

AITS (AI-empowered Intelligent Testing System) 详细设计方案

1. 项目概述与愿景

1.1 项目简介

AITS 是一个旨在利用人工智能技术赋能软件测试全流程的平台。它将先进的 AI 智能体 (Agent) 与传统的自动化测试方法深度融合，涵盖 UI 自动化、接口自动化等关键领域。平台致力于解决传统自动化测试中脚本维护成本高、测试用例设计覆盖率不足、缺陷定位困难以及性能瓶颈分析复杂等痛点。

1.2 核心理念

AI 驱动 (AI-Driven): 从测试用例生成、测试脚本编写、测试执行与结果分析到缺陷报告，全流程引入 AI 智能体，提升效率与智能化水平。

低代码/无代码 (Low-Code/No-Code): 面向测试工程师，提供自然语言交互、可视化拖拽等方式，降低 AI 和自动化技术的使用门槛。

一体化平台 (All-in-One): 整合 UI、接口等多种测试类型，在一个平台内完成测试资产的管理、执行和分析，打破数据孤岛。

可扩展与可集成 (Scalable & Integrable): 采用微服务架构和可插拔的智能体设计，易于扩展新的测试能力，并能与主流的 CI/CD、项目管理工具（如 Jenkins, Jira）无缝集成。

1.3 目标用户与核心价值

- 测试工程师 (QA Engineers):** 核心用户，使用平台进行日常的自动化测试设计、执行和分析工作。
- 测试经理 (QA Managers):** 使用平台的仪表盘和报告功能，监控项目质量、团队效率和测试覆盖率。
- 开发工程师 (Developers):** 查看详细的 AI 分析报告，快速定位和修复缺陷。

2. 技术栈选型

2.1 技术栈总览

- 后端框架:** Django 5.2.5 + Django REST Framework
- 前端框架:** Vue 3 + Element Plus + Vite
- 异步任务:** Celery 5.3.4 + Celery Beat + Redis
- 实时通信:** Django Channels + WebSocket
- AI引擎:** LangChain + LangGraph + 多LLM支持
- 向量数据库:** ChromaDB / Milvus (RAG知识库)

- **测试框架:** HttpRunner (API) + Playwright (Web UI)
- **数据库:** SQLite (开发) / PostgreSQL (生产)
- **报告工具:** Allure

2.2 前端

- **框架:** Vue 3 + Vite：提供极致的开发体验和高性能的运行时表现。
- **UI 库:** Element Plus / Ant Design Vue：成熟稳定，提供丰富的企业级组件。
- **状态管理:** Pinia：Vue 3 官方推荐，轻量、直观且类型安全。
- **图表库:** ECharts / D3.js：用于测试报告和性能数据的可视化展示。
- **代码编辑器:** Monaco Editor：提供类似 VS Code 的体验，用于在线编写和编辑脚本。
- **实时通信: Socket.IO / WebSocket** - (理由: 用于实时展示测试执行日志、任务状态更新和协作功能，提供流畅的用户体验)

2.3 后端

- **框架: Django 4.x + Django Rest Framework (DRF)** - (理由: DRF 提供了强大的序列化、认证和权限系统，能快速构建稳健的 RESTful API。Django ORM 功能成熟，管理后台开箱即用，非常适合快速开发企业级应用的核心业务逻辑)
- **异步处理: Celery + Redis** - (理由: Celery 是 Python 中最成熟的分布式任务队列，结合 Redis Broker，可以可靠地处理耗时的测试执行、AI 分析和报告生成任务，并通过 flower 进行任务监控)
- **数据库: PostgreSQL** - (理由: 除了稳定性，还支持 JSONB 字段类型，非常适合存储非结构化的测试步骤、执行结果和 AI 分析建议等数据)
- **缓存: Redis** - (理由: 除了作为 Celery Broker，还用于缓存用户会话、热点数据和 API 响应，降低数据库压力)
- **Web 服务器:** Nginx + Gunicorn - (理由同上)

