软 件 工 程 导 论

**实**

**验**

**报**

**告**

**题目：**

**学 院**

**年 级**

**组 员**

**年 月 日**

[1.引言 1](#_Toc115985307)

[1.1编写目的（必填） 1](#_Toc115985308)

[1.2项目背景 1](#_Toc115985309)

[1.3术语说明 1](#_Toc115985310)

[1.4参考资料 1](#_Toc115985311)

[2.软件结构 2](#_Toc115985312)

[2.1需求概述 2](#_Toc115985313)

[2.2软件结构（必填） 2](#_Toc115985314)

[3.程序设计说明 2](#_Toc115985315)

[3.1模块描述 2](#_Toc115985316)

[3.2功能（必填） 2](#_Toc115985317)

[3.3性能 3](#_Toc115985318)

[3.4输入项 3](#_Toc115985319)

[3.5输出项 3](#_Toc115985320)

[3.6算法 3](#_Toc115985321)

[3.7程序逻辑（必填） 3](#_Toc115985322)

[3.8接口 4](#_Toc115985323)

[3.9存储分配 4](#_Toc115985324)

[3.10注释设计 4](#_Toc115985325)

[3.11限制条件 4](#_Toc115985326)

[3.12测试要点 4](#_Toc115985327)

[3.13尚未解决的问题 4](#_Toc115985328)

**详细设计说明书**

详细设计 (又可称程序设计)说明书编制的目的是说明一个软件系统各个层次中的每个程序(每个模块或子程序)的设计考虑，为程序员编写程序提供依据。

详细设计说明书的重点是描述模块的执行流程。

# 1.引言

## 1.1编写目的（必填）

## 编写详细设计说明书的目的：

## •明确项目的开发目标和预期成果，指导开发人员按照统一的设计规范进行编码工作。

## •提高开发效率，减少因理解偏差导致的问题和返工。

## •作为项目验收的标准依据，确保最终产品满足业务需求。

## 详细设计说明书的读者对象：

## •开发团队成员，包括程序员、测试工程师和其他技术支持人员。

## •项目经理和产品经理，用于评估进度和资源分配。

## •客户方代表，用于审核设计方案和提出修改意见。

## 1.2项目背景

**待开发软件的名称**：

•财务管理系统

**任务提出者**：软工实验课程

**开发者**：软工四人小组(王微澜，罗子蘅，方镐翔，朱翌帆)

**用户**：个体经营者、家庭财务管理使用者

## 1.3术语说明(源码写出来之后再改)

**列出本文档中所用到的专门术语的定义和英文缩写词的原文**：

**用户** (User): 使用该软件的人。

**账户** (Account): 用户在系统中创建的身份标识。

**注册** (Registration): 新用户加入系统的过程。

**登录** (Login): 已有用户访问系统的过程。

**财务数据** (Financial Data): 包括收入、支出、资产和负债等相关信息。

**预算** (Budget): 对未来一段时间内收入和支出的计划安排。

**报表** (Report): 将财务数据以表格形式呈现的结果。

## 1.4参考资料（现在还写不了）

列举编写软件详细设计说明时所参考的资料，主要包括:

•引用的软件开发标准或规范。

•项目开发计划。

•需求规格说明书。

•概要设计说明书。

•测试计划(初稿)。

•用户操作手册(初稿)。

•文档中引用的其他资料等。

对每项应该给出详细的信息，包括标题、作者、 版本号、发表日期、出版单位或资料来源。

# 2.软件结构

## 2.1需求概述

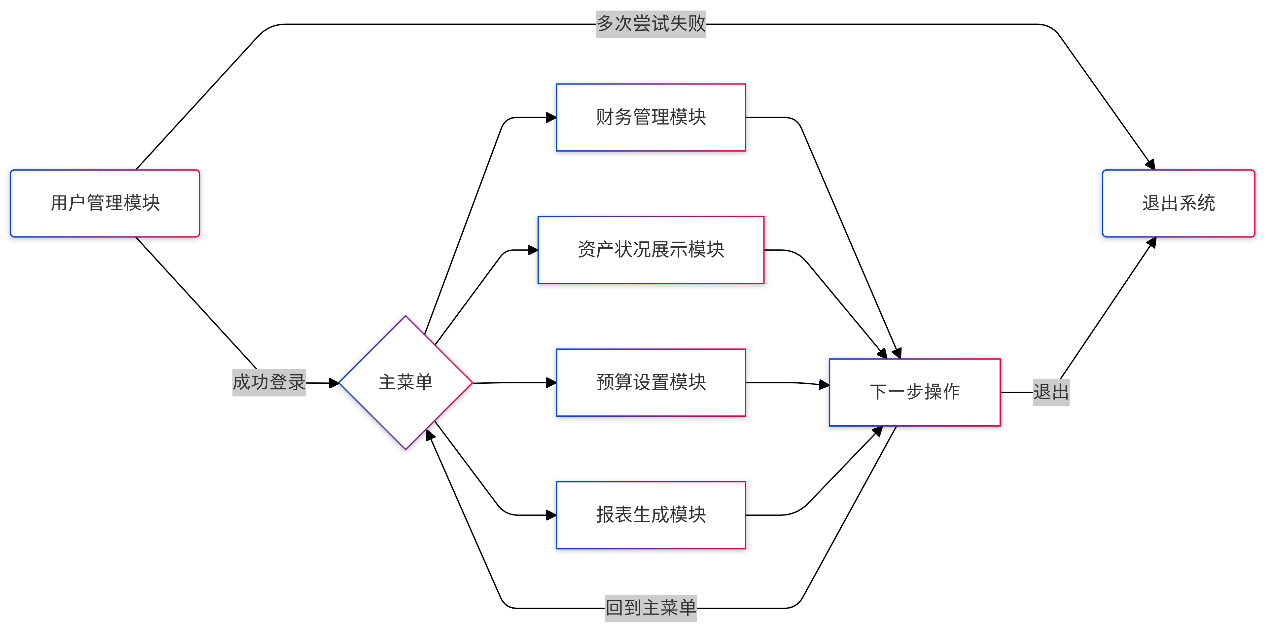
简述本软件的主要功能。

## 2.2软件结构（必填）

用一系列图表列出本软件系统内的每个程序(包括每个模块和子程序)的名称、英文标识，以及它们之间的层次结构关系。

# 3.程序设计说明

## 3.1模块描述



## 3.2功能（必填）

各模块的功能采用IPO图（输入-处理-输出图）描述如下：

**用户管理模块**

| **输入** | **处理** | **输出** |
| --- | --- | --- |
| 用户名、密码 | 验证用户身份 | 验证结果 |
| 注册信息 | 保存新用户信息 | 用户ID |

**财务管理模块**

| **输入** | **处理** | **输出** |
| --- | --- | --- |
| 收入/支出信息 | 保存收入/支出交易记录 | 交易确认 |

**预算管理模块**

| **输入** | **处理** | **输出** |
| --- | --- | --- |
| 预算类别、金额、时间 | 保存预算信息 | 预算设定确认 |

**报表生成模块**

| **输入** | **处理** | **输出** |
| --- | --- | --- |
| 报表时间范围、格式 | 生成财务报告 | 报表文件路径 |

**资产概况展示模块**

| **输入** | **处理** | **输出** |
| --- | --- | --- |
| 用户的请求 | 提供数据存储和检索服务 | 资产和财务数据 |

