厦門大學

电子商城系统 oomall

售后、服务模块软件测试报告

组　名　　空白对照组

组　别　　2-4 组

学　院　　信息学院

专　业　　软件工程

成　员　　李嘉琪、高凯琪、郭宇阳、胡雨璇、黄勖

日　期　　2023 年 12 月 19 日

目录

1. 引言

1.1 编写目的

本项目为厦门大学信息学院软件工程专业软件工程系大三上学期《面向对象分析与设计》、《软件工程导论》、《JavaEE 平台技术》课程大作业。本文档编写目的在于介绍课程大作业中的售后模块、服务模块的详细设计。

项目的总目标为开发一个能支持高并发、大负载的电子商城系统（OOMALL）的服务和售后模块。用户可以使用此系统完成和目前主流电商平台相似的购物流程，如购物、发货、售后等。本小组负责其中售后、服务模块的开发。项目计划选用 Java 代码，由五人进行敏捷开发，并进行软件功能，性能和安全性等方面的测试。

在此软件测试报告中，针对本小组负责的售后、服务模块进行了进一步的测试。软件测试报告的编写旨在为项目提供全面而系统的测试信息，以评估软件在各个方面的功能、性能和稳定性。通过报告，我们能够向项目管理者、开发团队和其他利益相关方传达测试的进展和结果，准确反映系统是否符合预期的质量标准。此外，测试报告还为发现的问题和缺陷提供详细的描述，以便开发团队能够及时修复。通过对测试覆盖范围、测试环境、测试用例、执行结果和问题汇总等方面的全面记录，测试报告有助于为软件交付提供决策支持，确保最终交付的系统达到用户期望的质量水平。

1.2 项目背景

本项目是厦门大学信息学院软件工程专业《软件工程》 《面向对象分析与设计》和《JavaEE 平台技术》三门课程的联合课程设计。2022 年秋季学期由软件工程专业 2020 级学生完成第一次迭代，2023 年秋季学期由软件工程专业 2021 级学生开始第二次迭代。

1.3 定义

- 本报告采用的术语符合国家标准《软件工程术语(GB/T11475-2006)》。
- DDoS 攻击：分布式拒绝服务攻击，利用客户或服务器技术，通过将多台计算机联合起来形成攻击平台，对一个或多个目标发动攻击，以成倍提高拒绝服务攻击的威力。DDoS 攻击通过大量合法请求占用网络资源，致力于瘫痪目标网络。
- 数据分片：一种将大型数据集分解成较小数据块并分散存储在多个节点上的技术。每个数据块分配到不同节点上，实现分布式存储和处理。
- 分布式数据库：一种数据库，其中数据存储在不同物理位置。数据可能存储在同一地点的多台计算机上，也可能分布在互连计算机网络中。
- SSL：安全套接层，以及其继任者传输层安全，用于为网络通信提供安全性和数据完整性的安全协议。
- TLS：传输层安全，用于在两个通信应用程序之间提供机密性和数据完整性。
- 访问令牌：是 Windows 操作系统安全性的概念。用户登录时，系统创建一个包含登录进程返回的 SID 以及本地安全策略分配给用户和用户安全组特权列表的访问令牌。
- 刷新令牌：用于获取访问令牌的凭据。授权服务器颁发刷新令牌给客户端，用于在当前访问令牌失效或过期时获取新的访问令牌，或者获取具有相同或更窄范围的附加访问令牌。

1.4 参考资料

- 《软件工程术语（GB/T11475-2006） 》
- 《需求规格说明书》
- 中国人民银行办公厅。

关于进一步加强无证经验支付业务整治工作的通知。 银办发[2017]217 号文

- 中国人民银行。
  关于规范支付创新业务通知。 银办发[2017]281 号文
- 中国人民银行。 关于印发 < 条码支付业务规范（试行）>的通知。
  银办发[2017]296 号文
- 郑志成。 京东到家支付平台的高可用性架构计。
  https://www.zhihu.com/question/527868488/answer/2438919186
- 微信支付。
  https://pay.weixin.qq.com/wiki/doc/apiv3/apis/chapter8_1_1.shtm
- 支付宝互联网平台交易查询接口。
  https://opendocs.alipay.com/open/02e7gm?ref=api

2. 测试计划执行情况

2.1 测试项目

- 单元测试方法：
- 白盒测试：针对每个 API 功能编写测试用例，验证函数的调用关系、分支的跳转关系和输出是否符合预期。确保覆盖所有独立执行路径，并测试所有的逻辑判定情况。
- 黑盒测试：采用等价类划分和边界值分析，重点测试每个 API 是否能完成预期功能。特别关注边界值和异常情况的处理。

本项目计划采用白盒+黑盒的测试方法进行单元测试，白盒测试中针对每个 API 功能，编写测试用例对 API 给定输入，验证函数的调用关系，分支的跳转关系，输出是否符合预期。黑盒测试中，采用等价类划分，边界值分析，着重测试每个 API 能否完成预期功能；采用 Jmeter 性能测试，检查程序是否能达到 1 秒 1000 个测试用例的性能需求，检查在大负载高并发的情况下程序是否会出现卡顿，崩溃。

- 面向对象测试方法：

  测试类之间的调用关系和继承封装关系是否符合预期。

  确保数据结构的正确性，检查类内部的数据结构是否满足设计要求。

  本项目计划采用面向对象的测试方法，测试所编写的类之间的调用关系，继承封装关系是否符合预期。

编写测试用例：

项目编写测试用例需要考虑一下几个方面：

1. 应保证所有独立执行路径至少被执行一次
2. 应保证对所有的逻辑判定，真/假情况都至少测试一次
3. 应保证在上下边界和可操作的范围内执行所有的循环
4. 应检查代码内部的数据结构是否正确
5. 应测试程序出错时的错误处理是否符合预期
6. 应测试接口之间的调用是否正确
7. 应对性能进行要求，关注程序测试用例的执行时间性能测试：

