

# 个人财务管理系统

陈楠，池承运，刘潇峰

电子信息与电气工程学院，上海交通大学，上海，200240

**概述:** 个人财务系统通过记录和管理收入、支出、储蓄等财务数据，帮助我们清晰了解资金流动、优化资源分配，实现财务目标。它不仅提升理财效率，还能避免过度消费和潜在的财务风险，是实现经济独立和长期财富积累的重要工具。本文档描述一个基于 Django 框架的个人财务管理系统的数据库设计。该系统主要包含四个模块：账本、账户、收入支出统计、良好可视化统计界面。项目源码链接：<https://github.com/Arcs-ur/PersonalFinanceNIS>。

**技术框架:** 整体实现基于网络前后端架构。采用 CoreUI-Bootstrap 为前端，用于实现响应式布局和组件化的界面设计；采用 Django 为后端框架，用于处理业务逻辑与数据库交互；前后端接口设计采用 Django REST framework 实现，利用后端来提供 RESTFUL API；在数据库方面，使用使用 Django ORM 与数据库 SQLite 交互完成；数据可视化采用 Chart.js 绘制图表。

**设计亮点:** 以可视化饼图的方式更加直观形象的展示用户的收支；账本 (AccountBook)、账户 (FundAccount) 等逻辑结构的的设计更加符合现实生活的真实使用场景；运行在本地可以让用户的数据隐私得到保护。

**模块介绍:** 详见本文第八部分：功能模块设计与实现、第九部分：系统演示。

**人员分工:** 详见本文第十一部分：整体总结的第一小节：分工。

# 目录

<b>1</b>	<b>需求分析</b>	<b>4</b>
1.1	信息要求 . . . . .	4
1.2	处理要求 . . . . .	4
1.3	安全性与完整性需求 . . . . .	4
1.3.1	安全性需求 . . . . .	4
1.3.2	完整性要求 . . . . .	5
<b>2</b>	<b>数据流图</b>	<b>5</b>
2.1	顶层数据流图 . . . . .	5
2.2	详细数据流图 . . . . .	6
<b>3</b>	<b>数据字典</b>	<b>6</b>
3.1	AccountBook 表 . . . . .	6
3.2	FundAccount 表 . . . . .	6
3.3	CustomUser 表 . . . . .	7
3.4	ExpenseEntry 表 . . . . .	7
3.5	IncomeEntry 表 . . . . .	7
3.6	MonthlyBudget 表 . . . . .	8
3.7	处理过程：绘制支出统计图表 . . . . .	8
3.8	处理过程：周（自定义时间）收入/支出图表 . . . . .	8
<b>4</b>	<b>概念设计</b>	<b>8</b>
4.1	实体 . . . . .	8
4.2	联系 . . . . .	8
4.3	E-R 图展示 . . . . .	9
<b>5</b>	<b>逻辑设计</b>	<b>9</b>
5.1	E-R 图到关系模型的转换 . . . . .	9
5.2	系统结构图 . . . . .	10
<b>6</b>	<b>物理设计</b>	<b>11</b>
6.1	存储安排 . . . . .	11
6.1.1	数据库选择 . . . . .	11
6.1.2	数据库模式 . . . . .	11
6.2	方法选择 . . . . .	11

6.2.1	数据索引 . . . . .	11
6.2.2	事务管理 . . . . .	11
6.3	存储路径 . . . . .	11
6.4	模块设计 IPO 表 . . . . .	12
<b>7</b>	<b>技术架构</b>	<b>12</b>
<b>8</b>	<b>功能模块设计与实现</b>	<b>12</b>
8.1	账本、账户管理模块 . . . . .	12
8.2	收支管理、统计模块 . . . . .	13
8.3	仪表盘模块 . . . . .	14
8.4	接口设计 . . . . .	14
<b>9</b>	<b>系统演示</b>	<b>21</b>
9.1	注册登录与身份认证 . . . . .	21
9.2	使用界面 . . . . .	22
9.3	账本页面 . . . . .	23
9.4	账户页面 . . . . .	24
9.5	收支页面 . . . . .	25
9.6	统计页面 . . . . .	26
9.7	基于 Chart.js 饼状图统计页面 . . . . .	27
<b>10</b>	<b>流程图</b>	<b>29</b>
<b>11</b>	<b>整体总结</b>	<b>29</b>
11.1	分工 . . . . .	29
11.2	任务完成情况 . . . . .	30
11.3	测试与改进 . . . . .	31
11.3.1	功能测试 . . . . .	31
11.3.2	性能优化 . . . . .	31
11.3.3	未来改进方向 . . . . .	31

# 1 需求分析

## 1.1 信息要求

- a. 用户：用户靠用户名和密码登录，系统会记录用户对应的头像和电子邮件地址。一个用户可以有多个账户、多个账本。
- b. 账户：账户有唯一 ID，且每个账户有一个余额字段，用于存储账户的当前余额。一个账户可被添加到多个账本。（比如张三的支付宝）
- c. 账本：账本有唯一 ID，一个账本下可以有多个账户。（比如李四的 2024 年度账本）
- d. 收入：每笔收入需要记录收入日期、金额、币种、来源、计入账户、计入账本。
- e. 支出：每笔支出需要记录支出日期、金额、币种、类别、支出账户、支出账本。

## 1.2 处理要求

- a. 用户管理：创建和管理用户。
- b. 账本管理：创建和管理账本。
- c. 账户管理：创建和管理账户。
- d. 收入支出统计：用户可以很方便地统计某个账本的收入支出统计，可以对支出进行分类统计，对收入的来源进行分类统计。
- e. 统计图表表示：用户可以通过图表来更加直观得查看支出收入统计，包括“一周收支统计”与“自定义时间收支统计”
- f. 可扩展性：可以根据实际需求进一步扩展，如管理账本和账户之间的交易记录等。

