一 程序功能介绍

1. 财务全流程管理

交易记录：添加/查看/导入导出（JSON）

智能分析：

- 收支平衡计算
- 消费热点识别
- 自动生成财务报告

2. AI 深度整合

生成消费建议结构：

1. 消费行为总结
2. 三条优化建议
3. 六个月存钱计划

技术特性：

- 30 秒超时控制
- 网络异常自动恢复
- 支持开发/生产双模式密钥

3. 数据可视化体系

即时分析：

- 消费类型占比（饼图）
- 月度对比（柱状图）

深度洞察：

- 年度趋势分析
- 地理热力图

4. 目标管理系统

存钱计划：

- 双进度跟踪（金额/时间）
- 动态成就提示
- 完成自动庆祝仪式

成就系统：

- 连续记账成就
- 百笔交易里程碑

5. 安全与体验

企业级安全：

- 密码强度策略
- 转账操作二次验证

交互优化：

- 表单自动完成
- 智能输入验证
- 上下文感知提示

技术架构亮点

1. 模块化设计：各功能组件高内聚低耦合
2. 响应式 UI：自适应布局+统一样式引擎
3. 数据读写：JSON 贯穿存储/传输/展示全流程
4. 异常处理：网络超时/JSON 解析/数据校验三重防护
5. 扩展接口：预留 analyzeHabits() 等扩展点

该项目实现了从交易记录到智能分析的完整财务生态，通过深度结合传统财务分析与现代 AI 技术，为用户提供专业级的消费洞察和优化建议。其模块化架构和完备的错误处理机制，为后续功能扩展奠定了坚实基础。

二 项目模块与类设计

1. 核心数据结构

Transaction 类 (Transaction.h)

交易记录基础结构

成员：

QDateTime time 交易时间

float amount 金额（单位：元）

QString location 消费地点

QString category 分类（默认“未分类”）

QString notes 备注

功能方法：

toJson() 序列化为 JSON 对象

fromJson() 从 JSON 解析

设计特点：支持自动记录时间，提供完整的 JSON 序列化能力

2. 数据管理层

DataModel 类 (DataModel.h/cpp)

核心数据容器

成员：

QList<Transaction> transactions 交易记录列表

功能方法：

addTransaction() 添加交易

saveToJson() 保存为 JSON 文件

loadFromJson() 从 JSON 加载

getTransactions() 获取交易列表（只读）

设计特点：集中管理所有交易数据，提供持久化支持

3. 财务分析引擎

FinancialAnalyzer 类 (FinancialAnalyzer.h/cpp)

财务计算核心

功能方法：

generateSummary() 生成财务摘要（总收入/总支出/余额/主要消费类别）
analyzeHabits() 消费习惯分析（预留接口）
算法特点：
自动区分收入/支出
智能识别高频消费类别
支持 Q_INVOKABLE 供 QML 调用

4. AI 服务模块
AIService 类 (AIService.h/cpp)
DeepSeek API 交互核心
关键组件：
QNetworkAccessManager 网络通信
QTimer 请求超时控制
DeepSeekPayload 专用请求结构体
核心流程：
1. 构建符合 JSON 请求
2. 添加 Bearer Token 认证
3. 处理响应数据（成功/超时/错误）
信号设计：
analysisComplete(QString) 分析完成
errorOccurred(QString) 错误通知

5. 用户界面系统
(1) 主界面 (MainWindow.h/cpp)
功能枢纽：
交易管理（添加/查看）
数据分析入口（报表/AI 分析/存钱计划）
系统功能（设置/用户管理）
创新组件：
存钱计划系统：
可视化进度展示（金额/天数双维度）
目标设定与达成奖励机制
JSON 持久化存储（Saveplan.json）
智能表单：
地点输入自动补全（历史记录学习）
实时金额格式化
表单验证引擎

(2) 分析对话框 (AnalysisDialog.h/cpp)
AI 分析结果展示器
特色功能：
进度对话框集成（超时自动提醒）
HTML 格式化输出（支持富文本展示）
异步请求管理（可中途取消）

(3) 报表系统 (ReportAnalysisDialog.h/cpp)
多维度数据可视化：
1. 消费类型饼图：分类占比分析
2. 趋势柱状图：月度/年度消费对比
3. 热力图：消费地点分布强度
技术亮点：
QtCharts 动态渲染
自动颜色梯度生成
响应式表格布局

(4) 用户子系统
登录/注册 (LoginDialog, RegisterDialog)
安全特性：
密码复杂度校验（数字+大小写字母）
用户数据加密存储（userdata.json）
防重复注册机制
个人成就 (MyProfileWindow)
成就体系：
连续记账周成就
百笔交易成就
统计维度：
活跃记账天数
交易总量

6. 辅助模块
启动画面 (SplashScreen.h/cpp)
品牌展示与加载进度反馈
自适应图片缩放
平滑进度动画
设置面板 (SettingsDialog.h/cpp)
账户安全操作：
退出登录
账户注销（二次验证）

三 小组成员分工情况

在本次“Folvo 智能记账本”项目的开发过程中，我们团队秉持高效协作、各尽其能的原则，对项目任务进行了明确的分工。每位成员都出色地完成了各自负责的模块，并通过有效的沟通与代码整合，共同推动了项目的顺利完成。具体分工如下：
1. 王子莜同学
(1)数据持久化与交互：负责设计和实现了项目的数据核心——与 JSON 文件的交互逻辑。他构建了稳定可靠的 DataModel，确保了账目数据的序列化（保存到 JSON）和反序列化（从 JSON 加载）过程的正确性和高效性，为整个应用的数据持久化存储奠定了坚实的基础。

