管理信息课程设计说明书

（IFM跨银行金融管理系统）

Management Information Systems Course Design Specification

（Interbank Financial Management System）

学院：天津科技大学 经济与管理学院

专业： 工业工程

姓名： 杨宸铭

##### 2024年6月28日

##### 摘 要

#### 本金融管理系统项目旨在开发一个跨银行金融系统的管理信息系统。系统主要功能包括用户注册与登录、用户信息管理、跨行转账、黄金和股票的购买与管理等。系统通过使用HTML、CSS、PHP和MySQLi等技术，实现了用户友好的前端界面与可靠的后端数据处理。

#### 在用户管理模块中，用户可以通过注册页面提交其个人信息，并在成功注册后登录系统。用户登录后，系统会验证其身份，并显示用户的详细信息，包括昵称、头像、账户余额等。用户还可以修改其个人信息，系统会实时更新数据库。

#### 跨行转账模块允许用户在不同银行账户之间进行资金转移。用户需选择转出和转入的银行卡号，并输入转账金额。系统会验证用户的余额是否足够，并在转账成功后更新用户的账户余额，同时记录转账交易信息。

#### 黄金管理模块包括黄金价格生成、黄金详情查询和黄金购买。系统会定期更新市场黄金价格，并根据不同品牌的系数生成各品牌的黄金价格。用户可以查看最新的黄金价格，选择品牌并输入购买重量进行购买。系统会验证用户的银行卡余额，并在购买成功后记录用户的黄金持有信息。

#### 股票管理模块类似于黄金管理，包括股票价格生成、股票详情查询和股票购买。系统根据一定的算法生成股票价格，并记录股票的历史价格。用户可以查看最新的股票价格，选择股票并输入购买数量进行购买。系统同样会验证用户的余额，并记录用户的股票持有信息。

#### 通过以上功能，系统实现了金融数据的有效管理和用户友好互动，能够为用户提供便利的金融服务体验。

#### 关键词：跨银行系统；综合管理；股票趋势；黄金趋势

##### ABSTRACT

#### This financial management system project develops a management information system for cross-bank financial transactions. Key functions include user registration and login, user information management, cross-bank transfers, and the purchase and management of gold and stocks. Utilizing HTML, CSS, PHP, and MySQLi, the system offers a user-friendly frontend interface and reliable backend data processing.

#### In the user management module, users can register, log in, and manage their personal information. The system verifies identities, displays user details, and updates the database in real time.

#### The cross-bank transfer module enables fund transfers between different bank accounts. Users select debit and credit card numbers, enter the transfer amount, and the system verifies balances, updates account information, and records transaction details.

#### The gold management module handles gold price generation, detail inquiry, and purchase. The system updates market gold prices, generates brand-specific prices, and verifies user balances before recording gold holdings upon purchase.

#### The stock management module includes stock price generation, detail inquiry, and purchase. The system generates stock prices, records historical prices, verifies user balances, and records stock holdings upon purchase.

#### These features ensure efficient financial data management and user-friendly interaction, providing a convenient financial service experience.

#### Keywords：Cross-bank System；Integrated Management；Stock Trends；Gold Trends

**目 录**

[第1章 调研结果和业务流程 1](#_Toc170988240)

[1.1 研究背景 1](#_Toc170988241)

[1.2 研究意义 1](#_Toc170988242)

[1.3 业务流程 2](#_Toc170988243)

[第2章 数据流程图 3](#_Toc170988244)

[2.1 数据流程图 3](#_Toc170988245)

[第3章 数据库设计 6](#_Toc170988246)

[3.1 数据表设计 6](#_Toc170988247)

[3.2 范式分析 9](#_Toc170988248)

[第4章 动态网页开发 12](#_Toc170988249)

[4.1 登录界面 12](#_Toc170988250)

[4.2 重置密码 13](#_Toc170988251)

[4.3 注册 15](#_Toc170988252)

[4.4 主界面 18](#_Toc170988253)

[4.5 股票 21](#_Toc170988254)

[4.6 黄金 31](#_Toc170988255)

[第5章 总结与体会 39](#_Toc170988256)

[5.1 小组成员分工情况 39](#_Toc170988257)

[5.2 收获 39](#_Toc170988258)

[5.3 改进设想 40](#_Toc170988259)

# 调研结果和业务流程

## 研究背景

在当今信息化和数字化快速发展的时代，金融机构和用户对数据处理和管理的需求愈发强烈。传统的金融管理

系统主要依靠手工操作和纸质文档，不仅效率低下，而且容易出现数据错误和信息滞后。随着互联网技术的发展，越来越多的金融机构开始采用信息系统来处理和管理金融数据，提高工作效率和服务质量。

在日常生活中，线上金融交易平台种类繁多，用户需要登录不同的平台才能查看各个银行和投资账户的独立信息，这无疑增加了管理和操作的复杂性。为了解决这一问题，本次课程设计旨在开发一个能够跨平台进行金融管理和操作的管理信息系统。这个系统可以将多个平台的数据整合到一个界面中，方便用户进行统一管理。

## 研究意义

本次课程设计的管理信息系统（Management Information System, MIS）主要聚焦于内部数据处理和交互功能，旨在通过模拟金融系统的关键功能，提升用户对金融数据管理的理解和操作能力。本系统通过整合多种金融功能模块，实现跨行转账、交易记录查询、黄金价格记录与分析、基金与股票推荐等多项功能，提供全面的金融数据处理解决方案。

本次课设的意义提升数据处理效率：通过自动化的系统处理，大幅提高数据处理的速度和准确性，减少人工操作的错误，提高金融机构的运营效率。

增强用户体验：通过提供友好的用户界面和智能推荐功能，帮助用户快速获取所需信息，并作出更合理的投资决策，提高用户满意度。

支持决策分析：系统通过对历史数据的分析和挖掘，提供详细的报表和数据分析结果，帮助金融机构和用户进行科学的决策支持。

推动金融科技发展：本系统的设计和实现为金融科技的发展提供了有力支持，通过模拟真实的金融业务场景，帮助学生和开发者更好地理解和应用金融信息系统技术。

培养专业人才：通过本系统的开发和使用，培养学生的实际操作能力和解决问题的能力，为未来从事金融科技领域的工作打下坚实基础。

## 业务流程

用户登录与注册：用户可以通过系统进行身份验证，包括新用户注册和已有用户的登录操作。此模块确保了用户的身份认证及访问控制。

用户信息管理：用户可以查看和修改个人信息。这一功能模块保障了用户信息的准确性和更新性，提升了用户体验。

金融产品查询与购买：股票与黄金交易：用户可以查询当前的股票和黄金价格，并进行购买操作。该模块实现了实时市场数据的展示与交易功能的集成。

银行转账与结果查询：用户可以进行银行转账操作，并查询转账结果。此功能模块确保了资金的安全转移和交易记录的透明性。

总资产查看：用户可以查看其在系统中的总资产情况，包括股票、黄金、现金等多种资产类别的汇总信息。该模块提供了用户资产的综合视图，便于用户进行财务管理。

用户登出：用户可以安全退出系统，确保会话的安全性和隐私保护。

这些业务流程通过有效的系统集成，实现了用户端各项操作的便捷性和高效性，充分利用了后台数据库的数据处理能力，为用户提供了全面的金融服务体验。。

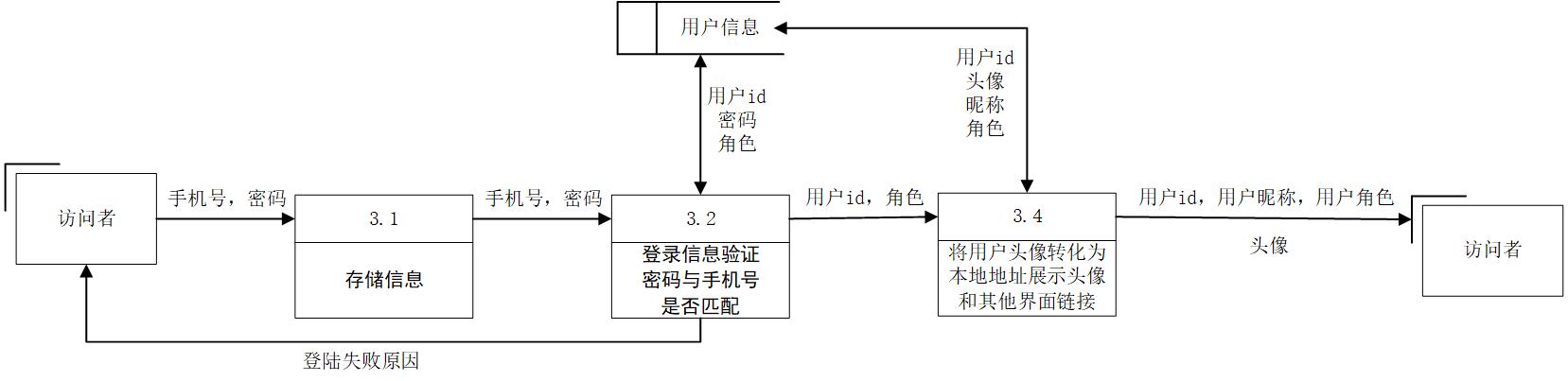
# 数据流程图

## 数据流程图

数据流程图（Data Flow Diagram， DFD/Data Flow Chart） ， 简称数据流图， 是一种描述系统数据流程的主要工具， 它用一组符号来描述整个系统中信息的全貌， 综合地反映出信息在系统中的流动、 处理和存储情况。 数据流程图概括地描述了系统的内部逻辑过程， 是需求分析结果的表达工具， 也是系统设计的重要参考资料， 是系统设计的起点。

