**White-Jotter（前后端分离的图书管理系统）**

**测试报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 日 期 | 版 本 | 说明 | 作 者 |
| 2025.9.10 | V1.1 | 完成引言 | 王祖鹏 |
| 2025.9.20 | V1.2 | 完成测试概要 | 王祖鹏 |
| 2025.9.30 | V1.3 | 完成测试总结 | 王祖鹏 |
| 2025.10.10 | V1.4 | 初稿完成 | 王祖鹏 |
| 2025.10.10 | V1.5 | 终稿完成与文档排版 | 阮泽同/王琢玉 |

**目 录**

[1引言 3](#_Toc35958945)

[1.1编写目的 3](#_Toc35958946)

[1.2背景 3](#_Toc35958947)

[1.3定义 3](#_Toc35958948)

[1.4参考资料 3](#_Toc35958949)

[2测试概要 4](#_Toc35958950)

[2.1动态测试内容和方法 4](#_Toc35958951)

[2.2静态测试内容和方法 5](#_Toc35958954)

[3 测试环境 7](#_Toc35958957)

[4测试总结 7](#_Toc35958958)

[4.1动态测试情况说明 7](#_Toc35958959)

[4.2缺陷情况说明 8](#_Toc35958960)

[4.3静态扫描情况说明 9](#_Toc35958961)

[4.4建议 11](#_Toc35958962)

[4.5评价 11](#_Toc35958963)

# 1引言

## 1.1编写目的

本文档旨在对White-Jotter（前后端分离的图书管理系统）的系统测试过程进行总结和分析，说明测试执行情况、缺陷发现和修复情况，评估当前版本软件的质量，并提供是否可交付的结论 。预期的读者范围包括：项目经理、测试经理、开发人员。

## 1.2背景的

**被测试软件系统名称**：White-Jotter

**任务提出者**：第八个组

**开发者**：第八个组

**用户**：在线图书系统访问者

**测试环境与实际运行环境的差异**：无特殊说明 。

## 1.3定义

|  |  |
| --- | --- |
| **术语** | **原词组/定义** |
| OK | Test case passed/通过 |
| NG | Test case failed/未通过 |
| POK | Partial OK/部分通过 |
| NT | Not Tested/未测试 |
| CRUD | Create, Read, Update, Delete (创建, 读取, 更新, 删除) |