2.4 AI 智能体 (Agent) 核心

- **核心框架: LangChain + LangGraph**
 - **LangChain:** 作为“胶水层”，用于标准化地集成和调用 LLMs、向量数据库、API 工具等。
 - **LangGraph:** 项目的核心。测试流程（尤其是缺陷定位和 UI 自愈）不是线性的，而是包含判断、循环和状态修正的图状结构。例如：“执行步骤 -> 成功 -> 下一步；失败 -> 截图 -> 分析 DOM -> 尝试备用定位器 -> 成功 -> 更新定位器并继续；多次失败 -> 记录缺陷 -> 结束”。LangGraph 完美地建模了这种复杂的、有状态的 Agent 工作流。
- **大语言模型 (LLM): 可插拔模型服务。** 后端设计一个 LLM 服务层，通过统一的接口调用不同的模型。初期可以集成 OpenAI API (GPT-4/GPT-4o) 以获得最佳效果，同时支持配置本地部署的开源模型（如 Llama3, Qwen）作为高性价比或数据私有化的备选方案。
- **向量数据库: ChromaDB / Milvus**
 - **ChromaDB:** 轻量级，易于上手，适合 MVP 阶段和中小型项目，可以快速进行原型验证。
 - **Milvus:** 功能更强大，支持分布式部署和海量向量检索，适合生产环境和大规模应用。
 - **数据源:** 用于 RAG 的知识库应包含：产品需求文档 (PRD)、API 规范 (Swagger/OpenAPI)、UI 设计稿截图、历史测试用例和缺陷报告。

- **视觉模型:** 千问VL / 豆包系列视觉语言模型 / Gemini-2.5-pro / UI-TARS - (理由: 对于 UI 测试, 特别是从截图生成页面对象模型 (POM) 的功能, 必须依赖视觉模型来理解图像内容)

2.5 测试执行引擎

- **UI 自动化:** Playwright (通过 Python 库调用)。 (理由: 相比 Selenium, Playwright 提供了更现代的 API、更好的自动等待机制和对网络请求的强大控制能力, 执行更稳定高效)
- **接口自动化:** HttpRunner。 (理由: 它本身就是一个完整的测试框架, 支持 YAML/JSON 格式定义用例, 无缝集成 Pytest, 支持丰富的断言和数据驱动, AI 生成的目标格式清晰明确)

3. 平台架构设计

平台采用前后端分离的微服务架构, 核心由 **前端应用**、**后端服务** 和 **AI 智能体核心** 三大部分组成。

3.1 架构分层



3.2 核心组件交互



3.3 架构流程说明

1. **前端应用 (Vue 3):** 用户交互的入口，负责测试用例管理、任务编排、实时进度监控和报告可视化的展示。通过 RESTful API 与后端服务通信。
2. **后端服务 (Django):**
 - **API Gateway:** 统一的请求入口，负责认证、路由和限流。
 - **用户与权限服务:** 管理用户、角色和项目权限。
 - **项目与测试管理服务:** 核心业务模块，管理项目、模块、测试用例、测试集、测试计划等。
 - **测试执行与调度服务:** 接收测试任务，通过 Celery 将任务分发到不同的测试执行节点。
 - **报告与分析服务:** 收集、处理和存储测试结果，生成多维度的测试报告。
3. **AI 智能体核心 (LangChain + LangGraph):** 平台的大脑。
 - **AI Agent Gateway:** 作为统一的接口，接收后端服务的调用请求，并分发给相应的智能体。
 - **各类智能体:** 每个智能体都是一个 LangGraph 应用，拥有自己的状态、工具和逻辑流程，专门负责特定领域的任务。
4. **基础设施:**
 - **测试执行节点集群:** 实际执行自动化脚本的环境，可以是 Docker 容器或物理机，安装了 Selenium, Playwright, Locust 等工具。
 - **向量数据库:** 存储项目相关的非结构化数据（需求文档、API 规范、UI 截图分析结果），为 RAG 提供支持。
 - **LLM API 服务:** 统一管理对不同大语言模型的 API 调用。

4、核心功能模块

4.1 项目管理模块 (apps/projects)

功能职责：

- 多项目、多环境管理
- 项目成员权限控制
- 知识库文档管理
- 环境配置管理 (API/Web/App)

核心模型:

- `Project`: 项目基础信息
- `ProjectMember`: 项目成员及权限
- `Environment`: 统一环境配置 (支持API/Web/App三类)
- `UploadedFile`: 文件上传管理