## 3.3性能

 **精度**：所有财务数据录入和计算须精确到小数点后两位。

 **灵活性**：系统需支持多种输入方式（手动输入、文件导入等）。

 **时间特性**：每个模块的响应时间应不超过2秒，以确保用户体验。

## 3.4输入项

描述每个输入项的特性：

* **用户信息**
  + **名称**: 用户名
  + **标识**: username
  + **数据类型和格式**: 字符串，长度限制为3-20字符
  + **有效范围**: 仅允许字母、数字和下划线
  + **输入方式**: 文本框
  + **数量和频度**: 每次登录需要一次
  + **输入媒体**: 键盘
  + **来源**: 用户
  + **安全保密条件**: 加密存储
* **财务数据**
  + **名称**: 收入金额
  + **标识**: income\_amount
  + **数据类型和格式**: 浮点数
  + **有效范围**: 0到无穷大
  + **输入方式**: 数字输入框
  + **数量和频度**: 每次添加收入需要一次
  + **输入媒体**: 键盘
  + **来源**: 用户
  + **安全保密条件**: 本地加密存储

## 3.5输出项

描述每个输出项的特性：

* **报告文件**
  + **名称**: 财务报告
  + **标识**: financial\_report
  + **数据类型和格式**: CSV/Excel
  + **有效范围**: 正常生成，不应为空
  + **输出形式**: 文件下载
  + **数量和频度**: 按需生成
  + **输出媒体**: 磁盘
  + **安全保密条件**: 仅对授权用户可见

