记账本应用设计与实现实验报告

1. 软件的主要功能

本次课程实验要求根据既定的 UML 设计文档完成一个可运行的软件系统。为了更好地覆盖数据管理、界面交互和简单的数据统计分析等方面，我选用了“个人记账本”作为实现主题。这个应用的业务流程较为清晰，功能范围适中，能够较好地体现从设计到实现的完整过程。

系统主要采用 C++ 语言开发，界面部分基于 Qt 6 框架，数据存储采用轻量化的 SQLite3 数据库，并通过 Qt 提供的数据访问模块进行连接与操作。目前完成的版本实现了一个基础可用的原型界面，包含多个标签页，可以进行日常记账常见的核心操作。虽然暂未实现 UML 图中计划的云同步、高级图表分析等扩展功能，但基本流程已经连贯可用。

核心模块如下：

交易记录模块（Req001）

作为整个系统的核心功能，用户可以在这里记录每一笔收入或支出。单条交易记录内容包括金额、类型、分类、关联账户、时间和备注。当用户添加或删除交易时，对应账户余额将同步更新，保证数据一致。用户还可以对记录进行修改和移除。

账户管理模块（Req003）

支持用户手动创建多个资金账户，例如现金、银行卡、第三方支付等。每个账户具有独立余额，有利于查看资金使用情况。

分类管理模块（Req002）

为了便于后期分析，支出和收入可以按自定义分类进行划分，如餐饮、交通、工资等。在添加交易时可用下拉框选择，提高效率。

预算提醒模块（Req005）

用户可为特定分类设置月度预算。当消费超过预算的 80% 时，系统会在新交易操作后以弹窗形式提醒，帮助用户合理规划支出。

2. 根据 UML 图实现代码

1. 代码规模说明

源代码约 650 行。其中约一半为界面和交互，另一半为数据库访问和数据处理。

2. UML 转换

将 UML 类图作为代码实现的核心。

- Domain 层实体

UML 里定义的 Transaction、Account、Category 和 Budget 等实体，被抽象为 C++ 的 struct 结构，定义在 database.h 中，用于承载数据。