## 1.3 安全性与完整性需求

### 1.3.1 安全性需求

#### (1) 数据保护

加密：所有存储的敏感数据需要采用强加密算法加密。

访问控制：用户账户和角色管理应限制对数据的访问。

隐私保护：遵守隐私法规，避免不必要的数据采集。

#### (2) 身份验证

强密码策略：要求用户设置复杂密码并定期更新。

#### (3) 安全日志与监控

日志记录：记录关键操作和安全事件。

异常监控：实时监测并报告异常活动。

### 1.3.2 完整性要求

#### (1) 数据一致性

数据校验：在输入和存储数据时实施校验规则，防止非法或错误的数据进入系统。

冗余与同步：在多模块之间保持数据同步，避免冲突。

#### (2) 数据备份与恢复

自动备份：定期对数据库进行完整备份，支持增量备份

容灾恢复：提供灾难情况下的数据恢复能力，确保最小化数据丢失

#### (3) 审计与验证

可审计性：系统提供日志和变更记录，支持用户和管理员查看数据历史和操作。

完整性检查：定期对数据进行完整性检查，以检测是否存在损坏或篡改。

## 2 数据流图

### 2.1 顶层数据流图

用户与系统交互，进行财务数据的录入、查询和管理；系统处理用户要求，并于数据库进行数据交换；系统生成各类报表供用户查看和分析。

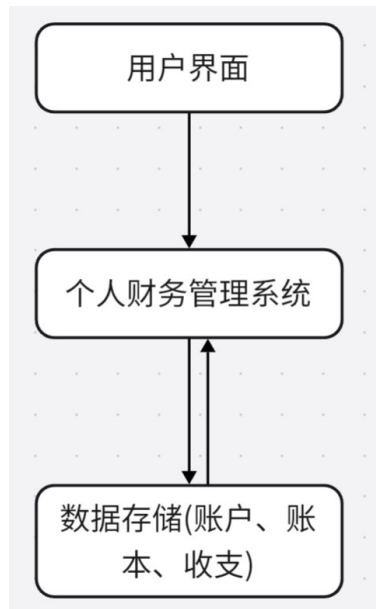


图 1: 顶层数据流图

2.2 详细数据流图

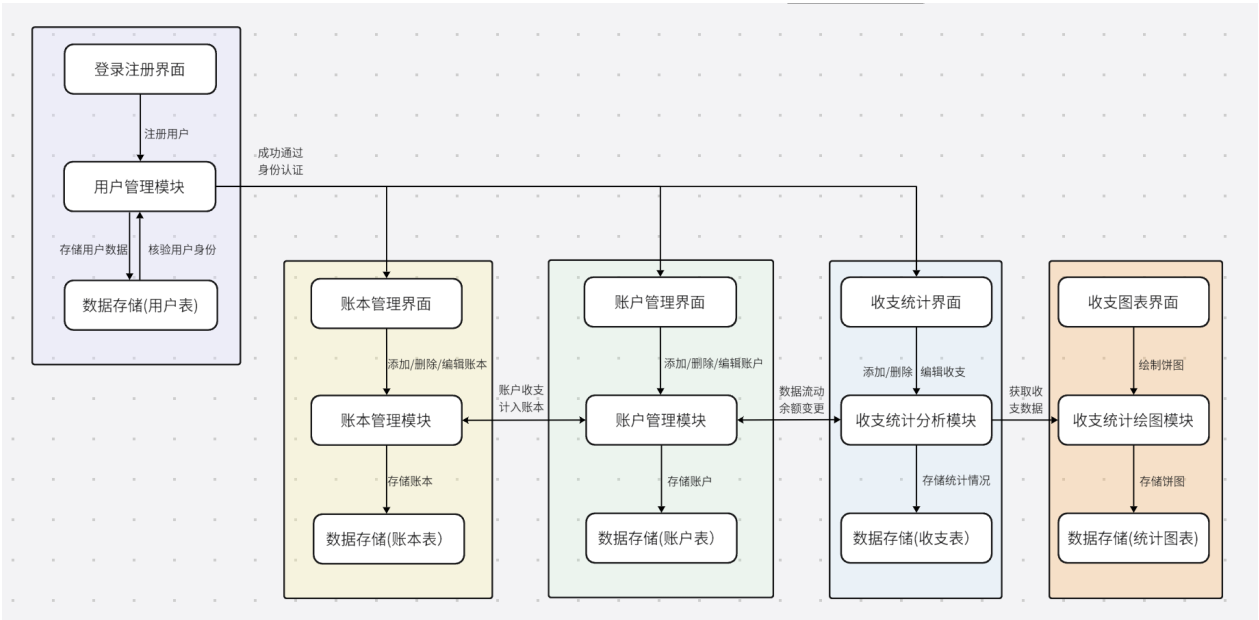


图 2: 详细数据流图

3 数据字典

3.1 AccountBook 表

数据项	含义说明	数据类型	约束条件
name	账本名称	Charfield(字符串)	最大长度为255位
account_book_id	唯一账本ID	UUIDField(唯一标识符-字符串)	取值唯一、主码，未被指定值时由uuid.uuid4()生成随机UUID值

图 3: AccountBook 数据字典

3.2 FundAccount 表

数据项	含义说明	数据类型	约束条件
name	账户名称	字符串	最大长度为255位
balance	账户余额	十进制数值	数字总位数为10，小数点后的位数为2
account_id	唯一账户ID	字符串-唯一标识符	取值唯一、主码，未被指定值时由uuid.uuid4()生成随机UUID值
account_book	关联到账本	字符串	与fund_accounts关联，级联删除

图 4: FundAccount 数据字典

### 3.3 CustomUser 表

数据项	含义说明	数据类型	约束条件
username	用户名	字符串	最大长度255位, 取值唯一
password	密码	字符串	最大长度为128位
email	存储电子邮件地址	电子邮件地址	非空
first_name	名字	字符串	最大长度30位, 可以留空
last_name	姓氏,以上均继承自AbstractUser	字符串	最大长度150位, 可以留空
groups	所属组	多对多关系对象类型	与customer_set关联
user_permissions	特定权限	多对多关系对象类型	与customer_user_permissions关联
phone_number	电话号码	字符串	最大长度15, 非空, 可存NULL值
profile_picture	头像图片	图像文件	存储位置为'profile_pictures/', 可留空, 可取NULL值
level	等级	整数	未指定值则取1

图 5: CustomUser 数据字典

### 3.4 ExpenseEntry 表

数据项	含义说明	数据类型	约束条件
expense_id	支出条目ID	自增的整数类型	主码
expense_category	支出种类, 用choices限制选择	字符串	最大长度50, 在expense_categories中选择, 默认值为'others'
amount	支出金额	十进制数值	数字总位数为10, 小数点后的位数为2
currency	货币种类, 用choices限制选择	字符串	最大长度10, 在currencies中选择, 默认值为CNY
expense_time	支出时间	日期和时间类型	
include_in_balance	是否计入收支	布尔值	默认为False
include_in_budget	是否计入预算	布尔值	默认为False
reimbursement	是否报销	布尔值	默认为False
account	关联的账户ID	字符串-外码	与expense_entries关联, 级联删除
account_book	关联的账本ID	字符串-外码	与expense_entries关联, 级联删除

