

PaperPass[免费版]AIGC检测报告

简明打印版

AIGC总体疑似度(高+中+轻): 4.03%

AIGC总体疑似度(高+中+轻): 2.26% (加权计算)

高度疑似AIGC占全文比:	0.0%
中度疑似AIGC占全文比:	0.0%
轻度疑似AIGC占全文比:	4.03%
不予检测文字占比:	30.21%

检测版本: 免费版(仅检测中文)

报告编号: FKZB6910D189F2E9B

论文题目: 个人记账本系统设计与建模实验报告

论文作者: 李禹江

论文字数: 3992

段落个数: 49

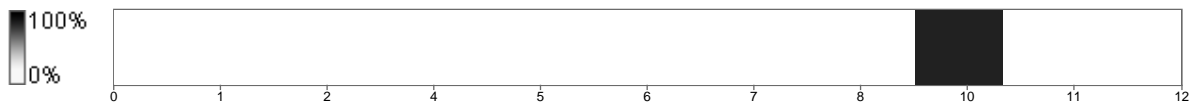
句子个数: 126

片段个数: 12

提交时间: 2025-11-10 1:38:17

查询真伪: <https://www.paperpass.com/check>

疑似度分布图:



正文中片段的不同颜色表示不同的疑似度范围

- 红色: AIGC生成疑似度在70%以上 (高度疑似)
- 橙色: AIGC生成疑似度在60%~70% (中度疑似)
- 紫色: AIGC生成疑似度在50%~60% (轻度疑似)
- 黑色: AIGC生成疑似度在50%以下
- 灰色: 过短片段、标题和英文等不予检测的文字

注: AIGC检测可有效识别文本是否部分或全部由AI模型生成, 检测结果与论文质量无关、仅表示论文中内容片段存在AI生成可能性的概率。

加权计算公式: (片段1疑似度 + 片段2疑似度 + ... + 片段n疑似度) / n
片段疑似度范围0.0~1.0, 合格片段按照0计算, 不计算不予检测文字

个人记账本系统设计与建模实验报告

一. 软件的主要功能

1. 主要功能

根据实验一的需求分析报告完成本记账软件的开发，本软件主要包括以下功能：

- 基础记账：支持基本记账和账目信息分类和管理
- 数据统计：基于matplotlib实现条形图、饼状图可视化收支
- 预算管理：设置消费预算，实时跟踪预算使用进度
- 汇率管理：支持人民币、美元、欧元、港币的基于汇率的兑换。（不支持联网查询即时汇率）
- 资金计划：资金计划与预算结合，相当于快捷设置预算，以及可以删除资金计划等，可以判断是否完成。
- 发票管理：支持电子发票文件关联与纸质发票描述记录（本次实现中仅包括账目-发票一一对应，并不能兼容多种发票格式智能获取发票内容添加账目）。
- 数据管理：本软件的账目使用本地 JSON 文件存储，支持相同数据结构的 JSON 文件作为账本的导入、导出及另存为功能。

其中发票管理、汇率管理、数据统计为本软件的特色功能。

2. 功能模块

- 账目管理模块：负责账目录入、删除、查询
- 预算管理模块：处理预算设置、修改及预算使用情况统计
- 汇率管理模块：管理多货币汇率数据，进行金额换算
- 资金计划模块：实现资金计划的创建、删除与列表展示
- 数据统计模块：通过图表可视化收支数据，生成统计分析结果
- 文件操作模块：负责账本数据的导入、导出、保存等文件交互
- UI 交互模块：提供用户与系统交互的可视化界面
- 数据存储模块：处理本地 JSON 文件的读写

3. 依赖环境

操作系统:

Windows: Windows 7 或更高版本 (推荐 Windows 10/11)

macOS: macOS 10.14 (Mojave) 或更高版本

Linux: Ubuntu 16.04 或更高版本, 或其他支持 Tkinter 的 Linux 发行版

python:

最低版本: Python 3.7

推荐版本: Python 3.8 或更高版本

不支持: Python 2.x 系列

python标准库:

tkinter >= 8.6

json >= 2.0

os >= 1.0

datetime >= 1.0

logging >= 1.0

typing >= 3.5.3

dataclasses >= 3.7

第三方依赖库:

tkcalendar >= 1.6.1 # 日期选择控件

matplotlib >= 3.3.4 # 数据可视化图表

4. 软件特点

本软件使用python语言进行开发, 基于python生态的完善性, 本软件可以在绝大多数平台上使用。

本软件实现了UI界面, 因此依赖于tkinter库。本软件并不依赖于pyQt, 因为pyQt的包管理实在是太差了。tkinter库操作简单, 且可以支持matplotlib库进行数据分析, 虽然界面相对较为简陋, 不过在使用功能上无伤大雅。

本软件目前使用离线模式, 不支持联网操作。且本软件并未使用数据库进行开发, 而使用json文件进行数据结构的设计和管理。我认为这样的举措可以使账本的分布式管理更加便捷, 同时简化了代码设计的难度, 也防止基于数据库的开发出现更多功能和安全上的bug。

本软件目前为单线程操作, 在避免并发bug的同时满足个人记账应用场景的需要。

二、根据UML图实现代码

这次实验干什么: 在实验二中, 我已经实验了一个功能完整的记账本软件。因此实验二生成的UML图是基于具体代码而非软件构想的, 但当时的具体代码非常不符合软件工程规范, 所以本次实验中, 我将根据UML图, 结合标准的软件工程规范, 对我的代码进行重构。

500行代码的要求： 在实验二中，我已经实验了一个功能完整的记账本软件。该软件代码未使用IDE，为逐行手敲完成，总代码为612行，去掉空行和注释总代码为550行以上。因为是基于python的tkinter库开发，且整个软件都放在一个类FinanceTrackerAPP下，所以这550行代码均为软件的核心逻辑代码，认为本软件应满足实验要求。经过重构后的软件代码为1230行，使用大语言模型添加了更多的注释，为了保持可读性加入了更多空行结构更加松散，压缩后代码大约为1000行左右。

1. 根据UML图实现代码

类图： 基于MVC（Model-View-Controller）架构设计了明确的数据实体，使用接口抽象依赖关系。

```
# Model
class Entry;      # 账目实体类
class Plan;       # 资金计划实体类
class DataManager; # 数据管理类
# View
class BaseView(ABC); # 视图基类
class MainView(BaseView); # 主界面视图
class ExchangeRateView(); # 汇率管理视图
class PlanView(); # 资金计划管理视图
# Controller
class FinanceController(); # 控制器
```

组件图：

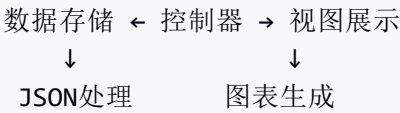
```

class FinanceController:
    def __init__(self):

        # 绑定视图回调函数
        def _bind_view_callbacks(self):
            # 运行应用
            def run(self):
                # 记账
                def record_entry(self):
                    # 验证账目数据
                    def _validate_entry_data(self, data: Dict[str, Any]) -> bool:
                        # 删除选中账目
                        def delete_entries(self):
                            # 设置预算
                            def set_budget(self):
                                # 导入账本
                                def import_data(self):
                                    # 另存为账本
                                    def save_as_data(self):
                                        # 保存账本
                                        def save_data(self):
                                            # 管理汇率
                                            def manage_exchange_rates(self):
                                                # 保存汇率
                                                def save_exchange_rates(self):
                                                    # 管理预算
                                                    def manage_plans(self):
                                                        # 添加预算
                                                        def add_plan(self):
                                                            # 删除预算
                                                            def delete_plan(self):
                                                                # 浏览发票文件
                                                                def browse_invoice_file(self):
                                                                    # 生成条形图统计
                                                                    def bar_analytics(self):
                                                                        # 生成饼状图统计
                                                                        def pie_analytics(self):
                                                                            # 计算总体财务状况
                                                                            def calculate_totals(self) -> Dict[str, float]:
                                                                                # 更新显示
                                                                                def update_display(self):
                                                                                    # 退出应用
                                                                                    def quit_app(self):

