

**Python程序设计课程设计**

**题 目**： 《银行管理系统》

**专业班级**：B2023计算机科学与技术1班

**学 院**： 大数据与计算机学院

小组成员

**学号**：202302150031 **姓名**： 魏菀材

**学号**：202302150057 **姓名**： 石松洲

**指导教师**： 陈海华

2024年 7 月任务分工情况说明

|  |  |  |
| --- | --- | --- |
|  | 姓名 | 分工 |
| 组长 | 石松洲 | 编写数据存储和后端模块、代码重构与优化、软件测试与BUG修复、文档撰写 |
| 组员 | 魏菀材 | 前端GUI设计与代码编写、软件测试与BUG修复、文档撰写、绘制流程图 |

目录

[第1章 需求分析 1](#_Toc170928845)

[1.1 项目概述 1](#_Toc170928846)

[1.2 系统功能需求 1](#_Toc170928847)

[1.2.1 账户管理 1](#_Toc170928848)

[1.2.2 账户交易 1](#_Toc170928849)

[1.2.3 账户安全 1](#_Toc170928850)

[1.2.4 用户界面 2](#_Toc170928851)

[1.2.5 系统性能和可靠性 2](#_Toc170928852)

[1.2.6 可用性 2](#_Toc170928853)

[第2章 总体设计 3](#_Toc170928854)

[2.1 系统功能结构 3](#_Toc170928855)

[2.2 系统功能模块 3](#_Toc170928856)

[2.2.1 账户管理模块 3](#_Toc170928857)

[2.2.2 业务模块 3](#_Toc170928858)

[2.3 系统开发环境 4](#_Toc170928859)

[2.4 技术栈 4](#_Toc170928860)

[第3章 详细功能设计与实现 5](#_Toc170928861)

[3.1 阐述程序已实现的功能 5](#_Toc170928862)

[3.1.1 首页功能设计 5](#_Toc170928863)

[3.1.2 注册开户功能设计 7](#_Toc170928864)

[3.1.3 各窗口功能设计 9](#_Toc170928865)

[3.1.4 登入功能设计 12](#_Toc170928866)

[3.1.5 取款功能设计 14](#_Toc170928867)

[3.1.6 存款功能设计 16](#_Toc170928868)

[3.1.7 转账功能设计 18](#_Toc170928869)

[3.1.8 查询余额功能设计 20](#_Toc170928870)

[3.1.9 修改密码设计 22](#_Toc170928871)

[3.1.10 切换用户状态功能设计 24](#_Toc170928872)

[3.1.11 查询交易明细 26](#_Toc170928873)

[3.2 前端GUI实现 28](#_Toc170928874)

[3.2.1 主窗口 28](#_Toc170928875)

[3.2.2 登入界面（业务界面） 32](#_Toc170928876)

[3.3 后端接口实现 40](#_Toc170928877)

[3.3.1 数据存储模块实现 40](#_Toc170928878)

[3.3.2 开户功能实现 40](#_Toc170928879)

[3.3.3 登入功能实现 40](#_Toc170928880)

[3.3.4 取款功能实现 40](#_Toc170928881)

[3.3.5 存款功能实现 41](#_Toc170928882)

[3.3.6 转账功能实现 41](#_Toc170928883)

[3.3.7 查询余额功能实现 41](#_Toc170928884)

[3.3.8 修改密码功能实现 41](#_Toc170928885)

[3.3.9 锁定/解锁账户功能实现 41](#_Toc170928886)

[3.3.10 查询交易明细功能实现 41](#_Toc170928887)

[第4章 总结 42](#_Toc170928888)

[4.1 项目成果 42](#_Toc170928889)

[4.1.1 项目开发总结 42](#_Toc170928890)

[4.1.2 项目潜在缺陷和面临的挑战 42](#_Toc170928891)

[4.2 结语 42](#_Toc170928892)

1. 需求分析
   1. 项目概述

广东白云银行系统是一个专为校园环境设计的综合性金融解决方案，旨在通过先进的信息技术提升银行服务的效率和质量。该系统主要面向学生、教职工及其家属使用，提供了一系列功能，以满足他们在日常金融活动中的需求。在该系统中，用户可以进行账户管理、资金转账、存取款操作，以及查询交易明细和账户余额，是一个用以管理个人财务的工具。通过这些功能，不仅提高了银行服务的效率，还为用户创造了一个更加便捷、安全和个性化的金融环境。它鼓励用户积极参与金融管理过程，同时为银行提供了强大的工具来支持他们的服务工作。通过持续的技术创新和用户反馈，该系统致力于成为教育领域中的领先金融服务平台，推动校园金融的现代化和个性化发展。模拟银行管理系统旨在提供一个虚拟平台，用于模拟银行的基本业务操作和管理，包括账户管理、存款取款、转账交易等功能。该系统将帮助用户理解银行业务流程，并提高业务操作的效率和准确性。