性能测试：

使用 Jmeter 进行性能测试，模拟大负载和高并发场景，评估系统的响应时间和吞吐量。

检查程序是否能达到性能需求，避免出现卡顿和崩溃情况。

测试名称	测试内容	测试目的	备注
对售后模块测试	AftersaleTest	对售后模块的 API 进行测试	在 12/28 号前完成测试，达到测试目标
对服务模块测试	ServiceTest	对服务模块的 API 进行测试	在 12/28 号前完成测试，达到测试目标

2.2 测试机构和人员

测试机构：2-4 组，指导老师邱明

负责人：指导老师邱明

职责分配：

1. 开发人员：

   单元测试：负责编写并执行单元测试用例，验证他们所编写的代码的正确性，确保每个功能模块都有相应的单元测试覆盖。

   集成测试：参与集成测试，确保各个模块的协同工作正常，需要检查接口之间的交互，以及模块之间的数据传递是否正确。

2. 测试人员：

   功能测试：负责设计和执行功能测试用例，验证产品功能是否符合需求和预期关注用户故事和需求的实现，以确保产品的功能完整性。

   非功能测试：进行非功能性测试，例如性能测试、安全性测试、可用性测试。需要评估产品在不同方面的表现，并提供改进建议。

3. 产品负责人：

   验收测试：产品负责人应该参与验收测试，验证开发团队交付的功能是否满足业务需求和预期，确认产品的质量和功能是否达到了接受的标准。

4. 敏捷负责人：

   支持测试：支持测试团队，提供必要的资源和环境，确保能够高效地执行测试任务。协调团队内部和团队间的沟通，解决问题并促进合作。

2.3 测试结果

2.3.1 售后模块测试

测试内容：

编号	API	功能	测试用例	描述
1	POST	顾客提交	TestAftersales1	成功申请

	/order/{oid}/orderitem/{id}/aftersales	售后申请	TestAftersales2	申请信息无效
			TestAftersales3	信息超出边界值
			TestAftersales4	用户无权限
2	GET /aftersales	顾客查询售后列表（根据售后状态分类查询）	TestAftersales1	默认成功查询
			TestAftersales2	根据售后状态成功查询
			TestAftersales3	用户无权限
3	GET/aftersales/orderitems	顾客查询所有的可申请售后的订单明细	TestAftersales1	默认成功查询
			TestAftersales2	根据页码页数成功查询
			TestAftersales3	退换货订单查询异常
			TestAftersales4	用户无权限
4	GET /aftersales/{id}	顾客根据售后单 id 查询售后单信息	TestAftersales1	成功查询
			TestAftersales2	用户无权限
			TestAftersales3	售后单不存在
5	PUT /aftersales/{id}	顾客修改售后单信息（只能在申请态）	TestAftersales1	成功修改
			TestAftersales2	用户无权限
			TestAftersales3	售后单不在申请态
			TestAftersales4	地址信息不完整
6	DELETE /aftersales/{id}	顾客取消售后	TestAftersales1	成功取消
			TestAftersales2	用户无权限
			TestAftersales3	取消已售后的售后单
			TestAftersales4	售后单不存在
7	POST /aftersales/{id}/arbitrations	顾客申请售后仲裁*	TestAftersales1	成功申请
			TestAftersales2	用户无权限
			TestAftersales3	售后单正在仲裁中

			TestAftersales4	未提供仲裁理由
8	DELETE /arbitration/{aid}	顾客取消仲裁*	TestAftersales1	成功取消
			TestAftersales2	用户无权限
			TestAftersales3	仲裁单已仲裁
			TestAftersales4	未提供仲裁单 id
9	GET /aftersales/{id}/express	顾客或商铺查询售后单对应的物流单号*	TestAftersales1	成功查询
			TestAftersales2	用户无权限
			TestAftersales3	未退换货售后单不存在
			TestAftersales4	已完成退换货售后单
10	PUT /shops/{shopid}/aftersales/{id}/confirm	商铺审核售后	TestAftersales1	成功审核
			TestAftersales2	用户无权限
			TestAftersales3	售后单不存在
			TestAftersales4	审核已处理售后单
11	PUT /shops/{shopid}/aftersales/{id}/receive	商铺验收售后商品	TestAftersales1	成功验收
			TestAftersales2	用户无权限
			TestAftersales3	售后单不存在
			TestAftersales4	审核非退换货售后单
			TestAftersales5	验收已处理的售后单
12	DELETE /shops/{shopid}/aftersale/{id}/cancel	商铺取消售后	TestAftersales1	成功取消
			TestAftersales2	用户无权限
			TestAftersales3	售后单不存在
			TestAftersales4	取消已处理的售后单
13	PUT /shops/{shopid}/arbitrations/{id}/response	商铺应诉仲裁*	TestAftersales1	成功应诉
			TestAftersales2	用户无权限

