

1. 引言

1.1 目的

本文档将详细说明植保精灵（PGuard）系统单元测试的计划、代码静态测试结果、测试用例的分析和设计过程、使用的测试方法、测试用例的执行过程以及缺陷的跟踪。最终确保植保精灵（PGuard）系统在实际部署前达到设计和功能要求，全面辅助农民、农业研究人员及种植爱好者，高效完成农田作物与园艺作物的病虫害精准检测及地块智能监管。

本文档的读者主要是开发(测试)经理、测试人员和开发人员。

1.2 测试策略

以类为单元，采用独立的单元测试策略，通过设计相应的驱动和桩的方法来测试类中的方法。在选择类的被测方法时，根据方法的规模和复杂度进行判定。非空非注释代码行数LOC>20,或者复杂度VG>3的方法进行单元测试，其他方法不进行单元测试。

执行单元测试的次序是根据《软件详细设计说明书》中的用例实现交互图，从图中最小依赖关系的类开始测试，再逐步扩大到依赖关系较强的类，直至所有类测试完毕。

1.3 范围

单元测试主要包含了计划阶段、设计阶段、实现阶段和执行阶段4个阶段。本单元测试计划是整个软件开发项目中的一部分，起始于详细设计阶段，直到单元测试阶段结束后终止。该计划主要处理与植保精灵（PGuard）系统单元测试有关的任务安排、资源需求、人力需求、风险管理、进度安排等内容。

1.4 参考文献

- 《软件需求规格说明书（Software Requirement Specification）》
- 《软件详细设计说明书（Software Design Descriptions）》
- 《用户界面规格说明书（User Interface Specification）》

1.5 术语

2. 测试项目

根据《软件详细设计说明书》中的详细设计内容，单元测试的**测试项目**如以下所示：

2.1 用户管理模块

设计类标识： `User` 设计类

方法标识符	方法名	代码行 (LOG)	复杂度(VG)
<code>get_token(form:SignInForm)</code>	<code>get_token</code>	16	4
<code>refresh_token(current_token:str, refresh_token:str)</code>	<code>refresh_token</code>	17	4

2.2 地块管理模块

设计类标识： `Plot` 设计类

方法标识符	方法名	代码行 (LOG)	复杂度(VG)
<code>get_all_plots(user: User)</code>	<code>get_all_plots</code>	14	4
<code>add_plot(plotName: str, plantName: str, user: User)</code>	<code>add_plot</code>	12	4
<code>get_plot_detail(plotId: str, user: User)</code>	<code>get_plot_detail</code>	15	4

2.3 智能检测模块

设计类标识： `Detect` 设计类

方法标识符	方法名	代码行 (LOG)	复杂度(VG)
<code>detect(model_type: str, image_path: str)</code>	<code>detect</code>	22	6
<code>do_detect(plotId: str, file: UploadFile, user: User)</code>	<code>do_detect</code>	27	5

2.4 统计分析模块

设计类标识： `Log` 设计类

方法标识符	方法名	代码行 (LOG)	复杂度(VG)
-------	-----	-----------	---------

<code>analyze_plot_details(plot_details: List[PlotDetails])</code>	<code>analyze_plot_details</code>	35	12
<code>get_summary(user: User)</code>	<code>get_summary</code>	17	5

3. 测试方法

总体上根据类规约和操作规约构建测试用例。

根据实际情况合理选择使用边界值分析法、等价类测试法、决策表测试法、正交实验法等黑盒测试方法以及语句覆盖、路径覆盖等白盒测试方法来构建基本的测试用例。

对于具有特殊需求的某些类可以辅以下面两种方法构造测试用例以做补充：

1. **状态转移测试方法：**对于状态转换的类可以使用状态转移测试的方法，即通过类的状态图建立状态转移树，进而构造出补充的测试用例；
2. **基于实现的测试方法：**借助传统逻辑覆盖测试法、数据流分析法等白盒测试方法构造补充的测试用例，以对程序的逻辑结构或数据流进行测试，来达到一定的代码覆盖率。

4. 测试通过/失败标准

测试**通过**的标准表述如下：

1. 所有单元测试的用例都被执行并通过；
2. 所有发现的缺陷都被修正并通过回归测试；
3. 所有被测对象的前置条件和后置条件组合覆盖率达到100%，或能明确给出不需要达到的理由；
4. 单元测试报告被授权人批准。

测试**失败**标准表述如下：

1. 严重缺陷密度大于15个/KLOC；
2. 发现软件结构有重大设计问题，其修改会导致20%以上的接口、功能、数量的变化，进一步测试相关特性已经无意义；
3. 发现关键功能未被设计，该功能的设计会导致20%以上的接口、功能、数量的变化，进一步测试相关特性已经无意义。

测试结果审批过程：开发人员提交单元测试报告一开发或测试经理签字并提交 SQA→SQA对报告进行评审并签字（测试经理参与）→产品经理签字。

5. 测试挂起/恢复的条件

测试**挂起**的条件有：

1. 当某个类在单元测试执行过程中发现有阻塞用例的时候，该类的单元测试被挂起；
2. 当有20%以上的被测类都遇到有阻塞用例时，所有类的单元测试都被挂起；
3. 当出现有新增需求的时候，与该需求相关的所有类的单元测试都被挂起；
4. 当开发人员提出要进行设计变更的时候，相关类的单元测试将被挂起。

测试**恢复**的条件有：

1. 测试被挂起的条件已经被解决；
2. 需要恢复测试的对象达到单元测试入口条件，在这里要求这些被测对象已经通过代码走读（要提交走读报告）和语法检查（要提交检查结果）。

6. 单元测试交付物

- 单元测试计划
- 单元测试设计规格
- 单元测试用例规格
- 单元测试用例脚本
- 单元测试驱动和桩代码
- 单元测试执行日志
- 单元测试报告

7. 单元测试任务

单元测试任务表如下表所示：

任务标识	任务描述	责任人	优先级	依赖关系
UT-TASK-001	单元测试计划制定	测试经理	高	无
UT-TASK-003	单元测试计划评审	SQA	中	UT-TASK-001
UT-TASK-005	单元测试计划修改	测试经理	中	UT-TASK-003
UT-TASK-007	单元测试设计规格制定	开发或测试人员	中	UT-TASK-003
UT-TASK-009	单元测试设计规格评审	SQA	中	UT-TASK-007

UT-TASK-011	单元测试设计规格修改	开发或测试人员	中	UT-TASK-009
UT-TASK-013	单元测试用例规格设计	开发或测试人员	高	UT-TASK-009
UT-TASK-015	单元测试用例规格评审	SQA	中	UT-TASK-013
UT-TASK-017	单元测试用例规格修改	开发或测试人员	中	UT-TASK-015
UT-TASK-019	单元测试驱动、桩、用例脚本代码实现	开发或测试人员	中	UT-TASK-015
UT-TASK-021	驱动、桩、脚本代码走查	SQA	低	UT-TASK-019
UT-TASK-023	驱动、桩、脚本代码修改	开发或测试人员	低	UT-TASK-021
T-TASK-025	单元测试执行及回归	开发或测试人员	高	UT-TASK-023
UT-TASK-027	单元测试报告	测试经理	高	UT-TASK-025
UT-TASK-029	单元测试报告审批	开发或测试经理	高	UT-TASK-027

8. 环境需求

硬件需求	2核CPU、2G内存的服务器
软件需求	Python 3.8
测试工具	FastAPI TestClient

9. 角色和职责

单元测试角色和职责如下表所示。

角色	职责
产品经理	解决资源需求，对单元测试结果进行监督。

开发经理	协助制定单元测试计划，安排单元测试任务。
测试经理	指定单元测试计划，安排单元测试任务，参与单元测试结果验收。
SQA	对单元测试过程进行监控。
开发和测试人员	完成单元测试需要的输入，并完成单元测试设计规格、单元测试用例规格、单元测试规程的制定，执行单元测试，记录发现问题，修改问题，并负责问题的回归测试。与此同时，负责定位和解决问题。

10. 代码静态测试

10.1 静态测试计划

10.1.1 测试目的

对PGuard农作物疫病监测系统进行全面静态测试，发现并解决潜在的代码质量问题；确保系统符合编码规范和最佳实践，提高代码可维护性和可读性；识别可能引起性能瓶颈、安全漏洞或逻辑错误的问
题；为后续动态测试和系统优化提供基础支持。

10.1.2 测试范围

- 本静态测试测试内容**包含**以下部分：
 - 后端组件：包括API接口、数据处理模块、数据库交互层等服务端代码。
 - 前端组件：包括用户界面、前端业务逻辑、交互组件等客户端代码。
 - 检测算法组件：基于YOLO的疫病检测算法代码。
- 本静态测试**不包含**以下内容：
 - 第三方库和框架的源代码。
 - 数据库结构和性能测试。
 - 动态功能测试和集成测试。

10.1.3 测试工具

- 后端与算法测试工具：
 - PyCharm Professional 2023.1 IDE 内置代码检查
 - Pylint 2.15.0 (Python代码规范检查工具)
- 前端测试工具：
 - ESLint 8.30.0 (JavaScript代码规范检查)

