# A. 代码库分析

## 架构概览

**“The Wizard”** 基于mcp-feedback-enhanced的架构，采用**四层设计**，通过单一会话和持久化Web界面实现人与AI的高效交互[[1]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L9-L13)。后端使用 FastAPI + WebSocket 提供服务，前端为HTML/JS实现的Web界面，并可选提供桌面应用界面。系统具备智能环境检测能力，可根据是在本地、SSH远程还是WSL环境调整UI启动方式[[2]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L10-L18)。总体架构如下：

* **第一层：MCP服务层（后端核心）** – 实现MCP协议工具。例如interactive\_feedback函数处理AI的反馈请求[[3]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L430-L439)。负责环境检测、启动Web UI等工作。
* **第二层：Web UI管理层** – 包括WebUIManager单例和WebFeedbackSession会话模型，管理会话生命周期和数据状态[[4]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L54-L62)[[5]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L60-L68)。
* **第三层：Web服务层** – FastAPI应用提供HTTP路由和WebSocket端点，实现与前端的通信[[6]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L61-L67)。
* **第四层：前端交互层** – 浏览器界面，包含HTML模板、CSS样式和模块化的JavaScript逻辑，实现提示管理、会话管理、自动提交、音效通知等丰富功能[[7]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L70-L78)。

这种分层使各模块职责清晰，易于维护和扩展[[8]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L152-L160)[[9]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L160-L165)。此外，通过环境检测，系统可自动判断运行环境（本地/远程/WSL）并选择启动浏览器或桌面应用，以保证在各种环境下都能使用[[10][11]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,provide%20exactly%20the%20same%20functionality)。下面我们详细介绍各主要组件。

## 主要组件及职责

### MCP服务器与后端逻辑

后端由一个MCP服务器进程组成，核心是server.py模块，实现模型上下文协议并提供增强的反馈收集功能[[12]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L3-L12)。它定义了两个主要工具函数： - interactive\_feedback(project\_dir, summary, timeout)：AI用来收集用户反馈的工具[[13]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L16-L24)。当AI调用此函数时，会启动或重用Web UI让用户提供反馈[[14]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L29-L38)[[15]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L53-L60)。 - get\_system\_info()：获取系统和环境信息的工具，可返回JSON格式的环境描述[[16]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L562-L571)[[17]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L573-L582)。

在interactive\_feedback中，系统首先执行**环境检测**[[18]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L454-L463)。代码会检查是否为SSH远程、WSL等情况：

is\_remote = is\_remote\_environment()  
is\_wsl = is\_wsl\_environment()  
debug\_log(f"环境偵測結果 - 遠端: {is\_remote}, WSL: {is\_wsl}")

is\_wsl\_environment()通过检测/proc/version和相关环境变量判断WSL环境[[19]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L142-L150)[[20]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L152-L161)；is\_remote\_environment()则检查SSH连接、远程开发容器、无显示（DISPLAY）等来判断是否远程/无GUI[[21]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L183-L191)[[22]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L207-L215)。如果是WSL，视为本地（因为可以调用Windows浏览器）；如果是SSH远程，则需要特殊处理界面启动方式。

环境检测的结果用于决定界面模式：代码偏好使用**Web UI**（浏览器）[[23]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L458-L466)[[24]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L469-L477)。在某些版本中，引入了桌面模式：通过环境变量MCP\_DESKTOP\_MODE可以强制使用桌面应用界面[[25]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L1126-L1135)[[26]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L1132-L1139)。如果启用桌面模式且可用，系统将通过Tauri启动一个内置浏览器窗口，否则默认直接在浏览器中打开Web界面。

后端逻辑负责在需要时**启动FastAPI服务器**和**WebSocket通信**：interactive\_feedback内部调用launch\_web\_feedback\_ui(project\_dir, summary, timeout)[[27]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L470-L477)，该函数在web/main.py中实现，负责初始化或获取WebUI管理器并启动服务（详见下文）。如果在调用过程中发生异常，后端会捕获并通过ErrorHandler记录错误日志，返回友好的错误信息给AI[[28]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L508-L517)[[29]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L520-L528)。例如，如果Web模块导入失败，函数会返回一个包含错误消息的结果字典，确保AI收到反馈而不是无响应[[30]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L540-L549)[[31]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L553-L559)。

### WebUI管理器与会话管理

**WebUIManager**（位于web/main.py）是Web界面的控制中枢，以单例模式运行。在首次使用时，AI调用launch\_web\_feedback\_ui会实例化或获取这个单例[[32]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L537-L544)。WebUIManager初始化时完成以下工作： - 设置主机和端口：默认绑定127.0.0.1:8765，支持通过环境变量MCP\_WEB\_HOST和MCP\_WEB\_PORT自定义[[33]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L38-L46)[[34]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L48-L57)。若端口被占用，会自动寻找可用端口[[35]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L80-L88)[[36]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L100-L108)。 - 启动FastAPI应用并挂载静态文件、模板路径[[37]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L101-L109)[[38]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L151-L159)。使用Jinja2模板引擎加载HTML模板（index.html和feedback.html）。 - 配置中间件（例如GZip压缩）和启动内存监控器[[39]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L107-L115)。 - 初始化会话管理结构：current\_session保存当前活动会话（最多一个），sessions字典保存所有会话记录以兼容旧逻辑（但新版主要是单一会话模式）[[40]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L113-L121)。在新版设计中，同时存在多个会话的情况已被简化，系统采用**“单一活跃会话”**模型，每次只处理一个会话，以避免资源开销和状态混乱[[41]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L10-L13)。 - 其它管理属性：例如global\_active\_tabs用于跟踪所有浏览器标签页的连接状态[[42]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L116-L124)；\_pending\_session\_update标记是否有会话更新等待通知前端[[43]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L120-L128)；cleanup统计和锁等[[44]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L124-L132)[[45]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L138-L145)。

**WebFeedbackSession**（位于web/models/feedback\_session.py）表示一次用户反馈会话，包含： - 基本属性：session\_id（UUID）、project\_directory（项目目录路径）、summary（AI提供的摘要/问题）[[46]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L124-L132)。 - WebSocket连接引用：websocket属性指向当前会话对应的WebSocket，如果有的话[[47]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L131-L138)。 - 收集的反馈数据：如feedback\_result（文本反馈内容），images（用户上传的图片列表），command\_logs（执行命令的日志），user\_messages（记录每次用户提交的内容）等[[48]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L130-L139)[[49]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L137-L145)。 - 反馈完成标志：feedback\_completed是一个threading.Event事件，用于在线程间同步等待用户反馈提交[[50]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L134-L141)。 - 子进程处理：process属性保存当前执行命令的子进程（如果有），用于在反馈阶段执行shell命令[[50]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L134-L141)。 - 状态管理：status字段表示会话状态（等待中WAITING、活跃ACTIVE、已提交反馈FEEDBACK\_SUBMITTED、完成COMPLETED等）[[51]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L32-L40)。状态流转按照单向顺序设计，每个会话从WAITING开始，用户提交反馈后进入FEEDBACK\_SUBMITTED，然后结束为COMPLETED或TIMEOUT等终态。 - 自动清理配置：会话有auto\_cleanup\_delay（如3600秒）和max\_idle\_time（如1800秒）[[52]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L126-L134)[[53]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L152-L160)。启动时会设定定时器，在会话过期或闲置过长时自动清理释放资源。 - 清理统计：记录清理次数、原因（超时/过期/内存压力/手动等）[[54]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L158-L167)。 - 活跃标签页跟踪：active\_tabs记录了连接到该会话的浏览器标签页及最后心跳时间，用于检测前端仍然在线[[55]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L168-L174)。

WebUIManager通过以下关键方法管理会话： - create\_session(project\_dir, summary): 创建新会话[[56]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L329-L338)。如果当前已有current\_session，会处理旧会话的状态： - 将旧会话的WebSocket连接临时保存，合并旧会话的活跃标签信息到全局[[57]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L348-L356)[[58]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L435-L443)。 - 如果旧会话状态是“已提交反馈”，则调用old\_session.next\_step()将其状态标记为已完成[[59]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L352-L361)。否则保持原状态。 - 确保旧会话在sessions字典中有记录（如果不在则补加）[[60]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L367-L375)。 - 调用old\_session.\_cleanup\_sync()同步清理旧会话资源，但**不关闭**旧的WebSocket（以便稍后转移）[[61]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L372-L380)。 - 实例化一个新WebFeedbackSession，并将current\_session指向它，同时存入sessions字典[[62]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L380-L388)。 - 继承旧的活跃标签页状态到新会话（这样前端能识别仍有已打开的页面）[[63]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L374-L382)[[64]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L378-L386)。 - 如果之前保存了旧的WebSocket连接，则将其直接赋给新会话的websocket属性，实现**WebSocket连接的切换**[[65]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L388-L396)。这样前端无需重新建立连接，新会话可以沿用。日志显示“已將舊 WebSocket 連接轉移到新會話”[[65]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L388-L396)。 - 如果没有旧连接（意味着这是第一次调用，前端尚未连上），则设置\_pending\_session\_update = True，表示下一次有WebSocket连接建立时，需要发送会话更新通知[[66]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L392-L399)。 - 返回新session\_id给调用者（不过MCP框架内部可能并不使用这个返回值）。

* get\_current\_session() / get\_session(session\_id): 获取当前或指定ID的会话对象[[67]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L400-L408)。
* remove\_session(session\_id): 移除并清理一个会话[[68]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L408-L417)。会停止会话、释放资源，从字典删除，若是当前会话则清空current\_session[[69]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L414-L422)。
* clear\_current\_session(): 清空当前活动会话（并从字典中删除）[[70]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L422-L431)。通常在整个流程结束或重置时调用。
* \_merge\_tabs\_to\_global(session\_tabs): 合并旧会话的活跃标签页信息到global\_active\_tabs，并剔除超过60秒未活动的条目，用于跟踪全局还在线的前端标签[[71]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L435-L444)[[72]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L440-L448)。
* get\_global\_active\_tabs\_count(): 获取全局仍活跃的标签页数量，同时清理过期的记录[[73]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L454-L463)。
* broadcast\_to\_active\_tabs(message): 向所有活跃的前端标签发送广播消息（通过当前会话的WebSocket）[[74]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L469-L473)[[75]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L471-L475)。如果当前没有连接则无法广播。

**FastAPI路由和WebSocket：** WebUIManager在初始化时调用setup\_routes(self)设置HTTP路由和WebSocket端点[[38]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L151-L159)。在web/routes/main\_routes.py中： - GET / 路由返回主页面。当没有活动会话时，返回index.html（等待页面）；有会话则返回feedback.html模板[[76]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L56-L65)[[77]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L62-L71)[[78]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L74-L82)。模板会填入会话相关变量，如项目目录、摘要、版本号等[[79]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L76-L84)。这机制确保用户刷新页面时，如果有正在进行的会话，可以直接展示反馈界面。 - GET /api/... 提供一些REST接口，例如获取翻译字符串、多会话状态等[[80]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L91-L100)[[81]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L120-L128)。主要用于前端初始化或调试（翻译用于国际化UI文本）。 - **WebSocket /ws:** 这是前端和后端实时通信的关键通道。在setup\_routes内定义了@manager.app.websocket("/ws")处理函数[[82]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L14-L21)。当浏览器连接此WebSocket： - 后端首先获取current\_session[[83]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L16-L24)。若没有会话，立即关闭连接并报错（代码4004 "No active session")[[84]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L18-L26)——因为按设计AI应先创建会话再连接。 - 然后await websocket.accept()接受连接[[85]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L23-L31)。 - 检查当前会话是否已有一个WebSocket连接且不同于新连接。如果是，则说明前一个连接可能断线或页面刷新了。系统将旧连接替换为新连接[[86]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L28-L36)。日志记录“會話已有 WebSocket 連接，替換為新連接”[[86]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L28-L36)。 - 将当前会话的websocket属性设置为新连接，并记录日志[[87]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L32-L40)。 - 发送初始消息：类型connection\_established，通知前端WebSocket已建立[[88]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L38-L46)。同时可能发送当前会话的信息（session\_id、summary等）[[89]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L44-L51)。尤其如果之前\_pending\_session\_update为True，后端会发出session\_updated消息告知前端载入新会话内容[[90]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L44-L52)。代码中看到，在WebSocket建立后，有逻辑发送session\_updated: new\_session\_created通知[[90]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L44-L52)，这样如果UI已经打开，无需重新打开页面，只是刷新内容。 - 还可能发送一次状态更新消息，包含会话状态和信息[[91]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L50-L58)。日志显示“已發送當前會話狀態到前端”[[91]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L50-L58)。 - 然后进入接收循环：data = await websocket.receive\_text()等待前端发送消息[[92]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L54-L62)。前端可能发送各种指令（见下节）。每当收到一条消息，解析为JSON，然后调用handle\_websocket\_message(manager, current\_session, message)处理[[93]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L60-L64)。 - 这个循环持续进行，直到发生异常或断开。捕获WebSocketDisconnect或ConnectionResetError时，记录日志并跳出循环[[94]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L68-L76)。在finally中，清理会话中的WebSocket引用（如果仍指向该socket）[[95]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L76-L83)。即用户关闭页面后，后端会知道连接断开，将current\_session.websocket设为None[[96]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L80-L83)。

* handle\_websocket\_message(...): 定义了如何处理来自前端的各种命令[[97]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L618-L626)。消息通过字段type区分：
* "submit\_feedback"：用户提交文字反馈（和可选图片）。调用session.submit\_feedback(feedback, images, settings)处理[[98]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L622-L628)。此方法会将反馈保存到会话对象，标记事件完成，并通知AI[[99]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L519-L528)[[100]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L533-L541)（详见后文）。
* "run\_command"：用户请求执行终端命令[[101]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L628-L634)。后端提取command字符串，若非空则调用session.run\_command(command)[[102]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L629-L637)。run\_command在WebFeedbackSession中实现，用于在project\_directory执行shell命令，并将输出通过WebSocket实时发送回前端。
* "get\_status"：前端请求会话状态[[103]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L634-L642)。后端直接通过WebSocket发送当前会话状态信息（SessionStatus等）[[104]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L636-L644)。
* "heartbeat"：前端的心跳消息（用于保活连接）[[105]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L644-L652)。后端更新last\_heartbeat时间，并回复一个heartbeat\_response[[106]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L646-L655)。
* "user\_timeout"：通知后端用户设置的超时时间到了[[107]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L661-L669)。后端会调用session.\_cleanup\_resources\_on\_timeout()清理会话[[108]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L663-L671)。这对应用户主动取消或结束会话的情况，收到后会关闭UI但保持服务器运行（日志提到“保持服务运行以支援持久性”[[108]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L663-L671)）。
* "pong"：前端pong响应，可用于计算延迟，这里只是记录日志[[109]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L670-L678)。
* "update\_timeout\_settings"：前端更新会话超时设置[[110]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L674-L682)。后端调用session.update\_timeout\_settings(...)调整用户自定义的超时功能（允许用户设置更短/更长的等待时间）。
* 其他未知type则记录日志忽略[[111]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L686-L689)。

通过上述机制，WebUIManager和WebFeedbackSession协同管理了整个会话过程：**创建会话 -> 启动服务 -> 建立连接 -> 交换消息 -> 等待反馈 -> 完成/清理**[[14]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L29-L38)[[15]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L53-L60)。

### Web服务层（FastAPI & WebSocket）

Web服务层由FastAPI提供HTTP接口和Jinja2模板渲染，同时处理WebSocket通信，实现双向实时传输。这个层在四层架构中属于第三层[[6]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L61-L67)。

**FastAPI应用**在WebUIManager初始化时创建：self.app = FastAPI(title="MCP Feedback Enhanced")[[37]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L101-L109)。然后静态文件和模板路径通过Starlette/FastAPI的机制挂载： - 静态文件目录映射到/static路径（在\_setup\_static\_files中实现）。 - 模板目录指向web/templates用于渲染HTML页面[[112]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L321-L329)。

**模板渲染：** 当用户通过Cursor IDE或浏览器访问http://localhost:8765/时，会命中GET /路由[[76]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L56-L65)。index.html模板负责显示“等待开始”的页面，没有开始会话时使用；一旦有活动会话，则渲染feedback.html模板[[113]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L62-L70)[[78]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L74-L82)。feedback.html包含整个交互界面的HTML结构，包括：顶部标题、任务摘要、反馈输入区、按钮，以及隐藏的各模块UI片段（如提示管理、会话历史、状态栏等）。模板变量如project\_directory、summary、layout\_mode会注入当前会话信息[[79]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L76-L84)。layout\_mode是界面布局设置，用户可在配置文件中设置默认纵向/横向布局[[114]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L26-L35)。

**WebSocket通信：** 前面介绍了WebSocket /ws的处理流程[[115]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L14-L22)。可以看到WebSocket使得前端的用户操作（提交、命令）能即时传到后端处理，而后端的结果（通知、命令输出）也能即时推送到前端[[116]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L720-L728)[[117]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L740-L748)。这种双向通信保证了AI助手和用户之间的交互实时同步[[15]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L53-L60)。

**接口与依赖：** FastAPI路由和WebSocket handler中的多数逻辑会调用WebUIManager和WebFeedbackSession的方法，所以Web服务层实际上起到了一个“胶水层”的作用：将HTTP/WS请求映射为对管理层组件的方法调用，再把结果返回给前端。这让大部分业务状态保存在WebUIManager/Session中，而路由函数本身保持精简。比如，websocket\_endpoint建立连接后，就不断地把收到的数据交给handle\_websocket\_message，使会话管理逻辑集中在那里，而不是散在路由代码里[[118]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L62-L65)。