```

应用模型

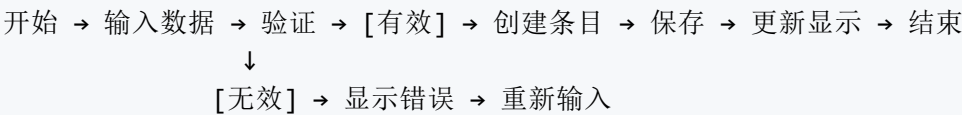


时序图：

以删除帐目为例：

- 1. 用户点击删除按钮 → View捕获事件
- 2. View调用Controller回调 → Controller验证选择
- 3. Controller调用Model删除数据 → Model更新存储
- 4. Controller通知View更新显示 → View刷新界面

活动图：



2. 代码运行结果

主窗口

账目录入

账目类型:Income

金额:0.0

货币类型:CNY

类别:Salary

日期:2025-11-09

发票类型:none

发票信息:

浏览

添加记账

删除选中

月度预算:0

设置预算

数据统计

条形图统计

饼状图统计

账目列表

导入账本

另存为

保存账本

管理汇率

资金计划

保存并退出

序号	类型	金额	货币	类别	日期	发票信息
----	----	----	----	----	----	------

添加账目

账目录入

账目类型:Income

金额:0.0

货币类型:CNY

类别:Salary

日期:2025-11-10

发票类型:none

发票信息:

浏览

添加记账

删除选中

月度预算:0

设置预算

个人记账本

数据统计

成功

i

账目记录成功

确定

条形图统计

饼状图统计

账目列表

导入账本

另存为

保存账本

管理汇率

资金计划

保存并退出

序号	类型	金额	货币	类别	日期	发票信息
1	Income	2000.00	CNY	Salary	2025-11-09	无
2	Income	2000.00	CNY	Salary	2025-11-09	无
3	Income	2000.00	CNY	Salary	2025-11-10	无

删除账目

个人记账本

账目录入

账目类型:Income

金额:0.0

货币类型:CNY

类别:Salary

日期:2025-11-10

发票类型:none

发票信息:

添加记账

删除选中

浏览

月度预算:0

设置预算

数据统计

成功

账目删除成功

确定

条形图统计

饼状图统计

账目列表

导入账本

另存为

保存账本

管理汇率

资金计划

保存并退出

序号	类型	金额	货币	类别	日期	发票信息
1	Income	2000.00	CNY	Salary	2025-11-09	无
2	Income	2000.00	CNY	Salary	2025-11-09	无

设置预算

账目录入

账目类型:Income

金额:0.0

货币类型:CNY

类别:Salary

日期:2025-11-10

发票类型:none

发票信息:

浏览

添加记账

删除选中

月度预算:1000.0

设置预算

数据统计

财务统计

4000.00

0.00

1000.00

3000.00

总收入总支出预算净收入

条形图统计

饼状图统计

账目列表

导入账本

另存为

保存账本

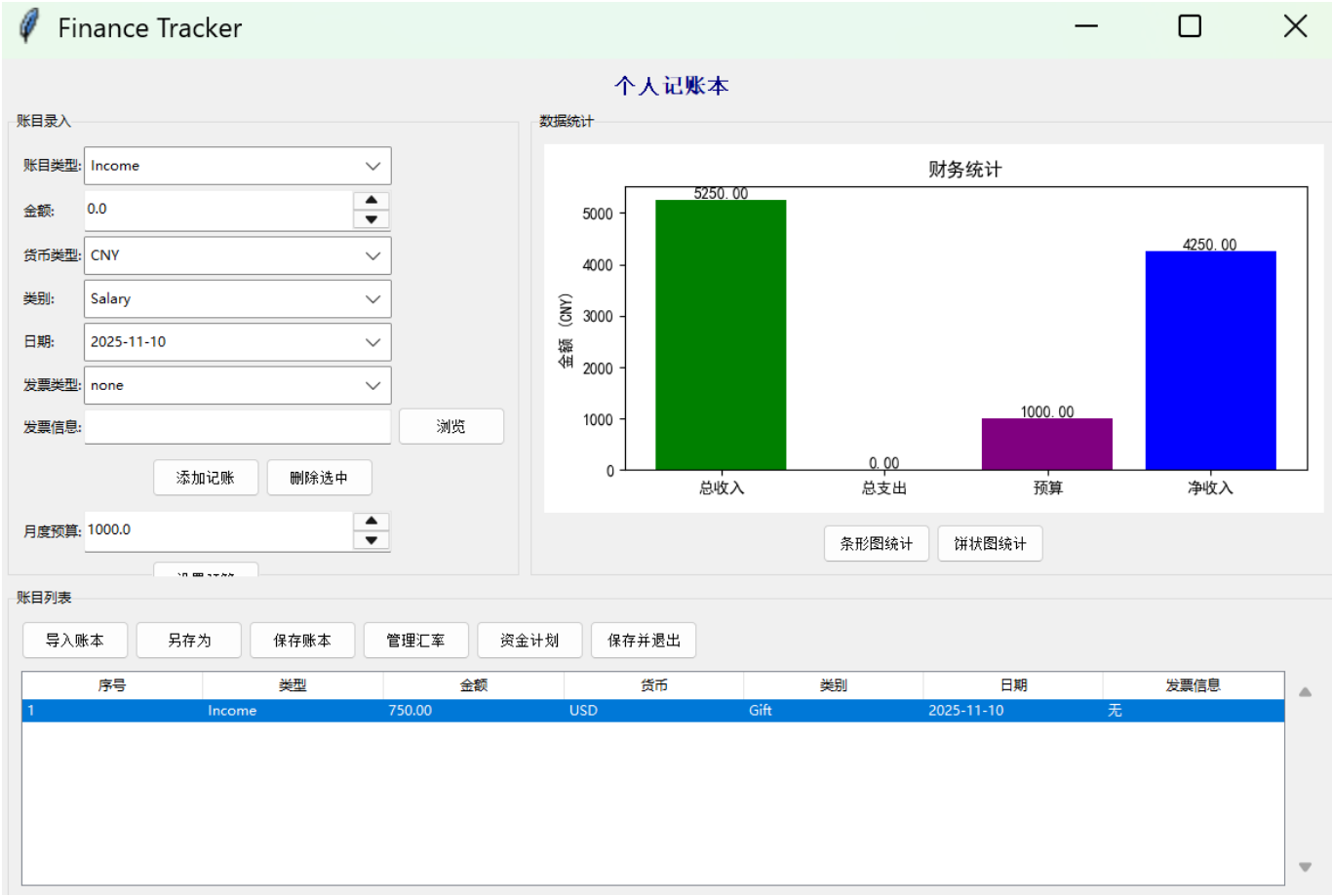
管理汇率

资金计划

保存并退出

序号	类型	金额	货币	类别	日期	发票信息
1	Income	2000.00	CNY	Salary	2025-11-09	无
2	Income	2000.00	CNY	Salary	2025-11-09	无

汇率管理



发票管理

(本软件发票管理目前只支持账目-发票的对应关系，并不能智能根据发票导入账目)

1	Income	750.00	USD	Gift	2025-11-10	无
2	Income	16000.00	CNY	Education	2025-11-10	paper: 转入学费
3	Expense	15600.00	CNY	Education	2025-11-10	electronic: D:/LEARNING_f

