

同伴课堂

系统测试计划

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 文件标识： | 第X组-同伴课堂- IT-PLAN |
| 当前版本： | 1.0 |
| 作 者： |  |
| 完成日期： | 2024-11-24 |

卓越工程综合实践课程

X班第X组

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| 1.0 |  |  | 2024/11/18至2024/11/24 |  |

目 录

[1. 测试范围与主要内容 4](#_Toc183946313)

[2. 测试方法 5](#_Toc183946314)

[3. 测试环境与测试辅助工具 6](#_Toc183946315)

[4. 测试完成准则 6](#_Toc183946316)

[5. 人员与任务表 7](#_Toc183946317)

[6. 缺陷管理与改错计划 7](#_Toc183946318)

# 1. 测试范围与主要内容

**1.1 功能测试**

确保各个功能模块按预期工作，验证系统的每个功能是否符合设计要求。

学生端功能

课堂：学生是否能够查看和加入课堂。

课程：学生是否能浏览课程内容、查看课程进度、查看课程资料。

答题：学生是否能够完成作业、答题，提交答案，查看答案反馈。

教师端功能

课堂：教师是否能够创建和管理课堂、查看学生参与情况、发布课堂内容等。

课程：教师是否能创建、修改课程内容，管理课程进度、发布课程资料。

发布题目：教师是否能够创建、发布题目，设置题目类型、难度等。

题库功能：教师是否能够管理题库，编辑、删除题目，组织题目分类。

**1.2 健壮性测试**

确保系统能够应对不同场景下的异常情况并稳定运行。

学生端：在网络不稳定或无网络的情况下，是否能正确提示并保持用户数据一致性。

教师端：在用户操作过程中，是否能处理并发请求、出现错误时是否能优雅地处理，避免数据丢失或界面崩溃。

后端：后端服务在高并发情况下是否能正常工作，是否能防止崩溃或超时。

**1.3 性能测试**

对系统的性能进行验证，确保在不同负载下系统表现稳定。

学生端和教师端负载测试：模拟多个用户并发操作，验证系统的响应时间和稳定性。

后端性能测试：测试接口响应时间，数据库查询效率等，特别是针对数据量大时的性能表现（例如，题库查询和课程信息加载）。

**1.4 用户界面测试**

验证前端界面的易用性、可访问性和一致性，确保用户体验良好。

学生端APP（Flutter）：界面是否符合设计规范，学生是否能轻松找到所需功能。

教师端Web（Vue）：是否符合教师使用的需求，界面是否直观、操作是否便捷。

响应式设计：确保教师端Web在不同设备上的显示效果良好，学生端APP在不同操作系统（iOS、Android）上的兼容性。

**1.5 安全性测试**

确保系统的数据安全性，防止恶意攻击和数据泄露。

身份验证与授权：检查学生端和教师端的登录、权限管理功能，确保不同用户角色的权限控制得当。

数据保护：确保用户信息、答题数据等敏感数据的传输和存储过程经过加密，防止数据泄露。

防止SQL注入与XSS攻击：验证系统是否能防止常见的安全攻击，如SQL注入、跨站脚本攻击等。

**1.6 安装与反安装测试**

验证系统的安装和卸载过程是否顺畅。

学生端APP安装与卸载：测试学生端APP的安装过程是否顺利，卸载后是否清除所有数据。

教师端Web部署：确保教师端Web应用能在目标服务器上正确部署，并且在不同环境下能顺利运行。

# 2. 测试方法

**2.1 黑盒测试**

黑盒测试是一种以用户的角度来验证系统功能的方法。测试人员不需要了解系统内部的实现细节，仅关注系统的输入和输出，测试是否满足需求规格和预期功能。它的主要特点是测试时不关注程序的内部逻辑或结构，而是完全基于功能需求。

功能测试：确保每个功能模块都按照需求正确运行。学生端的课堂功能是否能正常显示课程内容，学生是否能够提交答题，教师是否能发布题目等。

用户界面测试：检查用户界面的设计、易用性和符合性，确保用户能够方便地使用系统。学生端APP是否有清晰的导航，教师端Web界面是否能方便地进行题库管理。

兼容性测试：测试不同浏览器、操作系统或设备上的兼容性。测试教师端Web应用是否在Chrome、Firefox、Safari等主流浏览器上正常显示，学生端APP是否能在不同的Android和iOS版本上顺利运行。

