丰收宝盒

测试计划

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <11/9/2024> | <1.0> | 大致框架 | 吴卓宇 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 背景 4

1.3 范围 4

1.4 项目标识 6

2. 测试需求 7

3. 测试策略 8

3.1 测试类型 8

3.1.1 数据和数据库完整性测试 8

3.1.2 功能测试 9

3.1.3 业务周期测试 11

3.1.4 用户界面测试 14

3.1.5 性能评价 16

3.1.6 负载测试 18

3.1.7 强度测试 19

3.1.8 容量测试 21

3.1.9 安全性和访问控制测试 23

3.1.10 故障转移和恢复测试 26

3.1.11 配置测试 29

3.1.12 安装测试 31

3.2 工具 33

4. 资源 34

4.1 角色 34

4.2 系统 36

5. 项目里程碑 37

6. 可交付工件 38

6.1 测试模型 38

6.2 测试日志 38

6.3 缺陷报告 38

7. 附录 A：项目任务 39

测试计划

# 简介

## 目的

该计划将明确列出“丰收宝盒”高品质农产品甄选平台的各个功能模块，包括用户端和管理端的关键功能，如SKU管理、订单处理、个人信息管理、支付流程等。本测试计划包括用户界面测试、性能测试、安全性测试、数据库完整性测试等多种类型的测试，确保平台在功能、性能和安全性方面都达到预期目标。针对不同的测试需求，推荐分别采用功能测试、负载测试、强度测试、访问控制测试等策略，确保系统能够在各种场景下正常运行，并满足高并发用户的需求。试资源包括测试人员、测试工具、服务器、测试数据库等，测试团队将按照测试计划中的里程碑安排，分阶段进行测试工作。可交付元素包括测试报告、测试日志、缺陷报告、系统性能评估报告等，所有测试结果都将通过文档记录并进行总结，以确保问题得到解决并达到预期的产品质量。

## 背景

“丰收宝盒”是一个专为农民设计的高品质农产品电商平台，主要通过微信小程序提供服务。该平台旨在解决传统农产品销售渠道低效和不透明的问题，帮助农民通过直接面向消费者销售农产品，提高收入并塑造品牌价值。

该平台的主要功能包括商品的浏览与购买、SKU/SPU管理、订单处理、支付管理、用户个人信息管理、商家员工管理以及数据统计与展示。通过平台，用户能够方便地浏览和选购农产品，而商家则能够管理库存、订单、商品分类和销售数据。

平台的架构分为用户端和管理端两部分。用户端主要面向消费者，提供商品展示、购物车管理、订单支付等功能；管理端主要面向商家，提供商品信息维护、订单跟踪、员工管理等功能。平台依赖微信支付系统以及小程序的正常运行，确保顺畅的支付和购物体验。

## 范围

测试阶段：

* **单元测试**：确保系统各个功能模块的基本功能运行正常，涉及商品浏览、订单管理、SKU/SPU管理等功能模块。
* **集成测试**：测试模块之间的数据流通性与功能的整合，确保不同模块的交互准确无误。
* **系统测试**：针对整个系统的功能和性能，验证其能否支持预期的工作量和并发操作。
* **验收测试**：模拟用户使用场景，确保系统能够达到用户期望，并符合需求规格说明书中的要求。