资金计划管理

资金计划与预算功能结合

资金计划管理

类型

开始日期

结束日期

添加新计划

计划类型: monthly

开始日期: 2025-11-10

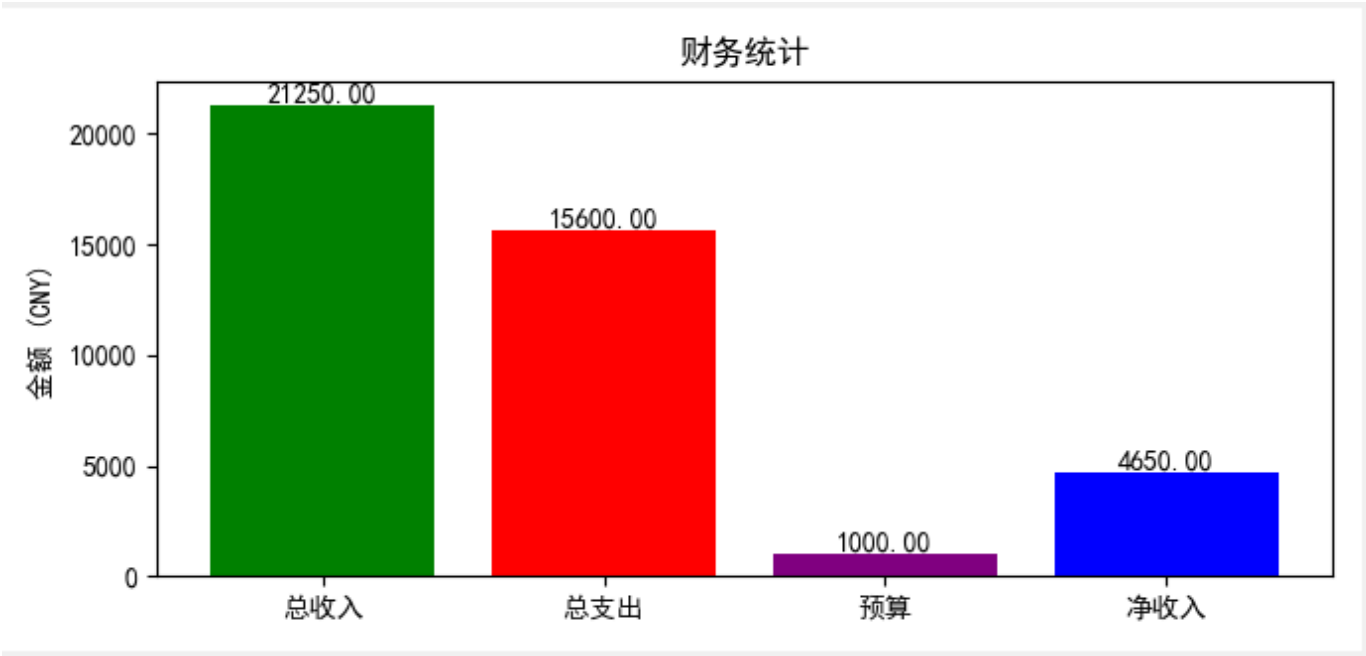
结束日期: 2025-11-10

花费限额: 500

省钱目标: 500


添加计划



删除选中计划








三、git 远程控制结果展示





```
PS D:\LEARNING_RESOURCE\Software Engineering\Projects\Lab03\Python-FinanceTracker> git add .
PS D:\LEARNING_RESOURCE\Software Engineering\Projects\Lab03\Python-FinanceTracker> git commit
[main 1dc55db] CODE: COMPLETE
14 files changed, 1177 insertions(+), 970 deletions(-)
create mode 100644 UML/UML_Activity.png
create mode 100644 UML/UML_Activity.puml
create mode 100644 UML/UML_Class.png
create mode 100644 UML/UML_Class.puml
create mode 100644 UML/UML_Component.png
create mode 100644 UML/UML_Component.puml
create mode 100644 UML/UML_Sequence.png
create mode 100644 UML/UML_Sequence.puml
create mode 100644 UML/UML_Sequence_Exchange_Rate_Management.png
create mode 100644 UML/UML_Sequence_Exchange_Rate_Management.puml
create mode 100644 UML/UML_Use_Case.png
create mode 100644 UML/UML_Use_Case.puml
rename Finance-Tracker.py => code/Finance-Tracker.py (97%)
create mode 100644 code/finance_data.json
PS D:\LEARNING_RESOURCE\Software Engineering\Projects\Lab03\Python-FinanceTracker> git push
fatal: unable to access 'https://github.com/GoldenRiver/Python-FinanceTracker.git/': Recv failure: Connection was reset
PS D:\LEARNING_RESOURCE\Software Engineering\Projects\Lab03\Python-FinanceTracker> git push
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 20 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (18/18), 197.41 KiB | 17.95 MiB/s, done.
Total 18 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/GoldenRiver/Python-FinanceTracker.git
 44d3614..1dc55db main -> main
PS D:\LEARNING_RESOURCE\Software Engineering\Projects\Lab03\Python-FinanceTracker> |
```