错误处理测试：验证系统如何处理无效输入、网络中断等异常情况。学生提交答案时如果网络断开，系统是否能够正确提示或保存未提交的答案。

性能测试：在黑盒测试中，可以进行一定的性能验证，评估系统在不同负载下的响应速度和稳定性。

适用场景：功能测试、界面测试、兼容性测试、性能测试等。

**2.2 白盒测试**

白盒测试是一种基于代码内部逻辑结构进行测试的方法，测试人员需要了解系统的源代码和内部实现细节。白盒测试的目标是确保程序的各个部分都按照预定的逻辑工作，验证系统内部的逻辑正确性、结构完整性和代码执行路径的覆盖。：

单元测试：测试代码中的最小模块（如类、方法、函数），确保每个单元的功能按预期工作。测试学生提交答题的接口是否能够正确接收数据并保存，教师发布题目的功能是否按要求处理题目数据。

代码逻辑验证：验证程序的控制流、分支、循环、条件等逻辑是否正确。测试在教师端发布题目时，是否会按预期执行逻辑，且题目数据被正确保存到数据库中。

路径覆盖：测试所有可能的执行路径，确保每条路径都得到了执行。检查不同输入（如正常、无效数据）下，系统是否按照预定路径处理请求。

异常处理测试：确保系统在处理异常情况时能正确抛出异常、捕获错误并进行适当的处理。测试当学生端提交答题请求时，后端是否能正确处理无效输入或空数据。

集成测试：测试多个模块或系统的集成，确保系统内部模块之间的数据流和交互正确。测试教师端发布的题目是否能正确同步到学生端，验证学生答题结果是否能正确回传到后端。

优点：

细粒度检查：能够深入到代码的每一行，发现潜在的代码错误、性能瓶颈等。

高效发现逻辑错误：能够提前发现代码中的逻辑问题、边界条件问题等。

支持优化：能够发现代码中的冗余部分，帮助开发人员优化代码。

适用场景：

单元测试：验证函数、类、模块等的内部实现是否正确。

接口和集成测试：验证不同模块之间的交互和数据流是否正确。

性能测试和优化：通过分析代码的执行路径，优化性能瓶颈。

回归测试：确保修改后的代码不破坏已有功能。

# 3. 测试环境与测试辅助工具

|  |  |
| --- | --- |
| 测试环境 | 华为云CodeArt测试环境 |
| 测试辅助工具 | Apifox |

# 4. 测试完成准则

**4.1 功能性测试用例通过率达到100%**

学生端（APP）功能性测试：

课堂、课程浏览功能是否正常。

答题功能是否按要求工作，包括答题页面展示、选择题提交、分数反馈等。

教师端（Web）功能性测试：

教师端课程管理、课堂管理功能是否按预期运行。

发布题目功能是否正常，包括题目类型支持、发布流程是否符合需求。

题库管理功能是否能够正确添加、删除、编辑题目。

后端（Spring Boot）功能性测试：

API接口测试，包括学生答题、教师发布题目、课程管理等接口是否正确响应。

数据库操作是否正确，题库数据、课堂数据、成绩数据等能正确存储和查询。

**4.2 非功能性测试用例通过率达到95%**

性能测试：

在模拟多用户访问的情况下，系统是否能够保持稳定，特别是教师端发布题目和学生端答题过程中的性能表现。

系统在高并发下的响应时间是否符合要求（例如，响应时间小于2秒）。

兼容性测试：

学生端APP是否能够在不同的Android和iOS设备上正常运行。

教师端Web应用是否兼容主流浏览器（如Chrome、Firefox、Safari等）。

安全性测试：

检查后端API是否存在常见的安全漏洞，如SQL注入、XSS等。

确保用户数据（如成绩、作业）在传输和存储时是加密的，保证用户隐私。

# 5. 人员与任务表

|  |  |  |  |
| --- | --- | --- | --- |
| 人员 | 角色 | 职责、任务 | 时间 |
|  | 测试人员 | 系统测试计划撰写、黑盒测试 | 2024.11.18至 2024.11.24 |
|  | 测试人员 | 白盒测试、测试报告撰写 | 2024.11.18至 2024.11.24 |