图 6: ExpenseEntry 数据字典

### 3.5 IncomeEntry 表

数据项	含义说明	数据类型	约束条件
income_id	收入条目ID	自增的整数类型	主码
income_category	收入种类, 用choices限制选择	字符串	最大长度50, 在income_categories中选择, 默认值为'others'
amount	收入金额	十进制数值	数字总位数为10, 小数点后的位数为2
currency	货币种类, 用choices限制选择	字符串	最大长度为10, 在currencies中选择, 默认值为CNY
income_time	收入时间	日期和时间类型	
include_in_balance	是否计入收支	布尔值	默认为False
account	关联的账户ID	字符串-外码	与income_entries关联, 级联删除
account_book	关联的账本ID	字符串-外码	与income_entries关联, 级联删除

图 7: IncomeEntry 数据字典

### 3.6 MonthlyBudget 表

数据项	含义说明	数据类型	约束条件
year	年份	整数	
month	月份(1-12)	整数	
total_budget	总预算	十进制数值	数字总位数为10, 小数点后位数为2
currency	货币种类, 用choices限制选择	字符串	最大长度为10, 在CURRENCIES中选择, 默认为CNY
remaining_budget	剩余预算	十进制数值	数字总位数为10, 小数点后位数为2

图 8: MonthlyBudget 数据字典

### 3.7 处理过程：绘制支出统计图表

说明：绘制所选日期内的各类支出饼图

输入：开始日期、截止日期

输出：支出饼图

处理：统计所选日期内的各类支出的金额，并计算总支出金额，然后绘制饼图

### 3.8 处理过程：周（自定义时间）收入/支出图表

说明：绘制本周收入/支出图表（按种类划分）

处理：统计本周的各类收入/支出的金额，并计算总收入/支出金额，然后绘制饼图

## 4 概念设计

### 4.1 实体

**用户** (用户 ID, 密码, 电子邮件地址, 电话号码, 权限, 等级)

**账本** (账本 ID, 账本名, 用户 ID)

**账户** (账户 ID, 账户名, 账户余额, 用户 ID)

**收入记录** (条目 ID, 种类, 金额, 货币, 时间, 是否计入收支, 账户 ID, 账本 ID, 用户 ID)

**支出记录** (条目 ID, 种类, 金额, 货币, 时间, 是否计入收支, 是否计入预算, 是否报销, 账户 ID, 账本 ID, 用户 ID)

### 4.2 联系

一个用户可以有多个账户，一个账户只属于一个用户；一个用户可以有多个账本，一个账本只属于一个用户；一个账本可以记录多个账户的收支，一个账户可以有多个账本记



录；一个 (账户, 账本) 可以有多条收入、支出记录，一条收入、支出记录对应一个 (账户, 账本)。

## 总结

用户与账户、账本都是一对多联系；账户与账本之间是多对多联系；(账户, 账本) 与收入记录、支出记录是一对多联系。

### 4.3 E-R 图展示

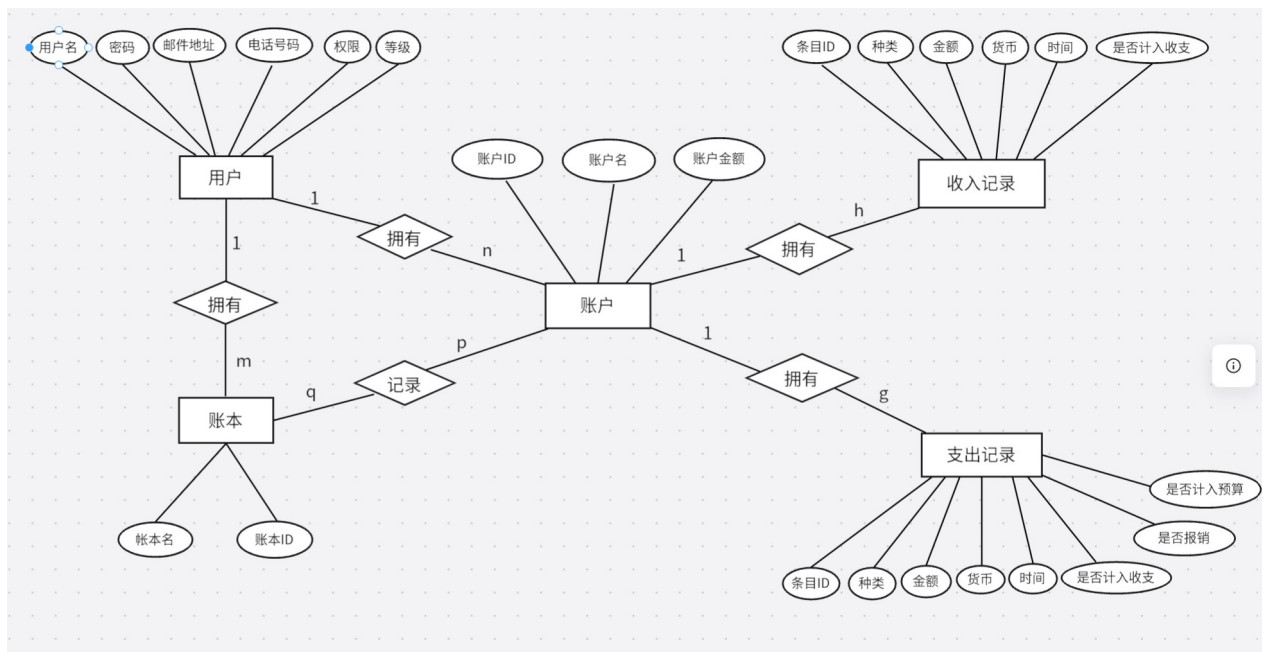


图 9: 概念设计 E-R 图

## 5 逻辑设计

### 5.1 E-R 图到关系模型的转换

五个实体 (用户、账户、账本、收入记录、支出记录) 分别转换成一个关系模式；账户与账本之间的多对多联系转换为一个关系模式，剩余的一对多联系与多端关系模式合并，则关系模型为：