* 登录到主界面

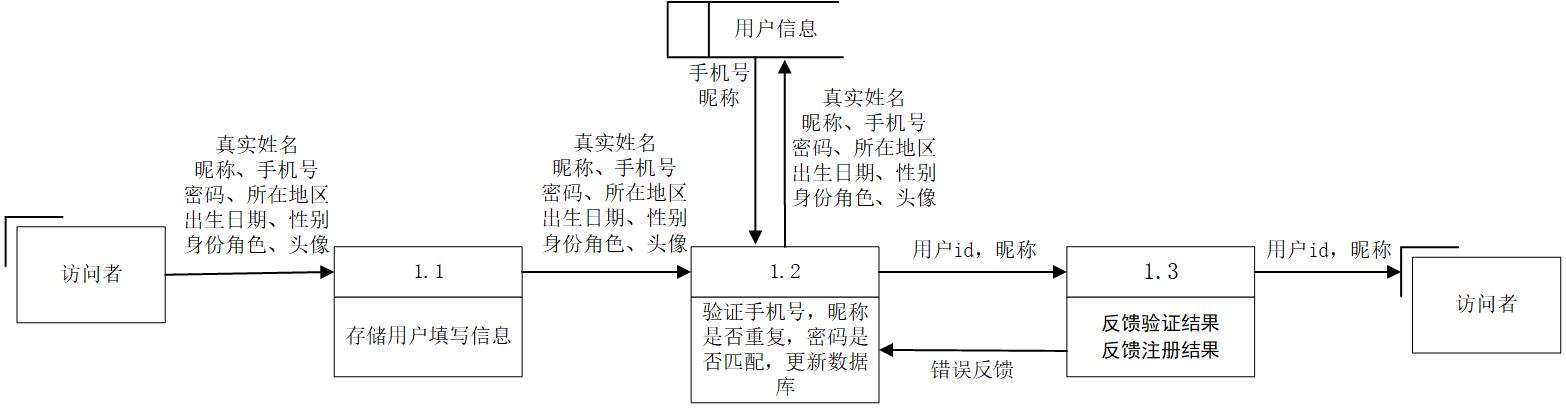
访客在登录界面输入手机号和密码，系统将这些信息存储并进行验证。如果手机号和密码匹配，系统会提取用户信息（包括用户ID、头像、昵称和角色），并将用户头像、本地地址和其他界面链接转化为展示头像，最终将用户ID、昵称和角色返回给访客，从而完成登录并进入主界面。登录失败时，系统会返回登录失败原因给访客。（如图2.1 登录主界面数据流程图所示）



### 图2.1 登录主界面数据流程图

* 注册

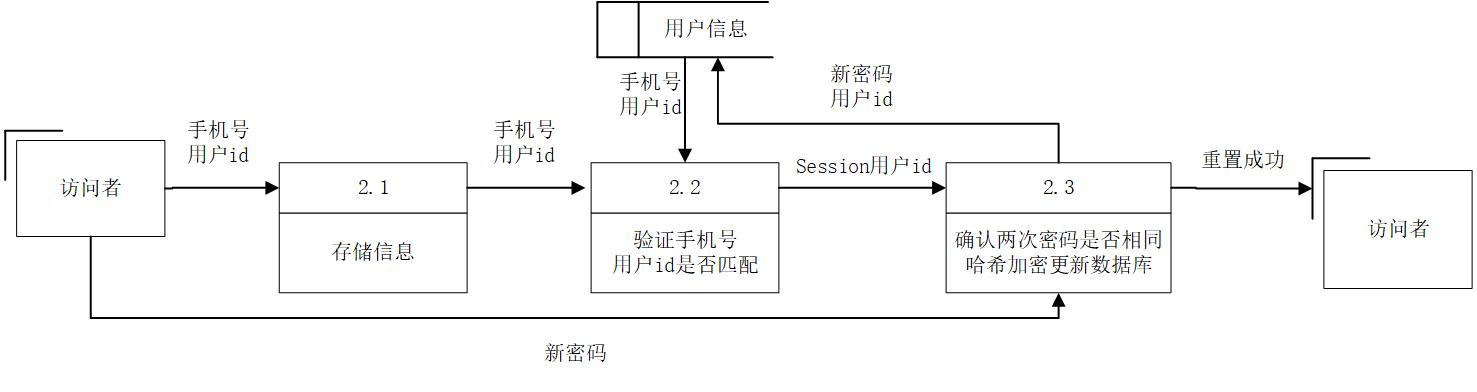
访客在注册界面填写真实姓名、昵称、手机号、密码、所在地、出生日期、性别、身份角色和头像等信息，系统将这些信息存储，并进行验证，检查手机号和昵称是否重复，密码是否匹配等。如果验证通过，系统会更新数据库，并返回用户ID和昵称；若验证不通过，系统会提供错误反馈，最终将验证结果和注册结果返回给访客。（如图2.2 注册数据流程图所示）



### 图2.2 注册数据流程图

* 重置密码

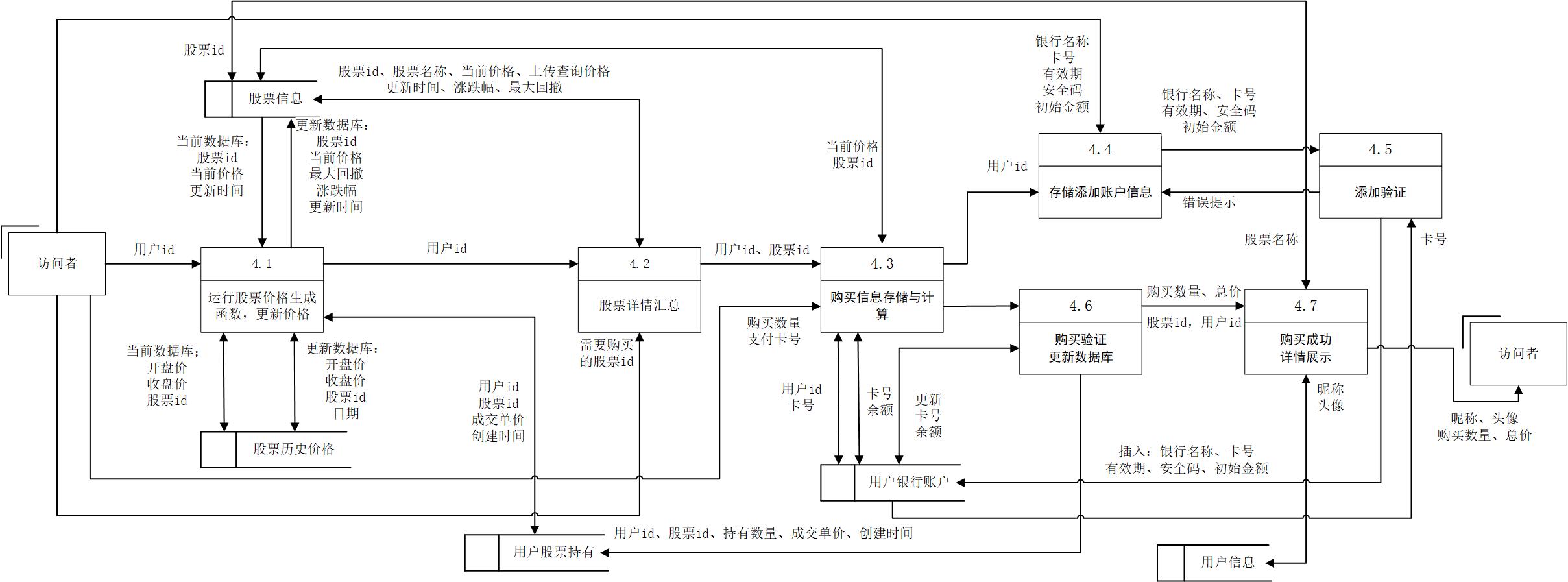
访客在重置密码界面输入手机号和用户ID，系统将这些信息存储并验证是否匹配。如果匹配成功，系统会创建一个会话，并允许访客输入新密码。系统确认新密码两次输入是否相同，并使用哈希算法更新数据库中的密码信息。如果重置成功，系统会反馈给访客重置成功的信息；若失败，则返回相应的错误原因。（如图2.3 重置密码数据流程图所示）



### 图2.3 重置密码数据流程图

* 股票

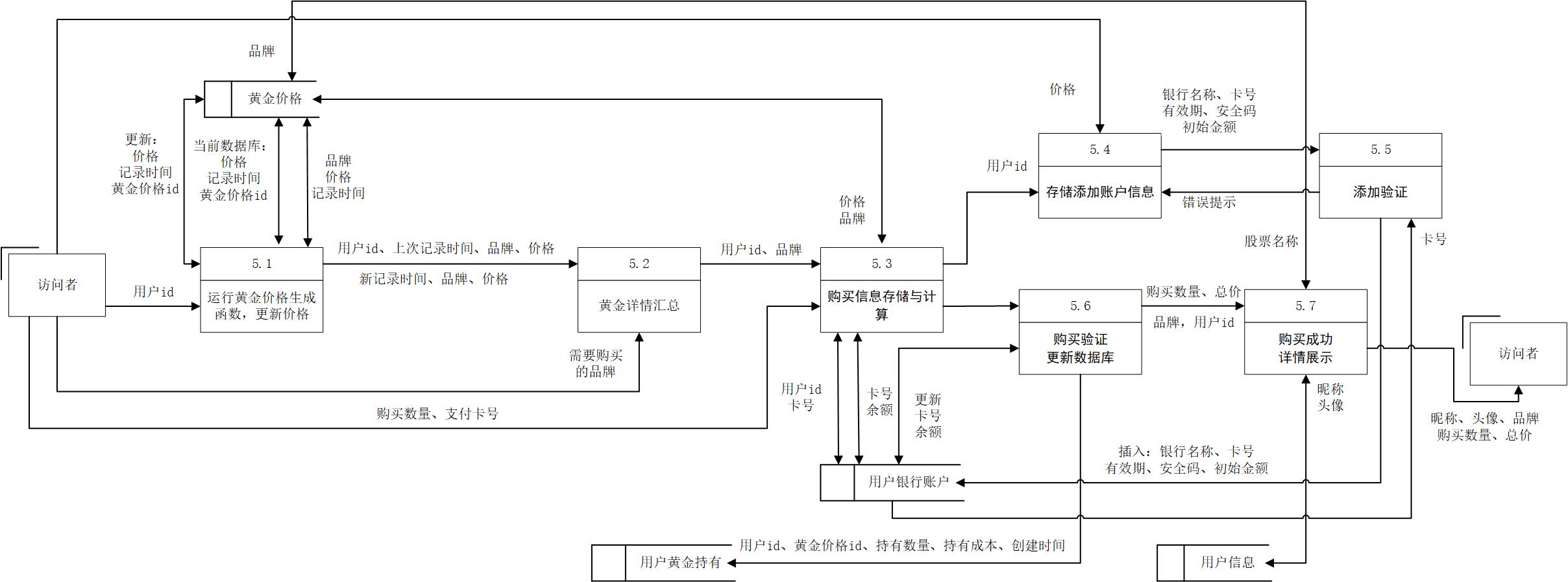
访客通过股票查询功能获取股票信息，包括股票名称、当前价格等。访客选择需要购买的股票后，系统会根据用户ID和购买量进行计算，并与用户银行账户进行支付验证。如果验证成功，系统会更新用户的持有股票信息和账户余额，并展示购买成功的详细信息；若验证失败，系统会显示错误提示。整个过程涉及股票信息的获取、更新、计算及银行账户的验证与更新。（如图2.4 股票数据流程图所示）



