实验 2 系统功能测试 报告

系统名称：KuangStudy（软件测试实验平台）
被测地址：`http://211.87.232.162:8080/`

1. 系统功能测试分析思路

1.1 明确质量要求和测试目标

- 质量要求：确保KuangStudy平台的各项核心功能（如登录、问答、博客、资源下载等）能够稳定、正确、安全地运行，满足用户的基本使用需求。
- 测试目标：发现系统在功能实现、输入校验、权限控制、异常处理和安全防护等方面的缺陷，验证各模块的功能完整性和健壮性，提升系统的用户体验和安全性。

1.2 功能分析

- 功能范围界定：根据系统首页和各模块页面，梳理出主要功能点，包括导航跳转、登录注册、问答发布与管理、博客发布与管理、资源下载、公告查看、个人信息管理等。
- 用例优先级划分：优先测试高风险和高频使用的功能（如登录、内容发布、下载），其次覆盖边界场景和异常输入（如XSS、超长输入、未授权访问）。
- 排除项说明：不测试第三方服务、友情链接等非核心功能，聚焦于平台自身实现的功能模块。

1.3 分析可用的时间、方法和工具

- 时间安排：根据实验周期，合理分配手动测试和自动化测试的时间，优先保证主流程和高优先级用例的覆盖。
- 测试方法：采用等价类划分、边界值分析、异常路径测试、权限测试和安全性测试等方法，兼顾正向和负向场景。
- 测试工具：手动测试结合自动化测试（Selenium+Java），利用断言和脚本自动执行高频、重复性强的用例，提高测试效率和准确性。

1.4 针对测试对象，采用方法完成测试设计

- 测试用例设计：针对每个功能点，结合需求和实际页面，设计覆盖正常流程、异常输入、权限边界和安全攻击的测试用例，明确操作步骤、预期结果和通过标准。

- 测试数据准备：准备典型的有效数据、无效数据、边界数据和恶意数据（如脚本注入、超长字符串等），确保测试的全面性和针对性。
- 测试覆盖说明：确保每个功能点至少有一条正向用例和一条异常用例，重点模块（如登录、内容发布）增加安全性和边界条件测试。

---

2. 具体功能分析与测试范围

2.1 功能点清单

根据首页分析，系统主要功能模块如下：		
模块	**功能点**	**测试范围**
导航系统	首页、问答、博客、资源库、公告链接跳转	验证所有链接是否有效
登录功能	登录按钮跳转、表单验证，注册	测试登录页功能（需手动访问 `/toLogin`）
问答模块	问题提交、列表展示，问题回答	输入验证、XSS防护（需实际操作）
博客模块	博客发布、编辑	权限控制（未登录禁止访问）
资源库	文件下载	下载链接有效性
通用功能	返回顶部按钮、响应式布局	跨浏览器兼容性

2.2 排除项

- 友情链接（实验要求不测试）。
- 第三方服务（如Bootstrap CDN加载）。

---

3. 测试用例设计

1. 通用设计原则

- **每个功能点**至少包含：
  - 1个正向测试用例
  - 1个边界值用例
  - 1个异常情况用例
- **输入字段**必须验证：
  - 空值提交
  - 超长输入
  - 特殊字符
  - SQL/XSS注入尝试