测试类型：

· **功能测试**：验证系统的各个功能模块是否按照设计正常运行，包括用户端的商品浏览、下单、支付，管理端的库存管理、订单处理等。

· **性能测试**：通过负载和强度测试，评估系统在高并发操作时的响应速度和稳定性，确保系统能支持1000名商家和10000名用户同时在线操作。

· **安全性测试**：测试平台的访问控制和数据保护措施，确保用户和商家信息的安全，特别是支付流程中的数据加密和用户身份验证。

· **用户界面测试**：确保用户界面的友好性和易用性，检查平台是否适应不同用户群体的使用习惯，并确保界面显示与设计一致。

测试对象：

· **接受测试的特性和功能**：包括商品管理、订单处理、用户信息管理、SKU/SPU管理、支付系统、数据统计等核心功能。

· **不接受测试的特性和功能**：第三方依赖系统（如微信支付）的内部功能将不进行单独测试，假设其正常运行

假设：

· 系统依赖微信小程序和微信支付的正常运行。

· 农民用户能够准确填写商品信息，确保库存和商品描述的正确性

风险与意外事件：

· 微信支付功能或小程序平台的异常运行可能影响系统测试的完整性。

· 商家未能及时更新商品信息或库存，可能导致数据测试结果不准确​

约束：

· 系统需在微信小程序环境和浏览器中运行，因此所有测试必须基于该环境。

· 支付安全性依赖于微信支付平台的加密和认证机制

## 项目标识

下表列出了制定*测试计划*所用的文档，并标明了文档的可用性：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **文档 （版本/日期）** | **已创建或可用** | **已被接受或已经过复审** | **作者或来源** | **备注** |
| 需求规约 | 🗹 是 🞏 否 | 🗹 是 🞏 否 | 侯杰 |  |
| 功能性规约 | 🗹 是 🞏 否 | 🗹 是 🞏 否 | 侯杰 |  |
| 用例报告 | 🗹 是 🞏 否 | 🗹 是 🞏 否 | 侯杰 |  |
| 项目计划 | 🗹 是 🞏 否 | 🗹 是 🞏 否 | 魏汝鑫 |  |
| 设计规约 | 🗹 是 🞏 否 | 🗹 是 🞏 否 | 魏汝鑫 |  |
| 原型 | 🞏 是 🗹 否 | 🞏 是 🞏 否 |  |  |
| 用户手册 | 🞏 是 🗹 否 | 🞏 是 🞏 否 |  |  |
| 业务模型或业务流程 | 🞏 是 🗹 否 | 🞏 是 🞏 否 |  |  |
| 数据模型或数据流 | 🞏 是 🗹 否 | 🞏 是 🞏 否 |  |  |
| 业务功能和业务规则 | 🞏 是 🗹 否 | 🞏 是 🞏 否 |  |  |
| 项目或业务风险评估 | 🞏 是 🗹 否 | 🞏 是 🞏 否 |  |  |

# 测试需求

功能性需求：

商品管理

* SKU/SPU管理：确保商家能够准确地添加、修改和管理商品的SKU和SPU信息，包括库存数量、商品属性等。
* 商品分类管理：验证商品分类是否清晰，确保商品展示符合用户浏览需求。

订单处理

* 订单创建与修改：确保用户能够正常创建订单，并且商家能够跟踪、修改订单状态。
* 支付流程：确保微信支付功能正常集成，支付过程流畅、可靠。
* 订单跟踪：用户能够实时查看订单状态，并在订单完成后进行评价。