**用户** (用户 ID, 密码, 邮件地址, 电话号码, 权限, 等级)，主码为用户 ID。

**账本** (账本 ID, 账本名, 用户 ID)，主码为账本 ID，外码为用户 ID。

**账户** (账户 ID, 账户名, 账户金额, 用户名 ID)，主码为账户 ID，外码为用户 ID。

**收入记录** (条目 ID, 种类, 金额, 货币, 时间, 是否计入收支, 账户 ID, 账本 ID, 用户 ID), 主码为条目 ID, 外码为账户 ID、账本 ID、用户 ID。

**支出记录** (条目 ID, 种类, 金额, 货币, 时间, 是否计入收支, 是否报销, 是否计入预算, 账户 ID, 账本 ID, 用户 ID), 主码为条目 ID, 外码为账户 ID、账本 ID、用户 ID。

**账户-账本** (账户 ID, 账本 ID), 主码为 (账户 ID, 账本 ID), 两者分别是外码。

## 5.2 系统结构图

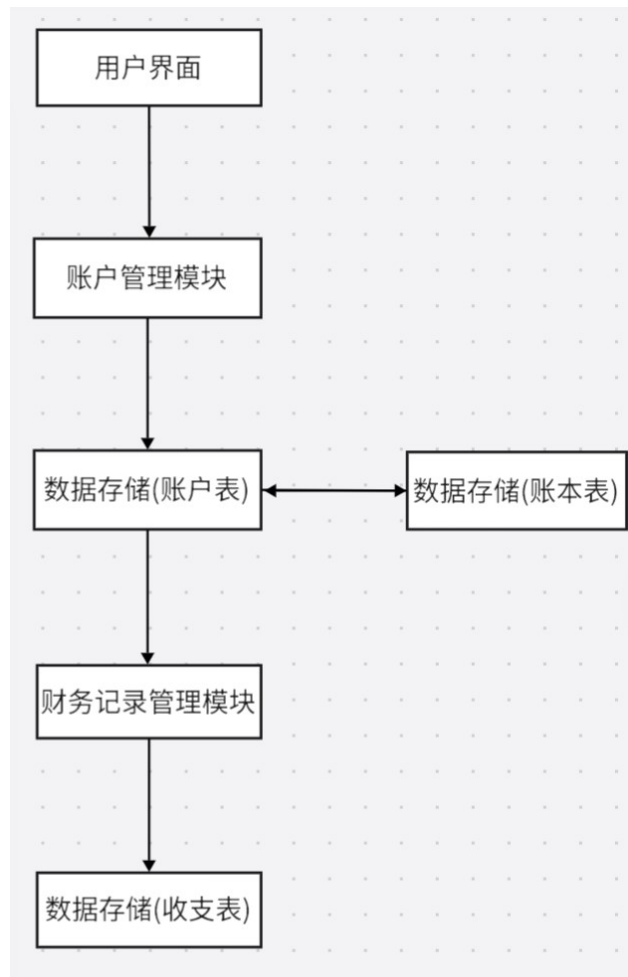


图 10: 系统结构图

## 6 物理设计

### 6.1 存储安排

#### 6.1.1 数据库选择

本项目使用 Django 默认的数据库引擎 SQLite, 易于配置和使用。

#### 6.1.2 数据库模式

每个模型映射为一个表, 字段映射为表的列。

### 6.2 方法选择

#### 6.2.1 数据索引

在 AccountBook、FundAccount、Transaction 的主码和外码字段建立索引, 优化查询速度。

#### 6.2.2 事务管理

使用数据库事务保证数据的一致性和完整性, 避免在操作多个表时出现数据不一致的情况。

### 6.3 存储路径

数据库文件存储在项目根目录下, 文件名为 db.sqlite3。

## 6.4 模块设计 IPO 表

模块编号	模块名称	功能描述	输入	输出
1	用户界面	提供账本、账户、收支记录的管理界面	用户输入	显示数据
2	账户管理模块	处理账户创建、修改、删除等操作	账户名称、余额、账本ID	更新后的账户信息
3	账本管理模块	处理账本的创建和管理	账本名称	更新后的账本信息
4	收支记录模块	处理收支记录的创建与查询	收入/支出金额, 类别, 账户ID	新增的收支记录
5	数据存储模块	负责所有数据的存储, 包括账户、账本、收支记录的存储	账本、账户、收支记录的数据	更新后的数据库数据
6	数据可视化模块	提供收支数据的可视化图表(饼图)	收支数据、日期、分类数据	图表数据

图 11: 模块设计 IPO 表

## 7 技术架构

- 后端框架: Django, 用于处理业务逻辑和数据库交互。
- 数据库: 使用 Django ORM 与数据库 (例如 SQLite 或 PostgreSQL) 交互。
- 前端框架: CoreUI-Bootstrap, 用于实现响应式布局和组件化的界面设计。
- 数据可视化: 使用 Chart.js 绘制图表。
- 接口设计: 后端提供 RESTful API, 通过 Django REST framework 实现。

## 8 功能模块设计与实现

### 8.1 账本、账户管理模块

账本管理功能通过 `accountbook` 应用实现, 包含以下功能:

- URL 配置:

```
path('add_accountbook/', views.add_accountbook, name='add_accountbook'),
path('update_accountbook/<uuid:accountbook_id>/',
     views.update_accountbook, name='update_accountbook'),
path('delete_accountbook/<uuid:accountbook_id>/',
```

```
views.delete_accountbook, name='delete_accountbook'),
path('accountbook_list/', views.accountbook_list, name='accountbook_list'),
```

- **视图函数**：使用 Django 的通用视图函数（CBV 或 FBV）处理表单提交和数据更新。
- **数据库模型**：AccountBook 和 FundAccount 表记录用户的账本和资金账户。账户管理模块的实现比较类似，在此不多赘述。

## 8.2 收支管理、统计模块

收支管理功能通过 consumptionANDincome 应用实现，支持添加、删除、更新和按分类统计收入与支出。具体功能包括：

- **URL 配置**：

```
path('add_income/', views.add_income, name='add_income'),
path('delete_income/<int:income_id>/',
     views.delete_income, name='delete_income'),
path('update_income/<int:income_id>/',
     views.update_income, name='update_income'),
path('expenses-by-category/', views.expenses_by_category_in_interval,
     name='expenses_by_category_in_interval'),
path('income-by-category/', views.income_by_category_in_interval,
     name='income_by_category_in_interval'),
```

- **数据库模型**：

- ExpenseEntry：记录支出的详细信息，包括金额、分类、时间等。
- IncomeEntry：记录收入的详细信息。

- **统计 API**：

```
expenses_by_category_in_interval(request):
    # 返回按分类的支出总和
income_by_category_in_interval(request):
    # 返回按分类的收入总和
```

### 8.3 仪表盘模块

仪表盘功能通过 dashboard 应用实现，使用 Chart.js 绘制图表，展示以下内容：

- **收入与支出分类统计饼图：**调用上述 API，根据时间范围筛选数据，动态更新图表。
- **前端代码：**

```
<div class="row">
  <div class="col-md-6">
    <canvas id="expensesCategoryChart"></canvas>
  </div>
  <div class="col-md-6">
    <canvas id="incomeCategoryChart"></canvas>
  </div>
</div>
```