需要注意，FastAPI服务器运行在**独立线程**中：WebUIManager调用start\_server()时，会启动一个线程运行Uvicorn服务器[[119]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L608-L616)。实现上，它创建threading.Thread来运行uvicorn.run(self.app, ...)[[120]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L134-L141)。这样后端主线程（AI调用的上下文）不会阻塞，可以异步等待用户反馈。开启线程后通常等待一小段时间以确保服务启动完成[[119]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L608-L616)[[121]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L610-L618)。一旦服务器跑起来，前端连接后，一切交互都在该线程（事件循环）中处理。因此，后端需要注意线程安全问题——比如WebUIManager的操作需用锁保护（它有\_initialization\_lock来防止重复初始化[[45]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L138-L145)）。但WebSocket和会话的处理主要在单线程的异步I/O循环内，所以通常是串行的，不会并发修改同一会话对象（除非我们自己启动线程执行某些任务，如命令输出监视，见下文）。

### 前端界面（浏览器 & 桌面应用）

前端界面分为两种运行模式：**Web UI**（在浏览器中打开）和**桌面应用**（Tauri封装的窗口）。无论哪种模式，界面功能基本一致，代码上也大部分共享。V1重点是Web UI，在浏览器中使用现代HTML5/JS实现。

前端HTML模板和静态资源位于mcp\_feedback\_enhanced/web/static和web/templates目录下： - feedback.html模板是主界面。里面包含了各种UI组件的占位，比如： - **交互区域**：显示AI给用户的摘要(summary)和提供文本输入框、发送按钮让用户提交反馈。 - **提示管理模块**：允许用户管理常用Prompt（提示语）的UI，含增删改查和统计[[122]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,file%20storage%2C%20privacy%20controls%2C%20history)。这个对应JS的PromptManager模块。 - **自动提交模块**：用户可设定一个倒计时自动提交反馈[[123]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,time%20statistics%2C%20flexible%20timeout%20settings)（例如让某些提示定时发送），对应AutoSubmitManager。 - **会话管理**：查看当前/历史会话记录、导出JSON/CSV等[[124]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=creating%20new%20sessions%20or%20commits,formats%20for%20enhanced%20content%20readability)。UI上通常是一个会话历史面板，由SessionManager模块控制。 - **音效通知**：当AI/系统有重要事件，会播放声音或系统通知[[125]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=export%20,formats%20for%20enhanced%20content%20readability)。AudioManager模块负责声音提示，例如收到新消息时“叮”的一声。 - **其他**：如实时统计、状态栏（显示WebSocket连接状态、延迟等），剪贴板一键复制项目路径等小功能[[126]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,formats%20for%20enhanced%20content%20readability)。

前端逻辑采用模块化JavaScript（ES6）。在web/static/js下，有多个JS文件，每个管理一块功能。例如： - app.js：主入口，初始化WebSocket连接，协调各模块。 - session-manager.js：管理多会话UI，维护会话列表、导出、清理操作（v2.4.3引入了更复杂的会话管理界面）。 - prompt-manager.js：提示语管理界面逻辑。 - auto-submit.js：自动提交倒计时逻辑。 - 还有处理图像上传的模块，命令执行输出显示的模块等。

CSS方面，采用响应式设计以兼容不同窗口大小[[127]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,Traditional%20Chinese%2C%20English%2C%20Simplified%20Chinese)。支持多语言，通过在前端加载不同语言的JSON资源来切换UI文本[[128]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L96-L105)[[129]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L100-L108)。

**桌面应用模式：** v2.5版本引入了基于Tauri的跨平台桌面应用[[130]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L30-L39)。Tauri应用包装了同样的前端内容，但运行在一个原生窗口中。实现上，Tauri项目位于src-tauri目录。桌面模式的启动： - 如果检测到MCP\_DESKTOP\_MODE=true，WebUIManager不会打开浏览器，而是调用launch\_desktop\_app(url)[[131]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L629-L638)[[132]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L630-L638)。该方法尝试导入mcp\_feedback\_enhanced.desktop\_app.launch\_desktop\_app函数。如果以pip安装包方式存在，就用之，否则退而求其次，导入开发环境模块mcp\_feedback\_enhanced\_desktop.launch\_desktop\_app[[133]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L669-L678)[[134]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L681-L690)。 - launch\_desktop\_app\_func是一个异步函数，会启动Tauri应用并返回其实例[[135]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L704-L712)。启动成功则保存实例引用（desktop\_app\_instance）[[136]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L706-L714)[[137]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L32-L40)。如果导入或启动失败，则回退到普通浏览器模式[[138]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L713-L721)。 - 桌面应用启动后，Tauri会在内部打开一个WebView加载相同的Web界面（一般会加载http://127.0.0.1:端口的地址或嵌入静态文件）。从用户看，效果如同一个独立应用窗口。 - 当用户提交反馈后，在桌面模式下，代码会立即关闭桌面窗口：session.submit\_feedback里检测到桌面模式，调用manager.close\_desktop\_app()停止应用[[139]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L554-L563)[[140]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L564-L570)。这一点和Web模式不同：Web模式下页面会保持打开等待后续可能的会话更新，而桌面模式设计上是在每次反馈提交后关闭窗口来节省资源（因为桌面窗口可能不会自动重用, 或者因为在远程场景下桌面模式其实不适用）。

由于当前重点在Web UI，桌面模式的部分可以暂不深入。需要确保的是我们的新功能在桌面模式下要么被禁用要么正常运作。例如，在桌面模式中，不存在浏览器多标签的问题，所以一些逻辑（如不重复打开窗口等）通过if desktop\_mode: ...处理好了[[131]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L629-L638)[[132]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L630-L638)。我们实现新功能时，也要注意检查MCP\_DESKTOP\_MODE，必要时调整行为（例如桌面模式可能无法弹出浏览器进行登录OAuth之类，但这些在当前场景不涉及）。

**同步机制：** 浏览器UI通过WebSocket实时获取更新：例如， - 当AI启动界面时，后端detect已有无活动标签。如果已有，走刷新逻辑，不开新窗口[[141]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L635-L644)[[142]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L638-L647)；如果没有，则open\_browser(url)在系统上打开浏览器访问界面[[143]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L649-L657)。 - 如果浏览器已开且连接，AI再次调用时，后端通过WebSocket发送session\_updated通知[[144]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L637-L646)[[145]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L640-L648)。前端收到后，会局部更新UI而非整页刷新，从而实现**无缝状态切换**（即在一次会话续集中用户界面不会闪烁或重置）[[146]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L62-L70)[[147]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L68-L76)。 - 提交反馈后，后端通过WebSocket告知前端反馈已收到，从而前端可以显示一个提示如“反馈已提交，等待AI响应...”[[148]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L540-L548)[[149]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L544-L552)。

### 环境检测与界面切换

正如前述，环境检测逻辑保证The Wizard在不同开发环境下都能工作，并选择合适的UI模式[[11]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,provide%20exactly%20the%20same%20functionality)。总结来说： - **本地环境（带图形界面）**：默认直接在本机浏览器打开Web界面[[150]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L137-L143)。使用Python内置webbrowser.open(url)调用系统默认浏览器[[150]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L137-L143)。WebUIManager的smart\_open\_browser会判断如果没有活跃标签，就调用open\_browser执行这个操作[[143]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L649-L657)。浏览器打开后，用户即可在本机查看UI。 - **WSL环境（Windows Subsystem for Linux）**：因为WSL本身没有浏览器，需要调用Windows的浏览器。代码在utils/browser.py提供open\_browser\_in\_wsl(url)来实现[[151]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L63-L71)[[152]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L70-L78)。它首先尝试通过cmd.exe /c start url来让Windows打开链接[[152]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L70-L78)；如果失败再尝试powershell.exe Start-Process[[153]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L87-L95)；再不行则尝试wslview命令[[154]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L104-L113)。这样确保在WSL环境下用户的Windows浏览器能够弹出页面。smart\_browser\_open内部通过is\_wsl\_environment()检测WSL并使用该逻辑[[155]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L131-L139)。日志会记录使用了哪种方式打开[[156]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L76-L83)。 - **SSH远程环境（或无DISPLAY的Linux服务器）**：此时无法在服务器上直接启动浏览器。系统的策略是：如果检测到SSH远程，它不会调用webbrowser打开，而是**建立SSH隧道**或提示用户自行建立。文档指出两种方案：配置--host 0.0.0.0监听所有接口并用SSH本地端口转发，或在远程上启动一个文本浏览器模式[[157]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L130-L139)[[158]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L152-L160)。在实现上，smart\_open\_browser首先检查MCP\_DESKTOP\_MODE[[131]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L629-L638)，然后检查活跃标签[[141]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L635-L644)。对于纯远程（无浏览器），由于webbrowser.open在无GUI环境可能抛异常或无效果，所以Minidoracat的README建议**使用SSH隧道**来从本地主机访问[[159]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L148-L156)。Minidoracat项目或Cursor在这种情况下可能会自动尝试用ssh端口转发（README提到“自动建立 SSH 隧道”，若失败需要手动建立[[157]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L130-L139)[[158]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L152-L160)）。我们的代码里，is\_remote\_environment()会返回True，然后: - smart\_open\_browser里如果is\_remote但不是WSL，它不会提前退出，而是走webbrowser.open[[132]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L630-L638)。在没有DISPLAY的linux上，Python的webbrowser模块通常尝试调用xdg-open之类，这会失败。这时异常会被捕获，fallback逻辑打印失败并仍尝试open\_browser[[160]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L653-L657)。 - 可以改进之处：对于明确的SSH场景，我们或许可以不要调用webbrowser，而是直接输出一条提示告诉用户通过SSH隧道访问，例如在服务器端打印“请在本地主机浏览器打开http://localhost:8765”。目前Minidoracat文档已经指导了这种做法，我们可以沿用。 - 另外，Minidoracat v2.6引入了**“桌面模式”**配置。如果用户在远程有显示环境（比如通过VSCode远程容器，但本地有界面），可以将MCP\_DESKTOP\_MODE设为true，在本地通过X11转发用桌面应用显示UI[[161]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L58-L65)。但这是高级配置，默认还是web模式配合隧道更常用。

* **Docker/容器环境**：视为特殊远程，推荐做法类似SSH，通过端口映射+本地浏览器访问[[162]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L172-L180)[[163]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L174-L183)。我们的is\_remote\_environment()也检查了Docker容器标志文件/.dockerenv[[164]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L194-L202)，会将其视为远端。处理与SSH类似。
* **环境变量**：可以通过MCP\_WEB\_HOST和MCP\_WEB\_PORT指定监听地址和端口。如果在远程想让其他机器访问，需要用--host 0.0.0.0或设置env，否则默认127.0.0.1只能本地访问。Minidoracat的部署指南明确提到了这一点[[159]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L148-L156)。我们的WebUIManager默认127.0.0.1，但文档示例在远程部署时用了mcp-feedback-enhanced web --host 0.0.0.0 --port 8000[[165]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L146-L154)。
* **桌面 vs 浏览器**：环境检测也关系到是否使用桌面应用。Minidoracat推荐在mac/Windows本地可以用桌面模式获得更原生体验[[166]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L60-L64)。但在SSH/WSL通常使用Web模式，因为桌面模式需要本地显示。我们的代码通过MCP\_DESKTOP\_MODE人为控制：用户必须主动配置为true，否则默认false即使用Web[[161]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L58-L65)。这样确保除非用户要求，否则不会意外尝试启动桌面app。在桌面模式下，smart\_open\_browser直接返回True不打开浏览器[[131]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L629-L638)[[132]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L630-L638)；同时launch\_web\_feedback\_ui中检查desktop\_mode决定调用launch\_desktop\_app或smart\_open\_browser[[167]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L1124-L1133)[[168]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L1130-L1138)。因此整个界面启动逻辑根据环境和配置自动选择最佳方案，真正做到了**“自动界面切换”**。

### 自动命令执行与反馈循环

“自动命令执行”指AI或系统在某些阶段自动运行预设命令来辅助开发流程。根据CHANGELOG，v2.6.0新增了在创建新会话或AI“提交”结果后自动执行命令的功能[[169]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L10-L18)。在当前代码中，这实现为： - WebFeedbackSession的run\_command方法，以及AI可通过特殊调用触发。例如，当AI结束一次任务，它可以要求执行git diff或npm test之类命令，然后将结果作为反馈内容返回给它自己，以决定下一步。这种“工具使用”由AI的提示规则控制，但本MCP服务器提供了执行命令并获取输出的能力。 - handle\_websocket\_message里，当前端发送{"type": "run\_command", "command": "..."}时，调用session.run\_command(command)[[170]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L628-L636)。不过目前代码中AI不会自主请求run\_command，通常是用户通过UI点击触发。Minidoracat提供了UI按钮让用户执行一些常用命令或AI提示用户运行测试。例如UI可能有一个“Run Tests”按钮，点下就发送run\_command: "pytest"消息。 - v2.6的“Auto Command Execution”指的是\*\*在特定事件后自动执行命令\*\*。例如，配置项目中，当AI新建一个会话，如果想自动跑一次单元测试，可以在配置中指定命令。Minidoracat README没有细节，但CHANGES提到“after creating new sessions or commits”自动执行命令[[169]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L10-L18)。这或许通过在create\_session或submit\_feedback`时检查配置然后调用run\_command实现。在当前代码中，我们没直接看到自动触发，但可能存在配置逻辑我们没展开。反正，该功能是可拓展的。

**反馈循环（Feedback Loop）**是整个系统的核心工作流：AI调用interactive\_feedback→ 用户通过UI提供反馈→ 后端将反馈返回AI→ AI据此调整→ 如有需要再次调用interactive\_feedback，循环往复，直到任务完成[[171]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L19-L27)[[15]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L53-L60)。这建立了一个人机协作的闭环。

具体流程已经在前文随组件阐述，这里从更高层总结： 1. **AI发起**：AI助手（如Cursor中的Claude模型）执行到某一步需要用户确认或反馈时，会调用interactive\_feedback(project\_dir, summary, timeout)工具[[172]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L29-L37)。它会将当前项目路径和一段总结（比如“我完成了X功能，请你检查代码。”）传给MCP服务器。 2. **服务器准备 UI**：后端验证参数，进行环境检测[[173]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L33-L41)[[18]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L454-L463)，初始化WebUIManager并创建新的反馈会话[[173]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L33-L41)[[174]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L34-L38)。如果之前已有活跃会话，则按上文所述更新会话而不重新打开页面[[146]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L62-L70)。 3. **启动Web服务**：如无运行则启动FastAPI服务器，分配端口，建立路由和WebSocket[[175]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L35-L43)[[176]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L37-L45)。然后调用smart\_open\_browser或launch\_desktop\_app打开界面[[167]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L1124-L1133)[[26]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L1132-L1139)。UI启动采用**“智能浏览器打开”**：若检测到已有浏览器页面连过来（global\_active\_tabs不为空），则不新开窗口而是走刷新流程[[141]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L635-L644)[[142]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L638-L647)；否则在本地用浏览器打开，或在WSL用Windows浏览器打开，或在SSH提供隧道方法。 4. **用户提供反馈**：用户查看AI给出的结果/摘要，在UI中输入反馈意见并点击提交[[15]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L53-L60)。这可能是文字，也可能附带截图/文件。前端通过WebSocket发送submit\_feedback消息[[15]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L53-L60)[[98]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L622-L628)。此外用户可以在反馈前使用UI中的“运行命令”功能执行一些命令验证AI的成果。例如，用户觉得需要跑一下单元测试再决定反馈，就点击“运行测试”，UI发送run\_command消息，后端执行测试命令并把输出通过WebSocket逐行返回前端[[177]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L715-L723)[[116]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L720-L728)。前端将命令输出显示在界面上，供用户参考。 5. **命令执行反馈（可选）**：如果用户执行了命令，Session会把输出累积到command\_logs[[178]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L496-L504)。当最终提交反馈时，这些日志也会成为反馈内容的一部分。create\_feedback\_text函数将用户文字、命令日志、图片信息组合成一段综合的反馈文本返回给AI[[179]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L279-L288)[[180]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L283-L290)。例如它会在反馈文本中包含一个“=== 命令执行日志 ===”部分，把shell输出附上[[181]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L280-L288)[[182]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L283-L291)。这样AI可以看到用户不仅给了评论，还跑了测试并提供了结果，从而AI能据此更好地调整[[181]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L280-L288)[[182]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L283-L291)。 6. **反馈提交**：Session的submit\_feedback会将feedback\_result设为用户文字，images设为图片数据，settings设为一些设置选项[[99]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L519-L528)[[100]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L533-L541)。然后把会话状态从WAITING更新为FEEDBACK\_SUBMITTED[[183]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L536-L544)[[184]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L538-L546)。最重要的是调用self.feedback\_completed.set()[[185]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L539-L542)——这会唤醒等待反馈的线程（后端AI调用处正等待这个事件)[[186]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L483-L490)。随即通过WebSocket给前端发送一个通知，类型为notification，代码为FEEDBACK\_SUBMITTED，表示反馈已收到，severity为success[[148]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L540-L548)[[149]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L544-L552)。前端可在UI上提示“反馈已成功发送”之类。对于桌面模式，还触发关闭桌面窗口[[139]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L554-L563)[[140]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L564-L570)。 7. **返回AI**：在后端AI调用上下文中，interactive\_feedback等待feedback\_completed事件[[187]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L483-L491)。一旦用户提交，事件被set，函数继续执行：汇总feedback\_items列表，将用户文本反馈和图片封装为MCP协议的返回（例如TextContent对象和Image对象列表）[[188]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L480-L488)[[189]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L492-L500)。然后返回这些内容[[190]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L498-L506)。AI接收到这些信息，就会将其纳入后续思考。例如，如果用户说“Looks good, proceed”，AI就继续完成任务；如果用户提出修改建议，AI就调整策略。 8. **重复/继续**：AI可能在后续又调用interactive\_feedback，检查第二轮反馈。例如在Happy Path场景，AI修正了问题后，再次请用户确认。这时MCP服务器检测已有活跃会话，因此不重新启动，只更新现有会话内容并通过WebSocket通知前端更新摘要[[146]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L62-L70)。前端无需刷新页面，只看到AI的新总结/提问。用户再次提交反馈... 如此循环。每次循环其实后端会create\_session创建新Session对象但继承旧连接[[191]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L338-L346)[[65]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L388-L396)，所以在用户看来是一场连续的会话。 9. **结束**：如果AI判断不需要更多反馈（例如用户明确说结束或AI达到停止条件[[192]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L440-L448)[[193]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L442-L450)），它将不再调用工具，流程结束。我们后台可能有个超时监测：wait\_for\_feedback用了timeout参数，如果超过比如10分钟没等到用户反馈，会执行清理，返回一个超时错误[[194]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L502-L510)。AI拿到这个超时错误后也会结束流程并通知用户超时了。

