**作业报告**

目录

[**需求分析** 1](#_Toc156312234)

[**问题分析** 2](#_Toc156312235)

[**设计** 2](#_Toc156312236)

[**模块划分设计** 2](#_Toc156312237)

[**用户表设计** 3](#_Toc156312238)

[**商品表设计** 3](#_Toc156312239)

[**购物车表设计** 4](#_Toc156312240)

[**消息表设计** 5](#_Toc156312241)

[**实现** 5](#_Toc156312242)

[**用户模块实现** 5](#_Toc156312243)

[**商城首页模块实现** 6](#_Toc156312244)

[**分类页模块实现** 7](#_Toc156312245)

[**购物车模块实现** 7](#_Toc156312246)

[**聊天页模块实现** 9](#_Toc156312247)

[**测试** 9](#_Toc156312248)

[**用户模块测试** 9](#_Toc156312249)

[**商城首页模块测试** 17](#_Toc156312250)

[**分类页模块测试** 20](#_Toc156312251)

[**购物车页模块测试** 23](#_Toc156312252)

[**聊天页测试** 27](#_Toc156312253)

[**源代码git** 32](#_Toc156312254)

**需求分析**

1. **用户模块**：系统需要支持用户注册、登录和退出功能。用户注册时需要提供必要的信息，如用户名和密码。用户登录时需要验证用户名和密码。用户退出时需要结束当前的会话。
2. **信息管理**：系统需要支持对用户数据模型和购物车数据模型的增删改查操作。用户数据模型包含用户的信息，如用户名、密码、头像等。购物车数据模型包含用户添加到购物车的商品信息，如商品ID、数量等。每个用户可以有多个购物车数据模型，实现一对多的关系。
3. **图片上传**：系统需要支持用户上传图片来设置自己的头像。用户可以从本地选择一张图片，然后上传到服务器。服务器需要保存用户上传的图片，并将图片的URL保存到用户数据模型中。
4. **站内消息**：系统需要支持用户之间的消息通信。用户可以发送消息给其他用户。系统需要保存用户发送的消息，并在接收者登录时显示新的消息。
5. **前端异步请求**：系统需要支持前端异步请求。在个人中心页面，用户可以异步地对自己的个人信息进行增删改查操作。这样，用户在更新信息时，页面不需要完全刷新，提高了用户体验。

**问题分析**

1. **用户模块问题：**

* **注册和登录验证：** 需要确保注册时提供的信息有效，并实现密码验证机制。
* **会话管理：** 需要确保用户在登录后的会话状态有效，并能正确退出。

1. **信息管理问题：**

* **数据模型设计：** 用户和购物车数据模型的设计需要满足一对多关系，并考虑到数据的一致性和完整性。
* **增删改查操作：** 系统需要提供有效的接口和逻辑来实现对用户和购物车数据的增删改查。

1. **图片上传问题：**

* **文件上传：** 确保文件上传的安全性和有效性，同时服务器端需要有相应的处理逻辑。
* **图片URL存储：** 确保成功上传的图片的URL正确保存到用户数据模型中。

1. **站内消息问题：**

* **消息通信：** 需要实现用户之间的消息传递系统，并保证消息的安全性和及时性。
* **消息保存：** 确保用户发送的消息能够被正确保存，并在需要时能够被检索。

1. **前端异步请求问题：**

* **异步请求处理：** 确保前端能够正确地发起异步请求，后端能够正确地处理这些请求，并在必要时返回合适的数据。

**设计**

**模块划分设计**

该项目可以分为以下几个模块：

1. **用户模块**：包含用户注册、登录、个人信息查看和编辑等功能。
2. **商城首页模块**：包含商品搜索、商品展示、添加至购物车等功能。
3. **分类页模块**：包含商品展示、商品分类展示、添加至购物车等功能。
4. **购物车页模块**：包含购物车列表展示、购物车操作等功能。
5. **聊天页模块**：包含接收消息列表展示、发送消息列表展示、发送消息、对消息进行操作等功能。

**用户表设计**

|  |
| --- |
| Java @Data public class User {  @TableId(type= IdType.*AUTO*)  private Long id; *// 用户ID，自增主键* private String username; *// 用户名* private String password; *// 用户密码* private String number; *// 用户手机号* private String email; *// 用户邮箱* private String avatar; *// 用户头像，存储图片的URL* } |

用户表设计包括以下字段：

1. **id（Long）**：用户ID，为自增主键，用于唯一标识每个用户。
2. **username（String）**：用户名，用于用户登录和显示。应该具备唯一性，以确保每个用户名对应一个用户。
3. **password（String）**：用户密码，用于登录验证。
4. **number（String）**：用户手机号，用于联系和身份验证。需要满足一定的手机号格式。
5. **email（String）**：用户邮箱，用于联系和接收系统通知。需要满足一定的邮箱格式。
6. **avatar（String）**：用户头像，存储头像图片的URL。提供了可视化的用户信息展示，增加用户体验。

**商品表设计**

|  |
| --- |
| Java public class Product {  @TableId(type= IdType.*AUTO*)  private Long id; *// 商品ID，自增主键* private String name; *// 商品名字* private double price; *// 商品价格* private String type; *// 商品种类* private String image; *// 商品图片* } |

该商品表设计包括以下字段：

1. **id（Long）**：商品ID，为自增主键，用于唯一标识每个商品。
2. **name（String）**：商品名称，用于标识商品的名字。
3. **price（double）**：商品价格，表示商品的售价。通常采用浮点型数据存储，以支持小数点后的金额。
4. **type（String）**：商品种类，表示商品所属的类别或类型。可以用于在系统中进行商品分类。
5. **image（String）**：商品图片，存储商品的图片URL。提供了可视化的商品展示，增加购物体验。

**购物车表设计**

|  |
| --- |
| Java public class Cart {  @TableId(type= IdType.*AUTO*)  private Long id; *// 商品ID，自增主键* private String name; *// 商品名字* private double price; *// 商品价格* private String type; *// 商品种类* private String image; *// 商品图片* private int count; *// 商品数量* private String username; *// 用户名* } |

购物车表设计包括以下字段：

1. **id（Long）**：购物车项ID，为自增主键，用于唯一标识每个购物车项。
2. **name（String）**：商品名称，表示购物车中商品的名字。
3. **price（double）**：商品价格，表示购物车中商品的售价。通常采用浮点型数据存储，以支持小数点后的金额。
4. **type（String）**：商品种类，表示购物车中商品所属的类别或类型。可以用于在购物车中展示商品分类。
5. **image（String）**：商品图片，存储购物车中商品的图片URL。提供了可视化的购物车展示，增加用户体验。
6. **count（int）**：商品数量，表示购物车中该商品的数量。用于记录用户选择购买的商品数量。
7. **username（String）**：用户名，表示购物车所属的用户。用于关联购物车项与用户。

**消息表设计**

|  |
| --- |
| Java public class Message {  @TableId(type= IdType.*AUTO*)  private Long id; *// 消息ID，自增主键* private String fromUsername; *// 发送者用户名* private String toUsername; *// 接收者用户名* private String content; *// 消息详情* private String state; *// 消息状态* } |

消息表设计包括以下字段：

1. **id（Long）**：消息ID，为自增主键，用于唯一标识每条消息。
2. **fromUsername（String**）：发送者用户名，表示消息的发送者。
3. **toUsername（String）**：接收者用户名，表示消息的接收者。
4. **content（String）**：消息详情，存储消息的具体内容。
5. **state（String）**：消息状态，表示消息的状态，例如已读、未读等。可以使用字符串表示不同的状态。

**实现**

**用户模块实现**

用户模块的实现包含了前端和后端的交互，实现了用户注册、登录、个人信息查看和编辑等功能。以下是对该模块实现的简要阐述：

1. **前端实现：**

* 使用了Bootstrap框架构建页面，提供了清晰的导航栏和个人中心布局。
* 通过Alpine.js管理前端逻辑，实现了模态框的展示和隐藏，以及与用户输入的双向数据绑定。
* 提供了用户注册、登录、退出登录、查看个人信息和编辑个人信息等操作的按钮，并展示用户的基本信息。

1. **后端实现：**

* 使用Spring Boot框架创建了一个ProfileController，处理与用户相关的HTTP请求。
* 利用UserMapper进行用户数据的持久化操作，包括注册、登录、信息加载、更新等。
* 使用HttpSession管理用户会话，实现登录状态的维护和退出登录功能。

1. **安全性和验证：**

* 注册时检查用户名是否已存在，防止重复注册。
* 登录时验证用户名和密码，提高系统安全性。
* 在编辑信息前进行验证，确保手机号和邮箱格式符合要求。

1. **用户交互：**

* 提供了友好的用户交互，通过模态框提供注册、登录、编辑等功能，提高了用户体验。
* 使用Axios进行异步HTTP请求，实现前后端的数据传输。

1. **用户会话管理：**

* 使用HttpSession管理用户会话，确保用户在登录状态下进行操作，防止未授权访问。

1. **界面展示：**

* 通过表格展示用户基本信息，包括用户ID、用户名、手机号和邮箱。
* 允许用户上传头像，并在界面上显示。

**商城首页模块实现**

商城首页模块采用了Spring Boot作为后端框架，通过RestController处理前端请求，与数据库进行交互。前端使用了Bootstrap和Alpine.js构建响应式页面，提供了良好的用户交互体验。

**后端实现：**

1. **Controller层：** IndexController类负责处理关于商品的请求，包括加载所有商品和根据关键字搜索商品。通过ProductMapper与数据库进行交互，返回相应的商品数据。

**前端实现：**

1. **页面结构：** 使用Bootstrap构建导航栏，搜索框和商品列表，Alpine.js处理页面交互逻辑。
2. **商品搜索：** 用户可以输入商品名或种类进行搜索。通过Alpine.js的数据绑定，搜索关键字将实时更新，点击搜索按钮将触发相关请求。
3. **商品列表：** 商品列表通过循环渲染展示，包括商品的图片、名字、种类、价格和添加到购物车的按钮。每个商品有一个详情模态框。
4. **模态框：** 点击商品的添加按钮会弹出一个模态框，展示商品的详细信息，包括图片、名字、种类和价格。用户可以选择将商品添加到购物车。

**分类页模块实现**

分类页模块使用Spring Boot作为后端框架，通过RestController处理前端请求，并与数据库进行交互。前端使用了Bootstrap和Alpine.js构建响应式页面，提供了不同分类的商品展示和添加到购物车的功能。

**后端实现：**

1. **Controller层：** CategoryController类负责处理不同分类的商品请求，包括加载所有商品、数码商品、电脑商品、家电商品、服饰商品和美妆商品。通过ProductMapper与数据库进行交互，返回相应的商品数据。

**前端实现：**

1. **页面结构：** 使用Bootstrap构建导航栏、分类按钮和商品列表，Alpine.js处理页面交互逻辑。
2. **分类按钮：** 用户可以点击不同分类的按钮，触发相关请求，从而加载对应分类的商品列表。
3. **商品列表：** 商品列表通过循环渲染展示，包括商品的图片、名字、种类、价格和添加到购物车的按钮。每个商品有一个详情模态框。
4. **模态框：** 点击商品的添加按钮会弹出一个模态框，展示商品的详细信息，用户可以选择将商品添加到购物车。

**购物车模块实现**

购物车模块采用了前后端分离架构，通过Spring Boot提供的RESTful API实现了购物车相关的功能。以下是详细实现说明：

**后端实现：**

1. **添加商品 (@PostMapping)：**

* 接收前端传来的商品信息，检查用户是否登录。
* 判断购物车中是否已有相同商品，若有则增加数量，若无则创建新的购物车项。
* 使用MyBatis Plus进行数据库操作。

1. **获取购物车 (@GetMapping)：**

* 检查用户是否登录，根据用户查询购物车中的商品列表。
* 返回购物车商品信息给前端。

1. **增加数量 (@PutMapping("/plus"))：**

* 接收前端传来的购物车项信息，增加商品数量。
* 使用MyBatis Plus进行数据库更新。

1. **减少数量 (@PutMapping("/minus"))：**

* 接收前端传来的购物车项信息，减少商品数量。
* 若数量减至0，则从购物车中移除该商品。
* 使用MyBatis Plus进行数据库更新或删除。

1. **清空购物车 (@DeleteMapping("/clear"))：**

* 清空用户购物车中的所有商品。
* 使用MyBatis Plus进行数据库删除。

**前端实现：**

1. **展示购物车列表：**

* 使用Bootstrap创建表格，显示商品的图片、名称、种类、价格、数量、总价等信息。
* 使用Alpine.js实现响应式交互，包括加一、减一、清空购物车和结算等功能。

1. **购物车操作：**

* 提供按钮实现商品数量的增减。
* 提供清空购物车和结算按钮，结算时计算总花费并清空购物车。

1. **实时更新 (x-data 和 x-init)：**

* 使用Alpine.js实时更新购物车信息，包括商品数量、总价等。

**聊天页模块实现**

聊天页模块使用Spring Boot后端和Alpine.js前端实现了基本的消息收发功能。具体实现说明如下：

1. **后端 (ChatController 类)：**

* 提供接口获取接收消息列表、发送消息列表、发送消息、标记消息已读、删除消息等功能。
* 使用MyBatis Plus进行数据库操作，通过HTTP请求实现前后端数据传递。

1. **前端 (HTML、Bootstrap、Alpine.js)：**

* 通过Bootstrap创建页面结构，包括导航栏和两个消息列表。
* 使用Alpine.js实现数据绑定和交互逻辑。
* 提供按钮实现写消息、刷新、已读、删除等功能。