			TestAftersales3	仲裁单不存在
			TestAftersales4	应诉已处理的仲裁单
14	PUT /shops/{shopid}/arbitrations/{id}/accept	管理员受理仲裁单	TestAftersales1	成功受理
			TestAftersales2	用户无权限
			TestAftersales3	商户 id 不为 0
			TestAftersales4	仲裁单不在申请中
15	PUT /shops/{shopid}/arbitrations/{id}/close	管理员仲裁结案	TestAftersales1	成功仲裁
			TestAftersales2	用户无权限
			TestAftersales3	商铺 id 不为 0
			TestAftersales4	仲裁单不在受理态
			TestAftersales5	非负责仲裁员结案
16	GET /shops/{shopid}/aftersales	商铺或管理员查看所有售后单（可根据售后类型和状态选择）	TestAftersales1	成功查看
			TestAftersales2	用户无权限
			TestAftersales3	商铺 id 不为 0
			TestAftersales4	无查询条件
17	GET /shops/{shopid}/aftersales/{id}	商铺或管理员根据售后单 id 查询售后单信息	TestAftersales1	成功查询
			TestAftersales2	用户无权限
			TestAftersales3	商铺 id 不为 0
			TestAftersales4	售后单 id 不存在
18	GET /shops/{shopid}/arbitrations	商铺或管理员查询顾客申请的仲裁单信息（根据仲裁单状态）	TestAftersales1	成功查询
			TestAftersales2	用户无权限
			TestAftersales3	商铺 id 不为 0
			TestAftersales4	无查询条件
19	GET /shops/{shopid}/arbitrations/{id}	商铺或管理员查询仲裁单详	TestAftersales1	成功查询
			TestAftersales2	用户无权限

		情信息	TestAftersales3	店铺 id 不为 0
			TestAftersales4	仲裁单 id 不存在

2.3.2 售后模块测试用例

1. POST /order/{oid}/orderitem/{id}/aftersales 顾客提交售后申请*

1. **正常流程测试用例:**

输入:

- 用户 token: "valid_token"
- 订单 id（oid）: 123
- 商品 id（id）: 456
- 请求体:

```
{
  "product_id": 789,
  "type": 1,
  "quantity": 2,
  "reason": "Product damaged",
  "consignnee": {
    "name": "John Doe",
    "mobile": "1234567890",
    "regionId": 1,
    "address": "123 Main St"
  }
}
```

输出:

```
{
  "errno": {},
  "errmsg": "string",
  "data": {
    "id": 1,
    "type": 1,
    "reason": "Product damaged",
    "conclusion": "string",
    "quantity": 2,
    "consignee": {
      "name": "John Doe",
      "mobile": "1234567890",
      "regionId": 1,
      "address": "123 Main St"
```

```
    },
    "status": 0,
    "gmt_apply": "string",
    "gmt_end": "string",
    "order_item_id": 456,
    "product_item_id": 789,
    "product_id": 123,
    "shop_id": 0,
    "inArbitrated": 0,
    "arbitration_id": 0,
    "customer_id": 0
  }
}
```


2. **异常流程测试用例:**

输入:

- 无效用户 token: "invalid_token"
- 无效订单 id（oid）: -1
- 无效商品 id（id）: -1
- 不完整的请求体:


```
{ "product_id": 789, "quantity": 2, "reason": "Product damaged" }
```


输出:


```
{ "errno": "error_code", "errmsg": "Error message" }
```


3. **边界值测试用例:**

输入:

- 最小订单 id 和商品 id: 0
- 最大订单 id 和商品 id: 2147483647
- 请求体字段边界值测试

输出: *(期望输出根据具体测试情况调整)*

4. **权限测试用例:**

输入:

- 不提供用户 token

输出:


```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```


输入:

- 使用不同用户的 token 访问其他用户的订单

输出:


```
{ "errno": "forbidden", "errmsg": "Forbidden access" }
```


2. GET /aftersales 顾客查询售后列表（根据售后状态分类查询）

1. **默认查询测试用例:**

输入:

- 用户 token: "valid_token"

- 默认页码和每页数目

期望输出：


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "page": 1,
    "pageSize": 10,
    "list": [
      {
        "id": 1,
        "type": 0,
        "reason": "Product damaged",
        "conclusion": "string",
        "quantity": 2,
        "consignee": {
          "name": "John Doe",
          "mobile": "1234567890",
          "regionId": 1,
          "address": "123 Main St"
        },
        "status": 0,
        "gmt_apply": "2023-01-01T12:00:00",
        "gmt_end": "2023-01-10T12:00:00",
        "order_item_id": 456,
        "product_item_id": 789,
        "product_id": 123,
        "shop_id": 0,
        "inArbitrated": 0,
        "arbitration_id": 0,
        "customer_id": 0
      },
      // Additional entries...
    ]
  }
}
```


2. **按售后状态查询测试用例：**

输入：

- 用户 token: "valid_token"
- 页码和每页数目
- 售后状态: 1 (例如，已处理)

期望输出：


```
{
  "errno": 0,
```

```
  "errmsg": "成功",
  "data": {
    "page": 1,
    "pageSize": 10,
    "list": [
      {
        "id": 2,
        "type": 1,
        "reason": "Incorrect item received",
        "conclusion": "Item replaced",
        "quantity": 1,
        "consignee": {
          "name": "Jane Doe",
          "mobile": "9876543210",
          "regionId": 2,
          "address": "456 Oak St"
        },
        "status": 1,
        "gmt_apply": "2023-02-01T12:00:00",
        "gmt_end": "2023-02-10T12:00:00",
        "order_item_id": 789,
        "product_item_id": 987,
        "product_id": 456,
        "shop_id": 0,
        "inArbitrated": 0,
        "arbitration_id": 0,
        "customer_id": 0
      },
      // Additional entries with status 1...
    ]
  }
}
```


3. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用不同用户的 token 查询

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. GET/aftersales/orderitems 顾客查询所有的可申请售后的订单明细

1. **默认查询测试用例:**

输入:

- 用户 token: "valid_token"
- 默认页码和每页数目

期望输出:

```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "page": 1,
    "pageSize": 10,
    "list": [
      {
        "productId": 123,
        "orderId": 456,
        "name": "Product A",
        "quantity": 2,
        "price": 20.0,
        "discountPrice": 18.0
      },
      // Additional entries...
    ]
  }
}
```

2. **按照页码和每页数目查询测试用例:**

输入:

- 用户 token: "valid_token"
- 页码: 2
- 每页数目: 5

期望输出:

```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "page": 2,
    "pageSize": 5,
    "list": [
      {
        "productId": 789,
        "orderId": 101,
        "name": "Product B",
        "quantity": 1,
```

```
            "price": 15.0,
            "discountPrice": 14.0
        },
        // Additional entries...
    ]
  }
}
```


3. **已经申请过退货或者换货的订单明细不会返回测试用例：**

输入：

- 用户 token: "valid_token"
- 已经申请过售后的订单明细

期望输出：


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "page": 1,
    "pageSize": 10,
    "list": [
      {
        "productId": 987,
        "orderId": 654,
        "name": "Product C",
        "quantity": 3,
        "price": 30.0,
        "discountPrice": 28.5
      },
      // Additional entries...
    ]
  }
}
```


4. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：


```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```


输入：

- 使用不同用户的 token 查询

期望输出：


```
{ "errno": "forbidden", "errmsg": "Forbidden access" }
```

4.GET /aftersales/{id} 顾客根据售后单 id 查询售后单信息

1. **正常流程测试用例:**

输入:

- 用户 token: "valid_token"
- 售后单 id: 123

期望输出:


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 123,
    "type": 1,
    "reason": "Product damaged",
    "conclusion": "Replacement approved",
    "quantity": 2,
    "consignee": {
      "name": "John Doe",
      "mobile": "1234567890",
      "regionId": 1,
      "address": "123 Main St"
    },
    "status": 1,
    "gmt_apply": "2023-01-01T12:00:00",
    "gmt_end": "2023-01-10T12:00:00",
    "order_item_id": 456,
    "product_item_id": 789,
    "product_id": 123,
    "shop_id": 0,
    "inArbitrated": 0,
    "arbitration_id": 0,
    "customer_id": 0
  }
}
```


2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:


```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```


输入:

- 使用不同用户的 token 查询

期望输出:


```
{ "errno": "forbidden", "errmsg": "Forbidden access" }
```


3. **售后单不存在的测试用例:**

输入：

- 用户 token: "valid_token"
- 不存在的售后单 id: 999

期望输出：

{ "errno": "not_found", "errmsg": "Aftersales not found" }

5. PUT /aftersales/{id} 顾客修改售后单信息（只能在申请态）

1. **正常流程测试用例：**

输入：

- 用户 token: "valid_token"
- 售后单 id: 123
- 请求体:

```
{
  "quantity": 3,
  "reason": "Change to a different color",
  "consignee": {
    "name": "Jane Doe",
    "mobile": "9876543210",
    "regionId": 2,
    "address": "456 Oak St"
  }
}
```

期望输出：

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用不同用户的 token 修改

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **售后单不在申请态的测试用例：**

输入：

- 用户 token: "valid_token"
- 已处理的售后单 id: 456
- 请求体:

```
{
  "quantity": 2,
  "reason": "Incorrect item received",
```

```
  "consignee": {
    "name": "John Doe",
    "mobile": "1234567890",
    "regionId": 1,
    "address": "123 Main St"
  }
}
```

期望输出：

```
{ "errno": "bad_request", "errmsg": "Aftersales can only be modified in the application state" }
```

4. **地址信息不完整的测试用例：**

输入：

- 用户 token: "valid_token"
- 售后单 id: 789
- 请求体（缺少手机号码）：

```
{
  "quantity": 1,
  "reason": "Change size",
  "consignee": {
    "name": "Alice",
    "regionId": 3,
    "address": "789 Pine St"
  }
}
```

期望输出：

```
{ "errno": "bad_request", "errmsg": "Mobile is required in consignee information" }
```

6.DELETE /aftersales/{id} 顾客取消售后

1. **正常流程测试用例：**

输入：

- 用户 token: "valid_token"
- 未售后的售后单 id: 123

期望输出：

```
{ "errno": 0, "errmsg": "成功" }
```

2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```

输入：

- 使用不同用户的 token 取消

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **取消已售后的售后单的测试用例：**

输入：

- 用户 token: "valid_token"
- 已售后的售后单 id: 456

期望输出：

{ "errno": "bad_request", "errmsg": "Aftersales can only be canceled in the not aftersales or in aftersales state" }

4. **售后单不存在的测试用例：**

输入：

- 用户 token: "valid_token"
- 不存在的售后单 id: 999

期望输出：

{ "errno": "not_found", "errmsg": "Aftersales not found" }

7.POST /aftersales/{id}/arbitrations 顾客申请售后仲裁*

1. **正常流程测试用例：**

输入：

- 用户 token: "valid_token"
- 未仲裁的售后单 id: 123
- 请求体:

{ "reason": "Seller not responding" }

期望输出：


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 456,
    "status": 2,
    "reason": "Seller not responding",
    "shop_reply": "string",
    "result": "string",
    "arbitrator": "string",
    "gmt_arbitration": "string",
    "gmt_accept": "string",
    "gmt_apply": "string",
    "gmt_reply": "string",
    "shop_id": 789,
    "aftersale_id": 123,
    "customer_id": 101
  }
```

}

2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用不同用户的 token 仲裁

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **售后单已经在仲裁中的测试用例：**

输入：

- 用户 token: "valid_token"
- 已仲裁的售后单 id: 789
- 请求体:

{ "reason": "Seller not responding" }

期望输出：

{ "errno": 705, "errmsg": "Aftersales is already in arbitration" }

4. **未提供仲裁理由的测试用例：**

输入：

- 用户 token: "valid_token"
- 未仲裁的售后单 id: 123
- 请求体（未提供 reason 字段）:

{}

期望输出：

{ "errno": "bad_request", "errmsg": "Reason is required for arbitration" }

8. DELETE /arbitration/{aid} 顾客取消仲裁*

1. **正常流程测试用例：**

输入：

- 用户 token: "valid_token"
- 未仲裁的仲裁单 id: 123

期望输出：

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用不同用户的 token 取消仲裁

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **仲裁单已仲裁的测试用例：**

输入：

- 用户 token: "valid_token"
- 已仲裁的仲裁单 id: 789

期望输出：

{ "errno": "bad_request", "errmsg": "Arbitration is already completed and cannot be canceled" }

4. **未提供仲裁单 id 的测试用例：**

输入：

- 用户 token: "valid_token"
- 请求体（未提供 aid 字段）：

jsonCopy code

{}

期望输出：

{ "errno": "bad_request", "errmsg": "Arbitration ID is required" }

9.GET /aftersales/{id}/express 顾客或商铺查询售后单对应的物流单号*

1. **正常流程测试用例：**

输入：

- 用户 token: "valid_token"
- 未退货或者换货的售后单 id: 123

期望输出：

```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "billCode": "1234567890",
    "status": 1,
    "company": "Express Company A",
    "list": [
      {
        "id": 1,
        "aftersale_id": 123,
        "express_id": 456
      },
      // Additional entries...
    ]
  }
}
```

2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用不同用户的 token 查询

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **未退货或者换货的售后单不存在的测试用例：**

输入：

- 用户 token: "valid_token"
- 不存在的售后单 id: 999

期望输出：

{ "errno": "not_found", "errmsg": "Aftersales not found" }

4. **已经退货或者换货的售后单的测试用例：**

输入：

- 用户 token: "valid_token"
- 已退货或者已换货的售后单 id: 789

期望输出：

{ "errno": "bad_request", "errmsg": "Aftersales with return or exchange cannot have an associated express order" }

10. PUT /shops/{shopid}/aftersales/{id}/confirm 商铺审核售后

1. **正常流程测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 店铺 id: 123
- 售后单 id: 456
- 请求体:

{ "confirm": true, "conclusion": "Return approved", "type": 1 }

期望输出：

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用顾客的 token 审核售后请求

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **售后单不存在的测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 店铺 id: 123
- 不存在的售后单 id: 999
- 请求体:

{ "confirm": true, "conclusion": "Return approved", "type": 1 }

期望输出：

{ "errno": "not_found", "errmsg": "Aftersales not found" }

4. **审核已处理的售后单的测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 店铺 id: 123
- 已处理的售后单 id: 789
- 请求体:

{ "confirm": true, "conclusion": "Return approved", "type": 1 }

期望输出：

{ "errno": "bad_request", "errmsg": "Aftersales has already been processed and cannot be confirmed again" }

11. PUT /shops/{shopid}/aftersales/{id}/receive 商铺验收售后商品

1. **正常流程测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 店铺 id: 123
- 售后单 id: 456
- 请求体:

{ "confirm": true, "conclusion": "Return received", "serialNo": "RMA123456" }

期望输出：

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用顾客的 token 验收售后商品

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **售后单不存在的测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 店铺 id: 123
- 不存在的售后单 id: 999
- 请求体:

{ "confirm": true, "conclusion": "Return received", "serialNo": "RMA123456" }

期望输出:

{ "errno": "not_found", "errmsg": "Aftersales not found" }

4. **审核非退换货售后单的测试用例:**

输入:

- 用户 token（商铺管理员）: "valid_shop_token"
- 店铺 id: 123
- 维修的售后单 id: 789
- 请求体:

{ "confirm": true, "conclusion": "Return received", "serialNo": "RMA123456" }

期望输出:

{ "errno": 703, "errmsg": "(id=789) Not a return or exchange aftersale" }

5. **验收已处理的售后单的测试用例:**

输入:

- 用户 token（商铺管理员）: "valid_shop_token"
- 店铺 id: 123
- 已处理的售后单 id: 789
- 请求体:

{ "confirm": true, "conclusion": "Return received", "serialNo": "RMA123456" }

期望输出:

{ "errno": "bad_request", "errmsg": "Aftersales has already been processed and cannot be received again" }

12.DELETE /shops/{shopid}/aftersale/{id}/cancel 商铺取消售后

1. **正常流程测试用例:**

输入:

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 售后单 id: 456

期望输出:

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入:

- 使用顾客的 token 取消售后

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **售后单不存在的测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 不存在的售后单 id: 999

期望输出：

{ "errno": "not_found", "errmsg": "Aftersale not found" }

4. **取消已处理的售后单的测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 已处理的售后单 id: 789

期望输出：

{ "errno": "bad_request", "errmsg": "Aftersale has already been processed and cannot be canceled" }

13. PUT /shops/{shopid}/arbitrations/{id}/response 商铺应诉仲裁*

1. **正常流程测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 仲裁单 id: 456
- 请求体:


```
{
  "reason": "Response to customer's arbitration",
  "last_Arbitration": {
    "id": 789,
    "status": 1,
    "reason": "Customer's arbitration",
    "shop_reply": "No response yet",
    "result": "Waiting for arbitration",
    "arbitrator": "",
    "gmt_arbitration": "2023-01-01T12:00:00Z",
    "gmt_accept": "2023-01-02T12:00:00Z",
    "gmt_apply": "2023-01-03T12:00:00Z",
    "gmt_reply": "",
    "shop_id": 123,
    "aftersale_id": 456,
    "customer_id": 101
```