(2)AI 智能分析模块： 作为项目的亮点功能,独立完成了 AI 分析模块的开发。该模块能够基于用户的历史收支数据，进行智能分析与总结，并为用户提供个性化的财务建议。这部分工作为我们的应用注入了“智能”的灵魂。

2. 符海宝同学

（1）主应用架构与 UI/UX 设计： 主导了应用的整体架构设计与核心用户体验的构建。他不仅从零开始构思并使用 Qt 框架实现了以图片为主体、现代化按钮栏布局的独特主界面，还负责了整个应用窗口的初始化、布局管理和多窗口跳转逻辑，确保了应用流畅、直观的交互流程。

（2）记账流水模块： 精心设计并开发了“查看账单”子窗口，该窗口能够清晰、有序地展示所有交易记录，并实现了按时间排序、滚动浏览等关键功能，是用户进行数据核对的主要界面。

（3）个人成就系统： 作为项目的核心创新点之一，他独立设计并开发了富有激励性的“个人成就”模块。通过游戏化的方式鼓励用户持续记账，极大地增强了产品的用户兴趣和使用乐趣。

（4）按钮与交互设计： 他负责了应用中所有主要按钮的设计与功能绑定，通过细致的 QSS 样式定制，实现了不同颜色、不同状态的按钮视觉效果。

3. 王程栋同学

（1）数据可视化——统计报表： 负责将枯燥的账目数据转化为直观的图表。他利用 Qt 的图表库实现了统计报表功能，能够以饼状图、柱状图等多种形式展示用户的收支构成和消费趋势，帮助用户一目了然地掌握自己的财务状况。

（2）用户安全——密码登录： 为了保障用户财务数据的私密性，设计并实现了应用的用户登录模块。他构建了密码验证机制，确保只有授权用户才能访问自己的账目信息，是应用安全性的重要保障。

（3）应用可配置性——设置模块： 还开发了“设置”功能模块，允许用户根据自己的偏好对应用进行个性化配置，。

四 项目总结与反思

经过紧张而充实的开发周期，我们的“Folvo 智能记账本”项目已成功实现预期功能并达到了预设目标。这次大作业不仅是对我们 Qt 编程技术的一次全面检验，更是一次宝贵的团队协作和项目管理实践。在此，我们对整个项目过程进行总结与反思。

一、 项目成果与亮点总结

1、完整的功能闭环： 我们成功构建了一个从用户登录、日常记账、数据可视化分析到智能建议、个性化设置的完整功能闭环，为用户提供了一站式的个人财务管理解决方案。

2、现代化的 UI/UX 设计： 我们摒弃了传统的列表式主界面，采用了更具吸引力的图片主体和模块化按钮栏布局，并通过 QSS 样式表实现了美观、统一且响应迅速的视觉效果。

3、智能化与游戏化结合： 项目的亮点在于 AI 智能分析和个人成就系统。AI 模块能够为用户提供超越简单记录的财务洞察力；而成就系统则通过正向激励，将枯燥的记账行为变得更具趣味性和持续性。

4、稳健的数据管理： 通过 JSON 文件进行数据持久化，并设计了清晰的加载、保存和追加逻辑，确保了用户数据的安全与完整。

5、模块化与高内聚： 在开发过程中，我们遵循了良好的软件工程实践，将不同功能（如数据模型、UI、各个功能窗口）解耦，使得代码更易于维护、扩展和团队分工协作。

二、 开发过程中的收获与反思

1、技术能力的提升：

深入掌握 Qt 框架： 从基础的控件使用、布局管理，到高级的信号槽机制、事件处理、QSS 样式定制、Model/View 架构以及多窗口通信，我们对 Qt 的理解和应用能力得到了质的飞跃。
软件工程实践： 我们学会了如何进行模块划分，如何设计类与接口，以及如何通过 Git 进行高效的团队代码协作与版本控制，避免了大量的代码冲突和集成问题。
2、团队协作的重要性：
沟通是关键： 定期的线上/线下会议和即时沟通工具的使用，确保了信息同步，让我们能够快速解决接口定义、功能依赖等问题。
明确分工与信任： 清晰的分工让每个人都能专注于自己的领域，而相互信任则让我们能够放心地依赖队友完成的模块，最终高效地完成系统集成。
3、遇到的挑战与解决方案：
UI 布局与 QSS 调试： 在实现自定义 UI 布局时，我们花费了大量时间调试 QGridLayout 的伸展因子和 QSS 样式表的细节，以达到理想的视觉效果。通过不断尝试和查阅文档，我们最终掌握了精确控制 UI 的技巧。
数据流管理： 项目初期，我们曾对数据的保存逻辑（覆盖 vs. 追加）有过讨论。通过明确“保存”是写回完整数据集这一核心原则，并考虑了“另存为”、“导入”等扩展功能，我们避免了潜在的数据丢失风险。
多窗口交互： 如何在主窗口和多个子窗口之间安全、高效地传递数据（如 DataModel 指针）和状态，是我们面临的另一个挑战。通过合理的父子关系设定和 Qt 的信号槽机制，我们构建了清晰的窗口通信模式。
三、 未来展望
虽然当前版本的“Folvo”已具备核心功能，但仍有广阔的提升空间：
云同步功能： 引入云端数据库（如 Firebase）或同步服务，实现跨设备数据同步。
预算管理系统： 开发更完善的预算设定和超支预警功能。
更深入的 AI 分析： 引入更复杂的机器学习模型，提供消费预测、异常检测等高级功能。
平台扩展： 探索将应用移植到移动平台（Android/iOS）的可能性。

总而言之，本次 Qt 大作业是一次极具价值的学习经历。我们不仅将课堂上学到的理论知识应用到了实践中，更重要的是学会了如何像一个真正的软件开发团队一样思考、沟通和协作。这些宝贵的经验将对我们未来的学习和职业生涯产生深远的影响。