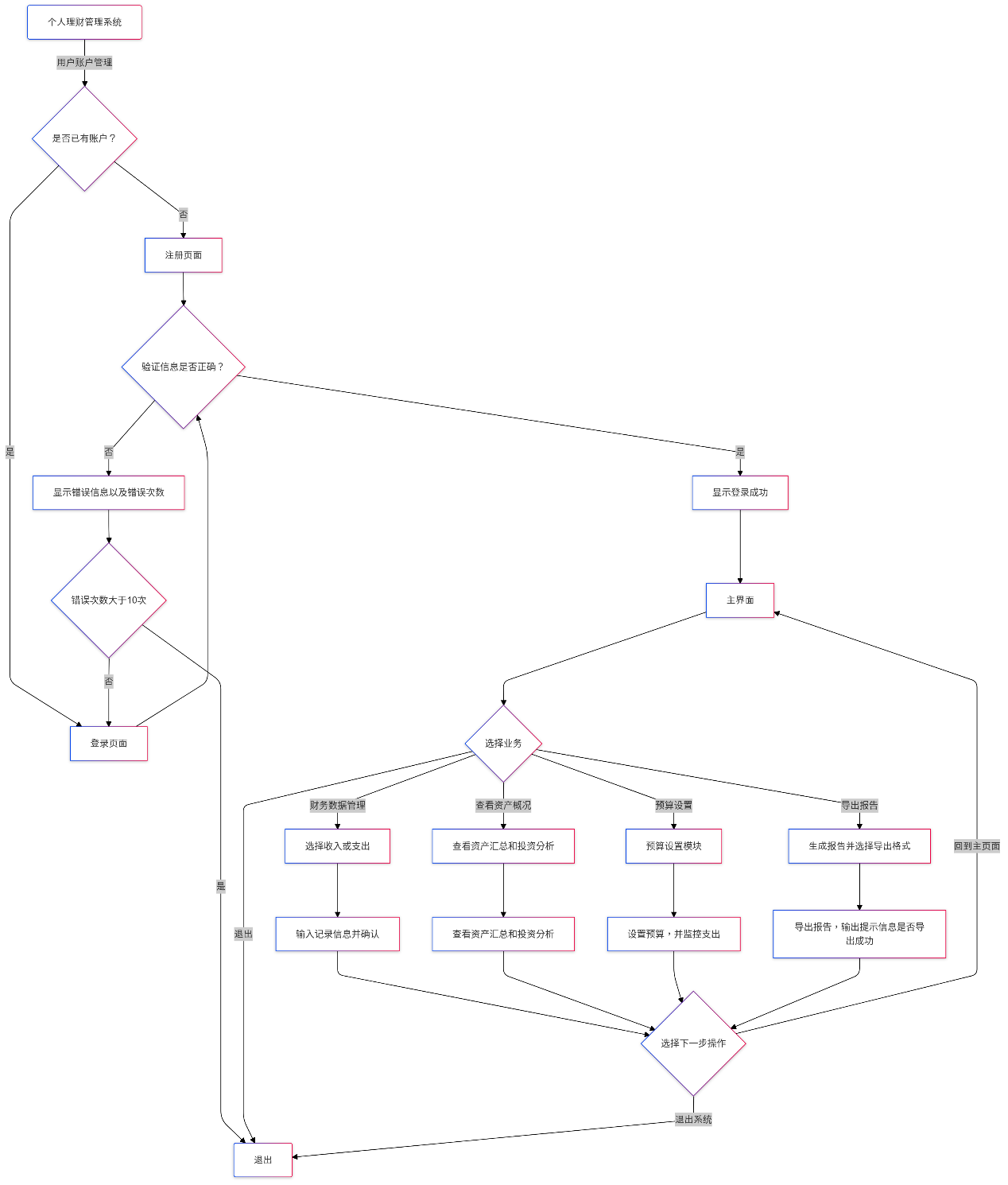
## 3.6算法（后续补充）

模块所选用的算法：

* **预算监控算法**
  + **公式**: 当前支出与预算额度的比值
  + **计算步骤**:
    1. 获取当前预算额度和支出数据
    2. 计算支出总额
    3. 计算支出比值
    4. 如果比值超过100%，则触发警报

