|  |
| --- |
| 软件工程综合实验H组 |
| 软件测试计划v1.2 |
| 基于Scrapy的模板化爬虫程序管理平台 |

|  |
| --- |
| 赵正阳、郭浩隆、沈一聪、梁远志、宋冰晨  2020-5-22 |

**修订和签署记录表**

**版本变更历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 日期 | 修改人 | 修改章节 | 备注 |
| 1.0 | 2020/5/17 | 宋冰晨 | 全部 | 建立初稿 |
| 2020/5/18 | 梁远志 | 2.2 | 补充调度器的单元测试假设 |
| 2020/5/18 | 沈一聪 | 2.2、2.5.5.2、2.5.5.3、2.5.5.4 | 补充除可用性测试之外，其他非功能性测试的假设和细节 |
| 2020/5/19 | 赵正阳 | 2.5.3 | 补充功能性测试范围工作表 |
| 1.1 | 2020/5/21 | 郭浩隆 | 1.4、2、3、4.1、4.2、4.3 | 补充图表标题、修正错误格式、补充部分图表说明 |
| 1.2 | 2020/5/22 | 宋冰晨 | 2.5.5 | 更正非功能性测试部分内容 |

**批准/审核人名单**

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 角色 | 批准/审核人 | 批准/审核日期 |
| 赵正阳 | 项目经理 | 批准人 | 2020/5/19 |
| 赵正阳 | 项目经理 | 批准人 | 2020/5/21 |
| 赵正阳 | 项目经理 | 批准人 | 2020/5/22 |

**参考文档**

|  |  |  |
| --- | --- | --- |
| 版本 | 日期 | 文档名 |
| 1.3.3 | 2020/4/16 | 基于Scrapy的模板化爬虫程序管理平台软件需求规格说明书v1.3.3 |
| 1.0 | 2020/5/7 | 基于Scrapy的模板化爬虫程序管理平台实现方案v1.0 |

**目录**

[1. 引言 - 1 -](#_Toc41050995)

[1.1 编写目的 - 1 -](#_Toc41050996)

[1.2 项目概述 - 1 -](#_Toc41050997)

[1.3 适用读者 - 1 -](#_Toc41050998)

[1.4 参考资料 - 1 -](#_Toc41050999)

[2. 测试策略 - 3 -](#_Toc41051000)

[2.1 测试目标 - 3 -](#_Toc41051001)

[2.2 测试假设 - 3 -](#_Toc41051002)

[2.3 测试原则 - 4 -](#_Toc41051003)

[2.4 数据策略 - 5 -](#_Toc41051004)

[2.5 测试范围和级别 - 5 -](#_Toc41051005)

[2.5.1 构建验证测试 - 5 -](#_Toc41051006)

[2.5.2 探索性测试 - 5 -](#_Toc41051007)

[2.5.3 功能性测试 - 6 -](#_Toc41051008)

[2.5.4 回归测试 - 6 -](#_Toc41051009)

[2.5.5 非功能性测试 - 6 -](#_Toc41051010)

[2.5.5.1 非功能性需求测试 - 7 -](#_Toc41051011)

[2.5.5.2 可用性测试 - 8 -](#_Toc41051012)

[2.5.5.3 性能测试 - 8 -](#_Toc41051013)

[2.5.6 用户验收测试 - 9 -](#_Toc41051014)

[3. 执行策略 - 11 -](#_Toc41051015)

[3.1 进入和退出标准 - 11 -](#_Toc41051016)

[3.2 验证和缺陷管理 - 11 -](#_Toc41051017)

[3.3 缺陷追踪 - 12 -](#_Toc41051018)

[4. 测试管理过程 - 13 -](#_Toc41051019)

[4.1 测试设计流程 - 13 -](#_Toc41051020)

[4.2 测试执行流程 - 13 -](#_Toc41051021)

[4.3 测试风险和缓解措施 - 14 -](#_Toc41051022)

[4.4 沟通计划和团队名册 - 14 -](#_Toc41051023)

1. **引言**

1.1 编写目的

本测试计划用于描述基于Scrapy的模板化爬虫程序管理平台(EasySpider)的测试方法和总体框架，主要包含以下三个方面。

* 测试策略：测试将基于的规则，包括项目的给定条件（例如目标、假设、时间安排）、建立有效测试的过程描述（例如进入/退出标准、创建测试用例、要执行的特定任务、数据策略）。
* 执行策略：描述如何执行测试、识别和报告缺陷以及实施修复的过程。
* 测试管理：测试的后勤工作以及执行过程中可能发生的事件（例如：沟通、风险和缓解措施）的管控处理流程。

