自动化测试框架规范 v1.0

1. 引言

目的：本规范旨在通过标准化测试用例的编写与维护流程，降低自动化测试的技术门槛，使得业务测试人员能够独立、高效地创建和维护符合自动化要求的测试用例，从而全面提升测试效率与软件质量。

适用范围：本规范适用于所有参与自动化测试用例设计、编写与维护的团队成员，特别是业务测试人员和功能测试工程师。

术语定义：

术语	定义
测试用例 (Test Case)	为某个特定目标而编制的一组测试输入、执行条件以及预期结果，用于核实软件的某个特定功能或路径是否按预期工作。
自动化脚本 (Automation Script)	由代码编写的、可自动执行测试用例的程序。
测试数据 (Test Data)	在测试执行过程中，作为输入或用于验证预期结果的数据。

2. 自动化测试框架概述

本框架采用分层设计理念，主要由以下几个核心组件构成：

测试用例层 (Test Case Layer):
存放具体的测试用例脚本，业务测试人员主要在此层级进行工作。

业务操作层 (Operation Layer): 封装了与业务相关的、可复用的操作，供测试用例调用。

API封装层 (API Layer): 负责与后端服务进行HTTP通信，将接口请求和响应进行封装。

公共组件层 (Common Layer): 提供日志、数据库操作、配置读取等通用功能。

数据层 (Data Layer): 管理测试数据，实现数据与脚本的分离。

框架通过Pytest测试执行引擎驱动，能够自动发现并执行测试用例，并利用Allure框架生成详细、可视化的测试报告。

3. 用例编写与维护规范

用例命名规则：

所有测试用例文件和函数均需遵循统一的命名约定：

文件名：`test_模块名_功能点.py` (例如: `test_user_login.py`)
函数名：`test_场景描述()` (例如: `test_login_with_valid_credentials()`)
命名必须清晰、具有业务可读性，便于快速理解用例的测试目的。

用例结构与要素：

每个测试用例都应在测试管理工具（如Jira, TestRail）中包含以下字段：

字段	说明	示例
用例ID	系统生成的唯一标识符	TC-001
用例标题	简洁描述测试目的	验证用户使用有效凭据成功登录
前置条件	执行用例前需满足的状态	用户已注册并处于未登录状态
测试步骤	详细、可执行的操作步骤	1. 打开登录页面 2. 输入用户名 3. 输入密码 4. 点击登录按钮
预期结果	每个步骤或最终的预期行为	页面跳转到用户首页，并显示欢迎信息
测试数据	用例所需的数据	用户名: testuser 密码: Password123
优先级	业务重要性级别	High / P0
标签/分类	用于归类和筛选	功能测试, 冒烟测试, 登录模块

测试数据管理：

测试数据统一存放在 `data` 目录下，按业务模块进行组织。推荐使用 YAML 或 Excel 文件进行管理，以实现数据与代码的完全分离。必须确保测试数据的独立性，避免用例间的数据耦合。

可自动化标识：

在测试管理工具中，为计划自动化的用例添加特定标签，如`automatable`。只有满足以下条件的用例才能被标记：场景稳定、需求明确、可重复执行。

用例变更管理：

当业务需求变更时，用例维护人员需及时更新测试管理工具中的用例。自动化工程师根据变更同步更新自动化脚本，并执行回归测试以确保变更的正确性。

4. 自动化脚本与用例的关联

本框架采用数据驱动模式。业务测试人员在 `data` 目录下维护测试数据文件，自动化脚本在运行时会动态读取这些数据，并根据数据内容执行相应的测试逻辑。例如，一个登录测试脚本会读取包含多组用户名和密码的数据文件，循环执行登录操作并验证结果。

5. 执行与报告

测试人员可通过执行特定的命令行指令来触发自动化测试。测试完成后，系统会自动生成 Allure 测试报告。报告中会详细展示每个用例的执行状态、步骤、日志和截图，便于快速定位问题。

6. 常见问题与最佳实践

常见问题：

问题：环境不稳定导致用例失败。解决：在执行前确认测试环境的健康状态，对于非应用本身问题导致的失败，在报告中进行特殊标记。

问题：测试数据过期或无效。解决：建立定期的数据维护机制，确保数据的有效性。

最佳实践：

保持用例的独立性：每个用例应能独立运行，不依赖于其他用例的执行顺序。

用例描述清晰：步骤和预期结果应明确、无歧义。

及时清理过期用例：对于已下线的功能，及时归档或删除相关用例。

7. 附录

用例模板示例：

请参考第3节中的用例结构与要素表格。