亮点特性:

- 统一环境模型，支持API、Web、App三种测试环境
- 细粒度权限控制 (Owner/Admin/Editor/Viewer)
- 文件去重机制 (基于文件哈希)

4.2 API测试模块 (`apps/api_testing`)

功能职责:

- OpenAPI/Swagger文档解析
- API端点管理
- 测试用例生成 (AI驱动)
- 测试套件管理
- 测试执行与报告

核心模型:

- `APISpecification`: API规范文档
- `APIEndpoint`: API端点信息
- `APITestCase`: 测试用例 (支持端点和场景两种类型)
- `APITestSuite`: 测试套件
- `APITestExecution`: 执行记录

核心服务:

- `APIParserService`: OpenAPI文档解析
- `ApiEndpointGenTestcaseService`: AI测试用例生成
- `HttpRunnerService`: HttpRunner脚本生成与执行

亮点特性:

- 智能场景生成 (基于业务逻辑的端到端测试)
- 多类型测试用例 (正向/负向/边界/安全)
- HttpRunner集成，支持YAML/Python脚本
- 异步执行，实时进度反馈

4.3 Web测试模块 (`apps/web_testing`)

功能职责:

- Web UI测试用例管理

- Playwright脚本生成与执行
- MidScene脚本支持 (视觉AI测试)
- 测试套件执行
- Allure报告生成

核心模型:

- `webUITestCase`: Web测试用例
- `webUITestsuite`: 测试套件
- `webUITestExecution`: 执行记录
- `MidSceneScript`: MidScene脚本

核心服务:

- `PlaywrightRunner`: Playwright脚本执行器
- `WebUIPlaywrightAgent`: AI驱动的Playwright脚本生成

亮点特性:

- AI驱动的测试用例生成 (LangGraph工作流)
- 支持Playwright和MidScene两种执行方式
- 实时执行进度 (WebSocket)
- Allure报告集成

4.4 AI核心模块 (`apps/ai_core`)

功能职责:

- 多LLM模型管理 (OpenAI/DeepSeek/Ollama/Qwen等)
- 视觉模型支持 (Qwen-VL/UI-TARS/Doubao Vision等)
- RAG知识库管理
- MCP (Model Context Protocol) 支持
- AI Agent工作流编排

核心模型:

- `LLMConfiguration`: LLM/视觉模型配置
- `RAGConfiguration`: RAG配置 (向量数据库)
- `MCPConfiguration`: MCP配置
- `LLMUsageLog`: LLM使用日志

核心服务:

- `ModelManager`: 多模型兼容管理器
- `RAGService`: RAG知识库服务
- `webUITestCaseGenerator`: Web UI测试用例生成器 (LangGraph)
- `ApiEndpointGenTestcaseService`: API测试用例生成服务
- `MidSceneAgent`: MidScene智能体

亮点特性:

- 统一的多模型管理 (LLM + Vision)
- 数据库驱动的配置管理 (无需环境变量)
- LangGraph工作流编排 (多Agent协作)

- RAG知识库增强 (ChromaDB/Milvus)
- MCP协议支持 (扩展AI能力)
- Token使用统计与日志

4.5 定时任务模块 (apps/scheduled_tasks)

功能职责:

- 统一的任务调度中心
- Cron表达式支持
- 多类型测试套件定时执行
- 执行历史与日志

核心模型:

- `ScheduledTask`: 定时任务配置
- `TaskExecutionLog`: 执行日志

核心服务:

- Celery Beat集成
- 动态任务调度

亮点特性:

- 统一调度中心 (Web/API/App测试)
- Cron表达式配置
- 执行历史追踪
- 支持多套件批量执行

4.6 用户管理模块 (apps/users)

功能职责:

- 用户认证与授权
- JWT Token管理
- 用户信息管理

核心特性:

- JWT认证 (Access Token + Refresh Token)
- Token黑名单机制
- 自定义用户模型

4.7 通用模块 (apps/common)

功能职责:

- WebSocket服务
- 异常处理
- 文件服务
- 任务工具函数

核心服务:

- `WebSocketService`: WebSocket消息推送
 - `FileService`: 文件上传/下载
 - `Taskutils`: Celery任务工具函数
-