```
  }
}
```


期望输出：


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 456,
    "status": 2,
    "reason": "Response to customer's arbitration",
    "shop_reply": "No response yet",
    "result": "Waiting for arbitration",
    "arbitrator": "",
    "gmt_arbitration": "2023-01-01T12:00:00Z",
    "gmt_accept": "2023-01-02T12:00:00Z",
    "gmt_apply": "2023-01-03T12:00:00Z",
    "gmt_reply": "2023-01-04T12:00:00Z",
    "shop_id": 123,
    "aftersale_id": 456,
    "customer_id": 101
  }
}
```


2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：


```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```


输入：

- 使用顾客的 token 应诉仲裁

期望输出：


```
{ "errno": "forbidden", "errmsg": "Forbidden access" }
```


3. **仲裁单不存在的测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 不存在的仲裁单 id: 999
- 请求体:


```
{
  "reason": "Response to customer's arbitration",
  "last_Arbitration": {
    "id": 789,
    "status": 1,
    "reason": "Customer's arbitration",
    "shop_reply": "No response yet",
```

```
    "result": "Waiting for arbitration",
    "arbitrator": "",
    "gmt_arbitration": "2023-01-01T12:00:00Z",
    "gmt_accept": "2023-01-02T12:00:00Z",
    "gmt_apply": "2023-01-03T12:00:00Z",
    "gmt_reply": "",
    "shop_id": 123,
    "aftersale_id": 456,
    "customer_id": 101
  }
}
```


期望输出：


```
{ "errno": "not_found", "errmsg": "Arbitration not found" }
```


4. **应诉已处理的仲裁单的测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 已处理的仲裁单 id: 789
- 请求体:


```
{
  "reason": "Response to customer's arbitration",
  "last_Arbitration": {
    "id": 789,
    "status": 3,
    "reason": "Customer's arbitration",
    "shop_reply": "Decision made",
    "result": "Seller wins",
    "arbitrator": "Arbitrator X",
    "gmt_arbitration": "2023-01-01T12:00:00Z",
    "gmt_accept": "2023-01-02T12:00:00Z",
    "gmt_apply": "2023-01-03T12:00:00Z",
    "gmt_reply": "2023-01-04T12:00:00Z",
    "shop_id": 123,
    "aftersale_id": 456,
    "customer_id": 101
  }
}
```


期望输出：


```
{ "errno": "bad_request", "errmsg": "Arbitration has already been processed and cannot be responded again" }
```

14. PUT /shops/{shopid}/arbitrations/{id}/accept 管理员受理仲裁单

1. **正常流程测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 无效的商铺 id: 0 (此端点要求 shopid 为 0)
- 仲裁单 id: 456
- 请求体:

{ "arbitratorId": 789, "arbitratorName": "Arbitrator X" }

期望输出：

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用商铺管理员的 token 受理仲裁单

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **商铺 id 不为 0 的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 商铺 id: 123
- 仲裁单 id: 456
- 请求体:

{ "arbitratorId": 789, "arbitratorName": "Arbitrator X" }

期望输出：

{ "errno": "bad_request", "errmsg": "Invalid shopid. Shopid must be 0 for this endpoint." }

4. **仲裁单不是申请中的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 商铺 id: 0
- 已处理的仲裁单 id: 789
- 请求体:

{ "arbitratorId": 789, "arbitratorName": "Arbitrator X" }

期望输出：

{ "errno": "bad_request", "errmsg": "Arbitration is not in the applying state and cannot be accepted" }

15. PUT /shops/{shopid}/arbitrations/{id}/close 管理员仲裁结案

1. **正常流程测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 无效的商铺 id: 0 (此端点要求 shopid 为 0)
- 仲裁单 id: 456
- 请求体:

{ "result": "Seller wins" }

期望输出：

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用商铺管理员的 token 结案仲裁单

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **商铺 id 不为 0 的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 商铺 id: 123
- 仲裁单 id: 456
- 请求体:

{ "result": "Seller wins" }

期望输出：

{ "errno": "bad_request", "errmsg": "Invalid shopid. Shopid must be 0 for this endpoint." }

4. **仲裁单不是受理态的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 无效的商铺 id: 0
- 申请中的仲裁单 id: 456
- 请求体:

{ "result": "Seller wins" }

期望输出：

{ "errno": "bad_request", "errmsg": "Arbitration is not in the accepted state and cannot be closed" }

5. **非负责仲裁员结案的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 无效的商铺 id: 0
- 受理态的仲裁单 id: 789
- 请求体:

{ "result": "Seller wins" }

期望输出：

{ "errno": "bad_request", "errmsg": "Only the assigned arbitrator can close the arbitration" }

16. GET /shops/{shopid}/aftersales 商铺或管理员查看所有售后单（可根据售后类型和状态选择）

1. **正常流程测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 开始时间: "2023-01-01"
- 结束时间: "2023-12-31"
- 页码: 1
- 每页数目: 10
- 售后类型: 1 (例如，退货)
- 售后状态: 2 (例如，已售后)

期望输出：


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "page": 1,
    "pageSize": 10,
    "list": [
      {
        "id": 456,
        "type": 1,
        "reason": "Defective product",
        "conclusion": "Refunded",
        "quantity": 1,
        "consignee": {
          "name": "John Doe",
          "mobile": "1234567890",
          "regionId": 456,
          "address": "123 Main St"
        },
        "status": 2,
        "gmt_apply": "2023-01-10T12:00:00Z",
        "gmt_end": "2023-01-15T12:00:00Z",
        "order_item_id": 789,
```

```
            "product_item_id": 101,
            "product_id": 111,
            "shop_id": 123,
            "inArbitrated": 0,
            "arbitration_id": 0,
            "customer_id": 789
        }
    ]
  }
}
```


2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用顾客的 token 查看商铺的售后单

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **店铺 id 不为 0 的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 商铺 id: 123
- 其他查询条件...

