

DesktopPet-PKU 项目报告

——“码力全开”队

一、程序功能介绍

DesktopPet-PKU 是一款集趣味养成、生活管理与 AI 交互于一体的 Qt 桌面应用程序。它旨在通过一个可爱的桌面宠物形象（主体形象灵感来自信科“攻城狮”，围绕其展开二创与动画的设计），为用户的日常学习和生活提供便利与乐趣。程序的核心是将一个可拖拽、有情感属性的虚拟宠物放置在桌面上，并围绕它构建一系列实用功能模块。

核心功能模块



- **互动宠物系统**

- 程序的核心，一个始终悬浮于桌面的宠物
- 拥有饱食度、清洁度、好感度等动态属性，会随时间推移而下降，需要用户通过“喂食”、“洗澡”等互动来维持

- **生活管理套件**

- **课程表 (Timetable)**: 提供可视化的周课程表，用户可直接在表格中编辑、保存课程信息，支持多行文本输入
- **日程计划 (Schedule)**: 简洁的任务列表 (To-Do List)，支持添加带优先级的任务，并在完成后标记
- **运动记录与统计**: 记录跑步、羽毛球等十余种运动，自动计算卡路里消耗，提供数据可视化图表（柱状图、饼图）

- **AI 智能助手 (ChatDialog)**

- 集成 DeepSeek 大语言模型，支持上下文连续对话
- AI 回复支持 Markdown 格式渲染

- **个性化与激励系统**

- **换肤系统**: 宠物形象来自信科“攻城狮”形象，有多套皮肤（工作风、运动风、学风）
- **成就解锁**: 特定皮肤需完成挑战解锁（如完成 10 个任务、累计跑步 85 公里）

- **数据持久化**

- 用户数据（课程表/日程/运动数据/皮肤状态）自动保存到本地文件（JSON/TXT）

二、项目各模块与类实现细节

①前端实现细节

主窗口与宠物交互：MainWindow 类

主要功能

程序启动入口和主界面，承载桌面宠物及功能模块入口。采用无边框设计，支持拖拽移动。

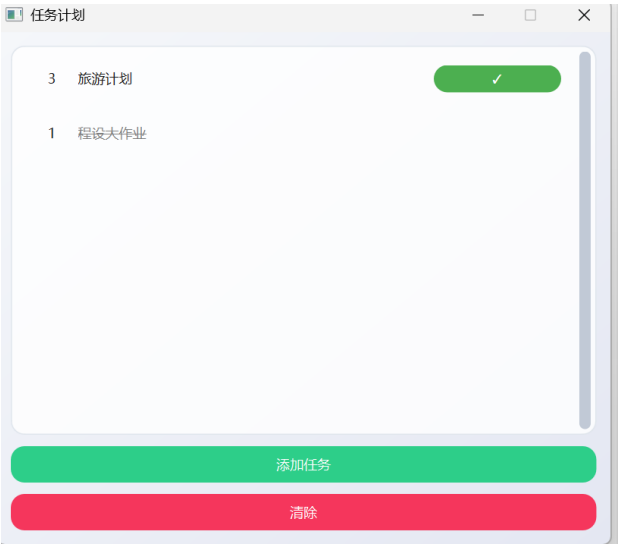
实现细节

- 窗口与视图
 - 通过 `setWindowFlags(Qt::FramelessWindowHint)` 和 `setAttribute(Qt::WA_TranslucentBackground)` 实现透明无边框窗口
 - 使用 `QGraphicsView / QGraphicsScene` 展示宠物图片
 - 重写 `mousePressEvent / mouseMoveEvent` 实现拖拽功能
- 属性系统
 - 三个 `QProgressBar` 可视化展示饱食度/清洁度/好感度（QSS 美化样式）
 - `QTimer` 每秒触发 `updateAttributes` 槽函数模拟属性消耗
- 交互与导航
 - “喂食”、“洗澡”按钮直接修改属性值
 - 功能按钮（课程表/日程/AI 聊天）创建对应对话框，使用 `QPointer` 管理窗口

生活管理模块

1. 日程计划：Schedule 类

- 功能：
 1. 任务添加：通过对话框创建新任务（内容+优先级）
 2. 状态管理：标记任务完成（带视觉反馈）
 3. 数据维护：
 - 自动保存/加载任务数据（JSON格式）
 - 清除所有任务（二次确认）
 4. 智能排序：未完成优先 > 同状态按优先级排序
 5. 成就系统：完成10个任务解锁皮肤待办事项列表，支持添加/清除/完成任务，成就解锁
- 成员变量：