## 1.4参考资料

《The art of software testing》（2012，Glenford J·Myers）

《Software testing》（2013，Ron Pattons）

software-testing-tutorial（2024，TutorialsPoint）

《软件需求清单模板.xlsx - 软件需求清单.csv》

《程序运行说明模板.docx》

《测试用例清单模板.xlsx - Test Cases测试用例.csv》

《测试用例清单模板.xlsx - Information文档信息.csv》

《缺陷报告模板.doc》

# 2测试概要

本次测试为系统测试，主要针对软件的功能、权限、性能和灵活性要求进行动态测试，以及代码的静态测试。

## 2.1动态测试内容和方法

### 2.1.1 测试内容

本次测试覆盖的功能模块包括：

(1)首页展示：广告展示、更新介绍、项目介绍功能。

(2)笔记本功能：文章发布、文章浏览功能（测试用例仅关注发布功能入口）。

(3)图书馆功能：信息预览、书籍分类、书籍搜索功能。

(4)后台管理：运行情况展示、用户管理、内容管理功能。

(5)权限管理：登录拦截、行为限制、用户注册功能（测试用例仅关注登录拦截和行为限制）。

(6)性能要求：吞吐量（响应时间）。

### 2.1.2 测试方法

针对不同的测试项，主要采用以下测试方法：

**(1)功能测试**：主要使用等价类测试、边界值测试和场景法，同时引入了AI辅助测试。

(a)等价类测试：用于验证大多数功能点，如广告展示、书籍分类、用户管理操作等。

(b)边界值测试：用于验证特定边界条件，如空内容更新编辑、信息不完整的图书展示、吞吐量临界值等。

(c)场景法：用于验证权限管理中的登录拦截功能。

(d)AI辅助测试：我们使用了不同AI辅助生成了测试用例、测试代码，以及基于指令的自动化AI测试，并辅助分析了系统缺陷，完成了AI辅助软件质量测试。

**(2)性能测试**：使用 Python 模拟并发请求的方法，对系统吞吐量下的最大等待时长进行测试，以验证其性能要求。

## 2.2静态测试内容和方法

### 2.2.1 测试内容和方法

本次静态测试主要针对 White-Jotter 系统的后端 Java 代码 进行源代码质量检查，遵循《阿里巴巴 Java 开发手册》和《Google Java Style Guide》中的主要编码规范。

测试采用 IDEA 内置代码检查工具（Code Inspection） 以及 SonarLint 插件 对系统代码进行自动化静态扫描，主要检测内容包括：

(1)命名规范：类名、方法名、常量命名是否符合驼峰命名及语义清晰原则。

(2)异常与日志处理：是否存在未捕获异常、是否正确使用日志记录代替 System.out.print。

(3)资源关闭与内存管理：数据库连接、输入流等资源是否在使用后及时关闭，避免内存泄漏。

(4)魔法值与硬编码：检查业务逻辑中是否存在未定义常量或硬编码的字符串。

(5)空指针风险：识别可能出现 NullPointerException 的高风险代码段。

通过静态扫描，系统共检测出约 35 条潜在问题，其中 30 条为低风险（命名与格式类），5 条为中风险（资源释放与异常处理问题）。所有中风险问题已在测试阶段修复。

### 2.2.2 对规范的理解

在执行静态测试过程中，主要依据以下几条开发规范，并结合实践进行说明：

**(1)命名规范（《阿里巴巴 Java 开发手册》）**

【强制】类名使用 UpperCamelCase 风格，方法名、参数名使用 lowerCamelCase 风格。

理解：命名规范是代码可读性与可维护性的基础。项目中如类 BookController、UserServiceImpl 都能直观体现模块职责。

示例：若使用 bookcontroller 或 Bookcontroller，则会造成代码审查与自动化文档生成中的不一致问题。

**(2)资源关闭规范（《Google Java Style Guide》）**

【推荐】在使用数据库连接、文件流等资源时，必须使用try-with-resources 语句自动关闭资源。

理解：在系统中，如 BookMapper 中的数据库查询操作采用try-with-resources 能有效避免连接未释放导致的内存泄漏问题。

示例：

try (Connection conn = dataSource.getConnection()) {

// 执行查询逻辑

}

这种方式比手动关闭更安全且可读性更好。

**(3)日志处理规范（《阿里巴巴 Java 开发手册》）**

【强制】禁止使用 System.out.print 输出日志，必须使用日志框架（如SLF4J）记录系统行为。

理解：日志框架支持多级别输出与统一管理，有利于后期问题定位。例如，logger.info("Book created successfully"); 代替直接输出能更好地维护生产环境稳定性。

综上，通过静态代码检查与规范化整改，White-Jotter 项目的代码质量得到显著提升，整体符合 Java 开发规范要求，为系统后续维护与扩展提供了良好基础。

# 3 测试环境

程序的运行条件如下：

|  |  |  |
| --- | --- | --- |
| **项目** | **配置/版本** | **备注** |
| 硬件环境 | CPU | i7 – 11800H |
| 硬盘 | BC711 NVMe sk hynix 512GB |  |
| 内存 | 16GB |  |
| 显示器 | 1080P |  |
| 支撑软件 | 操作系统 | windows10 |
| 开发平台 | idea2025.2.1 |  |
| 数据库 | mysql | 数据库名称：wj |
| 缓存服务 | redis5.0.14.1 |  |
| Java 环境 | jdk1.8（版本强制） |  |
| Node.js 环境 | nodejs10.24.1（版本强制） |  |
| 包管理工具 | npm6.9.0（版本强制） |  |
| 管理员账号 | admin/123 | 用于管理中心访问 |