期望输出：

{ "errno": "bad_request", "errmsg": "Invalid shopid. Shopid must be 0 for this endpoint." }

4. **无查询条件的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 商铺 id: 0
- 未提供其他查询条件...

期望输出：


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "page": 1,
    "pageSize": 10,
    "list": [
      {
        "id": 456,
        "type": 1,
        "reason": "Defective product",
        "conclusion": "Refunded",
```

```
            "quantity": 1,
            "consignee": {
              "name": "John Doe",
              "mobile": "1234567890",
              "regionId": 456,
              "address": "123 Main St"
            },
            "status": 2,
            "gmt_apply": "2023-01-10T12:00:00Z",
            "gmt_end": "2023-01-15T12:00:00Z",
            "order_item_id": 789,
            "product_item_id": 101,
            "product_id": 111,
            "shop_id": 123,
            "inArbitrated": 0,
            "arbitration_id": 0,
            "customer_id": 789
          },
          // Additional aftersales items...
        ]
    }
}
```


17.GET /shops/{shopid}/aftersales/{id} 商铺或管理员根据售后单 id 查询售后单信息

1. **正常流程测试用例:**

输入:

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 售后单 id: 456

期望输出:


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 456,
    "type": 1,
    "reason": "Defective product",
    "conclusion": "Refunded",
    "quantity": 1,
    "consignee": {
      "name": "John Doe",
      "mobile": "1234567890",
```

```
        "regionId": 456,
        "address": "123 Main St"
      },
      "status": 2,
      "gmt_apply": "2023-01-10T12:00:00Z",
      "gmt_end": "2023-01-15T12:00:00Z",
      "order_item_id": 789,
      "product_item_id": 101,
      "product_id": 111,
      "shop_id": 123,
      "inArbitrated": 0,
      "arbitration_id": 0,
      "customer_id": 789
    }
}
```


2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入：

- 使用顾客的 token 查看商铺的售后单

期望输出：

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **店铺 id 不为 0 的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 商铺 id: 123
- 其他查询条件...

期望输出：

{ "errno": "bad_request", "errmsg": "Invalid shopid. Shopid must be 0 for this endpoint." }

4. **售后单 id 不存在的测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 不存在的售后单 id: 999

期望输出：

{ "errno": "not_found", "errmsg": "Aftersale with id 999 not found" }

18. GET /shops/{shopid}/arbitrations 商铺或管理员查询顾客申请的仲裁单信息（根据仲裁单状态）

1. **正常流程测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 页码: 1
- 每页数目: 10

期望输出：


```
{
  "errno": 0,
  "errmsg": "成功",
  "page": 1,
  "pageSize": 10,
  "data": [
    {
      "id": 789,
      "status": 1,
      "reason": "Product not as described",
      "shop_reply": "We disagree with the customer's claim",
      "result": "",
      "arbitrator": "",
      "gmt_arbitration": "",
      "gmt_accept": "",
      "gmt_apply": "2023-02-01T12:00:00Z",
      "gmt_reply": "",
      "shop_id": 123,
      "aftersale_id": 456,
      "customer_id": 789
    }
  ]
}
```


2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：


```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```


输入：

- 使用顾客的 token 查看商铺的仲裁单

期望输出：


```
{ "errno": "forbidden", "errmsg": "Forbidden access" }
```


3. **店铺 id 不为 0 的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 商铺 id: 123
- 其他查询条件...

期望输出：

{ "errno": "bad_request", "errmsg": "Invalid shopid. Shopid must be 0 for this endpoint." }

4. **无查询条件的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 商铺 id: 0
- 未提供其他查询条件...

期望输出：


```
{
  "errno": 0,
  "errmsg": "成功",
  "page": 1,
  "pageSize": 10,
  "data": [
    {
      "id": 789,
      "status": 1,
      "reason": "Product not as described",
      "shop_reply": "We disagree with the customer's claim",
      "result": "",
      "arbitrator": "",
      "gmt_arbitration": "",
      "gmt_accept": "",
      "gmt_apply": "2023-02-01T12:00:00Z",
      "gmt_reply": "",
      "shop_id": 123,
      "aftersale_id": 456,
      "customer_id": 789
    },
    // ...其他数据
  ]
}
```


19.GET /shops/{shopid}/arbitrations/{id} 商铺或管理员查询仲裁单详情信息

1. **正常流程测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123

- 仲裁单 id: 789

期望输出：


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 789,
    "status": 1,
    "reason": "Product not as described",
    "shop_reply": "We disagree with the customer's claim",
    "result": "",
    "arbitrator": "",
    "gmt_arbitration": "",
    "gmt_accept": "",
    "gmt_apply": "2023-02-01T12:00:00Z",
    "gmt_reply": "",
    "shop_id": 123,
    "aftersale_id": 456,
    "customer_id": 789
  }
}
```


2. **权限测试用例：**

输入：

- 不提供用户 token

期望输出：


```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```


输入：

- 使用顾客的 token 查看商铺的仲裁单

期望输出：