10.1.4 测试标准与规范

- **编码规范：**
 - Python代码遵循PEP8编码规范
 - JavaScript代码遵循eslint推荐标准
 - Vue组件遵循vue/vue3-essential与vue推荐标准
- **质量指标：**
 - 静态分析问题：严重和高优先级问题必须全部修复
 - 安全漏洞：严重和高风险漏洞必须全部修复

10.2 工具辅助静态测试结果

10.2.1 后端组件工具静态测试

1. 总体检查结果

- 警告（Warning）：6 个
- 弱警告（Weak Warning）：43 个
- 拼写错误（Spelling Mistake）：3 个

2. 问题分组与说明

a. PEP8 命名约定违规

- 共计：36 个弱警告
- 问题类型：
 - 函数变量命名不符合规范（应为小写字母）
 - 函数形参命名不规范（应为小写字母）
- 主要涉及文件：
 - admin.py：11 个
 - detect.py：3 个
 - detectController.py：3 个
 - log.py：1 个
 - logController.py：4 个
 - plot.py：共计 12 个
 - plotController.py：2 个

b. 异常处理不明确

- 共计：2 个弱警告
- 问题类型：使用了过于宽泛的 except 子句

- 涉及文件：user.py、userController.py

c. 弃用函数或时间处理不规范

- 共计：6 个警告
- 问题类型：未使用时区感知型时间对象，应使用 `datetime.now(datetime.UTC)`
- 涉及文件：
 - user.py：3 个
 - userController.py：3 个

d. 作用域覆盖问题

- 共计：3 个弱警告
- 问题类型：局部变量名称遮蔽外部作用域中的变量（如 `refresh_token`）
- 涉及文件：user.py

e. 重复代码片段

- 共计：2 个弱警告
- 涉及文件：
 - user.py：重复代码行 22–60
 - userController.py：重复代码行 11–49

f. 拼写错误

- 共计：3 个拼写错误
- 关键词：yolov（应为 yolov5 或类似）
- 涉及文件：
 - config.py：2 个
 - detect.py：1 个

10.2.2 前端组件工具静态测试

1. 总体检查结果

- 错误（Error）：7 个
- 警告（Warning）：9 个
- 弱警告（Weak Warning）：58 个
- 拼写错误（Spelling Mistake）：22 个

2. 问题分组与说明

a. JavaScript 和 TypeScript

共计：7 个错误，9 个警告，58 个弱警告

b. 异步代码与 Promise

- 主要问题：异步函数调用未使用 await
- 涉及文件（弱警告 57 个）：
 - alert.ts: 1 处
 - HomePage.vue: 8 处
 - InfoPage.vue: 21 处
 - LoginPage.vue: 4 处
 - refresh.ts: 6 处
 - SignupPage.vue: 5 处
 - StripPage.vue: 12 处
- 其他问题：main.ts 中 startTokenRefreshTask() 返回的 Promise 被忽略（1 个弱警告）

c. 导入与依赖

- 问题类型：未使用的 import（共 4 个警告）
- 涉及文件：
 - HomePage.vue: 未使用 nextTick
 - InfoPage.vue: 未使用 onMounted
 - StripPage.vue: 未使用 nextTick
 - TabsPage.vue: 未使用 square

d. 代码质量 (ESLint)

- 共计错误：7 个
- 典型问题：
 - 未使用变量 (@typescript-eslint/no-unused-vars)
 - 多余的分号 (no-extra-semi)
 - 多余的布尔强制 (no-extra-boolean-cast)
 - 不推荐的类型 (@typescript-eslint/ban-types)
- 涉及文件：HomePage.vue、InfoPage.vue、LoginPage.vue、refresh.ts、StripPage.vue、TabsPage.vue、weather.ts

e. 未使用的符号

- 默认导出未使用：capacitor.config.ts
- 未使用函数：getAccessToken in refresh.ts

- 未使用形参：on、config in cypress.config.ts

f. 拼写错误（共 22 个）

常见错误词：

- Responce（应为 Response）
- confirmpassword（应为 confirmPassword）
- 其他驼峰命名中未分词拼写（如 dayweather, nighttemp, reporttime 等）

10.2.3 算法组件静态工具测试

1. 总体检查结果

- 拼写错误（Spelling Mistake）：2 个
- 其他类型问题：无（未发现命名规范、异常处理、重复代码等问题）

2. 问题分组与说明

a. 拼写错误

- 共计：2 个拼写错误
- 问题关键词：yolov（推测为 YOLOv 系列模型名称缩写）
- 涉及文件：
 - Grape.py：1 个
 - Potato.py：1 个
- 问题说明：静态检查工具将 yolov 识别为非标准单词，提示为拼写错误。

10.2.4 易受攻击的依赖项

通过IDE的易受攻击依赖项扫描，发现两个漏洞：

- vite@5.2.14（中危）
- axios@1.7.9（高危）

其中vite该版本具体存在以下漏洞：默认 CORS 设置不安全，可能被恶意网站读取开发服务器响应，特殊 URL 请求可绕过路径检查读取任意文件，恶意用户可通过 URL 参数访问非允许文件内容。

Axios在该版本具体存在以下漏洞：在使用绝对 URL 时，Axios 会绕过 baseUrl，可能引发 SSRF（服务器端请求伪造）与凭据泄露。此漏洞同时影响浏览器和 Node.js 环境。

10.3 问题分析与改进建议

10.3.1 后端改进建议

1. 代码规范与结构

- 修正所有PEP8命名规范违规问题，特别是admin.py和plot.py中的命名规范问题。
- 为所有函数添加文档字符串，说明功能、参数和返回值。

2. 安全性

- 更新依赖项vite及axios版本。

3. 时间处理

- 使用datetime.now(datetime.UTC)替代当前的时间处理方法，确保时区一致性

10.3.2 前端改进建议

1. 代码质量

- 修复所有ESLint报告的错误和警告
- 规范化命名约定，特别是驼峰式命名问题
- 清理未使用的导入和变量

2. 异步处理

- 添加所有缺失的await关键字到异步函数调用中
- 实现统一的Promise错误处理机制
- 添加加载状态指示和错误反馈

10.4 结论与总结

10.4.1 主要发现

通过本次对PGuard农作物疫病监测系统的静态测试，我们发现系统代码质量整体处于中等水平，存在多个**需要改进**的方面：

1. 代码规范遵循：

- 后端代码存在36个PEP8命名规范违规。
- 前端代码存在大量ESLint警告和错误，特别是在异步代码处理方面。

2. 安全与风险：

- 存在2个依赖项安全漏洞。

10.4.2 总体评价

PGuard系统的代码质量需要进一步提升，特别是在安全性、代码规范性和架构设计方面。虽然算法组件问题相对较少，但后端和前端组件存在较多的问题需要解决。系统整体未达到预定的代码质量目标，但大多数问题都是可修复的，不会影响系统的核心功能实现。

11. 测试用例设计及分析

11.1 用户管理模块

用户管理模块测试函数（方法）参考表：

标识符	名称	代码行（LOG）
LLD_001_FUN_001	get_token(form: SignInForm)	16
LLD_001_FUN_002	refresh_token(current_token: str = Depends(oauth2_scheme), refresh_token: str = Body(...))	17

11.1.1 get_token 测试分析与设计

1. 标识符定义

UT_TC_001_001

2. 被测特性

- 输入合法用户名和密码时，成功返回 token
- 用户名不存在时，返回 token 失败
- 用户名与密码不匹配时，返回 token 失败
- 输入任意参数为空时，返回 token 失败
- 数据库查询失败时，返回 token 失败

3. 测试方法

- 等价类测试法
 - 用户名参数的等价类划分考虑空和非空情况，对于非空情况，又可以划分为数据库中存在和不存在两种情况。
 - 密码参数的等价类划分考虑空和非空情况，对于非空情况，又可以划分为与用户名匹配与不匹配两种情况。
- 错误推测法

错误推测法关注非预期路径，针对 get_toekn 函数，需测试数据库连接失败的异常场景。

4. 测试项标识

测试项标识符	测试项描述	优先级
UT_TC_001_001_001	输入任意参数为空的情况	低
UT_TC_001_001_002	输入密码和用户名不匹配的情况	高

UT_TC_001_001_003	输入用户名不存在的情况	中
UT_TC_001_001_004	输入参数合法的情况	高
UT_TC_001_001_005	数据库连接失败的情况	低

5. 测试通过/失败标准

所有用例都必须被执行，且没有发现错误。

6. 对应用例

测试项编号	UT_TC_001_001_001	
优先级	低	
测试项描述	输入任意参数为空的情况	
前置条件	用户进入登录界面	
测试方法	等价类+边界值	
用例序号	输入	期望结果
001	SignInForm.userName="Jelly1106" SignInForm.password=""	400 错误，提示用户名不能为空
002	SignInForm.userName="" SignInForm.password="lyy3366291"	400 错误，提示密码不能为空