用户操作

* 用户信息管理：用户能够编辑和维护个人信息，包括联系方式和收货地址。
* 购物车管理：确保用户能够将商品加入购物车，并在购物前进行有效管理。

管理端功能

* 员工管理：验证商家能够添加、修改员工信息，并设置权限管理。
* 数据统计和展示：确保系统能够准确分析和展示销售数据、订单数据，为商家提供有效的决策支持。

非功能性需求：

性能需求：

* 系统必须支持至少1000名商家和10000名用户同时在线，确保系统响应速度在高负载下依然保持良好。

安全性需求：

* 支持微信用户的身份验证，确保用户个人信息和支付信息加密传输，避免数据泄露 。

可用性需求：

* 系统需提供简洁、直观的用户界面，确保农民和消费者能够轻松操作，减少学习成本。

兼容性需求：

* 系统需兼容微信小程序环境，确保在不同设备和浏览器中运行正常，确保界面的一致性和操作的流畅性。

# 测试策略

## 测试类型

### 数据和数据库完整性测试

|  |  |
| --- | --- |
| 测试目标： | 确保数据库访问方法和进程能够正常运行，确保所有数据在插入、修改、删除和查询过程中不会遭到损坏。包括确保商品SKU/SPU信息、用户订单数据和库存信息的完整性和准确性​ |
| 方法： | · **调用数据库访问方法**：通过API或数据库驱动程序，直接调用数据库访问方法，向数据库中填充有效和无效数据。  · **验证数据一致性**：检查数据库中存储的数据是否符合预期，确保所有数据库事件（如数据写入、更新和查询）都按设计要求正常出现。  · **无效数据测试**：模拟无效数据输入（例如，重复的商品编号、无效的库存数量等），确保数据库能正确处理异常数据并返回相应的错误信息。  · **完整性检查**：验证查询和检索的数据与输入数据一致，确保数据不会在传输或处理过程中丢失或被篡改。 |
| 完成标准： | · 所有数据库访问方法均按照设计的方式正常运行。  · 数据库中的数据在输入、更新、查询和删除操作中没有损坏，数据完整性和一致性得以保持。  · 数据库在遇到无效数据时能够正确处理并返回预期的错误消息 |
| 需考虑的特殊事项： | · 测试可能需要使用DBMS开发环境或数据库驱动程序，直接在数据库中插入、修改或删除数据。  · 测试过程应手动调用数据库进程，以便能够更清楚地监控数据流和处理情况。  · 使用小型数据库进行测试，数据库中的记录数量有限，以便更容易发现和排查异常情况​ |

### 功能测试

|  |  |
| --- | --- |
| 测试目标： | * **导航**：验证用户在平台上的导航是否顺畅，页面跳转是否正确。 * **数据输入、处理和检索**：确保平台能够正确接受用户输入的数据，处理业务逻辑，并能够正确检索数据。验证商品信息的输入、订单的创建与支付、库存管理等操作​(项目组选题说明)​(丰收宝盒总体需求规约)。 * **业务规则**：验证各项业务规则是否按照需求正确执行，例如订单创建条件、支付规则、库存更新逻辑​(丰收宝盒总体需求规约) |
| 方法： | · **用例测试**：执行系统中定义的各个用例，针对每个业务功能进行测试。   * **有效数据测试**：输入符合业务规则的有效数据（如正确的商品信息、有效的订单和支付数据），验证系统是否能返回正确的处理结果。 * **无效数据测试**：输入不符合业务规则的无效数据（如错误的商品ID、库存不足等），验证系统是否能返回相应的错误消息或警告。   · **黑盒测试**：通过图形用户界面 (GUI) 模拟用户操作，分析应用程序的输出，确保用户的每项操作得到正确的反馈。 |
| 完成标准： | · 所有计划的测试用例和功能测试都已执行完毕。  · 所有发现的缺陷均已修复，功能模块均按照需求正确运行。  · 系统能够在不同的业务场景下正确接受、处理和检索数据，且业务规则得到正确实施​ |
| 需考虑的特殊事项： | · 系统外部因素（如微信支付平台的依赖性）可能会对某些功能测试产生影响，应确保集成接口正常运行。  · 确保测试环境与生产环境一致，模拟真实用户场景，特别是不同用户角色（商家、用户）对功能的使用差异 |