 **Python-FinanceTracker** Public


 Pin  Watch 0



 main  1 Branch  0 Tags




  Code



 GoldenRiver CODE: COMPLETE	1dc55db · 1 minute ago	 2 Commits
 UML	CODE: COMPLETE	1 minute ago
 code	CODE: COMPLETE	1 minute ago







报告没写完，一会加报告（

 **Python-FinanceTracker** Public

 Pin  Watch 0

 main  1 Branch  0 Tags

  Code

 GoldenRiver DOC: ADD REPORT	b40b734 · now	 3 Commits
 UML	CODE: COMPLETE	4 minutes ago
 code	CODE: COMPLETE	4 minutes ago
 实验报告.md	DOC: ADD REPORT	now
 实验报告.pdf	DOC: ADD REPORT	now

ok现在有报告了

四、AI帮助我做了什么？

1. 分析主要功能 其实是实验一中甲乙双方对话中借助AI总结整理部分功能列表。在这里提一下。
2. 如何生成uml图？ 生成UML图的时候，因为对UML生成工具使用不熟练，不能快速上手生成多种类型的uml图，使用AI帮我生成了uml图生成方法。vscode中的plainUML并不能生成多种UML图，下载最新版的UML生成工具，了解UML图语法后解决了“如何生成多类UML图的问题”。
3. 给我的代码添加注释。 因为第三次实验距离第二次实验已经两个星期了，所以我已经对我的代码有点陌生了，而且我原本的代码可读性比较差，基本上对着GUI搭积木急就而成的，这也让我在重新阅读时感到吃力。尤其是tkinter库API中冗长的参数列表，在不分行的时候极难阅读，虽然能大致判断代码的作用，但是难以逐个参数调bug。因此我首先使用AI将代码功能注释，把函数调用分行，并让应用了更合理的代码风格（下面有提到），这样提高了我上手的速度（虽然后来重构以后传参更加规范，这些工作被覆盖了）。

AI 56% ,

4. 选择一个合适的架构 原版程序一个FinanceTrackerAPP包揽所有，代码耦合度超高，架构非常不合理，也非常难以拓展。我使用AI搜索合适的架构方式，AI提供了包括分层架构、微服务架构、事件驱动架构等多种架构方式。因为本次实验中我需要实现GUI，并且维护一个具体的账本，因此我使用了分层架构中的MVC架构。
5. 应用合适的代码风格 类名使用驼峰、函数名使用小写+下划线、变量名不适用驼峰。一些行可以拆开，一些地方需要添加足够的空格等等。因为代码比较长，所以让AI自动帮我修改并添加注释，省时省力省心。
6. 应用逻辑和代码 编写程序的过程并不是一帆风顺。第一次编写代码的时候东拼西凑，需要的窗口就往里添加，需要的逻辑就就地填写函数，总体来说BUG的出现不算频繁。但是使用新的架构后，因为对业务逻辑的不熟悉，频繁出现出乎预料的bug，以及很多因为工作量庞大导致的疏忽。下面列举一些有代表性的bug
 - 更好的errlog：本代码最初完全没有errlog，导致程序调试十分痛苦，在更新预算功能时出现难以调试的bug，程序在各种地方崩溃。于是交给AI，描述了预算功能需要实现的大概内容，并要求AI将代码中的各个流程的log补全。之后更新时遇到的小bug可以自己修复。
 - 变量名冲突：更新预算功能的时候，entry_type和type变量名出现冲突，导致账目统计的matplotlib报错。自己手动修改不全，交给AI修改。
 - 回调函数绑定：没有理顺代码执行顺序，导致视图初始化时间过早，此时回调函数都是空函数，导致按下任何键都没有反映。自己检查一段时间无果后交给AI，AI识别出来后进行修改。