1. **接收消息列表和发送消息列表：**

* 列表展示接收的消息和发送的消息，包括发送者、接收者、消息内容、消息状态等信息。
* 使用Alpine.js实时更新消息列表。

1. **写消息功能 (<div x-show="show\_modal">)：**

* 使用Bootstrap模态框展示消息编辑界面，包括接收者用户名和消息内容。
* 提供发送按钮，点击后通过Alpine.js调用后端接口发送消息。

1. **实时刷新和交互 (x-data 和 x-init)：**

* 使用Alpine.js的x-data和x-init实现数据的初始化和实时刷新。
* 提供刷新按钮，点击后调用后端接口刷新消息列表。

**测试**

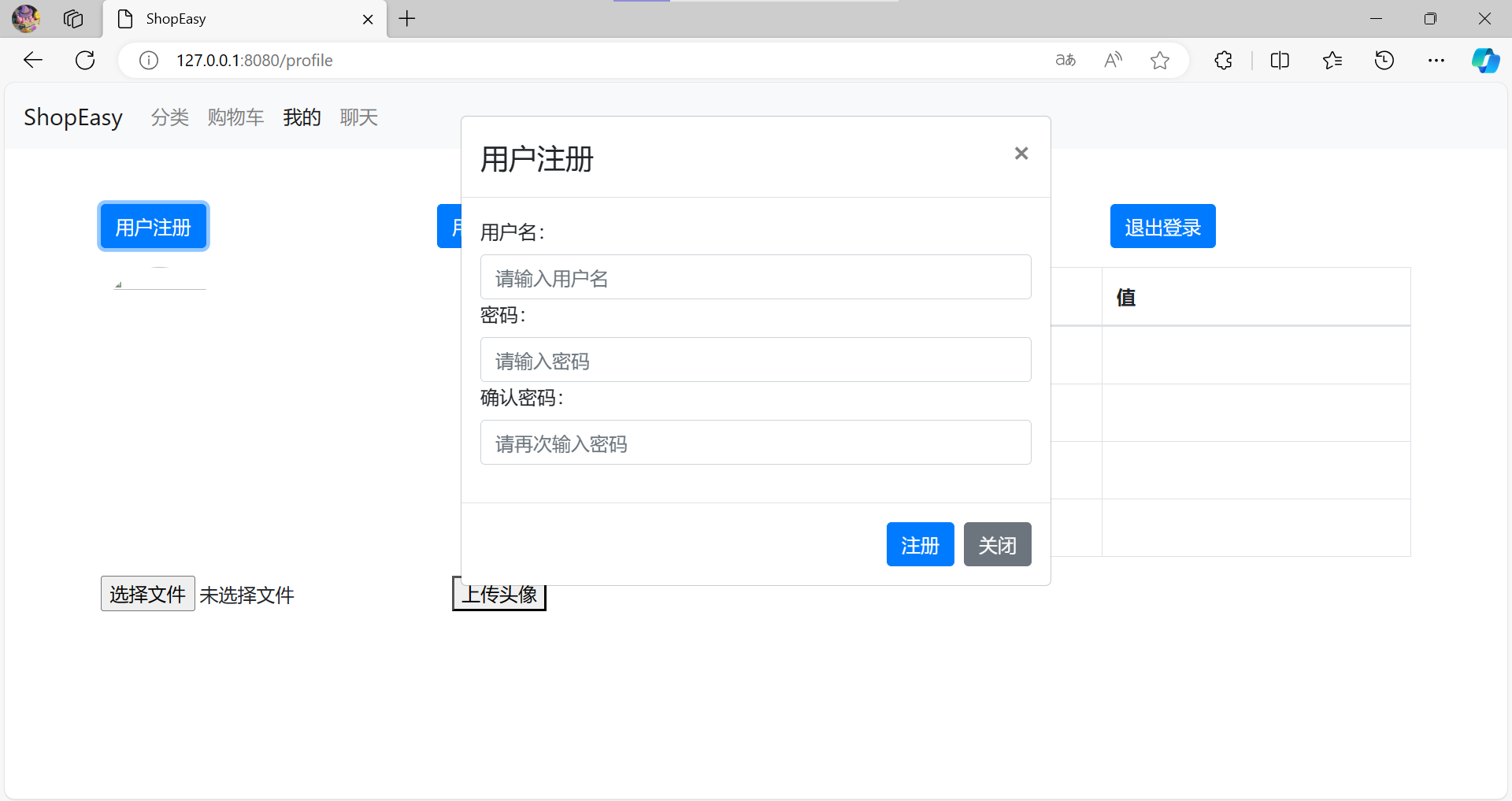
**用户模块测试**

1. 用户模块加载测试：点击顶部导航栏“我的”按键导航到用户模块。