* 1. 系统功能需求
     1. 账户管理

账户创建：用户可以自主创建银行账户。

账户信息查看：用户可以查看其账户的详细信息（如余额、账户锁定状态等）。

账户信息修改：用户可以修改其账户信息（如账户密码、账户锁定状态等）。

* + 1. 账户交易

存款：用户可以向其账户存入资金。

取款：用户可以从其账户取出资金。

账户间转账：用户可以将资金转入到其他账户。

* + 1. 账户安全

账户锁定和解锁功能，以防止未授权用户操作。

* + 1. 用户界面

清晰直观的用户操作界面。

实时显示系统时间和个性化欢迎信息。

错误处理和用户提示信息

* + 1. 系统性能和可靠性

系统应具备高可用性和错误恢复能力。

交易处理应快速且准确。

* + 1. 可用性

系统应易于使用，适合所有技术水平的用户。

1. 总体设计
   1. 系统功能结构

|  |
| --- |
|  |

* 1. 系统功能模块

本系统设计上将功能模块抽象为前端模块（GUI相关）、后端模块（处理功能请求）及数据存储模块（将系统信息持久化存储至磁盘），模块之间互相交互完成请求。

* + 1. 账户管理模块

账户管理模块主要提供了开户注册、账户锁定和解锁、修改密码等功能。我们的账户管理模块精心打造了一个用户友好的界面，使用户能够轻松地管理自己的银行账户。

* + 1. 业务模块

主要提供了取款、存款、转账、查询余额、查询明细等功能。我们的业务模块精心打造了一个用户友好的界面，使用户能够轻松地进行日常的银行交易和账户查询。

* 1. 系统开发环境

|  |  |
| --- | --- |
| 操作系统 | Window11 |
| 处理器 | 8th Gen Intel(R) Core(TM) i5-8265U 1.80 GHz |
| 内存 | DDR4-16GB-2400Mhz |
| Python版本 | 3.12.2、3.10.5 |

* 1. 技术栈

Python: 作为主要的编程语言。

Tkinter: Python的标准GUI库，用于创建和管理图形用户界面。

PIL (Pillow): 用于图像处理，主要用于加载和显示图像。

re (正则表达式库): 用于验证用户输入的格式。

1. 详细功能设计与实现
   1. 阐述程序已实现的功能
      1. 首页功能设计

|  |
| --- |
|  |
| 首页功能流程图 |

首页是广东白云银行系统的门户，设计上旨在为用户提供一个直观、易于导航的入口。首页集成了系统的主要功能，包括账户登录、开户注册以及实时的时间模块和个性化信息。通过精心设计的首页，用户可以快速访问所需的服务，同时获得一个概览性的信息展示，增强用户体验。在设计之初我们结合了广东白云学院的配色，主要的颜色为蓝色和白色，为更加融入广东白云学院的特色特地设计了与广东白云银行系统相结合的卡片和标题信息。这一系列的措施凸显了程序与白云学院的紧密联系，让用户感受到程序的专属性，从而提升用户对程序辨识度。

|  |
| --- |
|  |
| 首页界面图 |
|  |

* + 1. 注册开户功能设计

|  |
| --- |
|  |
| 注册开户流程图 |

注册开户功能是广东白云银行系统中的核心功能之一，它允许新用户快速创建自己的银行账户。该功能的设计旨在提供一个简洁、安全且易于操作的用户界面，为确保用户能够无障碍地完成开户流程，在输入开户密码的时候内置数字键盘适配触屏用户使用，与此同时确保所有提交的个人信息得到妥善处理和保护。