基于等价类划分和边界值分析，设计以下高发现率的测试用例：

用例编号	测试场景	操作步骤	预期结果	实际结果	是否通过
TC001	导航栏-问答页跳转	1. 访问首页 2. 点击"问答"链接	若未登录跳转到//toLogin； 若登录跳转到 `/question`，状态码200	未登录跳转到//toLogin； 登录跳转到 `/question`，状态码200	是
TC002	未登录访问博客编辑页	1. 直接输入URL：`/blog/write`	返回403或跳转到登录页	返回状态码500，没有此页面	否
TC003	登录-无效密码	1. 访问 `/toLogin` 2. 输入正确用户名+错误密码 3. 提交	提示"用户名或密码错误"	没有提示，页面重置， 需要重新输入用户名和密码	否
TC004	问答页-提交空标题	1. 登录后进入 `/question` 2. 提交标题为空的问题	提示"标题不能为空"	问题标题字段提示 "请填写此字段"	是
TC005	资源库-下载失效链接	1. 访问 `/download` 2. 点击任意下载链接	文件正常下载（或提示"资源不存在"）	没有下载按钮， 点击提取码按钮既不会跳转也不会复制， 下载连接需要自己复制，没有跳转按钮	否
TC006	博客页-超长内容提交	1. 登录后进入 `/blog` 2. 输入1000字符以上的内容提交 3.输入1000字符以上的标题提交	提示"内容超出限制"	内容可以使用大量字符 标题使用1000字符，返回500状态码，页面报错	否
TC007	登录-XSS攻击尝试	1. 在登录页用户名字段输入： `<script>alert(1)</script>` 2. 提交	过滤脚本或提示"输入非法"	成功防御攻击	是
TC008	移动端-导航菜单折叠	1. 缩小浏览器窗口至移动端尺寸 2. 点击汉堡菜单	菜单正常展开/折叠	菜单正常展开	是
TC009	公告页-未登录访问	1. 直接访问 `/say`	允许访问（或跳转登录页）	跳转登录页	是
TC010	返回顶部按钮功能	1. 滚动页面到底部 2. 点击"返回顶部"按钮	页面平滑滚动至顶部	页面平滑滚动至顶部	是
TC011	记住密码功能	登录时在"记住密码"上打勾	下一次登录密码被记住	注销账户重新登录时，用户名和密码框为空	否
TC012	博客内容嵌入XSS脚本	登录后进入博客发布页 在内容中输入： `<script>alert("XSS")</script>` 发布文章并访问该博客	脚本不执行，内容被转义或过滤	脚本不执行，显示内容为空	是
TC013	Markdown代码块嵌入JS	在Markdown中输入： `html\n<script>alert(1)</script>\n` 发布并访问	代码块仅显示文本，不执行脚本	代码块仅显示文本，不执行脚本	是
TC014	图片标签注入JS事件	插入图片标签： 1. `<!-- 样本1：利用onerror事件 -->`	图片加载失败，但onerror事件不触发 图片成功加载，但是不跳转链接	onerror事件成功，网站卡死 onload事件成功，跳转恶意网站	否

用例编号	测试场景	操作步骤	预期结果	实际结果	是否通过
		<img src="x" style="display:none" onerror="while(1) alert('死循环弹窗！')"> 2. <!-- 样本2：利用onload事件（需图片能加载） --> <img src= "https://picsum.photos/200" onload= "document.location='https://www.sdu.edu.cn/'"> 发布并访问			
TC015	问答功能测试	1.发布问题 2.接收回答 3.设置已解决	当答案符合问题时，提问者可以设置已解决	只要有回答就是已解决	否
TC016	个人信息更新 头像更新	1.点击个人资料进行更新 2.点击头像上传	个人资料更新为新输入的 头像更新为上传的	个人资料虽已成功保存， 但昵称未同步更新至博客、问答等需展示昵称的页面， 仍显示账户名。 头像无法上传	否
TC017	重复注册	利用资料重复注册	多次检查，防止并发操作攻击	只要用户名不重复就可以注册	否

4. 缺陷记录

缺陷记录（Bug List）

4.1 高优先级缺陷（安全/功能阻断）

缺陷ID：BUG-001
缺陷标题：图片标签注入导致XSS攻击
关联用例：TC014
严重程度：高
重现步骤：
1. 登录后访问`/blog/write`
2. 在内容中输入：
   ```
   <img src="x" onerror="while(1) alert('死循环弹窗')">
   ```
3. 发布并访问该博客
实际结果：

