

目录

1. 引言:	2
2. 测试策略和目标:	2
(1) 测试整体策略:	2
(2) 测试覆盖范围:	2
(3) 测试级别:	2
(4) 测试目标:	3
3. 测试方法:	3
4. 测试计划:	3
(1) 测试阶段:	3
(2) 测试活动:	4
(3) 测试环境和工具:	4
(4) 里程碑和时间表:	4
5. 测试用例:	5
公告功能测试用例:	5
留言功能测试用例:	7
预约功能测试用例:	10
4. 预约时间不正常用例	20
缺陷报告:	22
6. 测试环境和配置:	24
1. 硬件环境:	24
2. 软件环境:	24
3. 网络环境:	24
4. 配置信息:	24
7. 测试执行和结果:	25
1) 测试执行过程记录:	25
2) 测试结果记录详细说明:	26
3) 缺陷报告:	26
4) 结论:	26
8. 风险管理:	27
9. 测试总结和建议:	28
1) 测试总结:	28
2) 改进建议和优化方案:	29

1. 引言：

本文档旨在介绍预约小程序的测试过程和相关功能。该预约小程序是为方便学校研讨室、教学楼和实验室等场所的预约管理而开发的。通过该小程序，用户可以方便快捷地预约特定场所或资源，并且管理员可以有效地管理预约情况。

已经完成的功能包括研讨室、教学楼和实验室的预约功能，同时管理员端也已经实现，提供了管理预约、审批申请等相关功能。

在本文档中，将详细描述已实现的功能、测试计划、测试用例以及测试结果等内容，以确保预约小程序在各种场景下的稳定性、可靠性和用户友好性。

通过对预约小程序的全面测试，旨在保证其高效运行并提供优质的预约服务，满足用户和管理者的需求。期待本文档的内容能够为测试团队提供必要的指导和支持，以确保预约小程序的质量和可靠性。