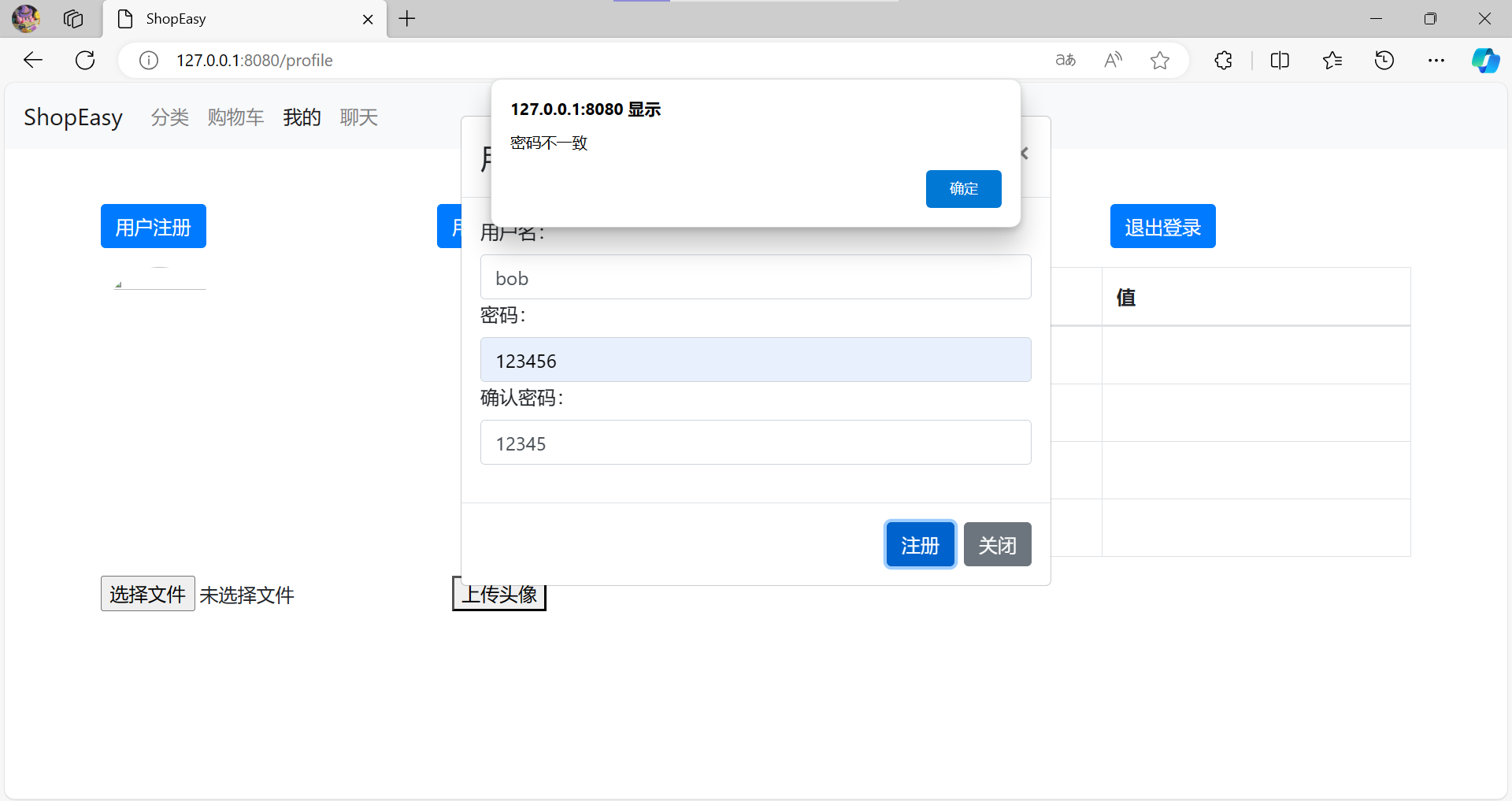


1. 用户注册测试：

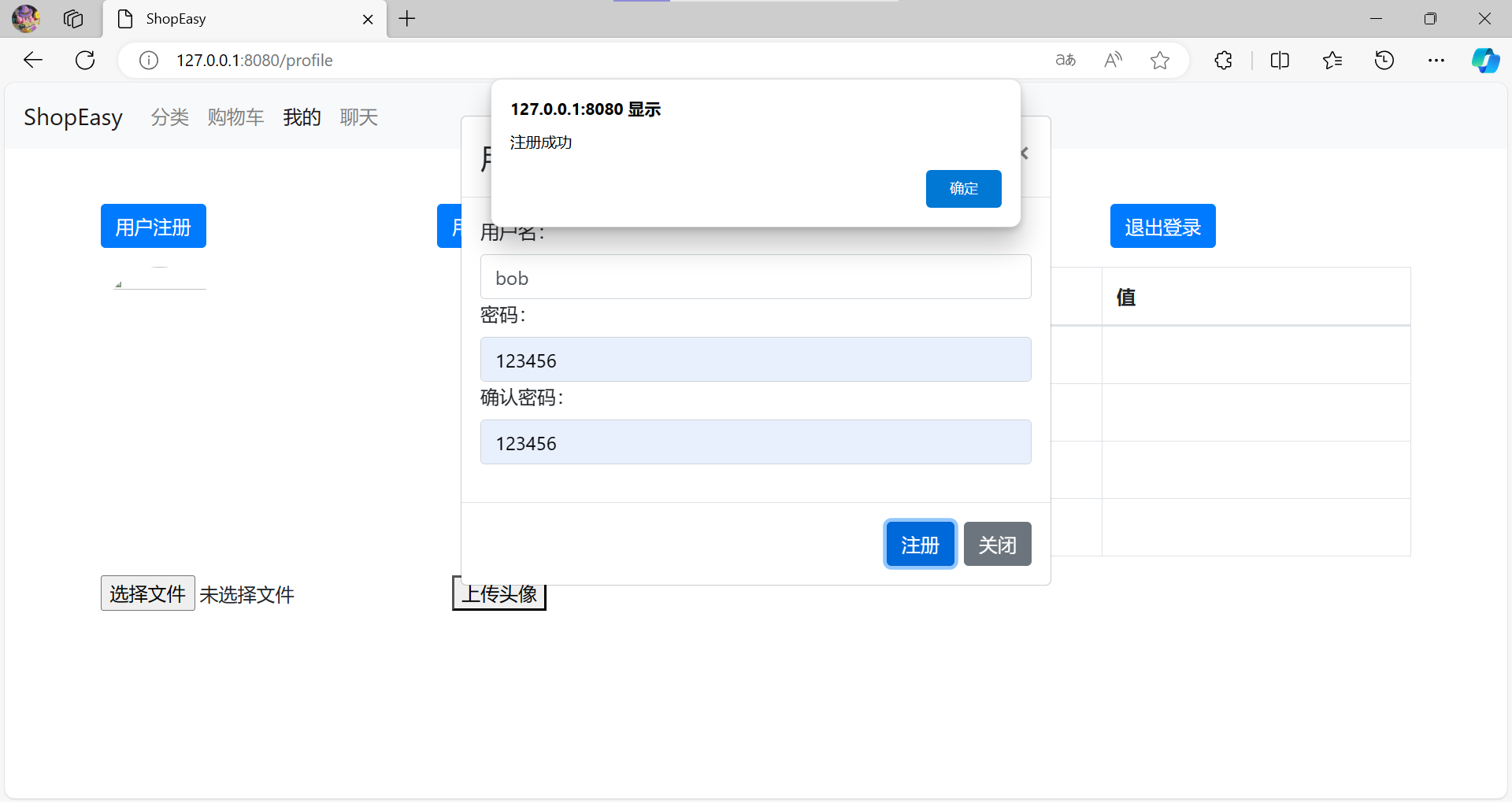
点击按键“用户注册”进行注册：



要求两次输入的密码一致：

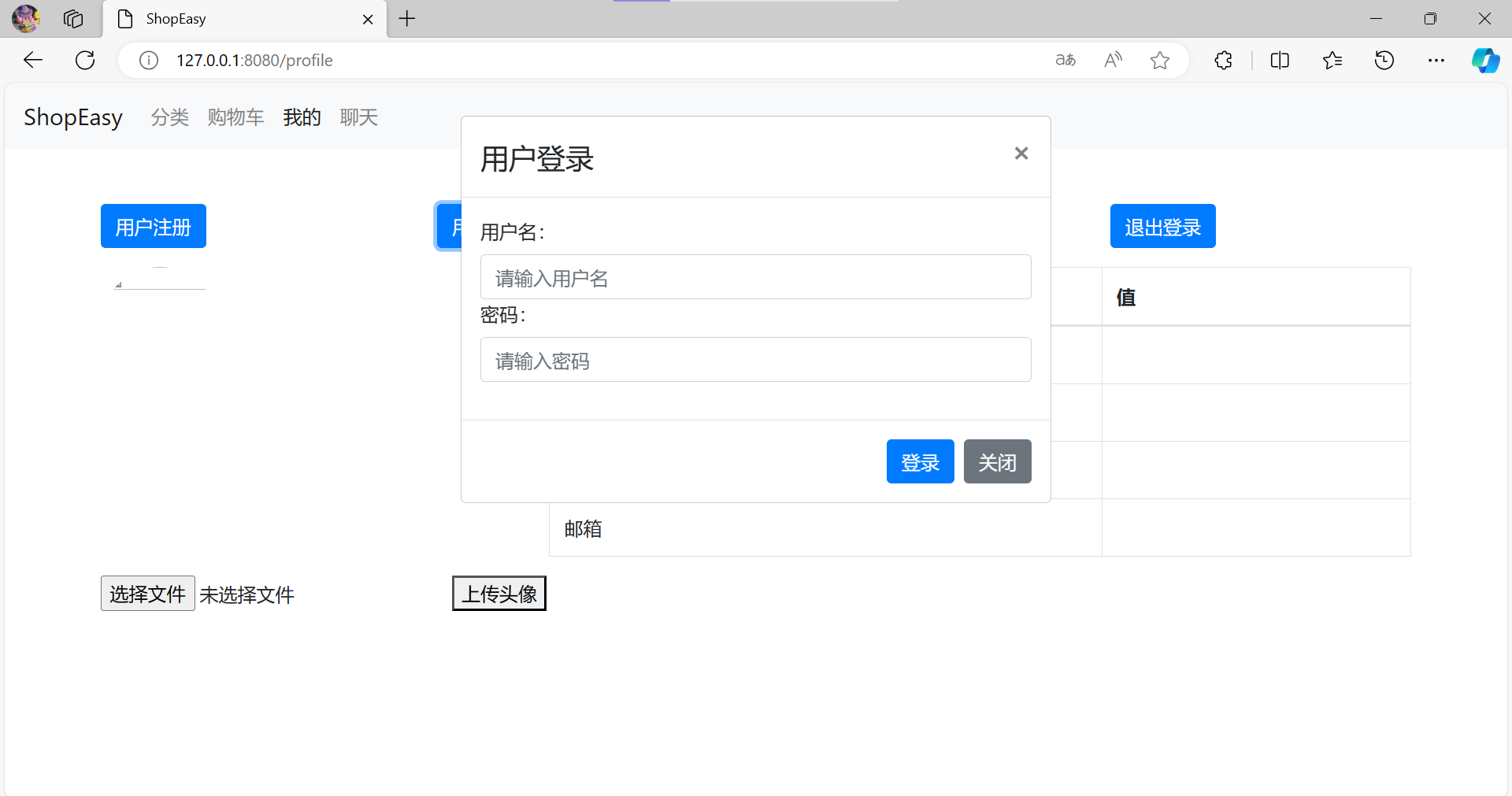


注册成功提示：

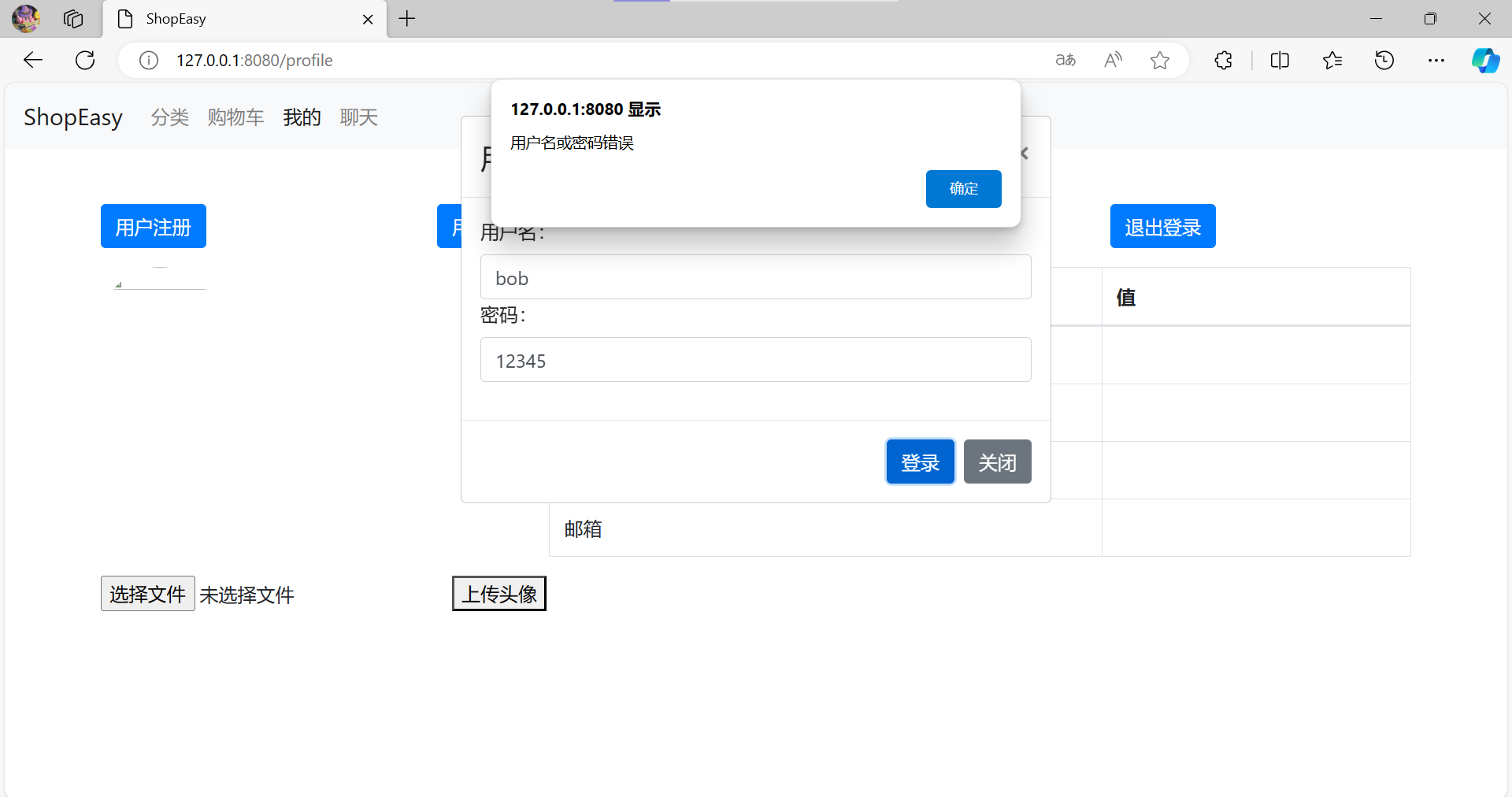


1. 用户登录测试：

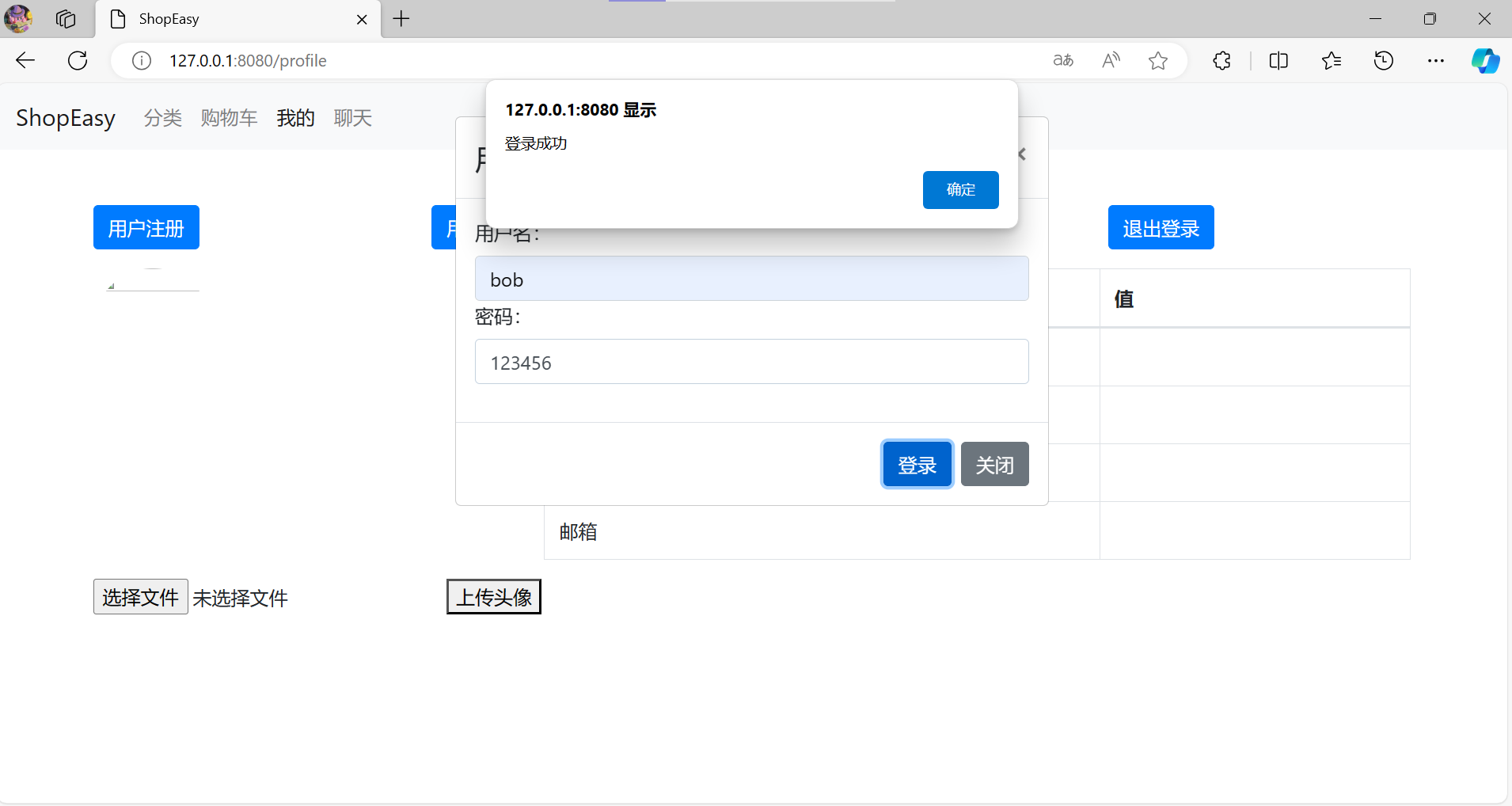
点击登录按键进行登录：



登录失败提示：



登录成功提示：

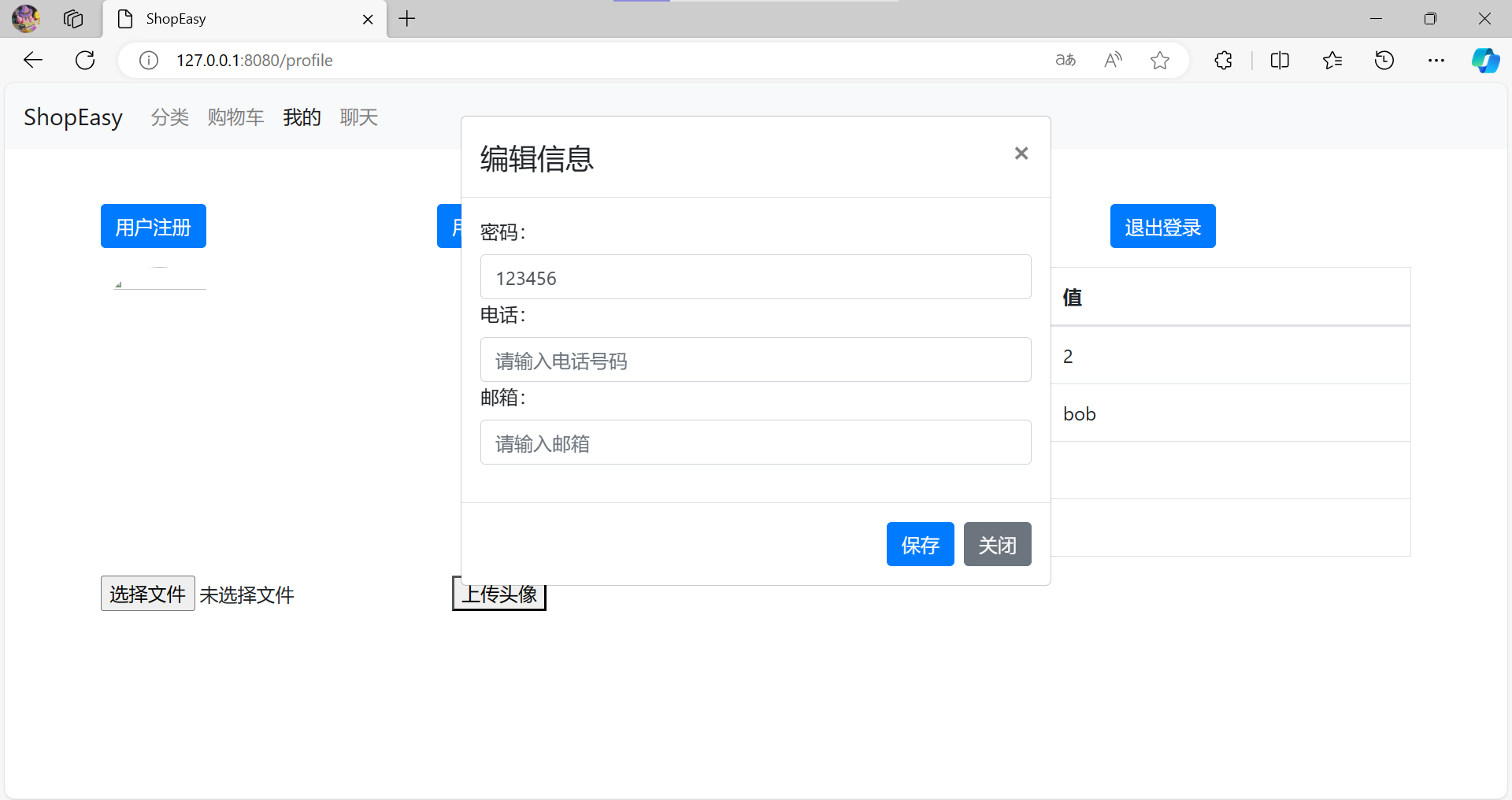


登录成功后个人页显示登录信息：