### 图2.4 股票数据流程图

* 黄金

访客通过系统查询黄金价格信息，系统根据当前数据库中的价格记录生成最新的黄金价格。访客选择需要购买的黄金品牌和数量，系统将购买信息存储并进行计算，包括用户ID、价格、品牌等。然后，系统会验证用户银行账户的支付信息，如果验证成功，系统会更新用户的黄金持有信息和账户余额，并展示购买成功的详细信息；如果验证失败，系统会显示错误提示。整个过程涉及黄金价格的查询与更新、购买信息的存储与计算，以及银行账户的验证与更新。（如图2.5 黄金数据流程图所示）



### 图2.5 黄金数据流程图

# 数据库设计

## 数据表设计

* 用户信息表设计过程

初始设计时，我们考虑用户注册和管理的基本需求，决定创建一个用户信息表来存储用户的基本信息。表单元素包括：用户ID、手机号、密码、头像、真实姓名、昵称、出生日期、性别、所在地区、角色、创建时间和更新时间。我们需要一个唯一标识符来区分不同用户，因此设置了用户ID作为主键，保证每个用户都有一个唯一的标识。手机号作为登录名，用于用户身份验证，非空且唯一。为了保护用户的隐私和账户安全，密码字段存储经过加密处理的用户密码。为了提升用户体验，头像字段存储用户头像的路径或编号，用户可选择预设头像或上传自定义头像。真实姓名、昵称、出生日期、性别和所在地区是可选字段，为提供更好的用户服务和个性化展示。角色字段用于区分用户权限，枚举类型包括‘个体用户’和‘管理员’。创建时间和更新时间字段记录用户信息的创建和最后一次更新的时间，默认值为当前时间，并在更新时自动修改。

### 图3.1 用户信息表

* 用户银行账户表设计过程

随着项目的推进，我们发现需要管理用户的银行账户信息，于是决定设计用户银行账户表来存储相关信息。表单元素包括：卡号、用户ID、银行名称、有效期、安全码、余额、创建时间和更新时间。每个银行账户都有唯一的卡号，作为主键以确保每个记录的唯一性。为了将银行账户与用户信息关联，我们增加了用户ID作为外键，关联用户信息表。银行名称有助于识别和分类银行账户，设为非空字段。银行账户的有效期和安全码是必须的信息，确保账户在有效期内使用和每次交易的安全性。余额字段用于存储银行账户的当前余额，设为非空字段，便于进行转账和查询操作。创建时间和更新时间字段记录账户信息的创建和最后一次更新的时间，确保信息的时效性。

### 图3.2用户银行账户表

* 交易信息表设计过程

为了记录用户之间的交易信息，我们决定设计交易信息表来存储每一笔交易的详细记录。表单元素包括：交易ID、转出卡号、转入卡号、交易金额、交易类型、备注、创建时间。每笔交易都有唯一的交易ID，作为主键，自动递增，确保记录的唯一性。为了记录交易的发起方和接收方，增加了转出卡号和转入卡号作为外键，关联用户银行账户表。交易金额字段存储每笔交易的金额，设为非空字段，确保交易记录的完整性。为了区分不同类型的交易，设立交易类型枚举字段，包括购物、服务、餐饮、交通、转账、黄金、股票、基金和其他。备注字段允许用户对交易进行注释和说明，设为可选字段。创建时间字段记录交易发生的时间，默认值为当前时间，确保交易记录的时效性。

### 图3.3 交易信息表

* 股票信息表设计过程

在项目发展过程中，我们需要管理和展示股票的相关信息，于是设计了股票信息表来存储这些数据。表单元素包括：股票ID、股票名称、初始价格、当前价格、上次查询价格、涨跌幅、最大回撤、创建时间和更新时间。每只股票都有唯一的股票ID，作为主键，自动递增，确保记录的唯一性。股票名称用于识别股票，设为非空字段。初始价格、当前价格和上次查询价格字段用于存储股票的价格信息，确保价格信息的完整性和时效性。为了分析股票的波动性和风险，涨跌幅和最大回撤字段记录股票的价格波动和最大损失。创建时间和更新时间字段记录股票信息的创建和最后一次更新的时间，确保信息的时效性和准确性。

### 图3.4 股票信息表

* 股票历史价格表设计过程

为了记录股票的历史价格信息，我们决定设计股票历史价格表来存储每日的开盘价和收盘价。表单元素包括：股票ID、日期、开盘价、收盘价。股票ID作为外键，关联股票信息表，用于标识每个股票的历史记录。日期字段记录每个交易日的日期，确保记录的唯一性和时间顺序。开盘价和收盘价字段分别记录每个交易日的开盘价和收盘价，确保价格数据的完整性。通过股票历史价格表，我们可以追踪每只股票的历史价格变化，进行趋势分析和预测。

### 图3.5 股票历史价格表

* 黄金价格表设计过程

为了模拟市场黄金价格波动，我们设计了黄金价格表来存储不同品牌的黄金价格。表单元素包括：黄金价格ID、价格、品牌、记录时间。每条黄金价格记录都有唯一的黄金价格ID，作为主键，自动递增，确保记录的唯一性。价格字段存储黄金的市场价格，品牌字段记录黄金的品牌信息，确保价格信息的完整性和可追溯性。记录时间字段记录价格生成的时间，确保价格数据的时效性。

### 图3.6 黄金价格表

* 用户黄金持有表设计过程

为了记录用户的黄金持有情况，我们设计了用户黄金持有表。表单元素包括：用户ID、黄金价格ID、持有数量、持有成本、创建时间。用户ID作为外键，关联用户信息表，确保每条记录属于一个特定用户。黄金价格ID作为外键，关联黄金价格表，记录用户持有的具体黄金。持有数量字段记录用户持有的黄金数量，持有成本字段记录用户购买黄金的总成本。创建时间字段记录持有记录的创建时间，确保数据的时效性和可追溯性。

### 图3.7 用户黄金持有表

* 用户股票持有表设计过程

为了记录用户的股票持有情况，我们设计了用户股票持有表。表单元素包括：用户ID、股票ID、持有数量、成交单价、创建时间。用户ID作为外键，关联用户信息表，确保每条记录属于一个特定用户。股票ID作为外键，关联股票信息表，记录用户持有的具体股票。持有数量字段记录用户持有的股票数量，成交单价字段记录用户购买股票的单价。创建时间字段记录持有记录的创建时间，确保数据的时效性和可追溯性。

### 图3.8 用户股票持有表

通过以上设计，我们逐步完善了数据库结构，确保各功能模块能够高效、安全地进行数据操作，确保数据的一致性和完整性。

## 范式分析

用户信息表（**用户ID**，手机号，密码，头像，真实姓名，昵称，出生日期，性别，所在地区，角色，创建时间，更新时间）

用户银行账户表（**卡号**，用户ID，银行名称，有效期，安全码，余额，创建时间，更新时间）

交易信息表（**交易ID**，转出卡号，转入卡号，交易金额，交易类型，备注，创建时间）

股票信息表（**股票ID**，股票名称，初始价格，当前价格，上次查询价格，涨跌幅，最大回撤，创建时间，更新时间）

股票历史价格表（**股票ID**，日期，开盘价，收盘价）

黄金价格表（**黄金价格ID**，价格，品牌，记录时间）

用户黄金持有表（**用户ID**，**黄金价格ID**，持有数量，持有成本，创建时间）

用户股票持有表（**用户ID**，**股票ID**，持有数量，成交单价，创建时间）

* 第一范式（1NF）分析

在满足第一范式要求的前提下，用户信息表、用户银行账户表、交易信息表、股票信息表、股票历史价格表、基金信息表、黄金价格表、用户理财偏好表、用户基金持有表、用户股票持有表和用户黄金持有表中均不存在可以被分割的列，且表中的各列无法重复，若录入相同的列，系统会报错。满足第一范式中的要求：每个表中每个属性不能再分，因此满足第一范式（1NF）。

* 第二范式（2NF）分析

在满足第一范式要求的前提下，所有表中的主键（如用户信息表中的用户ID、用户银行账户表中的卡号、交易信息表中的交易ID等）能够唯一标识每一行数据。表中的非主键字段完全依赖于主键，没有部分依赖。比如，用户信息表中的手机号、密码等字段完全依赖于用户ID，交易信息表中的转出卡号、交易金额等字段完全依赖于交易ID。因此，所有表均满足第二范式（2NF）的要求。

