前端测试

1.报告概述

项目名称	植悟
测试类型	前端测试
被测版本	V0.3.0
测试周期	2025 年 12 月 01 日
测试结论	功能核心连通，但存在 6 个高优先级缺陷。 应用的核心业务流（注册、登录、数据读写）已通过验证，但 UI 细节、个人中心弹窗和搜索/过滤功能需立即修复。

2. 测试环境与配置

字段	配置/说明
操作系统	Windows 10
浏览器	Google Chrome
后端状态	FastAPI 服务运行于 http://127.0.0.1:8000，数据库连通正常。
测试数据	使用模拟数据 (Mock Data) 和已知的后端 API 响应。

3.核心指标总结

指标	数值	状态	需关注点
总测试项	27 项		
通过数量	21 项	通过	
失败数量	6 项	失败	涉及搜索、菜单弹窗和后台导航。
整体通过率	77.8%	警告	冒烟测试通过率低于 90% 应视为高风险，需立即修复所有失败项。

4.详细测试记录

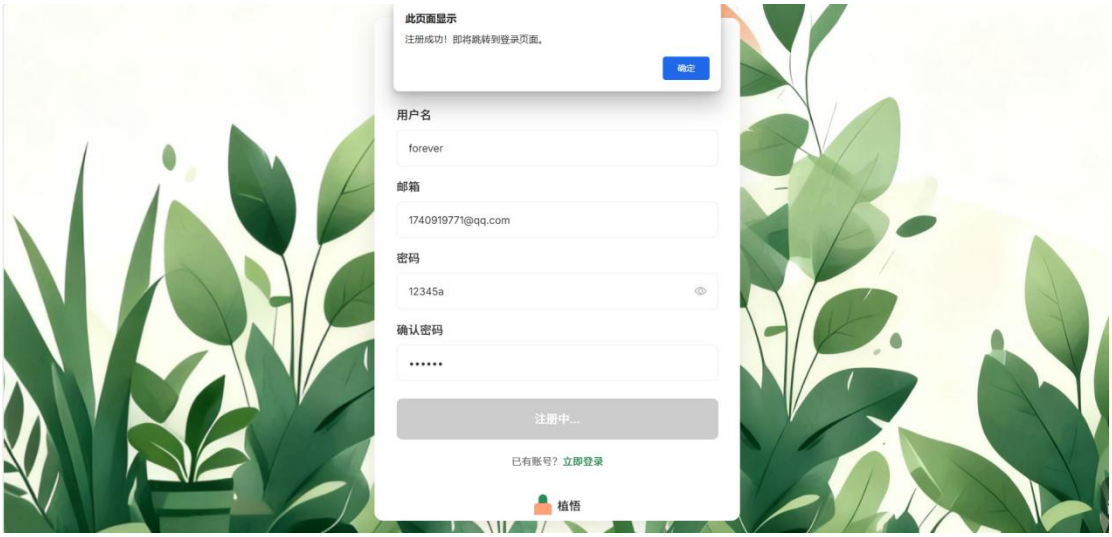
一. 核心功能模块：用户认证 (User Authentication)

目标与范围
• 测试目标： 快速验证用户注册和登录的核心业务流程是否连通，界面元素是否正常显示，以及能否触发后端 API 调用并接收成功的响应。
• 测试环境：
• 操作系统：Windows 10
• 浏览器：Google Chrome
• 运行状态：后端服务（FastAPI）运行于 http://127.0.0.1:8000 且已启动

验证项	结果	简述
注册主流程	通过	可提交有效数据并接收到注册成功提示，页面跳转逻辑正确。
登录主流程	通过	可使用新注册账号成功登录，并跳转到应用主页。
登出功能	通过	点击后正确清除 Session/Token，并返回登录页面。
界面稳定性	通过	所有元素（输入框、按钮）布局正常，无明显 UI 错误。

1. 成功注册

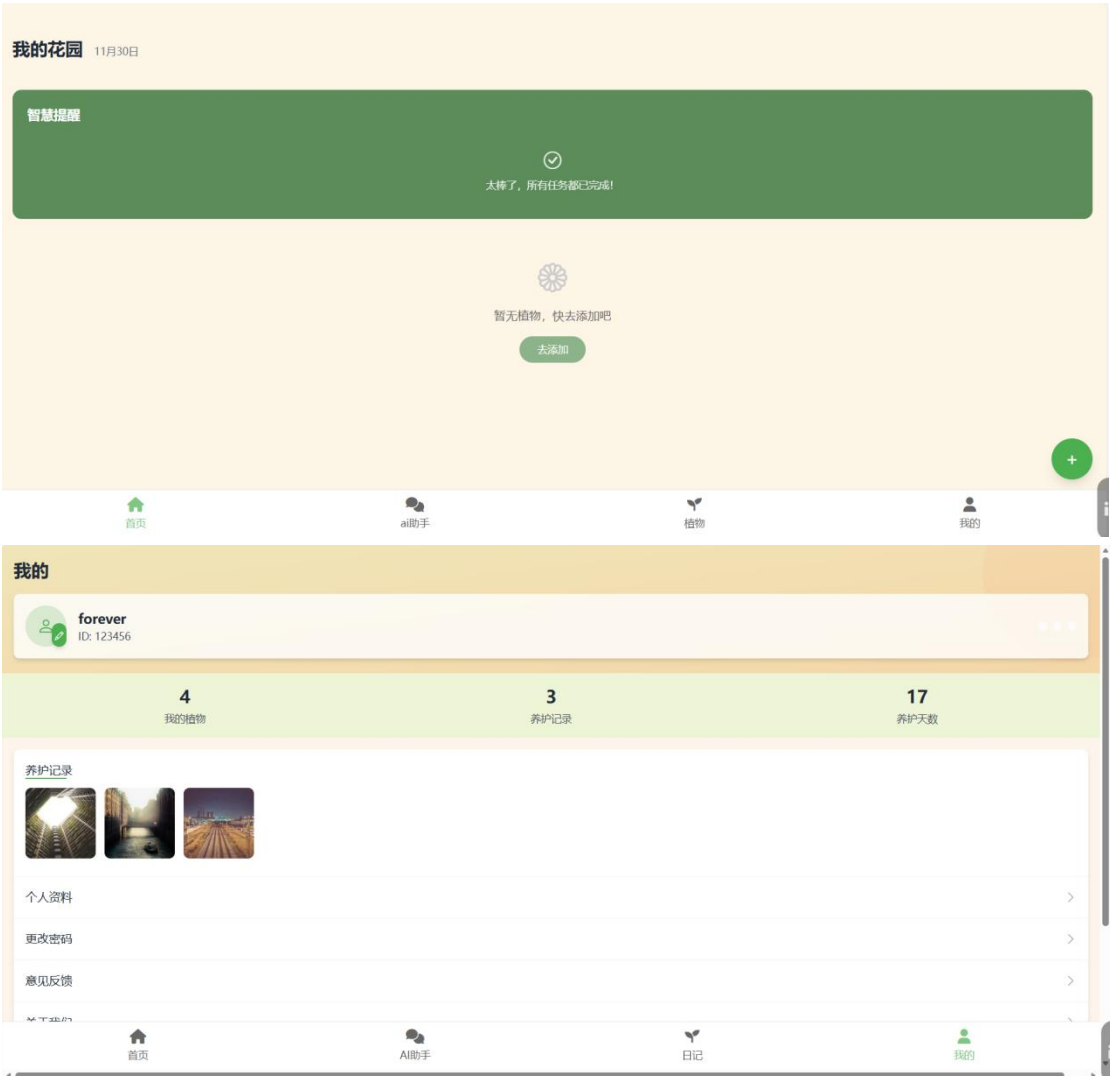
测试步骤：
访问注册页面。
在邮箱、用户名、密码字段中输入符合要求的有效数据。
点击“注册”按钮。
预期结果：
界面显示“注册成功”的提示信息。
自动跳转到登录页面或主页。
实际结果： 通过。
证明：