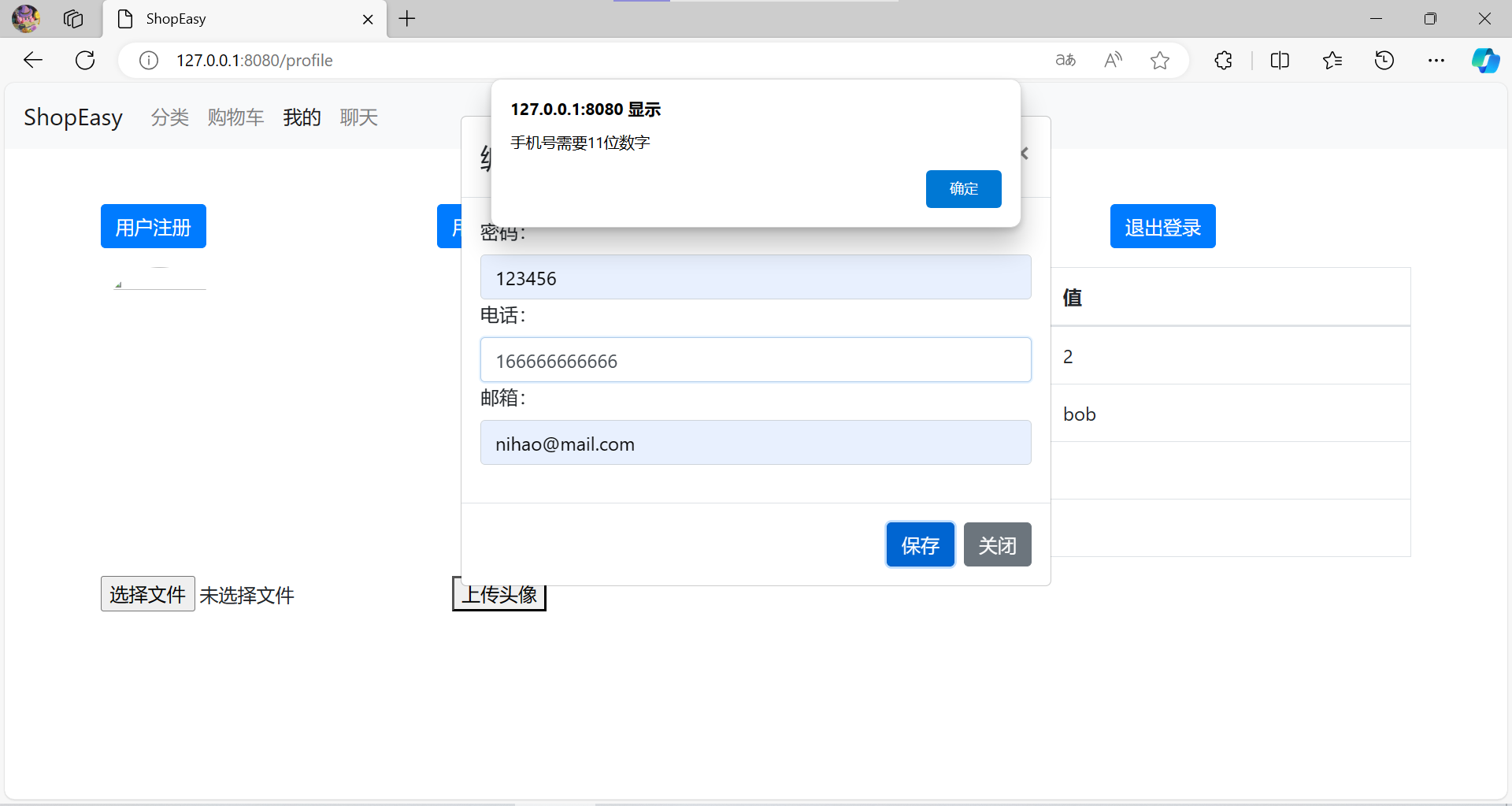


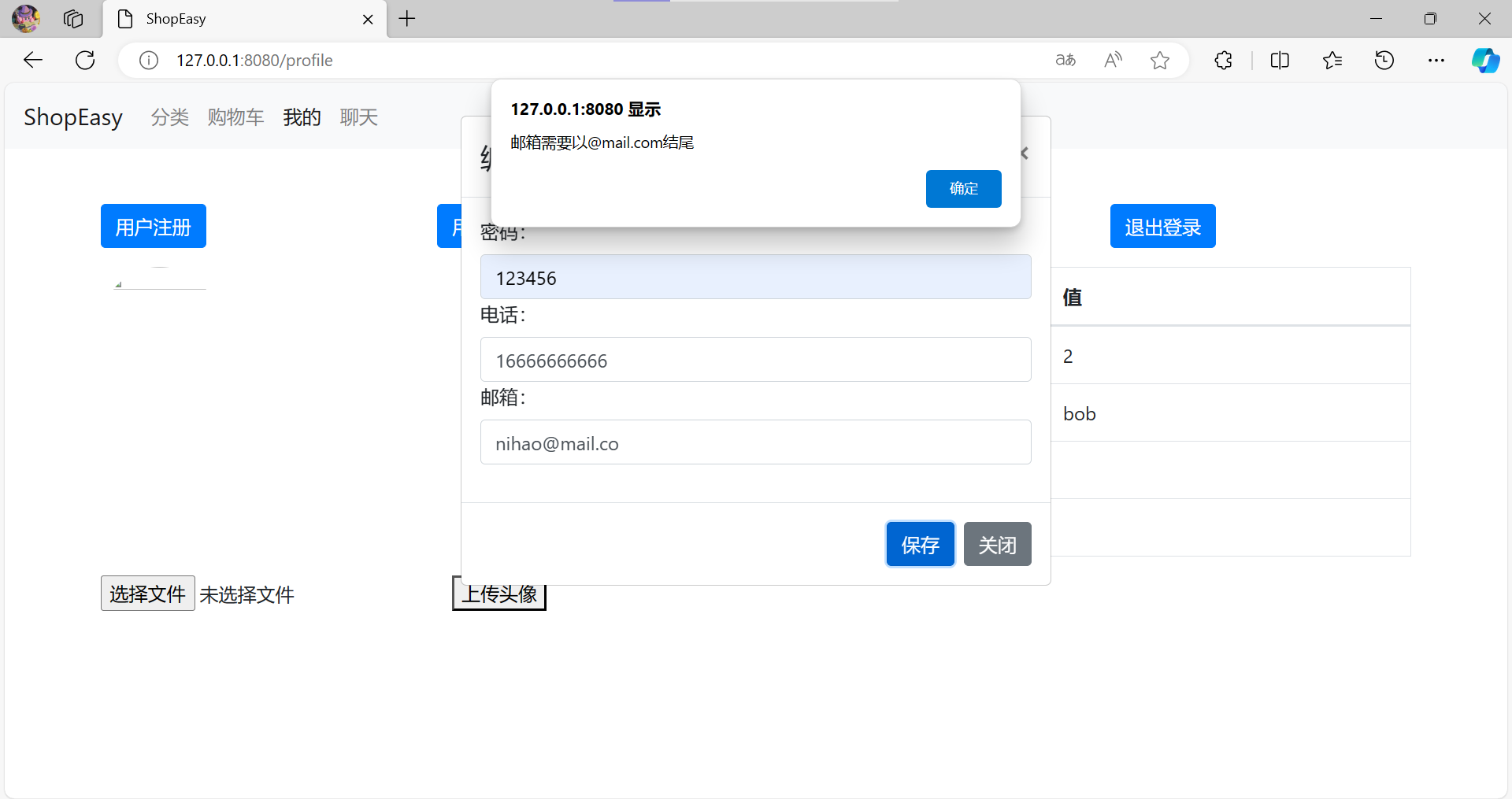
1. 个人信息编辑测试：

点击编辑信息对当前登录用户信息进行编辑：

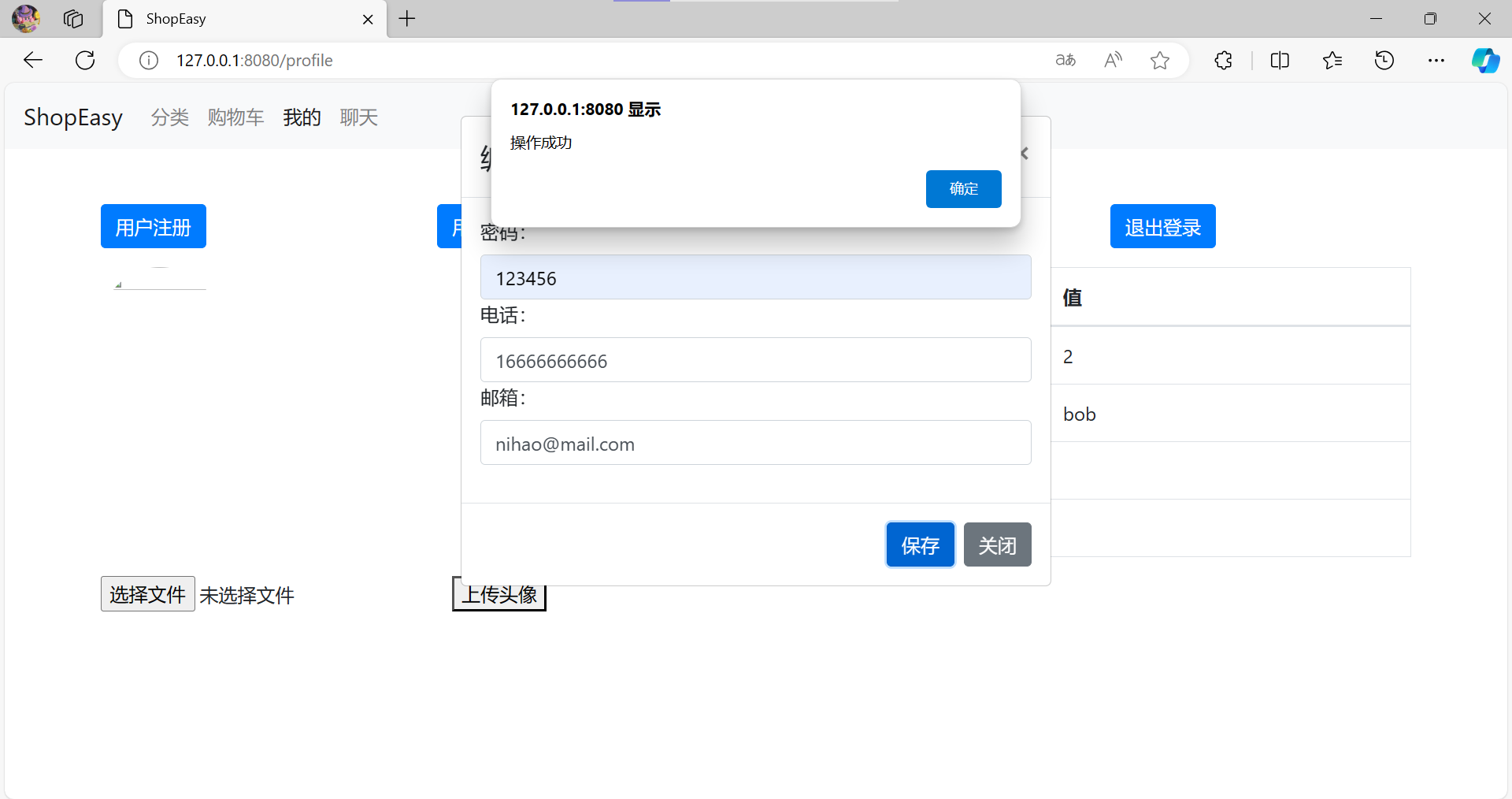


输入规范检测：





编辑成功测试：





编辑头像测试：点击选择文件选择文件上传，再点击上传头像设置用户头像。





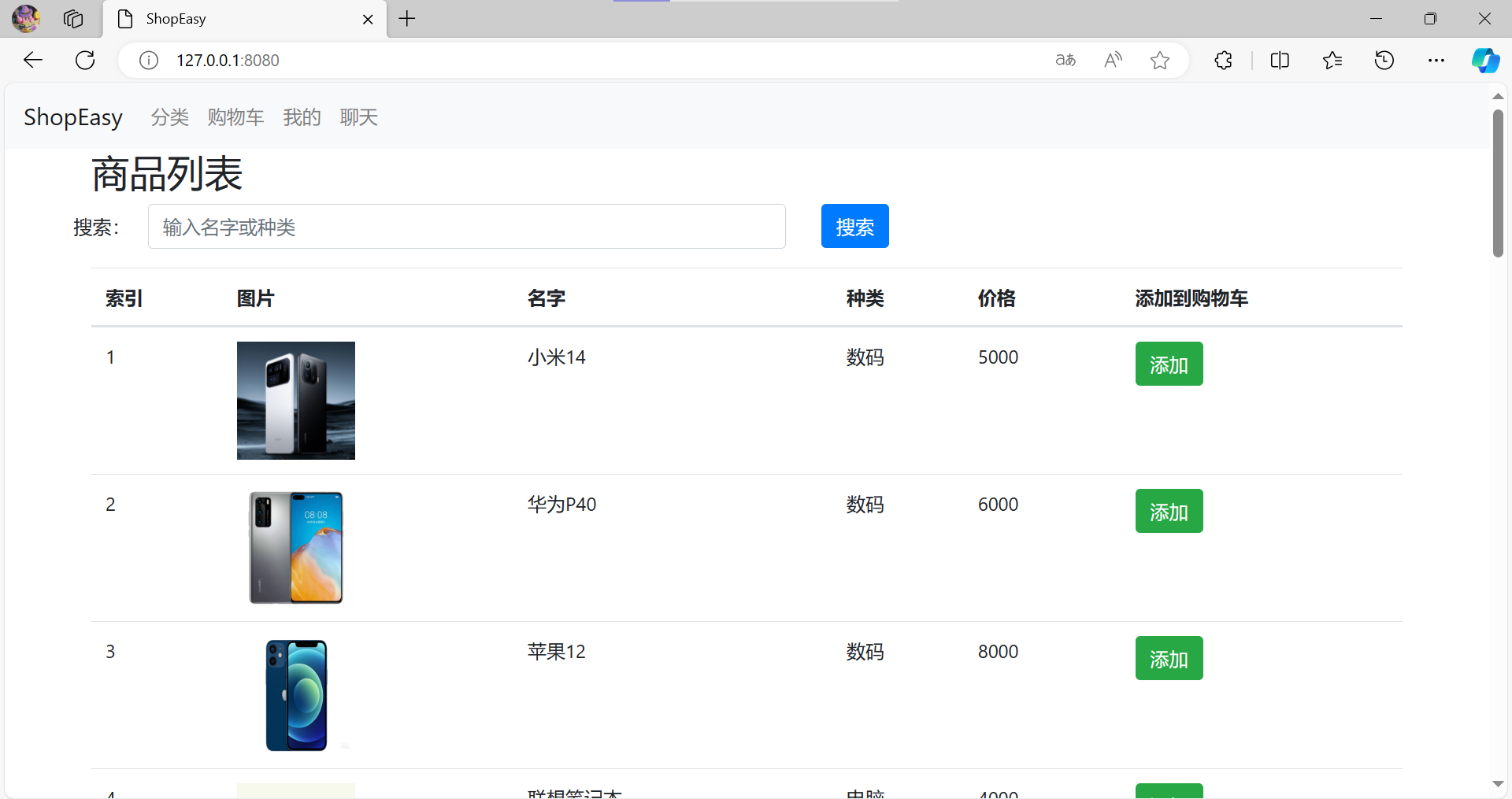
1. 退出登录测试：

点击退出登录按键退出登录：



**商城首页模块测试**

1. 商城首页模块加载测试：

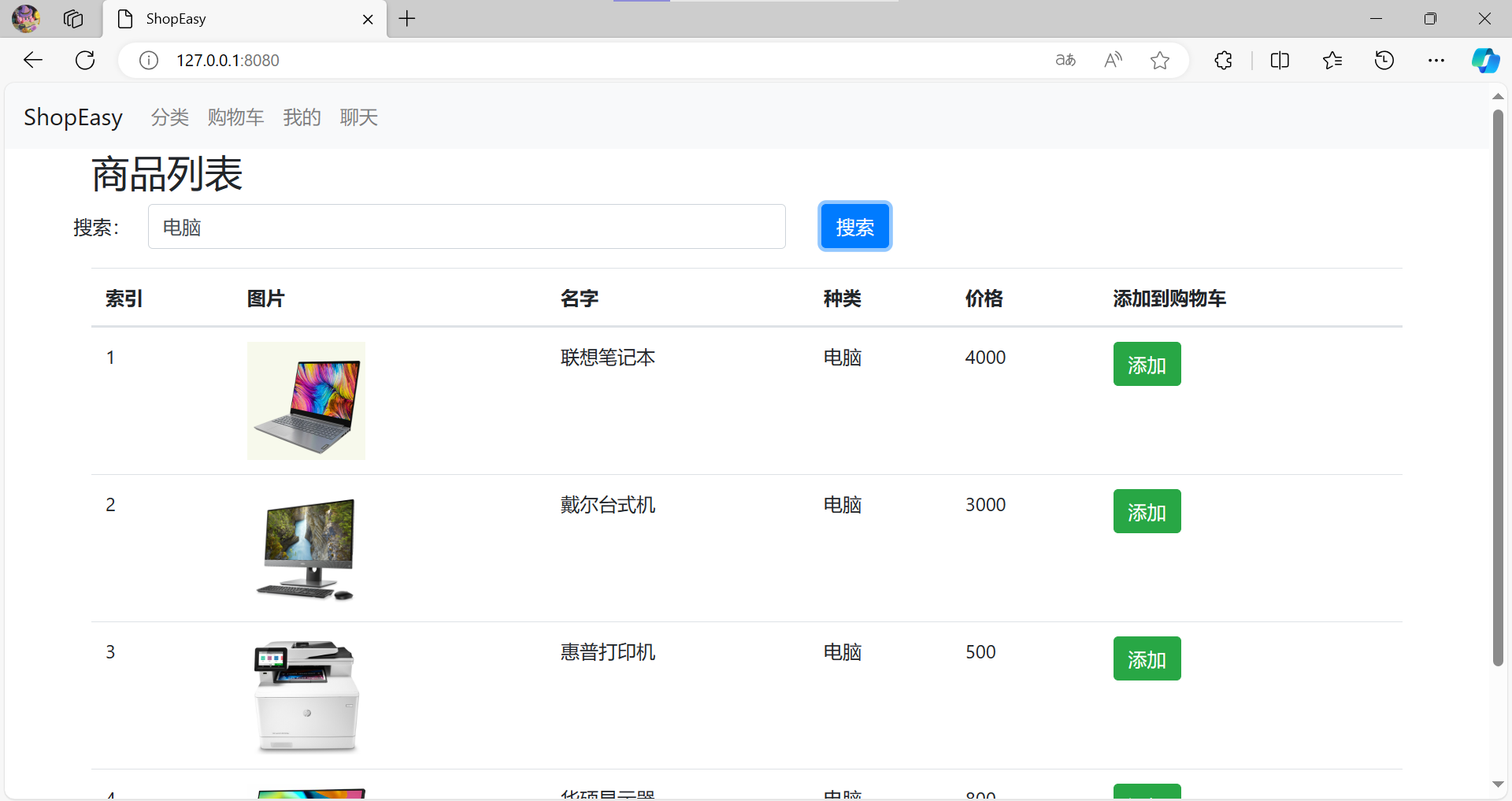


1. 商城首页搜索测试：

根据名字搜索：



根据种类搜索：