注册成功后为用户分配一位账户。

|  |
| --- |
|  |
| 注册成功图 |

* + 1. 各窗口功能设计

用户信息输入的安全措施：

所有输入字段都应进行格式验证和必填验证，确保信息的完整性和准确性。若用户输入不合法的输入项时（例：开户注册功能上密码不足六位整数），则弹出对话框提示用户。

|  |
| --- |
|  |
| 弹出提示错误信息框图 |

以及密码字段应隐藏输入内容，避免旁观者窥视。

|  |
| --- |
|  |
| 密码字段隐藏图 |

模态化窗口：

为防止用户做出一些系统不允许的操作，所有的窗口实现模态化，当弹出窗口时用户不得做一些违规操作。例如打开多个开户注册的窗口。



我们为此还定义了两种键盘一种是如上图所示，输入密码无需用到小数点。在业务的模块里面涉及到小数点的我们就会提供带小数点的数字键盘给用户。

|  |
| --- |
|  |
| 带小数点的数字键盘图 |

为了让用户更直观的操作，我们还进行了特别的优化，用户每次点窗口时都获取显示器的大小通过计算让弹出的窗口始终保持在正中央。

|  |
| --- |
|  |
| 每次都显示在正中央图 |

* + 1. 登入功能设计

|  |
| --- |
|  |
| 登入功能流程图 |

登入功能是广东白云银行系统中的关键组成部分，它允许已注册用户通过验证其凭据来访问其个人银行账户。该功能的设计旨在确保安全、快速且用户友好，从而提供无缝的用户体验，同时保护用户的账户安全免受未授权访问。

在3.1.2开户功能中注册之后用户可以点击来登入账户和密码来登入。当用户填写正确时通过后端验证即可正常做下一步业务。

|  |
| --- |
|  |
| 登入成功图 |

用户成功登入之后我们在业务界面，在左下角实时显示时间的模块。在右上角设计了欢迎模块，根据用户当地的时间显示相应的欢迎语，和显示相应登入的用户账户后4位。

|  |
| --- |
|  |
| 业务界面 |

* + 1. 取款功能设计

|  |
| --- |
|  |
| 取款功能流程图 |

取款功能是银行系统中最频繁使用的功能之一。当用户注册，成功登入之后，允许用户从其银行账户中提取现金，用户首先需要输入希望取款的金额，这可以通过一个简洁的数字输入界面完成，用户可以直接输入金额。

|  |
| --- |
|  |
| 取款成功图 |
|  |

在用户输入取款金额后，系统会自动验证用户的账户状态和余额是否允许进行此次取款。如果账户被锁定或余额不足，系统将显示相应的错误消息，并阻止交易继续。例如用户被锁定。

|  |
| --- |
|  |
| 取款失败图 |

* + 1. 存款功能设计

|  |
| --- |
|  |
| 存款功能流程图 |

与取款功能类似，存款功能它允许用户将资金存入个人银行账户。用户需要输入希望存入的金额。这通常通过一个数字键盘界面完成，用户可以快速输入。用户在输入所有必要信息后并确认无误后，可以提交存款请求。

|  |
| --- |
|  |
| 存款成功图 |

当然，当用户输入一些非法值时，系统将显示相应的错误消息，并阻止交易继续。

|  |
| --- |
|  |
| 存款失败图 |

* + 1. 转账功能设计

|  |
| --- |
|  |
| 转账流程图 |

转账功能，顾名思义它允许用户将资金从自己的账户转移到另一个账户。用户需要输入接收方的账号和转账金额。通过一个表单界面完成，在用户输入转账金额后，系统会自动验证用户的账户状态和余额是否允许进行此次转账。

例：我们给尾号0002用户转账

|  |
| --- |
|  |
| 转账成功图 |

如果账户被锁定、余额不足或接收方账户不存在，系统将显示相应的错误消息，并阻止交易继续。例：用户不存在

|  |
| --- |
|  |
| 转账失败图 |

* + 1. 查询余额功能设计

|  |
| --- |
|  |
| 查询余额流程图 |

查询余额，允许用户快速查看其银行账户的当前余额。系统应能够实时处理余额查询请求，并显示最新的账户余额，和用户当前的状态。

|  |
| --- |
|  |
| 查询余额图 |

* + 1. 修改密码设计

|  |
| --- |
|  |
| 修改密码流程图 |

修改密码，允许用户更新他们的账户密码。用户更改密码之前，必须首先验证其身份。需要用户输入旧密码、输入新的密码，并且需要再次输入以确认无误。才能成功更改密码。

用户在输入新密码并满足所有要求后，提交请求。系统验证新密码与当前密码不同，并处理密码更改。

|  |
| --- |
|  |
| 密码成功更改图 |

若用户输入的旧密码错误，则弹出提示反馈给用户。

|  |
| --- |
|  |
| 密码更改失败图 |

* + 1. 切换用户状态功能设计

|  |
| --- |
|  |
| 切换用户状态流程图 |

切换用户状态功能是广东白云银行系统中的一个关键安全特性，使用切换按钮。让用户可以选择锁定或解锁其账户，并提示他们完成状态切换。操作完成后，系统应通知用户状态已成功更新。