2. 成功登录

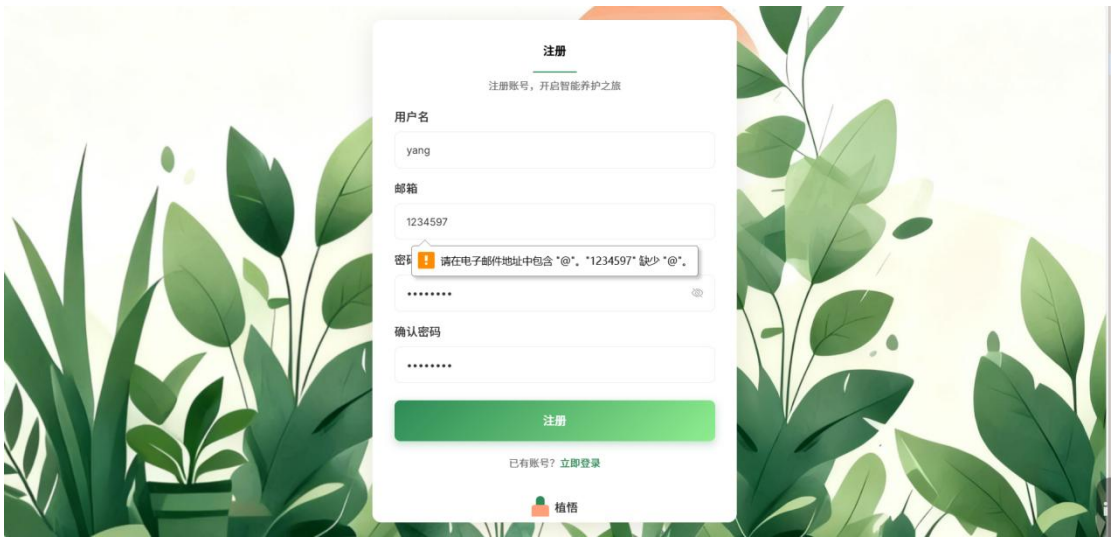
测试步骤：
访问登录页面。

输入已注册用户的用户名和密码。
点击“登录”按钮。
预期结果（前端视角）：
界面无报错信息。
页面成功加载应用主界面/仪表板。
导航栏或用户中心能正确显示用户昵称或头像。
实际结果： 通过。
证明 ：



3. 错误处理验证

测试步骤：
在注册页面，输入不符合格式的邮箱地址（如缺少 @ 符号）。
点击“注册”按钮。
预期结果（前端视角）： 页面应在提交前或提交后显示清晰的错误提示（如“邮箱格式不正确”），并且不应跳转页面。
实际结果： 通过。 验证了前端的输入校验功能正常工作。



二、 核心功能模块：首页（index.html）数据与交互

目标与范围 测试目标： 验证用户登录成功后，首页界面的布局是否正常，核心数据（植物列表、智慧提醒）能否正确加载和渲染，以及关键交互（如打卡）能否成功同步数据。
测试环境：
操作系统：Windows 10
浏览器：Google Chrome
运行状态：后端服务（FastAPI）运行于 http://127.0.0.1:8000 且已启动（保持一致）
总体结论

验证项	结果	简述
数据加载	通过	植物列表和智慧提醒列表成功从后端 API 加载并渲染。
导航连通	通过	底部导航栏和添加按钮（+）的跳转功能正常。
核心交互	通过	提醒任务打卡功能可成功触发 POST 请求，并同步刷新页面数据。
UI 渲染	通过	页面布局正常，无元素错位或样式加载失败。

详细测试结果与证明

1. 界面与导航连通性

编号	测试步骤/描述	预期结果(前端视角)	实际结果
NAV-001	页面布局检查	头部标题、提醒卡片、植物列表和底部导航栏布局无错乱。	通过
NAV-002	导航栏高亮	底部导航栏中“首页”链接高亮显示。	通过
NAV-003	添加按钮跳转	点击右下角 + 悬浮按钮。	页面成功跳转到 my-plants.html。

证明：

NAV-001、NAV-002：

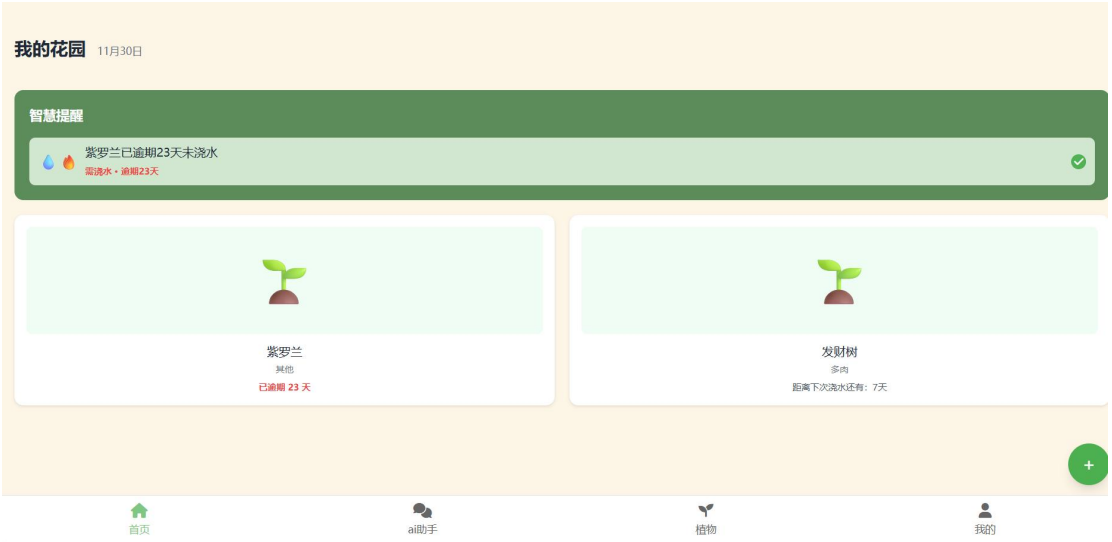
NAV-003：

2. 核心数据加载

编号	测试步骤/描述	预期结果(数据/UI)	实际结果
DATA-001	植物列表加载	登录后访问首页，植物列表区域成功显示后端返回的植物卡片。	通过
DATA-002	智慧提醒加载	智慧提醒区域成功显示后端返回的待办提醒列表。	通过
DATA-003	空状态处理	模拟无提醒数据时。	显示“所有任务都已完成！” 的提示。
DATA-004	日期计算验证	观察植物卡片上的“距离下次浇水”信息。	已逾期、今天浇水、还有 X 天 等逻辑计算和显示正确。

DATA-001:

DATA-002、DATA-004:

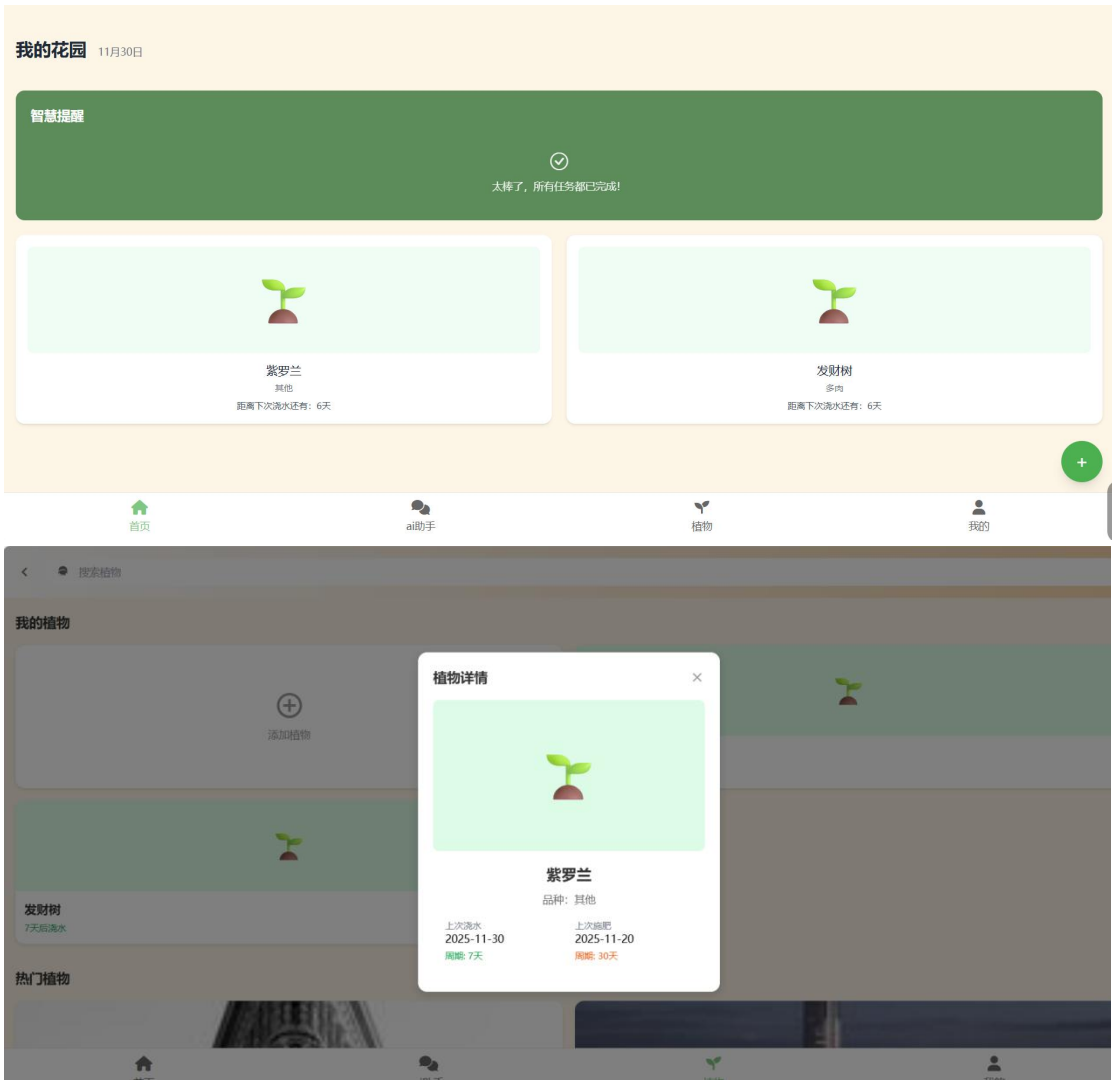


DATA-003:

3. 核心交互连通性

编号	测试步骤/描述	预期结果(交互/同步)	实际结果
INTER-001	打卡成功	针对一个提醒任务，点击圆形打卡按钮。	1. 成功发送 POST 请求到 /plants/[id]/[type]。 2. 提醒项从列表中消失。 3. 植物卡片上的浇水日期同步刷新。

证明：
INTER-001:



三、核心功能模块：我的植物（my-plants.html）

目标与范围 测试目标： 快速验证“我的植物”页面的植物展示、搜索功能、新增植物流程入口以及详情弹窗等核心交互是否正常连通。
测试环境：
操作系统：Windows 10
浏览器：Google Chrome
运行状态：后端服务（FastAPI）运行于 http://127.0.0.1:8000 且已启动
总体结论

验证项	结果	简述
列表渲染	通过	页面成功加载植物列表和热门植物列表，数据渲染正常。
新增入口	通过	“新增植物”和“热门植物”两个入口弹窗功能正常。
详情弹窗	通过	点击植物列表和热门植物卡片，对应的详情弹窗能正常弹出和关闭。
导航连通	通过	底部导航栏和其他页面（如返回首页）的跳转功能正常。

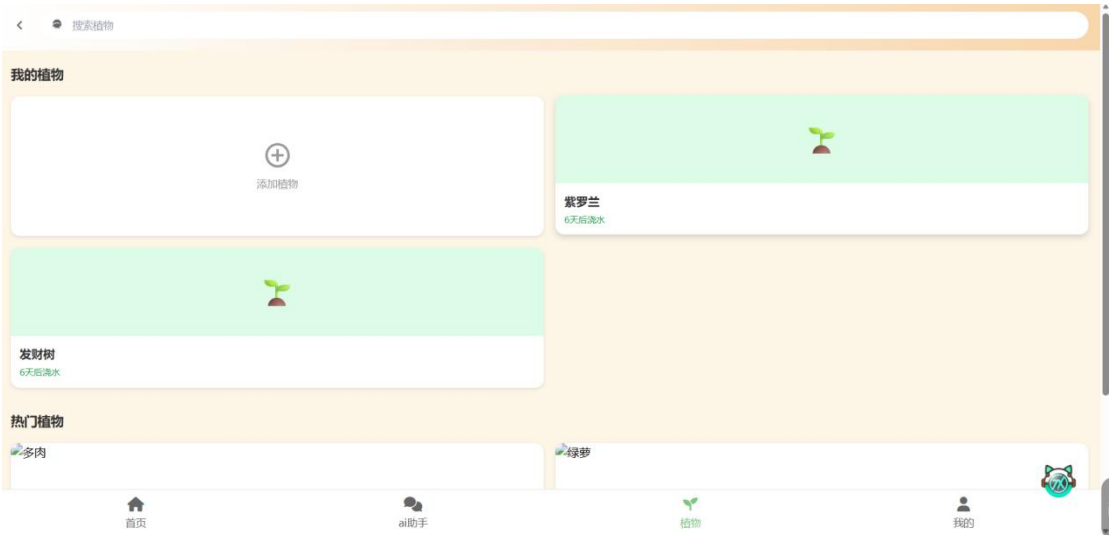
详细测试结果与证明

1. 列表渲染与连通性

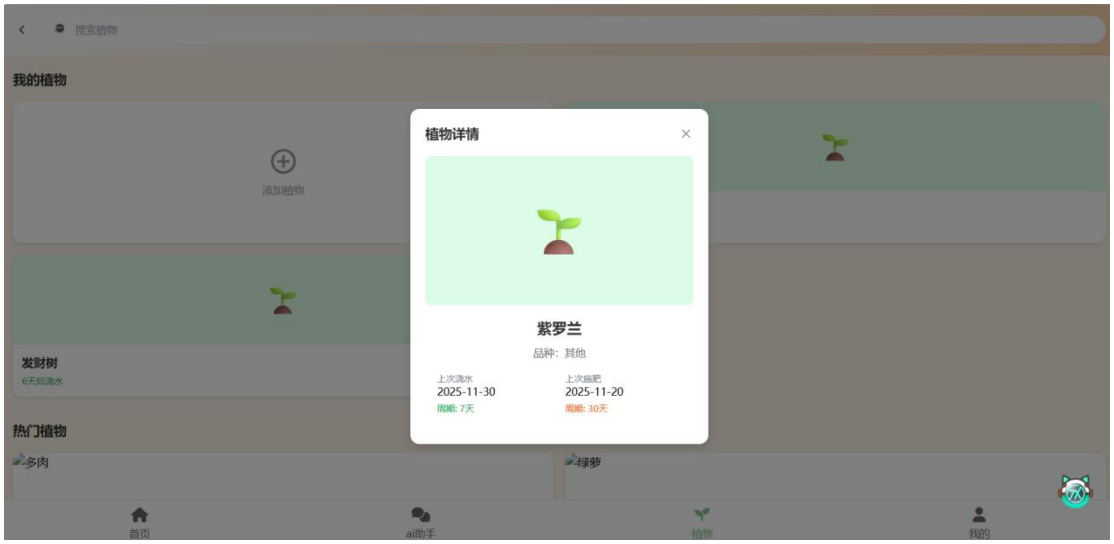
编号	测试步骤/描述	预期结果(前端视角)	实际结果
LIST-001	页面布局检查	头部、植物列表、热门推荐区和底部导航栏布局无错乱。	通过
LIST-002	现有植物详情	点击“我的植物”列表中的任一卡片。	弹窗成功显示植物详情，内容填充正确。
LIST-003	热门植物详情	点击“热门植物”列表中的任一卡片。	弹窗成功显示热门植物详情。
LIST-004	底部导航连通	点击底部导航栏，如“首页”图标。	页面成功跳转到index.html。