### 业务周期测试

|  |  |
| --- | --- |
| 测试目标 | 确保“丰收宝盒”平台及其后台进程能够按照业务模型和时间表正常运行。测试应验证系统在不同的时间段内（如每日、每周、每月）的事务和活动，尤其是与日期相关的事件（如订单生成、发货提醒等），是否能够按预期自动执行。 |
| 方法： | · **模拟多个业务周期**： 模拟系统在一年内的业务周期，执行在该期间可能发生的各种事务。通过增加每项功能的执行次数，模拟系统在不同时间段内的日常操作。  · **日期与时间测试**： 使用有效和无效的日期和时段，测试系统能否处理与时间相关的功能。例如，验证订单在截单时间后的处理、商品促销活动的开始与结束、库存数据的自动更新等。  · **周期性功能测试**： 在合适的时间启动或执行所有周期性功能，如每月的销售数据统计、定期库存检查提醒、农产品的季节性销售等，确保这些功能能按时自动运行。  · **数据验证**： 使用有效的数据执行各项业务功能，确保系统能够按时生成正确的事务和处理结果。使用无效数据验证系统的错误处理机制，确保错误信息和警告能够正常显示。  · **业务规则验证**： 确保在业务周期内各项功能的执行遵循预设的业务规则，如订单生成规则、支付流程、促销规则等。 |
| 完成标准： | · 所有计划的业务周期测试已执行完毕。  · 所有发现的缺陷已解决。  · 系统能够在不同时间段内按照设定的业务模型和时间表正确执行各项事务和活动，且业务规则正确应用​ |
| 需考虑的特殊事项： | · 系统日期和时间相关的事件可能需要特殊支持，确保测试环境中能够准确模拟不同的时间和日期事件。  · 需通过业务模型确定周期性功能的执行需求和相应的测试过程，确保周期性事务能够按时启动并正确执行 |

### 用户界面测试

|  |  |
| --- | --- |
| 测试目标： | · 确保用户通过浏览各个窗口、页面和功能模块时，能够正确执行操作，系统能够正确反映业务的功能和需求。  · 核查窗口之间的导航是否流畅，字段的输入是否符合预期，各种访问方法（如Tab键、鼠标点击、快捷键等）是否能够正常使用。  · 确保界面元素（如菜单、按钮、字段、窗口大小、位置、状态等）符合设计标准，且满足公司或行业的界面设计规范​ |
| 方法： | · **窗口和浏览测试**：为每个窗口或页面创建专门的UI测试用例，逐步核查从一个窗口到另一个窗口的跳转是否正确，字段之间的切换是否顺畅。  · **操作路径测试**：使用Tab键、鼠标点击、快捷键等不同的方式测试用户操作路径，确保所有操作方法都能正确执行，并检查操作反馈是否符合预期。  · **界面对象测试**：验证各个UI元素（如菜单、按钮、文本框、图标等）的显示、位置、大小、响应速度和功能是否符合标准设计要求。  · **兼容性测试**：在不同设备、屏幕分辨率和浏览器上运行界面测试，确保界面布局和功能在所有设备上保持一致且正常工作 |
| 完成标准： | · 界面中的所有窗口和页面能够正确反映平台的功能需求，导航和浏览体验符合设计预期。  · 各个窗口、字段以及功能模块之间的切换顺畅，所有访问方法均能正常使用。  · 所有UI元素符合设计规范，界面布局一致，功能响应正确，确保用户在各个设备和浏览器上的体验一致​ |
| 需考虑的特殊事项： | 某些定制或第三方对象的特性可能无法通过常规测试方法访问，可能需要特定的工具或方法进行测试 |

### 性能评价

|  |  |
| --- | --- |
| 测试目标： | * **正常工作负载**：验证平台在日常用户操作下的响应时间、事务处理速率等性能指标是否达到要求。 * **最大预期负载**：模拟系统在高峰期或最繁重的工作负载下的表现，确保平台能够处理大规模用户同时操作，验证系统能否稳定运行。 |
| 方法： | · **使用功能或业务周期测试的过程**：利用前面功能和业务周期测试中的测试过程，扩展到性能测试中，特别是通过增加并发事务和用户数量来模拟高负载场景。  · **增加事务数量**：通过修改数据文件，增加并发事务的数量，并使用脚本模拟多次迭代的用户操作。例如，模拟多用户同时浏览商品、下单、支付等高并发操作。  · **基准测试**：在单个用户和单个事务的基准条件下运行脚本，记录响应时间和事务处理速率。  · **多客户机并发测试**：使用多台客户机（虚拟或实际）同时运行测试脚本，模拟大量用户并发访问。每个客户机执行相同的操作，以评估系统在多用户负载下的性能表现。 |
| 完成标准： | · **单个事务或单个用户**：在每个事务的预期时间范围内成功完成测试，没有发生任何性能故障。  · **多个事务或多个用户**：在多用户并发环境中，所有测试脚本应在可接受的时间范围内完成，且没有发生性能瓶颈或系统崩溃。 |
| 需考虑的特殊事项： | · **后台工作负载**：在性能测试中，需在服务器上添加额外的后台工作负载，例如通过SQL调用直接向服务器分配事务，或通过虚拟用户负载工具模拟数百个客户机的访问​(rup\_tstpln)。  · **虚拟用户和网络负载**：通过远程终端仿真工具（Remote Terminal Emulation）来模拟大量用户负载，或者在实际客户机上运行多个并发脚本，观察系统在高负载下的响应。  · **测试环境**：性能测试应在专用计算机或专用的机时内执行，以确保测试环境的控制和测试结果的精确评估。  · **数据库规模**：性能测试应使用与生产环境中相同规模的数据库，或按比例缩放的数据库，确保测试结果的准确性和现实性。 |