整个反馈循环通过这种**人机对话+工具**的方式，避免了AI盲目执行可能错误的操作，把决策权交给用户审核，这正是PRD中强调的“开发者始终是最终控制者”的体现[[195]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L9-L12)。

### 并发与通信模式

The Wizard的实现需要在确保响应及时的同时，处理一定的并发/异步问题。主要的并发模式和同步机制有：

* **多线程**：后端采用一个线程运行FastAPI服务器（Uvicorn），同时主线程继续执行AI调用逻辑等待事件。这种架构让AI调用不会阻塞UI循环。WebUIManager的start\_server()会spawn一个server\_thread[[120]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L134-L141)，日志提示在指定host:port启动[[196]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L142-L149)。主线程通过feedback\_completed事件等待用户完成[[187]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L483-L491)。等待时，用asyncio.get\_event\_loop().run\_in\_executor(None, wait\_in\_thread)来阻塞等待事件，但不阻塞事件循环[[187]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L483-L491)[[197]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L485-L493)——实际等待是在Executor的线程中进行，从而主协程挂起让出控制，其他任务（比如命令输出）仍可运行。这种线程+事件结合确保了一个AI调用仅占用很少的协程资源。
* **异步IO**：FastAPI和WebSocket运行在异步事件循环。大部分操作（接收消息、发送消息）都是非阻塞的。命令执行输出的读取是阻塞IO，为此run\_command实现中用了asyncio.create\_task启动异步任务并run\_in\_executor读子进程的输出，逐行通过WebSocket发送[[198]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L713-L721)[[199]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L719-L727)。这样不会阻塞整个事件循环，其它WebSocket消息（心跳等）仍可处理。
* **锁与同步**：WebUIManager有\_initialization\_lock在init时使用，防止多次初始化发生并发（例如两个AI线程同时调用interactive\_feedback首次启动时）[[45]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L138-L145)。此外，feedback\_completed的wait/notify机制是同步用户反馈的关键，在线程间传递信号[[187]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L483-L491)[[185]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L539-L542)。
* **WebSocket保活**：前端发送heartbeat/pong来保活连接[[105]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L644-L652)[[109]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L670-L678)。Session记录last\_heartbeat[[105]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L644-L652)，可用于判断连接超时和清理。WebUIManager的global\_active\_tabs也基于定期刷新last\_seen时间，过期的就移除[[72]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L440-L448)。Minidoracat实现了**网络断线重连**机制：如果WebSocket断开，前端可自动重连[[200]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L40-L48)，并服务端替换连接[[86]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L28-L36)。v2.5还加了指数退避重连算法等以提高稳定性[[201]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L46-L50)。
* **单会话原则**：通过限制同时只有一个活动会话，系统避免了需要处理多个前端并发反馈的复杂情形。这极大简化了并发模型。虽然'sessions'字典允许存储多个Session对象，但逻辑上只current\_session真正活跃，其它要么已完成等待导出，或等待清理。这样，我们不用考虑两个会话争夺UI或AI的状况。为此，PRD也选择了在一次Spec-then-Code流程中，只处理当前任务直到完成，再开始下一个（而不是并行好几个Wizard会话）。
* **数据一致性**：由于UI和后台分离，通过WebSocket通信，必须小心保持一致性。例如，session\_id的切换、会话状态的更新，服务端都及时通知前端，并且前端在发送消息时也会带上type来指明操作针对当前session，否则如果两个session切换很快可能混淆。Minidoracat用 \_pending\_session\_update 和 session\_updated消息来确保过渡平滑[[66]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L392-L399)[[89]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L44-L51)。这些细节保证当AI创建新会话时，前端恰好已经连着旧会话，它会收到session\_updated然后加载新summary，不会把用户还留在旧状态页面上。

总之，The Wizard采用“单任务异步循环+辅助线程”的方式实现人与AI的同步互动，在需要等待时释放资源，让UI依然流畅，确保数据传递准确及时。

### 会话数据持久化与导出

为了增强可审查性和后续分析，系统会将每次用户反馈的数据保存，并允许导出整个会话记录[[124]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=creating%20new%20sessions%20or%20commits,formats%20for%20enhanced%20content%20readability)。

* **本地文件存储**：当用户提交反馈后，interactive\_feedback函数会调用save\_feedback\_to\_file(result)将反馈数据保存为JSON文件[[202]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L475-L483)。这发生在每轮交互完成后：返回给AI前即刻[[27]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L470-L477)[[202]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L475-L483)。保存内容包括用户的文字、图片（已转成base64以便JSON序列化[[203]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L242-L251)[[204]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L244-L253)）、命令日志等。文件名如果未指定则由create\_temp\_file生成，通常放在临时目录~/.cache/interactive-feedback-mcp-web/下[[205]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L63-L66)以feedback\_\*.json形式。日志会记录文件保存路径[[206]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L259-L264)。
* 例如，一次反馈可能保存为/home/user/.cache/interactive-feedback-mcp-web/feedback\_abc123.json，里面是:
* {  
   "interactive\_feedback": "用户写的文字",  
   "command\_logs": "=== 命令执行日志 ===\n...输出...",  
   "images": [ { "name": "screenshot.png", "data": "<base64>", "size": 12345 } ],  
   "settings": { ... }  
  }
* 方便日后复现或Debug。
* **导出历史**：SessionManager模块支持将会话历史导出为JSON、CSV或Markdown[[124]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=creating%20new%20sessions%20or%20commits,formats%20for%20enhanced%20content%20readability)。Minidoracat v2.6在界面中提供了导出按钮。实现应在前端收集当前session.user\_messages（里面记录了每次用户提交内容[[207]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L576-L585)[[208]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L580-L589)）或利用后台已经保存的JSON文件拼装。
* JSON导出可能就是反馈\_\*.json直接提供。
* CSV/Markdown则会把每条交互（AI summary, user feedback, images count, timestamp等）整理成行/段落。CHANGELOG确认支持多格式导出[[169]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L10-L18)。
* **会话版本控制**：当前没有真正的版本管理（比如保存每次用户修改草稿的版本）。但Session的user\_messages记录了每次提交的时间戳和内容[[207]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L576-L585)[[209]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L580-L588)。通过它可以重建出一个对话序列。例如Markdown导出可能把AI的每次Summary和用户Feedback按对话形式列出。国际化文本的存储也被考虑，日志里看到“session history multilingual support”fix[[210]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L22-L25)。
* **隐私和期限**：Minidoracat让用户自己掌控历史记录，应该不会上传这些数据，只存在本地。Session有auto\_cleanup但清理指的是内存和临时文件，不一定删除导出文件（除非临时目录被清空）。UI提供“清理会话”选项，可手动删除记录[[211]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L72-L79)(like clear history after done)。目前Wizard没有线上后端，所以数据持久性仅靠本地文件。
* **事件日志**：除了反馈内容，系统本身日志（通过debug\_log）也会详细记录行为，包括浏览器启动、命令执行、错误等，通常输出在stderr或文件。ErrorHandler也会把错误汇报ID和stacktrace保存（例如error\_logs）。PRD关心的**核心流程完成率**指标，需要在每次流程结束时打点。我们可利用日志或增加计数：例如在用户点击“Accept Code”时记录一次“completed\_ok”。可在后续实现。

总的来说，当前实现已经具备了基本的数据留存能力，这在建立信任上很重要——用户可以回顾AI每一步输出和自己反馈。为满足PRD度量，我们可能要**强化日志**，比如记录多少会话成功走完Happy Path，多少中途中止等。不过这些在V1可以通过分析日志来近似统计。

### 错误处理与系统弹性

系统通过多层次的错误处理和健壮性设计，来保证即使出错也尽可能优雅降级，不损害用户信任[[212]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L10-L18)。主要体现在：

* **统一错误处理框架：** utils/error\_handler.py定义了ErrorHandler类，用于捕获异常、分配错误ID、输出日志并生成用户友好的错误消息。后端所有主要操作都用try/except包裹，并调用ErrorHandler记录。例如，在interactive\_feedback顶层[[28]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L508-L517)捕获任意异常：
* error\_id = ErrorHandler.log\_error\_with\_context(e, context={...}, error\_type=ErrorType.SYSTEM)  
  user\_error\_msg = ErrorHandler.format\_user\_error(e, include\_technical=False)  
  debug\_log(f"回饋收集錯誤 [錯誤ID: {error\_id}]: {e!s}")  
  return [TextContent(type="text", text=user\_error\_msg)]
* 这样即使后端出错（比如无法启动Web UI模块），也会向AI返回一个文本内容“抱歉，發生了错误，ID:xxxx”，而不是直接抛异常中断[[213]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L510-L518)[[29]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L520-L528)。AI拿到这个文本会告诉用户发生了系统错误而非安静失败。
* **日志与调试：** debug\_log函数会将调试信息输出到标准错误或日志文件（取决于配置）。Minidoracat提供MCP\_DEBUG=true环境变量可开启详细日志。所有关键步骤都有日志：环境检测结果[[23]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L458-L466)、UI模式[[214]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L459-L466)、端口分配[[215]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L96-L104)、会话创建/切换[[216]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L343-L351)[[62]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L380-L388)、WebSocket事件[[217]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L26-L34)[[218]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L74-L82)、命令执行[[219]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L730-L738)[[117]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L740-L748)、清理资源[[220]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L780-L788)[[221]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L840-L848)等等。这些日志帮助开发者或运维迅速定位问题。
* **前端提示：** 前端有多种通知机制：
* WebSocket发送的notification消息会在UI以提示框/气泡显示，如“连接断开”、“反馈超时”、“命令执行错误”等[[148]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L540-L548)[[149]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L544-L552)。Severity字段指示是warning还是error等，可以着色突出。
* 音效和系统通知也会提醒用户，比如长时间无操作超时时可能弹系统通知[[222]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,formats%20for%20enhanced%20content%20readability)。
* 这些都提高了**透明度**：用户能看到Wizard内部发生了什么（如重新连接了，还是出错了）而不会莫名其妙。
* **网络故障恢复：** WebSocket掉线时，前端按照Minidoracat实现，会自动尝试重连；重连成功后，服务端替换连接并发送session\_updated消息确保UI同步。Release notes提到采用指数退避和抖动以避免频繁重连风暴[[201]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L46-L50)。如果网络暂时不稳，这套机制能自动恢复。而且SessionManager里有连接状态指示（UI顶上可能有个🟢/🔴灯显示WS连接状态）。
* **超时与资源清理：** 系统有完善的超时清理策略：
* 如果用户在timeout时间内没有提交反馈，wait\_for\_feedback将超时[[194]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L502-L510)。它调用\_cleanup\_resources\_on\_timeout()回收资源并抛出TimeoutError[[223]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L504-L512)。AI收到这个异常会结束等待，可能告诉用户“反馈超时了”。后端cleanup中，会关闭WebSocket连接并通知前端会话超时[[224]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L804-L813)[[225]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L819-L828)，然后终止任何正在运行的命令进程[[221]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L840-L848)。
* 用户也可以主动结束：UI上点“取消”，前端发送user\_timeout消息[[107]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L661-L669)，后端同样调用cleanup\_on\_timeout来处理[[108]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L663-L671)。与超时情况相同，UI会收到session\_cleanup通知。
* 每个Session还有自动过期清理：auto\_cleanup\_delay默认3600秒，一旦会话创建一小时后仍存在，就会由后台定时器触发清理（如果它早没完成或忘记关闭UI）。max\_idle\_time默认30分钟，若会话空闲（无任何交互）超过此，也可清理。清理时会区分原因设置SessionStatus为TIMEOUT或EXPIRED等[[226]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L870-L874)。这些确保长时间无人管的session不会无限占用内存或挂着端口。
* 清理操作本身也相当健壮：用try捕获各步骤异常，哪怕某一步失败也不会影响后续步骤[[227]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L784-L792)[[228]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L840-L849)。避免资源泄漏。
* **命令执行隔离：** run\_command时为了安全：
* 禁用了shell=True，仅直接执行解析后的命令列表[[229]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L686-L695)。
* 用shlex.split解析命令字符串，并进行危险片段过滤（如 &&, ||, ;, `, rm -rf 等）[[230]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L84-L93)[[231]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L98-L107)。若检测到这些高危模式，会raise异常拒绝执行[[232]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L102-L110)。
* 这样防止AI或用户执行诸如删除文件的恶意命令，提高安全性。
* 命令运行也加了超时：在cleanup时，如果子进程还没退出，先terminate，3秒后还不退出就kill掉[[228]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L840-L849)。避免卡死的子进程挂着。
* **异常情况处理：**
* 如果前端页面意外关闭：服务端通过WebSocket disconnect事件知道，把session.websocket设为None[[95]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L76-L83)。SessionCleanupManager可能随后清理这个会话（v2.4.3引入SessionCleanupManager专门定期清理超时会话文件等）。UI关闭不会导致AI挂起，因为AI一开始就有timeout机制，不管UI是否反馈最终都会返回（要么用户反馈，要么超时错误）。
* 如果AI端出问题，比如AI崩溃未调用我们继续，我们服务端session会在idle超时后自行清理，不会永驻内存。

综上，系统采用多重保障手段来提升健壮性：**出错不悬停、有日志可追溯、用户有提示、自动恢复继续**。这些都服务于建立用户对AI流程的信任：即使AI或工具出错，用户能及时知道，系统能安全恢复或至少失败得有迹可循，不会出现不可控的“黑箱”行为。

### 扩展性与插件接口

代码结构的模块化和配置驱动使其具备一定的可扩展性，为将来增加新功能、新工作流留下接口：

* **多模式支持：** 当前设计已经考虑了多种使用模式（快速脚本、TDD等）在UI上的入口。比如UI模式选择通过@TheWizard触发，目前虽然没有具体实现模式切换逻辑，但已经能够弹不同UI场景。在Minidoracat的架构图里，“四象限”的概念可能没有硬编码实现，但可以通过改变AI prompt的引导和前端表现支持不同协作模式。我们可以复用同一套基础架构，只是在AI调用和UI流程上做不同定制来实现不同模式。例如未来加入“TDD模式”，可以在AI prompt中标记，然后让UI显示不同引导文本和可能省略蓝图阶段，直接进入测试阶段等。这种灵活性来源于我们**解耦了AI逻辑和UI逻辑**：AI通过协议告知我们它需要什么（目前主要是需要一个反馈），UI通过用户操作提供什么。要支持新模式，大多是在AI侧决定流程，本工具需要提供对应UI交互能力即可。
* **前端模块化：** 前端各个功能是以JS模块形式实现，可比较容易添加新模块。例如PRD提出的“规格先行开发”需要一个**蓝图设计UI**和**测试用例UI**，这在前端可以作为新的模块加入。不需要推翻整个页面结构，可以在feedback.html里增加相应的组件区域，并写对应JS来handle。这种插件式前端开发门槛较低，因为已有PromptManager, SessionManager作参考，按需复制修改即可。例如可以新增blueprint-designer.js管理蓝图绘制，testcase-manager.js管理测试列表。
* 而且CSS采用响应式，新增组件只要跟随设计原则，也能在各种窗口下良好呈现。
* **后端命令/工具扩展：** MCP协议支持注册多个工具函数（fastmcp库可以@mcp.tool注册不同方法）。目前server.py实现了两个工具[[13]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L16-L24)。我们可以扩展更多，如@mcp.tool def open\_browser(url): ... 或 def run\_tests()之类。不过因为现在Focus在feedback，我们或许不需要AI直接调用更多工具。但有扩展空间：比如未来可以实现一个apply\_patch工具供AI自动修改文件，或run\_analysis工具给AI获取项目依赖信息。这些都可以通过在server.py新增函数，并利用fastmcp自动暴露给AI。
* 同时，session模型等已考虑了执行命令、上传图片等多模态交互。未来如果要扩展支持比如文件传输、大文本等，也可以在images机制上扩展成通用“附件”概念来上传文件内容等等。
* **配置与定制**： 项目支持通过JSON配置文件或环境变量定制行为。如~/.config/mcp-feedback-enhanced/ui\_settings.json可以存UI布局、是否启用详细base64图像、超时时长等[[233]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L330-L338)[[234]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L332-L340)。通过这些配置，用户/开发者无需改代码就能调整一些策略。PRD提及的“基于配置文件的可插拔引导框架”在V1不实现，但已有的配置机制是一个起点。未来可扩展一个配置schema用于定义不同工作流的步骤，这样切换模式就像加载不同配置一样。这需要一定重构，但目前代码结构已经把例如UI文本、多语言、端口选择等外部化，使之易改。
* **插槽/钩子（Hooks）：** 检查代码可以发现一些“hook points”。如：
* WebUIManager.\_init\_async\_components里可以并行初始化一些可选组件。现在调用了MemoryMonitor等。如果我们想加比如“性能监控模块”，可以仿照MemoryMonitor集成。
* PromptManager可以视为一个前端插件点——开发者可预置一组Prompt模板，文件位于web/locales目录，也可经UI添加。这相当于**小型“Prompt Store”**，给出了扩展AI行为的一个接口点。
* SessionCleanupManager在web/utils中独立出来，用于会话清理调度。如果以后我们要引入比如定期将历史记录打包归档的功能，可以在这里挂钩：在清理前把session数据写到某个总日志。
* **API参考文档：** 项目提供了docs/architecture/api-reference.md，里面记录了前后端消息类型、SessionStatus常量等。保持接口稳定和文档，对于扩展或第三方集成很重要。如果将来要让The Wizard支持插件（例如第三方开发一个新UI界面），有这些API文档会方便对接。当前Beta阶段大多内部使用，这算种潜在扩展性。
* **与其他平台集成：** 目前Wizard可以用于Cursor、Cline、Windsurf等多个IDE（它们都实现了MCP标准）。通过标准协议，Wizard的功能不局限于Cursor IDE——这也是扩展性的一种体现：不需要针对每个平台重新写，只要AI端支持MCP调用，我们这个后端就能服务。这样当产品拓展到别的开发环境或云IDE时，可重用绝大部分代码。