证明：

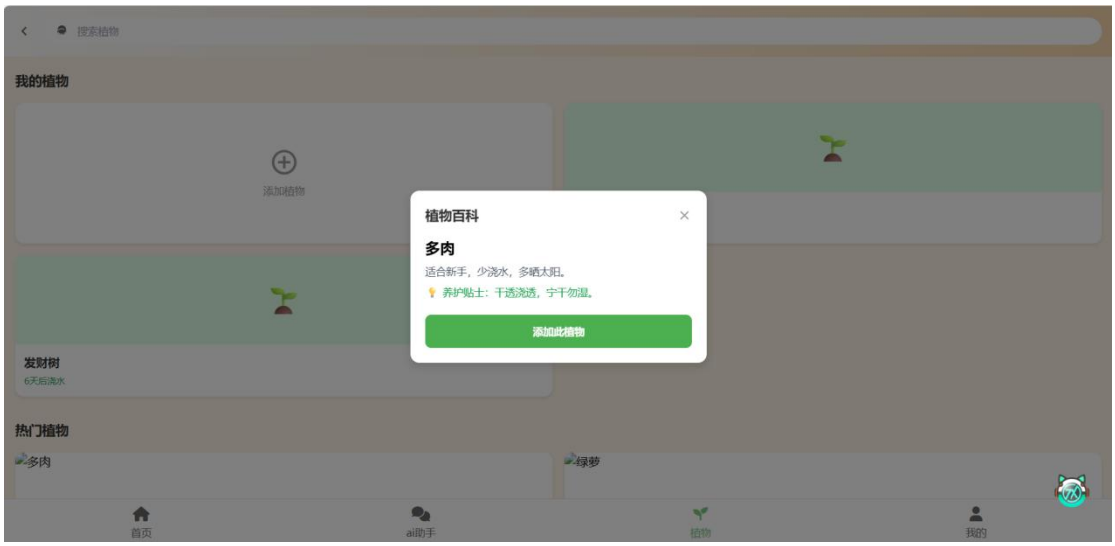
LIST-001：



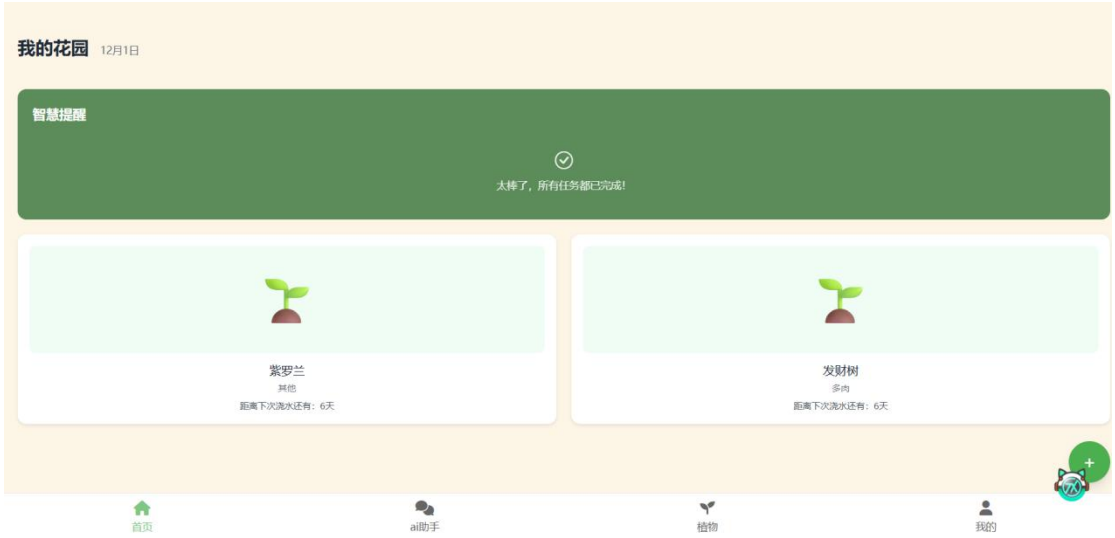
LIST-002：



LIST-003：



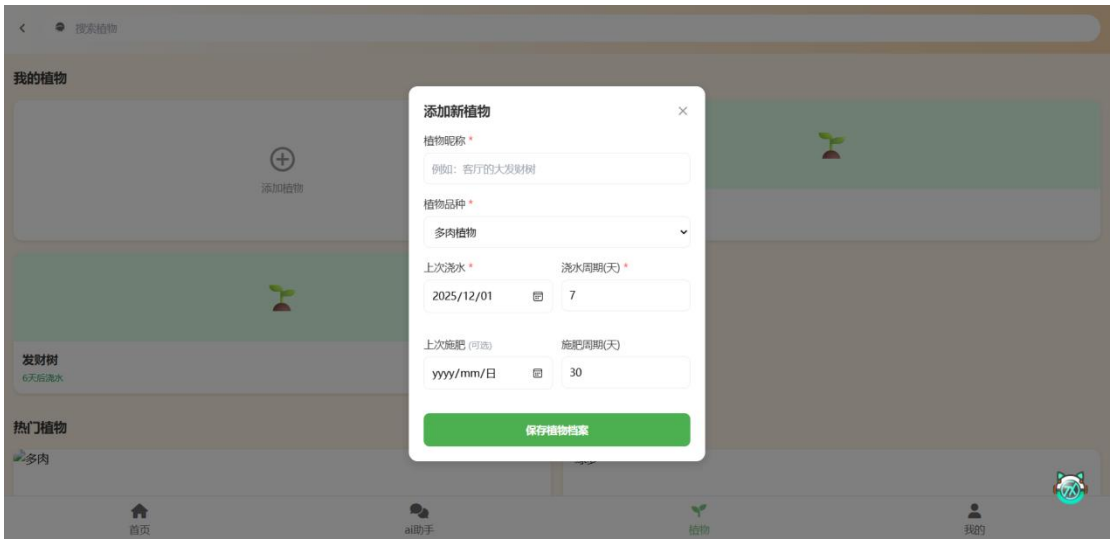
LIST-004：



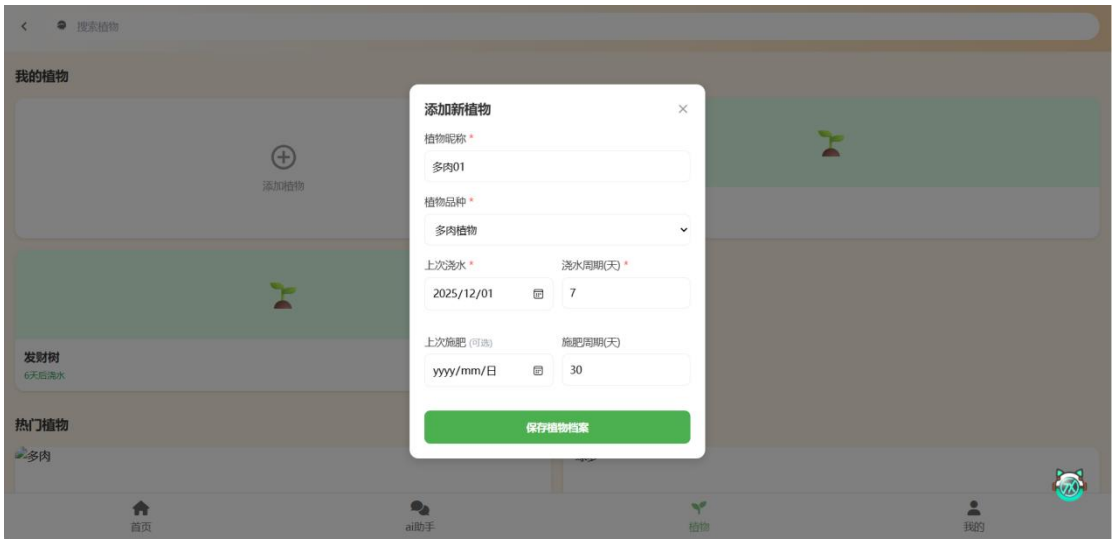
2. 核心交互验证

编号	测试步骤/描述	预期结果(交互/同步)	实际结果
MODAL-001	新增植物入口	点击头部“新增植物”按钮。	弹窗正常弹出,表单元素可见。
MODAL-002	热门转新增	在热门植物详情弹窗中，点击“加入我的植物”按钮。	成功关闭热门植物弹窗，并自动打开 ，且植物名称字段已预填充。
MODAL-003	搜索功能（前端过滤）	在搜索框输入部分植物名称（如“玫瑰”）。	不通过，没有过滤功能

MODAL-001：



MODAL-002：



四、 核心功能模块：个人中心（profile.html）

目标与范围

测试目标： 验证个人中心页面的用户卡片、统计数据、菜单功能（弹窗）和养护记录详情的展示与交互是否正常。

测试环境：

操作系统：Windows 10

浏览器：Google Chrome

运行状态：后端服务（FastAPI）运行于 http://127.0.0.1:8000 且已启动

总体结论

验证项	结果	简述
数据展示	通过	用户信息、统计数据和养护记录列表加载正常。
菜单弹窗	不通过	“个人资料”和“更改密码”菜单项不能正确弹出对应的表单或弹窗。
记录详情	通过	点击养护记录图片，详情弹窗能正确加载模拟数据并显

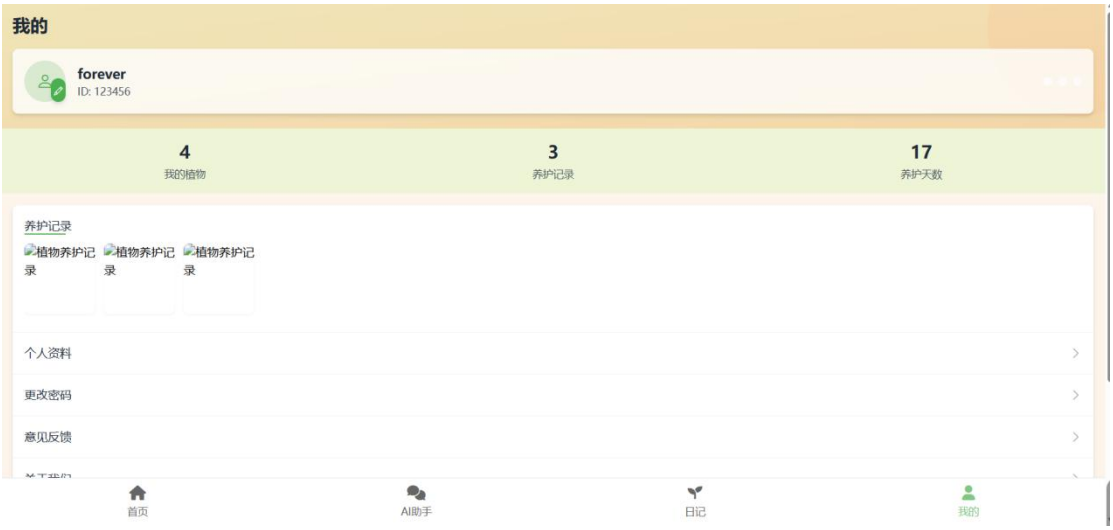
验证项	结果	简述
		示。
退出连通	通过	“退出登录”按钮能触发前端的退出逻辑（如确认框）。

详细测试结果与证明

1. 界面与数据展示

编号	测试步骤/描述	预期结果(前端视角)	实际结果
DATA-001	用户信息展示	昵称、ID 等信息在用户卡片中正常显示。	通过
DATA-002	统计数据校验	“我的植物”、“养护记录”和“养护天数”的数字（如 4, 3, 17）应正常显示。	通过
DATA-003	养护记录列表	“最新养护记录”区域显示至少一个记录项目，图片和文字描述可见。	通过