- 数据访问层

将 UML 中对数据库操作的接口，集中封装为一个 Database 类，由它负责连接 SQLite 并执行增删改查，向上层提供统一方法。

- 应用逻辑层

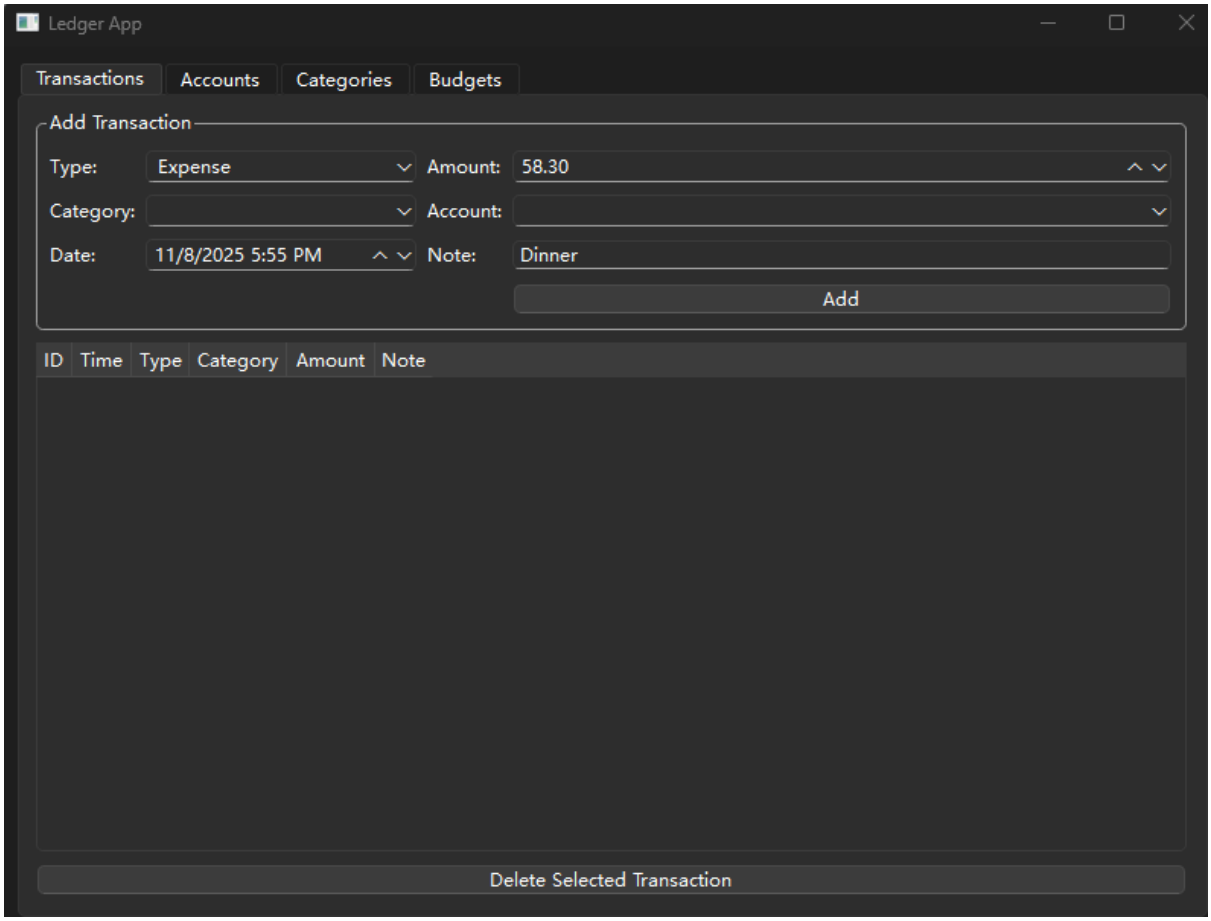
实验规模较小，我将这一部分逻辑集中处理在 MainWindow.cpp 中对应的槽函数里，例如交易添加按钮的槽函数承担了服务接口的职责。

- 界面层

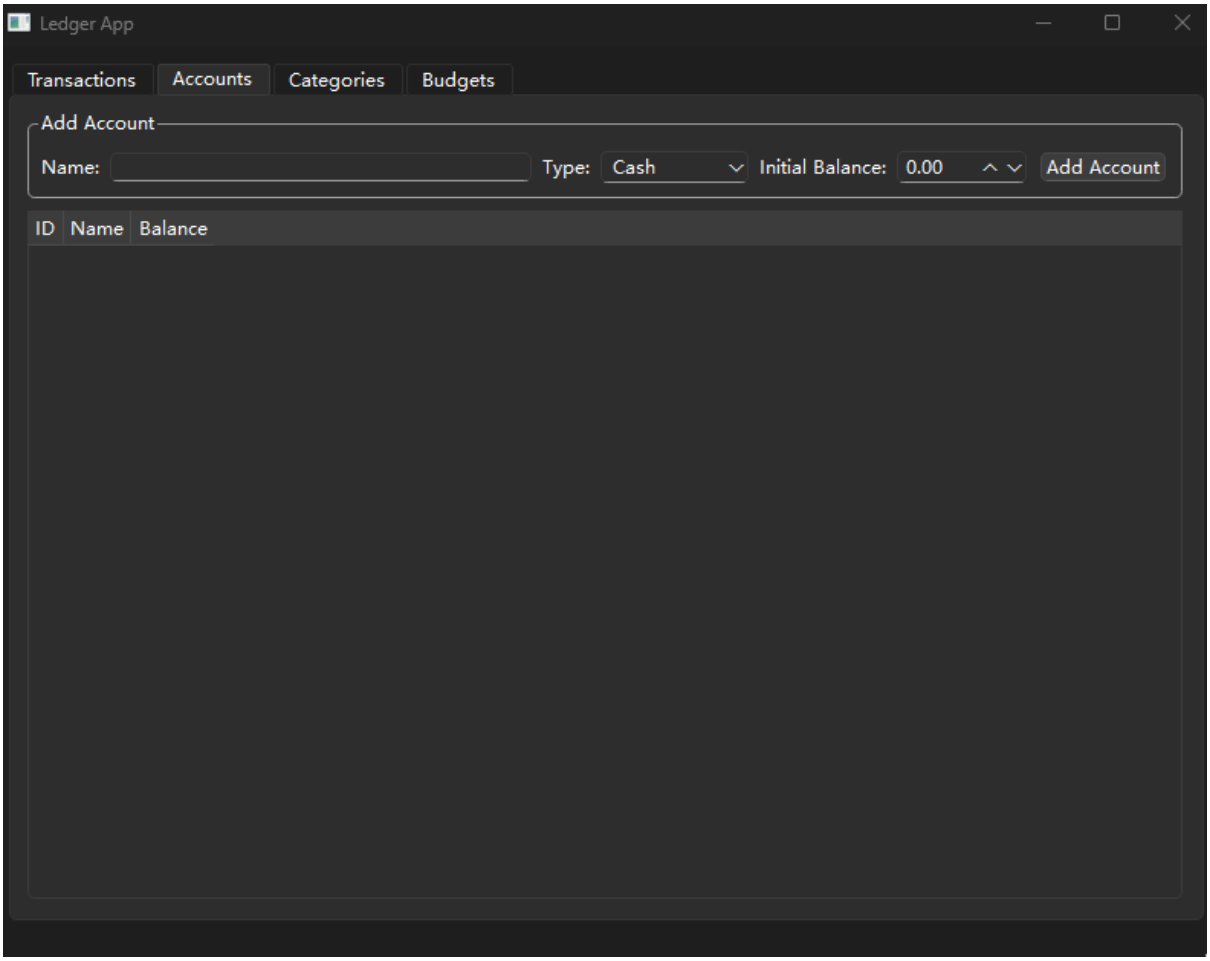
界面使用 Qt Designer 设计，通过 mainwindow.ui 配合 MainWindow 类，根据用户操作触发对应的槽函数。