总的来说，**The Wizard的架构是开放的**。新增功能多半可以在现有框架下添加模块、调用新的tool或调整配置实现，而不必重构核心。后续若要做“Guide Store”（引导工作流市场），可以考虑将每个模式的UI和逻辑抽象出来，通过配置或插件形式加载不同模式，这需要进一步设计，但目前的模块化已经朝这个方向靠拢。我们可以在满足PRD V1需求的同时，尽量以可扩展的方式实现，以便未来拓展轻松。

# B. PRD对照与差距分析

## 功能支持情况总览

以下将PRD中的主要功能需求与当前代码实现进行一一比对，标明其支持程度：

* **FR-1.1 智能任务分流（模式选择）**：*当用户通过@TheWizard发起任务时，系统提供四种协作模式选项。*
* **当前支持：** **未实现。** 目前interactive\_feedback接口没有区分模式，也没有前端UI让用户选择模式。系统默认走一种统一的反馈流程，没有四象限的选项界面。
* **说明：** 要实现此功能，需要在工具被调用时让前端显示模式菜单。当前代码未包含这部分逻辑和UI。
* **FR-1.2 专注规格先行模式**：*V1核心聚焦规格先行开发场景。*
* **当前支持：** **未完全支持。** 虽然当前反馈工作流可用于规格先行的目的，但没有针对“规格先行”的特殊流程控制。也就是说，没有单独处理设计规格、测试、实现的多阶段流程。
* **说明：** 目前流程是AI->用户反馈->AI调整这样通用循环，没有明确的“先规格再编码”步骤划分。
* **FR-2.1 规划阶段蓝图生成（时序图、类图）**：*系统必须在专用弹窗UI中，生成至少时序图和类图两种视图的蓝图，以供用户交叉验证。*
* **当前支持：** **未实现。** 当前UI没有绘制架构蓝图的功能，更没有专门弹窗展示Mermaid图表。AI也不会生成Mermaid图交给UI。
* **说明：** 代码未集成Mermaid或图形渲染库。要实现，需要AI生成diagram描述，前端渲染。当前缺少此逻辑。
* **FR-2.2 蓝图可视化双栏编辑器**：*UI必须提供双栏编辑器，允许开发者直接修改图表源码（如Mermaid），实时预览结果。*
* **当前支持：** **未实现。** UI无图表编辑器组件。只有一个通用的文本输入框供用户反馈，不支持Mermaid源码编辑/预览。
* **说明：** 需新增前端组件支持Mermaid live编辑预览。目前没有此类功能代码。
* **FR-2.3 自然语言修改蓝图**：*UI必须提供对话式输入框，允许用户用自然语言提出对蓝图的修改意见。*
* **当前支持：** **部分支持。** 前端已经有对话输入框，用户可以输入任何文字，包括修改建议。这其实可以承载蓝图修改的NL请求。
* **但：** 系统不具备理解“这是对蓝图的修改要求”并据此更新蓝图的自动机制。当前，用户反馈的自然语言只是传回AI，由AI自行决定如何处理。所以**支持程度取决于AI**：如果AI懂得根据NL修改Mermaid，它能工作，否则系统没特殊帮助。
* **说明：** 目前至少UI有输入框供自然语言输入，但没有专门的“修改蓝图”按钮或命令，需要AI逻辑配合。
* **FR-2.4 用户确认蓝图后才能继续**：*流程必须在用户明确点击“Confirm Blueprint”后才能继续。*
* **当前支持：** **未实现。** 目前没有“确认蓝图”这一步骤。整个流程没有明确的停留等待用户确认设计环节。用户反馈后AI立刻就可能继续，或者用户压根没有蓝图阶段。
* **说明：** 需要在UI/流程中引入显式确认动作。目前无此交互设计。
* **FR-3.1 执行阶段测试先行**：*用户批准蓝图后，系统必须先生成关键单元测试用例（在可编辑表格中），待用户点击“Approve Tests & Generate Code”审批后再继续。*
* **当前支持：** **未实现。** 没有自动生成测试用例呈现给用户的功能。AI不会在拿到反馈后专门停下来出测试列表供用户批注。UI也无表格组件来编辑/批准测试。
* **说明：** 虽然用户自己可以用run\_command执行测试并将结果作为反馈，但这不是AI生成的测试用例流程，也不在UI表格呈现。当前实现离这个需求有较大距离。
* **FR-3.2 严格根据蓝图和测试生成代码**：*用户批准测试后，系统必须严格依据已批准的蓝图和测试用例，在Cursor聊天框中生成实现代码。*
* **当前支持：** **部分支持。**
  + 实现代码的生成实际上由AI在Cursor聊天中完成，而MCP服务器的作用是将用户的确认/反馈提供给AI。现在系统确实将用户反馈（包括测试想法）返回给AI[[179]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L279-L288)[[182]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L283-L291)。如果用户在反馈中强调“按照之前设计和这些测试写代码”，AI就会倾向遵循。
  + 但**“严格依据”**无法由工具程序强制。当前没有机制检查AI输出是否吻合蓝图和测试，也没有拦截AI输出不符时纠正的功能。
* **说明：** 目前阶段，这更像AI自身的行为准则，不是软件功能。Wizard提供了蓝图和测试给AI（假设前面步骤实现），之后只能信赖AI遵循。除非实现FR-4.x的验证，否则工具不干预代码生成过程。
* **FR-4.1 验证阶段代码溯源蓝图**：*生成代码后，系统必须自动分析生成的代码，逆向生成新的DAAS图。*
* **当前支持：** **未实现。** 系统没有任何代码解析或逆向建模功能。目前MCP服务器并不读取项目代码内容来生成图表。
* **说明：** 这项需要全新开发。当前只在get\_system\_info提供基础环境信息[[16]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L562-L571)，没有针对项目代码结构的分析接口。
* **FR-4.2 蓝图对比验证**：*系统必须并排展示规划蓝图与代码溯源蓝图，供用户最终比对验证。*
* **当前支持：** **未实现。** 没有此UI。现有UI没有比较两个图的界面，也没有存储两个版本图的逻辑。
* **说明：** 要实现需在UI增加对比视图。目前完全缺失。
* **FR-4.3 用户点击“Accept Code”后流程才算完成**：
* **当前支持：** **未实现。** 目前没有“Accept Code”按钮或概念。会话结束条件只是AI不再调用或session超时/用户取消。
* **说明：** 没有专门记录用户接受代码的确认动作，也就无法计算完成率等指标。
* **V1范围内其他**：
* *支持流程中返回上一步修改，会话内版本控制*： **不支持。** 当前一旦进入下一阶段，没有内置功能跳回前一步，也没有UI让用户查看各阶段版本区别。要回退只能用户手动再描述，AI再呈现（没有自动“上一阶段”按钮）。
* *基于配置文件的可插拔引导框架*： **未实现。** 如上所述，目前没有读取一个JSON定义流程然后执行的机制。流程是硬编码在AI逻辑和MCP交互里的。
* *AI无法完成时的优雅降级*： **部分支持。** 如果AI输出“无法做到”，目前Wizard并无特殊处理，只是当普通回答对待。但Wizard的错误处理框架会抓工具自身的失败。严格来说，这个降级逻辑更多是在AI层（模型应该在不能完成时给出合理回复）。MCP这边并没有判断AI内容“无法完成”然后执行某种fallback，所以算未体现。

综合来看，**当前代码对PRD描绘的大部分特性支持不足**。现有实现只涵盖了“AI与用户互动收集反馈”这一通用功能，离“规格先行”的引导式多阶段流程还有明显差距。尤其是蓝图生成/编辑、测试用例审批、代码生成验证这些核心步骤都需要新增功能。

下面我们根据这些差距，提出高层的设计方案，说明如何在现有代码架构基础上实现PRD要求的各个功能。

## 缺失功能的实现方案

针对上述每项缺失或部分支持的功能，我们规划如下设计思路：

**1. 模式选择界面 (FR-1.1, FR-1.2)：** 在用户第一次触发TheWizard时，提供一个模式选择UI。 - *后端触发：* 当AI调用interactive\_feedback时，可以约定当summary为特定值（或增加一个参数）表示请求模式选择。例如AI可以调用interactive\_feedback(summary="\*\*MODE\_SELECTION\*\*")作为特殊标志。后端检测到这个标志，不创建普通会话，而是创建一个“模式选择会话”。 - *前端呈现：* 新增一个模式选择弹窗（Modal）。在index.html或feedback.html里，如果发现当前会话是模式选择状态，就显示包含四个按钮（分别标注快速脚本、探索原型、测试驱动、规格先行）的界面。用户点击某一按钮后，通过WebSocket发送一条例如{"type": "mode\_selected", "mode": "spec\_first"}的消息。 - *后端处理：* 在handle\_websocket\_message新增elif message\_type == "mode\_selected": ...[[97]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L618-L626)。后端收到后： - 记录选择的模式，可能保存在WebUIManager或Session对象中（如session.selected\_mode = "spec\_first"）。同时我们可以考虑立即结束模式选择会话，然后触发后续步骤。 - 将这个选择反馈给AI：有两种做法：其一，后端结束interactive\_feedback调用并返回一个标识choice的结果给AI。可以返回一个特殊的TextContent，比如 "MODE\_CHOSEN: spec\_first"。AI模型接收到后就知道用户选了规格先行，于是接下来按照该模式流程进行（比如先生成蓝图）。其二，完全不通过AI，由前端直接进入规格先行UI，让AI被动等待。这不太合适，因为AI需要知道模式以调整行为。所以前一种通过AI比较合理。 - 清理模式选择会话：模式选一次就用完，可以clear\_current\_session()。接下来AI会进行下一步调用（如准备蓝图），我们重新走普通会话流程。 - *AI协作：* 需要在AI的对话脚本里加入逻辑：当触发TheWizard时，先调用模式选择，让用户选模式；拿到用户选择后，如果是规格先行，就接着按照规格先行的子流程执行（调用蓝图生成工具等）。这部分需要Prompt编写和AI编程，但对MCP服务器来说，只需提供模式选择结果。 - *默认模式问题：* 如果希望降低初次使用复杂度，也可以默认选中某模式（如规格先行）并跳过界面，但PRD明确要四象限选择，所以应该实现UI。我们可以增加配置项允许开发者隐藏某些模式。

**2. 规格先行工作流多阶段实现：**（涵盖 FR-2.x, 3.x, 4.x） 按照PRD定义，规格先行开发模式包含**RIPER-5**的五阶段：Read, Insight, Plan, Execute, Refine[[171]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L19-L27)。对应我们UI： - **Read/Insight 阶段**: 其实就是模式选择→AI根据用户选的模式理解任务模式，准备进入Plan阶段。 - **Plan 阶段 (FR-2.1~2.4)**: 核心是**蓝图设计**。 - *AI生成蓝图*: 用户选了规格先行后，AI应该调用interactive\_feedback再次，但是这次summary应该包含它对功能的理解和**提议的设计蓝图**。在Prompt规则上，可以要求AI：“请输出Mermaid UML格式的时序图和类图作为蓝图，并调用工具展示给用户确认”。AI可能会把Mermaid文本放入summary（或者未来MCP或Wizard支持结构化传输图表）。 - *前端显示蓝图*: 我们需要渲染AI提供的Mermaid图。实现步骤： 1. 将Mermaid源码从AI的summary中提取。如果AI的回答是Markdown格式（带 mermaid ... 代码块），我们前端可以解析出其中的Mermaid文本。AI也可能以某种JSON结构传递图的定义，这需要事先约定。简单方案是把Mermaid文本直接作为summary字符串发送（AI在字符串里写Mermaid语法）。 2. 前端引入Mermaid库（可以通过在static/js中包含mermaid.min.js）。在feedback.html新增一个<div id="diagram-area">用于显示图。再新增一个<textarea id="diagram-src">显示Mermaid源代码。通过CSS让它们左右排列（双栏）。 3. 编写blueprint-manager.js：初始化时，从页面变量中获取Mermaid源码，调用mermaid.render进行渲染，将生成的SVG插入diagram-area。然后绑定textarea的输入事件：每当用户修改Mermaid源，设个短延时（比如500ms）重新渲染预览。这样用户看到改动即时反映。 4. 当AI第一次发送蓝图时，我们应该暂停AI流程，等待用户确认。可以将Session状态设为一个新状态如PLANNING。UI根据session.status==planning，显示“Confirm Blueprint”按钮且不显示发送反馈按钮（防止用户用普通反馈绕过确认）。实际上，我们可以仍用submit\_feedback机制，但将确认本身作为反馈内容的一部分，如 feedback = "*BLUEPRINT\_CONFIRMED*". 但更清晰的是定义一个新的消息类型confirm\_blueprint。 5. 用户可能也不满意蓝图，用两种方式调整：一是手动编辑Mermaid源码，这些改动前端已经让他做；二是用自然语言要求修改。对于后者，用户可以在对话输入框再说一些，比如“请把类图里的类A改名为B”。我们要支持再次让AI更新蓝图： - 用户点击“Submit Feedback”来发出自然语言意见。前端需要把用户的NL反馈和当前Mermaid源码一并发送，这样AI能知道上下文。可以构造一个复合消息：文本部分是用户自然语言，附带一个字段包含当前蓝图源码。比如：

{  
 "type": "submit\_feedback",  
 "feedback": "请把类A改名为B",  
 "blueprint": "<mermaid code here>"  
}

在handle\_websocket\_message处理时，如果检测到有blueprint字段且当前处于Plan阶段，可以将其合并到session.summary或存到session让AI获取。由于AI上下文里本就有之前发给它的蓝图，它可能也不需要我们明确传回Mermaid，因为AI自己有记忆。但保险起见，可以把编辑后的Mermaid也传给AI，让它对比。 - AI收到用户新的NL反馈后，会修改Mermaid并再调用工具（又一次Plan阶段调用）。后端WebUIManager由于已有session，检测有活跃WS连接，于是只发session\_updated来更新图，不新建窗口[[146]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L62-L70)。前端在handle session\_updated时应更新diagram源为新Mermaid并重新渲染。 - 如此循环直到用户满意并按下Confirm Blueprint。 6. 用户按“Confirm Blueprint”：前端发送{"type": "confirm\_blueprint", "blueprint": "<final mermaid>"}（也可不发送mermaid，因为后端session有最终版，但发送可双保险）。后端收到后： - 将session.status从PLANNING改为PLANNED（或ACTIVE准备下一阶段）。 - 触发Event让AI继续：可以直接set feedback\_completed事件，使interactive\_feedback`返回结果给AI。这次返回的内容可以是：我们希望AI知道用户确认了。可以简单返回一个TextContent如“[Blueprint Confirmed]”，加上最终蓝图文本（这可能超长，不宜全部放返回文本——模型上下文中已经有图，所以没必要重复）。也可以什么都不返回，直接结束工具调用。这里选前者可能更保险，让AI明确收到确认信号。AI逻辑见下一个阶段。 - *存储蓝图用于后续比较*：当用户确认时，我们应当将最终蓝图Mermaid源码保存下来，存入Session对象（如session.blueprint\_confirmed = mermaid\_text）以备Refine阶段对比使用。