测试项编号	UT_TC_001_001_002	
优先级	高	
测试项描述	输入密码和用户名不匹配的情况	
前置条件	用户进入登录界面	
测试方法	等价类	
用例序号	输入	期望结果
001	SignInForm.userName="Jelly1106" SignInForm.password="12345678"	400 错误，提示密码错误

测试项编号	UT_TC_001_001_003	
优先级	中	
测试项描述	输入用户名不存在的情况	
前置条件	用户进入登录界面	
测试方法	等价类	
用例序号	输入	期望结果
001	SignInForm.userName="Jelly1234" SignInForm.password="lyy3366291"	400 错误，提示用户不存在

测试项编号	UT_TC_001_001_004	
优先级	高	
测试项描述	输入参数合法的情况	
前置条件	用户进入登录界面	
测试方法	等价类	
用例序号	输入	期望结果
001	SignInForm.userName="Jelly1106" SignInForm.password="lyy3366291"	返回 <code>access_token</code> , <code>refresh_token</code> 两个令牌 以及 <code>token_type: bearer</code>

测试项编号	UT_TC_001_001_005	
优先级	低	
测试项描述	数据库连接失败的情况	
前置条件	用户进入登录界面	
测试方法	错误推测法	
用例序号	输入	期望结果
001	SignInForm.userName="Jelly1106" SignInForm.password="lyy3366291"	返回500 错误以及详细的错误信息

11.1.2

refresh_token

测试分析与设计

1. 标识符定义

UT_TC_001_002

2. 被测特性

- 输入合法的 `current_token` 和 `refresh_token` 时，成功返回新 `token`
- `refresh_token` 解码失败，返回新 `token` 失败
- `refresh_token` 无 `sub` 字段，返回新 `token` 失败
- `current_token` 已在黑名单中，返回新 `token` 失败
- 新 `token` 生成失败

3. 测试方法

- 等价类测试法
 - `refresh_token` 的等价类划分考虑有效和无效情况，对于无效情况，又可以划分为无法解码和无 `sub` 字段两种情况。
 - `current_token` 的等价类划分考虑未在黑名单中和已在黑名单中情况。
- 错误推测法

考虑新 `token` 生成失败的异常场景。

4. 测试项标识

测试项标识符	测试项描述	优先级
UT_TC_001_002_001	输入合法的 <code>current_token</code> 和 <code>refresh_token</code>	高
UT_TC_001_002_002	<code>refresh_token</code> 无法解码	高
UT_TC_001_002_003	<code>refresh_token</code> 无 <code>sub</code> 字段	高
UT_TC_001_002_004	<code>current_token</code> 已在黑名单中	中
UT_TC_001_002_005	新 <code>token</code> 生成失败的情况	高

5. 测试通过/失败标准

所有用例都必须被执行，且没有发现错误。

6. 对应用例

测试项编号	UT_TC_001_002_001

优先级	高	
测试项描述	输入合法的 <code>current_token</code> 和 <code>refresh_token</code>	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	<code>current_token=""</code> <code>refresh_token=""</code>	返回 200 和 <code>access_token</code>

测试项编号	UT_TC_001_002_002	
优先级	高	
测试项描述	<code>refresh_token</code> 无法解码	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	<code>current_token=""</code> <code>refresh_token=""</code>	401 错误，提示"无效的refresh token"

测试项编号	UT_TC_001_002_003	
优先级	高	
测试项描述	<code>refresh_token</code> 无 <code>sub</code> 字段	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	<code>current_token=""</code> <code>refresh_token=""</code>	401 错误，提示"无效的refresh token"

测试项编号	UT_TC_001_002_004	

优先级	中	
测试项描述	current_token 已在黑名单中	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	current_token="" refresh_token=""	401, "无效的access token"

测试项编号	UT_TC_001_002_005	
优先级	高	
测试项描述	新token生成失败的情况	
前置条件	用户已经登录成功	
测试方法	错误推测法	
用例序号	输入	期望结果
001	current_token="" refresh_token=""	500 错误，显示详细的错误信息

11.2 地块管理模块

11.2.1 get_all_plots 测试分析与设计

1. 标识符定义

UT_TC_002_001

2. 被测特性

- 当用户拥有地块时，成功返回所有地块列表
- 当用户没有地块时，返回提示信息
- 图片URL验证处理功能正常
- 用户认证失败时，无法获取地块信息
- 数据库查询失败时，返回获取地块失败

3. 测试方法

- 等价类测试法
 - 用户地块数量的等价类划分：无地块(0)、单个地块(1)、多个地块(>1)
 - 图片URL的等价类划分：有效URL、无效URL
 - 用户认证的等价类划分：有效用户、无效用户

● 错误推测法

错误推测法关注非预期路径，针对get_all_plots函数，需测试数据库连接失败的异常场景和用户认证失败的场景。

4. 测试项标识

测试项标识符	测试项描述	优先级
UT_TC_002_001_001	用户无地块的情况	中
UT_TC_002_001_002	用户拥有单个地块的情况	高
UT_TC_002_001_003	用户拥有多个地块的情况	高
UT_TC_002_001_004	地块包含无效图片URL的情况	中
UT_TC_002_001_005	用户认证失败的情况	高
UT_TC_002_001_006	数据库查询失败的情况	低

5. 测试通过/失败标准

所有用例都必须被执行，且没有发现错误。

6. 对应用例

测试项编号	UT_TC_002_001_001	
优先级	中	
测试项描述	用户无地块的情况	
前置条件	<div>1. 用户已登录系统</div> <div>2. 用户数据库中没有关联的地块</div>	
测试方法	等价类	
用例序号	输入	期望结果
001	user=已登录用户（无地块）	{

		<div>"message": "当前地块列表为空，尚未创建地块"</div> <div>}</div>
--	--	--

测试项编号	UT_TC_002_001_002	
优先级	高	
测试项描述	用户拥有单个地块的情况	
前置条件	<div>1. 用户已登录系统</div> <div>2. 用户数据库中有一个地块记录</div>	
测试方法	等价类	
用例序号	输入	期望结果
001	user=已登录用户（有1个地块）	<div>[</div> <div>{</div> <div>"plotId": "地块ID",</div> <div>"plotName": "地块名称",</div> <div>"plantName": "植物名称",</div> <div>"plantIconURL": "图标URL"</div> <div>}</div> <div>]</div>

测试项编号	UT_TC_002_001_003	
优先级	高	
测试项描述	用户拥有多个地块的情况	
前置条件	<div>1. 用户已登录系统</div> <div>2. 用户数据库中有多多个地块记录</div>	
测试方法	等价类	
用例序号	输入	期望结果
001	user=已登录用户（有3个地块）	<div>[</div> <div>{</div> <div>"plotId": "地块1ID",</div>

	<div>"plotName": "地块1名称",</div> <div>"plantName": "植物1名称",</div> <div>"plantIconURL": "图标1URL"</div> <div>},</div> <div>{</div> <div>"plotId": "地块2ID",</div> <div>"plotName": "地块2名称",</div> <div>"plantName": "植物2名称",</div> <div>"plantIconURL": "图标2URL"</div> <div>},</div> <div>{</div> <div>"plotId": "地块3ID",</div> <div>"plotName": "地块3名称",</div> <div>"plantName": "植物3名称",</div> <div>"plantIconURL": "图标3URL"</div> <div>}</div> <div>]</div>
--	--

测试项编号	UT_TC_002_001_004	
优先级	中	
测试项描述	地块包含无效图片URL的情况	
前置条件	<div>1. 用户已登录系统</div> <div>2. 用户数据库中有地块记录，且plantIconURL字段包含无效URL</div>	
测试方法	等价类	
用例序号	输入	期望结果
001	<div>user=已登录用户（有1个地块）</div> <div>plot.plotId.plantIconURL="无效URL"</div>	<div>[</div> <div>{</div> <div>"plotId": "地块ID",</div> <div>"plotName": "地块名称",</div> <div>"plantName": "植物名称",</div> <div>"plantIconURL": "经过</div> <div>validate_image_file处理后的URL"</div>

		}
]

测试项编号	UT_TC_002_001_005	
优先级	高	
测试项描述	用户认证失败的情况	
前置条件	用户未登录或token无效	
测试方法	错误推测法	
用例序号	输入	期望结果
001	无效的用户认证信息	401 错误，提示"未经授权的访问"

测试项编号	UT_TC_002_001_006	
优先级	低	
测试项描述	数据库查询失败的情况	
前置条件	1. 用户已登录系统 2. 数据库连接异常或查询操作失败	
测试方法	错误推测法	
用例序号	输入	期望结果
001	user=已登录用户 数据库连接异常	500 错误，提示"获取地块失败"及详细错误原因

11.2.2 add_plot 测试分析与设计

1. 标识符定义

UT_TC_002_002

2. 被测特性

- 当用户输入有效的地块名称和存在的植物名称时，成功创建地块
- 当输入的植物名称不存在时，返回404错误
- 当输入的参数无效时，返回400错误

- 当数据库操作失败时，返回500错误
- 用户认证失败时，无法创建地块

3. 测试方法

- 等价类测试法
 - 地块名称的等价类划分：有效地块名称、无效地块名称（如空字符串）
 - 植物名称的等价类划分：存在的植物名称、不存在的植物名称
 - 用户认证的等价类划分：有效用户认证、无效用户认证
- 错误推测法
 - **错误推测法**关注非预期路径，针对add_plot函数，需测试数据库连接失败的异常场景。