|  |
| --- |
|  |
| 切换用户状态成功图 |

为了防止误操作，我们还对此进行了优化，若用户处于相同状态时，重复该状态的切换则弹出错误信息提示用户。

例如用户处于锁定状态再次锁定：

|  |
| --- |
|  |
| 切换用户状态失败图 |

* + 1. 查询交易明细

|  |
| --- |
|  |
| 查询交易明细流程图 |

查询交易明细为用户提供一个清晰的列表视图，展示用户账户的所有交易记录，包括源账户、目的账户、操作和金额信息。用户可以轻松理解他们的交易历史。为了能让上述信息能够显示全，我们特此设计了X轴和 Y轴滚动条功能。

|  |
| --- |
|  |
| 查询交易明细图 |

* 1. 前端GUI实现

本项目的前端GUI实现主要依赖于tkinter、Pillow库构建图形用户界面为用户提供一个直观、易用的界面来进行各种银行操作，如开户、登录、存款、取款、转账等业务操作。

* + 1. 主窗口

主窗口是用户与系统的交互的最开始的窗口，提供了用户进行开户注册和登入的入口。

主窗口布局：

* 使用Tk()来建立主窗口一个顶级窗口作为整个程序的主窗口。
* 使用Photoshop软件来设计，主窗口的顶部标题和发行图片以及专为该系统设计的银行卡。
* 通过使用Canvas画布组件设置背景图片，增加了视觉效果和层次。
* 在背景图片画布上建立文本为“您的财富由我们掌控”作为该系统的口号置于画面的中下方以及建立文本“请插入你的卡片”提示在背景的正中央。
* 左下角动态显示当前时间。
* 主窗口的左侧主窗口中有两个按钮，分别是“开户注册（Create Account）”和“登录（Sign In）”，使用Button组件创建。
* 每个模块的弹出输入框都会支持数字键盘来输入。

1. 动态时间显示：

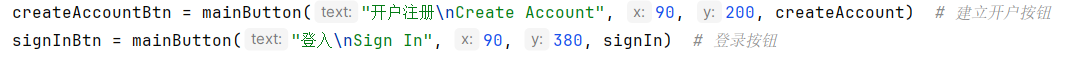
定义一个函数名为addTime()，在addTime()函数中先用time库的time.strftime函数返回以可读字符串表示的当地时间赋值给一个名为currentTime的变量名，再用Label控件来初始化该文本框。

在addTime()函数里面再嵌套一个名为update ()的函数用于动态更新Label控件中的text参数，使用after()方法在主循环中延迟毫秒（1000ms）后调用回调函数一次。代码如下。



1. 开户与登录按钮：

每个按钮都绑定了一个事件处理函数，当用户点击按钮时，会弹出相应的对话框来进行后续操作。



前端通过一个 Button 按钮展示开户功能，用户点击按钮后弹出对话框，用户输入密码后点击确定，前端校验密码格式正确后调用后端 bank.createAccount( )接口添加新账号。

前端通过一个 Button 按钮展示登入功能，用户点击后弹出对话框提示用户输入账号密码，点击确定后调用后端 bank.verify()判断密码是否正确。

|  |
| --- |
|  |

1. 数字键盘

为了方便用户输入，系统提供了一个虚拟的数字键盘。用户可以通过点击按钮来输入数字，也可以进行删除和清空操作。

定义了两个数字键盘给需要输入小数点的输入框和不需要小数点的控件。

代码如下：

|  |
| --- |
|  |

每个按钮都绑定了一个事件处理函数，用于处理用户的点击操作。

代码如下：

|  |
| --- |
|  |

* + 1. 登入界面（业务界面）

业务界面是用户在成功登录后进入的操作区域，它提供了一系列银行业务功能，包括取款、存款、转账、修改密码、查询余额和交易明细等。该界面为用户提供一个集中的、以便快速选择和执行所需的银行业务。