- **数据交互逻辑：**使用 JavaScript `fetch()` 函数获取 API 数据，并通过 Chart.js 动态更新图表。

### 8.4 接口设计

本系统的接口设计包括支出和收入相关的统计、管理、以及收支数据的可视化等功能。以下为各主要 API 的设计与实现说明：

#### 每周支出

- **URL:** `/consumptionANDincome/weekly-expenses-by-category/`
- **方法:** GET
- **功能描述:** 获取当前周内按支出类别分类的支出总额。
- **响应示例:**

```
{
  "expenses_by_category": [
    {"category": "Food", "total_amount": 150.00},
    {"category": "Transport", "total_amount": 50.00}
```

```
    ]  
  }  
}
```

## 每周收入

- **URL:** /consumptionANDincome/weekly-income-by-category/
- **方法:** GET
- **功能描述:** 获取当前周内按收入类别分类的收入总额。
- **响应示例:**

```
{  
  "income_by_category": [  
    {"category": "Salary", "total_amount": 3000.00},  
    {"category": "Investment", "total_amount": 200.00}  
  ]  
}
```

## 特定时间支出

- **URL:** /consumptionANDincome/expenses-by-category/
- **方法:** GET
- **功能描述:** 获取指定日期区间内按支出类别分类的支出总额。
- **请求参数:**

- start\_date: 起始日期 (格式: YYYY-MM-DD)
- end\_date: 截止日期 (格式: YYYY-MM-DD)

- **响应示例:**

```
{  
  "expenses_by_category": [  
    {"category": "Rent", "total_amount": 1200.00},  
    {"category": "Food", "total_amount": 300.00}  
  ]  
}
```

## 特定时间收入

- **URL:** /consumptionANDincome/income-by-category/
- **方法:** GET
- **功能描述:** 获取指定日期区间内按收入类别分类的收入总额。
- **请求参数:**
  - start\_date: 起始日期 (格式: YYYY-MM-DD)
  - end\_date: 截止日期 (格式: YYYY-MM-DD)
- **响应示例:**

```
{
  "income_by_category": [
    {"category": "Bonus", "total_amount": 500.00},
    {"category": "Salary", "total_amount": 4000.00}
  ]
}
```

## 添加账本

- **URL:** /add\_accountbook
- **方法:** GET, POST
- **功能描述:** 添加新的账本。
- **请求参数:**
  - name (字符串, 必填): 账本名称。
  - description (字符串, 选填): 账本描述信息。
- **响应示例:**

```
# GET 请求
{
  "form": "<AccountBookForm>"
}
```



```
# POST 请求 (成功)
HTTP 302 Redirect 到 \texttt{/accountbook\_list}

# POST 请求 (失败)
{
  "form_errors": {"name": ["此字段是必填项"]}
}
```

## 账本列表

- URL: /accountbook\_list
- 方法: GET
- 功能描述: 获取并显示账本列表, 可通过查询参数模糊搜索。
- 请求参数:
  - q (字符串, 选填): 用于模糊匹配账本名称的关键字。

- 响应示例:

```
{
  "accountbooks": [
    {"id": 1, "name": "家庭账本", "description": "记录家庭支出和收入"},
    {"id": 2, "name": "个人账本", "description": "个人的支出记录"}
  ]
}
```

## 删除账本

- URL: /delete\_accountbook/<accountbook\_id>
- 方法: GET
- 功能描述: 删除指定 ID 的账本。
- 请求参数:
  - accountbook\_id (整数, 必填): 要删除的账本 ID。

- **响应示例:**

HTTP 302 Redirect 到 \texttt{/accountbook\\_list}

## 添加账户

- **URL:** /add\_fundaccount
- **方法:** GET, POST
- **功能描述:** 添加新的账户信息。
- **请求参数:**

- name (字符串, 必填): 账户名称。
- balance (浮点数, 必填): 账户余额。

- **响应示例:**

# GET 请求

```
{  
    "form": "<FundAccountForm>"  
}
```

# POST 请求 (成功)

HTTP 302 Redirect 到 \texttt{/fundaccount\\_list}

# POST 请求 (失败)

```
{  
    "form_errors": {"name": ["此字段是必填项"]}  
}
```

## 账户列表

- **URL:** /fundaccount\_list
- **方法:** GET
- **功能描述:** 获取并显示账户列表, 可通过查询参数模糊搜索。

- **请求参数:**

- q (字符串, 选填): 用于模糊匹配账户名称的关键字。

- **响应示例:**

```
{
  "fund_accounts": [
    {"id": 1, "name": "储蓄账户", "balance": 1000.00},
    {"id": 2, "name": "信用账户", "balance": -200.00}
  ]
}
```

## 删除账户

- **URL:** /delete\_fundaccount/<account\_id>

- **方法:** GET

- **功能描述:** 删除指定 ID 的账户。

- **请求参数:**

- account\_id (整数, 必填): 要删除的账户 ID。

- **响应示例:**

HTTP 302 Redirect 到 \texttt{/fundaccount\\_list}

## 更新账户信息

- **URL:** /update\_fundaccount/<account\_id>

- **方法:** GET, POST

- **功能描述:** 更新指定账户的信息。

- **请求参数:**

- name (字符串, 必填): 新的账户名称。
  - balance (浮点数, 必填): 新的账户余额。

- **响应示例:**

```
# POST 请求 (成功)
HTTP 302 Redirect 到 \texttt{/fundaccount\_list}

# POST 请求 (失败)
{
  "form_errors": {"balance": ["请输入有效的金额"]}
}
```

## 仪表盘首页

- **URL:** /dashboard/
- **方法:** GET
- **功能描述:** 渲染仪表盘首页，显示与用户相关的核心信息。
- **请求参数:** 无
- **响应示例:**

```
{
  "template": "dashboard.html"
}
```

## 静态页面渲染

- **URL:** /pages/<template\_name>.html
- **方法:** GET
- **功能描述:** 根据请求路径动态加载并渲染指定的静态页面模板。
- **请求参数:**
  - `template_name` (字符串, 必填): 要加载的页面模板名称, 例如 `error-404.html`。
- **响应示例:**

```
# 成功加载模板
{
    "template": "pages/<template_name>.html"
}

# 模板不存在（自动返回 404 页面）
{
    "template": "pages/error-404.html"
}
```

## 9 系统演示

### 9.1 注册登录与身份认证

为保护用户隐私数据，该系统只会在本地地上运行，数据不会上传到服务器上。在代码实现中，尚未完善如 AccountBook 等模型中的 User 外键，如果部署到服务器上，该部分需要进一步完善。但该部分基本已开发完善。