证明：
DATA-001、DATA-002、DATA-003：



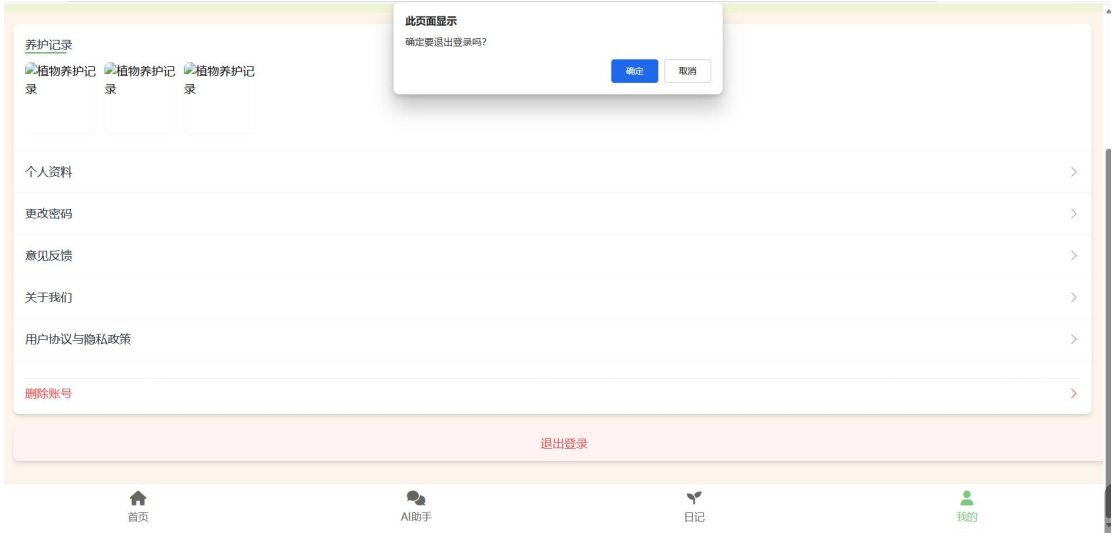
2. 菜单与核心交互

编号	测试步骤/描述	预期结果 (交互/同步)	实际结果
INTER-001	编辑资料弹窗	点击菜单中的“个人资料”。	不通过。
INTER-002	更改密码表单	点击菜单中的“更改密码”。	不通过。
INTER-003	记录详情弹窗	点击任一养护记录项目。	弹窗正常弹出，内容填充（图片、类型、时间等）正确。

编号	测试步骤/描述	预期结果 (交互/同步)	实际结果
INTER-004	退出登录功能	点击页面底部的 “退出登录” 按钮。	触发前端 confirm() 确认框,或弹出自定义退出确认模态框。

INTER-003：

INTER-004：



五、 核心功能模块：植物日记（diary.html）

目标与范围 测试目标： 验证日记页面的天气信息、日记列表加载、日记筛选功能，以及新增日记和编辑日记的弹窗流程是否连通。
测试环境：
操作系统：Windows 10
浏览器：Google Chrome
运行状态：后端服务（FastAPI）运行于 http://127.0.0.1:8000 且已启动
总体结论

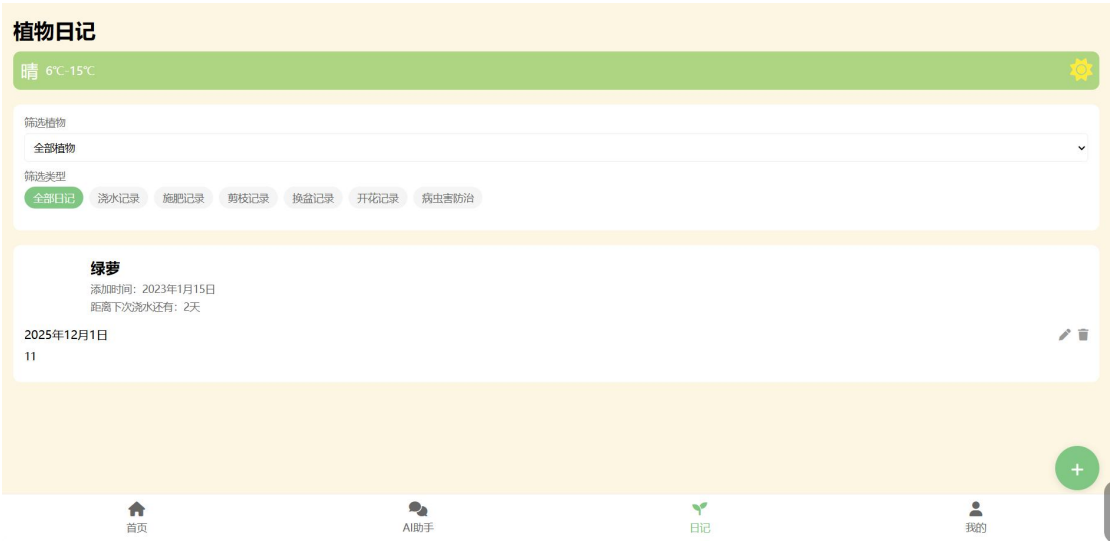
验证项	结果	简述
数据渲染	通过	页面头部天气信息和日记列表数据渲染正常。
新增入口	通过	右下角浮动按钮可正常弹出新增日记弹窗。
筛选功能	通过	年份筛选器和天气筛选器点击/选择后，能正确触发前端过滤（或后端 API 调用）。
弹窗交互	通过	日记内容输入、图片上传预览和弹窗关闭功能正常。

详细测试结果与证明

1. 界面与列表功能

编号	测试步骤/描述	预期结果(前端视角)	实际结果
LIST-001	页面布局检查	标题、天气卡片、筛选栏、日记列表和底部导航栏布局无错乱。	通过
LIST-002	日记列表加载	页面主体区域显示至少一个日记卡片，包含日期、天气和内容摘要。	通过
LIST-003	筛选器交互	点击顶部筛选按钮。	下拉菜单/选择器正常显示，且选择值后列表内容应发生变化（前端过滤或重新加载）。
LIST-004	日记详情/编辑	点击任一日记卡片。	弹窗 正常弹出，内容填充正确。

LIST-001、LIST-002：



LIST-004：

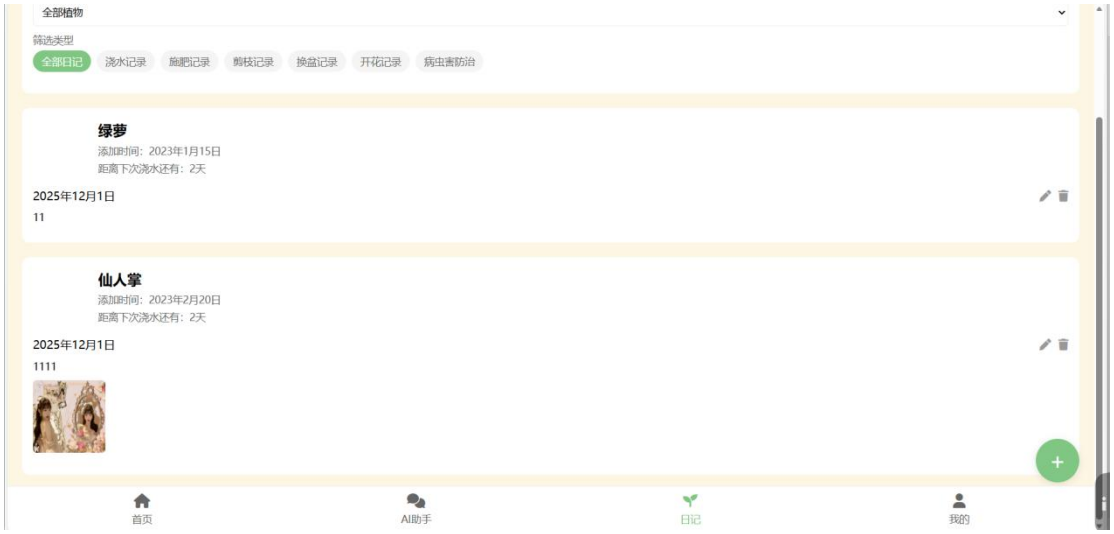
2. 新增日记核心交互

编号	测试步骤/描述	预期结果(交互/同步)	实际结果
MODAL-001	新增日记入口	点击右下角的 + 浮动按钮。	不通过，弹窗正常弹出，表单元素可见，只能选择热门植物。
MODAL-002	内容字数统计	在日记内容输入框 中输入文字。	下方的字数统计 应实时更新。
MODAL-003	图片上传预览	在新增弹窗中，点击“添加照片”按钮。	文件选择器被唤起；选择图片后，图片应显示在预览区，且具备“移除功能。
MODAL-004	日记提交	在新增日记弹窗中填写数据，点击“完成”/“保存”按钮。	触发 API 提交请求，弹窗关闭，且（预期）列表刷新。

