 前端

编号	函数声明	输入数据	期望结果	实际结果	测试结果
1	顶部导航栏—搜索框	{ searchQuery ： 'vue frontend' }	正常渲染，正常工作，跳转至搜索	正常渲染，正常工作，跳转至搜索	/
2	顶部导航栏—主页按钮	/	正常渲染工作	正常渲染工作	/
3	顶部导航栏—登录	click	正常渲染，正常工作，跳转至登录	正常渲染，正常工作，跳转至登录	/
4	顶部导航栏—注册	click	正常渲染，正常工作，跳转至注册	正常渲染，正常工作，跳转至注册	/
5	顶部导航栏—登出	click	正常渲染，正常工作	正常渲染，正常工作	/
6	注册	{ userName： 'testUser', email: '19204@pku.edu.cn', password: 'testpassword'}	正常渲染，正常工作	正常渲染，正常工作	/
7	注册	{ userName： '', email: '19204@pku.edu.cn', password: 'testpassword'}	正常渲染，返回错误提示，未输入用户名	正常渲染，返回错误提示，未输入用户名	/
8	注册	{ userName： 'testUser', email: '', password: 'testpassword'}	正常渲染，返回错误提示，未输入邮箱	正常渲染，返回错误提示，未输入邮箱	/
9	注册	{ userName： 'testUser', email: '19204@pku.edu.cn', password: ''}	正常渲染，返回错误提示，未输入密码	正常渲染，返回错误提示，未输入密码	/
10	注册	{ userName： 'testUser', email: '19204@pku.edu.cn', password: '1'}	正常渲染，返回错误提示，输入密码应大于 5 个字符并短于 0 个字符	正常渲染，返回错误提示，输入密码应大于 5 个字符并短于 0 个字符	/
11	登录	{ userName： 'testUser', password: 'testpassword'}	正常渲染，正常工作	正常渲染，正常工作	/
12	登录	{ userName： 'testUser', password: ''}	正常渲染，返回错误提示，未输入密码	正常渲染，返回错误提示，未输入密码	/
13	登录	{ userName ： '', password: 'testpassword'}	正常渲染，返回错误提示，未输入用户名	正常渲染，返回错误提示，未输入用户名	/
14	个人信息卡片组件	{ id: 0, name: 'testUser'， contact: '1804828593', graduate: 'Peking	正常渲染，正常工作	正常渲染，正常工作	/

		University', email: '19204@pku.edu.cn' ， avatarUrl：'https:xxx'}			
15	个人信息卡片组件	{}	正常渲染，缺失值占位缺省为空白	正常渲染，缺失值占位缺省为空白	/
16	标签卡片	{title: 'HOT TAGS', tags:['vue', 'angularJS', 'git']}	正常渲染，正常工作	正常渲染，正常工作	/
17	搜索卡片	{input: 'vue frontend', tags:['vue', 'frontend']}	正常渲染，正常工作，标签正常动态增加渲染	正常渲染，正常工作，标签正常动态增加渲染	/
18	单个博客卡片	{ id:0, title: "article1", content: "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", author: { id: 0, name: "testuser1", avatarUrl: "xxxx", contact: 'xxxxxxx', email: "xxxxxxxxxx", graduate: "xxxxxxxxxx" }, tags: [{ id: 0, name:"test1", owner:{ id: 0, name: "test", avatarUrl: "url", contact: "123123123", email: "123@pku.edu.cn", graduate: "peking University" }, description:"xxxxxxxxxxxxxx" }] }	正常渲染，正常工作	正常渲染，正常工作	/
19	个人后台—修改用户名	{input: 'testModify'}	正常渲染，正常工作	正常渲染，正常工作	/
20	个人后台—修改用户名	{input: ''}	正常渲染，提示无效修改输入	正常渲染，提示无效修改输入	/
21	个人后台—修改绑定邮箱	{input: '11249@pku.edu.cn'}	正常渲染，正常工作	正常渲染，正常工作	/