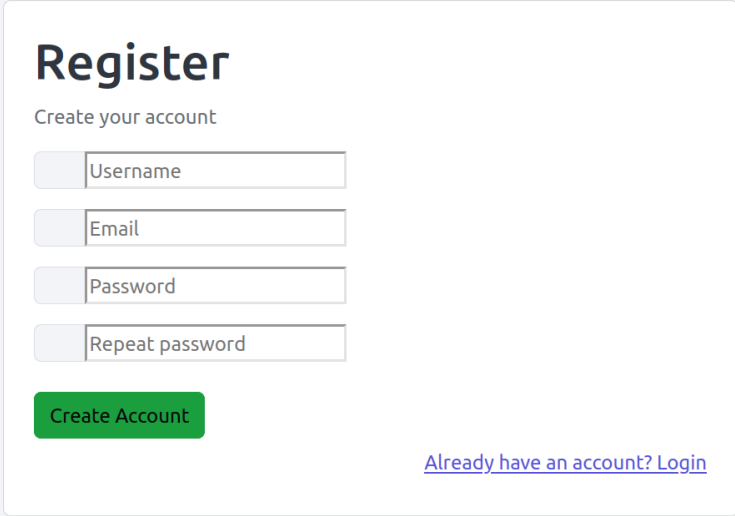


图 12: 注册页面

The image shows a 'Register' form with the title 'Register' and subtitle 'Create your account'. It contains four input fields: 'Username' (containing 'Arcsur'), 'Email', 'Password', and 'Repeat password'. A red error message '此用户名已被注册。' (This username is already registered) is displayed below the 'Username' field. Below the 'Repeat password' field, there is a red message 'This field is required.' and a green 'Create Account' button. At the bottom right, there is a link 'Already have an account? Login'.

图 13: 注册检查页面

我们选择已经注册成功的账户 Arcsur 登录：

The image shows a login and sign-up page. On the left, the 'Login' section has the title 'Login' and subtitle 'Sign In to your account'. It contains two input fields: 'Username' (containing 'Arcsur') and 'Password' (containing '\*\*\*\*\*'). Below these fields is a blue 'Login' button and a link 'Forgot password?'. On the right, the 'Sign up' section has the title 'Sign up' and a paragraph of placeholder text: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.' Below this text is a blue 'Register Now!' button.