* **Execute阶段 (FR-3.1, FR-3.2)**: 代码实现上分为“生成测试用例子阶段”和“生成代码子阶段”。
* *生成测试用例 (Test sub-stage)*：
  1. **AI生成测试**：在AI收到蓝图确认后，它按照RIPER-5流程应该进入Execute前的测试编写。AI调用interactive\_feedback再次，请求用户审批测试。调用时summary可能包含它拟定的测试用例清单或测试代码片段。我们需要解析并呈现。
  2. **UI呈现测试**：类似蓝图，我们新增一个UI组件“测试用例表格/列表”。考虑简单起见，用Markdown列出测试项或者用一个textarea显示测试代码。
  3. 如果想要结构化，可以把测试列成表格：每行一个测试名称、说明、是否通过(为空)。但让用户编辑表格比较麻烦，可能textarea让用户自由改会更直观。
  4. 假设AI提供的是测试代码片段（比如用伪代码或特定语言写的单元测试函数）。我们可以将其放在一个<pre><code>区域让用户看。如果需要编辑，可以把它放入一个文本区域允许修改。
  5. 主要加入UI元素：例如一个<div id="test-cases">容器，其中每个测试案例可以是<div class="test-case">内含描述。初版不用复杂表格，把AI的输出直接显示，然后在旁边放“Approve Tests”按钮。
  6. **反馈修改**：用户如果对测试不满意，可以：
  7. 直接编辑测试文本（比如增加一个case）。UI若是textarea，就可以编辑。如果我们希望结构化操作，也可以允许删除/勾选Case但时间有限也许不做太细，只提供整体编辑。
  8. 或用自然语言再让AI改（如“再加一个边界条件的测试”）。同蓝图阶段，用户可以在输入框提意见，然后AI再循环一次生成更新的测试列表，UI刷新显示。
  9. **确认生成代码**：当用户对测试满意，点击“Approve Tests & Generate Code”按钮。
  10. 前端发送{"type": "approve\_tests", "tests": "<maybe edited tests>"}。后端处理类似confirm\_blueprint：
      + 将最终测试内容存储（session.tests\_confirmed）。
      + 结束等待，唤醒AI。interactive\_feedback返回一个标志，如“[Tests Approved]”文本给AI。
  11. 之后AI会看到用户接受了测试，让它“严格依据蓝图和测试生成代码”[[235]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L12-L15)。AI就会退出MCP交互，直接在Cursor对话框输出代码，实现FR-3.2。此时Wizard的交互暂时告一段落，等待AI产出代码后再进入Refine。
* *技术要点*：
  + 如果我们不想定义新消息类型，也可直接在Confirm按钮的onClick把测试文本当作普通feedback发出，比如用户feedback：“Tests are okay.” 但最好明确定义类型，方便在AI prompt解析。AI那边可以设置：收到“[Tests Approved]”就开始写代码。
  + 测试文本可能较长。如果AI写了很多测试代码，用WebSocket传输没问题，但要注意前端textarea能否承载格式高亮（可能不需要高亮，纯文本即可）。
  + 可以提供“运行测试”按钮在此阶段，调用session.run\_command("pytest")在UI显示执行结果，以辅助用户决定是否接受测试。不过这取决于有可运行的代码，可能要在代码生成后才真正跑通。所以暂时可以不做自动运行。
  + Approve Tests按钮按下后，UI应切换到只读状态，等待AI给出代码结果。
* **Refine阶段 (FR-4.1~4.3)**: 验证生成代码与蓝图一致性。
* *触发*: 当AI输出代码并结束对话（或AI调用某个MCP指令来启动Refine），我们需要启动Refine UI。两种可能：
  1. **AI驱动**：AI在输出代码后，主动调用interactive\_feedback最后一次，可能带上“请验证代码与设计一致性”。若如此，我们Back-end就进入Refine会话。
  2. **用户驱动**：AI输出代码后对话结束。这时让用户主动点击IDE里比如一个“Verify Design”按钮触发Wizard refine功能。这可能通过Cursor的插件或命令再调用Wizard的MCP工具来分析代码。
* *代码分析生成蓝图*：
  + **方案A: AI生成** – 让AI自己解析代码，输出Mermaid图，再通过MCP给用户。这个需要AI再次调用工具并传图给UI。流程上，AI可能说“我生成了以下实现对应的图，请确认”，然后MCP像Plan阶段一样显示图。
  + 实现上，我们可以将最后一次interactive\_feedback用于Refine：AI提供summary包含Mermaid图（称为code\_blueprint）。Wizard前端把它渲染在一个对比界面。
  + 这种方法容易实现但有风险：如果AI写的代码有误，它自己画的图未必真实，也可能复制原图敷衍。为了保证可信，我们更倾向于工具自身解析代码。
  + **方案B: 工具生成** – Wizard读取用户项目代码，利用静态分析或现有库生成架构图。
  + 类图：可以基于源码解析类定义和关系。不同语言差异大，短期可考虑支持主流OOP语言(Java, Python)的class diagram。比如Python可用ast模块遍历类和函数定义，再用mermaid语法画出类及继承关系。时序图：难度更高，需要分析函数调用流程，这可能不容易静态完成。或简化为仅列出模块间调用关系。
  + 另一折衷：使用另一种AI工具。比如Wizard可以在后台调用一个大模型Prompt：“读这些代码，输出类图Mermaid”。这属于链式调用，MCP可以实现（Wizard也可作为客户端去调OpenAI API）。但这增加复杂度，也跟AI本身的可靠性挂钩。
  + V1时间有限，可能**暂不实现复杂代码解析**，而采取方案A或者一个简化策略：例如，不画时序图，只画类/模块依赖图，用简单方法提取文件和函数。
  + Considering the vision, PRD想要对比设计和实现，我们至少应给用户某种可视化。也许一开始不要求100%精确，只要用户看到**“左边规划图 vs 右边实现图”**直观比较就够了。即使右侧简单也行。
  + **实现**：
  + UI: 扩展Blueprint双栏区域成四格或上下两栏：一栏显示规划蓝图（Mermaid源码我们之前存了session.blueprint\_confirmed），另一栏显示代码蓝图（Mermaid生成后渲染）。我们可以复用mermaid渲染模块，两次调用render，一个渲染规划，一个渲染实现。
  + 在session中保留之前的蓝图源码，用于对比绘制。
  + 如果采用AI生成方案A：Refine阶段AI调用工具带Mermaid，我们在handle\_websocket\_message遇到新的type例如session\_refine（或普通submit\_feedback但session.status=REFINE），就拿到AI传的Mermaid字符串，把它存在session.generated\_blueprint，然后前端通过session\_updated获取并绘制。
  + 如果采用工具方案B：Refine会话开始时，不等待AI给图，而是Wizard自己读文件，生成Mermaid字符串，然后直接通过WebSocket发给前端（因为Refine开始时前端已连接，可以server主动推送消息，如类型code\_blueprint). 也可以在HTTP层，前端调用一个API如/api/generate-code-diagram获取图，然后JS渲染。为了一致性，用WS推送比较好，UI无需主动拉取。
  + Code diagram的Mermaid可能比较简单，不要细节过多。比如类图Mermaid:
  + classDiagram  
     class UserService {  
     +method1()  
     }  
     class UserController {  
     +methodA()  
     }  
     UserController --> UserService : calls
  + 展示调用关系。
  + 对于大型项目，我们可能只关注本次功能相关的类，哪些类？ 也许AI把涉及的类文件名告诉我们。一个简单heuristic: 比对实现前后git diff，拿出修改的文件涉及的类函数进行图生成。不过V1可不深入，假设AI在输出代码后会在prompt结束语指出主要改动模块，我们用那个信息。
* *用户对比与确认*：
  + UI将两个图并列显示（或上下显示）。可以加滚动同步方便用户比对，或只是粗略查看。
  + 用户若发现不一致，有两个可能动作：要么接受差异（可能设计变更是合理的），要么要求AI修改代码或设计。理论上可以再次发反馈要求修正实现以符合设计（或更新设计）。PRD未明确提支持修改，但原则上Refine后用户还能Reject code。这种循环或超出V1范围，但我们可以提示用户可以重新启动Wizard调整。
  + 提供“Accept Code”按钮。用户点击即意味着对最终代码满意。
  + 前端发送{"type": "accept\_code"}消息。后端:
    - 记录会话完成：可以在log输出Session completed successfully，并更新一些统计计数。
    - 调用clear\_current\_session()关闭会话[[236]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L422-L430)。这会cleanup资源并停用服务器或不立即停？Minidoracat设计单个服务进程可处理多次会话，不建议每次都Kill。可以持续运行等待下次请求。
    - 可能通知AI。但AI可能早已停止等待，因为Refine阶段可能是最后一个interactive\_feedback调用。Accept Code更多是给用户和统计用，不需要反馈AI。所以我们可以不做特殊返回，只是在UI上显示“Workflow completed!”。如果AI依然在等，我们也可以像之前一样给它一个确认信号，然后AI结束对话说“代码已通过验证”。
  + UI上，接受后可以自动关闭Wizard浏览器tab（或在界面上显示可以关闭窗口的信息）。由于Wizard是独立窗口，用户自己可以关。如果想自动，JS可调用window.close()（需要是用户触发事件才允许关闭自身窗体，大概Accept Code点击算用户事件，可以关闭）。
  + 如果用户迟迟不Accept，也不Reject，就放着。SessionCleanupManager或auto\_cleanup过些时间会关闭会话。没有Accept这次就不会计入Happy Path成功率（所以日志里那次session没有Success标记）。
* *Accept Code后的处理*:
  + 统计: 在Accept Code handler里我们增一计数器WizardSuccessCount++等，或者写入文件。这个用于后续计算完成率。
  + UI: 可能给个对话框“恭喜代码验收完成！”然后才关闭。
  + 如果用户不Accept就关掉窗口，则我们算未成功完成，在SessionCleanupReason会是EXPIRED或MANUAL。如果算核心指标，后续要区分这些。

**3. 返回上一步功能：** PRD希望在蓝图和测试阶段允许返回修改前一步。实现难度较大，但可以有简易方案： - 在测试阶段，提供一个“Back to Blueprint”按钮：按下时，前端发送{"type": "rollback\_phase", "target": "plan"}。后端： - 将当前测试会话的数据保留（以防回来还用），但UI要切换回蓝图编辑模式。 - 可以重新渲染上次确认的蓝图，让用户修改。并需通知AI。AI当时已经到了测试阶段，如何让它回退？只有一种：我们把会话状态改回Plan，用户改完蓝图后，再次Confirm Blueprint-> 那后面的AI逻辑就需要重新生成测试。相当于一个新循环。 - 这相当复杂，因为AI不容易“忘记”它写的测试。或许AI本身可以支持：用户说“我们改了蓝图，你重写测试”，AI可以做到。但为了简单，我们或许在V1不做“Back”按钮自动化流程，而是指导用户手动：比如UI上如果用户希望改设计，可以关闭当前流程或完成后重开Wizard做调整。这属于减Scope。 - 同理，在Refine阶段若发现不一致，可提供“Reopen Wizard to Refine Code”按钮：点击后可能启动一个新的Wizard会话针对差异，或提醒用户重新启动规格先行过程针对不一致的地方。 - 由于时间有限，**建议V1不深究返回上一步**，而将此文档列为风险/未来优化点。我们可以实现简单的“回退”触发，实际效果需要AI配合。

**4. 日志和指标收集：** - 增强日志：在每个阶段Mark：例如当蓝图确认时log "PLAN\_CONFIRMED"，测试确认时log "TEST\_CONFIRMED", 接受代码时log "ACCEPTED". 这样后期统计可以grep日志统计成功率。 - 统计存储：也可引入一个轻量数据库或csv：每次Session结束写一行session\_id, mode, completed(0/1), durations...。V1可暂时简单处理，用日志或计数文件。

**涉及模块修改：** - *前端*: 修改feedback.html，添加： - 模式选择modal及JS事件处理 - 蓝图编辑器区域（textarea + preview容器 + Confirm按钮） - 测试用例区域（容器 + Approve按钮） - 对比蓝图区域（规划 vs 实现）+ Accept按钮 - 相应的CSS样式，使这些元素在不同状态下显示/隐藏 - 在app.js或新模块中，根据session状态控制哪些区域可见。例如session.status=="planning"显示蓝图区域，隐藏输入框；status=="testing"显示测试区域等。 - *JS逻辑*: 新增 blueprint-manager.js, test-manager.js, refine-manager.js 等，或把简单逻辑写在app.js切换。 - *后端*: - WebUIManager 增加对阶段状态的管理，也许introduce SessionPhase枚举。或者重用SessionStatus拓展值如 "planning", "testing", "refine". - WebFeedbackSession 增加字段 blueprint\_code, test\_code 等存储最终确认的设计信息。 - handle\_websocket\_message 增加 case: "mode\_selected", "confirm\_blueprint", "approve\_tests", "accept\_code", 以及可能的 "rollback\_phase". - interactive\_feedback 流程中，根据当前模式和阶段做不同处理： \* 可能引入一个 mode参数传递（通过FastMCP invocation上下文，复杂）。 \* 或更直接，在 WebUIManager 上存当前模式和阶段，interactive\_feedback 每次调用先检查：如果当前模式==spec\_first: - 如果没有current\_session或current\_session.completed: 说明是新会话开始，第一步AI likely sending blueprint -> 标记session.status=PLANNING. - 如果session.status=PLANNING并且用户还没confirm就AI又调用（这种情况应该不会发生，AI应等confirm）。 - 如果session.status上次被置为PLANNED (blueprint done) : AI应该正在发送tests -> set session.status=TESTING. - 这些逻辑可能通过AI传递的内容或一个phase参数来判断。 \* 这个需要AI和工具间约定明确的phase transition。或通过不同 interactive\_feedback 工具函数分开实现每步，但MCP设计一个tool搞多个子流程也是可以的。 - 代码解析（Refine阶段）功能：可能创建 web/utils/code\_analyzer.py, 提供 parse\_project\_to\_mermaid(project\_dir) -> str。先实现简单类图。 - ErrorHandler可能需细化或增加特定错误，比如Mermaid解析失败错误，可提示用户图语法错。 - SessionCleanupManager：refactor其使用，如果长期会话，在Refine accept后立即清理session，而不是等超时。（Accept Code可以直接clear session）。

**5. 与当前设计的差异及冲突：**

上述方案在实现时，需要注意与现有架构的冲突并作相应调整：

* **会话生命周期**：当前设计假定每次interactive\_feedback调用对应一个新的WebFeedbackSession，用户反馈后session基本结束等待下次调用更新。然而规格先行流要求一个session跨多个阶段、多次用户反馈，不断演进。我们要**改变session的使用方式**：尽量让一个Session贯穿蓝图和测试阶段，不要每步都换新的Session对象。否则很难在Session里保持整个流程的数据（蓝图、测试）。
* 可能的改进：在规格先行模式下，create\_session不每次都清理旧的，而是继续使用同一个session对象，通过一个阶段字段控制逻辑。比如Session.phase从"planning"到"testing"再到"completed"。这样session.websocket始终不换，一直连着，省去反复转移连接的麻烦。
* 这和Minidoracat单活跃会话理念一致，但是他们的实现还是每次create新session再转移连接[[191]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L338-L346)[[65]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L388-L396)。我们改为一个session多轮交互，需要小心修改wait\_for\_feedback和feedback\_completed的用法：或许改为每阶段有各自的Event。但更简单是保持现状：每AI调用->新session，这样清晰，不破坏fastMCP返回处理。但是为保数据，我们可以把设计蓝图和测试存在WebUIManager中（全局），这样即使session切换，新session也能访问之前数据。
* 这有点Hack，但例如在create\_session时，如果发现上一个session的蓝图已确认且当前AI调用明显是在生成测试，那么可以把旧session的blueprint复制到新session属性上，以供Refine使用。
* 这种跨session数据共享可以通过WebUIManager属性或globalActiveTabs等暂存。需要明确标识当前处于Wizard流程的哪个阶段，也许引入WebUIManager.current\_workflow = {"mode": spec\_first, "blueprint": ..., "tests": ...}帮助协调。
* **AI模型配合**：本方案假定AI能遵循给定协议调用多次MCP，并根据用户确认推进。这需要对AI提示进行详细设计，而且模型行为难以完全保证。存在**风险**：AI可能在蓝图还未确认就把后续内容一并输出，或忘记等用户确认就跳过测试。为降低这种风险，Wizard可以采取一些**保护**：
* 如果AI提前输出代码而非等待测试确认，我们可以检测：比如在Testing阶段session，AI却返回一个长代码段文本而不是调用我们。我们可以通过Cursor侧监控AI输出或者Wizard接收到AI结束调用而UI还在测试确认等待，这种不匹配只能靠用户发现。技术上Wizard很难拦截AI在IDE里的行为。
* 我们能做的是**在Wizard UI强调流程**：如模式选了spec-first，则UI引导用户逐步走，不让他们跳步。AI端要严格按指南调用Tool。这个问题更多偏AI配置而不是代码，但我们在设计时要考虑**容错**，如AI没给测试就要写代码怎么办（Wizard可以在没有测试session就出现Accept Code等，引导用户自己觉得流程不对然后重来）。
* **UI/UX复杂度**：我们增加很多UI步骤，可能使得Wizard交互相对繁琐。为了避免干扰普通快速场景，我们应当**仅在spec-first模式下启用**这些UI组件。其它模式（若暂未实现）可以直接走旧流程，不出现蓝图/测试UI。
* 这需要在前端根据模式变量决定渲染。例如全局有window.currentMode，如果是"spec\_first"则启用相关UI，否则隐藏之。
* 同时，要照顾模式选择阶段UI——第一次模式选择完成后，才加载具体模式的界面。
* **与现有功能整合**：要确保不破坏以前的功能：
* 像Prompt管理、AutoSubmit、Session历史导出这些模块应该可以和平共存。Spec-first UI弹窗在主要区域显示时，仍应该允许用户展开侧边栏看Session历史吗？可以，但用户此时大概不会用那些次要功能，所以问题不大。
* run\_command的功能依旧可用。甚至在蓝图阶段用户也许能执行命令（尽管没有代码可跑）。我们可以控制：在Plan阶段禁用命令执行按钮，在Testing阶段可以启用比如“Run All Tests”按钮。防止用户在不恰当阶段按某些按钮。
* **性能**：生成并渲染图表会稍耗时间。Mermaid渲染大图或大量测试文字可能卡UI。我们应注意**不要阻塞主线程**：Mermaid本身在浏览器中渲染SVG可能大图会慢，但应该尚可。必要时，可分批渲染或提示用户稍等。我们可以在渲染前显示一个小Spinner动画提示。
* 还有，如果AI生成的Mermaid有错误，会导致渲染崩溃或报错。需要catch异常，前端显示“图表渲染失败，请检查Mermaid语法”。并可以在编辑器下方标红错误。这要求Mermaid提供error信息，我们可try-catch mermaid.init调用来实现。
* 运行shell命令大量输出在UI实时显示也可能性能问题，但Minidoracat已经考虑通过逐行读取输出[[237]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L710-L718)[[177]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L715-L723)和使用异步，不会冻住UI。我们加入自动跑测试功能时注意别一次输出几万行。
* **Backward兼容**：我们应该确保如果用户不用Spec-First模式，一切仍按旧方式工作。因此**模式选择可选**：用户若通过老方式调用Wizard（AI直接调用feedback工具不弹模式选），Wizard可以默认进类似老的交互，不多问就一个循环结束。我们可以将模式选择看成Wizard ChatGPT插件的新触发；如果老用户直接让AI调用feedbackwith summary tasks,Wizard可以识别lack mode, 继续普通loop。这样不吓到不需要复杂流程的用户。
* **Testing**：实现这些功能需要非常细致测试，不然Flow容易中断。我们后面详述测试计划。