* 第三范式（3NF）分析

在满足第二范式的前提下，所有表中的非主键字段都直接依赖于主键，而不是通过其他非主键字段传递依赖。例如，用户信息表中的非主键字段（如手机号、密码等）直接依赖于用户ID，用户银行账户表中的非主键字段（如用户ID、银行名称等）直接依赖于卡号。因此，所有表均满足第三范式（3NF）的要求：非主属性不传递依赖于主属性。

### 图3.9 设计器

# 动态网页开发

## 登录界面

在登录界面通过 id="phone" 和 id="password" 来存储填写的手机号和密码。首先，前端验证输入的手机号和密码格式。validatePhone() 函数获取 id="phone" 输入框的值，并使用正则表达式 ^\d{11}$ 检查手机号是否为11位数字。如果手机号格式正确，获取指示器元素 id="phone-validity" 并设置其类名为 valid，显示内容为“✔️”；否则设置类名为 invalid，显示内容为“❌”。类似地，validatePassword() 函数获取 id="password" 输入框的值，并使用正则表达式 ^[a-zA-Z0-9@&#]{6,15}$ 检查密码是否为6到15位的字母、数字和特殊字符组合。如果密码格式正确，获取指示器元素 id="password-validity" 并设置其类名为 valid，显示内容为“✔️”；否则设置类名为 invalid，显示内容为“❌”。此外，togglePasswordVisibility() 函数切换密码输入框的类型在 text 和 password 之间，并相应更改切换按钮的文本。showHint() 和 hideHint() 函数分别显示和隐藏密码提示信息。（如图4.1 登录输入所示）

### 

### 图4.1 登录输入

用户点击 submit 登录后将信息提交到 登录处理.php，先启动 session 连接数据库。用 $\_POST 来获取用户输入的手机号和密码，将存储输入的 phone 信息用 SQL 查询 select 选择 from 用户信息表中的 用户 id，密码，角色信息，where 手机号 = ?，这里设计了占位符？，提高数据库安全性，然后进行 stmt 预编译查询逻辑匹配查询到的手机号和密码是否匹配，execute预处理部分，然后 $result 出判断的结果。如果 $result 的结果反馈 > 0，表示手机号存在，那么进行用户手机号匹配的userid、密码角色的查询。用password\_verify将密码转换为哈希加密后的密码与查询到的密码匹配验证。如果密码匹配成功，创建对应的变量和数组来保存用户的 user\_id 和用户角色保存到变量session中，并跳转到主界面。将这些信息保存到 session 中，是为了在后续的页面访问中能够识别用户身份并确定用户权限。如果用户 id 与查询不符合则提示回到登录界面提示手机号不存在。如果用户 id 和密码不匹配回到主界面提示密码错误。（如图4.2 登录失败所示）

### 

### 图4.2 登录错误

## 重置密码

* 重置前证

在 `重置验证.js` 中，通过 `validatePhone` 函数验证手机号格式。`validatePhone` 函数首先从输入框获取手机号的值，并使用正则表达式检查手机号是否符合格式要求。正则表达式要求手机号为11位数字。根据检查结果，函数会更新手机号有效性指示器的类名和内容。如果手机号格式正确，指示器的类名被设置为 `valid`，内容为勾号；如果格式错误，类名被设置为 `invalid`，内容为叉号，并调用 `showPopup` 函数显示错误提示信息。`showPopup` 函数通过获取提示框元素，设置其内容为错误信息，并将其显示出来，3秒后自动隐藏。（如如图4.3 重置密码所示）

### 

### 图4.3 重置密码

用户点击“开始重置”按钮后，信息通过 POST 方法提交到 `重置验证.php`。在 `重置验证.php` 中，首先使用 `$\_POST['phone']` 和 `$\_POST['user\_id']` 接收提交的数据，并将这些数据存储在变量 `$phone` 和 `$user\_id` 中。接着，连接数据库，并编写 SQL 查询语句，查找用户表中是否有与输入的手机号和用户ID匹配的记录。使用 `$conn->prepare()` 函数创建一个预处理语句对象 `$stmt`，通过`$stmt->bind\_param() 函数将`$phone`和`$user\_id`变量的值绑定到查询语句中的占位符上，调用`$stmt->execute()` 执行预处理语句，使用`$stmt->get\_result()`函数获取查询结果，并将其存储在变量`$result`中。通过这种方式，可以确保输入的数据不会以未经处理的形式直接进入SQL查询。

如果查询到匹配的记录，则验证成功，设置成功信息并重定向到登录界面；如果查询不到匹配的记录，则设置失败原因并返回重置验证界面。

验证成功后，用户进入重置密码界面。在`重置密码.js`中，通过`validatePassword`函数验证新密码格式。`validatePassword`函数从输入框获取新密码的值，并使用正则表达式检查密码是否符合格式要求。正则表达式规定，密码可以包含大小写字母、数字和特殊字符，长度必须在6到15个字符之间。根据检查结果，函数会更新密码有效性指示器的类名和内容。如果密码格式正确，指示器的类名被设置为`valid`，内容为勾号；如果格式错误，指示器的类名被设置为`invalid`，内容为叉号。(如图4.4 重置输入)

### 

### 图4.4 重置输入

同时，通过`confirmPasswordMatch`函数验证新密码和确认密码是否一致。`confirmPasswordMatch`函数从输入框获取新密码和确认密码的值，并检查两者是否一致。如果一致，指示器的类名被设置为`valid`，内容为勾号；如果不一致，指示器的类名被设置为`invalid`，内容为叉号。用户输入的新密码和确认密码通过验证后，点击“提交”按钮，将新密码通过POST方法提交到`重置密码.php`进行处理。`重置密码.php`接收新密码数据，首先使用`$\_POST['password']`和`$\_POST['user\_id']`接收提交的密码和用户ID，并分别存储到`$password`和`$user\_id`变量中。接着，连接数据库，并编写SQL更新语句，更新用户表中的密码信息。使用 `$conn->prepare()` 函数创建一个预处理语句对象`$stmt`，通过 `$stmt->bind\_param()` 函数将`$password`和`$user\_id`变量的值绑定到更新语句中的占位符上，调用`$stmt->execute()`执行预处理语句。如果更新成功，设置成功信息并重定向到登录界面；如果更新失败，则设置错误信息并返回重置密码界面。

## 注册

* 注册填写

在登录界面上，用户可以点击“新用户前往注册”按钮，页面将跳转到注册界面。在注册界面，用户需要填写真实姓名、昵称、手机号、密码、确认密码、选择省份、出生日期、性别、身份角色，然后选择或上传头像。

在`注册界面.js` 中，通过`validatePassword` 函数验证密码格式。`validatePassword` 函数从输入框获取密码的值，并使用正则表达式检查密码是否符合格式要求。正则表达式规定，密码可以包含大小写字母、数字和特殊字符，长度必须在6到15个字符之间。根据检查结果，函数会更新密码有效性指示器的类名和内容。如果密码格式正确，指示器的类名被设置为 `valid`，内容为勾号；如果格式错误，指示器的类名被设置为 `invalid`，内容为叉号。

同时，通过`confirmPasswordMatch` 函数验证新密码和确认密码是否一致。`confirmPasswordMatch` 函数从输入框获取新密码和确认密码的值，并检查两者是否一致。如果一致，指示器的类名被设置为 `valid`，内容为勾号；如果不一致，指示器的类名被设置为 `invalid`，内容为叉号。

省份和城市的选择通过 `updateCity` 和 `updateDistrict` 函数实现。当用户选择省份时，`updateCity` 函数会被调用，获取所选下拉选框的provinceid根据选择的省份更新城市选项。函数获取省份选项的值，并重置城市选择框的内容。如果省份在城市选项中存在，则遍历城市选项，创建新的选项元素并添加到城市选择框中。接着调用 `updateDistrict` 函数更新区县选项。同样，当用户选择城市时，`updateDistrict` 函数会被调用，根据选择的城市更新区县选项。函数获取城市选项的值，并重置区县选择框的内容。如果城市在区县选项中存在，则遍历区县选项，创建新的选项元素并添加到区县选择框中。（如图 4.5注册城市选择所示）

### 

### 图4.5 注册城市选择

在注册界面中，用户可以选择预定义的头像或上传自定义头像。通过 `selectAvatar` 函数，用户可以选择预定义的头像。函数会设置隐藏输入框的头像ID值，并更新选中样式。例如，当用户点击一个头像时，`selectAvatar` 函数会被调用，传入头像的ID值，并将该值设置到隐藏输入框 `avatar` 中。同时，函数会移除所有头像的选中样式，并为选中的头像添加选中样式。

用户也可以通过 `triggerFileUpload` 函数触发文件选择窗口上传自定义头像。`triggerFileUpload` 函数会模拟点击文件输入框，打开文件选择窗口。当用户选择文件后，`previewImage` 函数会被调用。该函数获取选中的文件，并使用 `FileReader` 对象读取文件内容。读取完成后，函数会创建一个新的图像元素，将文件内容作为图像源，并将图像显示在预览区域。同时，隐藏输入框 `avatar` 的值会被设置为 `'custom'`，表示用户选择了自定义头像，并移除所有预定义头像的选中样式。（如图 4.6选择头像所示）

### 

### 图4.6 选择头像

* 注册提交验证

用户填写完所有信息后，点击“注册”按钮，信息通过 POST 方法提交到 `注册处理.php`。在 `注册处理.php` 中，首先使用 `$\_POST` 接收提交的数据，并将这些数据存储在相应的变量中。

点击注册后，首先进行数据的验证，包括通过检查手机号是否存在、密码是否符合要求、两次输入的密码是否一致等、。如果任何一项验证失败，页面将返回注册界面，并显示相应的错误信息，如果通过则对输入的密码进行hash加密存储到加密密码变量。