图 14: 登陆页面

## 9.2 使用界面

登录成功后会跳转至 Dashboard 页面，在稍后的展示中，我们会让这个页面丰富起来：

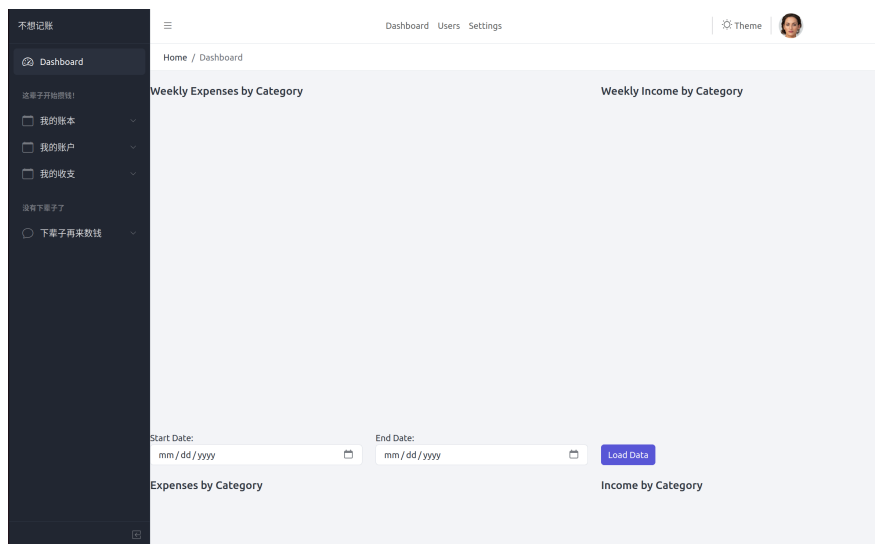


图 15: 初始 Dashboard 页面

## 9.3 账本页面

首先我们添加账本：

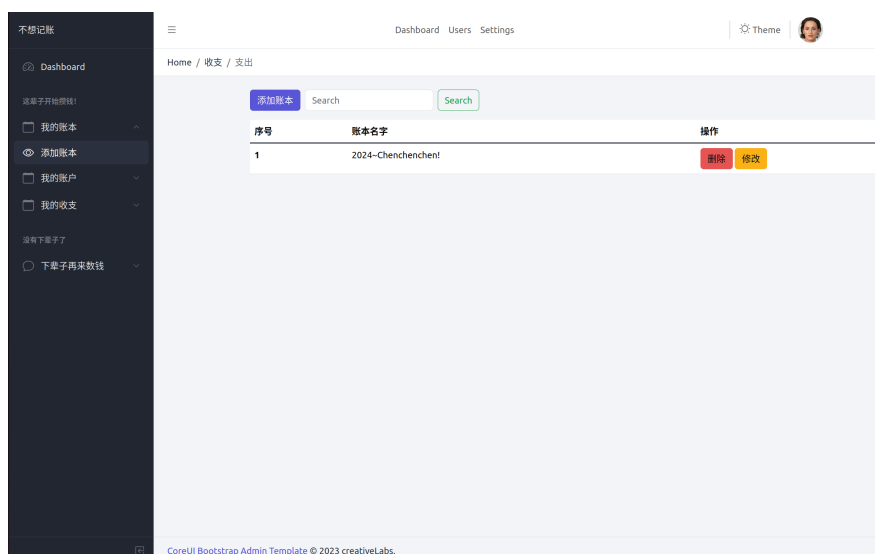


图 16: 添加账本

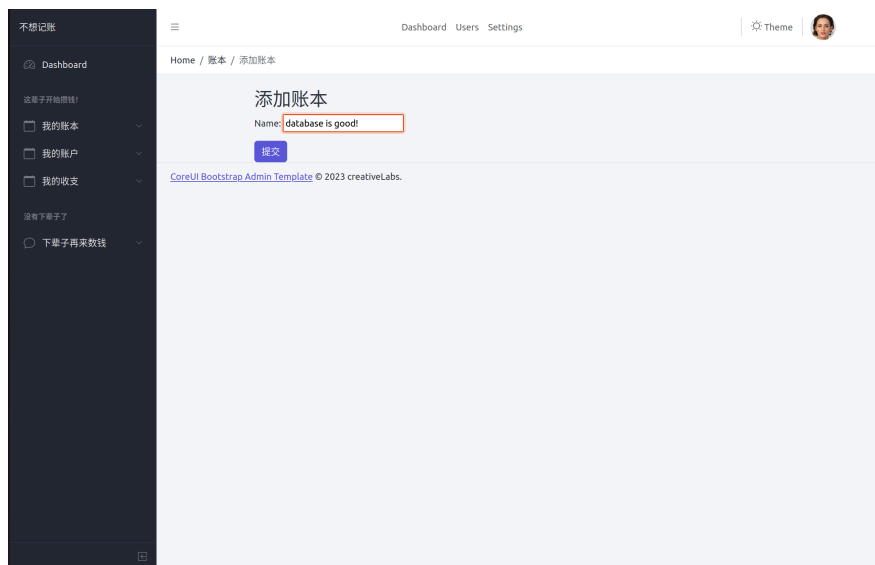


图 17: 添加账本具体内容页面

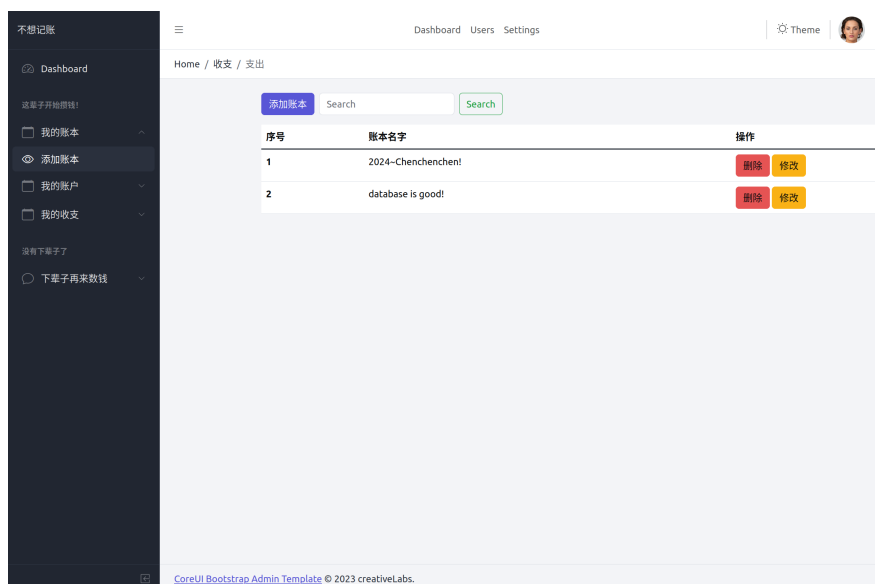


图 18: 账本添加成功

## 9.4 账户页面

在添加完账本后，我们来添加账户：



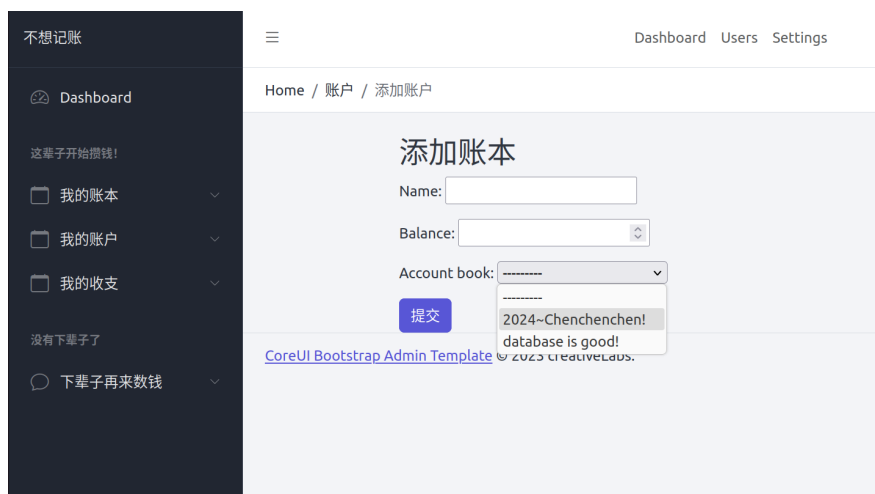


图 19: 添加账户

添加成功：

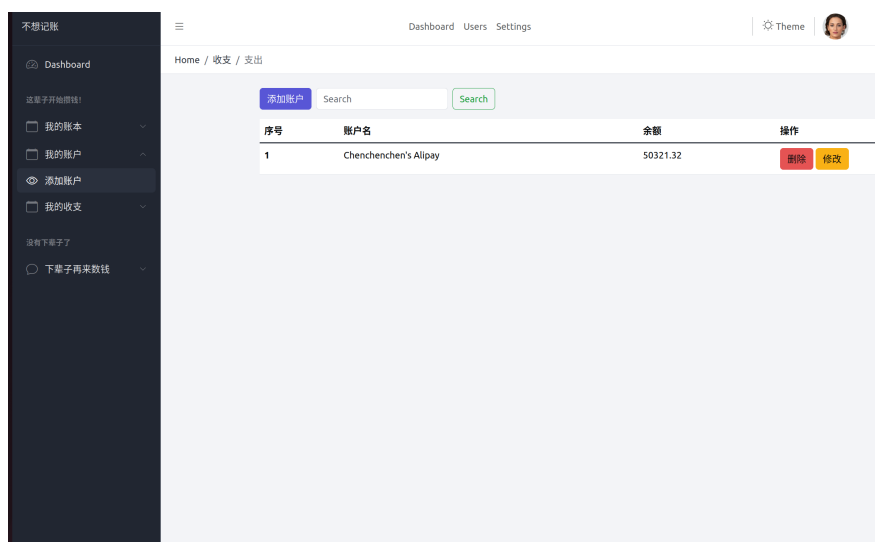


图 20: 添加账户初始余额

## 9.5 收支页面

接着我们来添加收支项：

## 添加收入记录

Income category: Salary ▼

Amount: 12000 ▲▼

Currency: CNY ▼

Income time: 11/20/2024, 10:10 AM 📅

Include in balance: ☒

Account: Chenchenchen's Alipay ▼

Account book: 2024~Chenchenchen! ▼

提交

图 21: 添加收入内容过程

Dashboard

Home / 收支 / 收入

添加收入

Search

Search

序号	记录账本	收入金额	来源	日期	账户	操作
1	2024~Chenchenchen!	12000.00	salary	Nov. 20, 2024, 10:10 a.m.	Chenchenchen's Alipay	<span>删除</span> <span>修改</span>