综上，为实现PRD V1需求，我们将在当前代码框架下： - 扩展前端UI以支持蓝图绘制/编辑、测试查看/编辑、最终对比等。 - 利用WebSocket机制增加新的消息类型，以在关键节点（确认蓝图/测试/最终确认）同步用户操作，并驱动AI流程。 - 在后端维护更多状态，让一个Wizard模式的交互跨越多个MCP调用依然能共享数据和状态。 - 增强AI和工具的协同，通过明确的协议让AI知道何时等待用户。 - 确保所有新增步骤符合三个产品原则：**用户控制**（每步都有确认按钮）、**过程透明**（用户能看到AI想要做的蓝图、测试，一目了然[[238]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L140-L149)）、**最小上下文切换**（尽可能集成在IDE或一个窗口内，不让用户来回切应用）。

虽然改动较大，但所幸架构已有所预留（单会话持久UI、模块化UI），预期可在此基础上实现这些功能。

## 新功能的技术风险评估

为引入上述新特性，我们需要评估潜在的技术风险和对系统产生的影响：

* **AI行为不可控风险**：规格先行流程需要AI严格按照步骤执行，但现实中模型可能偏离预期。例如AI可能在蓝图未确认前提前生成代码。这会打乱流程。我们可以在Prompt上强约束，但仍有风险。如果模型频繁不遵守协议，会让整个体验混乱甚至失败[[192]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L440-L448)[[193]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L442-L450)。这个风险需要通过大量Prompt调优和测试缓解。此外，我们的后端需设计得尽量宽容。例如如果AI跳步，我们UI提示用户重新开始或提供纠正措施，避免系统完全失效。
* **实现复杂度与稳定性**：我们增加了多阶段状态机，这大大提高了系统复杂性。新的分支、新的消息类型如果处理不周，可能引入bug导致死锁（如feedback\_completed事件用不好会导致AI端一直等）。必须小心理清每阶段的开始、等待和结束条件，确保不会漏唤醒或重复唤醒事件。特别注意Session的事件对象不能重复使用，需要每次fresh或者在不同阶段重置。
* **前端性能与兼容**：Mermaid渲染复杂图表可能较慢，尤其类图如果元素很多SVG会很大，导致浏览器卡顿。另外，前端需要兼容不同屏幕尺寸，双栏/四栏布局在小屏幕IDE窗口中是否可用？如果开发者把IDE窗口开半屏，能看清蓝图和代码对比吗？这些需要权衡。如果布局过于拥挤，用户体验不好。这不是功能错误但影响满意度。我们可考虑蓝图/对比图支持全屏查看（点一个放大按钮可以新窗口打开图），以防止UI空间不足的风险。
* **文件解析可靠性**（Refine阶段）：如果我们实现代码自动分析，必须考虑不同语言、代码风格。写通用解析很难，草率实现可能给出错误的实现图，从而使用户对比时反而产生误导，损害信任。这个风险挺高，所以可能V1并不实现深入解析，只生成粗粒度图。接受精度有限，用人工对比为主。这样做减小风险但偏离“自动验证”一些。需要在文档或UI说明“请人工核对”。
* **并发冲突**：开发者如果意外地同时打开两个Wizard会话（比如对同个项目并行启动两个spec-first流程），目前架构不支持多并发session[[41]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L10-L13)。有风险：globalActiveTabs机制可能混淆两个UI。我们可以假定用户不会这么做，但不排除可能性。一次只允许一个Wizard，否则后果未知。这个可以在UI上限制：如果已有活动会话正在Wizard流程，当用户试图再启一个时，弹警告或忽略第二个请求。
* **Backward兼容**：老的使用方式（比如只是调用feedback要求一个quick fix建议）在我们修改后仍需工作。不小心可能Break：
* 例如我们引入模式选择，万一用户**没有**通过模式选而AI直接进入spec-first prompt，我们后端会不会卡住? 需确保：如果AI没走模式选直接发需求summary，我们Wizard可以识别“没选模式那就当默认fast模式”直接出建议，不要强行弹模式UI然后AI端却不认识。我们可以以**向后兼容**原则：模式选择仅当AI明确请求，否則Wizard假定传统单反馈模式。不让无模式的调用陷入等待。
* 另一兼容性是Desktop Mode：我们大部分新功能专注Web UI，对桌面应用UI改动较少。桌面模式下Mermaid渲染也可以，因为本质也是浏览器环境。但UI交互如窗口关闭/尺寸不够等要测试。最坏情况，可在MCP\_DESKTOP\_MODE下禁用复杂模式：例如直接提示“规格先行模式请使用Web界面”或强制fallback浏览器。这牺牲桌面统一性，但保证不出错。风险在于如不处理好，桌面模式运行spec-first可能UI错乱或体验差，因为Tauri窗口可能更小不方便多栏显示。
* **调试难度**：引入多阶段，调试涉及AI→MCP→UI→AI闭环，较难reproduce问题。例如若AI没按协议，问题源头在AI。但Wizard开发者调试时看到Wizard停住也许误以为工具出错。这要求我们在日志中清晰标识各方行为。调试难度上升是一风险，不过可通过丰富日志和测试用例降低。
* **工期和复杂度风险**：这是项目管理方面的。实现以上功能改动面广，开发和测试周期长。若赶工上线，可能BUG较多影响用户印象。需要严格优先级管理（见下节路线图），确保最重要的链路先跑通。如果时间不够，一些次要功能（如回退、自动代码对比)可以弱化或推迟，保证核心闭环可靠。否则一旦V1推出但漏洞百出，用户会失去信任，不敢用Wizard，会质疑产品稳定性。
* **用户接受度**：技术上，Wizard变复杂了，用户能否理解并乐于使用？如果我们的UI不够直观，用户可能迷惑而中途放弃，这也会体现在“完成率”指标上。虽非直接技术bug，但UI/UX缺陷也是风险，需要通过用户测试迭代改进。

总之，新功能带来的风险点主要在**AI配合**、**状态管理复杂**和**性能/稳定**方面。我们将在开发过程中着重缓解这些风险，比如： - 频繁对AI试验，确保Prompt稳健； - 设计简明的状态流转逻辑，并尽量重用现有同步机制避免死锁； - 提供必要fallback，例如如果Mermaid渲染失败，至少显示文本而不中断； - Thorough testing（后文计划）来发现问题并修正。

## 架构重构建议

为更好地契合PRD需求和未来扩展，我们可以考虑在实现新功能的同时，适当进行一些架构重构或优化：

* **会话状态管理改进：** 当前SessionStatus只有简单几种，我们可以拓展使其反映工作流阶段。这有助于前后端同步和维护状态。例如增加：
* SessionStatus.PLANNING（规划中），TESTING（测试生成中），REFINING（验证中）。会话流转顺序按照Plan->Testing->Refine->Completed。
* WebUIManager和Session的逻辑要配合调整，不要在中途把会话标记Completed，而是保持Active跨多个步骤。这种重构涉及跳出Minidoracat原本“一问一答一会话”的框架，转换成“一次模式一会话，多次交互”，是重大改变。但胜在概念清晰，易于在代码中检查和控制步骤，不需隐式通过变量传递阶段。
* 当然，也可以不动SessionStatus，而通过Session里独立的phase属性实现，这实现较局部，不影响别处枚举。比如session.phase = "plan"|"test"|"refine"，专用于Wizard模式。但为了日志和统一，修改SessionStatus也无妨。
* 采用明确阶段状态后，我们可以在UI更简单地根据session.status渲染，不需要额外的全局变量同步阶段。
* **拆分Workflow逻辑**：目前所有模式的逻辑混杂在interactive\_feedback和handle\_message中。可考虑面向对象或策略模式：
* 定义一个基类 Workflow，包含方法如 on\_tool\_call(summary), on\_user\_feedback(feedback), get\_next\_prompt() 等，根据不同模式（SpecFirstWorkflow、QuickScriptWorkflow等）实现不同行为。
* WebUIManager根据当前模式实例化对应Workflow对象，并在各事件中委托它处理。例如，当用户confirm blueprint时，SpecFirstWorkflow可以知道下一步要等AI tests，不结束session；当用户feedback tests approved时，它知道流程结束AI要出码等。
* 虽然这有些Over-engineering，但如果我们考虑长期支持多个模式，这是更优雅的方式。V1或可不完全实现，只在内部逻辑上保持模式判断分支集中，方便以后抽象。
* **前后端协议标准化：** 目前前后端靠约定消息type。可以整理成一个小的协议文档或枚举。比如定义：
* const MessageType = {  
   MODE\_SELECTED: "mode\_selected",  
   BLUEPRINT\_CONFIRMED: "confirm\_blueprint",  
   TESTS\_APPROVED: "approve\_tests",  
   FINAL\_ACCEPT: "accept\_code",  
   // ...  
  }
* 并在后端做严格匹配，让不符合的type log warning[[111]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L686-L689)。这种明确协议利于维护，万一以后更多模式或步骤消息也不混乱。
* **UI组件解耦：** 我们加了很多UI元素，可以考虑做成可折叠面板或Tab形式，不同阶段不同tab。例如Plan/Execute/Refine三个tab，在界面顶端一步步亮起。这样结构清晰，也方便收纳——用户切tab看蓝图或看测试列表，而不是一股脑全在一页。
* 实现这需要改较多前端HTML。简化起见，我们也许不做tab，而是单页动态切换。目前Vue/React框架擅长做这种基于状态的UI切换，但我们项目用原生JS，需要手动控制DOM show/hide。
* 重构思路：逐步可以将UI各区域封装，例如用<div data-phase="planning">包住蓝图UI，data-phase="testing"包住测试UI。然后根据当前phase，加CSS类visible到对应div即可。这样比很多独立元素判断更整齐，也易于扩展更多阶段。
* **集成IDE的可能**： 为减少上下文切换(原则三)[[239]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L146-L154), 长期可以考虑将Web UI嵌入Cursor IDE界面。Cursor是否提供iframe或webview容器？如果能，我们可以把Wizard网页当嵌入pane而不是独立窗口。这样Wizard就在IDE内，真正做到不离开编码界面。实现上也许Cursor团队需要配合。暂时我们可以将其作为建议：提供一个选项“MCP\_WEB\_HOST=cursor-internal”就不启动浏览器，由Cursor plugin接管web内容。
* 这暂不在V1范围，不过架构上保持前后端通过HTTP/WS通信，使这个embedding可行。只要Cursor能加载我们http服务URL，就可在IDE内呈现Wizard UI。我们要确保UI不要有跨域问题、适应IDE窗口CSS等。
* **插件市场准备：** 虽V1不做“Guide-Store”，但我们可以有意识地把规格先行模式的一些素材独立出来。比如Mermaid模板或Prompts模板存在配置文件，以后新增不同模式可以添加相应配置而不用改代码。
* 例如，把“Plan阶段要求Mermaid图”作为一种guide定义：或许存成一个JSON:
* {  
   "mode": "spec\_first",  
   "stages": [  
   { "name": "plan", "expect": "mermaid\_diagrams", "confirmation": true },  
   { "name": "test", "expect": "test\_cases", "confirmation": true },  
   { "name": "code", "expect": "none", "confirmation": false }  
   ]  
  }
* 当然这需要完善的框架解析执行。现在只是假想，如能按此做，将来扩展maybe just adding JSON for new mode. V1时间可能不足完全走配置驱动，但可以部分参考：比如embedding stage names etc as constants, not hard-coded strings scattered.
* **模块解耦**： 某些manager类如 PromptManager, SessionManager等其实是客观存在的独立组件。我们大改Wizard核心同时，或许需要**避免改坏这些无关部分**。可以考虑保持它们原封，然后Spec-first新逻辑写在自己的manager或在app.js以条件包裹，不改动现有JS模块过多逻辑。以降低对原功能的影响。

上述重构建议中，有些可以在V1逐步进行（比如SessionStatus扩展、UI按phase组织），有些属于V2及以后规划（Guide配置框架）。在满足当前需求同时，我们倾向于**小步重构**而不是完全推翻： - 例如Session仍用现有类，仅在其上增加所需字段，不重写整个类。 - 复杂的模式逻辑，先用简单if/else实现，然后在未来版本再抽象优化。

通过这些调整，代码将更贴近PRD需求，也更易于后续维护和扩展新模式，为Wizard日后的发展铺平道路。

# C. 问题、风险与后续计划

最后，我们罗列在推进此设计时需要澄清的问题、需要关注的风险，以及推荐的实施步骤和测试计划。

## 待澄清的开放问题

在开始详细实现前，我们建议向产品经理或原开发者确认以下疑点，以避免曲解需求：

1. **AI与工具的交互顺序**：规格先行流程中，各阶段的AI行为是否固定？例如：AI会等待用户确认蓝图再继续吗？还是可能并行给出测试？我们假设按顺序，但希望确认AI的预期用法，以便我们严格按那个顺序要求模型。
2. **Mermaid图由AI产生的格式**：AI给出的蓝图具体以什么形式嵌入MCP请求？期望是Mermaid源码文本，但如何区分时序图和类图？需要协议：比如AI在summary里用特殊分隔标记两个diagram，还是分别调用两次？或传一个JSON包括两段Mermaid文本？我们需要和AI团队定义清楚，便于解析。
3. **用户修改蓝图的方式偏好**：我们的方案提供同时Mermaid源码编辑和NL对话两种途径。是否有必要支持两者？产品上，给高级开发者直接改Mermaid也许足够，自然语言修改可能没那么常用，但又怕有些用户不懂Mermaid。需确认目标用户是否愿意学习简单Mermaid语法（Mermaid并不复杂，对开发者来说容易）。如果大多数能接受手动改图，我们可以把NL修改降级为次要功能；否则需要着重测试AI对NL修改指令的响应，并优化那部分体验。
4. **测试用例展示粒度**：测试可能以文本描述还是实际代码？PRD写“关键单元测试用例”，也许更偏场景描述而非代码实现。我们需要确定：UI表格展示每个测试的名称和期望结果？还是显示测试函数代码片段？不同呈现形式决定UI做表格还是代码框。此需产品明确期望用户在这个阶段做什么（审查逻辑/边界是否覆盖即可，不一定看具体断言代码）。倾向于**文字描述** (“Test1: 当输入为空时应抛异常”) 比显示完整代码更易读。所以最好确认一下要不要显示具体测试代码。
5. **Refine阶段是否一定要自动化**：PRD愿景是自动对比。但如果时间不够或技术挑战大，可否V1范围内允许Refine阶段由用户手动验证，然后点Accept？换言之，如果Refine实在实现不了，我们可能暂不实现FR-4.x，只是在UI上引导用户肉眼对比。要与产品沟通这样是否接受，还是FR-4.x必须做。他们的北极星指标Happy Path Completion Rate似乎只关心用户接受代码与否，而不一定要有自动图对比。
6. **Accept Code的作用**：除了统计目的，这个确认有无进一步动作？比如按下后，Wizard是否应该自动将代码片段保存、运行测试验证一次、还是发pull request？当前理解只是标志结束。如果产品希望Accept Code还能触发比如在IDE里关闭Wizard pane、或者打个“已完成”标签，我们需要协调Cursor端做相应处理。
7. **多模式入口**：V1只做规格先行，但UI上是否仍然显示4个模式选项？还是只给一个？PRD上FR-1.1写提供四种选项，但FR-1.2说聚焦spec-first。可能意味着UI提供4象限按钮，但另外三种如果选了大概走普通反馈或简单实现，不重点完善。需要确认是否需要实现其他模式的基本流转，比如快速脚本可能Wizard上直接让AI给出脚本然后Ask for run or not，这些实现可能很简化也可。或可以先灰掉其他模式按钮告诉用户暂未支持深度功能。
8. **处理AI拒绝或能力不足**：如果AI在某一步回答“我无法生成蓝图”或“此任务过于复杂无法继续”，Wizard应如何响应？PRD说要优雅降级。是不是直接结束流程？还是提示用户这个模式走不通要不要切换其他模式？希望产品明确遇到AI无能为力的情况希望系统怎么做，以便我们设计UI（比如出现一个对话框“AI无法完成该请求，是否退出Wizard？”）。
9. **目标编程语言/框架**：Wizard V1主要面向何种开发? 这影响Refine代码分析。目前假设通用逻辑语言。如果产品说主要用户用Python/JS，那么我们Refine解析可聚焦这些语言提高准确率。若技术栈未知，我们解析就只能很粗糙，否则面面俱到很难。
10. **性能和资源**：有没有限制比如Wizard运行占用内存不应超过多少？Mermaid渲染大图、大量图片base64这些可能消耗内存。Minidoracat在cleanup里引用psutil看RSS[[227]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L784-L792)，或许有针对内存上限触发clean的想法[[220]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L780-L788)。我们想知道对性能的容忍度，以决定diagram解析深度和UI刷新频率等。比如embedding整个项目源码进AI来生成图，可能非常耗Token和时间，可能不行。需取得对效率的期望值。