4. 测试项标识

测试项标识符	测试项描述	优先级
UT_TC_002_002_001	成功创建地块的情况	高
UT_TC_002_002_002	植物名称不存在的情况	中
UT_TC_002_002_003	输入参数无效的情况	低
UT_TC_002_002_004	用户认证失败的情况	高
UT_TC_002_002_005	数据库操作失败的情况	低

5. 测试通过/失败标准

所有用例都必须被执行，且没有发现错误。

6. 对应用例

测试项编号	UT_TC_002_002_001	
优先级	高	
测试项描述	成功创建地块的情况	
前置条件	<div>1. 用户已登录系统</div> <div>2. 数据库中存在指定植物</div>	
测试方法	等价类	
用例序号	输入	期望结果

001	<div>user=已登录用户</div> <div>plotName="测试地块"</div> <div>plantName="萝卜"</div>	<div>{</div> <div>"plotId": "生成的ID",</div> <div>"plotName": "测试地块",</div> <div>"plantId": "植物ID",</div> <div>"plantName": "萝卜",</div> <div>"message": "地块创建成功"</div> <div>}</div>
-----	--	---

测试项编号	UT_TC_002_002_002	
优先级	中	
测试项描述	植物名称不存在的情况	
前置条件	<div>1. 用户已登录系统</div> <div>2. 数据库中不存在指定植物</div>	
测试方法	等价类	
用例序号	输入	期望结果
001	<div>user=已登录用户</div> <div>plotName="测试地</div> <div>块"plantName="不存在的植物"</div>	404 错误，提示"未找到指定植物"

测试项编号	UT_TC_002_002_003	
优先级	低	
测试项描述	输入参数无效的情况	
前置条件	<div>1. 用户已登录系统；</div>	
测试方法	等价类	
用例序号	输入	期望结果
001	<div>user=已登录用户</div> <div>plotName=""</div> <div>plantName="萝卜"</div>	400 错误，提示参数无效
002	<div>user=已登录用户</div>	400 错误，提示参数无效

	plotName="测试地块"	
	plantName=""	

测试项编号	UT_TC_002_002_004	
优先级	高	
测试项描述	用户认证失败的情况	
前置条件	1. 用户未登录或token无效；	
测试方法	错误推测法	
用例序号	输入	期望结果
001	无效的用户认证信息 plotName="测试地块" plantName="萝卜"	401 错误，提示"未经授权的访问"

测试项编号	UT_TC_002_002_005	
优先级	低	
测试项描述	数据库操作失败的情况	
前置条件	1. 用户已登录系统； 2. 数据库连接异常或操作失败	
测试方法	错误推测法	
用例序号	输入	期望结果
001	user=已登录用户 plotName="测试地块"、 plantName="萝卜" 数据库连接异常	500 错误，提示"创建地块失败"及详细错误原因

11.2.3 get_plot_detail 测试分析与设计

1. 标识符定义

UT_TC_002_003

2. 被测特性

- 当用户输入有效的地块ID且拥有该地块访问权限时，成功返回地块详情
- 当输入的地块ID格式无效时，返回400错误
- 当输入的地块ID不存在或用户无权访问该地块时，返回404错误
- 当获取地块日志（call_get_logs）失败时，可能影响返回结果
- 当图片URL无效时，validate_image_file函数会处理
- 当数据库操作失败时，返回500错误
- 用户认证失败时，无法获取地块详情

3. 测试方法

- 等价类测试法
 - 地块ID的等价类划分：有效的地块ID且属于用户、有效的地块ID但不属于用户、无效的地块ID格式
 - 图片URL的等价类划分：有效URL、无效URL
 - 用户认证的等价类划分：有效用户认证、无效用户认证
- 错误推测法
 - **错误推测法**关注非预期路径，针对get_plot_detail函数，需测试call_get_logs失败和数据库连接异常的场景。

4. 测试项标识

测试项标识符	测试项描述	优先级
UT_TC_002_003_001	成功获取地块详情的情况	高
UT_TC_002_003_002	地块ID格式无效的情况	中
UT_TC_002_003_003	地块不存在或无权访问的情况	高
UT_TC_002_003_004	地块图片URL无效的情况	中
UT_TC_002_003_005	用户认证失败的情况	高
UT_TC_002_003_006	数据库查询失败的情况	低
UT_TC_002_003_007	获取地块日志失败的情况	低

5. 测试通过/失败标准

所有用例都必须被执行，且没有发现错误。

6. 对应用例

测试项编号	UT_TC_002_003_001	
优先级	高	
测试项描述	成功获取地块详情的情况	
前置条件	<div><div>1.</div>用户已登录系统</div> <div><div>2.</div>数据库中存在用户有权访问的地块</div>	
测试方法	等价类	
用例序号	输入	期望结果
001	<div>user=已登录用户</div> <div>plotId="用户拥有的有效地块ID"</div>	返回包含完整地块详情的PlotDetails对象，包括 <div>plotId</div> ， <div>plotName</div> ， <div>plantId</div> ， <div>plantName</div> ， <div>plantFeature</div> ， <div>plantIconURL</div> 和 <div>logs</div> 字段

测试项编号	UT_TC_002_003_002	
优先级	中	
测试项描述	地块ID格式无效的情况	
前置条件	用户已登录系统	
测试方法	等价类	
用例序号	输入	期望结果
001	<div>user=已登录用户</div> <div>plotId="非UUID格式字符串"</div>	400 错误，提示"无效的地块ID格式"及相关错误详情

测试项编号	UT_TC_002_003_003	
优先级	高	
测试项描述	地块不存在或无权访问的情况	
前置条件	<div><div>1.</div>用户已登录系统</div> <div><div>2.</div>地块ID有效但不属于该用户，或地块不存在</div>	
测试方法	等价类	

用例序号	输入	期望结果
001	user=已登录用户 plotId="不属于该用户的有效地块ID"	404 错误，提示"未找到地块或无权访问"
002	user=已登录用户 plotId="不存在的有效地块ID格式"	404 错误，提示"未找到地块或无权访问"

测试项编号	UT_TC_002_003_004	
优先级	中	
测试项描述	地块图片URL无效的情况	
前置条件	<div>1. 用户已登录系统</div> <div>2. 数据库中存在用户有权访问的地块</div> <div>3. 地块关联的植物有无效的图片URL</div>	
测试方法	等价类	
用例序号	输入	期望结果
001	user=已登录用户 plotId="用户拥有的有效地块ID" 植物图片URL无效	返回地块详情，但 plantIconURL 字段为 validate_image_file 处理后的值

测试项编号	UT_TC_002_003_005	
优先级	高	
测试项描述	用户认证失败的情况	
前置条件	用户未登录或token无效	
测试方法	错误推测法	
用例序号	输入	期望结果
001	user=无效用户认证 plotId="任意地块ID"	401 错误，提示"未经授权的访问"

测试项编号	UT_TC_002_003_006	

优先级	低	
测试项描述	数据库操作失败的情况	
前置条件	<div><div>1.</div>用户已登录系统</div> <div><div>2.</div>数据库连接异常或操作失败</div>	
测试方法	错误推测法	
用例序号	输入	期望结果
001	<div>user=已登录用户</div> <div>plotId="有效地块ID"</div> <div>数据库连接异常</div>	500 错误，提示"获取地块详情失败"及详细错误原因

测试项编号	UT_TC_002_003_007	
优先级	低	
测试项描述	获取地块日志失败的情况	
前置条件	<div><div>1.</div>用户已登录系统</div> <div><div>2.</div>数据库中存在用户有权访问的地块</div> <div><div>3.</div>call_get_logs函数调用失败</div>	
测试方法	错误推测法	
用例序号	输入	期望结果
001	<div>user=已登录用户</div> <div>plotId="用户拥有的有效地块ID"</div> <div>日志服务不可用</div>	500错误，提示"获取地块日志失败"及详细错误原因

11.3 智能检测模块

11.3.1 detect 测试分析与设计

1. 标识符定义

UT_TC_003_001

2. 被测特性

- 葡萄相关检测

- 检测health情况
- 检测black measles情况
- 检测black rot情况
- 检测leaf blight情况
- 马铃薯相关检测
 - 检测health情况
 - 检测late blight情况
 - 检测early blight情况

3. 测试方法

- 等价类划分：检测各类 disease 是否正确识别并返回
- 异常路径测试：非法 model_type 输入

4. 测试项标识

测试项标识符	测试项描述	优先级
UT_TC_003_001_001	正确识别 Grape 的各类病害结果	高
UT_TC_003_001_002	正确识别 Potato 的各类病害结果	高
UT_TC_003_001_003	model_type 非法，抛出 ValueError	中