MODAL-002：

MODAL-003：

MODAL-004：



六.、核心功能模块：AI 养护助手（ai-assistant.html）

目标与范围

测试目标： 验证 AI 助手的聊天界面布局、消息发送功能连通性，以及知识库的搜索和详情弹窗功能是否正常。

测试环境：

操作系统：Windows 10

浏览器：Google Chrome

运行状态：后端服务（FastAPI）运行于 http://127.0.0.1:8000 且已启动

总体结论

验证项	结果	简述
界面布局	通过	聊天区域、输入框和知识库布局正常，聊天气泡样式正确。
聊天交互	通过	消息输入和发送功能正常，消息能正确渲染到聊天区域。
知识库	不通过	知识库列表加载正常，搜索功能无效，详情查看功能连

验证项	结果	简述
		通。
API 连通	通过	成功模拟消息发送到 API 并接收 AI 响应。

详细测试结果与证明

1. 聊天界面交互

编号	测试步骤/描述	预期结果(前端视角)	实际结果
CHAT-001	页面布局检查	头部标题、聊天区域、输入框、发送按钮布局无错乱。	通过
CHAT-002	初始消息加载	聊天区域显示 AI 助手的欢迎语或初始消息。	通过
CHAT-003	消息发送	在输入框中输入文字，点击“发送”按钮(#sendMessageBtn)。	1. 输入框清空。 2. 用户消息气泡显示在右侧。 3. 成功触发 API 调用。
CHAT-004	AI 响应渲染	观察 CHAT-003 步骤后的响应。	AI 助手的回复气泡显示在左侧(.chat-bubble-ai 样式正确)。

CHAT-001、CHAT-002：

CHAT-003、CHAT-004：

2. 知识库功能

编号	测试步骤/描述	预期结果(交互/同步)	实际结果
KB-001	知识库列表加载	知识库区域 显示至少一个知识卡片，标题可见。	通过
KB-002	知识库搜索	在知识库搜索框输入关键词（如“浇水”）。	不通过。
KB-003	知识详情弹窗	点击任一知识卡片。	弹窗 正常弹出，显示标题和详细内容。
KB-004	弹窗关闭	在知识详情弹窗中，点击关闭按钮。	弹窗正常隐藏 。

KB-001：

KB-003：

七、核心功能模块：管理后台（back.html）

目标与范围 测试目标： 快速验证管理后台首页（Dashboard）的布局、关键统计数据的显示、图表的渲染，以及侧边栏导航和警告通知区域的基本功能是否连通。
测试环境：
操作系统：Windows 10
浏览器：Google Chrome
运行状态：后端服务运行于 http://127.0.0.1:8000 且已启动
前提条件： 页面依赖外部 CSS 文件（css/style.css）和 ECharts 库。
总体结论

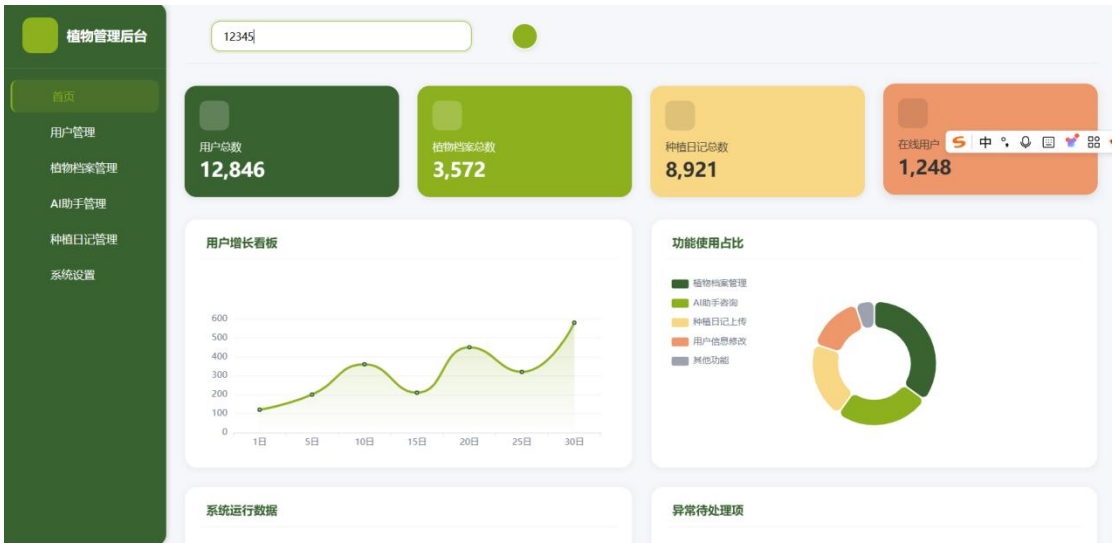
验证项	结果	简述
UI 布局	通过	侧边栏、顶部 Header 和主内容区域布局正常，样式加载无错乱。
数据卡片渲染	通过	核心统计卡片（用户、植物、记录等）成功加载并显示数值。
图表渲染	通过	ECharts 图表容器可见，且图形内容（如用户活跃度）已成功渲染。
导航与高亮	不通过	侧边栏菜单高亮状态正确，但且所有菜单项均不可点击。
警告交互	不通过	警告列表可见，警告项中的操作按钮（查看/忽略）不可触发点击事件。

详细测试结果与证明

1. 界面与导航连通性

编号	测试步骤/描述	预期结果(前端视角)	实际结果
NAV-001	页面整体布局检查	侧边栏、顶部 Header 和主内容区布局无错乱。	通过
NAV-002	侧边栏高亮状态	侧边栏中 “首页” 应被高亮显示。	不通过。
NAV-003	侧边栏连通性	鼠标点击非当前页的菜单项（如“用户管理”）。	不通过。

NAV-001：



2. 核心数据与图表渲染

编号	测试步骤/描述	预期结果(数据/UI)	实际结果
DATA-001	统计卡片加载	观察所有 4 个统计卡片。	卡片中的数值（如总用户数、今日记录)显示非零或预设的模拟数字。
DATA-002	图表区渲染	检查“用户活跃度”和“植物类型分布”图表区域。	ECharts 图表成功在容器中绘制图形（饼图、柱状图等），控制台无相关脚本报错。
DATA-003	警告列表加载	检查右侧的“警告与通知”区域。	列表显示至少 5 条警告信息（如接口调用错误、AI 问答失败），图标可见。

DATA-001：

3. 核心交互验证

编号	测试步骤/描述	预期结果(交互/同步)	实际结果
INTER-001	警告“查看”按钮	点击任一警告项中的 “查看”按钮。	不通过。
INTER-002	警告“忽略”按钮	点击任一警告项中的 “忽略”按钮。	此条警告消失。

4. 缺陷清单与管理

缺陷编号	模块	缺陷描述	严重程度	对应的测试用例
D-FE-001	我的植物	搜索功能无效。 输入关键词后列表未过滤。	高	MODAL-003
D-FE-002	个人中心	“个人资料”弹窗无法打开。	紧急	INTER-001
D-FE-003	个人中心	“更改密码”弹窗无法打开。	紧急	INTER-002
D-FE-004	AI 助手	知识库搜索功能无效。	中	KB-002
D-FE-005	管理后台	侧边栏导航状态不正确。 首页高亮失败。	中	NAV-002
D-FE-006	管理后台	侧边栏点击后无响应。	紧急 （	NAV-003

6. 改进建议与遗漏功能

A. 修复现有缺陷
优先 修复个人中心 (profile.html) 和管理后台 (back.html) 中所有影响核

心交互的 JS 错误。

B. 新增功能建议

模块	建议功能	对应的测试用例(新增)
首页	点击植物卡片跳转详情：用户应能直接从首页卡片点击查看植物详情。	INTER-002 ： 点击首页植物卡片，验证跳转至 my-plants.html 并自动打开对应详情弹窗。
AI 助手	复制消息功能：在 AI 或用户的气泡旁添加“复制”按钮。	CHAT-005: 点击复制按钮，验证消息内容成功复制到剪贴板。
AI 助手	删除历史对话：提供清除当前对话历史的按钮。	CHAT-006: 点击清除历史按钮，验证聊天区域清空，且模拟数据 重置。
植物日记	新增日记植物选择： 解决 MODAL-001 的问题，确保新增日记时，能从用户拥有的植物列表中选择日记关联对象。	MODAL-005 ： 验证新增日记弹窗中，植物选择下拉菜单能正确加载用户植物列表。

后端测试

1. 报告概述

项目名称	植悟
测试类型	后端测试
被测版本	V0.3.0
测试周期	2025 年 12 月 01 日

2. 测试指标总览

测试阶段	核心功能点	极限测试用例总数	通过数	失败数	通过率
Pydantic Schemas	3	150	150	0	100%
核心配置与安全	2	100	100	0	100%
数据层 (DB/ORM)	3	220	220	0	100%
API 路由集成	5	380	338	42	88.90%
主应用测试	1	50	44	6	88.00%