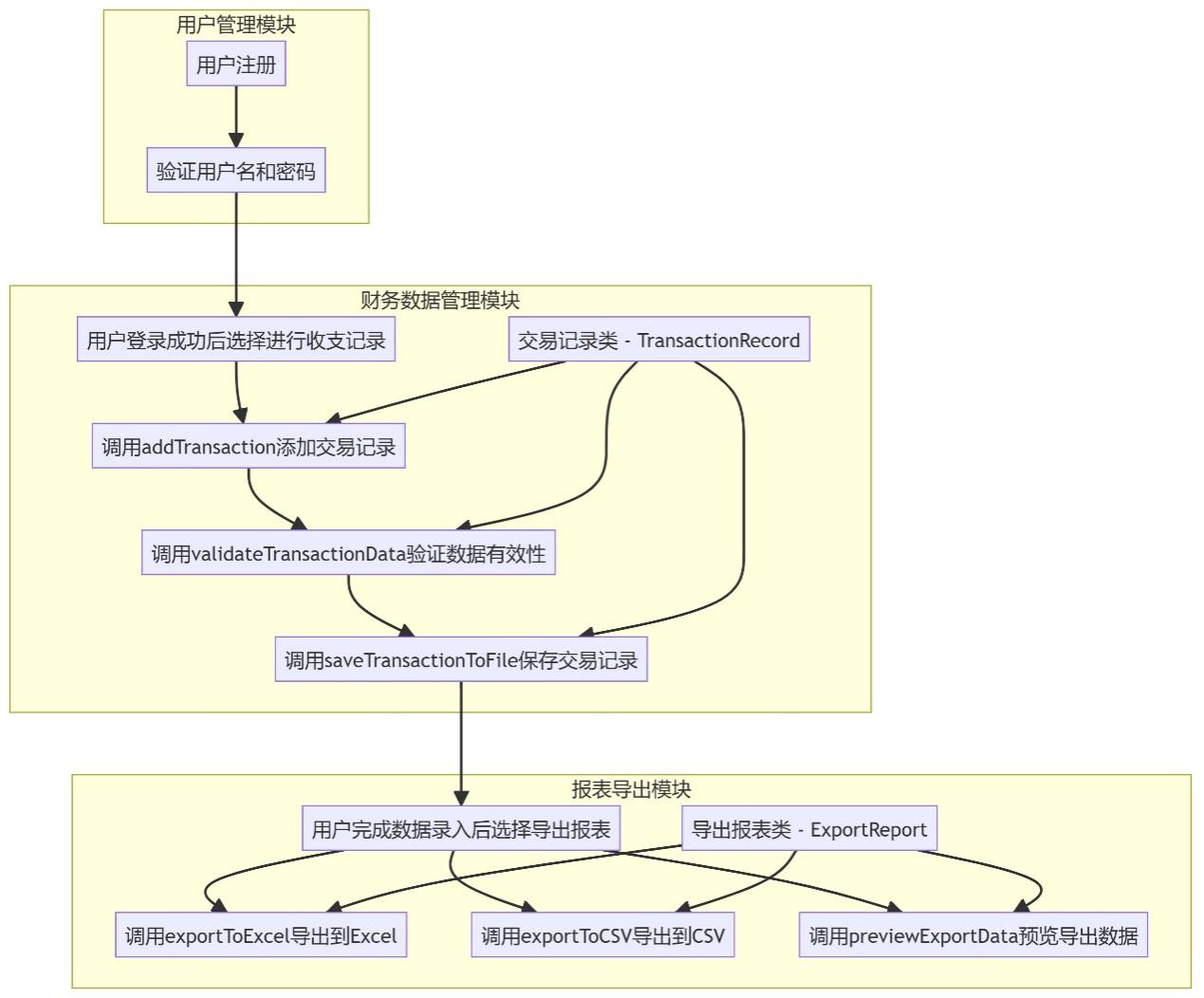
## 3.7程序逻辑（必填）

程序流程图



## 3.8接口

用图的形式说明本模块所隶属的上一层模块及隶属于本模块的下一层模块，说明参数赋值和调用方式，说明与本模块相直接关联的数据结构( 数据库、数据文卷)。



## 3.9存储分配

**存储分配方案**

1. **用户管理模块**
   * **存储内容**: 用户账户信息
   * **数据结构**:
     + UserAccount (用户名、加密密码)
   * **存储需求**:
     + 用户名 (字符串): 50 字节
     + 加密密码 (字符串): 128 字节
   * **总存储需求**: 每个用户约 178 字节
2. **财务数据管理模块**
   * **存储内容**: 用户的收入与支出记录
   * **数据结构**:
     + TransactionRecord (交易ID、类别、金额、日期、备注)
   * **存储需求**:
     + 交易ID (字符串): 36 字节 (UUID格式)
     + 类别 (字符串): 30 字节
     + 金额 (浮点数): 8 字节
     + 日期 (字符串): 10 字节
     + 备注 (字符串): 100 字节
   * **总存储需求**: 每条记录约 184 字节
3. **报表导出模块**
   * **存储内容**: 导出的报表数据
   * **数据结构**:
     + ExportReport (导出ID、开始日期、结束日期、格式、状态、文件路径)
   * **存储需求**:
     + 导出ID (字符串): 36 字节
     + 开始日期 (字符串): 10 字节
     + 结束日期 (字符串): 10 字节
     + 格式 (字符串): 10 字节
     + 状态 (字符串): 20 字节
     + 文件路径 (字符串): 200 字节
   * **总存储需求**: 每个导出记录约 286 字节

## 3.10注释设计

**注释设计的特点**

1. **清晰性**: 注释应简洁明了，能够让阅读者快速理解模块的功能和逻辑。避免使用复杂的术语或过于技术化的语言。
2. **一致性**: 注释风格应在整个代码库中保持一致，包括命名约定、格式和注释类型。这有助于提高代码的可读性和可维护性。
3. **可维护性**: 注释应随代码的修改而更新，确保它们始终与实现保持一致，避免过时的注释导致误解。
4. **详尽性**: 对于复杂的逻辑或重要的数据结构，提供足够的细节，以便其他开发者（或未来的自己）能够理解代码的实现和目的。
5. **结构化**: 使用标准化的格式，例如Javadoc风格的注释，能更好地支持自动文档生成工具，使文档化过程更加高效。