5. 测试通过/失败标准

所有用例都成功执行

6. 对应用例

测试项编号	UT_TC_003_001_001	
优先级	高	
测试项描述	识别葡萄的各种情况	
前置条件	用户已经登录成功，用户成功创建地块，用户正常上传了图片	
测试方法	等价类	
用例序号	输入	期望结果
001	file=health的图片	{

	model_type="Grape"	<div>"disease": "health",</div> <div>"confidence": 较高</div> <div>}</div>
002	file=black measles的图片 model_type="Grape"	<div>{</div> <div>"disease": "black measles",</div> <div>"confidence": 较高</div> <div>}</div>
003	file=black rot的图片 model_type="Grape"	<div>{</div> <div>"disease": "black rot",</div> <div>"confidence": 较高</div> <div>}</div>
004	file=leaf blight的图片 model_type="Grape"	<div>{</div> <div>"disease": "leaf blight",</div> <div>"confidence": 较高</div> <div>}</div>

测试项编号	UT_TC_003_001_002	
优先级	高	
测试项描述	识别马铃薯的各种情况	
前置条件	用户已经登录成功，用户成功创建地块，用户正常上传了图片	
测试方法	等价类	
用例序号	输入	期望结果
001	file=health的图片 model_type="Potato"	<div>{</div> <div>"disease": "health",</div> <div>"confidence": 较高</div> <div>}</div>
002	file=early blights的图片 model_type="Potato"	<div>{</div> <div>"disease": "early blight",</div> <div>"confidence": 较高</div> <div>}</div>

003	file=late blight的图片 model_type="Potato"	<pre>{ "disease": "late blight", "confidence": 较高 }</pre>
004	file=leaf blight的图片 model_type="Potato"	<pre>{ "disease": "leaf blight", "confidence": 较高 }</pre>

测试项编号	UT_TC_003_001_003	
优先级	中	
测试项描述	异常模型参数	
前置条件	用户已经登录成功，用户成功创建地块，用户正常上传了图片	
测试方法	错误推测法	
用例序号	输入	期望结果
001	model_type="Apple"	提示"非法模型类型"

11.3.2 do_detect 测试分析与设计

1. 标识符定义

UT_TC_003_002

2. 被测特性

- 用户打开有效的地块，正确格式的图片，且处于登录状态时，进行正常检测
- 用户上传的图片格式无效时，弹出提示
- 用户登录状态过期时，弹出提示
- 用户地块状态异常时，弹出提示

3. 测试方法

- 等价类测试法
 - 文件格式：格式正确、格式错误
 - 用户登陆状态：登录正常、登录异常

- 地块id：存在且属于用户、不存在、存在且不属于用户

● 错误推测法

- 输入错误格式的文件应该提示错误
- 输入错误格式的地块id应该提示错误

4. 测试项标识

测试项标识符	测试项描述	优先级
UT_TC_003_002_001	所有输入有效	高
UT_TC_003_002_002	传入文件格式错误	高
UT_TC_003_002_003	地块不存在或无权访问的情况	低
UT_TC_003_002_004	用户认证失败的情况	高

5. 测试通过/失败标准

所有用例都必须通过

6. 对应用例

测试项编号	UT_TC_003_002_001	
优先级	高	
测试项描述	所有输入有效	
前置条件	用户已经登录成功，用户成功创建地块	
测试方法	等价类	
用例序号	输入	期望结果
001	file=要检测的图片 plotId=要检测的地块 user=用户登录信息	正常返回测试结果

测试项编号	UT_TC_003_002_002	
优先级	高	
测试项描述	传入文件格式错误	

前置条件	用户已经登录成功，用户成功创建地块，提交的文件类型为非 .jpg/.jpeg/.png 的图片（如 .txt，.gif 等）	
测试方法	等价类	
用例序号	输入	期望结果
001	file = 文件格式为txt plotId = 合法 plotId user = 合法用户	返回 400 状态码，提示"文件格式错误"

测试项编号	UT_TC_003_002_003	
优先级	低	
测试项描述	地块不存在或无权访问的情况	
前置条件	用户已经登录成功	
测试方法	错误推测法	
用例序号	输入	期望结果
001	file=合法图片 plotId=非该用户地块 user=合法用户	返回 403，提示"无访问权限"
002	file=合法图片 plotId=不存在地块 user=合法用户	返回 404，提示"地块不存在"

测试项编号	UT_TC_003_002_004	
优先级	高	
测试项描述	用户认证失败	
前置条件	无	
测试方法	错误推测法	
用例序号	输入	期望结果
001	file=合法图片 plotId=合法plotId	返回 403,登录异常

11.4 统计分析模块

统计分析模块测试函数（方法）参考表：

标识符	名称	代码行（LOG）
LLD_004_FUN_001	analyze_plot_details(plot_details: List[PlotDetails])	57
LLD_004_FUN_002	reget_summary(user: User = Depends(get_current_user))	37

11.4.1 analyze_plot_details 测试分析与设计

1. 标识符定义

UT_TC_004_001

2. 被测特性

- 输入非None且元素属性合法的 `PlotDetails` 列表时，成功返回包括地块数、各作物地块数、当年月度疫病数、各作物疫病数、疫病总数和预测信息的统计信息
- `PlotDetails` 列表为None时，抛出TypeError异常
- `PlotDetails` 中有元素的时间戳字符串属性不符合格式时，服务器端记录异常日志，返回忽略异常元素后得到的统计信息

3. 测试方法

• 等价类测试法

本测试方法通过划分输入数据的等价类，确保每个等价类至少被测试一次，覆盖常见的有效和无效输入场景。

- plotDetails为None（无效等价类）：应抛出TypeError异常
- plotDetails为空列表（有效等价类）：应返回plot_count为0，monthly_disease_count全为0
- plotDetails中元素的logs属性为空（有效等价类）：应返回plot_count为1，monthly_disease_count全为0
- diseaseName为"健康"（有效等价类）：不计入病害统计
- diseaseName为具体病害（如"病害A"），时间为今年（有效等价类）：应正确计入monthly_disease_count和disease_count
- diseaseName为None（无效等价类）：只计数，不统计disease_count

- 大于0个PlotDetails和LogDetail（有效等价类）：应正确统计多个地块和日志的情况
 - logs中有今年和非今年的timeStamp（有效等价类）：只有今年的日志计入monthly_disease_count
 - 错误推测法
- 本测试方法关注于输入异常或数据异常时系统的健壮性和容错能力。
- timeStamp格式错误：应能捕获异常，monthly_disease_count不增加
 - diseaseName为None：不应导致程序崩溃，统计逻辑正常
 - plotDetails为None：应抛出TypeError异常
 - analyze_plot_details内部抛出异常时（如日期解析错误）：应能捕获并处理异常，保证测试通过或抛出预期异常

4. 测试项标识

测试项标识符	测试项描述	优先级
UT_TC_004_001_001	测试 plotDetails 为 None 时函数能正确抛出 TypeError 异常。	高
UT_TC_004_001_002	测试 plotDetails 为空列表时返回地块数为 0，月度病害统计全为 0。	高
UT_TC_004_001_003	测试地块日志属性 logs 为空时返回地块数为 1，月度病害统计全为 0。	中
UT_TC_004_001_004	测试日志 diseaseName 为“健康”时不计入病害统计。	中
UT_TC_004_001_005	测试日志 diseaseName 为“病害A”且为今年时能正确计入统计。	高
UT_TC_004_001_006	测试日志 diseaseName 为 None 时只计数不统计 disease_count。	中
UT_TC_004_001_007	测试日志 timeStamp 格式错误时不计入统计且不抛异常。	高
UT_TC_004_001_008	测试多个地块和日志时能正确统计各项数据。	高
UT_TC_004_001_009	测试日志中有今年和非今年的 timeStamp 时只统计今年的日志。	高

5. 测试通过/失败标准

所有用例都必须被执行，且通过断言。

6. 对应用例

测试项编号	UT_TC_004_001_001

优先级	高	
测试项描述	测试 plotDetails 为 None 时函数能正确抛出 TypeError 异常	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	None	500错误，抛出TypeError异常

测试项编号	UT_TC_004_001_002	
优先级	高	
测试项描述	测试 plotDetails 为空列表时返回地块数为 0，月度病害统计全为 0	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	[]	<pre>{ 'plot_count': 0, 'plant_plot_count': {}, 'monthly_disease_count': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'plant_disease_count': {}, 'disease_count': {}, 'prediction': 'mocked_prediction' }</pre>

测试项编号	UT_TC_004_001_003	
优先级	中	
测试项描述	测试地块日志属性 logs 为空时返回地块数为 1，月度病害统计全为 0	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果

001	<pre>{ 'plotId': 'p1', 'plotName': '地块1', 'plantId': 'pl1', 'plantName': '植物A', 'plantFeature': 'feature', 'plantIconURL': 'icon', 'logs': [] }</pre>	<pre>{ 'plot_count': 1, 'plant_plot_count': {'植物A': 1}, 'monthly_disease_count': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'plant_disease_count': {}, 'disease_count': {}, 'prediction': 'mocked_prediction' }</pre>