22	个人后台—修改绑定邮箱	{input: ''}	正常渲染，提示无效修改输入	正常渲染，提示无效修改输入	/
23	个人后台—修改毕业院校信息	{input: 'ANew University'}	正常渲染，正常工作	正常渲染，正常工作	/
24	个人后台—修改毕业院校信息	{input: ''}	正常渲染，提示无效修改输入	正常渲染，提示无效修改输入	/
25	个人后台—修改联系方式	{input: '18940375921'}	正常渲染，正常工作	正常渲染，正常工作	/
26	个人后台—修改联系方式	{input: ''}	正常渲染，提示无效修改输入	正常渲染，提示无效修改输入	/
27					/
28	浏览页面—浏览	\	正常渲染出浏览帖子页面	正常渲染出浏览帖子页面	/
29	浏览帖子—编辑按钮	click	正常渲染，正常工作，跳转到发布页面	正常渲染，正常工作，跳转到发布页面	/
30	发布页面—发布帖子	{input:Postinfo}	正常渲染，正常工作	正常渲染，正常工作	/

4 集成测试

4.1 页面跳转测试

编号	跳转目的 path	携带参数	所在页面 / 组件	测试结果
1	/login	/	顶边栏	跳转正常
2	/register	/	顶边栏	跳转正常
3	/index	/	顶边栏	跳转正常，回到 index 页面
4	/index	/	登录页面	登陆成功后跳转正常，且 index 页面成功显示为已陆
5	/index	/	注册页面	登陆成功后跳

				转正常
6	/search	context: string	顶边栏	跳转成功且成功传参（显示正确搜索结果）

7	/search	{
		Context:string,
		Tags: tag[]
		}
			主页	跳转成功且成功传参（显示正确搜索结果）

8	/user	UserInfo：Object	主页	跳转成功且传参成功
9	/userboard	/	个人信息卡片	跳转成功，进入个人后台
10	/postblog	/	个人信息卡片	跳转成功
11	/postblog	Postinfo:Object	浏览博客页面	跳转成功且传参成功
12	/readblog	BlogId: Int	博客条目组件	跳转成功且传参成功（后续成功加载博客内容）

4.2 其他部分

1. 后端的集成测试：

根据约定，后端单体程序的开发架构分为控制层、服务层和持久层三层。在单元测试过程中，可以利用借助 MockMvc、Mockito 等 Mock 工具，模拟前端的调用和下层类的方法调用；在集成测试阶段，应当将代码中涉及到的 Mock 对象逐步替换为实际的对象。例如，Controller 层的功能依赖 Service 层的对象，在单元测试阶段，测试人员编写被依赖的 Service 类的 Mock 对象；而在集成测试阶段，将其替换为实际被依赖的对象实现。因此后端的集成测试复用了单元测试的代码，逐步将后端项目中的各类集成起来。

2. 前后端对接：

由于项目的前端部分业务逻辑相对复杂，因此前端测试暂时采用人工手动测试的方法。对于前端而言，绝大多数页面跳转需要后端数据的支持，因此前后端对接的测试可以与界面测试合并。我们考察了页面测试中对前后端接口访问的覆盖情况；对于没有被覆盖到的情况，前端测试人员进行了手动测试。

3. 微服务：

采用 docker+微服务的方法，可以增强项目的可移植性。集成测试的前两点测试时都是针对直接在测试机上运行的后端以及尚未打包的前端项目进行的；在此部分，测试人员将后端 3 个微服务以及与 Nginx 打包在一起的前端分别构建为 4 个镜像，和其他需要的镜像（如数据库、缓存等）部署在测试机中的 docker 虚拟网络中，检查微服务发现机制、服务转发机