* 触发无限弹窗导致页面卡死
预期结果：
* 应过滤`onerror`等危险属性
* 或转义为纯文本显示
修复建议：

1. 后端使用HTML净化库（如Jsoup）
2. 前端禁用`innerHTML`直接渲染

缺陷ID：BUG-002
缺陷标题：超长标题导致服务器500错误
关联用例：TC006
严重程度：高
重现步骤：
1. 在博客发布页输入1000+字符标题
2. 点击发布
实际结果：
- 返回HTTP 500状态码
预期结果：
- 前端限制输入长度（如maxlength=100）
- 后端返回400错误并提示"标题过长"

4.2 中优先级缺陷（功能异常）

缺陷ID：BUG-003
缺陷标题：资源库下载功能失效
关联用例：TC005

严重程度：中
重现步骤：
1. 访问`/download`
2. 尝试点击下载链接
实际结果：
- 无下载按钮，需手动复制链接
预期结果：
- 应提供显式下载按钮或自动触发下载

缺陷ID：BUG-004
缺陷标题：问答模块"已解决"逻辑错误
关联用例：TC015
严重程度：中
重现步骤：
1. 用户A提问
2. 用户B提交任意回答
3. 用户A标记"已解决"
实际结果：
- 任意回答均可标记为已解决
预期结果：
- 仅提问者可手动设置有效答案

4.3 低优先级缺陷（体验问题）

缺陷ID：BUG-005
缺陷标题：登录失败无错误提示
关联用例：TC003

严重程度：低

重现步骤：

1. 输入正确用户名+错误密码
2. 点击登录

实际结果：

- 页面刷新且无提示

预期结果：

- 保留用户名
- 显示"密码错误"提示

缺陷ID：BUG-006
缺陷标题：记住密码功能失效
关联用例：TC011
严重程度：低
重现步骤：
1. 勾选"记住密码"后登录
2. 退出重新访问
实际结果：
- 密码框未自动填充
预期结果：
- 应通过Cookie保存加密密码

缺陷统计

严重等级	数量	缺陷ID
高	2	BUG-001, BUG-002
中	2	BUG-003, BUG-004

严重等级	数量	缺陷ID
低	2	BUG-005, BUG-006

---

5. 自动化测试脚本（Selenium+Java）

5.1 脚本源代码

KuangStudyIntegratedTest.java

5.2 脚本开发问题及解决方案

问题1：动态元素加载延迟导致定位失败

- **现象**：测试运行时抛出 `NoSuchElementException`，元素未找到。
- **原因**：页面元素异步加载，脚本执行时元素尚未渲染完成。
- **解决**：

```
// 使用显式等待（最长10秒）
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
```

```
WebElement element =
wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector(".dynamic-
element")));
```

- **应用场景**：所有涉及动态加载页面的测试用例（如TC006、TC012）。

问题2：Chrome版本与WebDriver不匹配

- **现象**：初始化驱动时抛出 `SessionNotCreatedException`。
- **原因**：本地Chrome浏览器版本与ChromeDriver版本不一致。
- **解决**：

1. 检查Chrome版本：`chrome://settings/help`
2. 下载对应版本的ChromeDriver：ChromeDriver官网
3. 更新测试代码中的驱动路径：

```
System.setProperty("webdriver.chrome.driver",
"path/to/matching/chromedriver.exe");
```

- **影响用例**：所有基于Chrome的测试用例（如TC001-TC016）。

问题3：弹窗意外中断测试执行

- **现象**：`UnhandledAlertException`（如TC014中的"死循环弹窗"）。
- **原因**：XSS测试触发浏览器弹窗，未被处理。
- **解决**：

```
try {
    Alert alert = driver.switchTo().alert();
    driver.quit();

     // 重新初始化浏览器
    ChromeOptions options = new ChromeOptions();
    options.addArguments("--remote-allow-origins=*");
```

```
        driver = new ChromeDriver(options);
        driver.manage().window().maximize();
        // 重新导航到基准URL
        driver.get(BASE_URL);
    } catch (NoAlertPresentException e) {
        // 无弹窗则继续执行
    }
```

