名单序号：60

武汉理工大学

《现代软件工程学》

课程报告

学生姓名 丁涛

学号 2025304939

2025 -- 2026 学年 第 1 学期

序号	评价项目		比重	得分
1	开发项目	团队完成度	30%	
2		个人贡献度	40%	
3	课程总结		20%	
4	格式规范及其他		10%	
合计				

目录

第一部分 开发项目

一、项目概述

本项目聚焦于中小型宠物医院信息化管理需求，开发了一套基于 Web 的宠物医院管理系统，通过前后端分离架构（前端 Vue3+Element Plus/TypeScript，后端 Spring Boot，数据库 MySQL 8.0），实现了用户管理、预约挂号、在线咨询、宠物管理、寄养服务、领养服务等核心功能，旨在解决传统宠物医院人工管理效率低、信息分散等问题，支持多角色协同作业与高效业务流转。

项目团队由 3 名成员组成，采用敏捷开发模式，以 8 周为总周期划分为 4 个 Sprint 迭代，明确各阶段目标与交付物。团队组织形式清晰：余文涛担任项目负责人，负责需求管理、项目规划与架构设计；王梓璇担任全栈开发工程师，负责前后端开发与数据库设计；本人（丁涛）担任测试工程师，全程负责系统质量保障工作，通过多层次测试体系与缺陷管理机制，确保系统功能完整性、稳定性与用户体验。

二、个人职责

作为项目唯一的测试工程师，本人全面负责系统质量保障全流程工作，核心职责如下：

测试策略制定：基于项目需求与架构设计，建立覆盖单元测试、集成测试、系统测试、性能测试、安全测试的多层次测试体系，明确各阶段测试目标、范围、工具与验收标准，确保测试工作有序开展。

测试用例设计：针对各功能模块（用户注册登录、预约挂号、领养申请、在线咨询等），结合业务流程与边界条件，设计详细的功能测试用例，覆盖正向场景、异常场景与极端场景，保障测试覆盖率。

多阶段测试执行：执行单元测试（协助开发验证核心逻辑）、集成测试（验证模块间接口协作）、系统测试（全流程功能验证）、性能测试（高并发场景响应能力）、安全测试（遵循 OWASP 标准），及时发现系统缺陷。

缺陷全生命周期管理：建立缺陷分级标准（致命、严重、一般、轻微），通过缺陷报告记录问题详情、复现步骤与优先级，跟踪缺陷修复进度，验证修复效果，确保闭环管理。

测试文档输出：编制测试计划、测试用例集、测试报告、缺陷统计分析报告等文档，为项目决策与后续维护提供依据。

自动化测试落地：开发自动化测试脚本，提升回归测试效率，保障迭代过程中原有功能稳定性。

质量指标监控：跟踪关键质量指标（功能完整度、测试覆盖率、缺陷修复率、系统响应时间等），及时预警质量风险，推动团队持续优化。

三、个人工作成果

1. Sprint 1（第 1-2 周：基础功能实现阶段）

完成《宠物医院管理系统测试计划》编制，明确测试范围覆盖用户管理、宠物档案管理、基础权限控制等核心模块，确定单元测试覆盖率≥80%、系统功能测试覆盖率 100% 的目标。

设计基础功能测试用例集，涵盖用户注册、登录、宠物信息添加 / 修改 / 查询等场景，共输出测试用例 120 余条，覆盖正向输入、数据合法性校验、异常处理等关键节点。

协助开发工程师进行单元测试，使用 JUnit 工具验证用户服务、宠物服务等核心模块的业务逻辑，发现并提交数据校验不严谨、权限控制缺失等轻微缺陷 3 个，均已同步修复。

2. Sprint 2（第 3-4 周：核心业务完善阶段）

执行核心功能测试，重点验证预约挂号流程、医生排班管理等关键业务，通过 Postman 工具进行接口测试，覆盖预约时间冲突检测、医生可用性校验等核心逻辑，发现严重缺陷 2 个（预约状态更新异常、医生信息查询超时）、一般缺陷 5 个，推动开发团队在 1.5 天内完成修复。

设计性能测试方案，明确测试场景（100 人并发在线预约）、指标（响应时间≤2 秒、系统无崩溃），搭建性能测试环境，为后续压力测试做好准备。

输出《Sprint 2 测试报告》，总结核心功能测试结果，提出预约流程交互优化建议 3 条，被开发团队采纳。

3. Sprint 3（第 5-6 周：增值功能开发阶段）

执行全功能回归测试，覆盖寄养服务、领养服务、用户个人中心等新增模块，结合自动化测试脚本（使用 TestNG 工具）完成核心流程回归，共执行测试用例 280 余条，测试覆盖率达 82%。

开发自动化测试脚本 15 个，覆盖用户登录、预约提交、领养申请等高频场景，减少重复测试工作量，回归测试效率提升 40%。

完成 Sprint 3 缺陷统计与分析，本阶段共发现缺陷 48 个（致命 2 个、严重 8 个、一般 15 个、轻微 23 个），推动修复 41 个，总体修复率 85.4%，