3. 代码编译与运行结果

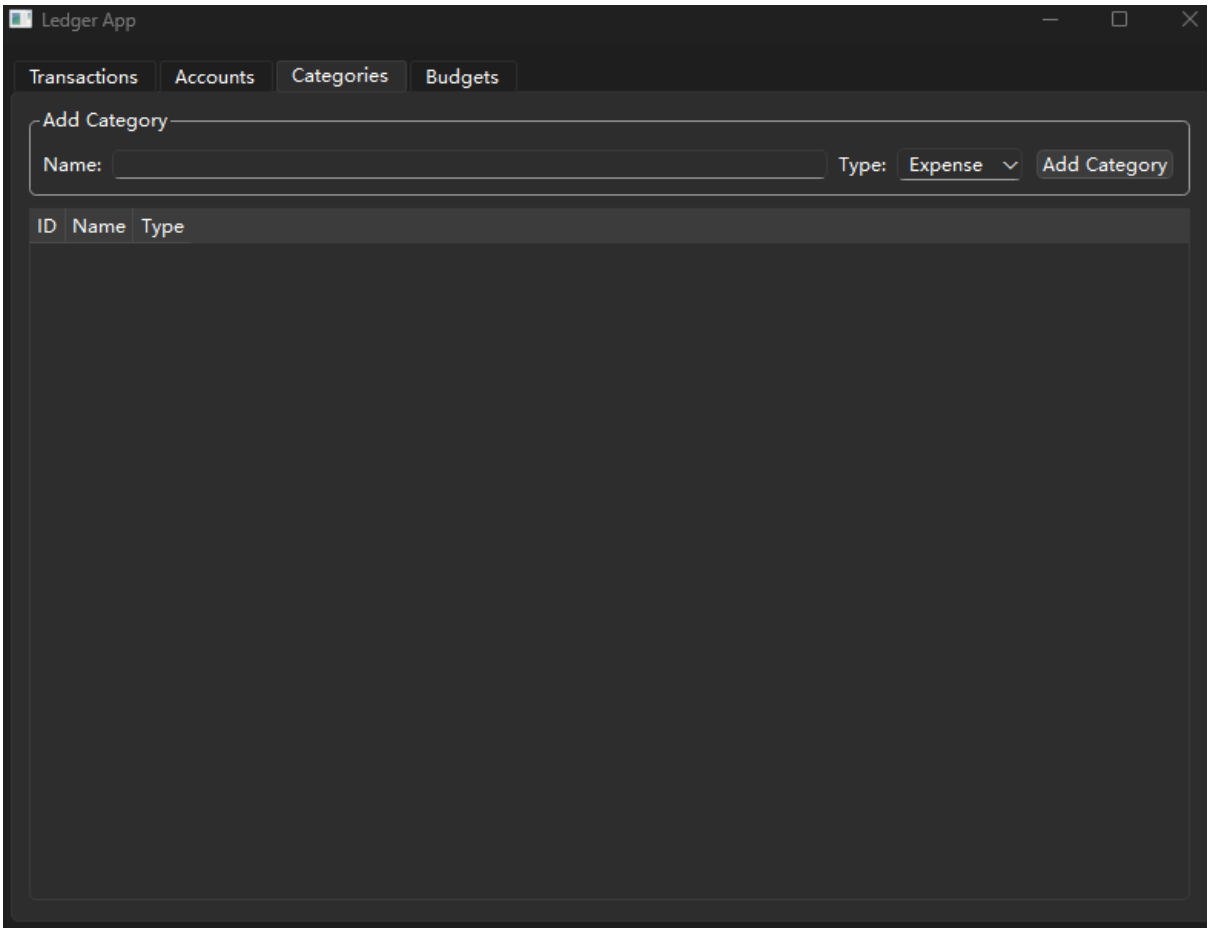
1. 主界面与核心功能：程序启动后，显示一个包含四个标签页的主窗口：Transactions、Accounts、Categories 和 Budgets。