**注释设计的规则**

1. **模块首部注释**:
   * 包含模块名称、功能描述、作者信息、版本号、创建日期和修改历史。
   * 提供模块的总体概述，帮助理解模块的用途。
2. **分支点注释**:
   * 在条件判断或逻辑分支处添加注释，说明判断条件及相应的处理逻辑。
   * 确保在关键决策点清晰表述意图。
3. **变量注释**:
   * 对每个变量的功能、类型、范围和默认值等进行描述。
   * 有助于其他开发者快速了解每个变量的用途和限制。
4. **方法注释**:
   * 包括方法功能描述、参数说明、返回值说明以及可能抛出的异常。
   * 明确方法的行为和预期，可以提高代码的可重用性和可靠性。
5. **逻辑注释**:
   * 对使用的算法、复杂逻辑或关键实现进行详细解释。
   * 便于后续开发者理解实现思路，特别是在非直观的代码段中。

以下为示例：

•加在模块首部的注释。

/\*\*

\* 模块名称: 财务数据管理模块

\* 功能: 处理用户的收入和支出记录，包括数据的录入、验证、更新和删除。

\* 作者: [作者姓名]

\* 版本: 1.0

\* 创建日期: [创建日期]

\* 修改历史:

\* - [日期] - 初始创建

\* - [日期] - [修改内容]

\*/

•加在各分支点处的注释。

// 检查交易类型并执行相应的处理

if (transactionType.equals("收入")) {

// 处理收入类型的交易

} else if (transactionType.equals("支出")) {

// 处理支出类型的交易

} else {

// 抛出异常，处理无效的交易类型

}

•对各变量的功能、范围、缺省条件等所加的注释。

/\*\*

\* 交易记录类

\*/

public class TransactionRecord {

private String transactionId; // 交易的唯一标识符

private String category; // 交易类别，如“收入”或“支出”

private double amount; // 交易金额，必须为正数

private String date; // 交易日期，格式为“YYYY-MM-DD”

private String note; // 备注信息，默认为空字符串

// 构造方法

public TransactionRecord(String transactionId, String category, double amount, String date, String note) {

// 初始化交易记录

}

}

•对使用的逻辑所加的注释等。

/\*\*

\* 验证交易数据的有效性。

\*

\* 检查：

\* - 交易类型是否有效

\* - 金额是否为正数

\* - 日期格式是否正确

\*

\* @param transactionType 交易类型，如“收入”或“支出”

\* @param amount 交易金额

\* @param date 交易日期

\* @return 如果所有验证通过，则返回true，否则返回false

\*/

public boolean validateTransactionData(String transactionType, double amount, String date) {

// 检查交易类型

// 检查金额

// 检查日期格式

return true; // 所有验证通过

}

## 3.11限制条件

**财务数据管理模块的运行限制条件**

1. **环境限制**:
   * 该模块要求在支持Java的操作系统上运行，推荐的操作系统包括Windows、macOS和主要的Linux发行版。
   * 必须确保Java Runtime Environment (JRE) 已安装，且版本与开发时使用的版本兼容。
2. **数据存储限制**:
   * 所有财务数据将保存在本地存储中，因此需确保用户设备有足够的存储空间以容纳用户的财务记录和报告。
   * 不支持云存储或在线数据同步，所有数据处理和存储均为离线操作。
3. **用户限制**:
   * 该模块设计为单用户使用，不支持多用户同时访问或并发操作。
   * 用户必须注册并登录才能使用模块的全部功能，系统会限制未注册用户的访问权限。
4. **数据有效性限制**:
   * 输入数据必须符合预定义格式，例如金额必须为正数，日期格式需为“YYYY-MM-DD”。
   * 在录入交易数据时，系统会进行实时验证，确保数据的准确性和完整性，错误数据将无法保存。
5. **功能限制**:
   * 当前版本的模块不支持自定义分类，用户只能选择预设的交易类别。
   * 仅支持基本的收入与支出记录，复杂的财务分析和预测功能可能在后续版本中实现。
6. **性能限制**:
   * 在处理大量数据时（例如数千条交易记录），可能会影响系统的响应时间，用户应耐心等待数据加载。
   * 系统设计的平均响应时间为2秒以内，某些复杂操作（如报表生成）可能需要更长时间。