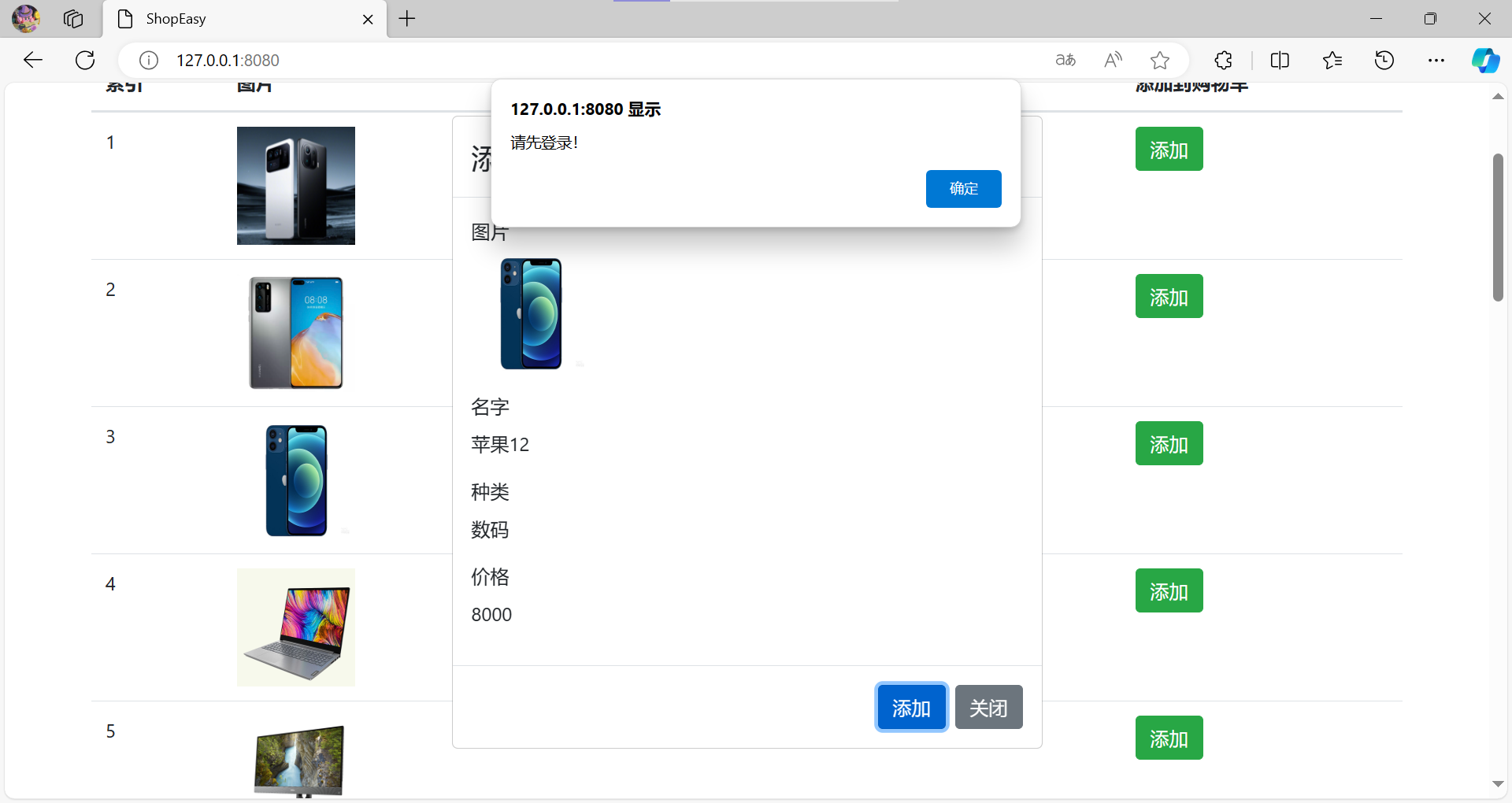


1. 添加到购物车测试：

点击添加对对应商品进行浏览：



再点击添加测试添加到购物车功能：



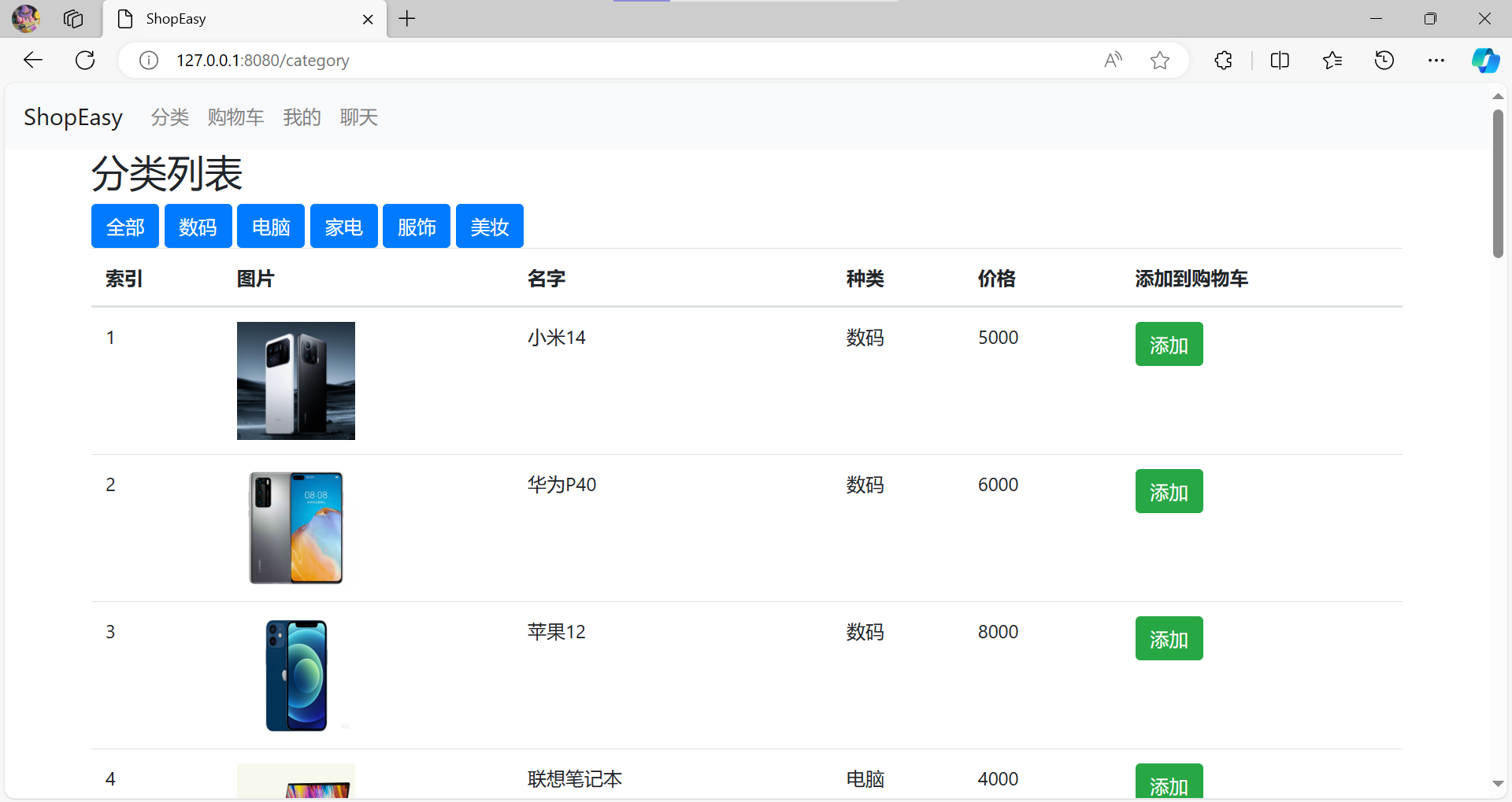
需登录才能添加到购物车，登陆后点击添加，提示添加成功：



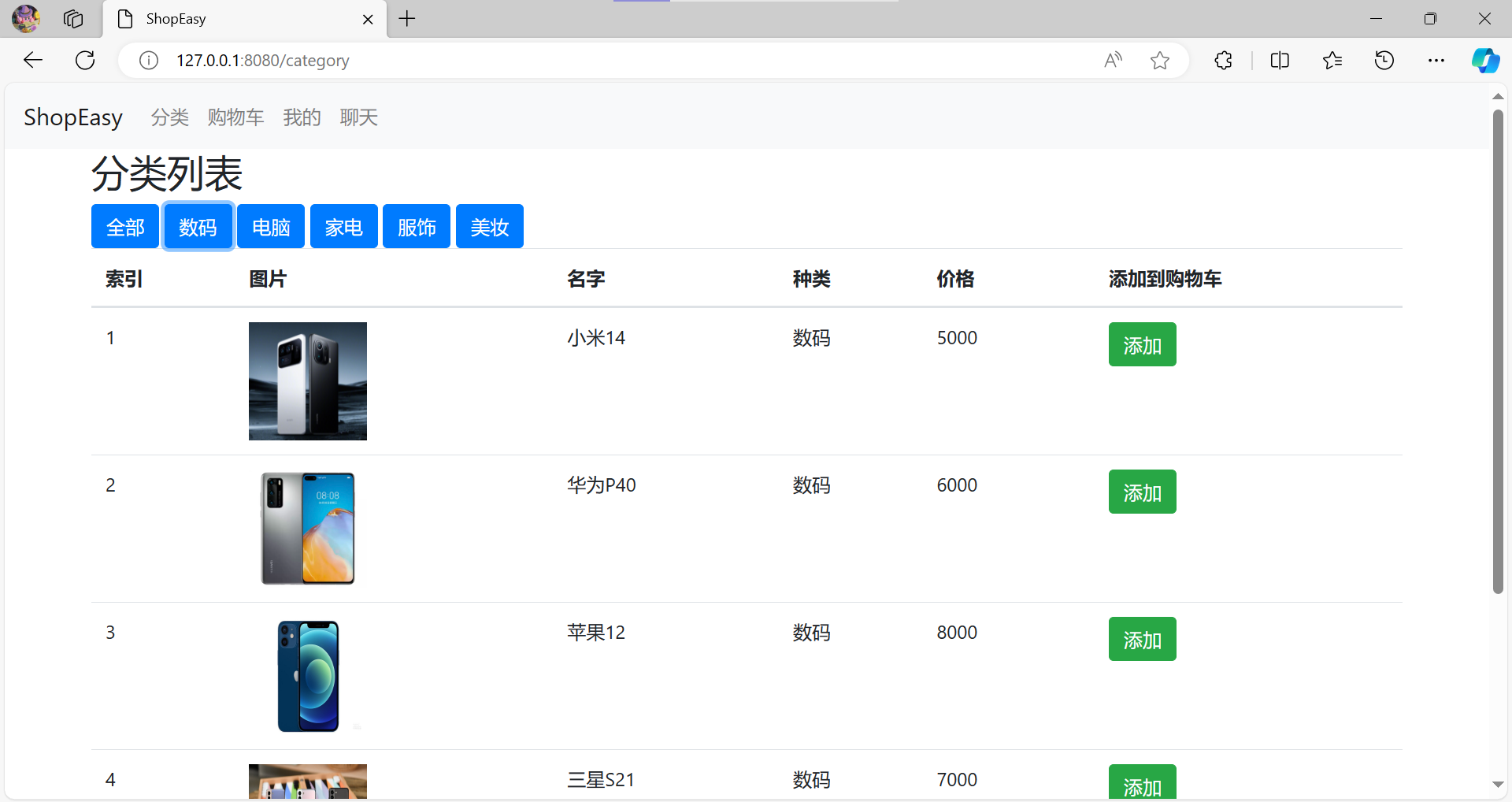
**分类页模块测试**

1. 分类页加载测试：

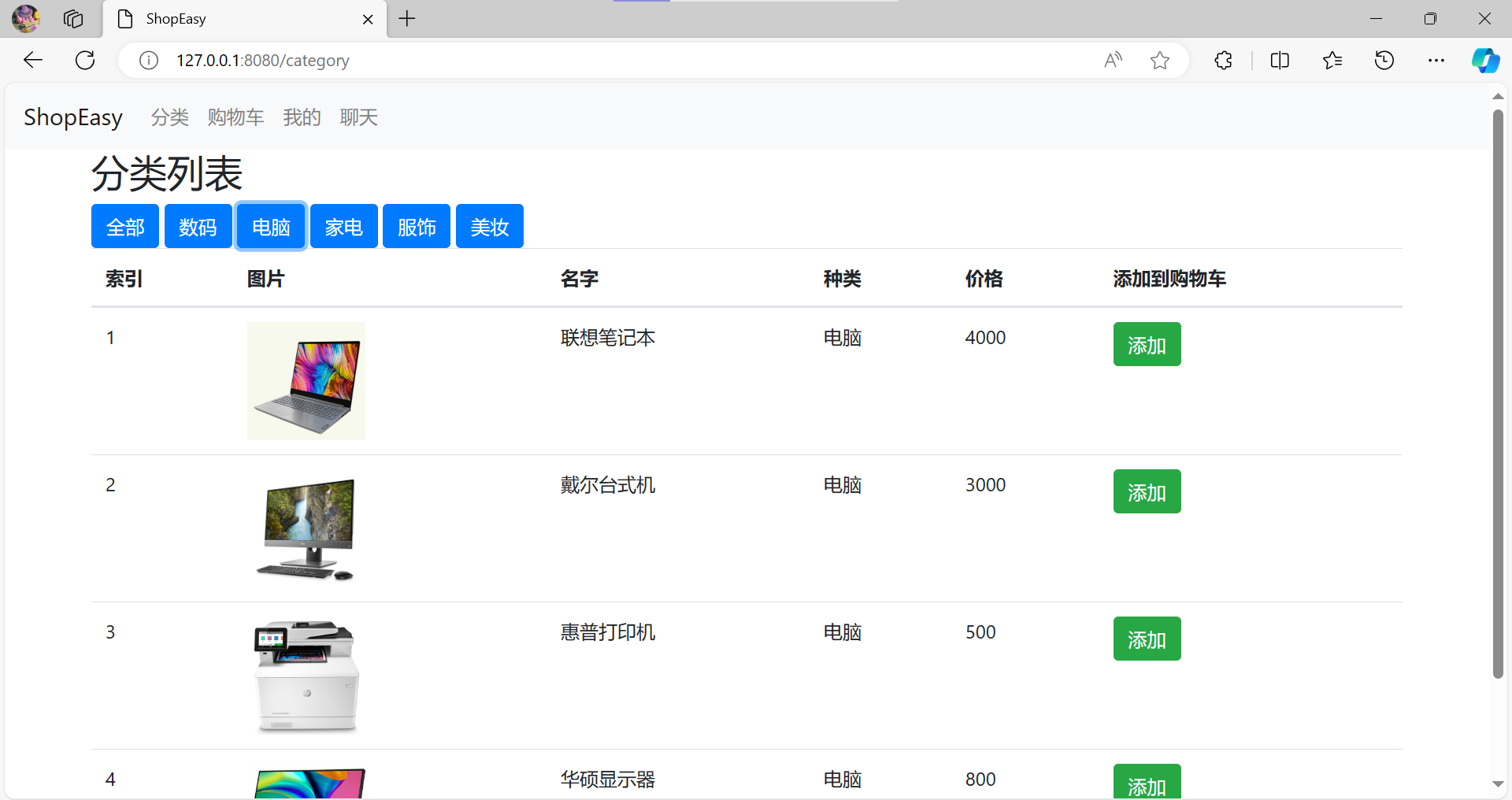
点击分类加载分类页，此时显示的是全部商品：



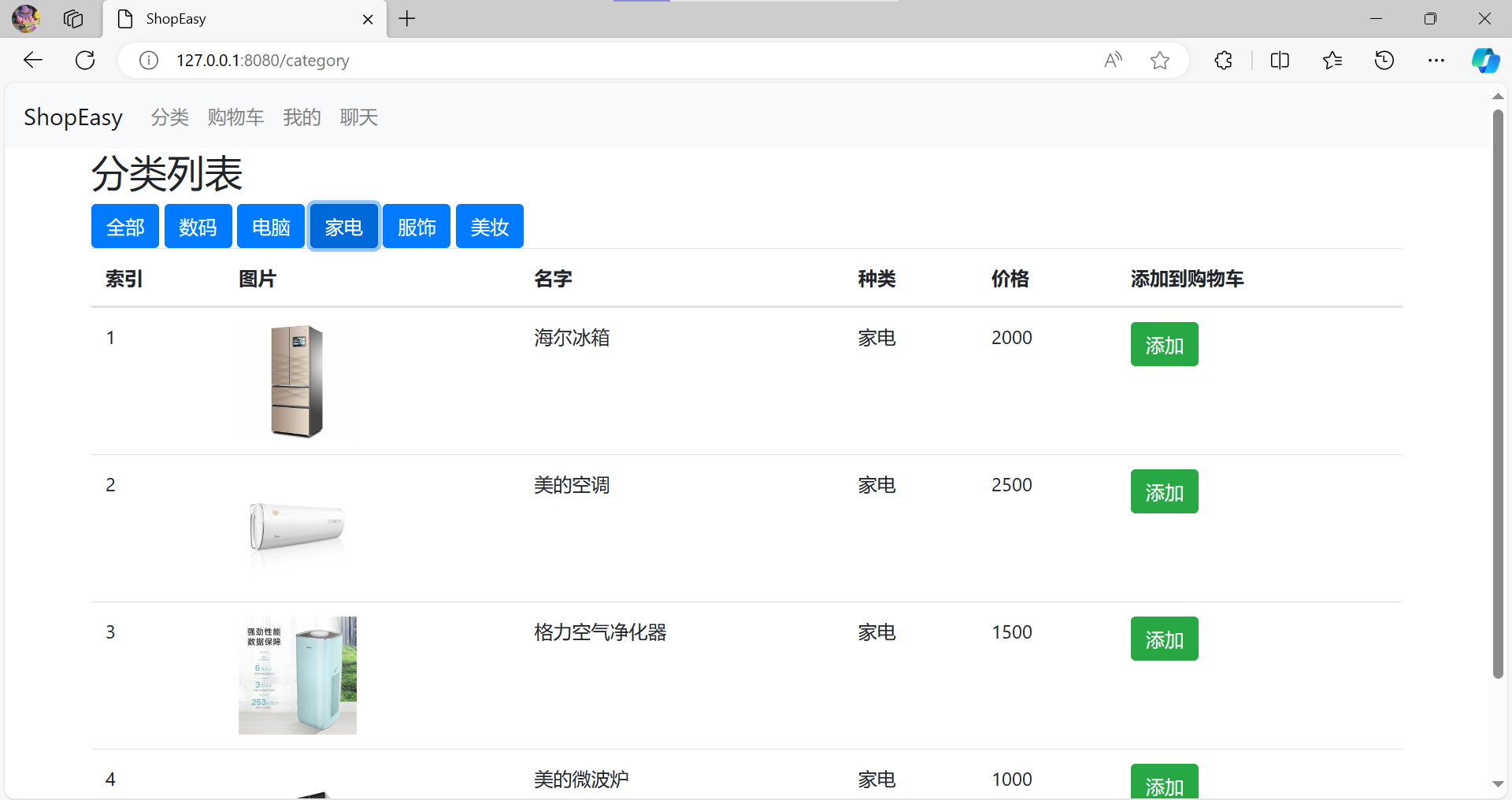
1. 数码类商品列表加载测试：



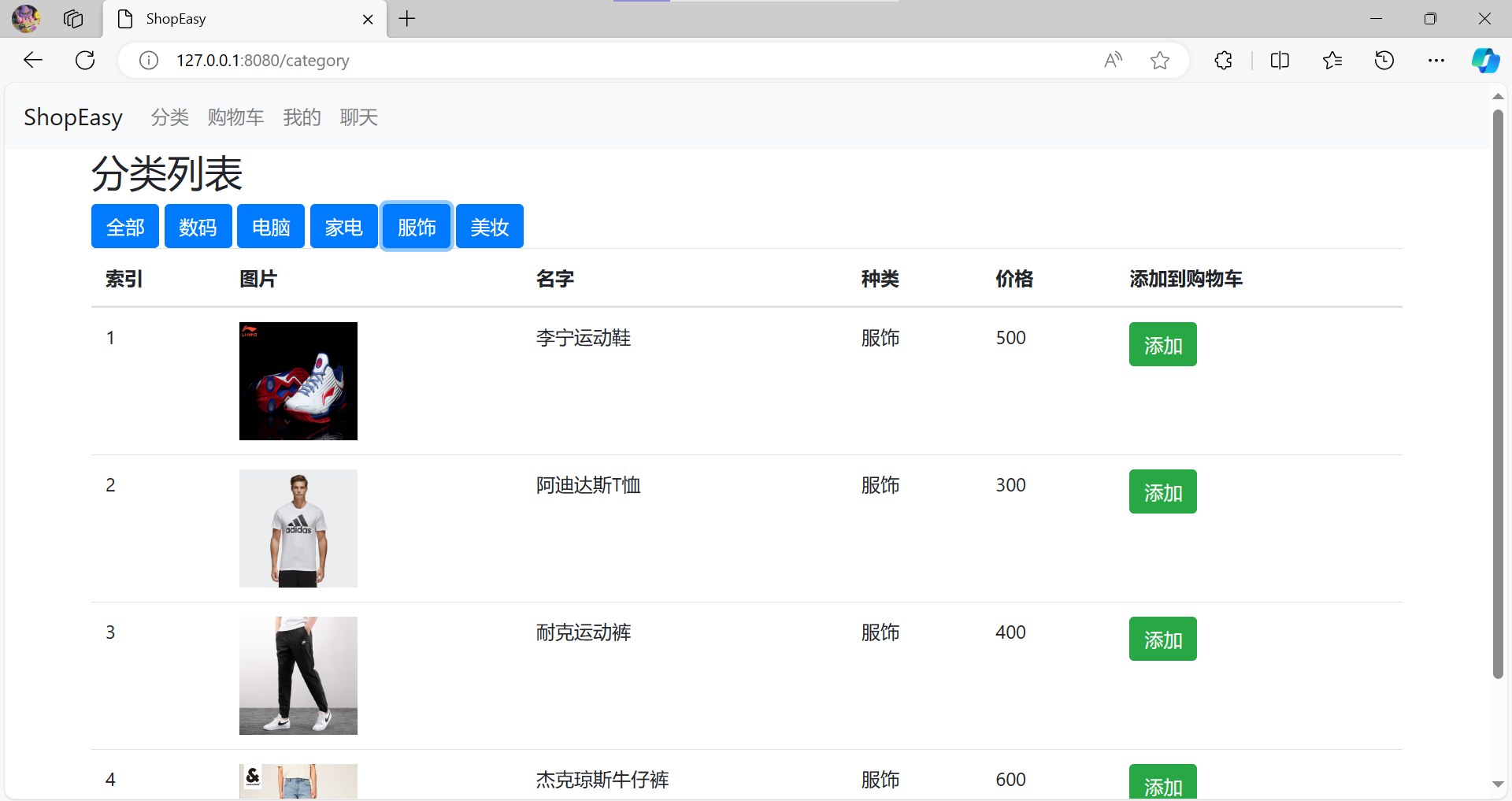