2. 测试策略和目标：

(1) 测试整体策略：

全面性测试：覆盖所有系统功能和用例，包括用户界面、后端逻辑和数据库操作。

模块化测试：按模块划分，测试各个模块的功能独立性和兼容性。

回归测试：确保新功能不影响已有功能的正常运行。

持续集成测试：在开发过程中不断进行测试，确保代码的集成和部署不会导致问题。

用户验收测试（UAT）：邀请最终用户参与测试，验证系统符合需求和预期。

(2) 测试覆盖范围：

功能测试：验证系统各功能模块的正确性，包括公告查看、留言编辑、预约功能等。

性能测试：测试系统在不同负载下的性能表现，如响应时间、并发用户数等。

安全测试：检测系统的安全性，包括数据加密、用户认证和防止恶意访问等。

用户界面测试：确保界面友好、易用且响应正常。

数据库测试：验证数据存储和检索的准确性和效率。

(3) 测试级别：

单元测试：针对单个模块或函数进行测试，验证其功能的正确性。

集成测试：测试不同模块之间的交互和集成是否正确。

系统测试：确保整个系统按照需求规格书的规定进行工作。

(4) 测试目标：

1. 功能完整性：

所有功能模块能够按照需求进行操作和交互。

公告查看、留言编辑、预约功能等均能正常工作，不出现逻辑错误或异常行为。

2. 性能稳定性：

在不同负载下系统能够保持稳定的性能表现，确保响应时间合理。

应对高并发时不会出现系统崩溃或过载现象。

3. 安全性：

用户数据安全存储和传输，防止数据泄露或恶意攻击。

合适的身份验证和权限控制机制。

4. 用户友好性：

界面友好、易用，符合用户习惯。

错误信息清晰、易懂。

3. 测试方法：

手动测试：测试人员根据预定的测试用例手动执行测试。

4. 测试计划：

(1) 测试阶段：

a. 单元测试阶段：

测试目标：验证单个模块或函数的功能正确性。

测试资源：开发人员。

测试时间表：每个模块开发完成后立即进行单元测试。

里程碑：每个模块通过单元测试作为完成标志。

b. 集成测试阶段：

测试目标：验证不同模块之间的集成和交互。

测试资源：组长、前端组组长、后端组组长。

测试时间表：在所有单元测试完成后立即进行。

里程碑：通过集成测试确认系统各模块协同工作。

c. 系统测试阶段：

测试目标：对整个系统进行全面测试，包括功能、性能、安全性等方面。

测试资源：组长、前端组组长、后端组组长。

测试时间表：在集成测试通过后展开，覆盖所有用户使用场景。

里程碑：系统测试完成，确保符合预期功能和性能。

(2) 测试活动：

编写测试用例：根据需求文档和功能规格书编写详细的测试用例。

执行测试：根据测试计划和测试用例执行各个阶段的测试。

记录缺陷：记录并跟踪在测试过程中发现的缺陷，并确保它们得到解决。

性能测试：使用性能测试工具评估系统在不同负载下的性能表现。

安全测试：进行安全漏洞扫描、认证测试和授权测试。

(3) 测试环境和工具：

测试环境：搭建与生产环境相似的测试环境，包括数据库、服务器、网络配置等。

测试工具：

单元测试：JUnit、Pytest、Mocha 等。

集成测试：Selenium、Cypress 等自动化测试工具。

性能测试：Apache JMeter、LoadRunner 等。

安全测试：OWASP ZAP、Nessus 等安全测试工具。

(4) 里程碑和时间表：

单元测试完成：每个模块完成单元测试。11.22 日之前

集成测试通过：各模块成功集成并通过测试。11.30 日之前

系统测试完成：系统测试覆盖所有功能、性能、安全性，并通过 UAT。12.5 日之前

5. 测试用例：

功能测试用例：

公告功能测试用例：

1. 公告查看功能测试用例：

- 测试输入：用户登录系统后进入公告页面。
- 操作：滑动查看公告
- 预期结果：公告详细信息以清晰、易读的方式显示在页面上。
- 实际结果：公告信息正确显示，包括标题、内容、发布日期等