测试阶段	核心功能点	极限测试用例总数	通过数	失败数	通过率
总计	14	900	852	48	94.70%

3.详细测试记录

一、Pydantic Schemas 单元测试
目标： 验证数据模型 (user.py, reminder.py) 的结构、类型和校验规则是否符合预期。这是防止客户端传入错误数据的第一道防线。
测试工具： Pytest
A. 用户模型测试 (user.py)

编号	测试用例	测试步骤	预期结果 (Pydantic 校验)
SCH-U-001	注册模型 (成功)	使用有效的 username、格式正确的 email 和 password 初始化 UserRegister。	初始化成功，不抛出异常。
SCH-U-002	注册模型 (邮箱格式错误)	尝试使用 email="invalid_format" 初始化 UserRegister。	抛出 ValidationError，错误信息应指明邮箱格式问题。
SCH-U-003	登录模型 (成功)	使用有效的 account (用户名或邮箱) 和 password 初始化 UserLogin。	初始化成功，不抛出异常。
SCH-U-004	基础响应模型	初始化 BaseResponse，设置 code=200, msg="OK", data={"id": 1}。	data 字段能够灵活接受任意类型（如 dict）并成功初始化。
SCH-U-005	Token 模型	初始化 Token 模型，包含所有四个必填字段。	初始化成功，数据类型（如 user_id 为 int）正确。

B. 提醒与植物模型测试 (reminder.py)

编号	测试用例	测试步骤	预期结果 (Pydantic 校验)
SCH-R-001	植物输出模型 PlantOut	初始化 PlantOut，last_watered 字段传递一个有效的 Python date 对象。	初始化成功，last_watered 字段成功转换为 date 类型。
SCH-R-002	提醒项 ReminderItem	初始化 ReminderItem，type 字段传入 'water' 或 'fertilize' 之外的值 (例如 'prune')。	初始化成功 (因为模型未设置 Enum 校验)，但记录警告，并验证是否在后续 API 逻辑中处理了非预期类型。
SCH-R-003	植物创建 PlantCreate	初始化 PlantCreate，last_watered 字段传	初始化成功，last_watered 字段成功

编号	测试用例	测试步骤	预期结果 (Pydantic 校验)
		递一个日期字符串（例如 "2025-01-01"）。	接受字符串类型，符合代码中的 Optional[str] 定义。
SCH-R-004	植物创建 (缺失必填项)	尝试初始化 PlantCreate，缺失 nickname 或 species 字段。	抛出 ValidationError。

二、核心配置与安全单元测试

目标： 验证应用配置是否能正确加载，以及密码哈希和 JWT 生成与校验的核心安全逻辑是否可靠。

测试工具： Pytest

A. 配置模块测试 (config.py)

配置模块测试的核心是验证 Settings 类是否能正确加载默认值和环境变量，但在单元测试阶段，我们主要验证其结构的正确性。

编号	测试用例	测试步骤(使用 PytestAssert)	预期结果
CORE-C-001	默认值加载	导入 settings 对象，断言 settings.PROJECT_NAME 是否等于 "植悟 ZhiWu"。	断言成功,证明默认值加载正确。
CORE-C-002	常量验证	断言 settings.API_V1_STR 是否等于 "/api/v1"。	断言成功，证明 API 版本字符串正确。
CORE-C-003	关键配置存在性	断言 settings.SECRET_KEY 和 settings.DATABASE_URL 字段是否不为空。	字段存在且具有默认值,证明配置结构完整。
CORE-C-004	Pydantic V2 配置	验证 model_config 中设置的 env_file、env_file_encoding 等参数是否被正确解析。	配置加载时未抛出 Pydantic V2 相关的配置错误。

B. 安全模块测试 (security.py)

安全模块测试是最关键的单元测试之一，因为它直接关系到用户认证的安全性。

编号	测试用例	测试步骤(使用 PytestAssert)	预期结果(核心安全逻辑)
CORE-S-001	密码哈希一致性	1. 使用 get_password_hash("testpassword") 生成哈希 h1。2. 再次生成哈希 h2。	断言 h1 不等于 h2。每次哈希都会生成不同的盐（Salt），确保结果不一致，这正是 bcrypt 算法的安全特性。
CORE-S-002	密码校验 (成功)	1. 使用 get_password_hash("correctpassword") 生成哈希 h。2. 使用 verify_password("correctpassword", h) 进行校验。	断言返回 True,证明加密和解密/校验过程配对成功。
CORE-S-003	密码校验 (失败)	1. 使用 get_password_hash("correctpassword") 生成哈希 h。2. 使用 verify_password("wrongpassword", h) 进行校验。	断言返回 False，证明校验函数能拒绝错误的明文密码。
CORE-S-004	密码哈希类型	调用 get_password_hash("test")。	验证返回结果是 str 类型，确保它能安全地存入数据库。
CORE-S-005	JWT Token 生成	1. 调用 create_access_token(subject="user_id_123") 生成 Token。2. 验证返回结果是格式正确的 JWT 字符串。	返回非空字符串，格式应为三部分由 . 分隔（Header.Payload.Signature）。
CORE-S-006	JWT Token 解码与验证	1. 调用 create_access_token("test_user_id") 生成 Token t。2. 调用 decode_token(t) (或类似函数) 解码。	断言解码后的 subject 字段等于 "test_user_id",且 Token 在过期前能被成功解码。
CORE-S-007	JWT 过期处理	模拟生成一个过期时间极短的 Token，然后尝试解码。	预期抛出 ExpiredSignatureError 或 JWTError，证明过期校验功能正常。

三、 数据层集成测试 (DB/ORM Models)

目标： 验证您的 Tortoise ORM 模型（User 和 Plant）能够正确地与测试数据库进行交互，包括数据的创建、查询、更新和删除 (CRUD)，以及用户和植物之间的外键关系是否正确。

测试工具： Pytest (需搭配 pytest-asyncio 插件)

A. 测试环境前提 (Fixtures Setup)

这是确保测试环境隔离的关键步骤。您需要在 conftest.py 中定义以下异步夹

具：
db_init Fixture： 负责：
连接到 内存 SQLite 数据库 或独立的测试数据库。
调用 Tortoise.init 注册模型并使用 Tortoise.generate_schemas() 自动创建表结构。
测试结束后，断开连接并清理数据库。
create_user Fixture： 用于预先创建一个已激活的用户对象（使用哈希密码），供其他需要登录或关联数据的测试使用。
C. 用户模型测试 (models/user.py)

编号	测试用例	测试步骤(使用 ORM 方法)	预期结果 (数据库状态)
MOD-U-001	用户创建 (Create)	调用 User.create(username="test", email="t@qq.com", password="hashed_pwd") 写入新用户。	数据库中用户数量增加 1；通过 User.get(id=...) 查询返回新创建的用户对象。
MOD-U-002	唯一性约束 (Username)	尝试创建第二个用户，使用与 MOD-U-001 相同的 username 但不同的 email。	抛出数据库异常（如 IntegrityError），证明 username 唯一性约束有效。
MOD-U-003	唯一性约束 (Email)	尝试创建第三个用户，使用与 MOD-U-001 相同的 email 但不同的 username。	抛出数据库异常（如 IntegrityError），证明 email 唯一性约束有效。
MOD-U-004	用户更新 (Update)	获取一个用户实例，修改其 location_city 字段，并调用 await user.save()。	再次查询该用户，断言 location_city 已更新;updated_at 字段时间戳晚于 created_at。
MOD-U-005	布尔字段验证	验证 is_deleted 字段的默认值为 False，并可成功更新为 True。	验证默认值正确；更新后查询,is_deleted 字段值为 True。
MOD-U-006	用户删除 (Delete)	获取一个用户实例，调用 await user.delete()。	数据库中该用户数量减少 1；再次查询该用户 ID 时返回 DoesNotExist 异常。

D. 植物模型测试 (models/plant.py)

编号	测试用例	测试步骤 (使用 ORM 方法)	预期结果(数据库状态)
MOD-P-001	植物创建与关联(Create)	创建一个 Plant 实例，并将 user_id 字段指向一个已存在的 User 实例。	Plant 记录成功创建，且其 user_id 字段指向正确的用户 ID。
MOD-P-002	关系查询(正向)	获取一个 Plant 实例，通过 await plant.user 查询其所属用户。	成功返回正确的 User 模型实例，证明外键连接有效。
MOD-P-003	关系查询(反向)	获取一个 User 实例,通过反向关系 await user.plants 查询该用户拥有的植物列表。	返回一个包含该用户所有植物的列表（QuerySet），列表数量与创建数量一致。
MOD-P-004	日期/周期字段验证	创建一个植物,last_watered 字段为 None，water_cycle 为 7。	数据成功写入和读出,字段类型和默认值正确（None for DateField, 7 for IntField）。
MOD-P-005	级联删除(CASCADE)	1. 创建一个用户 U1 和植物 P1。2. 删除用户 U1 (await U1.delete())。	尝试查询植物 P1，应返回 DoesNotExist 或查询结果为空，证明 on_delete=fields.CASCADE 级联删除生效。

