

# Kindness Companion (善意伴侣) v3.0 项目进展与计划

最后更新: 2023-05-10

## 1. 项目概述

本应用旨在通过技术传递善意，鼓励用户实践并反思日常善行。v3.0 版本采用 "API 优先" 策略，将 AI 功能通过调用云端 API 实现，以提高可移植性和简化打包。

核心理念包括鼓励实践、促进反思、提供 AI 陪伴、可视化进步和尊重隐私。






## 2. 当前进展 (已完成 / 基本完成)

- ☒ **基础框架:**
  - 项目结构已按 README.md (API 优先版) 搭建，包含 frontend, backend, ai\_core, api, resources, tests 等目录。
  - 使用 Python 3.10+ 和 PySide6 构建。
- ☒ **核心本地功能 (Backend):**
  - backend/database\_manager.py: SQLite 数据库连接和基本操作封装。已添加 AI 同意字段 (ai\_consent\_given)。
  - backend/user\_manager.py: 用户注册、登录、密码管理。已添加 AI 同意状态管理方法 (get\_ai\_consent, set\_ai\_consent)。
  - backend/challenge\_manager.py: 预设挑战的加载、用户订阅管理。
  - backend/progress\_tracker.py: 基础的打卡记录、数据查询（总次数、分类次数、连胜等）逻辑。
  - backend/reminder\_scheduler.py: 使用 APScheduler 实现本地提醒的基础框架。
- ☒ **前端 UI (Frontend):**
  - frontend/main\_window.py: 应用主窗口框架。已集成 PetWidget 并连接用户状态信号。
  - 各子界面 (challenge\_ui.py, checkin\_ui.py, profile\_ui.py, progress\_ui.py, reminder\_ui.py, user\_auth.py) 的基本 UI 文件已创建。
  - frontend/progress\_ui.py: 实现了进度统计、日历高亮、打卡记录表格、成就徽章的加载和展示逻辑。
  - frontend/widgets/: 包含一些自定义控件，如 AnimatedMessageBox, BaseDialog。
  - frontend/widgets/ai\_consent\_dialog.py: ☒ 已创建 AI 同意对话框 UI。

- `frontend/pet_ui.py`:  已实现 AI 电子宠物交互界面逻辑，连接 `ai_core.pet_handler`。已集成 AI 同意检查和对话框弹出逻辑。已实现 GIF 动画圆形遮罩和透明背景。
- `frontend/user_auth.py`:  实现了登录和注册界面。已集成 `PetWidget` 以在登录界面显示动画，并修复了动画不显示的 **Bug**。
-  应用配置与资源:
  - `main.py`: 应用入口，包含字体加载 (`load_fonts`)、主题管理 (`ThemeManager` 支持浅色/深色模式切换)、基础组件初始化。
  - `resources/`: 包含图标、字体、样式表 (QSS)、动画等资源。
  - `resources.qrc` 和 `resources_rc.py`: 用于将资源编译进应用。
-  打包与分发:
  - `README.md` 提供了详细的 macOS PyInstaller 打包指南，包括 `.spec` 文件配置示例。
-  AI 核心 API 集成 (`ai_core/`):
  - `api_client.py`:  封装了通用的 API 请求逻辑。
  - `dialogue_generator.py`:  已实现调用 ZhipuAI API 生成对话。
  - `emotion_analyzer.py`:  已实现调用 ZhipuAI API 分析情感。
  - `pet_handler.py`:  已实现核心宠物交互逻辑，集成对话和情感分析。
  - `report_generator.py`:  已实现调用 ZhipuAI API 生成个性化周报，包含丰富的上下文信息和数据比较。
-  隐私与安全:
  -  实现明确的用户同意 (Opt-in) 流程: 在 `frontend/pet_ui.py` 中首次调用 AI 功能前检查同意状态 (`user_manager.get_ai_consent`)，如果未同意则显示 `AIConsentDialog`，并根据结果更新数据库 (`user_manager.set_ai_consent`)。已修复同意对话框在每次登录时都弹出的 **Bug**。
  -  确保 `config.py` 中的 API Key 安全管理 (已添加到 `.gitignore`)。