- **关键用例**：TC014（XSS测试）、TC015（恶意脚本检测）。

问题4：文件上传功能验证失败

- **现象**：`ElementNotInteractableException`，文件输入框不可操作。
- **原因**：`<input type="file">` 被隐藏或需通过JavaScript触发。
- **解决**：本身网页缺少这个功能，不写这部分代码了
- **影响用例**：TC016（头像上传测试）。

问题5：跨页面数据同步验证失败

- **现象**：个人资料更新后，昵称未同步显示在其他页面（如博客页）。
- **原因**：前端缓存或未触发全局状态更新。
- **解决**：

```
// 强制刷新页面并验证
driver.navigate().refresh();
WebElement displayedName =
wait.until(ExpectedConditions.presenceOfElementLocated(
    By.cssSelector(".user-nickname")));
Assert.assertEquals("新昵称", displayedName.getText());
```

- **关键用例**：TC016（昵称同步测试）。

问题6：Markdown编辑器内容注入失败

- **现象**：`JavaScriptExecutor` 无法注入内容到编辑器（TC012）。
- **原因**：编辑器未完全初始化或API名称不符。

- **解决**：

```
// 等待编辑器初始化后操作
wait.until(d -> ((JavascriptExecutor)d).executeScript(
    "return typeof testEditor !== 'undefined'"));
((JavascriptExecutor)driver).executeScript(
    "testEditor.setMarkdown(arguments[0]);", "<script>alert('XSS')</script>");
```

- **关键用例**：TC006（长内容提交）、TC012（XSS测试）。

问题7：测试依赖顺序导致失败

- **现象**：TC016需先登录，但前序测试未正确登出。
- **原因**：测试用例未完全隔离。
- **解决**：

```
@BeforeEach
public void setUp() {
```

```
    driver.get(BASE_URL + "/logout"); // 确保每次测试前状态干净
}
```

- **影响范围**：所有依赖登录状态的用例（如TC011-TC016）。

问题7：无法删除测试遇到一个带有 `onerror` 事件处理程序的 `<img>` 标签，并且该事件导致死循环弹窗的博客

- **现象**：TC014，产生之后无法删除，影响学校服务器
- **原因**：死循环弹窗产生速度大于我点击速度
- **解决**：
- 禁用 JavaScript
  如果你只是想浏览网页而不希望 JavaScript 执行，可以通过禁用 JavaScript 来避免 onerror 事件触发。

> 在浏览器中禁用 JavaScript：
> 打开 Chrome 开发者工具（F12）。
> 转到“Settings”选项卡。
> 在“Preferences”下，找到“Debugger”部分。
> 禁用 JavaScript（或者使用插件如 NoScript 来控制 JavaScript 的执行）。

- **影响范围**：TC014。

问题8：Markdown编辑器无法注入长内容

- **现象**：
  `((JavascriptExecutor)driver).executeScript("testEditor.setMarkdown(arguments[0]);", longContent)` 未生效，编辑器内容仍为空。
- **原因**：
  1. 编辑器未完全初始化时调用API。
  2. 超长内容可能触发前端限制（如长度校验）。
- **解决**：

```
// 1. 确保编辑器初始化完成
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(d -> (Boolean) ((JavascriptExecutor)d).executeScript(
    "return typeof testEditor !== 'undefined' && testEditor.isLoaded();"));

// 2. 分块注入内容（避免前端限制）
String chunk = new String(new char[500]).replace("\0", "a"); // 每次注入500字符
((JavascriptExecutor)driver).executeScript(
    "testEditor.setMarkdown(arguments[0]);", chunk);
Thread.sleep(500); // 短暂等待

// 3. 追加剩余内容
((JavascriptExecutor)driver).executeScript(
    "testEditor.appendMarkdown(arguments[0]);", chunk);
```