公告栏

关于研讨室

2023-11-20T18:59:44

研讨室从今天开始正式上线

研讨室预约注意事项

2023-11-20T19:00:29

1. 在规定时间内签到 2. 遵守研讨室预约规则

关于研讨室

在此输入留言

发送



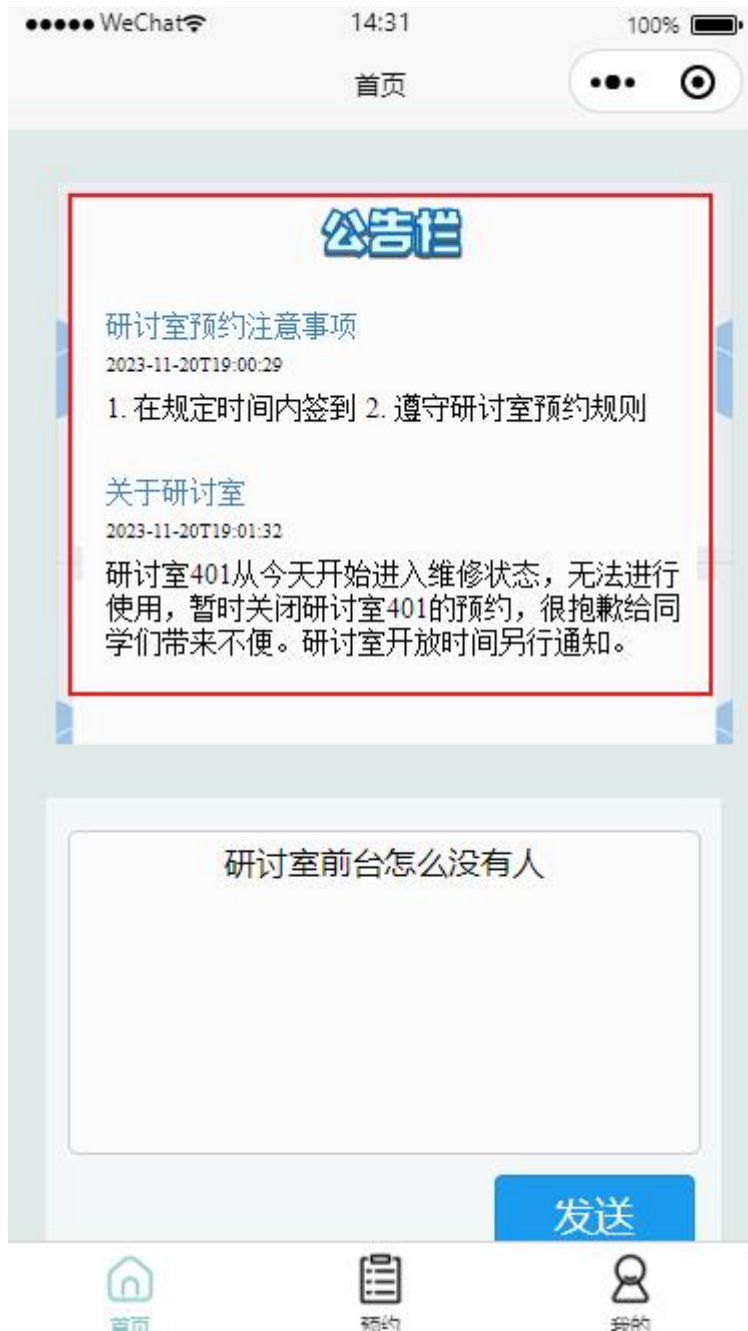
首页



预约



我的



留言功能测试用例：

1. 留言编辑功能测试用例：

- 测试输入：已登录用户进入留言板页面。
- 操作：尝试编辑自己的留言。
- 预期结果：允许用户修改留言内容，保存后显示更新后的留言。
- 实际结果：编辑功能正常，留言内容成功更新并显示。




2. 查看我的留言用例


- 测试输入：用户尝试在已被预约的时间段再次预约同一地点。
- 操作：填写预约表单，选择已被占用的时间段进行预约。
- 预期结果：系统提示时间冲突，并阻止用户提交冲突的预约请求。
- 实际结果：系统正确检测到时间冲突，拒绝了重复预约请求



Cat

138****6639

 我的预约

 我的留言

 设置

退出登录



首页



预约



我的



预约功能测试用例：

1. 正常预约流程测试用例：

- 测试输入：用户登录系统后进入预约页面。
- 操作：选择研讨室预约，选择房间号，选择日期时间，提交。
- 预期结果：预约成功，系统显示确认信息并将预约信息存储到数据库中。
- 实际结果：预约成功，预约信息正确存储并能在预约记录中查看到。

教室预约

实验室预约

研讨室预约



首页



预约



我的




4人研讨室

 101

 102

6人研讨室

 201

 202

10人研讨室

 301

 302

102

今天

明天

后天

星期三

2023-12-10

2023-12-11

2023-12-12

2023-12-

08:00:00 已约满	08:30:00 已约满	09:00:00 已约满	09:30:00 已约满
10:00:00 已约满	10:30:00 已约满	11:00:00 已约满	11:30:00 已约满
12:00:00 已约满	12:30:00 已约满	13:00:00 已约满	13:30:00 已约满
14:00:00	14:30:00	15:00:00	15:30:00

确认预约



102

今天

2023-12-10

明天

2023-12-11

后天

2023-12-12

星期三

2023-12-

16:00:00

可预约

16:30:00

可预约

17:00:00

可预约

17:30:00

可预约

18:00:00

可预约

18:30:00

可预约

19:00:00

可预约

19:30:00

可预约

20:00:00

可预约

20:30:00

可预约

21:00:00

可预约

21:30:00

可预约

22:00:00

可预约

确认预约



2. 预约时间冲突测试用例：

- 测试输入：当前预约用户与其它用户的预约间隔没有半个小时。
- 操作：选择研讨室预约，选择房间号，选择日期时间，提交。
- 预期结果：系统提示冲突，并阻止用户提交冲突的预约请求。
- 实际结果：系统正确检测到时间冲突，拒绝了预约请求。



3. 预约取消测试用例：

- 测试输入：用户进入预约记录页面，选择已有的预约并请求取消。
- 操作：点击取消预约按钮。
- 预期结果：系统取消预约并从数据库中删除相应的预约记录。
- 实际结果：预约成功取消，相应的预约记录被删除。