测试项编号	UT_TC_004_001_004	
优先级	中	
测试项描述	测试日志 diseaseName 为“健康”时不计入病害统计	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	<pre>{ 'plotId': 'p1', 'plotName': '地块1', 'plantId': 'pl1', 'plantName': '植物A', 'plantFeature': 'feature', 'plantIconURL': 'icon', 'logs': { 'logId': '1', 'timeStamp': '2025- 04-01 12:00:00.000', </pre>	<pre>{ 'plot_count': 1, 'plant_plot_count': {'植物A': 1}, 'monthly_disease_count': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'plant_disease_count': {}, 'disease_count': {}, 'prediction': 'mocked_prediction' }</pre>

	<pre>{ 'diseaseName': '健康', 'content': 'test', 'imagesURL': 'url' }</pre>
--	---

测试项编号	UT_TC_004_001_005	
优先级	高	
测试项描述	测试日志 diseaseName 为“病害A”且为今年时能正确计入统计	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	<pre>{ 'plotId': 'p1', 'plotName': '地块1', 'plantId': 'pl1', 'plantName': '植物A', 'plantFeature': 'feature', 'plantIconURL': 'icon', 'logs': { 'logId': '1', 'timeStamp': '2025-04-01 12:00:00.000', 'diseaseName': '病害A', 'content': 'test', 'imagesURL': 'url' } }</pre>	<pre>{ 'plot_count': 1, 'plant_plot_count': {'植物A': 1}, 'monthly_disease_count': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0], 'plant_disease_count': {'植物A': 1}, 'disease_count': {'病害A': 1}, 'prediction': 'mocked_prediction' }</pre>

测试项编号	UT_TC_004_001_006	
优先级	中	
测试项描述	测试日志 diseaseName 为 None 时应抛出TypeError异常	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	<pre>{ 'plotId': 'p1', 'plotName': '地块1', 'plantId': 'pl1', 'plantName': '植物A', 'plantFeature': 'feature', 'plantIconURL': 'icon', 'logs': { 'logId': '1', 'timeStamp': '2025- 04-01 12:00:00.000', 'diseaseName': None, 'content': 'test', 'imagesURL': 'url' } }</pre>	500错误，抛出TypeError异常

测试项编号	UT_TC_004_001_007	
优先级	高	
测试项描述	测试日志 timeStamp 格式错误时抛出异常	
前置条件	用户已经登录成功	
测试方法	错误推测法	

用例序号	输入	期望结果
001	<pre>{ 'plotId': 'p1', 'plotName': '地块1', 'plantId': 'pl1', 'plantName': '植物A', 'plantFeature': 'feature', 'plantIconURL': 'icon', 'logs': { 'logId': '1', 'timeStamp': '2025/04/01 12:00', 'diseaseName': '病害 A', 'content': 'test', 'imagesURL': 'url' } }</pre>	抛出异常

测试项编号	UT_TC_004_001_008	
优先级	高	
测试项描述	测试多个地块和日志时能正确统计各项数据	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	<pre>[{ 'plotId': 'p1', 'plotName': '地块1', 'plantId': 'pl1',</pre>	<pre>result: { 'plot_count': 2, 'plant_plot_count': {'植物A': 1, '植 物B': 1}, 'monthly_disease_count':</pre>


```
'plantName': '植物A',
'plantFeature': 'feature',
'plantIconURL': 'icon',
'logs': [
  {
    'logId': '1',
    'timeStamp': '2025-04-01 12:00:00.000',
    'diseaseName': '病害A',
    'content': 'test',
    'imagesURL': 'url'
  },
  {
    'logId': '2',
    'timeStamp': '2025-05-01 12:00:00.000',
    'diseaseName': '病害B',
    'content': 'test',
    'imagesURL': 'url'
  }
],
{
  'plotId': 'p2',
  'plotName': '地块2',
  'plantId': 'pl2',
  'plantName': '植物B',
  'plantFeature':
'feature2',
  'plantIconURL': 'icon2',
  'logs': [
    {
      'logId': '1',
      'timeStamp': '2025-04-01 12:00:00.000',
```

```
[0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0,
0],
'plant_disease_count': {'植物A': 2,
'植物B': 2},
'disease_count': {'病害A': 2, '病害
B': 2}, 'prediction':
'mocked_prediction'
}
```

	<div>'diseaseName': '病害A',</div> <div>'content': 'test',</div> <div>'imagesURL': 'url'</div> <div>},</div> <div>{</div> <div>'logId': '2',</div> <div>'timeStamp': '2025-05-01 12:00:00.000',</div> <div>'diseaseName': '病害B',</div> <div>'content': 'test',</div> <div>'imagesURL': 'url'</div> <div>}</div> <div>]</div> <div>}</div> <div>]</div>	
--	--	--

测试项编号	UT_TC_004_001_009	
优先级	高	
测试项描述	测试日志中有今年和非今年的 timeStamp 时只统计今年的日志	
前置条件	用户已经登录成功	
测试方法	等价类	
用例序号	输入	期望结果
001	<pre>{ 'plotId': 'p1', 'plotName': '地块1', 'plantId': 'pl1', 'plantName': '植物A', 'plantFeature': 'feature', 'plantIconURL': 'icon', 'logs': [{</pre>	<pre>{ 'plot_count': 1, 'plant_plot_count': {'植物A': 1}, 'monthly_disease_count': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0], 'plant_disease_count': {'植物A': 1}, 'disease_count': {'病害C': 1}, 'prediction': 'mocked_prediction' }</pre>

```
    'logId': '3',
    'timeStamp': '2024-06-01 12:00:00.000',
    'diseaseName': '病害C',
    'content': 'test',
    'imagesURL': 'url'
  },
  {
    'logId': '3',
    'timeStamp': '2025-06-01 12:00:00.000',
    'diseaseName': '病害C',
    'content': 'test',
    'imagesURL': 'url'
  }
]
```

11.4.2 get_summary 测试分析与设计

1. 标识符定义

UT_TC_004_002

2. 被测特性

- 用户余额不足时抛出异常。
- 若该用户无地块则返回统计为零的结果。

3. 测试方法

边界值分析

- 用户余额取-1, 0, 1, 大于1
- 用户地块数列表元素数为0个, 1个, 多个

4. 测试项标识

--	--	--

测试项标识符	测试项描述	优先级
UT_TC_004_002_001	用户余额数不是正值	高
UT_TC_004_002_002	用户余额数大于等于1，用户地块列表元素数为多个	高
UT_TC_004_002_003	用户地块列表元素数为1个	中
UT_TC_004_002_004	用户地块列表元素数为0个	中

5. 测试通过/失败标准

所有用例都必须被执行，且通过断言。

6. 对应用例

测试项编号	UT_TC_004_002_001	
优先级	高	
测试项描述	用户余额数不是正值	
前置条件	用户已经登录成功	
测试方法	边界值	
用例序号	输入	期望结果
001	<pre>{ 'userId': '1', 'userName': 'Fulham', 'password': 'Pass_word', 'location': 'London', 'sumCount': 0 }</pre>	400错误，响应余额不足消息
002	<pre>{ 'userId': '1', 'userName': 'Fulham', 'password': 'Pass_word', 'location': 'London', 'sumCount': -1 }</pre>	400错误，响应余额不足消息

测试项编号	UT_TC_004_002_002	
优先级	高	
测试项描述	用户余额数是正值	
前置条件	用户已经登录成功，数据库地块表中有多组元组与此用户关联	
测试方法	边界值	
用例序号	输入	期望结果
001	<pre>{ 'userId': '1', 'userName': 'Fulham', 'password': 'Pass_word', 'location': 'London', 'sumCount': 1 }</pre>	<pre>{ 'plot_count': 2, 'plant_plot_count': {'植物A': 1, '植物B': 1}, 'monthly_disease_count': [0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0], 'plant_disease_count': {'植物A': 1, '植物B': 1}, 'disease_count': {'病害A': 1, '病害B': 1}, 'prediction': 'mocked' }</pre>
002	<pre>{ 'userId': '1', 'userName': 'Fulham', 'password': 'Pass_word', 'location': 'London', 'sumCount': 10 }</pre>	<pre>{ 'plot_count': 2, 'plant_plot_count': {'植物A': 1, '植物B': 1}, 'monthly_disease_count': [0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0], 'plant_disease_count': {'植物A': 1, '植物B': 1}, 'disease_count': {'病害A': 1, '病害B': 1}, 'prediction': 'mocked' }</pre>

测试项编号	UT_TC_004_001_003
-------	-------------------

优先级	中	
测试项描述	用户地块列表元素数为1个	
前置条件	用户已经登录成功，数据库地块表中有1个元组与此用户关联	
测试方法	边界值	
用例序号	输入	期望结果
001	<pre>{ 'userId': '1', 'userName': 'Fulham', 'password': 'Pass_word', 'location': 'London', 'sumCount': 10 }</pre>	<pre>{ 'plot_count': 1, 'plant_plot_count': {'植物A': 1}, 'monthly_disease_count': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'plant_disease_count': {}, 'disease_count': {}, 'prediction': 'mocked_prediction' }</pre>