1. 电脑类商品列表加载测试：



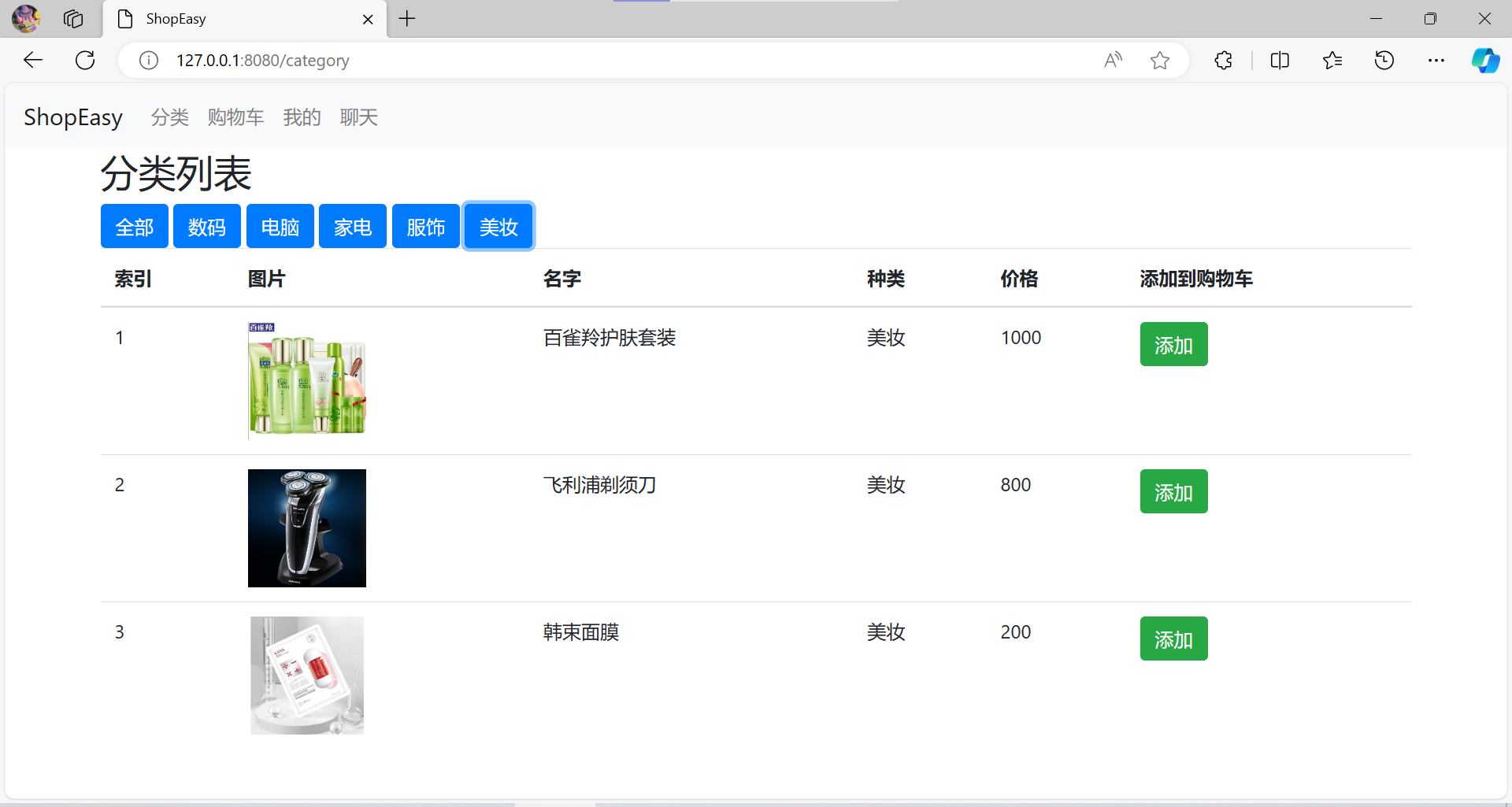
1. 家电类商品列表加载测试：



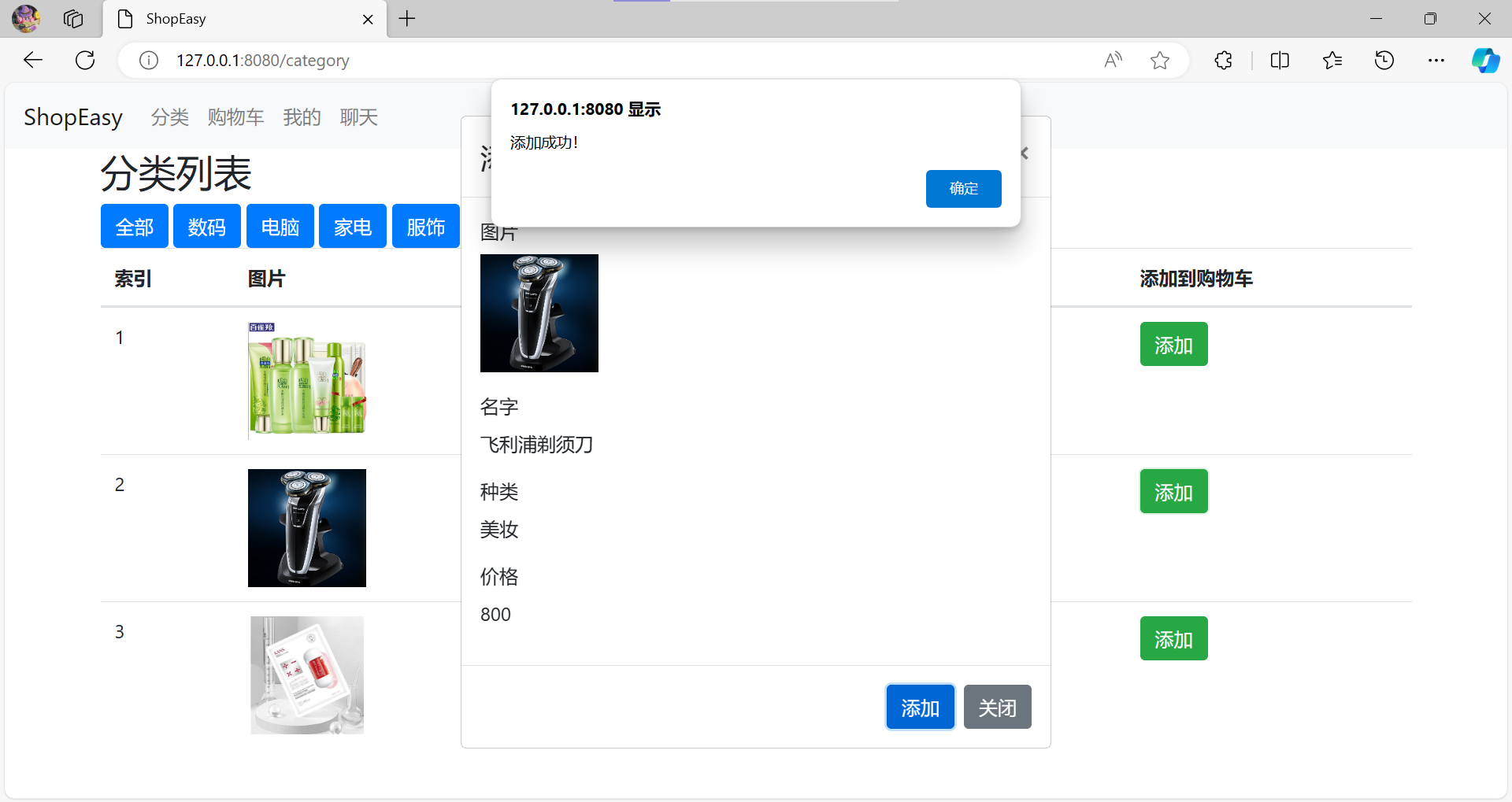
1. 服饰类商品列表加载测试：



1. 美妆类商品列表加载测试：



1. 商品添加至购物车测试：



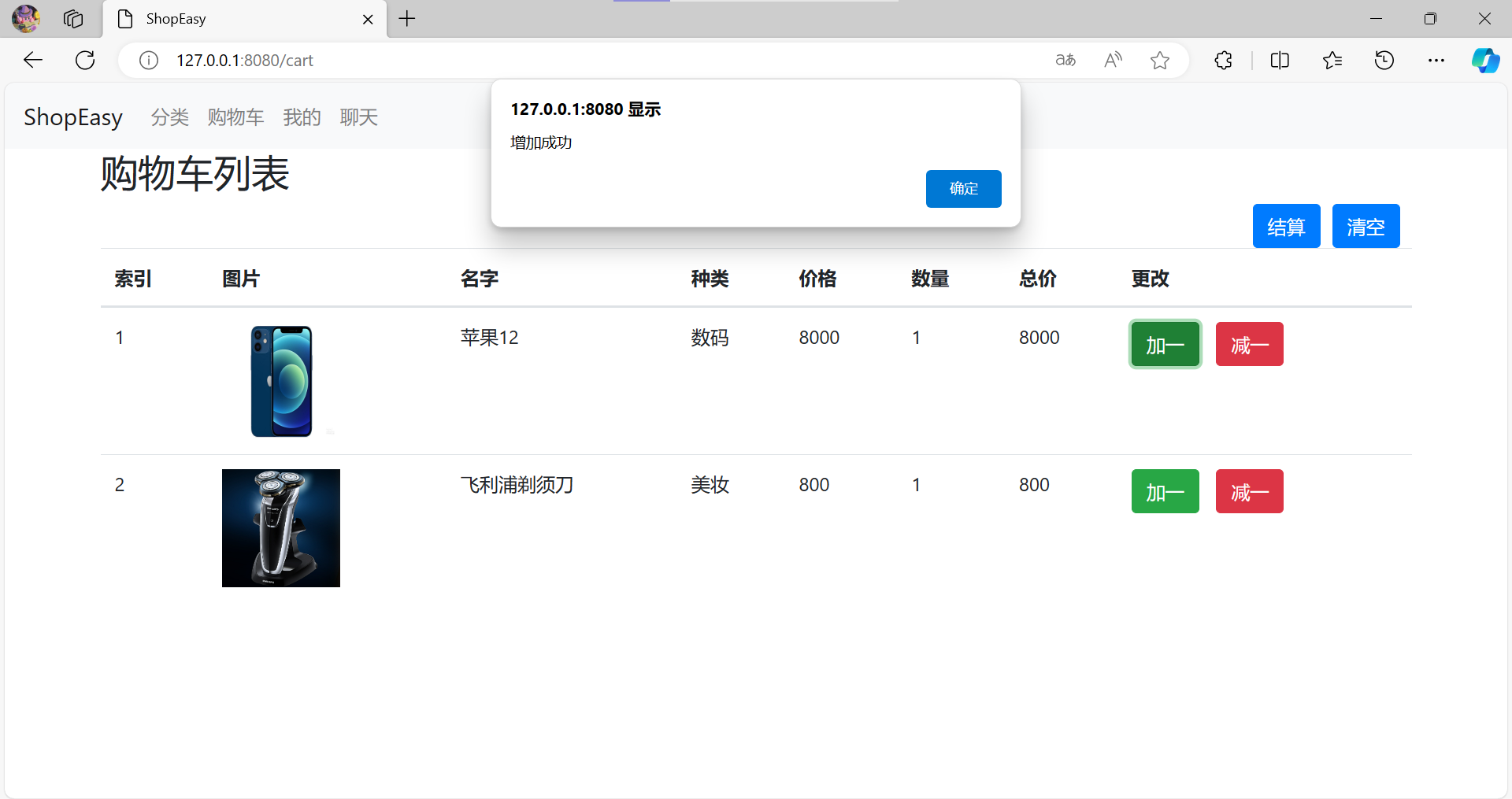
**购物车页模块测试**

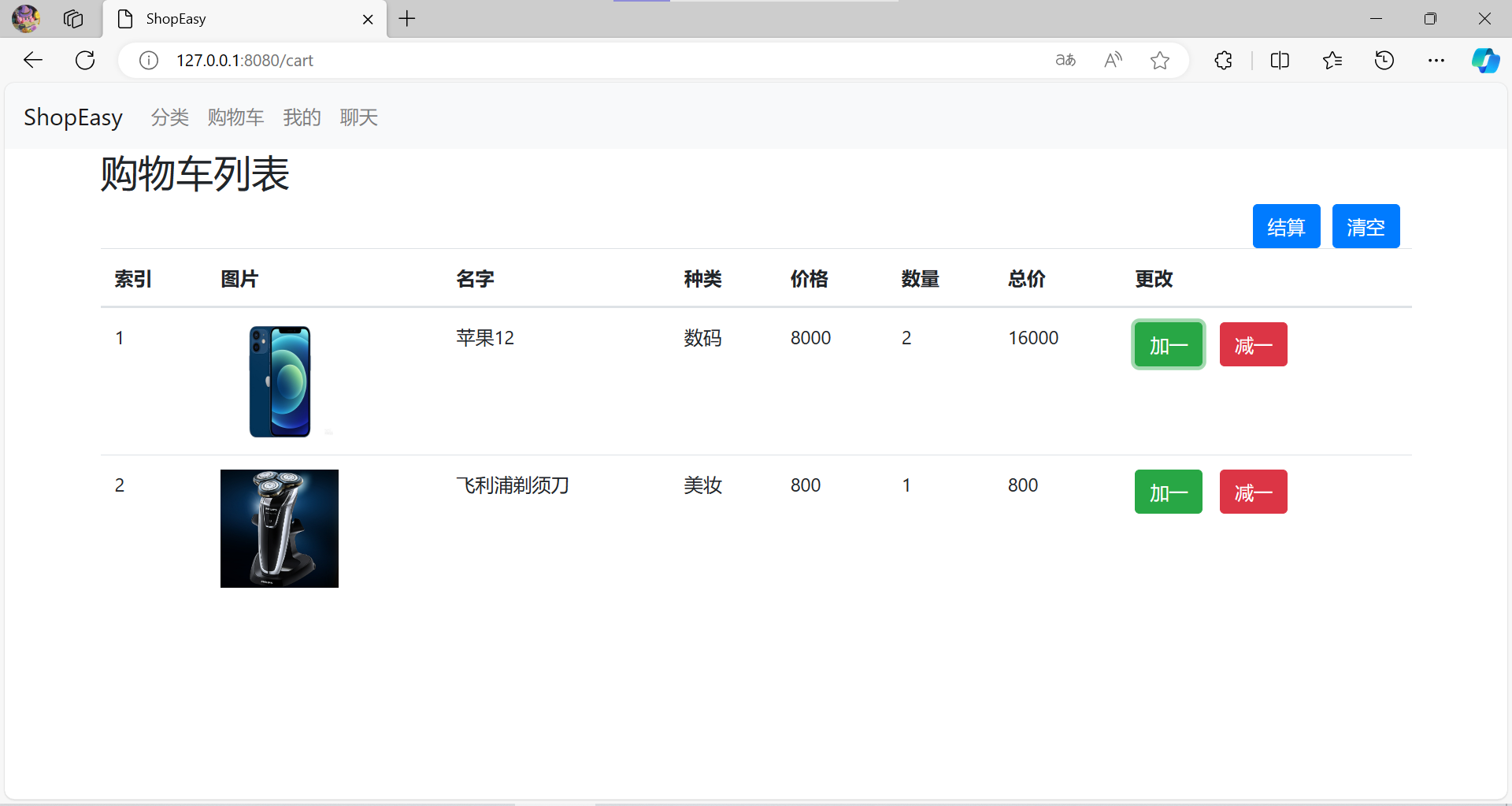
1. 购物车列表加载测试：

点击购物车按键加载购物车：

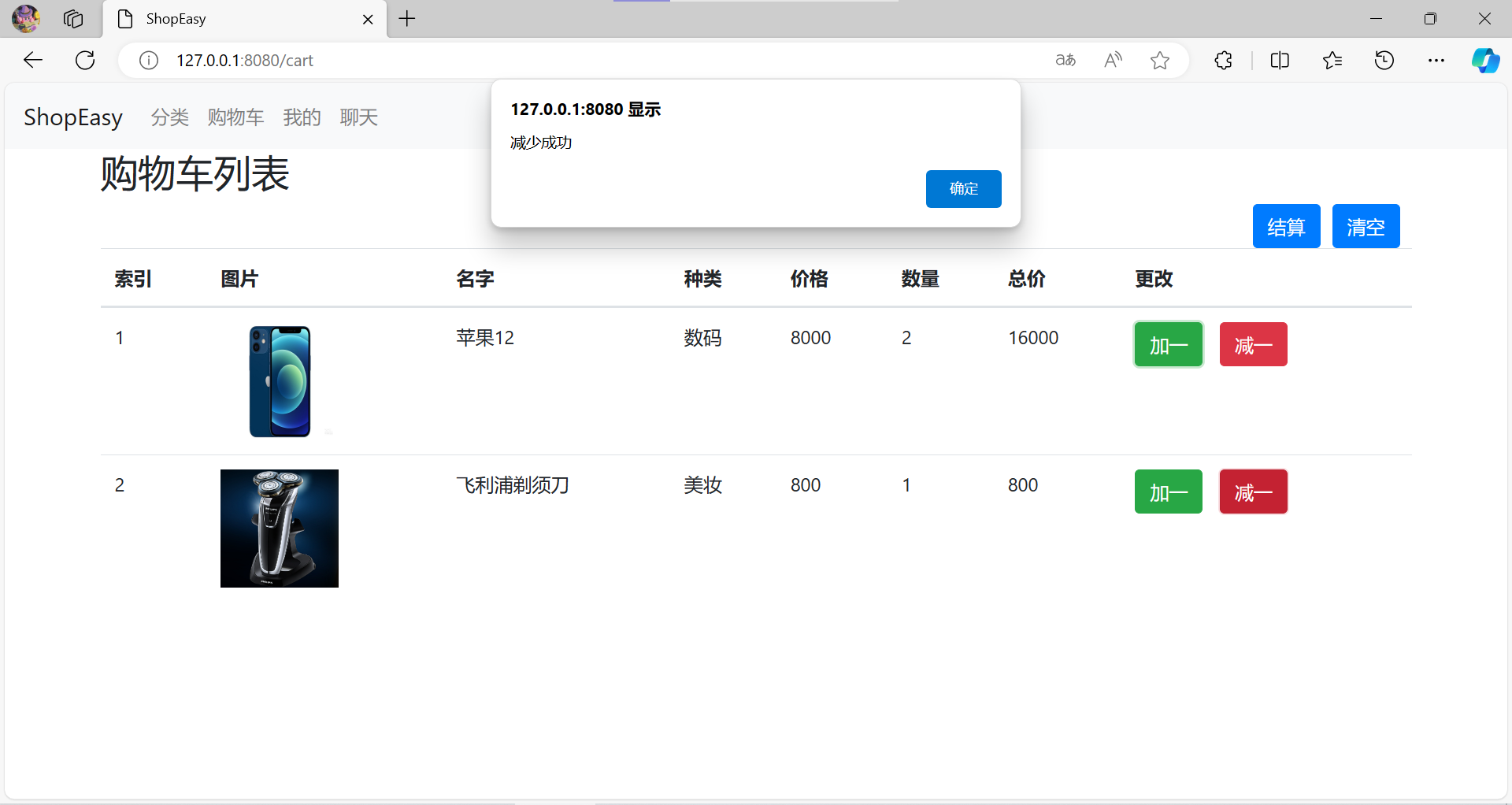


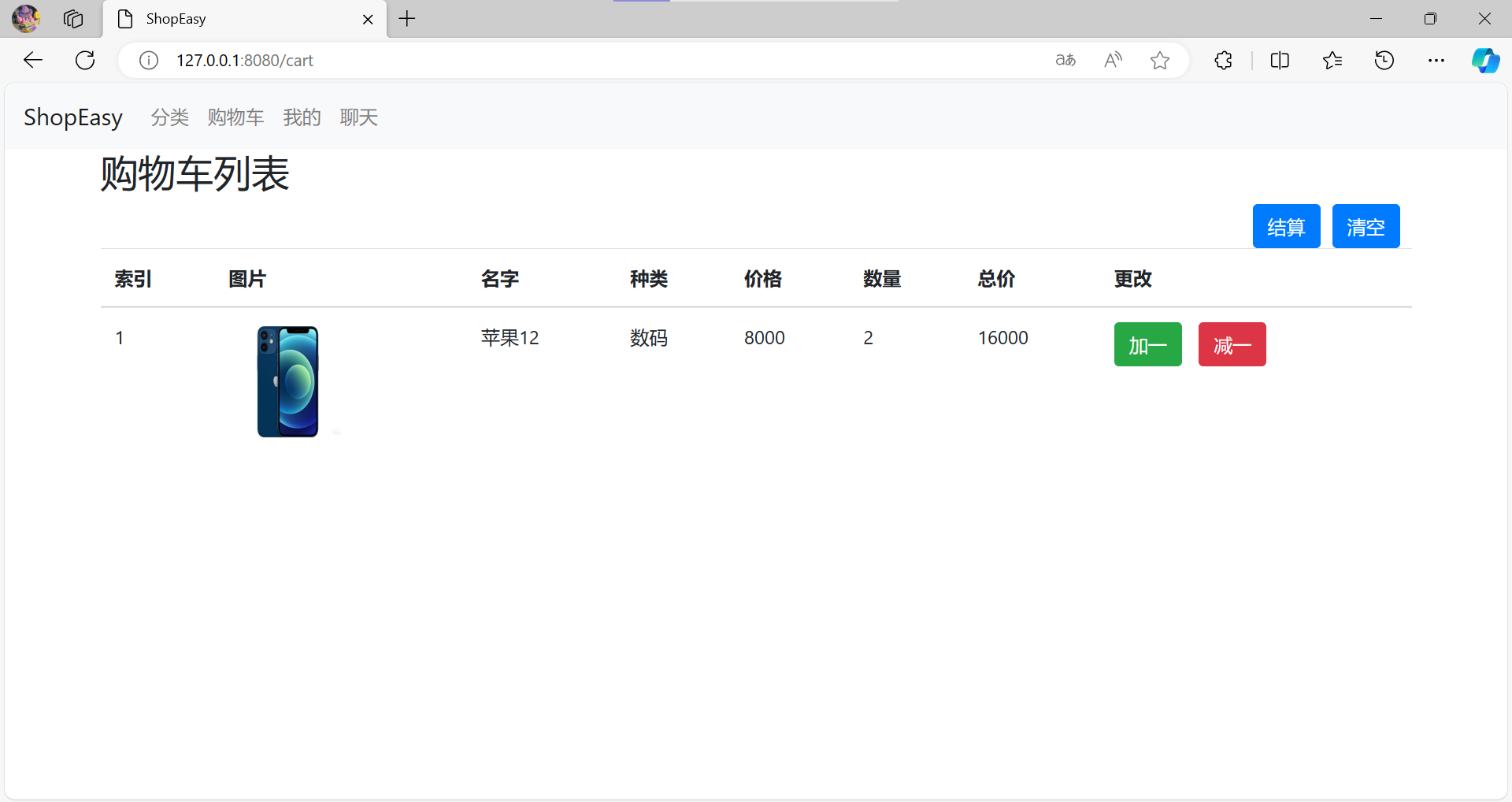
1. 购物车加一功能测试：