1.2 项目概述

EasySpider为用户提供了针对多个特定网站的爬虫模板（与Scrapy爬虫脚本相对应），用户无需编写代码即可使用爬虫采集数据。同时用户可以在平台上运行和管理自己的爬虫程序。此外，EasySpider在爬虫脚本中加入了应对反爬虫的机制来增强爬虫程序的鲁棒性，通过增加分布式爬虫机制来提高爬取效率。

1.3 适用读者

本文档的适用读者主要分为如下五类。

* 项目团队成员：执行本文档中指定的任务，并提供针对本文档的意见和建议。
* 项目经理：计划整个项目进度表中的测试活动，审查相关文档，根据此处指定的任务跟踪测试的执行情况，批准文档并对结果负责。
* 利益相关者的代表如评审人：参加用户验收测试UAT(User Acceptance Test)，以确保业务与测试结果保持一致。
* 开发团队：确保测试计划和可交付成果与设计一致，提供测试环境并遵循与缺陷修复相关的程序。
* 业务设计相关人员：提供有关功能更改的意见。

1.4 参考资料

[1] Test Plan Tutorial: A Guide To Write A Software Test Plan Document From Scratch [OL]. <https://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/>

[2] 维基百科Usability testing [OL]. <https://en.wikipedia.org/wiki/Usability_testing>

[3] ISO 9241-11:2018(en)[OL]. <https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:en>

[4] 可用性测试：任务评估模型与计量方式[OL]. <http://www.woshipm.com/user-research/751870.html>

[5] 如何进行可用性测试？这里有一份全面的可用性测试指南[OL]. <http://www.woshipm.com/pd/1016493.html>

1. **测试策略**

2.1 测试目标

该测试的目的是验证项目实现的功能是否符合规范。

该测试将执行和验证测试脚本，修复之后重新测试所有高和中等严重程度的缺陷，对较低严重程度缺陷进行优先级排序，以供将来通过CR(Change Request)修复。

测试的最终结果是有两个：

* 可用于生产的软件；
* 一组稳定的测试脚本，可重复用于单元测试。

2.2 测试假设

**1. 关键假设**

* 在开始功能测试之前，需要准备好和生产类似的数据，并且可以在软件中使用。

**2．一般假设**

* 项目团队具有必要的知识和经验，或者已经在测试过程中接受了充分的培训。
* 项目经理将审查并签署所有测试交付物。
* 测试团队假定开发人员将适当支持测试设计和执行过程中所需的所有必要输入，如准备测试环境。
* 开发团队将根据测试团队提供的软件问题报告对缺陷进行修复。在开始缺陷修复之前和完成缺陷修复之后，都需要将情况通知测试团队。
* 所有缺陷都将带有快照。
* 系统将被视为黑匣子，如果数据信息在浏览器和报告中正确显示，则将假定数据库运行正常。

1. **构建验证测试假设**

* 开发团队采用Pull Request进行代码协作。
* 开发团队在GitHub仓库中通过GitHub 操作创建自定义持续集成(CI)和持续部署(CD)工作流程，且配置CI工作流程在Pull Request时运行，配置CD工作流程在代码推送到master分支时运行。

1. **探索性测试假设**

* 构建验证测试(Build Verification Test)一旦通过，将进行探索性测试(Ad hoc Test)。

1. **回归测试假设**

* 在开发团队完成缺陷修复之后，测试团队将进行回归测试(Regression Test)。

1. **功能性测试假设**
2. 单元测试

* 对于Django，测试团队使用Django内置的单元测试完成测试脚本的编写；对于Scrapy和调度器，使用Unittest和Coverage软件进行单元测试并导出覆盖报告。

1. 功能测试

* 测试团队已完成所有单元测试，且所有单元测试通过。

1. 集成测试

* 测试团队已完成所有功能测试，且所有功能测试通过。

1. 场景测试

* 测试团队已完成所有集成测试，且所有集成测试通过。

1. 系统测试

* 如果测试团队完成了针对用户界面的可用性测试，针对注入攻击的安全性测试，针对浏览器的兼容性测试，效能、负载、压力测试以及分布式与IP代理池性能测试，则标志系统测试完成。

1. **非功能性测试假设**

* 开发团队已完成WEB UI开发且部署至服务器，测试团队可以通过浏览器访问网页。

1. **用户验收测试假设**

* 本测试将由终端用户（包括评审人）执行，并由用户形成软件问题报告反馈给项目组。

2.3 测试原则

本项目的测试原则包括：

* 测试将重点关注业务目标的实现程度和完成质量；
* 测试活动建立在先前阶段的基础上，以避免重复工作；
* 测试环境和数据将尽可能模拟生产环境；
* 测试过程定义明确且灵活，能够根据需要进行更改；
* 支持测试活动的所有团队拥有一致的软件程序；
* 测试将分为不同的阶段，每个阶段执行不同测试，且都有明确定义的目标；
* 测试是可重复、可量化和可衡量的活动；
* 设置进入和退出标准。