在处理头像选择时，如果用户选择了预定义头像，`selectAvatar` 函数会设置隐藏输入框的头像ID值，并在提交表单时将该值与其他用户信息一起提交。如果用户上传自定义头像则查询用户max\_id,执行max\_id+1的计算结果存储为新的用户的id，`previewImage` 函数会在本地预览图像如果检测到对应名称的文件数大于0，则获取选中的文件，创建reader onload回调函数创建新的图像元素，设置图像源为文件的内容，将图像添加到预览区域，并在提交表单时将隐藏输入框的值设置为 `'custom'`。在 `注册处理.php` 中，如果检测到用户选择了自定义头像，系统会处理上传的图像文件，将其保存到服务器上的指定目录中，并使用唯一的文件名保存文件。文件名会使用自动生成的用户ID加5的数值，然后命名为“用户头像+计算的数值+.jpg”。例如，如果生成的用户ID是1001，则文件名会是“用户头像1006.jpg”。保存文件后，将头像id存储到$new\_avatar\_id变量

接着，编写SQL插入语句，INSERT将不同变量存储的用户添加的信息插入到用户表中。对插入结果设置query值。如果query===true，则插入成功，设置成功信息并重定向到注册成功展示界面，展示用户的用户id；如果插入失败，则设置错误信息并返回注册界面。（如图 4.7注册成功所示）

### 

### 图4.7 注册成功

## 主界面

进入主界面后，用户可以看到左上方显示的头像和用户名。用户点击头像后，头像会放大显示。页面中还显示了用户的真实姓名和用户ID，点击姓名和ID会跳转到用户信息界面。此外，页面还有四个按钮，分别对应转账、黄金、股票和总资产，点击这些按钮会跳转到对应的界面。点击退出登录按钮会删除当前的Session登录状态并回到登录界面。（如图 4.8主界面所示）

### 

### 图4.8 主界面

* 显示用户头像和信息

在主界面的顶部栏，有一个包含用户头像和信息的部分。用户的头像显示在左上方，点击头像后会调用 viewFullImage 函数。该函数接收头像图片的路径作为参数，通过获取全屏显示的覆盖层元素和图像元素，将传入的图片路径设置为全屏图像元素的 src 属性，并将覆盖层的显示样式设置为 flex，从而显示覆盖层和放大的头像图像。当用户点击覆盖层时，会调用 closeFullImage 函数，将覆盖层的显示样式设置为 none，隐藏覆盖层和全屏图像。（如图 4.9头像预览所示）

### 

### 图4.9 头像预览

在PHP代码中，用户头像、昵称、用户ID和角色信息的部分通过服务器端获取数据，并嵌入到HTML中。具体流程如下：服务器端获取用户的头像路径、昵称、用户ID和角色信息，然后将这些数据通过PHP的 htmlspecialchars 函数进行处理，确保安全后嵌入到HTML中。头像图片元素的 src 属性使用PHP生成的路径，并设置 onclick 事件，当用户点击头像时调用JavaScript函数 viewFullImage 显示全屏头像。用户昵称和用户ID显示在可点击的链接中，链接指向用户信息页面。用户角色信息也显示在顶部栏中。

在头像显示部分，PHP代码根据用户的头像ID决定显示哪个头像。首先，将用户头像ID转换为整数，并初始化头像路径和描述。然后，通过判断头像ID是否在预定义范围内（1到5）来设置对应的头像路径和描述。如果头像ID在预定义范围内，使用 switch 语句选择相应的头像路径和描述。如果头像ID不在预定义范围内，则认为用户上传了自定义头像，将路径设置为存储自定义头像的目录，并加上时间戳以确保浏览器每次都能加载最新的图像。最终，头像路径和描述信息将被嵌入到HTML中，用于显示用户头像。

* 跳转到用户信息界面

页面中还显示了用户的真实姓名和用户ID。用户点击这些元素时，会通过一个 onclick 事件处理程序跳转到用户信息界面。当用户点击姓名或ID时，onclick 事件处理程序将 window.location.href 属性设置为用户信息界面的URL，从而实现页面跳转到用户信息界面。（如图 4.10用户信息跳转所示）

### 

### 图4.10 用户信息跳转

* 跳转到不同功能界面

页面上有四个按钮，分别对应转账、黄金、股票和总资产。这些按钮设置了 onclick 事件处理程序，当用户点击按钮时，页面会跳转到对应的功能界面。每个按钮的 onclick 事件处理程序会将 window.location.href 属性设置为对应功能界面的URL，从而实现页面跳转到相应的功能界面。

* 退出登录

点击退出登录按钮会调用 logout 函数，将页面跳转到处理退出登录的页面 退出处理.php。logout 函数将 window.location.href 属性设置为 退出处理.php。在 退出处理.php 中，当前的Session变量会被删除，并且用户将被重定向到登录界面。

## 股票

* 股票价格生成

在 股票价格生成.php 文件中，系统首先包含 数据库连接.php 文件，使用 PDO 对象建立与数据库的连接。设计一个 connectDatabase() 函数来处理数据库连接，并返回数据库连接对象。

使用 SELECT 语句查询股票信息表中的所有股票ID、当前价格和更新时间，设计一个 getStockInfo($db) 函数，该函数接受数据库连接对象作为参数，并返回查询结果数组 $stocks，其中包含每只股票的ID、当前价格和上次更新时间。

获取当前系统时间，使用 new DateTime() 函数获取当前时间，并存储在 $current\_time 变量中。

设计一个 calculateTimeDifference($lastUpdate, $current\_time) 函数，计算每只股票的上次更新时间与当前时间的差异天数，并返回 $days\_diff 变量。

设计一个 getPriceChangeTimes($days\_diff, $lastUpdate, $current\_time) 函数，根据差异天数和时间段确定价格变动次数：

如果 $days\_diff 小于等于1，分时间段确定价格变动次数：

获取上次更新时间的小时和分钟，分别存储在 $last\_hour 和 $last\_minute 变量中。

获取当前时间的小时和分钟，分别存储在 $current\_hour 和 $current\_minute 变量中。

* 当上次更新时间在凌晨0点至9点29分之间，并且当前时间也在相同时间段，则价格变动次数为0。当上次更新时间在此时间段，而当前时间在9点30分至12点59分之间，则价格变动次数为1。当上次更新时间在此时间段，而当前时间在13点至14点59分之间，则价格变动次数为2。当上次更新时间在此时间段，而当前时间在15点至23点59分之间，则价格变动次数为2。
* 当上次更新时间在9点30分至12点59分之间，并且当前时间也在相同时间段，则价格变动次数为0。当上次更新时间在此时间段，而当前时间在13点至14点59分或15点至23点59分之间，则价格变动次数为1。
* 当上次更新时间在13点至14点59分之间，无论当前时间在哪个时间段，价格变动次数均为0。

如果 $days\_diff 大于1，根据时间段计算总变动次数：

* 当当前时间在0点至9点29分之间，则价格变动次数为2倍的天数。
* 当当前时间在9点30分至12点59分之间，则价格变动次数为2倍的天数加1。
* 当当前时间在13点至14点59分或15点至23点59分之间，则价格变动次数为2倍的天数加2。

使用几何布朗运动模型计算价格变动，设置预期回报率（$mu）和波动率（$sigma）。设计一个 calculateNewPrice($currentPrice, $mu, $sigma, $changeTimes) 函数，该函数接受当前价格、预期回报率、波动率和变动次数作为参数，返回新的价格。

在 foreach 循环中，对每只股票进行多次价格计算，模拟价格波动。使用随机数生成函数生成随机变量，模拟价格的随机变化。

几何布朗运动公式，见式4.1所示：

（式4.1）

是新股票价格

是当前股票价格，即$currentPrice

是预期回报率。

是波动率。

是时间间隔。

是随机变量（标准正态分布）。

转化为代码的过程：初始化 ​ 为当前股票价格 $currentPrice。使用 mt\_rand() 函数生成随机数并转换为标准正态分布变量 。计算新的股票价格，循环变动次数 $changeTimes 次，更新​ 为新的 ​，最后的 作为计算得到的新的股票价格。将计算结果存储在 $newPrice 变量中。最后使用如下代码来进行计算：

for ($i = 0; $i < $total\_changes; $i++) { $epsilon = mt\_rand() / mt\_getrandmax(); $epsilon = sqrt(-2 \* log($epsilon)) \* cos(2 \* M\_PI \* mt\_rand() / mt\_getrandmax()); $newPrice = $newPrice \* exp(($mu - $sigma \* $sigma / 2) \* 1 + $sigma \* $epsilon \* sqrt(1)); }

将计算得到的新的当前价格更新到股票信息表，设计一个updateStockPrice($db,$stockID,$newPrice)函数，接受数据库连接对象、股票ID和新的当前价格作为参数，执行 UPDATE 语句。对于每只股票，在foreach循环中逐条执行更新操作，更新当前价格和上次查询价格。

* 开盘收盘价的计算

查询用户股票持有表中的创建时间最新的日期和成交单价，设计一个getLatestTransactionDateAndPrice($db)函数，从用户股票持有表中获取创建时间最新的日期和成交单价。使用SELECT语句查询最新的创建日期和成交单价。根据创建日期分组，找到每组中最早和最晚的成交单价，分别作为开盘价和收盘价。使用INSERT INTO语句，将这些数据插入到股票历史价格表中。

* 最大回撤

从用户股票持有表中获取股票ID、开盘价和收盘价，设计一个 getOpenClosePrices($db, $stockID) 函数，从股票历史价格表中获取开盘价和收盘价。使用 SELECT 语句查询特定股票的开盘价和收盘价。设计一个 calculateMaxDrawdown($openPrices, $closePrices) 函数，找出开盘价和收盘价中的最大值和最小值，计算最大回撤。