## 3.12测试要点

给出对本模块进行单元测试的主要测试要求，包括对测试的技术要求、输入数据、预期结果等的规定。

**单元测试主要测试要求**

**1. 技术要求**

* **开发环境: 使用与模块相同的Java版本进行测试，确保兼容性。**
* **测试框架: 采用JUnit或TestNG等流行的Java测试框架进行单元测试。**
* **覆盖率要求: 代码覆盖率应达到85%以上，确保大部分代码路径经过测试。**
* **异常处理: 测试必须包括对异常情况的验证，确保代码在异常发生时能够正确处理。**

**2. 输入数据**

* **用户注册测试:**
  + **输入数据：有效用户名、有效密码、有效邮箱。**
  + **输入数据（边界条件）：用户名长度为1字符和最大限制字符数。**
  + **输入数据（无效条件）：已存在的用户名、无效的邮箱格式、弱密码。**
* **交易录入测试:**
  + **输入数据：有效的交易类别（如“收入”或“支出”）、正金额、正确的日期格式、备注。**
  + **输入数据（无效条件）：负金额、错误的日期格式、空的交易类别。**
* **交易验证测试:**
  + **输入数据：各种组合的交易类型（收入和支出），包括边界情况。**
  + **输入数据（无效条件）：不支持的交易类型、超出限制的金额。**

**3. 预期结果**

* **用户注册测试:**
  + **预期结果：成功注册的用户应返回成功消息，并在数据库中生成用户记录。**
  + **对于无效输入，应返回相应的错误消息，且不应创建用户记录。**
* **交易录入测试:**
  + **预期结果：有效交易应成功录入，并在交易记录中保存正确的信息。**
  + **对于无效输入，应返回错误提示，且不应保存无效的交易记录。**
* **交易验证测试:**
  + **预期结果：有效交易数据应返回“验证通过”。**
  + **对于无效交易数据，应返回“验证失败”并指明错误原因。**

**4. 性能要求**

* **执行时间: 每个测试用例应在合理时间内完成（如500毫秒以内），以确保模块在高负载下的响应能力。**

## 3.13尚未解决的问题

**1. 数据安全性问题**

* **问题描述**: 当前设计中对用户数据的加密和存储方式尚未明确。
* **解决建议**: 设计者需决定采用何种加密算法保护用户敏感信息（如密码），并确保数据在存储和传输过程中的安全性。

**2. 数据备份与恢复机制**

* **问题描述**: 尚未设计数据备份和恢复方案，以防止数据丢失或损坏。
* **解决建议**: 考虑定期备份用户的财务数据，并设计简单易用的恢复机制，确保用户能够在数据丢失后迅速恢复。

**3. 性能优化**

* **问题描述**: 模块在处理大量交易记录时的性能表现尚未评估。
* **解决建议**: 在软件开发的后期进行性能测试，并根据测试结果优化数据处理算法和存储结构，以确保在高负载情况下仍能保持良好的响应速度。

**5. 用户界面友好性**

* **问题描述**: 目前界面的设计原型尚未经过用户测试，可能缺乏用户友好性。
* **解决建议**: 在开发过程中进行用户体验（UX）测试，根据用户反馈调整界面设计，确保操作的简便性和直观性。

**6. 交易类别的灵活性**

* **问题描述**: 当前设计仅支持预设的交易类别，无法满足用户自定义需求。
* **解决建议**: 设计一个用户可以添加、编辑和删除自定义交易类别的机制，以提高系统的灵活性和适用性。