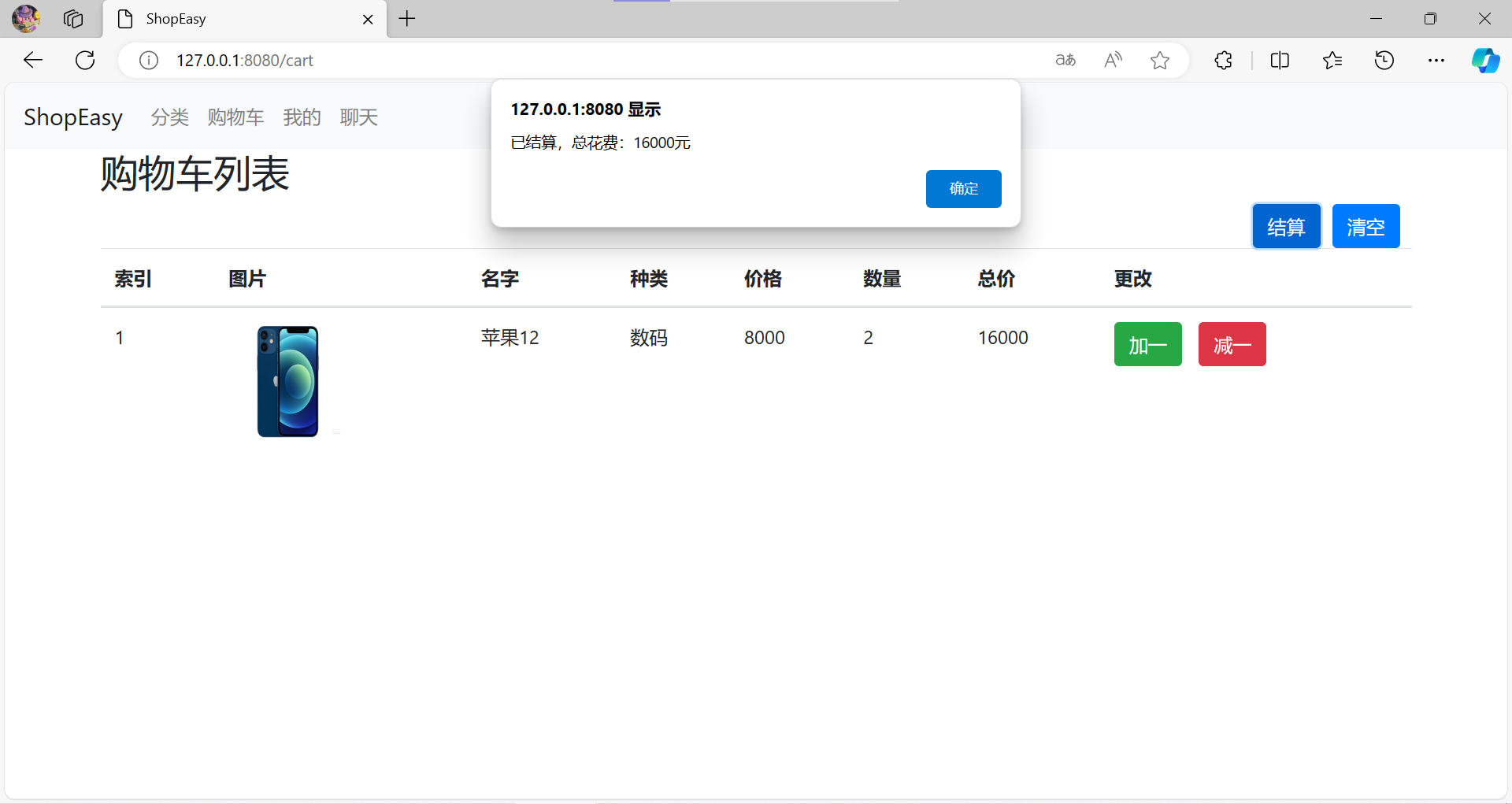


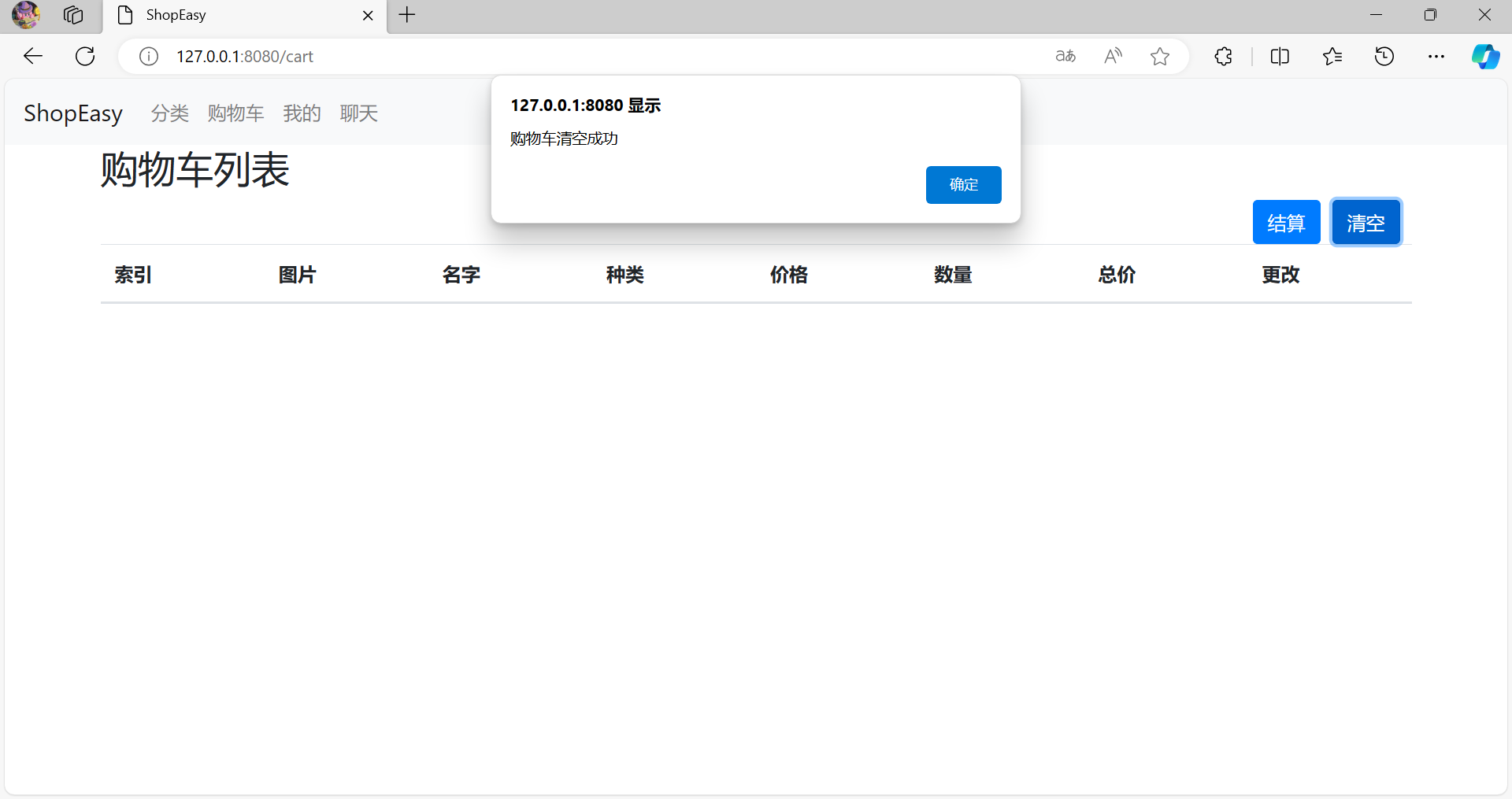
1. 购物车减一功能测试：





1. 购物车结算功能测试：

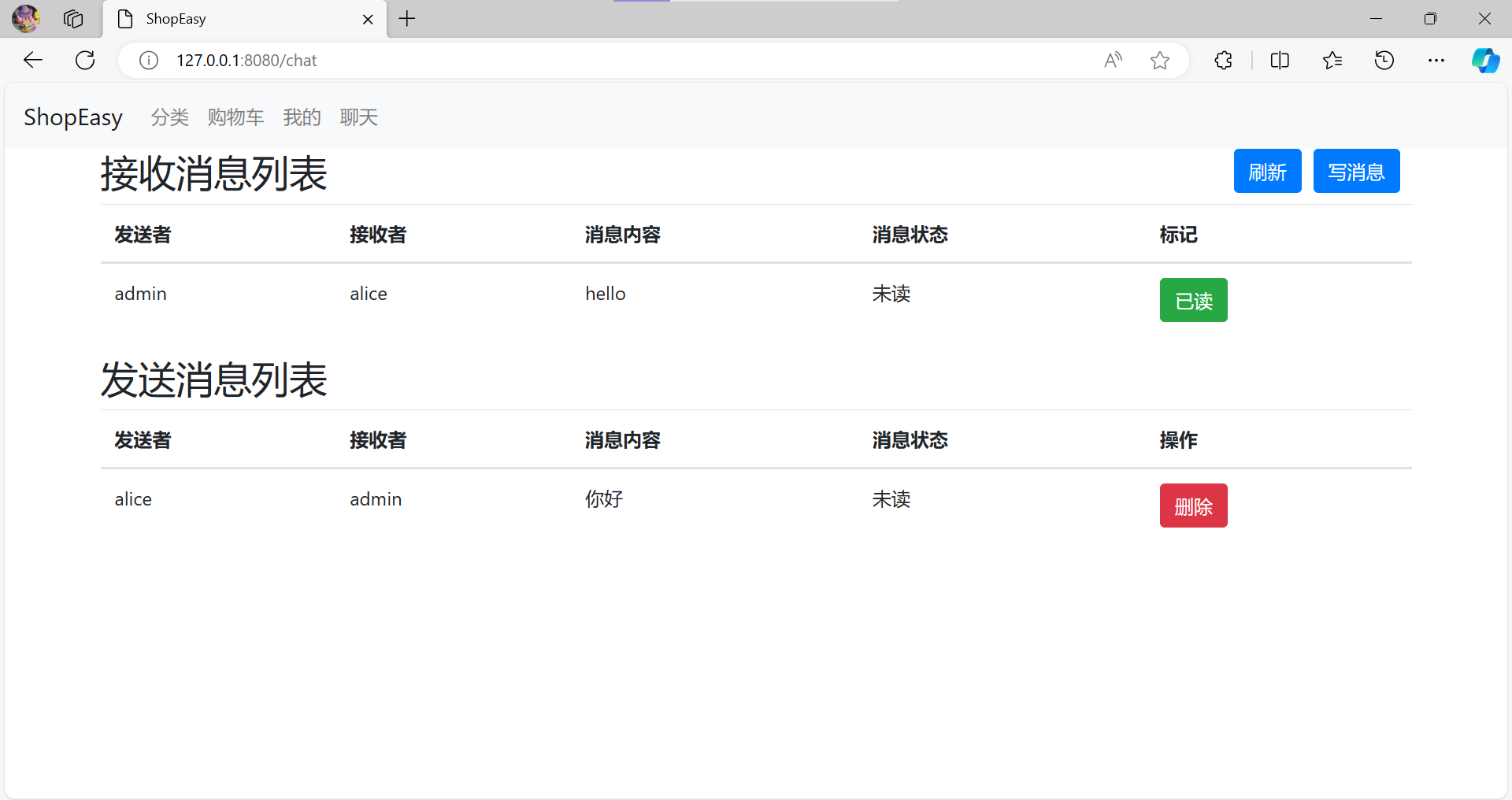




**聊天页测试**

1. 发送、接收消息列表加载测试：

当前登录者为alice，则显示该用户的接收消息列表和发送消息列表：

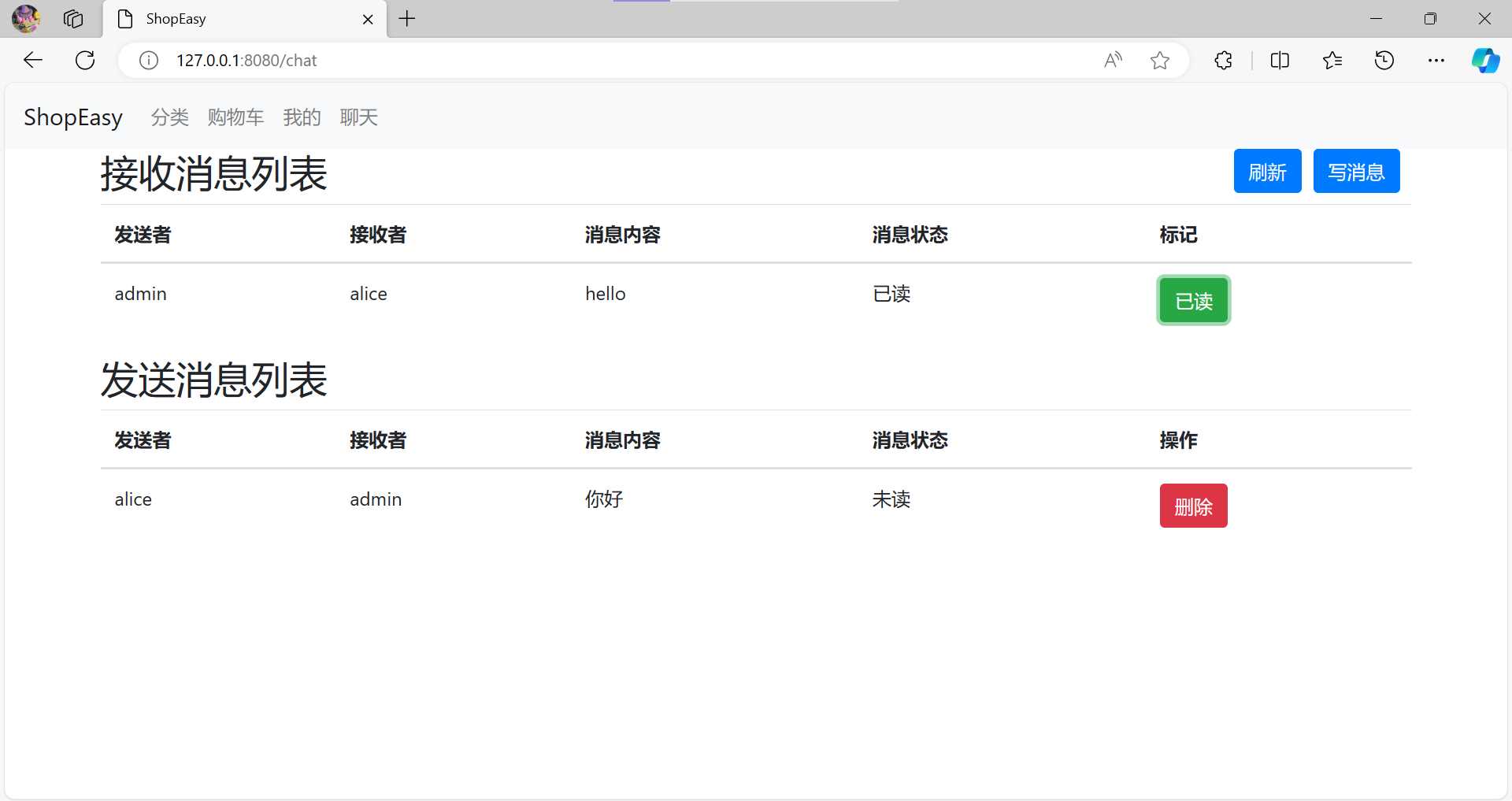


当前登录者为admin，则显示该用户的接收消息列表和发送消息列表：



1. 对接收消息标记功能测试：

在alice用户对接收到的消息做已读操作：



在admin用户点击刷新，可以看到发送的消息已被读取：