变量名	类型	描述
taskData	QJsonArray	存储任务的JSON数组
totalCompleted	int	累计完成任务计数
AchievementV	const int	成就解锁阈值(默认10)
ui->listWidget	QListWidget*	任务列表展示控件

- 实现：
 - sortTasks()
通过 std::sort 实现任务智能排序，优先显示未完成任务，同状态任务按 priority 值升序排列。
 - createTaskItem()
为每个 Task 对象创建可视化列表项，通过 contentLabel 设置样式，并为未完成任务添加 completeBtn 按钮。
 - saveTasks()
将 taskData 和 totalCompleted 封装为JSON格式，使用 QFile 写入 user_data/tasks.json 文件实现数据持久化。
 - updateAchievements()
当 completedCount 达到 AchievementV 阈值时，更新 user_data/lock.txt 文件实现成就解锁。
 - resizeEvent()
重载函数响应窗口大小变化，动态计算 itemHeight 并调整列表项尺寸，实现响应式布局。

2. 课程表： Timetable 类

- 功能：
 - 课程表管理：
 - 支持单元格多行文本编辑（带自动换行）
 - 实时保存/加载课程数据（TXT格式）
 - 清空课表功能（含二次确认）
 - 成就系统：添加20个课程解锁"学院风"皮肤
 - UI/UX特性：
 - 透明背景+圆角单元格设计
 - 响应式布局（自动调整行高）
 - 关闭未保存提示机制
 - 数据持久化：
 - 自动创建user_data目录
 - 换行符转义存储（ \n → \\n ）



• 成员变量：

变量名	类型	描述
ui->tableWidget	QTableWidget*	课程表主体控件
textEditDelegate	TextEditDelegate*	单元格多行文本编辑委托
AchievementVI	const int	成就解锁阈值(默认20)
saved	bool	数据保存状态标记

• 实现：

- 1. TextEditDelegate::createEditor()
通过创建 QTextEdit 编辑器支持单元格内的多行文本输入和自动换行功能。
- 2. Timetable::saveTimetable()
将表格数据转换为CSV格式并保存到文件，同时处理换行符转义和成就解锁检测。
- 3. Timetable::resizeTableToViewport()
动态计算表格总高度（包含表头和内容高度），实现自适应布局。
- 4. Timetable::on_tableWidget_cellChanged()
单元格内容变更时自动调整行高并更新 saved 状态标记。
- 5. Timetable::closeEvent()
重写关闭事件处理函数，在未保存时显示 TimetableRemind 提醒对话框。

运动管理模块

1. 运动主界面： Sports 类

• 功能：

- 1. 进度追踪：
 - 双进度条显示跑步里程(85km)和次数(10次)目标
 - 实时刷新进度数据（JSON格式存储）
- 2. 成就系统：
 - 完成85km里程+10次跑步解锁"运动风"皮肤
 - 修改 lock.txt 标记解锁状态
- 3. 数据管理：
 - 自动创建 user_data 目录并保存 runningTask.json
 - 重置功能含二次确认保护
- 4. 子窗口管理：



- 运动记录(records)和统计(stats)独立窗口
- 窗口关闭自动刷新主界面数据展示运动目标（如“85 公里跑步挑战”），提供记录/统计入口

• 成员变量：

变量名	类型	描述
ui->progressDistance	QProgressBar*	里程进度条控件
ui->progressCount	QProgressBar*	次数进度条控件
sportsData	QJsonObject	存储运动数据的JSON对象
recordWin	records*	运动记录子窗口指针
statsWin	stats*	数据统计子窗口指针

• 实现：

1. resetUi()
动态构建响应式UI布局，应用CSS样式定制圆形按钮和渐变色进度条，支持窗口尺寸自适应。
2. updateRunningProgress()
校验单次跑步距离(≤10km)，更新 sportsData 并触发成就检测，进度满时修改 lock.txt 解锁皮肤。
3. loadData()/saveData()
通过 runningTask.json 实现数据持久化，自动创建 user_data 目录保障跨会话进度保存。
4. on_btnReset_clicked()
重置前弹窗二次确认，清空进度数据并更新UI，防止误操作导致数据丢失。
5. on_btnRecord_clicked()
创建 records 子窗口并绑定 destroyed 信号到 refreshData ，实现关闭子窗口时自动刷新主界面。