2. 账户与分类管理：在 Accounts 和 Categories 标签页，可以成功添加新的账户和分类。添加后，列表会自动刷新。

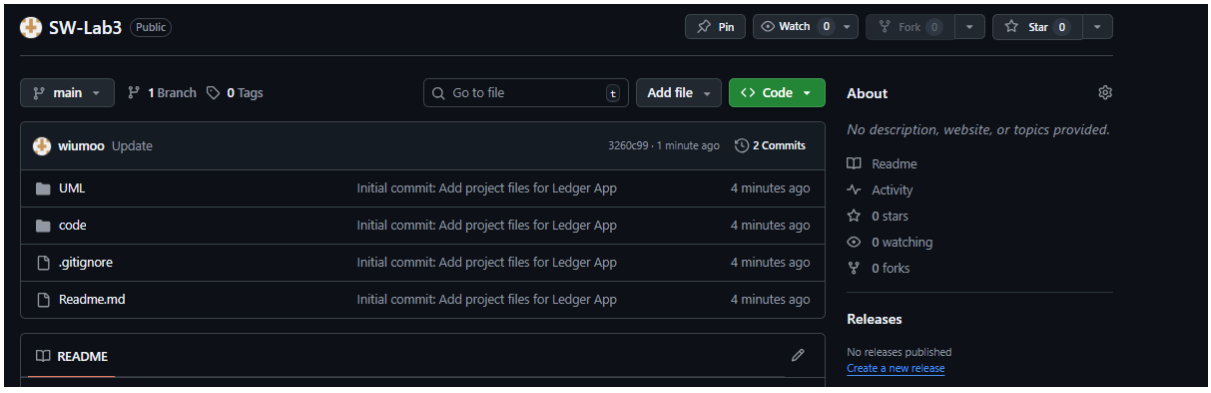


3. 在 Transactions 标签页，选择类型为 Expense，填入金额、选择分类和账户，点击 Add 按钮。交易被成功记录到下方的表格中，同时对应账户的余额减少。



4. 关闭或重新打开程序，之前添加的账户、分类、交易和预算信息都依然存在，证明数据已成功保存到 SQLite 数据库中。

4. Git 远程代码管理展示：



5. 大语言模型使用心得：

本次实验中我使用了 Gemini2.5pro 帮我生成完成本次实验的 To-List 来一步一步完成。