## 3. 进行中 / 待办事项 (TODO)

### 3.1 MVP (最小可行产品)

-  AI 核心 API 集成 (`ai_core/`):
  -  `report_generator.py`: 已实现调用云端文本生成 API 生成善举报告的逻辑，包含丰富的上下文信息和数据比较。
  -  `dialogue_generator.py`: 已实现调用 ZhipuAI API 生成对话，支持多种事件类型 (打卡、反思、用户消息等)。
  -  `emotion_analyzer.py`: 已实现调用 ZhipuAI API 分析情感，将用户文本分类为积极、消极或中性。
  -  `pet_handler.py`: 已实现核心宠物交互逻辑，根据事件类型和情感分析结果生成合适的对话和动画建议。

- 🎨 前端 AI 功能集成 ( frontend/ ):
- ✅ 已在 progress\_ui.py 中实现 AI 报告展示功能，包括报告生成按钮、报告显示区域、加载状态显示和错误处理。报告生成前会检查用户是否已同意 AI 功能。
- ✅ 已在 pet\_ui.py 中实现 AI 宠物交互界面，支持圆形遮罩的 GIF 动画显示、对话气泡和用户消息输入。
- ✅ 已在 main\_window.py 中集成 PetWidget，使其在应用主界面中可见并能响应用户状态变化。
- 🛡️ 隐私与安全:
- ✅ 已在 profile\_ui.py 中实现 AI 功能设置开关，允许用户通过复选框启用/禁用 AI 功能，并更新 ai\_consent\_given 状态。设置更改后会显示确认消息，并通过信号通知应用的其他部分。
- ✅ 已实现 widgets/ai\_consent\_dialog.py 对话框，详细说明 AI 功能的数据使用情况并获取用户同意。
- ✅ 已在 user\_manager.py 中实现 AI 同意状态管理方法，支持查询和更新用户的 AI 功能同意状态。
- ✅ 已在 database\_manager.py 中添加 ai\_consent\_given 字段，用于存储用户的 AI 功能同意状态。
- 🧪 测试:
- ✅ 已为 report\_generator.py 添加全面的单元测试，包括 API 请求的构建和响应的处理逻辑 (使用 mock)。
- ✅ 已为 dialogue\_generator.py 添加单元测试，覆盖不同事件类型的对话生成和错误处理。
- ✅ 已为 emotion\_analyzer.py 添加单元测试，测试情感分析功能和 API 调用逻辑。
- ✅ 已为 pet\_handler.py 添加单元测试，测试不同事件类型的处理和动画建议生成。
- **TODO:** 继续完善 AI 核心模块的单元测试覆盖率，特别是边缘情况和错误处理。

## 3.2 未来增强特性 (Post-MVP)

- 🎯 AI 个性化推荐 ( ai\_core/ & frontend/ ):
- recommendation\_engine.py: **TODO:** 实现调用推荐系统 API 或文本分析/向量数据库 API 的逻辑，基于用户历史打卡记录和偏好推荐新的善行挑战。(README.md:58)
- frontend/challenge\_ui.py: **TODO:** 在挑战浏览界面集成推荐结果，添加"为您推荐"部分，突出显示个性化推荐的挑战。
- 实现思路:
  - 收集用户历史数据（已完成的挑战、打卡频率、偏好类别）
  - 通过 ZhipuAI API 分析用户模式并生成推荐
  - 在前端以吸引人的方式展示推荐结果
- 🎮 AI 优化激励机制 ( ai\_core/ ):