制是否正常，以及是否可以通过浏览器正常访问到前端并与后端进行交互。测试通过后，将多个容器的配置编写为脚本，方便在生产环境中复现运行环境。

5 确认测试和系统测试

5.1 功能测试

测试编号	测试用例名称	用户输入	期望结果	实际结果
1	注册	个人联系方式和密码等	显示注册是否成功的提示	符合预期
2	浏览博客	-	显示博文列表和个人信息	符合预期
3	查询博文	标题搜索关键字、标签	查询出的博文	符合预期
4	浏览博文详情	-	博文内容	符合预期
5	登录	个人联系方式和密码	登录成功或失败的提示	符合预期
6	订阅博主	被订阅者 ID	订阅成功或失败的提示	符合预期
7	评论博文	被评论博文 ID 以及评论内容	在博文下显示评论内容	符合预期
8	点赞博文	-	博文被点赞数显示+1	符合预期
9	收藏博文	-	博文被收藏数显示+1	符合预期
10	新建博文	博文内容及标签	在个人中心中显示新博文	符合预期
11	管理博文	编辑、删除博文等操作	在个人中心中显示操作的结果	符合预期
12	管理个人信息	编辑个人信息	在个人中心中显示新个人信息	符合预期

对一些要个要求数据正确性的模块，需要用等价类划分的方法额外设计测试内容。下面是对“创建博文”功能模块的测试设计以及测试结果，采用了等价类划分的测试方法。

表 等价类划分

输入条件	有效等价类	编号	无效等价类	编号
博文标题	长度在 1~32 字符之间	1	（空）	5
			长度超过 32	6
博文简介	长度在 1~128 字符之间	2	（空）	7
			长度超过 128	8
博文内容	长度在 1~5000 字符之间，可包含图片 URI	3	（空）	9
			长度超过 5000	10
权限	“公开”或“不公开”	4	没有选择权限	11

表 测试用例设计

用例编号	输入				预期输出	覆盖范围
	博文标题	博文简介	博文内容	权限		
10.1	Kotlin 的历史	Kotlin 语言历史概要	Kotlin 语言历史内容	公开	创建博文成功	1,2,3,4
10.2	（空）	Kotlin 语言历史概要	Kotlin 语言历史内容	公开	创建博文失败	5
10.3	*** (长度超过 32)	Kotlin 语言历史概要	Kotlin 语言历史内容	公开	创建博文失败	6
10.4	Kotlin 的历史	（空）	Kotlin 语言历史内容	公开	创建博文失败	7
10.5	Kotlin 的历史	*** (长度超过 128)	Kotlin 语言历史内容	公开	创建博文失败	8
10.6	Kotlin 的历史	Kotlin 语言历史概要	（空）	公开	创建博文失败	9
10.7	Kotlin 的历史	Kotlin 语言历史概要	*** (长度超过 5000)	公开	创建博文失败	10
10.8	Kotlin 的历史	Kotlin 语言历史概要	Kotlin 语言历史内容	没有选择权限	创建博文失败	11

表 测试结果记录

用例编号	预期结果	实际结果	测试结果	测试日期	测试人
10.1	创建博文成功	创建博文成功	通过	2023.03.10	徐海洲
10.2	创建博文失败	创建博文失败	通过	2023.03.10	徐海洲
10.3	创建博文失败	创建博文失败	通过	2023.03.10	徐海洲
10.4	创建博文失败	创建博文失败	通过	2023.03.10	徐海洲
10.5	创建博文失败	创建博文失败	通过	2023.03.10	徐海洲
10.6	创建博文失败	创建博文失败	通过	2023.03.10	徐海洲
10.7	创建博文失败	创建博文失败	通过	2023.03.10	徐海洲
10.8	创建博文失败	创建博文失败	通过	2023.03.10	徐海洲

5.2 界面测试

测试编号	测试项	测试评价
1	顶边栏搜索输入文本框	正常输入内容
2	MultiSearch 组件中的搜索输入文本框	正常输入内容
3	顶边栏登录，注册按钮	正常点击跳转
4	顶边栏个人主页按钮	正常点击跳转