图 22: 添加收入成功

添加支出是同理过程，不再赘述。再次查看账户余额：

添加账户

Search

Search

序号	账户名	余额	操作
1	Chenchenchen's Alipay	62198.32	<span>删除</span> <span>修改</span>

图 23: 增加收支后的账户新余额

## 9.6 统计页面

查看统计页面

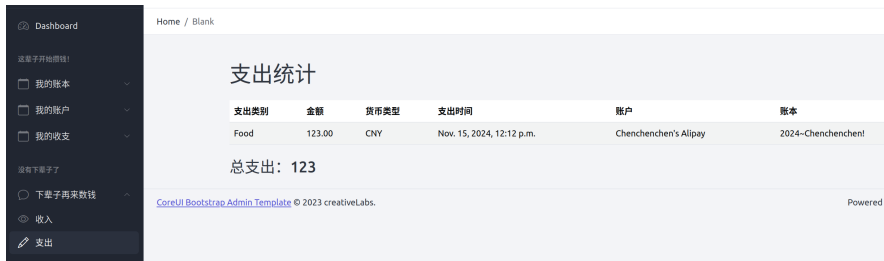


图 24: 支出统计页面

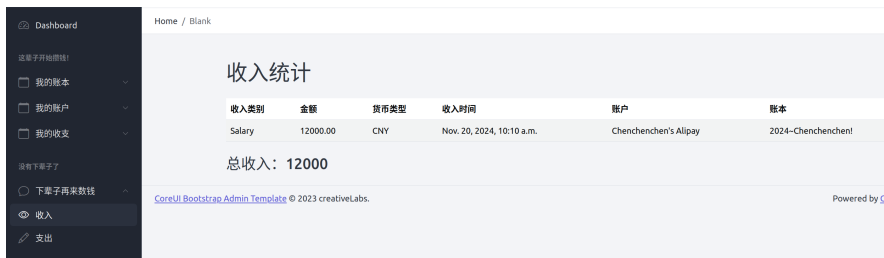


图 25: 收入统计页面

## 9.7 基于 Chart.js 饼状图统计页面

然后再添加许多不同的收支项，此时再查看 Dashboard 页面，会生成在指定时间段内的依据收入支出类别绘制的可视化饼状图，方便用户查看。

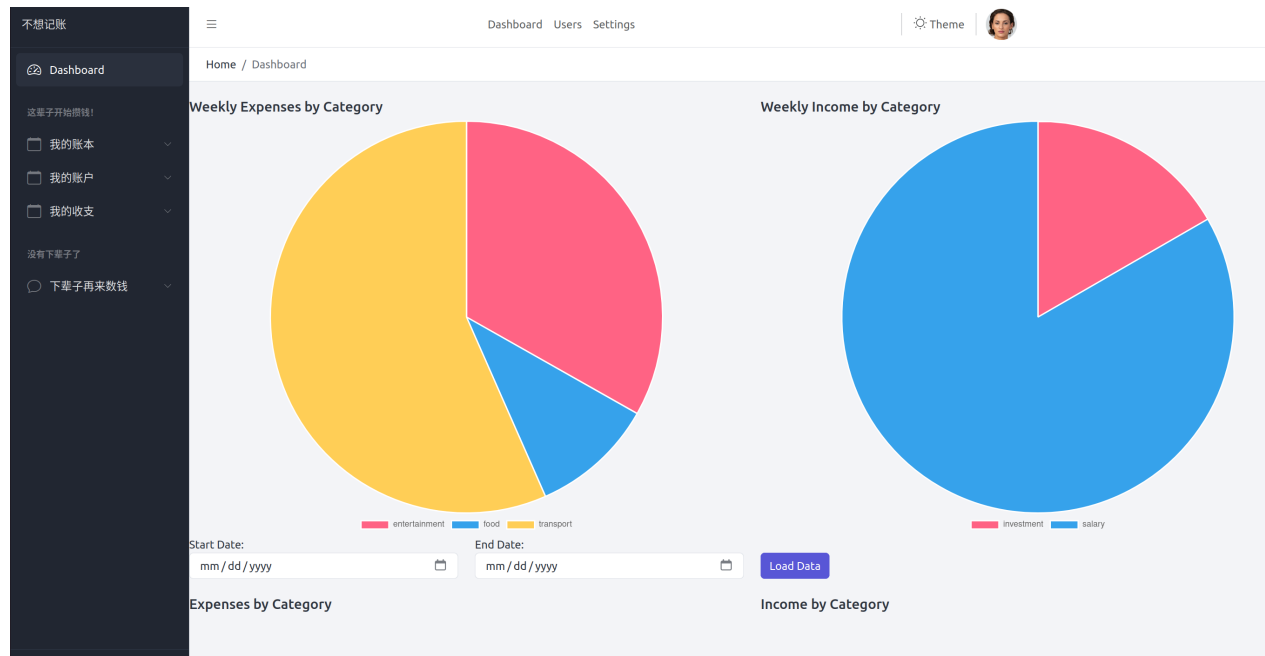


图 26: 一周收支统计页面



图 27: 自定义时间收支统计页面

## 10 流程图



图 28: 总体流程图

## 11 整体总结

### 11.1 分工

陈楠

- 整体基本框架的搭建、设计

- 需求分析、概念设计
- 数据模型的设计
- 登录、注册功能开发
- 账本相应功能开发
- 账户相应功能开发
- 收支相应功能开发
- 收入、支出统计功能开发
- 系统演示撰写
- 整体总结撰写
- 系统维护测试

## 池承运

- 统计模块开发
- Dashboard 模块开发
- 饼状图可视化模块开发
- 技术架构撰写
- 功能模块设计与实现撰写
- 系统维护测试

## 刘潇峰

- 需求分析撰写
- 数据流图撰写
- 概念设计撰写
- 逻辑设计撰写
- 物理设计撰写
- 系统维护测试

## 11.2 任务完成情况

### 陈楠

- 整体完成良好
- 用户认证部分可以进一步开发，将其作为外键与账本、账户绑定，便于多用户能够在服务器上同时互不干扰地使用。

## 池承运

- 整体完成良好
- 可以进一步开发更多新功能，比如定期存钱功能，从微信记账、支付宝收支截图中自动识别收支项添加。

## 刘潇峰

- 整体完成良好

## 11.3 测试与改进

### 11.3.1 功能测试

- **测试环境:** 使用 Postman 进行 API 测试，确保各模块接口能够正常响应。
- **测试用例:**
  - 用户登录验证是否成功。
  - 新增账本、账户，验证数据是否正确存储。
  - 添加收入与支出条目，验证是否成功写入数据库。
  - 统计功能是否按分类正确统计收入与支出。
  - 图表是否能动态更新并正确展示数据。

### 11.3.2 性能优化

- 在高频查询字段上添加数据库索引，例如 `date` 和 `category`。
- 利用缓存机制优化统计功能，避免重复计算。
- 优化前端数据交互，通过 AJAX 异步更新部分页面内容，减少整体页面刷新。

### 11.3.3 未来改进方向

- **多语言支持:** 添加多语言切换功能，支持国际化。
- **移动端优化:** 使用响应式布局进一步优化移动设备上的界面体验。
- **高级统计功能:** 引入更多数据挖掘算法，为用户提供智能财务分析建议。