- `gamification_optimizer.py`: **TODO**: 实现调用机器学习/分析 API 的逻辑, 为调整游戏化元素提供建议。(#README.md:65)
- **实现思路**:
  - 分析用户参与模式 (活跃时间、完成率、连续打卡等)
  - 通过 AI API 生成个性化的激励策略 (提醒时间、成就解锁节奏、挑战难度)
  - 动态调整应用中的游戏化元素以提高用户参与度
- 📁 **匿名善意墙 (api/ & frontend/)**:
  - `api/community_handler.py`: **TODO**: 实现自建后端 API (Flask/FastAPI) 的端点, 用于发布和获取匿名分享 (需要配合云数据库)。(#README.md:70)
  - `frontend/community_ui.py`: **TODO**: 实现展示匿名善意墙的前端界面, 包括浏览、发布和互动功能。
  - `api/app.py`: **TODO**: 注册 `community_handler` 蓝图, 配置路由和中间件。
  - **实现思路**:
    - 使用 Flask/FastAPI 创建轻量级后端 API
    - 集成 Firebase Firestore 或 Supabase 作为云数据库
    - 实现内容审核机制 (可选择使用 AI 内容审核 API)
    - 设计吸引人的社区界面, 支持点赞、评论等互动功能
- 📦 **打包与分发**:
  - **TODO**: 完善 macOS 的代码签名和公证流程, 确保应用可以在 macOS 上顺利安装和运行。(#README.md:328)
  - **TODO**: (可选) 研究其他平台 (Windows, Linux) 的打包流程, 确保跨平台兼容性。
  - **实现思路**:
    - 使用 PyInstaller 创建独立可执行文件
    - 为 macOS 实现代码签名和公证
    - 为 Windows 创建安装程序 (可选)
    - 为 Linux 创建适当的包格式 (可选)

## 4. 下一步计划 (优先级)

1. ✅ **实现 AI 功能设置开关**:
  - ✅ 已在 `profile_ui.py` 中实现 AI 功能设置开关, 允许用户通过复选框启用/禁用 AI 功能, 并更新 `ai_consent_given` 状态。
  - ✅ 已实现 `widgets/ai_settings_widget.py` 组件, 提供更详细的 AI 功能说明和设置选项。
  - ✅ 已在 `user_manager.py` 中实现 AI 同意状态的管理逻辑, 确保用户偏好被正确存储和应用。
2. ✅ **在前端展示 AI 报告**:
  - ✅ 已完成 `ai_core.report_generator` 的实现和测试, 支持生成个性化的周报。

- ☒ 已确认 `progress_ui.py` 中的报告展示功能正常工作，包括报告生成按钮、报告显示区域、加载状态显示和错误处理。
  - ☒ 已实现 `AIReportThread` 类，确保报告生成在后台线程中进行，不会阻塞 UI。
3. ☒ 完成 AI 核心单元测试:
- ☒ 已为 `report_generator.py` 添加全面的单元测试，包括 API 请求的构建和响应的处理逻辑。
  - ☒ 已为 `dialogue_generator.py` 添加单元测试，覆盖不同事件类型的对话生成和错误处理。
  - ☒ 已为 `emotion_analyzer.py` 添加单元测试，测试情感分析功能和 API 调用逻辑。
  - ☒ 已为 `pet_handler.py` 添加单元测试，测试不同事件类型的处理和动画建议生成。
4. 完善 UI 设计与用户体验:
- **TODO:** 实现温暖主题的界面样式，确保与用户偏好一致。
  - **TODO:** 优化深色模式适配，确保所有界面元素在深色模式下显示正常。
  - **TODO:** 调整字体大小和布局，提高可读性和整体美观度。
  - **TODO:** 确保界面在不同缩放比例下保持良好的样式和布局。
  - **TODO:** 参考 Sourcio UI/UX 设计风格，优化前端样式。
5. 完善 MVP 测试:
- **TODO:** 使用 `pytest-qt` 进行 GUI 组件测试，确保界面交互正常。
  - **TODO:** 进行手动集成测试，验证各模块之间的协作是否正常。
  - **TODO:** 测试不同操作系统和屏幕分辨率下的应用表现。
  - **TODO:** 进行用户体验测试，收集反馈并进行优化。
6. 探索未来增强功能:
- **TODO:** 实现 AI 个性化推荐功能，根据用户历史数据推荐合适的善行挑战。
  - **TODO:** 开发 AI 优化激励机制，提高用户参与度和持续性。
  - **TODO:** 考虑实现匿名善意墙功能，促进用户之间的善行分享和互动。

## 5. 详细技术实现说明

### 5.1 AI 核心模块实现

AI 核心模块采用 API 优先策略，通过调用 ZhipuAI API 实现各种 AI 功能:

#### 1. 对话生成 ( `dialogue_generator.py` )

- 使用 ZhipuAI 的 GLM-4-flash 模型生成自然、个性化的对话响应
- 根据不同事件类型（打卡、反思、用户消息等）构建专门的提示词
- 实现错误处理和重试机制，确保 API 调用的稳定性

#### 2. 情感分析 ( `emotion_analyzer.py` )

- 分析用户反思文本和消息的情感倾向（积极、消极、中性）

- 使用低温度参数确保分类结果的一致性和准确性
- 实现结果验证和后处理，处理意外的 API 响应

### 3. 宠物交互 ( `pet_handler.py` )

- 整合对话生成和情感分析功能，提供完整的宠物交互体验
- 根据事件类型和情感分析结果选择合适的动画和对话内容
- 实现多层错误处理，确保即使 API 调用失败也能提供基本功能

### 4. 报告生成 ( `report_generator.py` )

- 聚合用户的打卡数据、成就和进度信息
- 构建详细的提示词，包含丰富的上下文信息和数据比较
- 生成个性化的周报，突出用户的进步和成就

## 5.2 前端 AI 功能集成

前端模块通过以下方式集成 AI 功能：

### 1. AI 宠物界面 ( `pet_ui.py` )

- 实现圆形遮罩的 GIF 动画显示，支持多种情绪状态
- 提供对话气泡和用户消息输入功能
- 集成事件处理系统，响应用户操作和应用状态变化

### 2. AI 报告展示 ( `progress_ui.py` )

- 实现报告生成按钮和报告显示区域
- 使用后台线程生成报告，避免阻塞 UI
- 提供加载状态显示和错误处理机制

### 3. AI 功能设置 ( `profile_ui.py` 和 `widgets/ai_settings_widget.py` )

- 提供 AI 功能开关和详细说明
- 实现用户同意流程和状态管理
- 通过信号机制通知应用其他部分用户的 AI 功能偏好变化

## 5.3 隐私与安全实现

为保护用户隐私和数据安全，实现了以下机制：

### 1. 用户同意管理

- 在数据库中添加 `ai_consent_given` 字段，存储用户的 AI 功能同意状态
- 实现 `get_ai_consent` 和 `set_ai_consent` 方法，管理用户的同意状态
- 提供详细的同意对话框，说明数据使用情况和用户权利

### 2. API 密钥管理

- 使用 `config.py` 安全存储 API 密钥（已添加到 `.gitignore`）
- 实现 `get_api_key` 方法，安全地获取和使用 API 密钥

- 提供错误处理机制，处理 API 密钥缺失或无效的情况

## 5.4 测试策略实现

为确保应用的质量和稳定性，实现了以下测试策略：

### 1. AI 核心单元测试

- 使用 unittest 框架和 mock 对象测试 AI 核心模块
- 模拟 API 响应，测试正常和异常情况下的行为
- 验证提示词构建、响应解析和错误处理逻辑

### 2. 前端组件测试

- 使用 pytest-qt 测试 UI 组件的行为和交互
- 验证信号和槽连接的正确性
- 测试用户输入处理和界面更新逻辑

### 3. 集成测试

- 测试前端和后端模块的协作
- 验证数据流和状态管理
- 确保各组件在实际使用场景中正常工作

## 6. 用户界面优化计划

根据用户反馈和设计目标，计划进行以下界面优化：

### 1. 主题与样式

- 实现温暖主题的界面样式，使用柔和的色彩和圆角设计
- 优化深色模式，确保所有元素在深色背景下清晰可见
- 参考 Sourcio UI/UX 设计风格，提升整体视觉体验

### 2. 布局与排版

- 调整字体大小，提高可读性
- 优化界面布局，减少视觉混乱
- 确保各页面保持一致的设计语言和交互模式

### 3. 响应式设计

- 优化不同屏幕尺寸和分辨率下的显示效果
- 确保界面元素在缩放时保持良好的比例和布局
- 测试不同设备和显示设置下的用户体验

### 4. 交互优化

- 统一按钮大小和样式，提供一致的交互体验
- 优化表单和控件的布局和反馈机制

- 添加适当的动画和过渡效果，提升用户体验