5	顶边栏登出按钮	正常点击登出
6	顶边栏主页按钮	正常点击跳转
7	用户卡片——写博客按钮	正常点击跳转
8	MultiSearch 组件中的增加标签	正常输入标签，增加标签，删除标签
9	MultiSearch 组件中的搜索按钮	正常点击跳转
10	Hot Tags 卡片中的单个 tag 按钮	正常点击跳转
11	博客卡片用户名按钮	正常点击跳转
12	博客卡片文章标题按钮	正常点击跳转
13	个人后台信息输入框	正常点击跳转
14	个人后台更新信息按钮	正常点击跳转
15	个人后台可点击头像	正常跳出资源管理器选择头像
16	用户卡片关注	正常点击
17	用户卡片取消关注按钮	正常点击
18	登录页面用户信息输入框	正常输入，离开文本框后如果输入内容不合法会边框红色警告
19	登陆页面登录按钮	正常输入，输入信息不合法会弹出框提示
20	注册页面个人信息输入框	正常输入，离开文本框后如果输入内容不合法会边框红色警告
21	注册页面注册按钮	正常输入，输入信息不合法会弹出框提示

5.3 性能测试

测试编号	性能名称	输入数据	期望性能	实际性能	测试结果
1	浏览并发量	随机	并发>1000	并发>1000	通过
2	浏览主页延迟	随机	延迟<1000ms	延迟<1000ms (avg: 23ms)	通过
3	搜索延迟	随机	延迟<1000ms	延迟<1000ms (avg: 17ms)	通过
4	浏览个人信息延迟	随机	延迟<1000ms	延迟<1000ms (avg: 13ms)	通过
5	上传图片延迟	图片大小小于 2MB	延迟<1000ms	延迟<1000ms (avg: 143ms)	通过
6	发布博文延迟	随机	延迟<1000ms	延迟<1000ms (avg: 14ms)	通过

5.4 其他部分

1. 文档测试（软件配置审查）：文档测试是确认测试的一部分，应该在系统测试前完成。在此项测试中，我们检查是否存在与需求规格说明书以及设计文档中不符的内容，并检查文档是否有遗漏、描述不清的问题。
2. 安装测试和可移植性测试：由于系统使用 docker 部署，因此影响可移植性的主要是 CPU 的架构。除了在 amd64 架构的目标机上运行，我们还尝试在 x86 和 arm64 架构的机器上安装部署；由于数据库、缓存、服务发现镜像齐全，项目在 arm64 架构的机器是表现出较强的可移植性；由于没有 x86 的部分依赖的实现，项目无法迁移安装到此类架构的服务器上。

6 综合评价

6.1 软件能力

经过测试，当前的软件开发基本满足了《WeBlog 个人博客系统需求规格说明书》中提出的功能和性能的要求。本软件可以较完善地实现浏览博客、查询博客、创建管理博文、评论点赞关注转发等功能，页面跳转逻辑正常；在用户使用过程中，可以正确地认知到各项功能与操作，软件易用性较好，并具有良好的可移植性，能够支持小规模的访问。

6.2 缺陷和限制

目前软件开发在如下方面还有一定的不足：
1. 安全问题：项目中关于密码的部分仍然以明文方式传输，且在数据库中直接存储用户的明文密钥，安全性保障缺失；
2. 部分高级功能，如系统内置通知、基于 ElasticSearch 的博文搜索等没有完成，将来可以根据需要进一步地完善；
3. 对用户权限的管理还有待进一步完善，从各 API 接口分别管理权限转变为统一管理；
4. 前端界面在某些尺寸的浏览器下布局会产生混乱。

6.3 建议

1. 安全性方面；在传输上，前端应该对密码加密；在存储上，应对用户密码加盐处理；
2. 功能方面：可以考虑在精力允许的前提下进一步开发；
3. 权限方面：寻找更合适的权限管理框架；
4. 界面方面：完善并适配相关相关动态调整前端组件界面显示的代码。

6.4 结论

项目开发符合预期目标，能够交付使用。
项目仍有进一步增加功能、优化性能的空间，若时间允许，可以考虑进一步开发和优化。