### 强度测试

|  |  |
| --- | --- |
| 测试目标： | · 服务器上几乎没有或完全没有可用的内存（RAM和DASD）。  · 最大数量的客户机同时连接或模拟。  · 多个用户对相同数据/账户执行相同的事务。  · 处理最繁重的事务量或最差的事务组合 |
| 方法： | · **资源限制测试**：减少或限制服务器上的可用内存（RAM）和磁盘存储空间（DASD），在这些条件下运行系统功能测试，模拟系统资源不足时的运行状态。  · **多客户机并发测试**：使用多台客户机同时运行相同或互补的事务，模拟系统在处理最繁重事务量或最差事务组合时的性能表现。事务可能包括商品浏览、下单、库存更新等。  · **系统压力增加**：通过减少资源的可用性，持续增加系统负载，直至系统达到极限。测试如何处理高负荷的同时并发访问 |
| 完成标准： | · 所有计划的测试均已执行完毕，且系统在达到或超出指定的资源限制时没有出现软件故障或崩溃。  · 如果系统出现故障，则条件必须超出指定范围或达到极端水平，超出实际预期使用条件​ |
| 需考虑的特殊事项： | · **增加网络强度**：可能需要使用网络工具来增加网络工作负载，通过加载消息或信息包对网络压力进行测试。  · **减少数据库可用空间**：暂时减少DASD可用空间，以限制数据库增长，从而测试系统如何应对空间不足。  · **多客户机同步访问**：确保多个客户机对相同数据或账户的同时访问达到同步状态，测试系统在高竞争条件下的响应情况 |

### 安全性和访问控制测试

|  |  |
| --- | --- |
| 测试目标： | · **应用程序级别的安全性**：确保不同用户类型只能访问其授权使用的功能或数据。例如，普通用户可以浏览和下单，但不能进行库存管理；而管理员可以访问和管理所有商品和订单数据。  · **系统级别的安全性**：验证只有具备系统访问权限的用户才能登录系统，并且只能通过相应的网关或身份验证访问系统和应用程序。 |
| 方法： | · **应用程序级别的安全性**：   1. **确定用户类型及权限**：列出系统中每种用户类型（如普通用户、商家、管理员）及其被授权使用的功能和数据权限。 2. **创建用户类型测试**：为每个用户类型设计测试用例，确保他们只能执行被授权的事务。例如，普通用户无法删除数据，而管理员可以完全管理订单和商品数据。 3. **权限验证测试**：修改用户类型的权限设置，重新运行测试，确保系统能够正确限制或允许用户访问指定的功能和数据。   · **系统级别的安全性**：   1. 验证用户只能通过有效的登录方式进入系统，并且未经授权的用户无法访问系统。 2. 测试远程访问的安全性，确保所有外部访问都经过适当的网关和身份验证。 |
| 完成标准： | · 各种已知的用户类型只能访问其对应权限范围内的功能或数据，所有事务均按照预期的安全性规则执行。  · 未授权的用户不能访问系统或进行敏感操作，系统能够有效防止未经授权的访问。 |
| 需考虑的特殊事项： | 需要与网络或系统管理员共同检查和确认系统的访问权限设置，确保网络管理或系统管理的职能与应用程序安全性测试保持一致。如果系统访问控制涉及网络层面的安全性，则可能需要额外的网络工具支持。 |
| 完成标准： | · 在各种软件和硬件组合下，所有事务均能成功执行，且未发生任何故障或异常。  · 系统能够在不同的硬件和软件配置中保持性能稳定，用户体验一致。 |
| 需考虑的特殊事项： | · **非测试对象软件**：需要确定哪些非测试对象软件（如常用的办公软件或驱动程序）会在客户机工作站中运行，并评估其对系统性能的潜在影响。  · **应用程序资源占用**：确定常用应用程序正在处理什么数据（例如，大型Excel电子表格或复杂的Word文档），并模拟在这种情况下系统的运行情况。  · **记录系统配置**：测试时需记录所有硬件和软件配置，包括网络服务器、数据库、Netware等，以确保测试环境可重复且符合实际生产环境。 |