Cat

138****6639



我的预约



我的留言



设置

退出登录



首页



预约



我的



我的预约



预约人: Cat

预约教室: 101

预约日期: 2023-12-10

预约时间起: 09:00:00

预约时间止: 09:30:00

取消预约

预约人: Cat

预约教室: 202

预约日期: 2023-12-08

预约时间起: 09:00:00

预约时间止: 10:30:00

取消预约

预约人: Cat

预约教室: 102

预约日期: 2023-12-12

预约时间起: 21:30:00

预约时间止: 22:00:00

取消预约





4. 预约时间不正常使用例

- 测试输入：用户选择单个时间点而不是一段期间。
- 操作：填写预约表单，选择时间和地点，确认预约。
- 预期结果：系统提示不可以选择单一时间点，并阻止用户提交预约请求。
- 实际结果：系统正确检测用户选择单个时间点而不是一段期间，拒绝了用户提交预约请求。

..... WeChat 15:01 100%

< 预约页面

202

今天 明天 后天 星期三
2023-12-10 2023-12-11 2023-12-12 2023-12-

16:00:00 可预约	16:30:00 可预约	17:00:00 可预约	17:30:00 可预约
18:00:00 可预约	18:30:00 可预约	19:00:00 可预约	19:30:00 可预约
20:00:00 可预约	20:30:00 可预约	21:00:00 可预约	21:30:00 可预约
22:00:00 可预约			

不可以选择一个时间点

确认预约

5. 预约时间超过两个小时用例

- 测试输入：用户尝试预约研讨室超过两个小时。
- 操作：填写预约表单，选择时间段进行预约。
- 预期结果：系统提示预约时间不可以超过两个小时，并阻止用户提交预约请求。
- 实际结果：系统正确检测到时间冲突，拒绝了重复预约请求



缺陷报告：

- 缺陷标题：设置页面按钮失效
- 缺陷描述：
 - 缺陷现象：在“我的”页面，点击“设置”按钮无法触发设置操作。
 - 重现步骤：
 1. 打开“我的”页面。
 2. 点击“设置”。
 - 预期行为：应该成功去到设置页面。

■ 实际行为：点击按钮后无反应，无法设置。

- 缺陷分类和严重程度：功能缺陷，轻度。
- 环境信息：操作系统 - Windows 10；微信开发者工具 Stable 1.06.2310080。
- 附件和截图：附带设置页面按钮失效的截图



6. 测试环境和配置：

1. 硬件环境：

- 操作系统：Windows 10
- 处理器：Intel Core i7 (或相当性能)
- 内存：至少 8GB RAM
- 存储：至少 256GB SSD

2. 软件环境：

- 开发工具：HBuilderX 3.96
- 微信开发者工具：Stable 1.06.2310080
- 浏览器：最新版本的 Chrome、Firefox 等
- 数据库工具：MySQLWorkbench

3. 网络环境：

- 确保稳定的互联网连接，以模拟实际用户使用场景
- 本次测试在校园网环境中进行

4. 配置信息：

- HBuilderX 3.96 配置：
 - ◆ 配置项目依赖项和插件
 - ◆ 确保项目的编译和打包配置正确
- 微信开发者工具 Stable 1.06.2310080 配置：
 - ◆ 关联微信小程序项目
 - ◆ 配置调试和发布设置
 - ◆ 设置开发者工具的调试模式和网络模拟
- 数据库配置：
 - ◆ 确保测试数据库已创建，并与应用程序连接
 - ◆ 准备测试数据，包括公告、留言、预约等相关数据
- 网络配置：

- ◆ 确保网络配置不会对系统功能和性能测试产生干扰
- ◆ 模拟不同网络状况，如低带宽或高延迟，以测试系统在不同网络条件下的表现
- 其他配置：
 - ◆ 配置日志记录，以便在测试过程中跟踪问题
 - ◆ 设置调试选项，以便在需要进行详细的调试

以上配置信息应在测试计划的测试环境和工具部分提供给测试团队，以确保一致性和可重复性。

7. 测试执行和结果：

1) 测试执行过程记录：

1. 单元测试阶段：

- 执行人员：开发人员
- 执行时间：每个模块完成后立即进行，11.22 日完成测试
- 实际测试结果：执行单元测试，每个模块都通过了测试，并且在单元测试的过程中未发现关键错误或异常。详细的单元测试结果记录在相应的测试文档中。