2. 运动记录： records 类

• 功能：

1. 多运动类型支持：
 - 10种运动项目独立记录（跑步、羽毛球、游泳等）
 - 每种运动提供专属卡片设计（不同背景色）
2. 智能计算：
 - 实时计算卡路里消耗（基于预设代谢系数）
 - 输入验证（时间0-600分钟，距离0-50km）
3. 数据管理：
 - 自动记录当前日期并保存至JSON数据文件



- 清空输入框的提交后处理

4. 可视化反馈：

- 跑步/羽毛球提交后播放透明遮罩GIF动画
- 跑步里程>2km触发主页面进度更新

• 成员变量：

变量名	类型	描述
m_sports	Sports*	主窗口引用指针
ui->runDisEdit	QLineEdit*	跑步距离输入框
ui->runTimeEdit	QLineEdit*	跑步时间输入框
各运动 TimeEdit	QLineEdit*	各运动时长输入框
各运动 CalLabel	QLabel*	卡路里显示标签

• 实现：

1. records() 构造函数

应用CSS渐变背景(a1c4fd→c2e9fb)和卡片化设计，为10种运动设置差异化背景色，连接所有输入框的 textChanged 信号到计算槽函数。

2. calculateRunningCalories()

读取跑步时间和距离，按 时长×代谢率 公式实时计算卡路里，动态更新 runCalLabel 显示结果。

3. on_runButton_clicked()

验证输入有效性后保存跑步记录，播放400×367透明 running.gif 动画，距离>2km时触发 Sports::updateRunningProgress() 。

4. on_badmintonBtn_clicked()

提交羽毛球记录后播放400×400透明 badminton.gif 动画，实现2000ms超时保护确保窗口关闭。

5. calculateSwimmingCalories()

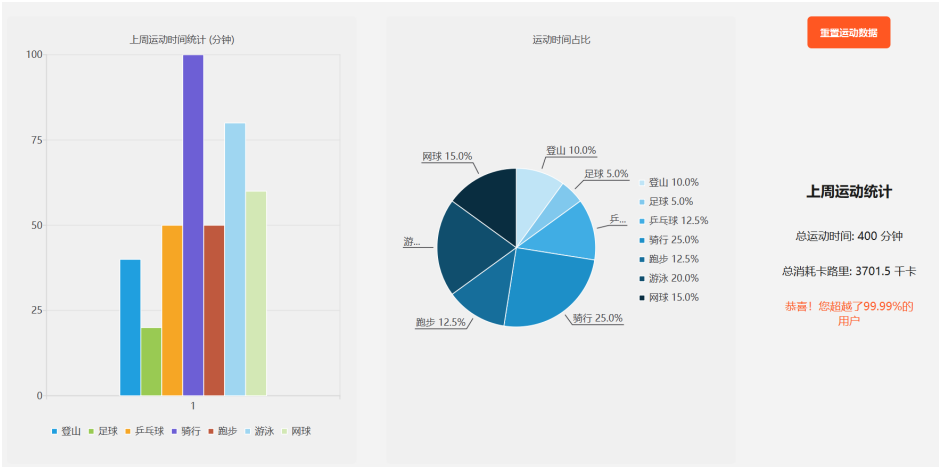
代表其他运动卡路里计算逻辑，使用 swimming_cal 代谢系数计算消耗值，实时更新游泳卡路里标签。

3. 运动统计： stats 类

• 功能：

1. 数据可视化：

- 柱状图展示上周各运动项目时长
- 饼状图显示运动时间占比分布



2. 多维分析：

- 计算总运动时间和消耗卡路里
- 生成超越99%用户的激励提示

3. 数据管理：

- 支持重置所有运动数据（含二次确认）
- 自动加载 sport_record.json 运动记录

4. 多语言支持：

- 自动转换运动项目英文标识为中文标签

• 成员变量：

变量名	类型	描述
barChartView	QChartView*	柱状图展示控件
pieChartView	QChartView*	饼状图展示控件
sportTimes	QMap<QString, double>	运动项目-时长映射表
sportCalories	QMap<QString, double>	运动项目-卡路里映射表
resetButton	QPushButton*	数据重置按钮

• 实现：

1. loadSportsData()

从 sport_record.json 加载数据，解析时间与卡路里信息，自动跳过空文件或无效数据。

2. createBarChart()

创建柱状图展示各运动时长，使用中文标签映射（如"running"→"跑步"），无数据时显示占位提示。

3. createPieChart()

生成饼状图显示运动占比，自动计算百分比标注切片标签，零数据时显示透明占位块。

4. updateTextStats()

计算总运动时长和卡路里消耗，生成激励文案（"超越99.99%用户"），数据分析驱动UI更新。

5. onResetClicked()