### 安装测试

|  |  |
| --- | --- |
| 测试目标： | 验证“丰收宝盒”平台能够在各种硬件配置和安装场景下正确安装，并在安装后能够立即正常运行。具体包括以下场景：   * **首次安装**：确保在从未安装过“丰收宝盒”的新计算机上，系统能够正确安装。 * **更新安装**：验证在已安装过相同版本或较早版本的计算机上，系统的更新安装能够顺利进行。 * **异常安装**：模拟异常情况（如磁盘空间不足、缺少目录创建权限）下的安装行为，确保系统能够处理这些异常，并提供相应的提示或错误信息 |
| 方法： | · **开发安装脚本**：手动或自动开发安装脚本，验证目标计算机的初始状态（是否已安装系统，安装的版本）。  · **执行安装**：针对不同场景（首次安装、更新安装、异常安装）执行安装过程，观察是否有错误提示及是否能顺利完成安装。  · **运行功能测试**：使用预先确定的功能测试脚本，验证安装后的系统能否正确执行核心事务，确保主要功能如商品管理、订单处理等功能正常运行。 |
| 完成标准： | · “丰收宝盒”在各种硬件配置和安装场景下，均能成功完成安装，并能够正常运行，无重大故障或功能缺失。  · 更新安装不会影响现有数据或功能，且系统能顺利过渡到新版本 |
| 需考虑的特殊事项： | · 选择需要验证的关键事务和功能，确保在每次安装后，所有核心模块（如订单处理、支付系统、商品管理）都能正常运行。  · 在异常安装场景下，重点验证系统的错误处理机制，确保用户能够获得清晰的错误提示，并能采取正确的补救措施。 |

## 工具

此项目将使用以下工具：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 工具 | 厂商/自行研制 | 版本 |
| 测试管理 |  |  |  |
| 缺陷跟踪 | Jira |  |  |
| 用于功能性测试的 ASQ 工具 | Selenium |  |  |
| 用于性能测试的 ASQ 工具 | Apache JMeter |  |  |
| 测试覆盖监测器或评价器 | Cobertura |  |  |
| 项目管理 | Jira |  |  |
| DBMS 工具 | MySQL |  | 8.0 |

# 资源

## 角色

下表列出了在此项目的人员配备方面所作的各种假定。

|  |  |  |
| --- | --- | --- |
| 人力资源 | | |
| 角色 | 推荐的最少资源  （所分配的专职角色数量） | 具体职责或注释 |
| 测试经理，  测试项目经理 | 1 | 进行管理监督。  职责：   * 提供技术指导 * 获取适当的资源 * 提供管理报告 |
| 测试设计员 | 1 | 确定测试用例、确定测试用例的优先级并实施测试用例。  职责：   * 生成测试计划 * 生成测试模型 * 评估测试工作的有效性 |
| 测试员 | 2-3 | 执行测试。  职责：   * 执行测试 * 记录结果 * 从错误中恢复 * 记录变更请求 |
| 测试系统管理员 | 1 | 确保测试环境和资产得到管理和维护。  职责：   * 管理测试系统 * 授予和管理角色对测试系统的访问权 |
| 数据库管理员 | 1 | 确保测试数据（数据库）环境和资产得到管理和维护。  职责：   * 管理测试数据（数据库） |
| 设计员 | 1 | 确定并定义测试类的操作、属性和关联。  职责：   * 确定并定义测试类 * 确定并定义测试包 |
| 实施员 | 1 | 实施测试类和测试包，并对它们进行单元测试。  职责：   * 创建在测试模型中实施的测试类和测试包 |