# 6. 缺陷管理与改错计划

**6.1 缺陷管理流程**

**6.1.1 缺陷识别与报告**

缺陷报告：在测试过程中，任何测试人员发现的缺陷必须通过缺陷管理工具（如JIRA）报告。每个缺陷报告应包含以下信息：

缺陷描述：简洁明确地描述缺陷的具体表现和影响。

复现步骤：如何重现缺陷的详细步骤。

严重性等级：根据缺陷对系统功能或用户体验的影响程度，确定缺陷的优先级（如：高、中、低）。

截图/日志：提供缺陷发生时的截图或相关日志文件，帮助开发人员更快定位问题。

环境信息：缺陷发生的操作系统版本、浏览器版本、APP版本等，确保开发人员能够在相同环境下重现问题。

**6.1.2 缺陷分类与优先级排序**

分类：根据缺陷的性质，进行分类（如功能性缺陷、性能问题、安全漏洞、UI问题等）。

优先级排序：基于缺陷的影响程度、出现频率以及解决的紧迫性，将缺陷划分为不同的优先级：

P1（高优先级）：系统核心功能出现故障，影响正常使用（例如学生端无法答题、教师端无法发布题目等）。

P2（中优先级）：功能性缺陷或性能问题，系统运行不顺畅但不影响核心功能。

P3（低优先级）：UI优化问题、轻微的兼容性问题等，对用户体验有影响但不阻碍核心功能。

**6.1.3 缺陷分配与修复**

开发人员分配：每个缺陷报告会根据其分类与优先级，分配给相应的开发人员进行修复。开发人员需在规定时间内开始解决问题。

修复时限：根据缺陷的严重性等级，确定修复时限：

P1缺陷：立即处理，修复时间一般不超过24小时。

P2缺陷：2-3个工作日内修复。

P3缺陷：一周内修复或作为后续优化项处理。

**6.1.4 缺陷验证与关闭**

修复验证：开发人员修复缺陷后，测试人员需要进行验证，确保缺陷已经被修复且修复后的功能没有引入新的问题（即回归测试）。

缺陷关闭：验证无误后，缺陷报告将标记为“关闭”状态。如果缺陷无法在规定时间内修复，需重新评估优先级或推迟处理，直至下一个版本。

**6.2 改错流程**

改错流程是指开发人员修复缺陷后，如何验证和确保系统的正确性，避免引入新问题的过程。这个流程强调高效修复和回归验证，确保系统的稳定性。

**6.2.1 缺陷修复**

修复开发：开发人员根据缺陷报告的复现步骤进行缺陷定位，修改代码以修复问题。修复代码时需要考虑以下几个方面：

确保修复不影响其他功能：修改后的代码必须经过全面的单元测试，确保不引入新的问题。

考虑性能与安全性：修复时应优化代码，避免性能瓶颈和安全漏洞。

**6.2.2 回归测试**

回归测试目标：在修复缺陷后，回归测试确保修改的部分不会影响系统的其他功能。特别是，教师端、学生端、后端API等模块在修复缺陷后必须进行回归测试，确保系统稳定性。

回归测试用例：回归测试包括所有涉及修改功能的测试用例，以及一些常规的关键路径用例（例如学生端答题、教师端发布题目等）。

**6.2.3 修复验证**

修复验证：修复后的代码必须经过测试人员验证，确保问题已经解决，且系统整体功能没有受到影响。验证内容包括：修复的缺陷是否完全消失；是否出现新的回归缺陷；关键功能是否正常运作

。

**6.2.4 修复确认与发布**

确认修复：开发和测试人员在确保修复后的系统稳定并通过回归测试后，进行修复确认。确认无误后，缺陷报告可以标记为“已解决”并准备发布。

发布管理：修复完成的版本需要经过集成测试和用户验收测试，确保修复后的版本符合生产环境的要求，且没有引入新问题。最终的版本将进入发布流程。