# 4测试总结

## 4.1动态测试情况说明

本次测试共执行 80个测试用例，其中包含了由人工智能辅助生成的x个测试用例。在对部分较容易的缺陷进行修复后，测试结果统计如下：

|  |  |  |
| --- | --- | --- |
| **结果项** | **数量** | **占比** |
| OK (通过) | 76（57 + 19） | 95.00% |
| POK (部分通过) | 2 | 2.50% |
| NG (未通过) | 2 | 2.50% |
| NT (未测试) | 0 | 0.00% |
| 用例总个数 | 80 | 100.00% |

注：原始缺陷报告显示修复前有 21 个 NG，57个 OK，修复后通过回归测试，目前总 NG 数量为 2 个，总 OK 数量为 76 个.

测试结果表明，系统在以下方面符合需求：

权限控制与登录拦截：未登录用户访问管理中心时能强制跳转到登录页. 不同权限用户（系统管理员、内容管理员、测试人员）进入管理中心时，系统能正确地只加载其权限支持的模块. 后端系统能够有效拦截越权请求（如内容管理员强制发送系统管理员独有请求）.

性能吞吐量：在 10 次并发量请求的情况下，平均最大等待时长约为 1954ms ~ 2191ms. 该结果远低于需求的 5000 毫秒响应时间要求，满足系统应支持至少每秒 10 次并发量的吞吐量要求.

图书核心功能：书籍分类、正常搜索（书名/作者）、鼠标悬停展示简介、翻页等核心功能运行正常

## 4.2缺陷情况说明

通过对测试用例的执行结果进行分析，系统仍存在以下 4 个遗留缺陷（NG/POK），其中 2 个为功能性缺陷（NG），2 个为数据/安全隐患缺陷（POK/NG）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **功能模块** | **严重程度** | **缺陷标题** | **描述和影响** |
| HOME-AD-04 | 首页展示-广告展示 | 中 | 管理员无法编辑广告内容 | 管理员无法对现存广告进行编辑和查找. 广告内容以静态资源存放，后端缺少处理代码，导致前端内容无法动态维护. |
| HOME-UPD-02 | 首页展示-更新介绍 | 中 | 管理员无法编辑更新介绍内容 | 管理员无法对现存的更新介绍进行修改和增添. 更新介绍以静态资源存放在前端代码中，缺乏相应的请求和后端处理代码. |
| LIB-INFO-06 | 图书馆功能-信息预览 | 低 | 图书信息不完整时，未正确提示 | 部分图书信息不完整时，没有正确提示“信息缺失”，而是显示了错误数量的分隔符，影响用户体验. |
| ADMIN-STAT-02~06 | 后台管理-运行情况展示 | 高 | 后台管理首页统计数据为静态，无法更新 | 管理中心首页显示的各项统计数据（新用户数、注册曲线、书籍浏览量等）以静态资源存放于前端，不会实时更新网站运行情况，后端缺乏相应处理逻辑. 严重影响系统运营的可维护性. |
| LIB-SRCH-01 | 图书馆功能-书籍搜索 | 中 | 无法按照图书简介关键字进行搜索 | 用户输入图书简介中的关键字搜索时，系统错误地返回了按照图书名称搜索的结果. 影响搜索功能的准确性. |
| NOTE-ART-02 | 笔记本功能-文章发布 | 低 | 管理员文章发布存在安全隐患 | 文章管理允许插入任意类型的文件，存在安全隐患，可能导致系统被恶意文件攻击. |

## 4.3静态扫描情况说明

本次静态测试使用 SonarLint 插件（集成于 IntelliJ IDEA）及 IDEA 内置 Code Inspection 工具 对 White-Jotter 项目后端源代码进行静态扫描。扫描范围覆盖 controller、service、mapper、model、config 等主要模块，共计约 52 个 Java 文件。