最大回撤公式，见式4.2所示  
 （式4.2）

转化为代码的过程：使用 max() 函数找出 $openPrices 和 $closePrices 数组中的最大值。使用 min() 函数找出 $openPrices 和 $closePrices 数组中的最小值。计算最大回撤，将结果存储在 $maxDrawdown 变量中。在 foreach 循环中，对每只股票分别计算最大回撤。

将最大回撤更新到股票信息表，设计一个 updateMaxDrawdown($db, $stockID, $maxDrawdown) 函数，执行 UPDATE 语句。

* 涨跌幅计算

从股票历史价格信息表中获取半个月内的收盘价和开盘价，设计一个 getRecentPrices($db, $stockID) 函数，从股票历史价格信息表中获取半个月内的收盘价和开盘价。使用 SELECT 语句查询特定日期范围内的开盘价和收盘价。设计一个 calculatePriceChange($openPrices, $closePrices) 函数，筛选出最近日期，并找到收盘价的最大值和开盘价的最小值，计算涨跌幅。

涨跌幅公式，见式4.3所示 （式4.3）

其中最大收盘价是半个月内所有收盘价中的最大值，最小开盘价是半个月内所有开盘价中的最小值。

转化为代码的过程：使用 max() 函数计算 $closePrices 数组中的最大值。使用 min() 函数计算 $openPrices 数组中的最小值。计算涨跌幅，将结果存储在 $priceChange 变量中。在 foreach 循环中，对每只股票分别计算涨跌幅。

将涨跌幅更新到股票信息表，设计一个 updatePriceChange($db, $stockID, $priceChange) 函数，执行 UPDATE 语句。

对每只股票，使用 SELECT MAX(日期) AS 最新日期 FROM 股票历史价格 WHERE 股票ID = ? 语句查询最新日期。准备和执行查询语句，获取最新日期，将其存储在 $latest\_date 变量中。使用 SELECT 开盘价, 收盘价 FROM 股票历史价格 WHERE 股票ID = ? AND 日期 BETWEEN ? AND ? 语句查询指定日期范围内的开盘价和收盘价。将查询结果存储在 $prices 数组中，通过 while 循环逐行获取查询结果并存储。

从 $prices 数组中提取开盘价和收盘价列，并分别存储在 $open\_prices 和 $close\_prices 数组中。使用 max() 函数计算 $close\_prices 数组中的最大收盘价，使用 min() 函数计算 $open\_prices 数组中的最小开盘价。如果最小开盘价为0，涨跌幅设为0，否则使用式4.3的公式计算。

使用 UPDATE 语句将计算得到的涨跌幅更新到股票信息表中，执行查询语句并确认更新成功。

* 股票详情

在股票详情界面，系统通过查询数据库获取股票的详细信息，并将这些信息显示在网页上。首先，系统引用股票价格生成脚本，通过执行包含股票价格生成逻辑的文件来生成最新的股票价格并更新数据库。引用该文件后，变量会包含股票价格生成脚本中的返回值（如果有的话）。接下来，系统通过SQL查询从数据库中获取股票的详细信息，包括股票ID、名称、当前价格、上次查询价格、更新时间、涨跌幅和最大回撤。查询结果存储在一个变量中。如果查询失败，系统会输出错误信息并终止执行。

系统使用循环遍历查询结果，将每行数据存储在一个数组中。方法用于获取结果集中的每一行，并返回一个关联数组。循环结束后，系统关闭数据库连接。然后在HTML页面中，系统通过HTML和PHP相结合的方式显示股票的详细信息，包括股票名称、上次查询价格、当前价格、涨跌幅和最大回撤。

系统首先检查 `$stocks` 数组是否为空。如果数组为空，页面会显示“暂无数据”的提示。此时，系统通过 `if (empty($stocks))` 判断数据是否为空，如果为空则输出提示信息。如果 `$stocks` 数组不为空，系统会使用 `foreach` 循环遍历数组中的每只股票，并将每只股票的信息显示在一个HTML表格行 (`<tr>`) 中。股票名称会作为一个链接，指向股票购买页面。链接的URL通过 `urlencode` 函数进行编码，以确保URL中的特殊字符被正确处理。系统显示每只股票的上次查询价格，并在价格下方显示上次查询记录的时间。时间信息来自于 `$price\_info['last\_record\_time']` 变量。系统显示每只股票的当前价格，并在价格下方显示当前查询的时间。时间信息来自于 `$price\_info['current\_time']` 变量。系统显示每只股票的涨跌幅和最大回撤信息。这些数据直接从 `$stocks` 数组中获取，并通过 `htmlspecialchars` 函数进行转义后显示在表格中。

系统通过包含股票价格生成脚本，执行其中的逻辑来生成最新的股票价格并更新数据库。这个文件负责生成最新的股票价格，并更新数据库中的相关信息。引用这个脚本后，页面会在显示股票详情之前，确保股票价格是最新的。系统连接数据库，并执行查询语句从数据库中获取所有股票的详细信息，包括股票ID、名称、当前价格、上次查询价格、更新时间、涨跌幅和最大回撤。查询结果存储在一个变量中。方法用于执行查询，并返回一个结果对象。如果查询失败，系统会输出错误信息并终止执行。系统使用循环遍历查询结果，将每行数据存储在一个数组中。方法用于获取结果集中的每一行，并返回一个关联数组。每次循环迭代都会将当前行的数据添加到数组中。循环结束后，系统关闭数据库连接。（如图 4.11股票详情所示）

### 

### 图4.11 股票详情

* 股票购买

在股票界面，用户可以查看股票的详细信息点击具体的股票名称则进行对应的购买界面。系统在页面加载时，通过SQL查询从数据库中获取所选股票的详细信息，包括股票ID、名称和当前价格。查询结果存储在变量中，系统同时显示一个下拉菜单供用户选择支付卡，用户所有的银行账户会列在下拉菜单中。这些银行账户信息通过查询用户的银行账户表获取，并填充到下拉菜单中。（如图 4.12股票购买所示）

### 

### 图4.12 股票购买

用户选择一个银行账户后，系统通过AJAX请求查询该账户的余额并实时显示。JavaScript函数 fetchBalance 负责发送请求并更新余额显示。通过JavaScript创建一个XMLHttpRequest对象，用于与服务器进行交互。配置请求，设置请求方法为POST，目标URL为获取余额.php，并将请求设置为异步。设置请求头，指定发送的数据类型为URL编码格式。监听请求状态变化，当请求完成且响应成功时，获取页面中具有id为availableBalance的元素，并设置其文本内容为返回的余额数据。发送请求数据，其中包含所选账户的账号信息。

用户输入想要购买的股票数量，系统自动根据当前价格计算总价并实时显示。JavaScript函数calculateTotal负责计算总价并更新显示。通过JavaScript DOM获取页面中具有id为quantity的输入框的值，即用户输入的购买数量。获取页面中具有name属性为pricePerShare的输入框的值，即股票的单价。将购买数量与股票单价相乘，计算总价，并存储在变量中。获取页面中具有id为totalPrice的元素，并设置其文本内容为计算得到的总价。使用toFixed(2)方法确保总价显示为两位小数。

用户填写完购买信息后点击“购买”按钮，系统接收到购买请求并进行处理。用户点击“购买”按钮后，系统会验证用户账户的余额，确保余额足够支付购买总价。如果余额不足，系统会提示用户余额不足，终止交易。（如图4.13 购买确认所示）

### 

### 图4.13 购买确认

这一步通过查询银行账户表获取用户的账户余额并进行比较实现。如果余额足够，系统会扣除购买总价，并更新用户银行账户的余额。系统执行一个更新操作，将新的余额值写入数据库。接着，系统将购买记录插入到用户股票持有表中，更新用户持有的股票数量和成交单价。系统执行一个插入操作，将购买记录写入数据库。系统记录此次交易信息，生成交易记录，包含转出卡号、转入卡号、交易金额和交易类型等，并将其插入到交易记录表中。

* 账户添加

在账户添加界面，用户可以通过填写表单来添加新的银行账户。首先，用户在表单中输入银行名称、卡号、有效期、安全码和余额。JavaScript函数 validateForm 在表单提交时执行，验证输入的银行账户信息是否符合要求。具体步骤如下：

通过JavaScript DOM获取页面中具有id为cardNumber的输入框的值，并去除空格。检查卡号是否为11到16位数字，如果不符合，加入错误信息。通过JavaScript DOM获取页面中具有id为cardExpiry的输入框的值，并检查其格式是否为MM/YY。如果格式正确，进一步检查日期的有效性。通过JavaScript DOM获取页面中具有id为security\_code的输入框的值，检查安全码是否为3位数字。如果存在任何错误信息，调用 showAlert 函数显示错误信息，并阻止表单提交。

当用户输入卡号时，JavaScript函数 formatCardNumber 自动格式化卡号，使其每四位数字之间有一个空格。具体步骤是，通过JavaScript DOM获取输入框的值，去除所有空格，然后重新插入空格，使卡号每四位数字分组显示。

用户填写完表单并通过验证后点击“提交”按钮，系统接收到提交请求并进行处理。系统首先验证用户输入的银行账户信息，确保信息的完整性和有效性。如果信息有误，系统会提示用户重新输入。验证通过后，系统将新的银行账户信息插入到用户银行账户表中，并提示用户账户添加成功。系统跳转回股票购买界面，用户可以在购买界面的支付卡下拉菜单中看到新添加的银行账户。（如图4.14 添加账户所示）

### 

### 图4.14 添加账户

* 购买详情

当用户完成购买后，系统会显示购买详情界面。页面会从数据库中查询并显示用户的交易信息，包括股票名称、购买数量、成交单价和总价。这些信息是在用户提交购买请求时保存到数据库中的。