2.4 数据策略

在功能测试中，测试人员自行创建任务从而完成任务模块的测试。如果测试人员在创建任务的过程中遇到问题，开发人员应给予指导和帮助。

2.5 测试范围和级别

2.5.1 构建验证测试

**目的：**确保提交代码通过编写的Django单元测试

**范围：**所有Django模块

**测试人员**：测试团队

**方法**：测试团队使用Django内置的单元测试完成测试脚本的编写，在GitHub上自定义持续集成(CI)工作流程，并配置该流程在Pull Request时运行

**时间安排**：测试最开始阶段

**备注**：本测试属于冒烟测试(Smoke Test)，如果测试不通过，提交的代码无法合并到master分支。

2.5.2 探索性测试

**目的：**确保在下一级实验（功能性测试）开始前，关键缺陷已被清除

**范围：**模板浏览模块、任务操作模块、数据下载模块

**测试人员**：测试团队

**方法**：在软件中执行，不需要任何测试脚本和文档

**时间安排**：在完成构建验证测试之后

2.5.3 功能性测试

**目的：**检查应用程序的功能（通过提供输入来执行功能测试，并验证应用程序的输出）

**范围：**Excel工作表“功能性测试范围.xlsx”详细介绍了功能性测试的范围。

**测试人员**：测试团队

**方法**：对于单元测试，分别执行针对Django、Scrapy和调度器编写的测试脚本；对于功能测试、集成测试、场景测试采用手工方式进行测试。

**时间安排**：在完成探索性测试之后

**测试前提和约束**：

1. 需求规格说明书（包括需求用例）、测试需求规格说明书（包含测试用例）必须在本测试执行阶段开始之前处于可用状态；
2. 按照单元测试、功能测试、集成测试、场景测试的顺序执行，只有前面的测试通过才能进行后面的测试；
3. 功能测试、集成测试、场景测试开始之前要求软件系统已部署到服务器上。

**测试可交付成果**：

表2.1 功能性测试可交付成果

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **可交付成果** | **作者** | **审核人** |
| 1 | 测试结果（覆盖表） | 测试团队 | 项目经理 |

2.5.4 回归测试

**目的：**对一个新的版本，重新运行之前的测试用例，检查新版本和已知的版本

相比是否有退化。

**测试人员**：测试团队

**方法**：开发人员根据每个测试阶段交付的软件问题报告修复缺陷，通知测试团队重新执行本阶段的所有测试用例。如果测试通过，则进入下一个测试阶段；反之，开发人员根据新的问题报告继续修复本阶段缺陷。

**时间安排**：在功能性测试的每个阶段（单元测试、功能测试、集成测试、场景测试）结束之后进行。

2.5.5 非功能性测试

2.5.5.1 非功能性需求测试

**目的：**评估项目是否完成需求规格说明书规定的非功能性需求

**范围：**

下表详细介绍了非功能性需求测试的范围。

表2.2 非功能性需求测试范围

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **测试用例名称** | **对应需求名称** | **对应需求章节** |
| NFT01 | 禁用JavaScript访问 | 可访问性 | 3.3.2.(1) |
| NFT02 | 浏览器兼容性 | 兼容性 | 3.3.2.(2) |
| NFT03 | 响应式设计 |
|  | / | 易用性 | 3.3.2.(3) |
| NFT04 | 从提供模板创建新站点爬虫 | 模板的可扩充性 | 3.3.2.(4) |
| NFT05 | 用户密码强度要求 | 安全性 | 3.3.2.(5) |
| NFT06 | SQL注入攻击 | 防攻击性 | 3.3.2(6) |
| NFT07 | CSRF攻击 |
| NFT08 | XSS攻击 |
|  | / | 可维护性 | 3.3.2.(7) |
|  | / | 可靠性 | 3.3.2.(8) |
| NFT09 | 页面访问加载时间 | 良好的访问性能 | 3.3.2.(9) |
| NFT10 | 爬取具有反爬虫机制的站点 | 应对反爬虫 | 3.3.2.(10) |
| NFT11 | 爬虫节点失效 | 故障处理 | 3.6 |