## 系统

下表列出了测试项目所需的系统资源。

|  |  |
| --- | --- |
| 系统资源 | |
| 资源 | 名称/类型 |
| 数据库服务器 | MySQL 8.0 |
| —网络或子网 | 127.0.0.1 |
| —服务器名服务器名 | 局域网 |
| —数据库名 | 丰收宝盒测试数据库 |
| 客户端测试 PC | 标准配置，32GB RAM，WIndow10 |
| —包括特殊的配置需求 |  |
| 测试存储库 | 存储测试结果、日志和缺陷报告 |
| —网络或子网 | 局域网 |
| —服务器名服务器名 | 本电脑 |
| 测试开发 PC | 标准配置，适用于测试 |

# 项目里程碑

|  |  |  |  |
| --- | --- | --- | --- |
| **里程碑任务** | **工作量** | **开始日期** | **结束日期** |
| 制定测试计划 | 5人天 | 2024-9-9 | 2024-9-14 |
| 设计测试 | 10人天 | 2024-9-15 | 2024-9-20 |
| 实施测试 | 15人天 | 2024-9-21 | 2024-9-26 |
| 执行测试 | 20人天 | 2024-9-27 | 2024-10-7 |
| 评估测试 | 5人天 | 2024-10-7 | 2024-10-10 |

# 可交付工件

## 测试模型

· **内容**：测试模型将包括所有测试用例、测试脚本、测试数据和测试环境设置的详细信息。这些模型将用于支持功能测试、性能测试、负载测试、强度测试等 。

· **创建工具**：使用ASQ工具，如Selenium用于功能测试，Apache JMeter用于性能测试，Cobertura用于测试覆盖监测 。

· **创建人员**：测试设计员、测试工程师

· **交付对象**：测试模型文档、测试用例

· **交付时间**：2024-9-15

## 测试日志

· **内容**：测试日志记录每次测试的执行结果，包括测试开始时间、结束时间、测试步骤、实际结果、测试状态等 。

· **记录方法**：使用测试管理工具（如Jira、TestRail）自动生成和管理测试日志 。

· **创建人员**：测试工程师、测试员

· **交付对象**：测试日志报告

· **交付时间**：2024-9-26

## 缺陷报告

· **内容**：缺陷报告记录在测试过程中发现的所有缺陷，包括缺陷描述、严重程度、重现步骤、修复状态等信息 。

· **跟踪工具**：使用缺陷跟踪工具（如Jira）记录和管理缺陷 。

· **创建人员**：测试工程师、测试员

· **交付对象**：缺陷报告文档、缺陷跟踪记录

· **交付时间**：2024-10-7

# 附录 A：项目任务

以下是一些与测试有关的任务：

• 制定测试计划

* + 确定测试需求
  + 评估风险
  + 制定测试策略
  + 确定测试资源
  + 创建时间表
  + 生成测试计划

• 设计测试

- 准备工作量分析文档

- 确定并说明测试用例

- 确定并结构化测试过程

- 复审和评估测试覆盖

• 实施测试

* + 记录或通过编程创建测试脚本
  + 确定设计与实施模型中的测试专用功能
  + 建立外部数据集

• 执行测试

-

执行测试过程

- 评估测试的执行情况

- 恢复暂停的测试

- 核实结果

- 调查意外结果

- 记录缺陷

• 评估测试

- 评估测试用例覆盖

- 评估代码覆盖

- 分析缺陷

- 确定是否达到了测试完成标准与成功标准