通过与相关人员讨论以上问题，我们能更明确边界和细节，从而调整设计实现，确保不偏离产品意图。

## 主要技术风险与未知

在实现和部署过程中，我们需重点关注以下技术风险和未知数：

* **AI模型协作风险**：如前所述，AI是否严格按照预期顺序调用工具是未知的。哪怕OpenAI模型经过指令调优，也难保证100%。我们需要准备应对：如果模型没按剧本来，我们系统稳妥退出或提示，不死锁。这个风险贯穿始终，只能通过Prompt测试尽量降低，但无法根除。
* **Mermaid复杂度**：Mermaid绘制大型流程图或复杂类图可能性能下降或出现布局问题。尤其时序图如果序列长，SVG宽度可能超出视窗。未知点在于典型蓝图规模有多大；PRD场景大型复杂代码库的功能，也许图不会太简单。我们可能需要手动调整Mermaid配置（如{"sequence":{"diagramMarginX":50}}之类）来确保可读性。这在开发前很难完全预料，需在实际案例中调校。
* **WebSocket负载**：一次Mermaid源码可能上KB甚至几十KB，传输问题不大。但如果用户项目很大，AI也许生成超长图甚至上百KB文字，WS传输可能慢或分片，不致命但需测试。还有大量测试用例文本亦然。网络不好的情况下，会不会导致反馈超时？幸好Wizard默认timeout较长(600s)，而Mermaid文本量就算1MB也在秒级传完，不会超时。但如果用户网极差WS中途断,Wizard有重连但AI端没有二次等待概念。这类边缘风险需要文档提醒用户保持网络等。
* **用户行为未知**：用户可能会在Wizard界面执行非预期操作，例如：
* 在蓝图没确认时就尝试发送其他反馈（比如乱写一句话按Enter）。我们UI应禁止Enter默认发送（可以改成只有按Confirm按钮才提交Plan阶段反馈），否则模型可能收到不完整信息。
* 用户可能在任何阶段点击侧边的Session历史导出按钮。要确保即便流程未完，也能正确导出历史（或禁止导出未完成的session，以免误导）。
* 用户可能反悔想终止流程。现在可以关闭窗口或点取消。Wizard要正确处理取消，确保AI不一直等。fastmcp的timeout会起作用，但最好我们提供UI“Abort”按钮显式取消并告诉AI停止。未完全明确这在PRD范围吗？心力损耗角度看，也许可以取消。
* 这些用户行为对我们流程是未知的，需尽量想全并测试。同时产品可以给些UX指导比如没有Abort按钮那用户就只能等或关掉, 这可能不是最友好。我们Risks中列出以提醒关注。
* **Refine代码解析准确性**：如前述，静态代码分析很难完美。我们也无法投入巨大精力构造AST解析各语言。使用AI又引入另一个AI,复杂度更高也可能费用不菲。这部分实现度未知，可能最终只实现类图对比。这意味着Refine验证会弱一些，能否达到建立信任的作用存疑。我们需监控用户反馈：如果他们仍不放心代码，需要新的验证方式（比如实际跑测试，或性能分析)。
* **指标收集有效性**：Happy Path Completion Rate要统计分母和分子。我们要区分哪些session属于Spec-First flow（模式判断），且真正用户接受了(完成)和哪些没完成。很多因素导致未完成：用户中途关闭、AI失败、超时等等。我们fear有些session被我们cleanup认为fail但用户后来又重试。统计有一定误差。技术上，我们可以保证每个Accept Code都log，加上session开始log，这就有分母分子。但**如何计算**（比如时间范围、一次聊天含多个spec-first session或1个？) 这些需要定义。
* **与Cursor集成变动**：Wizard基于MCP 2.0标准[[240]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L17-L25)。如果Cursor团队在未来版本改变MCP调用方式，或引入新内置功能，会不会干扰Wizard？例如Cursor也许推出内置简单Refine对比功能，那我们的Refine显得多余甚至冲突。这种不确定性我们只能保持关注Cursor更新。技术上确保我们的server对Cursor请求的兼容性，不依赖特定Cursor实现细节（fastmcp应该屏蔽了大部分differences）。
* **多语言和国际化**：PRD原文中文，但UI/README支持中英繁。我们要考虑新UI文字（如“Confirm Blueprint”, “Approve Tests”等）多语言。前端locales已有框架，我们需要添加对应翻译。若忽略此，会导致界面部分英文部分中文不一致。翻译工作量小但别忘了。为unknown的是是否需要支持更多语言？Minidoracat支持简中、繁中、英文。我们新增词条也要提供三语翻译，否则切换语言可能看到英文占位。时间紧张也许只提供英文+简中更新，繁中稍后补上。
* **音视频等其他媒介**：PRD没提，但万一用户想附截图或录屏证明bug，Wizard支持图片上传已经有。这可能在Spec-first flow某步发生，比如用户在Refine说“看，运行截图”，上传图片。我们的流程并未禁用images，但UI某些阶段没专门位置显示。当前设计images统一在feedback text里引用。若用户在Blueprint stage上传图片（也许想补充设计图草稿），Wizard后端会包含在feedback\_data.images传AI[[181]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L280-L288)。AI未必会用，但技术上没问题。风险是我们UI可能没考虑在Blueprint界面显示缩略图或移除图的功能。如果这样的用例常见，需要增加。但属于边缘暂不优先。
* **测试维度风险**：由于时间所限，我们不可能穷尽每种场景测试，例如各种语言项目。上线后可能暴露Corner cases，需及时响应更新版本。这需要我们架构尽量松耦合，发布修补能快速进行，不要写死难以改。尽量在配置/文档上留后手。

综合来说，我们清楚**最大的不确定性在AI配合**，其次**系统状态复杂性**。我们将在开发周期内通过频繁测试原型，逐步减少未知，留有余量处理没想到的问题。

## 路线图与分阶段计划

为了降低实施风险并逐步验证，我们建议按以下阶段推进开发：

**阶段1：模式选择 & 基础流程** （时间：1周）  
*目标：* 实现模式选择UI和基础的多阶段框架，跑通“选择Spec-First -> AI给出蓝图（哪怕是假数据）-> 用户确认 -> AI继续 -> 用户确认测试 -> AI模拟代码生成”的骨架流程，无需真实Mermaid解析和代码对比。  
*任务：* - 前端：实现模式选择Modal；实现蓝图、测试阶段基本UI框架（可以先不渲染图，用占位符文本，例如“[Diagram here]”、“Test1...TestN”）。 - 后端：handle消息类型mode\_selected, confirm\_blueprint, approve\_tests，使得当用户按按钮后，能通知AI并推进Session状态。使用临时模拟数据：比如AI调用Plan阶段时，Wizard直接填入一段固定Mermaid文本当蓝图返回给前端；测试同理给固定测试列表。 - 串联：无需真实AI，在测试环境手工或写脚本模拟AI -> Wizard -> 用户 -> Wizard -> ...链条，看能否走完。在日志中看到每一步都按预期。 - 验证：在浏览器中手动点击流程（可能需要通过调试接口触发，因为AI未集成真实Mermaid输出，可以临时写个前端按钮“Simulate AI blueprint”来假装AI输出）。确保前端按钮使后端状态切换正确。Focus是UI多阶段切换和事件通知机制完整没问题。

*里程碑*：模式选择出现，选“规格先行”进入蓝图画面，点确认进入测试画面，点确认进入（模拟的）代码完成画面。各阶段UI元素切换正确，WS消息收发顺畅。

**阶段2：蓝图生成与编辑** （时间：1周）  
*目标：* 引入Mermaid渲染和编辑功能，展示真实AI生成的蓝图，并支持用户修改。  
*任务：* - 集成Mermaid.js前端库，完成blueprint-manager.js：能渲染AI提供Mermaid源码为SVG；监测textarea编辑事件进行实时更新。 - 调整AI prompt/逻辑：让AI在Plan阶段调用Wizard时包含Mermaid图。与AI团队合作测试。必要时Wizard可以在AI summary中提取Mermaid代码块。 - 处理渲染错误：Mermaid解析失败时捕获异常，在UI上显示错误提示（比如高亮有错行）。 - 后端可能不需要太多变，因为AI发送什么Wizard就显示什么。不过需要确保AI内容顺利传到前端：如AI summary里Mermaid代码带 ``` 的情况，我们需要strip掉Markdown标记才能渲染。Implement一个小函数处理这一点。 - 测试：用一些简单案例，如要求AI输出2-3个类的关系的Mermaid，看UI渲染如何。手动编辑Mermaid语法，看预览变化是否正确同步。 - 完善Confirm逻辑：编辑后按Confirm Blueprint，看后端拿到最新源码而不是AI原始的。可以在confirm\_blueprint handler把textarea内容随消息发送，这已在设计中。 - Edge测试：测试在WSL/远程下Mermaid能否渲染。这个mostly前端任务，与环境无关，浏览器端渲染Mermaid即可。测试Chrome/Firefox。

*里程碑*：AI输出一个Mermaid时序图/类图字符串，前端能正确显示图像。用户修改一些节点文字，预览更新，点击Confirm，后端日志确认收到修改后的Mermaid文本。

**阶段3：测试用例生成与审批** （时间：1周）  
*目标：* 显示AI建议的测试用例列表，允许用户调整并确认，然后AI继续到代码阶段。  
*任务：* - 前端：完善测试阶段UI。用Markdown或简单列表显示AI输出的测试cases。如AI输出:

1. 输入值为负时，应抛出InvalidArgument异常   
2. 输入值为0时，应返回1

我们前端可将其转成<ul><li>等。若AI给的是代码块（e.g., JUnit tests code），我们也原样显示在<pre>里，让用户review code。 - 若要编辑，简单起见提供一个大文本框包含所有测试文本供编辑。因为让用户逐条edit UI有点过度，直接文本框效率更高。 - 后端：AI如何发送测试？可能文本或代码，Wizard不需要理解，直接呈现即可。Confirm Tests按钮需要类似confirm\_blueprint处理，取textarea内容（或如果不编辑则取AI原文）发回AI。 - 更新Session状态流程：Plan->Testing->(confirmed)complete for code. Possibly interactive\_feedback在Testing调用完会session.wait\_for\_feedback等用户approve。我们的Confirm Tests handler应触发feedback\_completed事件解堵AI[[186]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L483-L490)。 - Ensuring AI prompt: AI应该在Wizard返回Tests Approved后才真正写代码。这个需要测试多轮。Worst case,Wizard假定AI在tests那步invoke interactive\_feedback,我们做好UI,用户approve->Wizard返回,AI继续写代码。 - 测试：实际让AI生成一些测试（可以小问题，比如“写个阶乘函数”蓝图->AI输出2个测试cases)，检查UI显示对。尝试用户修改case文本（比如增加Case3），approve，看AI是否会在编码时体现。 - 如果AI忽略用户改动继续原实现，这也是AI能力问题，但Wizard完成其职责。 - 另需检查假如用户想第2步就skip tests(不approve就close)是否妥善清理session(Idle cleanupcover it after一段时间). - 这个阶段UI interplay比蓝图简单，因为mostly readonly. Nonetheless test on small screens and multiple browsers.

*里程碑*：AI生成的测试用例在UI可见，用户可改动。点击Approve Tests后，模拟AI输出代码。Wizard成功将流程推进到最终。

**阶段4：Refine 验证阶段** （时间：1周）  
*目标：* 实现代码与蓝图对比的可视化验证，并让用户做最终接受。  
*任务：* - 确定实现方式：视讨论结果，决定用AI生成实现图还是Wizard解析。由于时间，倾向AI生成Mermaid实现图: - 调整AI Prompt：在用户批准测试后，AI写完代码，应该再调用Wizard例如: interactive\_feedback(summary="CODE\_DIAGRAM: ```mermaid\nclassDiagram...\n```\n"). Wizard识别 summary 起始含CODE\_DIAGRAM标记，就认为进入Refine阶段。 - WebUIManager.create\_session碰到这个会phase=Refine，store diagram text. - 前端：Refine UI（对比视图）。可以把之前保存的session.blueprint\_confirmed渲染左侧，新收到的Mermaid渲染右侧。用一个refine-manager.js或直接在app.js操作Mermaid两次输出不同容器。 - UI上显示“请比较设计和实现图是否一致”。提供Accept Code按钮。 - If we try partial static analysis: - Implement code\_analyzer.py for one language (maybe Python or Java). For simplicity/time, skip due to risk. - Possibly fallback: if AI未提供diagram，我们至少可以列出改动的文件列表或函数列表给用户看看。 - Handle large diagrams: if too unwieldy, user可横向滚动查看。Mermaid类图会自动布局但是节点太多可能挤。We might need to test on a real example to see performance. - Accept Code按钮：在后端标记Session完成。返回AI一个最后的确认消息（AI可能不需要但我们可以让AI输出一句“我已完成Refine”然后结束）。 - Logging：打日志“Session X completed successfully with AcceptCode”. - 测试：最难模拟，因为需要AI配合。可能用我们自己构造Mermaid diagram for code side feed into Wizard to simulate. Eg: - After tests approved, have our test harness call Wizard with a mermaid of code structure. - See if front-end shows original vs new diagram properly. - Then click Accept Code, check session closed and success logged. - Verify that if diagrams differ obviously, user can spot. This is subjective, but we can test with a scenario intentionally altering one class name in code diagram, see if visible. - Confirm no resource leaks: Accept后server still running but session gone. Possibly test starting a new spec-first session after one completed, ensure it works (global state reset properly).

*里程碑*：在测试环境，可看到左侧原设计图，右侧实现图（即使简单）。用户决定接受，按Accept Code后Wizard显示流程完成信息，后台统计完成。

**阶段5：完善与收尾** （时间：1-2周，包括测试）  
*目标：* 修复前面阶段发现的问题，完善用户体验，以及编写测试用例和文档。  
*任务：* - 回顾阶段1-4里程碑测试中发现的Bug进行修复。 - UI优化： - 布局调整：确保各部分在不同窗口大小下也不重叠或溢出。也许对Mermaid svg加个max-width:100% css之类。让双栏上下栏在窄屏时自动变上下堆叠（flex-wrap）。 - 增加提示文本：如在蓝图编辑区上方加一句说明“您可以直接编辑架构图代码或用中文描述修改”。在测试区上方写“请审查下列测试用例，您可编辑它们”。 - 如果模式选择UI包含未实现模式，可以灰掉3个按钮或加tooltip“敬请期待”。以免用户点了别的进入undefined流程。 - 国际化文本补充：将新增按钮和提示翻译成英文、繁体中文。更新web/locales/en/translation.json等文件。 - 完成单元测试脚本： - 针对后端核心：测试Session状态流转函数、错误处理(ErrorHandler.format\_user\_error)、Mermaid提取函数等。 - 针对协议：模拟WebSocket消息顺序，测试handle\_websocket\_message能按期望改变Session状态和触发Event。可用pytest+Starlette TestClient+websockets库connect本地server进行集成测试。 - 对Mermaid渲染逻辑，无法用pytest跑浏览器渲染，但可以退而其求：测试我们提取Mermaid文本函数，对于各种AI summary格式（带```mermaid、或纯文本block）能正确取出内容。 - 测试server在各种环境标志组合下host/port是否正确选择(这部分Minidoracat已有,我们验证别因改动而break). - 编写文档： - 更新README.md/简体中文.md，加入TheWizard使用说明，重点介绍Spec-First模式的操作步骤、界面说明。比如插入几张示意图（可选）。 - 如果已有Cursor用户指南，也同步更新Spec-First内容。 - 写开发者说明：说明新消息类型和接口，方便未来维护和调优AI Prompt用。 - 性能测试： - 用一个稍大项目试运行Wizard，看内存占用。可以基于Minidoracat提供的MemoryMonitor log看peak RSS[[227]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L784-L792)在各阶段。特别看看Mermaid渲染大图的时候内存有没有暴涨（Mermaid渲染主要在浏览器，占用客户端内存，对server无影响；server内存主要受image base64大小）。 - 人工减缓网络速度 (Chrome devtools调慢)，测试WS big message情况下UI是否照常（应该就是晚一点到，不会错乱）。 - Multi-turn stress: 让AI blueprint revise 5次循环，看Wizard handle不断session\_updated流是否稳定，UI不会堆积过多中间状态出问题。

* 安全检查：
* 确认我们的safe\_parse\_command规则是否需要扩展针对spec-first场景（也许不用）。
* 确认图片等上传在Mermaid preview阶段不会XSS（Mermaid library已知有XSS风评，但DOMPurify integrated? We could parse text ourselves or trust it? Quick fix: ensure .sanitize(true) in mermaid config to sanitize any HTML in diagrams).
* Crisp handling of any user-provided content in UI to avoid injection。Mermaid injection risk比较低，因为Mermaid不应该执行JS，但Mermaid解析器一旦有漏洞可能注入。可考虑upgrade mermaid to latest secure version。

*里程碑*：Wizard V1功能完整，在内部经过全流程测试通过，关键路径稳定运行。单元测试全通过，核心场景集成测试OK。准备发布。

**阶段6：用户测试与优化** （时间：1周）  
*目标：* 通过有限的真实用户试用，发现最后的可用性问题，做些优化。  
*任务：* - 招募几位经验开发者试用Spec-First流程，让他们用自己的一个功能开发场景试一下Wizard。观察他们是否理解界面引导，有无困惑。 - 收集他们的反馈：例如蓝图图表是否易懂、测试呈现方式是否符合习惯、流程是否太慢或太繁琐某一步。 - 根据反馈进行微调UI措辞、交互顺序。 - 特别关注**心力损耗**：Wizard是否真的减少了不确定性or反而让流程更长？如果后者，我们需要考虑减步骤或提供跳过选项（如无需蓝图时可以直写代码？但那违背规格先行模式，可能提供退出Wizard功能）。 - 检查北极星指标：虽然正式上线才有统计意义，但可以通过内部模拟看完成率可能多少。比如10个人里有几人顺利走完Accept Code。如果低，我们需分析原因（AI问题？UI问题？）以改进下版。

*里程碑*：产品认可Wizard V1效果，用户测试未发现阻塞性问题，可以进入正式发布。

以上阶段安排大致6-7周开发+测试时间。如果资源不足，可调整优先级（见风险部分）：例如Refine阶段自动比对可以延后实现，用人工对比代替，以保证核心Plan->Execute闭环按期稳定交付。