其中致命缺陷修复率 100%（平均修复时间 4 小时），严重缺陷修复率 87.5%（平均修复时间 1.2 天）。

输出《缺陷统计分析报告》，识别出 “领养申请状态流转不清晰”“寄养服务费用计算逻辑漏洞” 等共性问题，提出流程优化建议 2 条。

4. Sprint 4（第 7-8 周：优化与测试阶段）

执行压力测试与性能测试，模拟 100 人并发在线场景，验证系统响应时间（平均 1.8 秒）、稳定性（持续运行 24 小时无崩溃），满足设计要求；针对预约模块高并发下的轻微延迟问题，提出数据库索引优化建议，被采纳后响应时间缩短至 1.2 秒。

开展安全测试，遵循 OWASP 标准，检查接口权限校验、数据传输加密等情况，发现敏感信息未脱敏、接口未防重复提交等问题 3 个，推动开发团队采用 AES-256 加密算法优化数据传输，完善权限控制机制。

支持用户验收测试（UAT），协助项目团队与潜在用户沟通，收集测试反馈，整理用户体验优化建议 5 条（如简化预约操作步骤、优化咨询消息提醒），推动迭代优化。

编制《系统测试总报告》，汇总 4 个 Sprint 的测试结果，核心质量指标均达标：功能完整度 96%、系统可用性 99.2%、用户体验满意度 4.2/5.0、代码测试覆盖率 82%。

5.其他关键成果

建立标准化缺陷管理流程，规范缺陷提交、评估、修复、验证、关闭全流程，确保缺陷跟踪无遗漏，项目整体缺陷平均修复周期控制在 1.1 天，低于目标值（≤2 天）。

输出完整的测试文档集，包括测试计划 1 份、测试用例集 4 套、测试报告 5 份、缺陷统计分析报告 3 份、自动化测试脚本 1 套，为项目后续维护提供完整的质量追溯依据。

全程参与团队周例会与 Sprint 复盘，提出质量改进建议 8 条（如开发阶段引入单元测试门禁、接口文档规范化），推动团队协作效率提升，需求响应时间缩短至 18 小时（低于目标值≤24 小时）。

第二部分 课程总结

一、开发实践与课程理论的深度融合

本次宠物医院管理系统项目的开发过程，是对《现代软件工程学》课程理论的全面实践与验证。作为测试工程师，我将课程中软件过程管理、质量管理、敏捷开发等核心知识，深度融入测试全流程，实现了理论与实践的高效转化。

在软件过程管理方面，团队采用课程重点讲解的敏捷开发模式，以 8 周为周期划分为 4 个 Sprint 迭代，这种“小步快跑、持续迭代”的方式，与传统瀑布模型相比，更能快速响应需求变化。我严格遵循迭代节奏，在每个 Sprint 初期参与需求分析与任务拆解，结合课程所学的“测试尽早介入”原则，提前设计测试计划与用例，避免了开发完成后才发现核心逻辑缺陷的被动局面。例如在 Sprint 2 核心业务完善阶段，通过提前研读预约挂号流程的需求文档，我预判了“时间冲突检测”“医生可用性校验”等关键节点的测试重点，设计了 20 余条异常场景用例，最终在测试中精准发现了预约状态更新异常的严重缺陷，为开发修复争取了时间。

在质量管理方面，课程中强调的“多层次测试体系”是我开展工作的核心框架。结合项目实际，我落地了单元测试、集成测试、系统测试、性能测试、安全测试的全链路测试策略：单元测试配合开发团队使用 JUnit 工具验证核心业务逻辑，确保底层代码质量；集成测试通过 Postman 与 TestNG 工具，重点验证 RESTful API 接口的兼容性与数据传输准确性，覆盖了用户服务、预约服务等 8 个核心服务的接口交互；系统测试从用户视角出发，模拟真实业务场景，完成了从注册登录到领养完成的全流程验证；性能测试参考课程中高并发测试的方法论，模拟 100 人同时在线预约的场景，通过监控响应时间、服务器负载等指标，验证了系统的性能达标；安全测试则严格遵循课程推荐的 OWASP 标准，重点检测接口权限校验、敏感信息传输等环节，发现并推动修复了“聊天记录未脱敏”“接口未防重复提交”等安全隐患。

二、个人职责相关的特色与亮点实践

1. 测试工具的系统化应用

项目中，我整合了多款课程推荐及行业主流工具，构建了高效的测试工作流，大幅提升了测试效率与准确性：

接口测试工具：使用 Postman 创建接口测试集合，针对 /login、/appointment、/adopt 等核心接口，编写了包含正向请求、参数异常、权限不足等场景的自动