测试项编号	UT_TC_004_001_004	
优先级	中	
测试项描述	用户地块列表元素数为0个	
前置条件	用户已经登录成功，数据库地块表中无元组与此用户关联	
测试方法	边界值	
用例序号	输入	期望结果
001	<pre>{ 'userId': '1', 'userName': 'Fulham', 'password': 'Pass_word', 'location': 'London', 'sumCount': 10 }</pre>	<pre>{ 'plot_count': 0, 'plant_plot_count': {}, 'monthly_disease_count': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'plant_disease_count': {} }</pre>

12. 测试脚本

代码块

```
1  class TestGetAllPlots:
2      """get_all_plots 方法测试类"""
3
4      @pytest.mark.asyncio
5      async def test_get_all_plots_empty_list_UT_TC_002_001_001(self):
6          """用户无地块的情况 - UT_TC_002_001_001"""
7          mock_user = MagicMock()
8
9          with patch('service.plot.get_user_plots', new_callable=AsyncMock) as
mock_get_plots:
10              mock_get_plots.return_value = []
11
12              result = await get_all_plots(user=mock_user)
13
14              assert result == {"message": "当前地块列表为空，尚未创建地块"}
15              mock_get_plots.assert_called_once_with(mock_user)
16
17      @pytest.mark.asyncio
18      async def test_get_all_plots_single_plot_UT_TC_002_001_002(self):
19          """用户拥有单个地块的情况 - UT_TC_002_001_002"""
20          mock_user = MagicMock()
21
22          # 模拟地块数据
23          mock_plot = MagicMock()
24          mock_plot.plotId = "plot123"
25          mock_plot.plotName = "测试地块"
26          mock_plot.plantId.plantName = "测试植物"
27          mock_plot.plantId.plantIconURL = "http://example.com/icon.jpg"
28
29          with patch('service.plot.get_user_plots', new_callable=AsyncMock) as
mock_get_plots, \
30              patch('service.plot.validate_image_file') as mock_validate:
31
32              mock_get_plots.return_value = [mock_plot]
33              mock_validate.return_value = "http://example.com/icon.jpg"
34
35              result = await get_all_plots(user=mock_user)
36
37              assert isinstance(result, list)
38              assert len(result) == 1
39              assert result[0]["plotId"] == "plot123"
40              assert result[0]["plotName"] == "测试地块"
41              assert result[0]["plantName"] == "测试植物"
42              assert result[0]["plantIconURL"] == "http://example.com/icon.jpg"
43
```

```
44     @pytest.mark.asyncio
45     async def test_get_all_plots_multiple_plots_UT_TC_002_001_003(self):
46         """用户拥有多个地块的情况 - UT_TC_002_001_003"""
47         mock_user = MagicMock()
48
49         # 模拟多个地块数据
50         mock_plots = []
51         for i in range(3):
52             mock_plot = MagicMock()
53             mock_plot.plotId = f"plot{i}"
54             mock_plot.plotName = f"地块{i}"
55             mock_plot.plantId.plantName = f"植物{i}"
56             mock_plot.plantId.plantIconURL = f"http://example.com/icon{i}.jpg"
57             mock_plots.append(mock_plot)
58
59         with patch('service.plot.get_user_plots', new_callable=AsyncMock) as
mock_get_plots, \
60             patch('service.plot.validate_image_file') as mock_validate:
61
62             mock_get_plots.return_value = mock_plots
63             mock_validate.side_effect = lambda url: url
64
65             result = await get_all_plots(user=mock_user)
66
67             assert isinstance(result, list)
68             assert len(result) == 3
69             # 验证每个地块的基本信息
70             for i, plot in enumerate(result):
71                 assert plot["plotId"] == f"plot{i}"
72                 assert plot["plotName"] == f"地块{i}"
73                 assert plot["plantName"] == f"植物{i}"
74
75     @pytest.mark.asyncio
76     async def test_get_all_plots_invalid_image_url_UT_TC_002_001_004(self):
77         """地块包含无效图片URL的情况 - UT_TC_002_001_004"""
78         mock_user = MagicMock()
79
80         mock_plot = MagicMock()
81         mock_plot.plotId = "plot123"
82         mock_plot.plotName = "测试地块"
83         mock_plot.plantId.plantName = "测试植物"
84         mock_plot.plantId.plantIconURL = "invalid_url"
85
86         with patch('service.plot.get_user_plots', new_callable=AsyncMock) as
mock_get_plots, \
87             patch('service.plot.validate_image_file') as mock_validate:
```



```

89         mock_get_plots.return_value = [mock_plot]
90         mock_validate.return_value = "http://example.com/default.jpg"
91
92         result = await get_all_plots(user=mock_user)
93
94         assert isinstance(result, list)
95         assert len(result) == 1
96         assert result[0]["plantIconURL"] ==
97             "http://example.com/default.jpg"
98         # 验证validate_image_file被正确调用
99         mock_validate.assert_called_once_with("invalid_url")

```

13. 用例的执行及分析报告

13.1 测试的执行过程

1. 环境准备

- 安装项目运行所需依赖
- 安装pytest及相关插件以及其他可能需要的库

Environment

Python	3.8.0
Platform	Windows-10-10.0.26100-SP0
Packages	<ul style="list-style-type: none"> • pytest: 8.3.5 • pluggy: 1.5.0
Plugins	<ul style="list-style-type: none"> • anyio: 4.5.0 • asyncio: 0.24.0 • html: 4.1.1 • metadata: 3.1.1 • mock: 3.14.1

2. 编写测试用例

根据第11部分设计的测试用例进行测试用例的编写，其中需设置模拟对象（Mocks/Stubs/Fakes）来隔离被测单元，控制依赖项的行为；并创建或加载测试用例所需的特定输入数据。

3. 执行测试

在不同文件中为不同的模块编写测试用例并分别执行。

4. 记录与修复bug

- 当测试失败时，控制台会显示错误位置和差异
- 在测试用例表中记录bug
- 根据错误堆栈检查代码逻辑并解决bug

- 重跑失败用例验证修复效果

5. 生成测试报告

13.2 测试结果

13.2.1 用户管理模块

测试用例表

测试项编号	用例序号	测试用例描述	bug	bug解决情况
get_token 方法				
UT_TC_001_001_001	001	输入密码为空的情况	字段类型不匹配错误	将测试用例中的 <code>userId</code> 改为正确的 <code>UUID</code> 格式
	002	输入用户名为空的情况		
UT_TC_001_001_002	001	输入密码和用户名不匹配的情况	字段类型不匹配错误	将测试用例中的 <code>userId</code> 改为正确的 <code>UUID</code> 格式
UT_TC_001_001_003	001	输入用户名不存在的情况	字段类型不匹配错误	将测试用例中的 <code>userId</code> 改为正确的 <code>UUID</code> 格式
UT_TC_001_001_004	001	输入参数合法的情况		
UT_TC_001_001_005	001	数据库连接失败的情况	异常消息不匹配，测试模拟数据库异常，期望返回具体错误消息，但实际返回固定提示 <code>"服务器内部错误"</code>	修改测试断言，匹配实际返回的固定消息（因为考虑到生产代码可能对异常消息进行了统一封装）
refresh_token 方法				
UT_TC_001_002_001	001	输入合法的 <code>current_token</code> 和 <code>refresh_token</code>	在测试中，同步函数 <code>is_token_blacklisted</code> 和 <code>invalidate_token</code> 被错误地使用了 <code>AsyncMock</code> 进	将 <code>test_user_service.py</code> 中对 <code>is_token_blacklisted</code> 和 <code>invalidate_token</code> 的 mock 从 <code>AsyncMock</code> 修改为 <code>MagicMock</code>

			行 mock。当异步函数 <code>refresh_token</code> 调用这些同步 mock 对象时，行为不符合预期	
UT_TC_001_002_002	001	<code>refresh_token</code> 无法解码		
UT_TC_001_002_003	001	<code>refresh_token</code> 无 <code>sub</code> 字段		
UT_TC_001_002_004	001	<code>current_token</code> 已在黑名单中		
UT_TC_001_002_005	001	新 <code>token</code> 生成失败的情况	<code>refresh_token</code> 函数最初没有捕获由 <code>create_access_token</code> mock 抛出的通用 <code>Exception</code> ，并将其转换为 <code>HTTPException</code>	修改 <code>refresh_token</code> 函数，在其 <code>try...except</code> 块中添加了对通用 <code>Exception</code> 的捕获，并将其转换为 <code>HTTPException(status_code=500, ...)</code>