**测试人员**：沈一聪

**时间安排**：在完成探索性测试之后

**测试前提和约束**：

1. 需求规格说明书（包括需求用例）、测试需求规格说明书（包含测试用例）必须在本测试执行阶段开始之前处于可用状态；

2. 本测试开始之前要求软件系统已部署到服务器上。

**测试可交付成果**：

表2.3 非功能性需求测试可交付成果

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **可交付成果** | **作者** | **审核人** |
| 1 | 测试结果（覆盖表） | 沈一聪 | 项目经理 |

2.5.5.2 可用性测试

**目的：**通过用户的使用来评估项目是否满足用户需求

**范围：**EasySpider的WEB UI

**测试人员**：测试团队

**评估指标**：

ISO9241中对“可用性”的定义是：特定用户在特定的使用场景中，为了达到特定目标而使用某产品时，所感受到的有效性、效率和满意度。在定义好了用户、场景和目标的前提下，可用性包含了下面三个维度。

* 有效性(Effectiveness)：用户完成特定目标的正确和完整程度。
* 效率(Efficiency)：用户完成特定目标的效率，与消耗的资源（如时间）成反比。
* 满意度(Satisfaction)：用户使用产品时感受到的主观满意程度。

良好的可用性必须能够同时满足有效性、效率和满意度三个条件，但是这三个维度也有层次之分，一般来说，有效性问题 > 效率问题 > 满意度问题。

**待评估任务**：创建任务、重命名任务、删除任务、暂停后运行任务、下载数据

**方法**：

采用认知走查的方法进行可用性测试：前端开发人员模拟用户执行待评估任务，检查用户的任务目标和心理认知是否可以顺利执行下一步操作，填写可用性报告；测试团队非前端开发人员在进行功能性测试时可以向前端开发人员报告软件在有效性、效率、满意度方面的问题，由前端开发人员补充到可用性报告中。

**时间安排**：在完成探索性测试之后

**测试可交付成果**：

表2.4 可用性测试可交付成果

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **可交付成果** | **作者** | **审核人** |
| 1 | 可用性报告 | 宋冰晨 | 项目经理 |

2.5.5.3 性能测试

**目的**：评估系统的分布式爬虫的性能优化效果、系统平稳性以及不同并发数下系统的性能。

**范围：**

下表详细介绍了性能测试的范围。

表2.5 性能测试范围

|  |  |  |
| --- | --- | --- |
| **测试名称** | **对应需求名称** | **对应需求章节** |
| 分布式爬虫的性能优化测试 | / | / |
| 平稳性测试 | / | / |
| 负载/压力测试 | / | / |

**测试人员**：沈一聪

**评估指标与方法**：

* 分布式爬虫的性能优化测试

分别采用EasySpider和一般的爬虫脚本完成同样工作量的爬虫任务，对比的评估指标是爬取时间。

* 平稳性测试

采用EasySpider在一段较长时间内持续爬取数据，评估系统吞吐量与时间的关系。

* 负载/压力测试

测试不同并发数下系统的性能指标，对比的评估指标是Http\_load提供的bytes/connection、fetches/sec等结果。

**时间安排**：在完成探索性测试之后

**测试可交付成果**：

表2.6 性能测试可交付成果

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **可交付成果** | **作者** | **审核人** |
| 1 | 性能测试报告 | 沈一聪 | 项目经理 |

2.5.6 用户验收测试

**目的：**该测试重点在于验证业务逻辑，它允许终端用户（包括评审人）在项目进入生产阶段前完成对项目的最终审查。

**测试人员**：终端用户（包括评审人）

**方法**：终端用户（包括评审人）执行设计的所有测试用例。

**时间安排**：在完成所有其他级别的测试之后。只有在本测试完成后，项目才能投入生产。

**测试可交付成果**：

表2.7 效能、负载、压力测试可交付成果

|  |  |  |
| --- | --- | --- |
| **序号** | **可交付成果** | **作者** |
| 1 | 软件评审报告 | 终端用户（包括评审人） |
| 2 | 软件问题报告 | 终端用户（包括评审人） |

1. **执行策略**

3.1 进入和退出标准

每个测试阶段的进入标准指的是执行此测试所需的前提条件，退出标准是为了继续实施下一步测试而需要满足的理想条件。进入和退出标准都应该是灵活的。如果未达到标准，测试团队将评估风险，确定缓解措施并提出建议。项目经理根据建议做出最终的决定。

可供参考的退出标准如下表所示。

表3.1 退出标准

|  |  |  |  |
| --- | --- | --- | --- |
| **退出标准** | **测试团队反馈**  **（是/否）** | **开发团队反馈**  **（是/否）** | **备注** |
| 执行了100％的测试脚本 |  |  |  |
| 测试脚本的通过率达到95％ |  |  |  |
| 无高严重程度的缺陷 |  |  |  |
| 已修复95％的中等严重程度的缺陷 |  |  |  |
| 所有剩余的缺陷将被取消或记录为将来版本的变更请求 |  |  |  |
| 测试脚本捕获并记录了所有预期结果和实际结果 |  |  |  |