重置前弹出二次确认对话框，调用 sptRec.reset() 清空数据源，完成后刷新所有图表

AI 智能助手： ChatDialog 类

主要功能

现代化聊天界面，支持上下文对话和富文本展示。

实现细节

- 布局与样式

- 使用 QSplitter 分割聊天区/输入区
- 深度 QSS 样式表实现气泡对话风格

- 消息展示

- QTextBrowser 显示对话
- addMessage 函数构建 HTML 结构实现左右对齐气泡
- **Markdown 处理**：通过 QMarkdownTextDocument 转换 Markdown → HTML
- “思考中...” 消息替换机制

- 用户交互

- 重写 keyPressEvent 支持 Enter / Ctrl+Enter 发送
- 自定义 paintEvent / mousePressEvent 实现标题栏“设置”图标

②后端及核心服务

API 服务： DeepSeekService 类

主要功能

封装 DeepSeek API 的 HTTP 通信细节，解耦网络请求与 UI 逻辑。

实现细节

- 网络通信：

- 使用 QNetworkAccessManager 处理异步网络操作
- sendMessageWithHistory 构建 QNetworkRequest 并设置 Authorization 头部

- JSON 处理：

- QJsonObject / QJsonArray 构建请求体

- 异步机制：

- QNetworkReply::finished 信号绑定 onApiResponse 槽函数
- 解析后发射 responseReceived / errorOccurred 信号

Markdown 渲染器： QMarkdownTextDocument 类

主要功能

轻量级 Markdown → HTML 转换器（支持标题/加粗/斜体/行内代码/列表）。

实现细节

- 解析逻辑：



- setMarkdown 分割输入字符串为 QStringList
- 逐行处理：startsWith() 识别标题/列表， QRegularExpression 替换文本样式
- 集成使用：
 - ChatDialog 接收回复 → 创建转换器对象 → setMarkdown() → toHtml() → 显示

运动数据模型： Sport_Item 类

主要功能

运动数据存储/计算/文件操作的核心后端。

实现细节

- 数据结构：
 - 管理 user_data/sportsData.json 文件（含 time_data / weekly_time_data / *_records 数组）
- 核心方法：
 - 为每种运动提供 add...Record 方法（如 addBadmintonRecord ）
- 文件操作：
 - readSportsJson / writeSportsJson 封装文件 I/O
 - reset 方法恢复数据初始状态

三、小组成员分工情况

成员	分工
宋澄	负责了sports与records的后端实现，涉及图片动画的路径管理，ui界面的美化与总体调整，update不同更新版本的项目，维护github项目，管理qt项目的推进计划与小组讨论等合作进度
邱国宏	负责了实现sports,record,stats页面,schedule功能和页面,ai聊天功能和页面,以及协同整合调试整体项目
李孟桐	负责clotheschanging,mainwindow及相关文件,timetable及相关文件的编写，负责主界面设计，宠物互动代码，课程表模块前后端设计，换肤模块前后端设计以及皮肤制作

四、项目总结与反思

项目总结

成功结合虚拟宠物与生活管理工具，实践了多项 Qt 核心技术：

- **UI 设计**：QSS / 自定义绘图实现美观界面
- **模型/视图/委托**：TextEditDelegate 解决复杂控件需求
- **网络编程**：QNetworkAccessManager 异步通信
- **数据处理**：QJsonDocument 持久化复杂数据
- **图表可视化**：QtCharts 转换数据为直观图表
- **软件工程**：前后端分离提高可维护性

反思与展望

1. **代码冗余问题**：
 - Sport_Item.cpp 的 add...Record 函数存在重复代码 → 需重构为通用方法（遵循 DRY 原则）
2. **数据管理方式**：
 - 多独立 JSON/TXT 文件管理低效 → 如果有大数据管理可引入 SQLite 统一管理
3. **可扩展性**：
 - 添加新运动/成就需修改多处代码 → 应设计数据驱动架构（外部配置文件动态加载）
4. **UI界面的美观性**：
 - 桌面宠物形象与AI chat部分的UI界面尚有美化空间 → 进一步优化动画设计，适当引入图像渲染技巧

从项目中的收获：

1. 利用github管理工程文件，进行协同合作
2. 实现前端后端的分工协作，着重注意接口的测试与优化
3. 学习到QT项目中gif动画的插入与位置的控制
4. 积累了一些数据写入读出与数据文件存放位置的经验，和处理文件交互产生的bug的心得，将类似功能的数据文件存放在user_data文件夹中，用统一的方式读取和写入，保持代码的可维护性