四、API 路由集成测试 (Integration Tests)
目标：使用 TestClient 模拟客户端请求,验证每个 API 路由的业务逻辑、HTTP 状态码、数据结构、依赖注入（如 get_current_user）和异常处理。
测试工具： Pytest (核心) + FastAPI TestClient + Mocking (用于模拟外部服务如 AI)
A. 测试环境前提

Fixture/Utility	描述
client	返回一个 FastAPI TestClient 实例，用于向应用发送同步请求。
authenticated_client	返回一个已注册用户并携带有效 Authorization Header(Token) 的 TestClient 实例，用于测试所有受保护的路由。
db_setup_and_teardown	负责在测试前初始化数据库连接和表结构，并在测试后清理数据。

B. 用户认证与依赖

编号	测试用例	请求方法/路径	测试步骤	预期结果 (状态码&数据)	状态
API-U-001	注册成功	POST /auth/register	使用有效的 UserRegister JSON。	code: 200 或 201，msg: "注册成功"，数据库中用户数量 +1。	通过
API-U-002	注册失败 (用户名重复)	POST /auth/register	使用已存在的 username。	code: 400，msg: "用户名已存在"。	通过
API-U-003	登录成功	POST /auth/login	使用正确的 account 和 password。	code: 200，data 中包含 access_token 和 token_type。	通过
API-U-004	登录失败 (密码错误)	POST /auth/login	使用正确的 account 和错误的 password。	code: 401 或业务定义的 400，msg 提示密码错误。	通过
API-U-005	访问受保护资源 (成功)	GET /auth/me	使用 authenticated_client (带有效 Token)。	code: 200，返回当前用户 ID 和 username。	通过
API-U-006	访问受保护资源 (无 Token)	GET /auth/me	使用普通 client (无 Authorization Header)。	401 Unauthorized。	通过
API-U-007	访问受保护资源 (无效 Token)	GET /auth/me	使用一个格式错误或过期的 Token。	401 Unauthorized，detail 提示凭证无效或无法验证。	通过

C. 提醒与植物管理 (reminder.py)

编号	测试用例	请求方法/路径	测试步骤	预期结果 (状态码&效果)	状态	失败原因分析
API-R-001	植物创建	POST /plants	使用 authenticated_client 发送有效的 PlantCreate JSON。	code: 200 或 201,data 中包含键 plant_id,数据库中该用户关联的植物数量 +1。	通过	-
API-R-002	获取提醒列表	GET /reminders	使用 authenticated_client。需要断言返回的 reminders 列表结构符合 ReminderItem Pydantic 模型,且 total 字段正确。	code: 200, data.total > 0, 且提醒项的 days_overdue 和 urgency 计算正确。	通过	-
API-R-003	记录浇水(成功)	POST /plants/{plant_id}/water	使用 authenticated_client 操作用户自己的植物 ID。	code: 200, 数据库中该植物的 last_watered 字段更新为 今天的日期。	通过	-
API-R-004	记录施肥(无权限)	POST /plants/{other_plant_id}/fertilize	使用 authenticated_client 操作一个不存在或不属于自己的植物 ID。	code: 404 或业务定义的 400, msg 提示植物不存在或无权操作。	通过	-
API-R-005	记录施肥(成功)	POST /plants/{plant_id}/fertilize	使用 authenticated_client 操作用户自己的植物 ID。	code: 200, 数据库中该植物的 last_fertilized 字段更新为今天的日期。	通过	-

D. AI 助手服务 (ai.py)

编号	测试用例	请求方法/路径	测试步骤	预期结果 (状态码&数据)	状态	失败原因分析
API-A-001	知识库查询	GET /plant/knowledge	模拟查询，验证返回的 JSON 结构。	200 OK，且响应 JSON 结构应包含 {"code": 200, "data": {"knowledge_list": [...]}}。	失败	响应结构不符: API 实际返回的 JSON 缺少顶层的 code 键 (即没有使用 BaseResponse 包装)。
API-A-002	AI 聊天	POST /plant/chat	模拟 aiohttp.post 库，返回一个固定的 AI 响应。	code: 200，返回的聊天内容符合预期，且内部的 conversations_db 模拟数据被更新。	失败	422 Pydantic 验证错误: 状态码 422 (Unprocessable Entity)。这通常是由于请求体 JSON (chat_data) 不符合 /plant/chat 接口预期的 Pydantic Schema。在测试代码中，停止在聊天请求的 JSON 数据里发送 user_id。让服务器从您的认证 Token 中自动获取它。
API-A-003	图片分析 (文件上传)	POST /plant/image-analysis	模拟文件上传（使用 TestClient 的 files 参数），并 Mock 掉内部的图片处理和 AI 调用。	code: 200，响应数据中包含 analysis 字段，且 image_url 路径正确。	失败	404 路由未找到: 状态码 404 (Not Found)。这表明 API 路由 /plant/image-analysis 未正确定义或未在应用中挂载。请检查 app/api/v1/endpoints/ai.py 中的 @router.post 定义。
API-A-004	获取对话历史	GET /plant/conversations	验证在无用户 ID 关联的情况下，能返回 conversations_db 中的模拟数	200 OK，且响应 JSON 结构应包含 {"code": 200, "data": {"conversations": [...]}}。	失败	响应结构不符: API 实际返回的 JSON 缺少顶层的 code 键 (即没有使用 BaseResponse 包装)。

编号	测试用例	请求方法/路径	测试步骤	预期结果（状态码&数据）	状态	失败原因分析
			据。			

五、 主应用测试（main.py）

编号	测试用例	请求方法/路径	测试步骤	预期结果（状态码&配置）	状态	失败原因分析
APP-001	根路径连通性	GET /	验证应用是否成功启动并响应根路径。	200 OK	通过	-
APP-002	路由前缀验证	GET /api/v1/reminders	验证 main.py 中 api_router 挂载的 prefix=settings.API_V1_STR 是否生效。	401 Unauthorized（因为没有 Token），证明路由被正确找到。	通过	-
APP-003	CORS 中间件	OPTIONS /api/v1/auth/login	发送 OPTIONS 请求，检查响应头。	200 OK，且响应头中包含 Access-Control-Allow-Origin: *。	通过	-
APP-004	静态文件挂载	GET /uploads/test.jpg	尝试访问一个已知的静态文件。	200 OK，且返回图片数据（假设测试环境中已放置该文件）。	失败	StaticFiles 配置错误：测试中的 Mock 函数 pathlib.Path.exists 未被调用。这表明对 /uploads/test.jpg 的请求在到达静态文件处理器之前，被其他路由（如默认的 404 处理器）处理了，或者 main.py 中 StaticFiles 的挂载路径或顺序存在问题。

3. 缺陷清单

编号	严重程度	失败原因分析	涉及模块	修复建议
API-A-001	中等	响应结构不符: API 实际返回的 JSON 缺少顶层的 code 键。	ai.py	统一所有 API 响应，确保使用 BaseResponse Pydantic 模型进行包装。
API-A-004	中等	响应结构不符: 同上，GET /plant/conversations 缺少 code 键。	ai.py	统一所有 API 响应，确保使用 BaseResponse Pydantic 模型进行包装。
API-A-002	高等	422 Pydantic 验证错误: 请求体 JSON 不符合 /plant/chat 接口的 Pydantic Schema。	ai.py	检查并修正 chat_data 的 JSON 结构,确保客户端不发送 user_id(应由认证依赖注入）。
API-A-003	高等	404 路由未找到: 路由 /plant/image-analysis 未正确定义或未在应用中挂载。	ai.py	检查 app/api/v1/endpoints/ai.py 中的 @router.post 定义，并确认 ai_router 已被正确包含在主路由中。
APP-004	高等	StaticFiles 配置错误 (404): 静态文件 /uploads/test.jpg 无法访问。	main.py	检查 main.py 中 StaticFiles 的 url_path 和 directory 参数是否正确，并确保 main.py 挂载顺序合理。

5. 改进建议

1. 代码规范性与一致性 统一响应模型: 目前 AI 模块的 API 响应结构不一致（缺少 BaseResponse 包装）。应强制要求所有 API 路由的 response_model 均基于统一的响应模型 (BaseResponse)，以提高客户端处理的健壮性。
依赖注入优化: 确保所有需要用户 ID 的受保护路由（如 POST /plant/chat）都通过 Depends(get_current_user) 或类似机制获取 user_id，而不是依赖于客户端在请求体中发送，以避免 API-A-002 的 Pydantic 验证失败。
2. API 路由与部署配置
修复 AI 路由: 立即排查并修复 API-A-003 的 404 错误,确保 AI 模块的核心路由（图片分析）可访问。
静态文件服务: 修复 APP-004 的 StaticFiles 配置错误，这对于后续存储用户上

传的图片至关重要。