业务界面布局：

* 将不同的业务功能按照逻辑分组排列，以方便用户识别和选择。
* 每个业务功能都通过一个按钮来表示，按钮上有业务名称和图标，以提高识别度。
* 在界面顶部，根据当前时间和用户的账户信息，显示个性化的欢迎信息，信息包括用户的账户尾号和根据时间段变化的问候语。
* 系统支持多种交易操作，如存款、取款和转账。每种操作都有相应的界面和逻辑处理。
* 并且涉及到输入的都会内置数字键盘的控件。
* 每种交易操作都通过一个单独的模态对话框来处理。
* 交易操作调用银行系统的API，并根据操作结果给出反馈。

1. 取款功能

前端通过一个 Button 按钮展示取款功能，用户点击按钮后弹出对话框，当然在前面提到对话框内置了数字键盘。输入框用户输入金额后点击确定，前端校验金额格式正确后调用后端bank.withdrawal()为用户减少余额。

代码如下：

|  |
| --- |
|  |

1. 存款功能

与上述取款功能类似，输入框用户输入金额后点击确定，前端校验金额格式正确后调用后端的API即bank.makeDeposit()接口，不同的是这是为用户增加余额。

代码如下：

|  |
| --- |
|  |

1. 转账功能

点击转账Button控件后，弹出对话框输入目的账号和目标金额，用户完成输入后点击确认按钮，前端会判断目标账号是否存在、转账金额是否大于余额、输入的值合不合法，判断无误后调用bank.transfer()API来为目的账户增加余额，为源账户减余额。

代码如下：

|  |
| --- |
|  |

1. 查询余额功能

点击查询余额的Button控件后，弹出messagebox提示框，再调用后端即API bank.getLockState()来获取用户的锁定状态，和调用bank.getBalance()，API获取用户的余额信息然后报告到messagebox上。

代码如下：

|  |
| --- |
|  |

1. 修改密码功能

点击修改密码的Button控件后， 弹出对话框输入旧密码、新密码和确认新密码，用户完成输入后点击确认按钮，前端会判断输入的密码合不合法、新密码和确认密码不一致、新旧密码不相同和旧密码错误，判断无误后调用后端API，即bank.resetPassword()来为用户修改密码。

代码如下：

|  |
| --- |
|  |

1. 切换用户状态功能（锁定/解锁账户）

点击锁定/解锁账户的Button控件后，弹出对话框，先用后端API接口bank.getLockState（）判断账户是否被锁定来初始化Checkbutton的状态，再创建一个BooleanVar变量来跟踪Checkbutton的状态，当勾选对话框后会判断对话框是否被选中然后修改Label的文本到对应所选择的状态。当用户点击确认按钮后，判断设定的状态和用户是否合法，无误后调用后端的API即bank.setLocked()来为用户切换账户状态。

代码如下：

|  |
| --- |
|  |

1. 查询交易明细功能

点击查询交易明细的Button控件后，弹出对话框，在对话框上创建Frame，再Frame上创建一个Text控件。为了避免账户交易信息显示不全加入了滚动条的功能来显示更多的交易明细。调用后端API即bank.getTransactions()返回用户的源、目的账户、操作和金额到Text控件上。

代码如下：

|  |
| --- |
|  |

* 1. 后端接口实现

后端部分被剥离到 bank.py 文件中，将银行系统后台抽象为Bank类，提供全局的bank实例对象，并向外暴露接口以供前端调用。账户数据结构被抽象为Account类，交易明细被抽象为Transaction类。

|  |
| --- |
|  |
| 后端类与数据结构定义 |

程序启动时，初始化 bank 对象，调用数据存储模块加载 data.csv 中存储的账户信息，加载 transaction.csv 中存储的所有交易明细信息；遍历所有账号获得已被分配出去的账号的最大值。

|  |
| --- |
|  |
| 数据加载 |

* + 1. 数据存储模块实现

数据存储模块被剥离到 db.py 中，使用 csv 格式存储数据，提供两个对外暴露的类 Database 和 ListDatabase 以供调用。

Database：提供键值对形式的数据库，读取指定 csv 文件并将每行用逗号分割，第一项当成键，后续项当成值放进列表内。对外提供 get() 方法根据键获得值，提供 keys() 方法支持遍历所有数据项，提供 set() 方法插入或更新对应键所对应的值，并持久化保存到对应的 csv 文件中。

|  |
| --- |
|  |
| Database 数据加载逻辑 |

ListDatabase：提供按行划分的数据库，没有键，读取指定 csv 文件并将每行用逗号分割并放进列表内。提供 items() 方法用于遍历所有行，提供 insert() 方法插入新的行并持久化保存到对应的 csv 文件中。