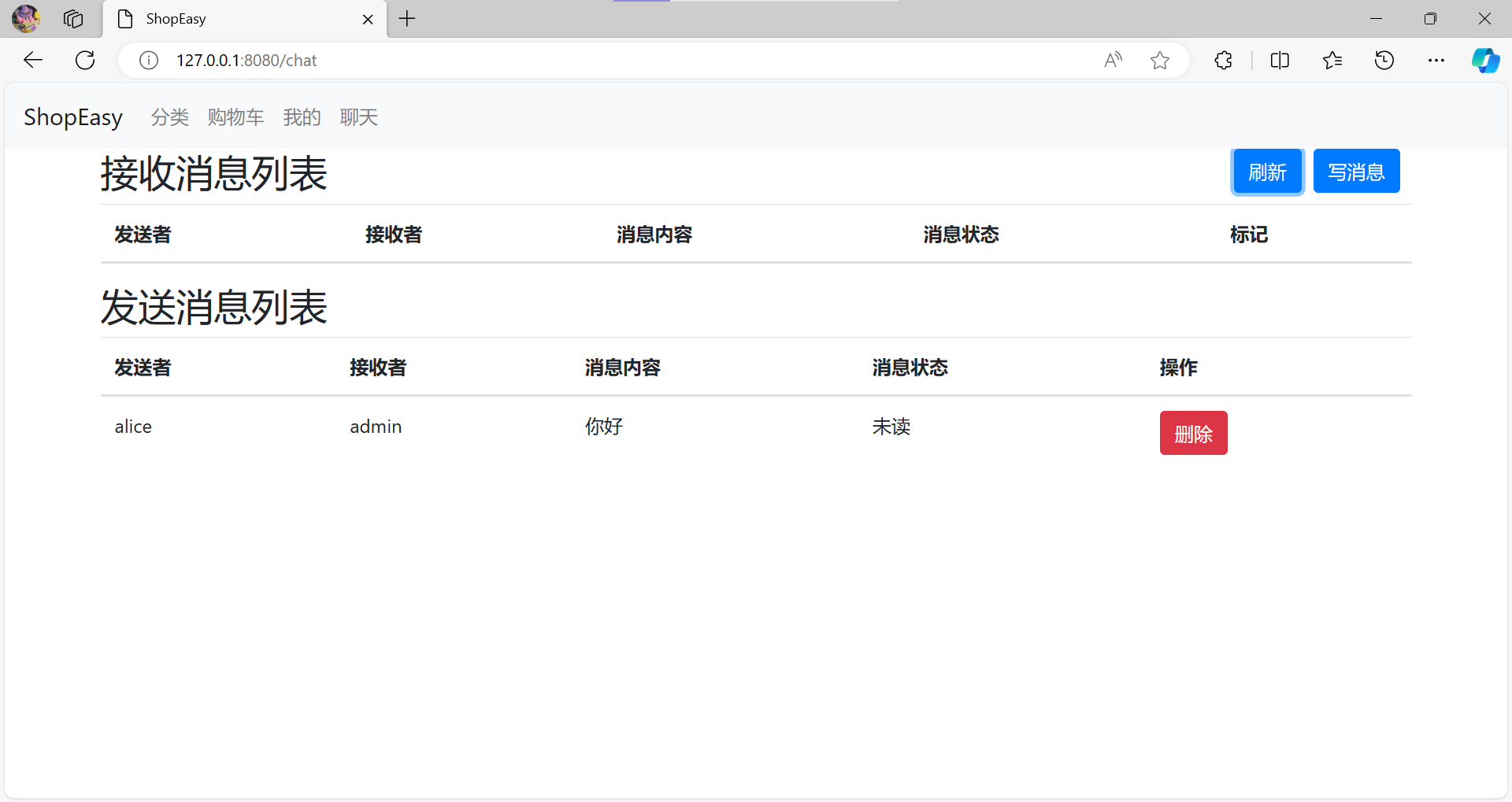


1. 对发送消息删除功能测试：

在admin用户对发送消息删除：

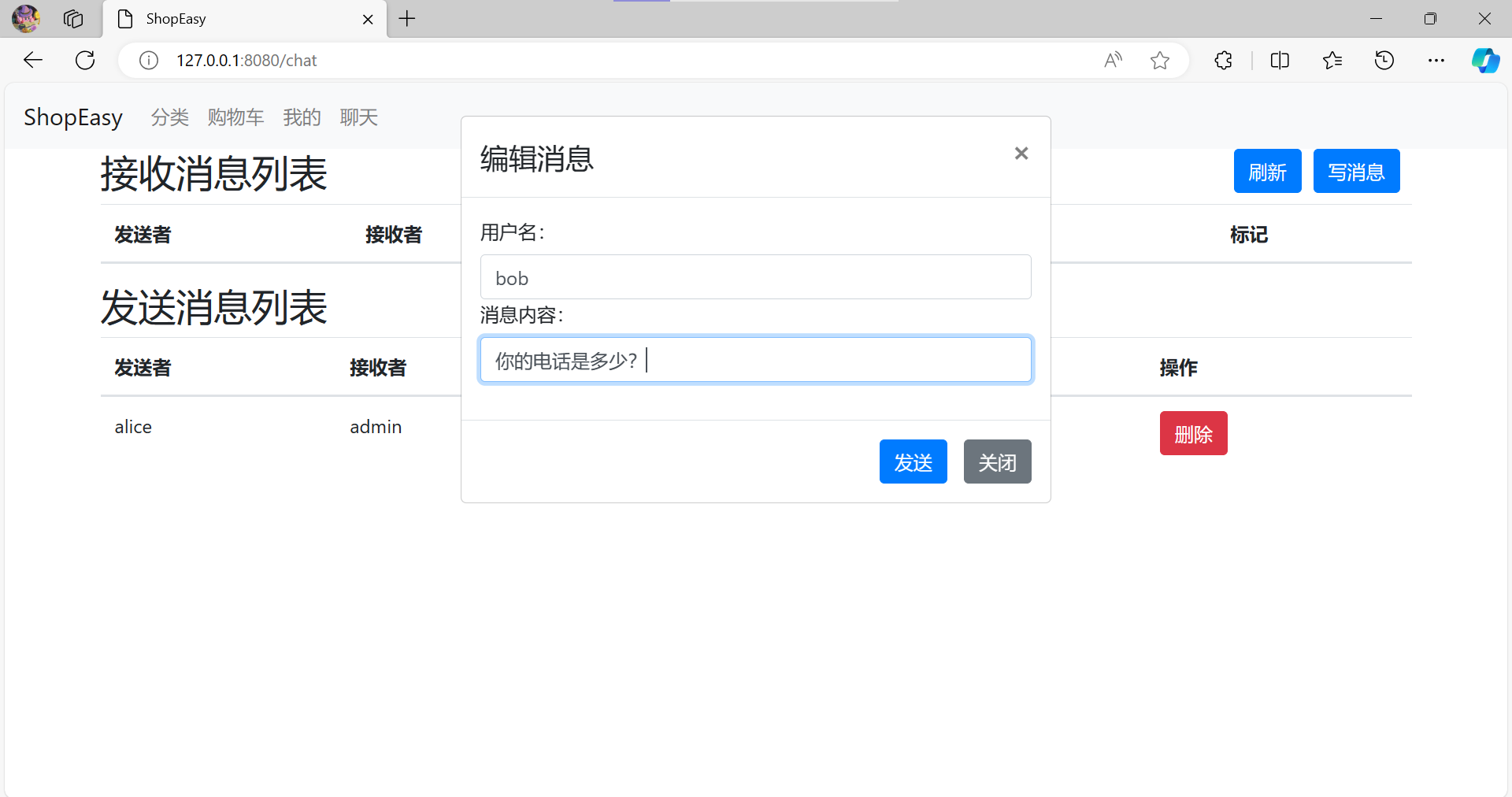


在alice用户点击刷新按键：

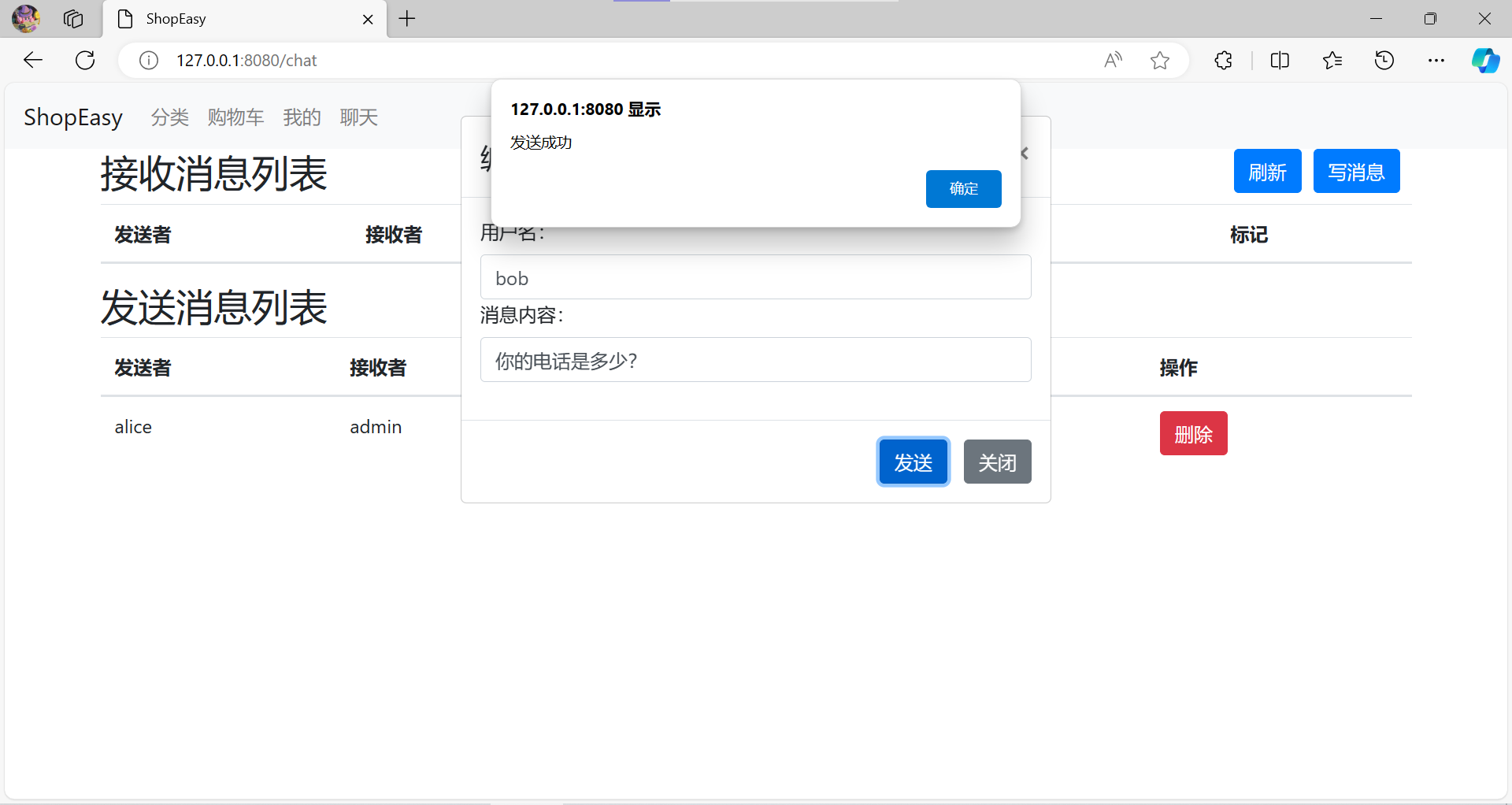


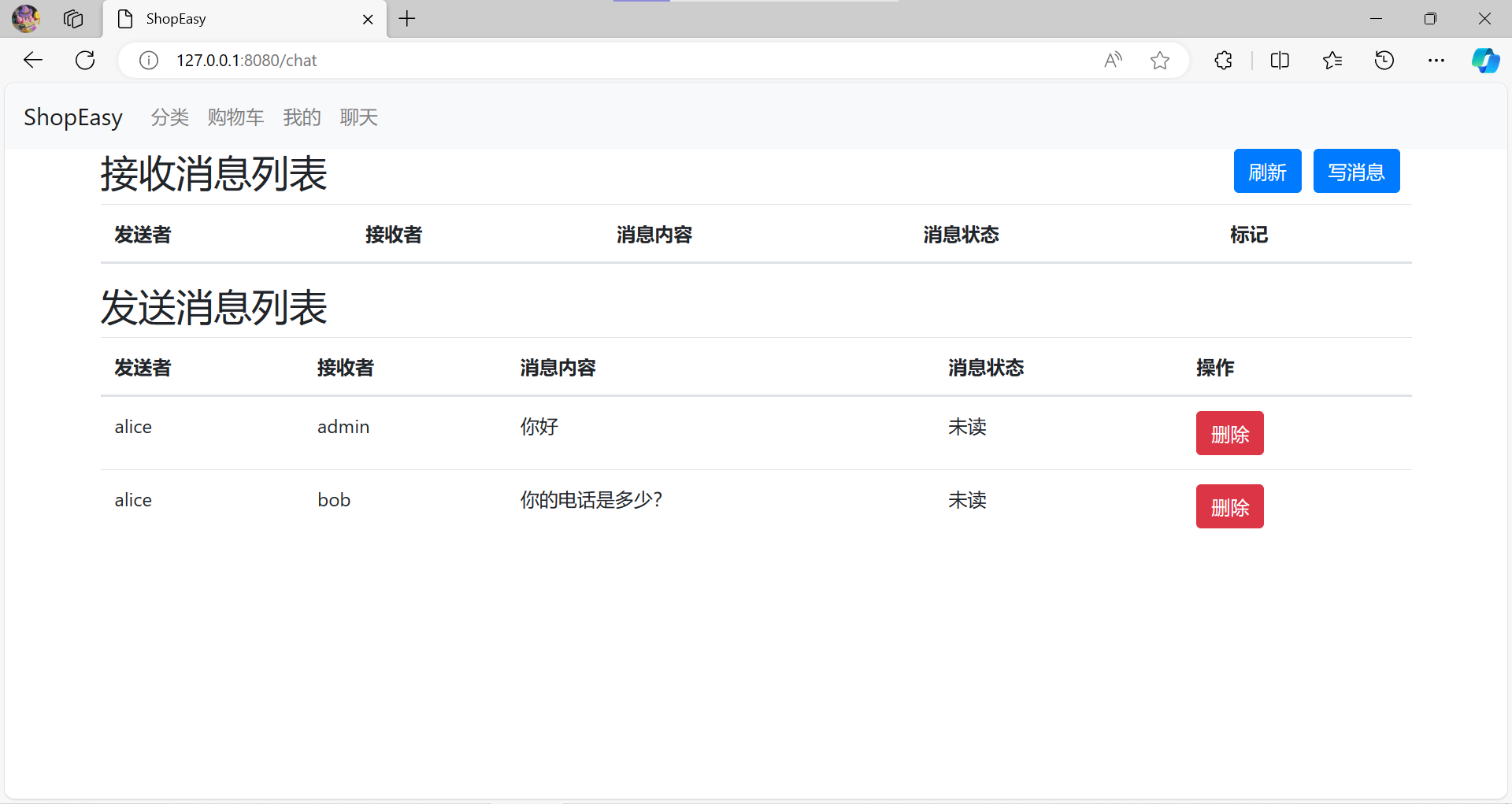
1. 写消息功能测试：

在alice用户点击写消息按键：



点击发送，看到发送成功提示：





在bob用户可以看到接收到alice的消息：



**源代码git**

[ShopEasy](https://github.com/Tingwuren/ShopEasy)（该仓库为github个人私有仓库）。

源代码git详情请参考附件ShopEasy.zip。