3.2 验证和缺陷管理

测试人员有责任报告缺陷，并将它们链接到相应的用例以及分配初始严重程度。当开发人员修复缺陷后，测试人员需重新测试并决定是否通过。开发团队有责任根据软件问题报告修复缺陷。如有问题，应与报告缺陷的测试人员沟通。

可供参考的缺陷严重程度等级如下表所示。

表3.2 缺陷严重程度等级

|  |  |
| --- | --- |
| **严重程度** | **造成的影响** |
| 1（高） | * 数据丢失 * 崩溃、界面锁死 * 安全问题 * 阻碍主要功能的使用 |
| 2（中） | * 阻碍部分功能的使用 * 界面不是所见即所得 * 可扩展性 * 场景不完整 |
| 3（低） | * 小的可用性问题 * 性能问题、间歇性的停顿 |

3.3 缺陷追踪

以下流程图描述了缺陷跟踪过程。

图3.1 缺陷追踪流程图



* 缺陷跟踪过程从测试人员出发，首先由测试人员报告缺陷
* 然后开发人员对缺陷进行修复
* 修复完成后测试人员对系统进行重新测试
* 如果通过测试则在软件问题报告中填写通过
* 否则由开发人员对软件缺陷重新检查并修复。

1. **测试管理过程**

4.1 测试设计流程

图4.1 测试设计流程

* 测试人员需要理解每个需求；
* 根据原型、需求用例来准备相应的测试用例，以确保涵盖所有需求；
* 每个测试用例都将接受业务设计人员的审查，审查出的缺陷如测试用例缺少将报告给测试团队；
* 测试人员将对评审缺陷进行返工。

4.2 测试执行流程

图4.2 测试执行流程

* 测试人员分步执行测试用例的每个步骤；
* 如果出现任何故障，需要在覆盖表中标记用例为失败状态，否则标记为通过状态；
* 在问题报告中详细说明故障步骤以及提供屏幕快照；
* 重复上述过程，直到所有测试用例都完全执行，并将问题报告提交给开发人员；
* 在必要时需要向开发人员解释缺陷；
* 在随后的循环中，将测试修复的任何缺陷，并在循环期间在覆盖表中更新结果，最终完成所有的测试用例。

4.3 测试风险和缓解措施

可供参考的测试风险和缓解措施如下表所示。

表4.1 缺陷严重程度等级

|  |  |  |  |
| --- | --- | --- | --- |
| **风险** | **严重程度** | **造成的影响等级** | **缓解措施** |
| **时间表**  测试时间表很紧。 | 高 | 高 | 测试团队提前准备测试任务以及与相关方及早沟通。 |
| **缺陷**  在测试后期发现缺陷。较晚发现的缺陷很可能是由于需求不明确所致，解决这些问题很耗时。 | 中 | 高 | 制定缺陷管理计划以确保及时沟通和解决问题。 |
| **“新”的缺陷**  在测试过程中，很有可能发现一些“新”缺陷，并需要花费较多时间来解决。如果这些问题严重程度很高，将极大地影响整个项目进度。 | 中 | 高 | 明确缺陷管理和问题管理过程以立即提供解决方案。 |

4.4 沟通计划和团队名册

以下列表定义了与项目测试的管理、计划、执行相关的角色以及期望。

表4.2 团队名册

|  |  |  |
| --- | --- | --- |
| **角色** | **姓名** | **期望** |
| 项目经理 | 赵正阳 | * 审核测试计划、测试策略和测试评估的内容 |
| 测试团队 | 赵正阳 | * 设计测试用例 * 执行和验证测试用例 * 缺陷修复后重新执行测试 * 记录缺陷并确定其严重等级 |
| 郭浩隆 |
| 沈一聪 |
| 梁远志 |
| 宋冰晨 |
| 开发团队 | 赵正阳 | * 审核测试可交付成果（测试计划、测试脚本、测试用例等）并提供及时的反馈 * 协助验证测试结果 * 根据软件问题报告修复缺陷 |
| 郭浩隆 |
| 沈一聪 |
| 梁远志 |
| 宋冰晨 |
| 部署负责人 | 梁远志 | * 将项目部署到服务器 * 协助测试团队和评审人完成项目在服务器上的所有测试 |
| 业务设计负责人 | 宋冰晨 | * 审核测试用例 * 协助测试团队完成测试用例的设计 |
| 郭浩隆 |