|  |
| --- |
|  |
| ListDatabase 数据加载逻辑 |

* + 1. 开户功能实现

后端将目前最大的账号数字进行自增以分配出新的账号，创建新的 Account 对象，初始化密码、金额等属性并插入到 accounts 表中，调用数据存储模块将新创建的账号保存到文件中，并将新分配的账号返回给前端。

|  |
| --- |
|  |
| 开户功能实现 |

* + 1. 登入功能实现

后端接收前端传来的账号密码，先查询 accounts 表判断该账号是否存在，存在则比对密码并返回比对结果，不存在则直接返回 False 表示账号或密码错误。

|  |
| --- |
|  |
| 账号密码验证逻辑 |

* + 1. 取款功能实现

后端接收前端传来的当前账户和金额，根据账号查询到当前余额进行比对，如果账户已被锁定、金额非法或金额大于余额则抛出异常给前端，否则将余额扣减掉取款金额（这里需要使用 Python Decimal 高精度数学运算以避免浮点数精度损失），持久化保存到文件中，记录交易信息，将当前余额返回给前端。

|  |
| --- |
|  |
| 取款逻辑 |

* + 1. 存款功能实现

后端接收前端传来的当前账户和金额，如果金额非法则抛出异常给前端，否则获取账户当前余额并加上金额（使用 Python Decimal 高精度数学运算以避免浮点数精度损失），保存到文件中，记录交易信息，将当前余额返回给前端。

|  |
| --- |
|  |
| 存款逻辑 |

* + 1. 转账功能实现

后端接收前端传来的当前账户、目标账户和金额，根据账号查询到当前余额进行比对，如果当前账户已被锁定、目标账户不存在、金额非法或金额大于余额则抛出异常给前端，否则扣减当前账户余额、增加目标账户余额（使用 Python Decimal 高精度数学运算以避免浮点数精度损失），保存到文件中，记录交易信息，将当前账户余额返回给前端。

|  |
| --- |
|  |
| 转账逻辑 |

* + 1. 查询余额功能实现

后端接收前端传来的当前账号，从accounts表中查询到对应 Account 对象，返回其上的余额属性给前端。

|  |
| --- |
|  |
| 查询余额逻辑 |

* + 1. 修改密码功能实现

password 是 Account 对象其上的一个属性，后端接收前端传来的当前账号，从accounts表中查询到对应 Account 对象，重设 password 属性并保存到文件。

|  |
| --- |
|  |
| 修改密码逻辑 |

* + 1. 锁定/解锁账户功能实现

locked 是 Account 对象其上的一个布尔值属性。后端接收前端传来的当前账号，从accounts表中查询到对应 Account 对象，重设 locked 属性并保存到文件。当发生取款、转账等操作时会优先检查 locked 属性，以确保被锁定的账户无法发生让余额减少的交易。

|  |
| --- |
|  |
| 账户锁定/解锁逻辑 |

* + 1. 查询交易明细功能实现

transactions.csv 是用于保存交易明细的一个文件，当每个交易发生时，交易明细会被保存到其中。后端接收前端传来的用户当前账号，查询到所有和此账户有关的交易，并返回列表到前端。

|  |
| --- |
|  |
| 查询交易明细逻辑 |

1. 总结
   1. 项目成果

我们花费了一周的时间，使用 Python 语言编写了一个简单易用的银行管理系统，使用 Git 作为版本管理系统，并将项目上传到了 Github 以协作编程。经过多轮测试，我们的系统表现出了良好的稳定性和可用性，同时拥有简单易懂、美观大方的用户页面，任何用户都能轻松上手操作。

* + 1. 项目开发总结

这次开发过程对我们两个人都是一次宝贵的经历，让我们对 Python 这门语言有了更深的了解，也让我们丰富了实战中协作编程的经验，比如解决 git 合并冲突的经历。

* + 1. 项目潜在缺陷和面临的挑战

本项目取得了一定的成果，但仍然有可以优化的地方，如：用户界面可以设计的更为美观；可以添加更多的高级功能，如添加管理员功能等；账户数据使用明文存储在文件中，存在一定的安全风险，存储前可以对其进行加密或哈希处理。

* 1. 结语

本项目的开发过程让我们提升了自己的 Python 技能，也培养了用编程思维抽象化复杂问题的能力，还获得了协作开发的宝贵经历。这些知识一定能让我们在未来的工作学习中更加强大！