系统先启动 session 连接数据库。用 $\_POST 来获取用户的交易ID，将其存储在变量中。然后，使用 SQL 查询从交易表中选择交易记录，包括股票名称、购买数量、成交单价和总价。SQL 查询语句为 SELECT 股票名称, 购买数量, 成交单价, 总价 FROM 交易表 WHERE 交易ID = ?，这里使用了占位符 ? 来防止SQL注入，提高数据库安全性。

创建预处理语句对象，使用 prepare($sql) 函数。接着，将交易ID绑定到SQL查询中的占位符上。然后执行预处理语句，并使用绑定查询结果到变量。最后，使用 fetch() 获取结果并存储在变量中。

系统还会显示用户使用的支付卡号和支付后的账户余额。这些信息通过查询用户的银行账户表获取，并显示在页面的相应位置。先准备SQL查询语句 SELECT 卡号, 余额 FROM 银行账户表 WHERE 用户ID = ?，用来获取用户使用的支付卡号和支付后的账户余额。然后，创建预处理语句对象，使用 prepare($sql) 函数。接着，使用绑定参数，将用户ID绑定到SQL查询中的占位符上。调用执行预处理语句，并使用绑定查询结果到变量。最后，使用 fetch() 获取结果并存储在变量中。

系统会显示此次交易的时间。交易时间是在用户提交购买请求时保存到数据库中的，并通过SQL查询获取。准备SQL查询语句 SELECT 交易时间 FROM 交易表 WHERE 交易ID = ?，用来获取交易时间。创建预处理语句对象，使用 prepare($sql) 函数。接着，使用绑定参数，将交易ID绑定到SQL查询中的占位符上。调用执行预处理语句，并使用绑定查询结果到变量。最后，使用 fetch() 获取结果并存储在变量中。

在页面加载时，系统会执行上述SQL查询，从数据库中获取交易信息、支付信息和交易时间。查询结果存储在相应的变量中，并在页面的相应位置显示。用户可以点击“返回”按钮，返回股票购买界面或股票详情界面。点击“返回”按钮时，系统会将用户重定向到股票购买界面或股票详情界面。（如图4.15 购买成功所示）

### 

### 图4.15 购买成功

## 黄金

* 黄金价格生成

在黄金价格生成过程中，系统首先通过包含数据库连接文件来确保与数据库的连接正常。接下来，系统定义了各个黄金品牌的系数，这些系数用于计算每个品牌的具体价格。品牌系数包括纯度、工艺和尺寸的调整系数。

定义品牌参数的设计：

品牌参数定义如下，每个品牌包含纯度、工艺和尺寸的调整系数。例如，“小铭黄金”的纯度系数为1.01，工艺系数为1.01，尺寸系数为0.99；“小杨黄金”的纯度系数为0.98，工艺系数为1.00，尺寸系数为1.02；“小孙黄金”的纯度系数为1.00，工艺系数为1.01，尺寸系数为0.99。

获取当前市场均价及上次记录时间：

系统通过SQL查询从数据库中获取市场均价和上次更新时间。如果查询结果不为空，则从结果中提取价格和记录时间；如果查询结果为空，则设定初始市场均价为 548.0 元，并将上次更新时间设定为当天的时间。

计算时间差：

系统使用 strtotime 函数将上次记录时间字符串转换为时间戳，并计算当前时间和上次记录时间之间的差值（以秒为单位），存储在变量 $time\_difference 中。

价格波动算法：

系统通过设定的参数和随机数生成价格波动。使用自定义的模拟函数 simulatePriceFluctuation 计算价格波动，具体过程包括以下几个步骤：

生成一个范围在 [-1, 1] 的随机数，模拟标准正态分布的随机波动**：**$epsilon = (mt\_rand() / mt\_getrandmax()) \* 2 - 1;

**计算新价格**：使用集合布朗运动模型，使用当前价格乘以带有正弦函数波动和噪声的系数，得到新的价格。具体的数学公式见式4.4所示：

（式4.4）

其中公式与代码一一对应关系如下：

是新黄金价格，代码中的 $new\_price = $current\_price \* exp(...)

并确保价格在合理范围内：return round(max(540, min(550, $new\_price)), 2);

是当前黄金价格，代码中的 $current\_price

是预期回报率，代码中的 $mu

是波动率，代码中的 $sigma

是时间间隔，代码中的 $delta\_t = $time\_seconds / 3600，

将时间差 $time\_seconds 转换为小时，以小时为单位计算时间步长

是准正态分布的随机数，代码中的 $epsilon = (mt\_rand() / mt\_getrandmax()) \* 2 – 1

，代码中的 exp(($mu - 0.5 \* $sigma \* $sigma) \* $delta\_t + $sigma \* sqrt($delta\_t) \* $epsilon)

计算各品牌价格，系统使用市场均价和品牌系数计算每个品牌的具体价格。具体过程包括以下步骤：

遍历品牌列表：对于每个品牌，获取其纯度、工艺和尺寸系数。

计算品牌价格：使用如下式4.5所示的计算品牌价格：

（式4.5）

例如，如果市场均价为550元，某品牌的纯度系数为1.01，工艺系数为1.01，尺寸系数为0.99，则该品牌的价格为（见式4.6所示）：

（式4.6）

更新品牌价格：

使用 UPDATE 语句更新各品牌价格和当前时间。系统使用预处理语句对象执行更新操作，将计算出的品牌价格和当前时间绑定到SQL查询中的占位符上。

查询上次记录的品牌价格：

系统通过SQL查询从数据库中获取上次记录的每个品牌的价格和更新时间，并将结果存储在变量中。

返回结果：

系统返回包括上次记录时间、上次市场均价、上次品牌价格、当前时间、当前市场均价和当前品牌价格的信息。

* 黄金详情

页面加载时，系统通过SQL查询从数据库中获取最新的黄金价格信息，包括市场均价和各品牌价格。系统使用SQL查询语句 SELECT 品牌, 当前价格, 上次更新时间 FROM 黄金价格表 来获取这些信息。创建预处理语句对象，使用 prepare($sql) 函数。接着，执行预处理语句，并使用 bind\_result 绑定查询结果到变量 $brand, $currentPrice, $lastUpdateTime。最后，使用 fetch() 获取结果并存储在变量中。

系统将上次记录的黄金价格信息显示在左侧信息框中。上次记录的信息包括记录时间、市场均价和各品牌的价格。这些数据通过SQL查询从黄金价格历史表中获取。系统使用SQL查询语句 SELECT 记录时间, 市场均价, 品牌, 价格 FROM 黄金价格历史表 ORDER BY 记录时间 DESC LIMIT 1 来获取上次记录的信息。创建预处理语句对象，使用 prepare($sql) 函数。接着，执行预处理语句，并使用 bind\_result 绑定查询结果到变量 $recordTime, $marketPrice, $brand, $price。最后，使用 fetch() 获取结果并存储在变量中。

当前记录的信息显示在右侧信息框中。当前记录包括最新的记录时间、市场均价和各品牌的价格。各品牌的价格作为链接，点击后用户将被带到相应品牌的黄金购买页面。系统使用类似于上次记录的信息获取方法，通过SQL查询从黄金价格表中获取当前记录的信息。创建预处理语句对象，使用 prepare($sql) 函数。接着，执行预处理语句，并使用 bind\_result 绑定查询结果到变量 $currentRecordTime, $currentMarketPrice, $currentBrand, $currentPrice。最后，使用 fetch() 获取结果并存储在变量中。

用户点击刷新按钮后，系统重新运行黄金价格生成部分，获取最新的市场均价和各品牌价格，并更新页面内容。这一功能通过调用黄金价格生成的相关函数实现。系统会再次查询数据库，获取最新的黄金价格信息，并更新页面显示。

在 黄金管理.js 文件中，定义了页面加载和用户交互的逻辑。页面加载时，系统获取最新的黄金价格信息，并显示在页面上。系统通过 JavaScript 的 DOM 操作获取页面中的元素，并为各个交互元素（如支付卡选择、数量输入等）添加事件监听器。

当用户选择支付卡时，系统会通过 AJAX 请求查询该账户的余额并实时显示。JavaScript 函数 fetchBalance 负责发送请求并更新余额显示。通过创建一个 XMLHttpRequest 对象，配置请求方法为 POST，目标 URL 为 获取余额.php，并将请求设置为异步。设置请求头，指定发送的数据类型为 URL 编码格式。监听请求状态变化，当请求完成且响应成功时，获取页面中具有 id 为 availableBalance 的元素，并设置其文本内容为返回的余额数据。发送请求数据，其中包含所选账户的账号信息。

当用户输入购买数量时，系统自动根据当前价格计算总价并实时显示。JavaScript 函数 calculateTotal 负责计算总价并更新显示。通过 JavaScript DOM 获取页面中具有 id 为 quantity 的输入框的值，即用户输入的购买数量。获取页面中具有 name 属性为 pricePerGram 的输入框的值，即黄金的单价。将购买数量与黄金单价相乘，计算总价，并存储在变量中。获取页面中具有 id 为 totalPrice 的元素，并设置其文本内容为计算得到的总价。使用 toFixed(2) 方法确保总价显示为两位小数。（如图4.16 黄金详情所示）

### 

### 图4.16 黄金详情

用户点击品牌价格链接后，系统带用户进入黄金购买页面，并传递选定品牌的相关信息，供用户进行购买操作。点击链接时，系统会通过 URL 参数传递品牌信息，并重定向到黄金购买页面。

* 黄金购买

在黄金购买界面，用户可以查看黄金的详细信息并进行购买操作。系统在页面加载时，通过SQL查询从数据库中获取黄金的详细信息，包括黄金品牌、当前价格和其他相关信息。查询结果存储在变量中，并显示在页面的相应位置。黄金品牌和当前价格显示在页面上，供用户参考。系统同时显示一个下拉菜单供用户选择支付卡，用户所有的银行账户会列在下拉菜单中。这些银行账户信息通过查询用户的银行账户表获取，并填充到下拉菜单中。系统从数据库中查询用户的银行卡信息。SQL查询语句为 SELECT 卡号, 余额 FROM 银行账户表 WHERE 用户ID = ?，用来获取用户绑定的银行卡信息。创建预处理语句对象，使用 prepare($sql) 函数。将用户ID绑定到SQL查询中的占位符上，调用 execute() 执行预处理语句，并使用 bind\_result 绑定查询结果到变量 $cardNumber, $balance。最后，使用 fetch() 获取结果并存储在变量中。