测试报告

0 Failed, 11 Passed, 0 Skipped, 0 Expected failures, 0 Unexpected passes, 0 Errors, 0 Reruns

Show all details / Hide all details

Result	Test	Duration	Links
Passed	unittests/test_user_service.py::test_get_token_empty_password_UT_TC_001_001_001_001	32 ms	
Passed	unittests/test_user_service.py::test_get_token_empty_username_UT_TC_001_001_001_002	3 ms	
Passed	unittests/test_user_service.py::test_get_token_incorrect_password_UT_TC_001_001_002	4 ms	
Passed	unittests/test_user_service.py::test_get_token_user_not_found_UT_TC_001_001_003	3 ms	
Passed	unittests/test_user_service.py::test_get_token_success_UT_TC_001_001_004	4 ms	
Passed	unittests/test_user_service.py::test_get_token_database_failure_UT_TC_001_001_005	5 ms	
Passed	unittests/test_user_service.py::test_refresh_token_success_UT_TC_001_002_001_001	4 ms	
Passed	unittests/test_user_service.py::test_refresh_token_invalid_refresh_token_decoding_UT_TC_001_002_002_001	3 ms	
Passed	unittests/test_user_service.py::test_refresh_token_no_sub_in_refresh_token_UT_TC_001_002_003_001	3 ms	
Passed	unittests/test_user_service.py::test_refresh_token_current_token_blacklisted_UT_TC_001_002_004_001	3 ms	
Passed	unittests/test_user_service.py::test_refresh_token_new_token_creation_failure_UT_TC_001_002_005_001	4 ms	

13.2.2 地块管理模块

测试用例表

测试项编号	测试用例描述	bug	bug解决情况
-------	--------	-----	---------

	用例 序号			
get_all_plots 方法				
UT_TC_002_001_001	001	用户无地块的情况		
UT_TC_002_001_002	001	用户拥有单个地块的情况		
UT_TC_002_001_003	001	用户拥有多个地块的情况		
UT_TC_002_001_004	001	地块包含无效图片URL的情况		
UT_TC_002_001_005	001	用户认证失败的情况		
UT_TC_002_001_006	001	数据库查询失败的情况		
add_plot 方法				
UT_TC_002_002_001	001	成功创建地块的情况		
UT_TC_002_002_002	001	植物名称不存在的情况	DoesNotExist异常处理不匹配，测试中异常参数初始化失败，实际返回的固定错误提示为"无效的植物ID格式"	修改测试断言，将DoesNotExist异常修改为ValueError异常（因为源代码中可能对异常进行了统一封装
UT_TC_002_002_003	001	输入地块名称为空的情况		
	002	输入植物名称为空的情况		
UT_TC_002_002_004	001	用户认证失败的情况		
UT_TC_002_002_005	001	数据库操作失败的情况		
get_plot_detail 方法				
UT_TC_002_003_001	001	成功获取地块详情的情况	Pydantic模型验证错误：	已修复：

			1. logs.0.diseaseName 字段期望string类型，但传入了None 2. logs.0.imagesURL 字段期望string类型，但传入了list类型	1. 将mock_logs中diseaseName从None改为空字符串" 2. 将mock_logs中imagesURL从[]改为空字符串"
UT_TC_002_003_002	001	地块ID格式无效的情况		
UT_TC_002_003_003	001	地块ID不属于该用户	异常处理逻辑问题： 404异常被外层异常处理器包装成500错误，导致测试期望状态码不匹配	已修复： 在 service/plot.py中添加了对HTTPException的直接重抛，避免被通用异常处理器包装
	002	地块ID不存在	同上： 404异常被包装为500错误	已修复： 同上述解决方案
UT_TC_002_003_004	001	地块图片URL无效的情况		
UT_TC_002_003_005	001	用户认证失败的情况		
UT_TC_002_003_006	001	数据库查询失败的情况		
UT_TC_002_003_007	001	获取地块日志失败的情况		

Summary

20 tests took 67 ms.

(Un)check the boxes to filter the results.

0 Failed

20 Passed

0 Skipped

0 Expected failures

0 Unexpected passes

0 Errors

0 Reruns

Result

Test

Duration

Links

Show all details

Hide all details

13.2.3 智能检测模块

测试用例表

测试项编号	用例序号	测试用例描述	bug	bug解决情况
detect 方法				

UT_TC_003_001_001	001	正确识别 Grape 的各类病害结果		通过
UT_TC_003_001_002	001	正确识别 Potato 的各类病害结果		通过
UT_TC_003_001_003	001	model_type 非法，抛出 ValueError		通过
do_detect 方法				
UT_TC_003_002_001	001	所有输入有效		通过
UT_TC_003_002_002	001	传入文件格式错误		通过
UT_TC_003_002_003	001	地块不存在或无权访问的情况		通过
UT_TC_003_002_004	001	用户认证失败的情况		通过


Summary

7 tests took 414 ms.

(Un)check the boxes to filter the results.

☒ 0 Failed, ☒ 7 Passed, ☐ 0 Skipped, ☒ 0 Expected failures, ☒ 0 Unexpected passes, ☒ 0 Errors, ☒ 0 Reruns

Show all details / Hide all details

Result 	Test	Duration	Links
Passed	softtest/test.py::TestDoDetect::test_UT_TC_003_001_001_grape_disease_detection	40 ms	
Passed	softtest/test.py::TestDoDetect::test_UT_TC_003_001_002_potato_disease_detection	89 ms	
Passed	softtest/test.py::TestDoDetect::test_UT_TC_003_001_003_invalid_model_type	31 ms	
Passed	softtest/test.py::TestAPI::test_UT_TC_003_002_001_valid_inputs	69 ms	
Passed	softtest/test.py::TestAPI::test_UT_TC_003_002_002_invalid_file_format	41 ms	
Passed	softtest/test.py::TestAPI::test_UT_TC_003_002_003_invalid_plot_access	74 ms	
Passed	softtest/test.py::TestAPI::test_UT_TC_003_002_004_authentication_failure	72 ms	

13.2.4 统计分析模块

测试用例表

测试项编号	用例序号	测试用例描述	bug	bug解决情况
analyze_plot_details 方法				
UT_TC_004_001_001	001	plotDetails 为 None 的情况。		
UT_TC_004_001_002	001	plotDetails 为空列表的情况。		
	001			

UT_TC_004_001_003		地块日志属性 logs 为空列表的情况		
UT_TC_004_001_004	001	日志 diseaseName 为“健康”的情况		
UT_TC_004_001_005	001	日志 diseaseName 为“病害A”且为今年时的情况		
UT_TC_004_001_006	001	日志 diseaseName 为 None 的情况	未抛出异常，diseaseName为None的一项被统计	更改业务逻辑，出现None即抛出异常
UT_TC_004_001_007	001	日志 timeStamp 格式错误的情况	未抛出异常，timeStamp有误的一项被忽略，其余内容正常统计	更改业务逻辑，出现timeStamp格式有误的即停止统计并抛出异常
UT_TC_004_001_008	001	多个地块和日志的情况		
UT_TC_004_001_009	001	日志中同时有今年和非今年的 timeStamp 的情况		
get_summary 方法				
UT_TC_004_002_001	001	用户余额数不是正数的情况		
UT_TC_004_002_002	001	用户余额数大于等于1，用户地块列表元素数为多个的情况		
UT_TC_004_002_003	001	用户地块列表元素数为1个的情况		
UT_TC_004_002_004	001	用户地块列表元素数为0个的情况		

Summary

13 tests took 56 ms.

(Un)check the boxes to filter the results.

2 Failed

11 Passed

0 Skipped

0 Expected failures

0 Unexpected passes

0 Errors

0 Reruns

Result

Test

Duration

Links

Show all details

Hide all details

14. 缺陷跟踪

采用PingCode平台进行缺陷追踪，以下是单元测试的缺陷追踪情况：

6月2日：单元测试完成后紧急修复了涉及用户登陆操作、最高优先级的 `get_token` 方法

#	编号	标题	状态	负责人	创建时间	优先级	
1	DEMO-79	 get_token	已修复	 Jose	今天 16:14	Highest	
2	DEMO-80	 refresh_token	处理中	 Jose	今天 16:15	Highest	
3	DEMO-81	 get_plot_detail	处理中	 Jose	今天 16:17	Normal	
4	DEMO-82	 analyze_plot_details	处理中	 Jose	今天 16:18	Higher	

6月5日：修复了设计用户登陆凭证刷新，同样最高优先级的 `refresh_token` 方法

#	编号	标题	状态	负责人	创建时间	优先级	
1	DEMO-79	 get_token	已修复	 Jose	6月2日 16:14	Highest	
2	DEMO-80	 refresh_token	已修复	 Jose	6月2日 16:15	Highest	
3	DEMO-81	 get_plot_detail	处理中	 Jose	6月2日 16:17	Normal	
4	DEMO-82	 analyze_plot_details	处理中	 Jose	6月2日 16:18	Higher	

6月9日：修复了中优先级的 `get_plot_detail` 与 `analyze_plot_detail` 接口

#	编号	标题	状态	负责人	创建时间	优先级	
1	DEMO-79	 get_token	已修复	 Jose	6月2日 16:14	Highest	
2	DEMO-80	 refresh_token	已修复	 Jose	6月2日 16:15	Highest	
3	DEMO-81	 get_plot_detail	已修复	 Jose	6月2日 16:17	Normal	
4	DEMO-82	 analyze_plot_details	已修复	 Jose	6月2日 16:18	Higher	