**（1）扫描指标与检测内容**

静态扫描主要关注以下五类质量指标：

|  |  |  |
| --- | --- | --- |
| **检测类别** | **检查内容** | **检测目的** |
| 命名规范 | 类、方法、常量命名是否符合 Java 命名标准 | 提高代码可读性与一致性 |
| 代码复杂度 | 方法的圈复杂度（Cyclomatic Complexity）是否超过 10 | 避免出现难以维护的高耦合函数 |
| 异常与日志 | 异常是否正确捕获与记录，是否存在 System.out.print | 提高系统稳定性与问题追踪能力 |
| 资源管理 | 数据库连接、文件流是否及时关闭 | 防止资源泄漏与内存占用过高 |
| 潜在缺陷 | 未使用变量、空指针风险、重复代码 | 降低运行时错误风险 |

**（2）扫描结果统计**

静态扫描结果如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| **指标类别** | **检出问题数** | **严重程度** | **修复状态** |
| 命名规范问题 | 18 | 低 | 已全部修复 |
| 代码格式问题 | 12 | 低 | 已全部修复 |
| 异常处理问题 | 3 | 中 | 已修复 |
| 资源释放问题 | 2 | 中 | 已修复 |
| 潜在逻辑风险 | 0 | —— | —— |
| 总计 | 35 | —— | —— |

从结果可见，系统整体代码规范性较高，主要问题集中在命名与格式层面，无严重风险项。经整改后重新扫描，已无新的告警项。

**（3）问题分析与整改举例**

(a)命名不规范

原代码：public class usercontroller {}

修改后：public class UserController {}

说明：统一采用首字母大写的驼峰命名，提高模块识别度。

(b)日志输出不规范

原代码：System.out.println("book created");

修改后：logger.info("Book created successfully");

效果：通过 SLF4J 统一管理日志输出层次，方便后期调试与维护。

(c)资源关闭风险

原代码：

Connection conn = dataSource.getConnection();

Statement stmt = conn.createStatement();

修改后：

try (Connection conn = dataSource.getConnection();

Statement stmt = conn.createStatement()) {

...

}

效果：使用 try-with-resources 自动关闭资源，避免潜在内存泄漏。

**（4）结论**

通过静态代码分析，White-Jotter 项目整体结构清晰、代码规范，符合《阿里巴巴 Java 开发手册》和《Google Java Style Guide》要求。经修复后，系统无高风险或潜在运行性缺陷。代码可维护性良好，静态扫描得分达到**A级**（良好）。

## 4.4建议

针对主要的遗留缺陷，提出如下改进建议：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **缺陷编号/模块** | **缺陷标题** | **紧迫程度** | **建议的修改方法** | **负责人** |
| ADMIN-STAT | 统计数据为静态 | 高 | 后端开发：新增 API 接口获取实时统计数据；前端开发：修改管理中心首页，调用新接口实现数据动态更新。 | 代武君 |
| HOME-AD/HOME-UPD | 无法编辑广告/更新介绍 | 中 | 后端开发：新增数据库表和 CRUD 接口支持广告/更新介绍的动态管理；前端开发：开发对应的管理页面。 | 王琢玉/阮泽同 |
| LIB-SRCH-01 | 无法按简介搜索 | 中 | 后端开发：修改图书搜索的 SQL 语句和业务逻辑，确保搜索范围包含图书简介字段。 | 王祖鹏 |
| NOTE-ART-02 | 文件上传安全隐患 | 中 | 后端开发：在文件上传 API 中增加文件类型校验（白名单机制），限制可上传的文件类型为图片等安全格式。 | 刘宸一/李龙烁 |

## 4.5评价

在对部分高严重性缺陷进行修复后（如管理员权限控制漏洞、核心功能入口丢失等），系统整体稳定性和功能完整性得到了显著提升。80个测试用例中，未通过用例仅占5%（NG+POK）。

尽管仍有部分遗留问题（如后台管理数据静态、搜索功能不完善等），但考虑到绝大部分核心功能（权限管理、图书浏览/分类/搜索、文章发布等）已正常运行并满足性能要求，可以认为系统当前版本已基本满足核心业务需求，可以交付使用。