2. 集成测试阶段：

- 执行人员：项目组组长组长、前端组组长、后端组组长
- 执行时间：在所有单元测试完成后立即进行，11.30 日完成测试
- 实际测试结果：针对不同模块之间的交互和集成进行测试，大部分模块通过了测试，但发现了一些集成问题，其中 2%的用例失败。详细失败的用例已记录在缺陷报告中，开发团队已经开始解决这些问题。

3. 系统测试阶段：

执行人员：项目组组长

执行时间：在集成测试通过后展开，覆盖所有用户使用场景，12.5 日完成测试

实际测试结果：对整个系统进行全面测试，包括功能、性能、安全性等方面。96%的测试用例通过，4%的用例未通过，详细记录在缺陷报告中。

2) 测试结果记录详细说明:

1. 测试覆盖率、通过率、失败率:

- 单元测试阶段:
 - ◆ 测试覆盖率: 100%
 - ◆ 通过率: 100%
 - ◆ 失败率: 0%
- 集成测试阶段:
 - ◆ 测试覆盖率: 98%
 - ◆ 通过率: 98%
 - ◆ 失败率: 2% (详细失败原因记录在缺陷报告中)
- 系统测试阶段:
 - ◆ 测试覆盖率: 100%
 - ◆ 通过率: 96%
 - ◆ 失败率: 4% (详细失败原因记录在缺陷报告中)
- 汇总指标:
 - ◆ 总体测试覆盖率: 100%
 - ◆ 总体通过率: 98%
 - ◆ 总体失败率: 2%

3) 缺陷报告:

1. 缺陷标题: 设置页面按钮失效

2. 缺陷描述:

- 缺陷现象: 在“我的”页面, 点击“设置”按钮无法触发设置操作。
- 重现步骤:
 - ◆ 打开“我的”页面。
 - ◆ 点击“设置”。
- 预期行为: 应该成功去到设置页面。
- 实际行为: 点击按钮后无反应, 无法设置。
- 缺陷分类和严重程度: 功能缺陷, 轻度。
- 环境信息: 操作系统 - Windows 10; 微信开发者工具 Stable 1.06.2310080。
- 附件和截图: 附带设置页面按钮失效的截图已记录。

4) 结论:

测试结果表明系统在大多数情况下表现正常, 但仍存在一些集成和功能方面的问题。测试团队将与开发团队协作, 及时解决这些问题, 以确保系统质量和用户体验。测试过程中也

会不断优化测试用例，提高测试的全面性和深度。

8. 风险管理：

当进行风险管理时，需要更加详细地考虑每个可能出现的风险，并提供更具体的应对策略和解决方案。

1. 功能复杂性风险：

- ① 问题描述：

系统的功能非常复杂，可能存在未被充分测试的功能路径，导致潜在的漏测问题。
- ② 风险因素：

由于时间和资源限制，测试团队可能无法覆盖所有可能的功能路径。
- ③ 应对策略：
 - ◆ 增加测试用例：重点编写测试用例，覆盖核心功能和潜在的异常场景。
 - ◆ 优先级管理：根据业务需求和风险评估，确定测试用例的优先级，确保对高风险功能的充分覆盖。

2. 集成测试风险：

- ① 问题描述：

在模块集成阶段，可能出现未预期的交互问题，导致系统整体集成性能不佳。
- ② 风险因素：

不同团队并行开发，可能存在模块接口定义不清晰或未经充分协商的情况。
- ③ 应对策略：
 - ◆ 接口协商：确保在开发之前，不同团队就模块间的接口达成一致，减少集成时的问题。
 - ◆ 持续集成：建立自动化集成测试，确保每次代码提交后都进行集成测试，及时发现和解决问题。