化测试脚本，支持一键执行与结果导出，累计覆盖接口 80 余个，接口测试覆盖率达 90%；

自动化测试工具：采用 TestNG 框架编写回归测试脚本，聚焦用户注册、预约提交、领养申请等高频核心流程，实现了 Sprint 迭代中的自动化回归测试，将重复测试工作量减少 40%，确保了新增功能不影响原有业务；

任务与缺陷跟踪工具：使用 Trello 看板可视化管理测试任务状态，配合 Excel 构建缺陷跟踪表，严格按照课程所学的缺陷分级标准（致命、严重、一般、轻微）对问题进行分类，明确修复优先级与责任人。在 Sprint 3 中，通过缺陷跟踪表分析发现“界面文案错误”“操作流程冗余”等轻微缺陷占比达 48%，据此向团队提出了 UI 优化建议，提升了用户体验满意度。

性能测试工具：借助 JMeter 模拟多用户并发访问场景，测试系统在高峰期的响应能力，通过调整线程数、循环次数等参数，精准定位了预约模块的数据库查询瓶颈，为开发团队的性能优化提供了数据支撑。

2. 设计模式理解与测试适配

虽然我的核心职责是测试，但课程中学习的设计模式知识，为我设计更全面的测试用例提供了重要支撑。团队在系统设计中采用了“策略模式 + 工厂模式”解耦第三方服务，定义了统一的服务接口与独立的实现类。基于对该设计模式的理解，我在测试中不仅验证了当前集成的服务功能，还额外设计了“第三方服务切换”的场景测试，通过模拟服务接口变更、连接超时等异常情况，验证了系统的扩展性与容错性，确保后续集成新的第三方服务时，无需大幅修改测试用例。

此外，针对系统的多角色权限管理设计（Spring Security 框架 +@Secured 注解），我结合课程中“权限测试”的方法论，设计了角色权限矩阵测试用例，覆盖了 ADMIN、DOCTOR、CLIENT 等 6 种角色的所有 API 访问权限组合，共发现 3 处权限控制漏洞（如普通用户可访问管理员的部门管理接口），推动开发团队完善了 SecurityConfig 配置。

3. AI 工具的辅助应用探索

在课程学习中接触到 AI 在软件工程中的应用后，我尝试将 AI 工具融入测试工作，取得了良好效果：

测试用例生成：使用 ChatGPT 辅助设计边界值测试用例，例如针对“宠物年龄”“预约时间”等字段，输入需求描述后，AI 快速生成了“年龄为负数”“预约时间为过去日期”“预约时间与医生已有排班冲突”等 10 余种异常场景，补充了人工设计的遗漏点；

缺陷分析：对于测试中发现的复杂缺陷（如预约状态流转异常），通过 AI 工具梳理业务逻辑链路，快速定位可能的代码问题节点，为开发排查提供了方向，平均缩短缺陷定位时间 30%；
测试报告撰写：利用 AI 工具辅助整理测试数据，自动生成缺陷统计图表与性能测试指标分析，使测试报告更直观、专业，提升了团队沟通效率。特别是直接生成 mermaid 语言，方便成图。

4. 缺陷管理的规范化实践

参考课程中缺陷管理的全生命周期理论，我建立了标准化的缺陷处理流程：从缺陷发现、提交报告、优先级评估，到分配开发修复、验证修复结果，再到缺陷关闭或记录，每个环节都有明确的操作规范与文档记录。在缺陷分级上，严格遵循“致命 - 严重 - 一般 - 轻微”的四级标准，例如将“系统崩溃”“核心功能无法使用”列为致命缺陷，要求 4 小时内响应修复；将“界面文案错误”列为轻微缺陷，纳入后续迭代修复清单。Sprint 3 中，共发现缺陷 48 个，通过规范化管理，最终修复率达 85.4%，其中致命缺陷修复率 100%，平均修复周期 1.1 天，低于团队设定的 2 天目标，有效保障了系统质量。

三、代码阅读总结-Loguru

1. Loguru 最打动我的是"简单好用还靠谱"。它对外接口特别简洁，不用复杂配置就能直接用，但内部设计很扎实——用专门的类封装日志相关信息，还考虑到了异常序列化这种容易出问题的细节，避免因第三方库异常导致日志系统崩溃，真正做到了"用户用着简单，底层藏着巧思"。

2. 它的可扩展性特别好。新增 JSON Lines 格式输出时，完全不用改核心代码，只要实现一个自定义 Sink 类，再挂个便捷方法就行，还能兼容原来的配置选项。这让我明白，好的代码设计就该"不折腾老功能，方便加新功能"，后续做项目拓展时可以借鉴这种思路。

3. 对我的测试工作很有启发。Loguru 的结构化日志（比如 JSON 格式）包含了时间、模块、进程等完整信息，排查问题时能快速定位，这让我意识到测试时不仅要验证"日志有没有输出"，更要关注"日志能不能帮上忙"。另外，它对边界场景的处理（比如无名模块、不可序列化的异常），也提醒我测试基础工具时，要多考虑极端情况，确保工具本身稳定靠谱。