用户选择一个银行账户后，系统通过AJAX请求查询该账户的余额并实时显示。JavaScript函数 fetchBalance 负责发送请求并更新余额显示。通过JavaScript创建一个XMLHttpRequest对象，用于与服务器进行交互。配置请求，设置请求方法为POST，目标URL为获取余额.php，并将请求设置为异步。设置请求头，指定发送的数据类型为URL编码格式。监听请求状态变化，当请求完成且响应成功时，获取页面中具有id为availableBalance的元素，并设置其文本内容为返回的余额数据。发送请求数据，其中包含所选账户的账号信息。

用户输入想要购买的黄金数量，系统自动根据当前价格计算总价并实时显示。JavaScript函数calculateTotal负责计算总价并更新显示。通过JavaScript DOM获取页面中具有id为quantity的输入框的值，即用户输入的购买数量。获取页面中具有name属性为pricePerGram的输入框的值，即黄金的单价。将购买数量与黄金单价相乘，计算总价，并存储在变量中。获取页面中具有id为totalPrice的元素，并设置其文本内容为计算得到的总价。使用toFixed(2)方法确保总价显示为两位小数。（如图4.17 黄金购买界面所示）

### 

### 图4.17 黄金购买界面

用户填写完购买信息后点击“确认购买”按钮，系统接收到购买请求并进行处理。用户点击“确认购买”按钮后，系统会验证用户账户的余额，确保余额足够支付购买总价。如果余额不足，系统会提示用户余额不足，终止交易。这一步通过查询银行账户表获取用户的账户余额并进行比较实现。系统通过POST方法接收用户提交的购买信息，包括购买数量、选择的银行卡等。通过SQL查询用户选择的银行卡的当前余额。SQL查询语句为 SELECT 余额 FROM 银行账户表 WHERE 卡号 = ?。创建预处理语句对象，使用 prepare($sql) 函数。将银行卡号绑定到SQL查询中的占位符上，调用 execute() 执行预处理语句，并使用 bind\_result 绑定查询结果到变量 $balance。最后，使用 fetch() 获取结果并存储在变量中。将查询到的余额与购买总价进行比较。如果余额不足，系统会提示用户余额不足并终止交易。如果余额足够，系统会继续处理购买请求。更新用户银行卡的余额。SQL更新语句为 UPDATE 银行账户表 SET 余额 = 余额 - ? WHERE 卡号 = ?。创建预处理语句对象，使用 prepare($sql) 函数。将购买总价和银行卡号绑定到SQL查询中的占位符上，调用 execute() 执行预处理语句。将购买记录插入到用户黄金持有表中，更新用户持有的黄金数量和成交单价。SQL插入语句为 INSERT INTO 用户黄金持有表 (用户ID, 黄金品牌, 持有数量, 成交单价) VALUES (?, ?, ?, ?)。创建预处理语句对象，使用 prepare($sql) 函数。将用户ID、黄金品牌、购买数量和成交单价绑定到SQL查询中的占位符上，调用 execute() 执行预处理语句。记录此次交易信息，生成交易记录，包含转出卡号、转入卡号、交易金额和交易类型等，并将其插入到交易记录表中。SQL插入语句为 INSERT INTO 交易记录表 (转出卡号, 转入卡号, 交易金额, 交易类型) VALUES (?, ?, ?, ?)。创建预处理语句对象，使用 prepare($sql) 函数。将转出卡号、转入卡号、交易金额和交易类型绑定到SQL查询中的占位符上，调用 execute() 执行预处理语句，最后用alert设置购买成功的提示。（如图4.18 黄金购买成功提示）

### 

### 图4.18 黄金购买成功提示

* 黄金购买详情

在黄金购买详情界面，系统首先获取用户的购买信息，包括购买数量、总价和黄金品牌，这些信息通过 $\_GET 方法从URL参数中获取。系统通过 $\_GET['quantity'] 获取用户购买的黄金克数，通过 $\_GET['totalPrice'] 获取总价，通过 $\_GET['brand'] 获取黄金品牌。接着，系统启动会话并连接数据库，通过会话变量 $\_SESSION['user\_id'] 获取当前登录用户的ID，并通过SQL查询从数据库中获取用户的昵称和头像信息。查询语句为 SELECT 昵称, 头像 FROM 用户信息 WHERE 用户ID = ?。创建预处理语句对象，使用 prepare 函数，将用户ID绑定到SQL查询中的占位符上，调用 execute 执行预处理语句，并使用 get\_result 获取查询结果。将结果存储在变量 $user 中，并从中提取昵称和头像编号。

系统根据用户的头像编号，选择对应的预设头像或用户上传的头像。如果头像编号在预设范围内，则通过 switch 语句选择对应的头像路径和描述；否则，使用用户上传的头像，并在路径中添加时间戳以防止缓存问题。预设头像的路径在预设目录中，用户上传的头像在用户头像目录中。具体路径和描述通过 switch 语句进行设置。

在HTML页面中，系统通过HTML和PHP相结合的方式显示用户的购买详情信息，包括用户头像、昵称、购买数量、购买成功信息、创建时间、黄金品牌和总价。用户头像通过 <img> 标签显示，路径和描述分别通过 htmlspecialchars($avatarPath) 和 htmlspecialchars($avatarAlt) 设置。其他信息通过 htmlspecialchars 函数进行转义后显示在相应的HTML标签中。（如图4.19 黄金购买成功所示）

### 

### 图4.19 黄金购买成功

# 总结与体会

## 小组成员分工情况

21083113罗宇孙：主界面，用户信息，总资产查询功能

21083114杨宸铭：登陆，注册，重置密码，黄金，股票功能

21083115刘小杨：银行转账功能，

## 收获

通过这次管理信息系统课程设计，我收获颇丰。

首先，我深入了解了如何设计规范化的数据库表结构。通过减少数据冗余和提高数据一致性，保证了数据库的高效运行。同时，通过范式分析，我确保了数据库表结构符合三范式，显著提升了数据库的性能。在实际操作中，我掌握了使用PHP连接MySQL数据库，并能熟练进行增删改查等操作。这让我深刻理解了数据库操作的基本原理和方法。此外，我还学会了使用预处理语句和绑定参数来提高查询的安全性，有效防止了SQL注入攻击。

用户会话管理是系统设计中的重要一环，通过使用PHP的$\_SESSION全局变量，我实现了用户的登录、注销和权限控制，并能够根据不同用户角色显示相应的功能界面和操作选项。在前端开发方面，我通过使用HTML表单与PHP脚本的交互，实现了用户数据的提交和处理，并能在前端动态显示数据库中的数据。

在整个项目中，我实现了管理信息系统的各个功能模块，如用户登录、跨行转账、黄金管理和股票管理等。在用户登录模块中，我实现了用户注册、登录验证和密码重置功能，确保了用户信息的安全性和完整性。跨行转账模块中，用户可以方便地进行资金转账，系统会实时更新账户余额并记录交易信息。在黄金管理模块中，系统可以动态生成黄金价格，并支持用户查询和购买黄金。在股票管理模块中，系统能够生成和更新股票价格，并提供详细的股票信息和购买功能。

通过这个项目，我不仅了解了各功能模块的逻辑关系和数据交互流程，还掌握了如何将这些模块有效地集成到系统中。同时，我也深刻认识到管理信息系统在实际应用中的重要性和广泛应用前景。这次课程设计的实践经验，为我今后的学习和工作奠定了坚实的基础。

此外，通过这次课程设计，我还学会了很多实用的技术和工具，包括JavaScript代码、Markdown、PlantUML、Mermaid、JSON、Word、Visio等软件的使用。这些工具的掌握，进一步提升了我的项目开发能力和效率。

总体而言，这次管理信息系统课程设计不仅让我在技术上有所提升，更让我对系统设计和开发有了更深入的理解和实践经验。

## 改进设想

在此次管理信息系统和数据库课程设计中，我们提出了一系列改进设想，以进一步完善系统功能并提升用户体验。

首先，我们计划在管理员身份登录的用户信息管理界面中增加多项新功能。这些改进不仅将提升管理的便利性，还将确保系统的安全性和数据的完整性。我们设想增加上证指数和沪深指数的计算功能，以使股票模块的功能更加全面和专业。这些指数的引入有助于用户更直观地了解市场动态，并为其投资决策提供有力支持。

此外，我们计划为黄金模块增加历史价格的折线图功能。通过动态绘制黄金价格的历史变动图表，用户将能够更加直观地观察价格走势，进行趋势分析和投资决策。这一功能不仅将提升用户的使用体验，还将充分展示数据库数据的可视化应用。

在基金管理和交易记录查询方面，我们设想根据数据库的信息相应增加相关功能。通过系统化地整合和展示基金信息及用户交易记录，用户将能够更加方便地管理和查询其投资情况。这些改进将使系统功能更加完善，用户能够全面掌握其投资活动和资金动向。

为了进一步增强系统的安全性，我们计划在用户登录模块中添加验证码功能。通过增加这一安全措施，可以有效防止恶意登录和攻击，确保用户账户信息的安全性。

综上所述，通过这些改进设想，我们不仅希望提升系统的实用性和安全性，还希望更加充分地利用数据库中的数据资源。这些设想和优化措施，将使得管理信息系统在实际应用中更加高效、可靠，为用户提供更加优质的服务体验。这些改进设想为我们在数据库和管理信息系统领域的进一步研究和应用奠定了坚实的基础。