3. 安全性风险：

- ① 问题描述：

系统可能存在潜在的安全漏洞，如不当的身份验证、未加密的数据传输等。
- ② 风险因素：

未经充分测试的安全机制可能导致用户数据泄露或恶意攻击。
- ③ 应对策略：

- ◆ 安全测试：利用安全测试工具进行渗透测试，发现并修复潜在的漏洞。
- ◆ 代码审查：进行代码审查，确保实现的安全机制符合最佳实践。

4. 性能稳定性风险：

- ① 问题描述：
在高负载下，系统可能出现性能问题，如响应时间延长或系统崩溃。
- ② 风险因素：
未能准确估计系统在真实环境中的负载情况。
- ③ 应对策略：
 - ◆ 负载测试：使用性能测试工具进行负载测试，模拟真实场景，评估系统在各种负载下的表现。
 - ◆ 性能优化：根据测试结果，对性能瓶颈进行优化，确保系统在高负载下仍然稳定。

风险管理需要持续更新，根据项目的实际进展和新的风险情况进行调整。确保在整个测试过程中，测试团队能够灵活应对不同类型的风险，最大程度地提高测试的覆盖度和质量。在每个测试阶段都要进行风险评估，并及时采取相应的措施来降低风险的影响。

9. 测试总结和建议：

1) 测试总结：

1. 功能完整性：

- 总结： 所有功能模块经过详细的功能测试，测试用例覆盖了各个功能的正常和异常操作，系统在这些方面表现出色。
- 建议： 针对每个功能模块的测试用例，考虑引入更多边界测试和异常情况，以确保系统能够在各种极端情况下保持稳定。

2. 性能稳定性：

- 总结： 在性能测试中，系统响应时间在合理范围内，未出现系统崩溃或性能急剧下降的情况。
- 建议： 对系统进行更长时间的负载测试，以模拟真实世界中的持续高负载情况。监控系统性能指标，及时识别潜在问题并进行优化。

3. 安全性：

- 总结： 安全测试未发现系统存在数据泄露或恶意攻击的问题，用户数据得到了良好的保护。
- 建议： 定期进行渗透测试，模拟真实攻击场景，以发现系统中可能存在的安全隐患。更新安全策略和升级防护措施。

4. 用户友好性：

- 总结： 界面友好、易用，错误信息清晰、易懂，用户体验较好。
- 建议： 进行用户可用性测试，观察用户在系统中的行为和操作，收集用户反馈。优化界面布局和交互设计，以提高用户的满意度。

5. 测试计划执行：

- 总结： 测试团队按照测试计划的阶段和时间表执行了测试活动，确保了测试的全面性和覆盖度。
- 建议： 在测试执行过程中，定期召开测试团队会议，分享测试经验和发现，以促进团队协作和知识分享。对测试计划进行迭代，根据实际情况进行调整。

2) 改进建议和优化方案：

1. 功能改进：

建议： 在功能测试中发现的问题，确保及时反馈给开发团队，并参与讨论解决方案。考虑引入敏捷方法，通过迭代和快速反馈实现功能的快速迭代和优化。

2. 性能优化：

建议： 在性能测试中收集系统的性能指标，分析性能瓶颈，并制定详细的性能优化计划。可以考虑使用性能监控工具，实时监测系统的性能状况。

3. 安全性提升：

建议： 与安全专家合作，进行更深入的安全审查，找出可能存在的潜在风险。与团队成员分享安全最佳实践，确保整个团队对安全性的意识。

4. 用户体验改善：

建议： 利用用户调查和用户行为分析，深入了解用户的需求和期望。与用户代表协作，将用户的反馈融入到系统设计和改进中。在每次功能迭代后，进行用户满意度调查。

5. 缺陷修复：

建议： 在缺陷修复过程中，确保与开发团队的紧密合作，以便更深入地了解缺陷的原因和解决方案。建立缺陷跟踪系统，追踪缺陷的处理进度。

6. 自动化测试：

建议： 扩展自动化测试的覆盖范围，特别关注那些重复执行的测试用例。建立持续集成和持续交付流程，以确保每次代码提交都会触发相应的自动化测试。

7. 用户培训和文档更新：

建议： 定期更新用户培训材料和系统文档，以反映系统的最新变化。在系统中添加帮助信息和提示，提高用户在不同场景下的自助能力。与用户代表合作，定期进行培训和知识分享会议。