## 测试与验证方案

为了确保最终系统的正确性和可靠性，我们制定如下测试和验收计划：

**单元测试：** - *后端单元测试*： - **Session逻辑**：模拟创建session、提交反馈、清理等，验证Session.status和Session数据字段的变化是否符合预期。例如：create\_session后status应WAITING，submit\_feedback后status应FEEDBACK\_SUBMITTED且feedback\_completed event set[[241]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L144-L151)[[185]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L539-L542)。还可以测试多次create\_session旧session转移逻辑[[191]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L338-L346)[[65]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L388-L396)确保没有遗漏边case。 - **WebSocket消息处理**：使用pytest-asyncio模拟一个WebSocket连接（FastAPI提供testclient.websockets），发送mode\_selected、confirm\_blueprint等消息，检查后台对Session和事件的影响。例如，发送mode\_selected(spec\_first)后current\_session.mode应设为spec\_first。发送confirm\_blueprint后session.status应从PLANNING->PLANNED，feedback\_completed应触发。我们甚至可以在测试里patch interactive\_feedback coroutine，使其不真的起服务器线程而直接调用内部逻辑，以验证状态流转，不测试外部I/O。 - **ErrorHandler**：构造一个自定义异常，调用ErrorHandler.format\_user\_error，看返回字符串是否符合期望格式(无技术细节的友好中文/英文)。确保包括error\_id。可以测试log\_error\_with\_context是否把错误写入日志（捕获日志输出）。 - **Utility**：safe\_parse\_command函数给各种输入测试（合法命令、不安全命令、空命令）确保抛异常或返回正确解析列表[[230]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L84-L93)[[231]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L98-L107)。Mermaid提取函数: 准备几段字符串，带/不带 ```mermaid 标记，看看我们提取函数能正确取出Mermaid内容和识别多个diagram的分隔。 - **Code分析（若实现)**：若实现了code\_analyzer.py，可提供示例源码片段，调用解析函数，看输出Mermaid文字是否正确反映类/函数关系。 - *前端单元测试*：由于我们未使用框架，前端测试较难全面。可以借助Jest或简单地手写几个HTML + 调用JS函数的scenario: - Mermaid渲染测试：这需要在带DOM环境运行前端JS。也许可以用Headless Chrome或JSDOM加载feedback.html，注入mermaid CDN，然后调用window.mermaid.render确保不会抛异常。还可检查输出SVG包含某些我们预期的节点名。 - JS模块函数测试：把 blueprint-manager.js 里的函数比如 updateDiagramPreview 抽出测试。或者Simulate: 在JSDOM创建textarea和div元素，调用我们的onTextAreaChange，看div.innerHTML是不是Mermaid生成的svg（这个test复杂，因为Mermaid渲染异步promise，需要复杂环境，不一定值得投入大量精力）。 - 前端更多采用**手工测试**更实际，因为视觉UI的细节和交互难用自动测试完全覆盖。

**集成测试：** - *端到端模拟*： 尽管AI交互部分真实测试困难（模型不可预测），我们可以使用“假AI”来驱动Wizard，验证主要交互通路： 1. 用TestClient启动Wizard server线程。 2. 模拟AI调用模式选择：直接调用await interactive\_feedback(summary="模式选择", ...)，由于Wizard目前无特殊处理该summary，它会正常启动UI服务。我们接着模拟前端WS连接：TestClient.ws\_connect("/ws")获取websocket。 3. 模拟用户点spec-first按钮：test\_ws.send\_json({"type": "mode\_selected", "mode": "spec\_first"})。看Wizard server返回的interactive\_feedback结果：应该包含TextContent "用戶選擇了 spec\_first"之类（视我们实现而定）。AI接收到，即我们测试代码可以检查interactive\_feedback返回。 4. 模拟AI后续调用Plan：await interactive\_feedback(summary="<Mermaid diagram in text>", ...)。WizardUIManager不会创建新session而update current\_session吗？这个需调试。我们可以在TestClient里直接请求GET /feedback页面看是否正确渲染模板有Mermaid文本（但是没有浏览器执行JS所以看不到图SVG，但可确认Mermaid源码出现在HTML或script block里）。 5. 更可靠的：利用Selenium等驱动浏览器真的加载我们UI，然后自动点击。这偏UI测试，我们可以在开发机上手动完成。Continuous integration暂不跑UI交互。 - *典型场景真实运行*： 在实际环境试几个scenario： - **简单正向案例**：例如开发一个求和函数。AI按Spec-first: 给出简单类图(也许无类，只是流程)，用户Confirm；AI给出两三个测试用例，用户Approve；AI输出代码（很短），Wizard对比（蓝图基本空 vs 实现没有类，Refine价值不大但看看UI表现）；用户Accept Code。检查日志有没有错误，session清理是否正常，UI行为体验顺畅。 - **修改蓝图案例**：稍复杂需求，如实现Stack类。AI出蓝图，用户发现少考虑线程安全，用NL要求AI改。AI改出锁机制的类图。用户直接Confirm。确保Wizard支持这个迭代过程（session没有混乱，UI能更新图）。 - **测试不通过案例**：AI根据蓝图生成测试，但用户觉得漏了关键Case，用户直接在测试列表textarea中添加一个Case，然后Approve。AI拿到经修改的测试。然后看看AI代码实现有没有包括那个Case对应逻辑（需要AI聪明配合，此场景难强求，但至少Wizard传过去了）。 - **Refine不一致案例**：我们人为制造不一致，让AI在实现时偏离蓝图（比如蓝图有类A但AI写成类B）。看Refine阶段图对比：用户肉眼是否能发现？Wizard有没有提示？目前不自动提示，需要用户自己看。这个测试更多是评估UX：看看一般人能否通过两图发现差异。这个或许让测试者来验证。 - **异常/超时案例**：测试用户长时间不操作： \* 在蓝图阶段开启Wizard，让它等待2分钟（低于timeout 600s），看有没有任何异常（应该无，依然等待）。 \* 超过10分钟（可临时把timeout调短便于测试），看Wizard是否超时关闭session，AI收到超时消息没有（可以mock AI在test code里await interactive\_feedback raise TimeoutError）。 \* 在测试阶段或Refine阶段，用户关闭浏览器窗口，Wizard日志应提示WS断开[[94]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L68-L76)。AI若等待则最终TimeoutError。这样session应该被cleanup标记TIMEOUT。 \* 多测试remote: 启动Wizard在无DISPLAY环境simulate remote, ensure that it didn't try to open a browser but logs such instruction. - **多模式交叉**：如果多模式UI显示4象限，测试选其他模式按钮: \* 例如选“快速脚本”模式，目前Wizard没实实现special process，可能就当普通feedback处理。测试Wizard是否至少不报错，例如可以让AI准备一种简单prompt，对应Wizard直接收集一次feedback完事。要看模式选择对其它模式的fallback机制是否健壮。如果choose "fast script",Wizard可以立即结束模式选择会话并返回AI一个“fast\_script mode chosen”但随后AI可能自己直接output code或ask for minimal feedback。我们暂时可模拟就选其他看看Wizard有没有Unhandled情况（应该没有，因为会话还是切换mode然后就Stop until AI next call, not complex).

**用户验收测试：** - *Alpha内部测试*：如前述Phase6，让几位公司内熟悉需求的开发者操作一遍，重点关注： - 他们是否理解Wizard每一步想要他们做什么？（界面文字是否清晰） - 他们对AI产生的蓝图和测试是否信任/满意？这涉及AI质量，不是Wizard程序本身，但Wizard表现也很重要——图有没有渲染错、测试易读否。 - 整个流程耗时他们能接受吗？是否觉得比不使用Wizard更省心？收集主观感受。 - *Beta测试（可选）*：让少部分真实用户使用，收集日志和反馈。重点看Happy Path Completion Rate在小样本的表现。如果很低，找出原因改进。Beta期间重点监控Wizard日志： - 比如看看多少session卡在Plan没confirm或Test没approve超过长时间->说明用户弃用了。 - 收集用户吐槽UI的点。

**性能与负载测试：** - 由于Wizard主要供单开发者异步使用，并发低，所以无需大规模并发测试。但可做一些： - 启动Wizard服务然后同时从两条线程simulate AI calls,确保WebUIManager锁机制使其中一个等待或拒绝。Minidoracat design single session,如果并发调用可能一个被拒绝with an error. - 测试当Project目录很大(比如含上千文件)时Wizard性能。比如Refine阶段若尝试解析全项目将挂很久，所以明确避免这种操作。Focus在Wizard自身err handling: e.g. if code analyzer loop through 1000 files might blow up time, ensure we set a reasonable limit or skip. - Memory: Use memory\_monitor (psutil) to snapshot usage after major steps, ensure no leak after session end (the memory freed count in log[[242]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L860-L868)should reflect images logs cleaned).

**安全测试：** - 有意识构造些可能危害的输入: - 在Mermaid源码里插入 <script>alert('xss')</script>检查Mermaid渲染是否原样输出script导致XSS。如果Mermaid未净化，我们可以手动对Mermaid输入进行转义或配置mermaidAPI.initialize({securityLevel:'strict'})之类来防XSS。 - 尝试通过Wizard run\_command执行不安全命令（Wizard会拒绝常见危险串，但例如rm变形?). Testing safe\_parse\_command with strings like "rm -rf /" expects ValueError. - 尝试在feedback文本中注入HTML/JS看前端是否会渲染。（feedback显示一般都在 code block or pre? Actually feedback from AI is shown in Markdown mode with DOMPurify cleaning asMinidoracat said they've integrated DOMPurify[[243]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L54-L58). We should confirm front-end still cleans potential malicious content in summary. We'll do code review ensure that or test by injecting a link or script in AI summary see if it's inert).

**验收标准：** - 所有测试用例通过后，满足： - **功能正确**：按PRD流程可以走通，包括典型与边界场景，最终用户能完成一次规范先行开发，并确认代码。 - **性能可接受**：流程各阶段等待时间合理（取决于AI速度，但Wizard自身开销不明显增加用户等待）。UI交互流畅无明显卡顿。 - **稳定性**：多轮次交互不崩溃，错误情况有适当提示处理。长时间运行无资源泄露或崩溃。 - **用户体验良好**：内部评估UI易用，外部测试用户能理解并完成，没有表重大困惑。Happy Path完成率在内部测试中接近100%（除非AI因素）。正式版本当然追求大部分用户都能走完。

一旦达到这些标准，就可以认为本设计实现成功，The Wizard V1满足预期，可以交付给用户试用并收集进一步反馈。

[[1]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L9-L13) [[2]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L10-L18) [[4]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L54-L62) [[5]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L60-L68) [[6]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L61-L67) [[7]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L70-L78) [[8]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L152-L160) [[9]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L160-L165) [[41]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L10-L13) [[195]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L9-L12) [[238]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L140-L149) [[239]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L146-L154) [[240]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md#L17-L25) GitHub

<https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/system-overview.md>

[[3]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L430-L439) [[12]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L3-L12) [[13]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L16-L24) [[16]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L562-L571) [[17]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L573-L582) [[18]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L454-L463) [[19]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L142-L150) [[20]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L152-L161) [[21]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L183-L191) [[22]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L207-L215) [[23]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L458-L466) [[24]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L469-L477) [[27]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L470-L477) [[28]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L508-L517) [[29]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L520-L528) [[30]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L540-L549) [[31]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L553-L559) [[32]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L537-L544) [[164]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L194-L202) [[179]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L279-L288) [[180]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L283-L290) [[181]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L280-L288) [[182]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L283-L291) [[188]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L480-L488) [[189]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L492-L500) [[190]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L498-L506) [[192]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L440-L448) [[193]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L442-L450) [[202]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L475-L483) [[203]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L242-L251) [[204]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L244-L253) [[206]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L259-L264) [[212]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L10-L18) [[213]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L510-L518) [[214]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L459-L466) [[233]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L330-L338) [[234]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py#L332-L340) GitHub

<https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/server.py>

[[10]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,provide%20exactly%20the%20same%20functionality) [[11]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,provide%20exactly%20the%20same%20functionality) [[122]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,file%20storage%2C%20privacy%20controls%2C%20history) [[123]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,time%20statistics%2C%20flexible%20timeout%20settings) [[124]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=creating%20new%20sessions%20or%20commits,formats%20for%20enhanced%20content%20readability) [[125]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=export%20,formats%20for%20enhanced%20content%20readability) [[126]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,formats%20for%20enhanced%20content%20readability) [[127]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,Traditional%20Chinese%2C%20English%2C%20Simplified%20Chinese) [[222]](https://github.com/Minidoracat/mcp-feedback-enhanced#:~:text=,formats%20for%20enhanced%20content%20readability) GitHub - Minidoracat/mcp-feedback-enhanced: Enhanced MCP server for interactive user feedback and command execution in AI-assisted development, featuring dual interface support (Web UI and Desktop Application) with intelligent environment detection and cross-platform compatibility.

<https://github.com/Minidoracat/mcp-feedback-enhanced>

[[14]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L29-L38) [[15]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L53-L60) [[146]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L62-L70) [[147]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L68-L76) [[171]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L19-L27) [[172]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L29-L37) [[173]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L33-L41) [[174]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L34-L38) [[175]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L35-L43) [[176]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md#L37-L45) GitHub

<https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/interaction-flows.md>

[[25]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L1126-L1135) [[26]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L1132-L1139) [[33]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L38-L46) [[34]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L48-L57) [[35]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L80-L88) [[36]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L100-L108) [[37]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L101-L109) [[38]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L151-L159) [[39]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L107-L115) [[40]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L113-L121) [[42]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L116-L124) [[43]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L120-L128) [[44]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L124-L132) [[45]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L138-L145) [[56]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L329-L338) [[57]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L348-L356) [[58]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L435-L443) [[59]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L352-L361) [[60]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L367-L375) [[61]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L372-L380) [[62]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L380-L388) [[63]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L374-L382) [[64]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L378-L386) [[65]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L388-L396) [[66]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L392-L399) [[67]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L400-L408) [[68]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L408-L417) [[69]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L414-L422) [[70]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L422-L431) [[71]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L435-L444) [[72]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L440-L448) [[73]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L454-L463) [[74]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L469-L473) [[75]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L471-L475) [[112]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L321-L329) [[119]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L608-L616) [[120]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L134-L141) [[121]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L610-L618) [[131]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L629-L638) [[132]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L630-L638) [[133]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L669-L678) [[134]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L681-L690) [[135]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L704-L712) [[136]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L706-L714) [[137]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L32-L40) [[138]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L713-L721) [[141]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L635-L644) [[142]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L638-L647) [[143]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L649-L657) [[144]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L637-L646) [[145]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L640-L648) [[160]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L653-L657) [[167]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L1124-L1133) [[168]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L1130-L1138) [[191]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L338-L346) [[196]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L142-L149) [[215]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L96-L104) [[216]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L343-L351) [[236]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py#L422-L430) GitHub

<https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/main.py>

[[46]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L124-L132) [[47]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L131-L138) [[48]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L130-L139) [[49]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L137-L145) [[50]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L134-L141) [[51]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L32-L40) [[52]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L126-L134) [[53]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L152-L160) [[54]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L158-L167) [[55]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L168-L174) [[99]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L519-L528) [[100]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L533-L541) [[116]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L720-L728) [[117]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L740-L748) [[139]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L554-L563) [[140]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L564-L570) [[148]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L540-L548) [[149]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L544-L552) [[177]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L715-L723) [[178]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L496-L504) [[183]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L536-L544) [[184]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L538-L546) [[185]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L539-L542) [[186]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L483-L490) [[187]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L483-L491) [[194]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L502-L510) [[197]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L485-L493) [[198]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L713-L721) [[199]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L719-L727) [[205]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L63-L66) [[207]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L576-L585) [[208]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L580-L589) [[209]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L580-L588) [[219]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L730-L738) [[220]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L780-L788) [[221]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L840-L848) [[223]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L504-L512) [[224]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L804-L813) [[225]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L819-L828) [[226]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L870-L874) [[227]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L784-L792) [[228]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L840-L849) [[229]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L686-L695) [[230]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L84-L93) [[231]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L98-L107) [[232]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L102-L110) [[237]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L710-L718) [[241]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L144-L151) [[242]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py#L860-L868) GitHub

<https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/models/feedback_session.py>

[[76]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L56-L65) [[77]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L62-L71) [[78]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L74-L82) [[79]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L76-L84) [[80]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L91-L100) [[81]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L120-L128) [[82]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L14-L21) [[83]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L16-L24) [[84]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L18-L26) [[85]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L23-L31) [[86]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L28-L36) [[87]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L32-L40) [[88]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L38-L46) [[89]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L44-L51) [[90]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L44-L52) [[91]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L50-L58) [[92]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L54-L62) [[93]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L60-L64) [[94]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L68-L76) [[95]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L76-L83) [[96]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L80-L83) [[97]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L618-L626) [[98]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L622-L628) [[101]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L628-L634) [[102]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L629-L637) [[103]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L634-L642) [[104]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L636-L644) [[105]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L644-L652) [[106]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L646-L655) [[107]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L661-L669) [[108]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L663-L671) [[109]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L670-L678) [[110]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L674-L682) [[111]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L686-L689) [[113]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L62-L70) [[114]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L26-L35) [[115]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L14-L22) [[118]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L62-L65) [[128]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L96-L105) [[129]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L100-L108) [[170]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L628-L636) [[217]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L26-L34) [[218]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py#L74-L82) GitHub

<https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/routes/main_routes.py>

[[130]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L30-L39) [[161]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L58-L65) [[166]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L60-L64) [[169]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L10-L18) [[200]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L40-L48) [[201]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L46-L50) [[210]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L22-L25) [[211]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L72-L79) [[235]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L12-L15) [[243]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md#L54-L58) GitHub

<https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/RELEASE_NOTES/CHANGELOG.md>

[[150]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L137-L143) [[151]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L63-L71) [[152]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L70-L78) [[153]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L87-L95) [[154]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L104-L113) [[155]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L131-L139) [[156]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py#L76-L83) GitHub

<https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/src/mcp_feedback_enhanced/web/utils/browser.py>

[[157]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L130-L139) [[158]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L152-L160) [[159]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L148-L156) [[162]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L172-L180) [[163]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L174-L183) [[165]](https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md#L146-L154) GitHub

<https://github.com/Minidoracat/mcp-feedback-enhanced/blob/541ca1e5db8f5d1357d856f163feaacda1eb7144/docs/architecture/deployment-guide.md>