- **关键用例**：TC006（长内容提交）、TC012（脚本注入）。

问题9：按钮状态无法正确检测

- **现象**：
  `elementToBeClickable` 误判按钮状态（如按钮实际被禁用但测试通过）。
- **原因**：
  1. 前端异步更新按钮状态（如提交后禁用按钮）。
  2. 按钮通过CSS伪类（如 `:disabled`）控制，Selenium无法直接检测。
- **解决**：

```
// 1. 显式检查disabled属性
WebElement submitBtn = wait.until(ExpectedConditions.presenceOfElementLocated(
    By.cssSelector("button[type='submit']")));
wait.until(d -> !submitBtn.getAttribute("class").contains("disabled")
    && !Boolean.parseBoolean(submitBtn.getAttribute("disabled")));

// 2. 通过JavaScript直接检查按钮状态
boolean isDisabled = (Boolean) ((JavascriptExecutor)driver).executeScript(
    "return arguments[0].disabled ||
arguments[0].classList.contains('disabled');",
    submitBtn);
Assert.assertFalse(isDisabled, "按钮应处于可点击状态");
```

- **影响用例**：TC006（提交按钮）、TC016（资料更新按钮）。

---

6. 测试环境

- **操作系统**：Windows 11
- **浏览器**：Chrome `https://storage.googleapis.com/chrome-for-testing-public/135.0.7049.114/win64/chromedriver-win64.zip`
- **工具**：Selenium WebDriver `3.141.59`, JUnit 5, JDK 17

---

7. 体会与改进建议

1. 体会

- 测试思维的提升：通过本次系统功能测试，进一步体会到测试不仅仅是“找Bug”，更重要的是站在用户和开发者的双重视角，发现潜在的风险和用户体验问题。
- 用例设计的重要性：合理的用例设计能极大提升测试效率和缺陷发现率。等价类、边界值、异常路径等方法在实际测试中非常实用。

- 自动化与手动测试结合：自动化脚本能高效覆盖大量重复性场景，但对于界面交互、异常弹窗等复杂场景，手动测试依然不可或缺。
- 安全意识增强：实际测试中发现XSS、权限绕过等安全问题，深刻认识到安全测试在功能测试中的重要地位。
- 团队协作体验：测试过程中与同学交流用例设计和缺陷复现方法，体会到团队协作和知识共享对提升测试质量的积极作用。

2. 改进建议

- 完善输入校验：建议开发团队在前后端均增加对输入内容的长度、格式和安全性校验，防止异常数据和恶意攻击。
- 优化用户提示：对于登录失败、操作异常等场景，建议增加明确的错误提示，提升用户体验。
- 增强权限控制：对敏感操作（如内容发布、编辑、删除）应严格校验用户身份，防止未授权访问。
- 自动化测试覆盖率提升：建议持续完善自动化测试脚本，覆盖更多边界和异常场景，减少回归测试的人力成本。
- 缺陷管理流程规范化：建议建立标准的缺陷报告和跟踪机制，便于开发和测试团队高效沟通和问题闭环。

3. 其他体会感想

- 测试带来的成就感：每当发现一个隐藏较深的Bug并推动修复，都会有很强的成就感，体会到测试工作对软件质量的直接贡献。
- 对开发的反向促进：测试过程中对系统架构和代码逻辑有了更深入的理解，也反向促进了自己对开发工作的认识和能力提升。
- 对细节的关注：测试工作锻炼了自己对细节的敏感度，学会了从不同角度审视同一个功能点，发现更多潜在问题。
- 持续学习的动力：在测试过程中遇到新技术、新工具（如Selenium、自动化脚本），激发了持续学习和探索的兴趣。
- 职业素养提升：通过规范的测试流程和文档编写，提升了自己的职业素养和团队协作能力，为今后从事相关工作打下了基础。

最终交付物：

- 测试用例表（含实际结果）。
- 自动化脚本（Java文件）。
- 测试总结报告（本文档）。