```
{ "errno": "forbidden", "errmsg": "Forbidden access" }
```


3. **店铺 id 不为 0 的测试用例：**

输入：

- 用户 token（管理员）: "valid_admin_token"
- 商铺 id: 123
- 仲裁单 id: 789
- 其他查询条件...

期望输出：


```
{ "errno": "bad_request", "errmsg": "Invalid shopid. Shopid must be 0 for this endpoint." }
```


4. **仲裁单 id 不存在的测试用例：**

输入：

- 用户 token（商铺管理员）: "valid_shop_token"
- 商铺 id: 123
- 不存在的仲裁单 id: 999

期望输出：

{ "errno": "not_found", "errmsg": "Arbitration with id 999 not found" }

2.3.3 服务模块测试

测试内容：

编号	API	功能	测试用例	描述
1	POST /shops/{did}/products/{id}/maintainers/{mid}/service/{sid}	商户选择服务商提供的地区服务，与具体商品关联	TestService1	成功选择服务
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务时间冲突
2	PUT /shops/{did}/product_services/{id}	商户修改服务商在某个地区的商品服务*(暂停：从有效变为无效；恢复：从无效变为有效)	TestService1	成功修改
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务时间冲突
			TestService5	服务暂停和恢复
3	DELETE /shops/{did}/product_services/{id}	商户取消服务商在某个地区的商品服务*	TestService1	成功取消
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务不存在
4	GET /shops/{did}/products/{id}/region/{rid}	商户通过商品和地区找到服务商	TestService1	成功查找
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务商不存在
			TestService5	地区 id 不存在
5	POST /internal/shops/{did}/products/{id}/region/{rid}/serviceOrders	创建服务单	TestService1	成功创建
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务商不存在
			TestService5	地区 id 不存在
6	GET /shops/{did}/serviceOrder/	店铺根据服务单 id	TestService1	成功查询
			TestService2	用户无权限

	{id}	查询服务单信息	TestService3	参数缺失
			TestService4	服务单不存在
7	PUT /adminusers/{aid}/maintainers/{mid}/account	平台管理员审核服务商开户*	TestService1	成功审核开户
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务商不存在
8	PUT /adminusers/{aid}/maintainers/{mid}/modify	平台管理员审核服务商变更*	TestService1	成功审核变更
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务商不存在
9	PUT /adminusers/{aid}/services/{sid}/service	平台管理员审核服务*	TestService1	成功审核服务
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务不存在
10	GET /maintainers/{mid}/serviceOrders/{id}	服务商根据服务单id查询服务单信息*	TestService1	成功查询服务单
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务单不存在
11	PUT /maintainers/{mid}/serviceOrders/{id}/receive	服务商收到寄出商品	TestService1	成功收到
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务单不存在
			TestService5	无效的请求体
12	PUT /maintainers/{mid}/serviceOrder/{id}/finish	服务商完成服务单	TestService1	成功完成
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务单不存在
			TestService5	无效的请求体
13	PUT /maintainers/{mid}/serviceOrders/{id}/cancelOrder	服务商取消服务单	TestService1	成功取消
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	服务单不存在
			TestService5	无效的请求体
14	POST /maintainers/{mid}/categories/{id}/service	服务商定义在某个或多个地区为某种商品的服务*	TestService1	成功定义
			TestService2	用户无权限
			TestService3	参数缺失
			TestService4	无效的请求体
15	PUT /maintainers/{did}/services	服务商修改某服务	TestService1	成功修改

	/{id}/changeRegion	的服务范围(把 region 放 body 里，responses 结果详情)*	TestService2	用户无权限
			TestService3	参数缺失
			TestService4	无效的请求体

2.3.4 服务模块测试用例

1.POST /shops/{did}/products/{id}/maintainers/{mid}/service/{sid} 商户选择服务商提供的地区服务，与具体商品关联

1. **正常流程测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 商品 id: 456
- 服务商 id: 789
- 服务 id: 987
- 请求体:

```
{
  "beginTime": "2023-01-01T00:00:00Z",
  "endTime": "2023-12-31T23:59:59Z",
  "invalid": 0,
  "priority": 1000
}
```

期望输出:

```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 1234,
    "name": "Service for Product X",
    "product_id": 456,
    "service_id": 987,
    "beginTime": "2023-01-01T00:00:00Z",
    "endTime": "2023-12-31T23:59:59Z",
    "priority": 1000,
    "status": 0
```

```
  }
}
```

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入:

- 使用顾客的 token 进行请求

期望输出:

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **参数缺失测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 缺失商品 id、服务商 id、服务 id 等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameters: id, mid, sid" }

4. **服务时间冲突的测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 商品 id: 456
- 服务商 id: 789
- 服务 id: 987
- 已存在相同时间范围的服务

期望输出:

{ "errno": "conflict", "errmsg": "Service time conflict. Please choose a different time range." }

2.PUT /shops/{did}/product_services/{id} 商户修改服务商在某个地区的商品服务*(暂停：从有效变为无效；恢复：从无效变为有效)

1. **正常流程测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 商品服务 id: 456
- 请求体:

```
{
  "beginTime": "2023-02-01T00:00:00Z",
  "endTime": "2023-02-28T23:59:59Z",
```

```
  "invalid": 0,
  "priority": 1200
}
```

期望输出:

```
{ "errno": 0, "errmsg": "成功" }
```

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```

输入:

- 使用顾客的 token 进行请求

期望输出:

```
{ "errno": "forbidden", "errmsg": "Forbidden access" }
```

3. **参数缺失测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 缺失商品服务 id 等参数

期望输出:

```
{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }
```

4. **服务时间冲突的测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 商品服务 id: 456
- 已存在相同时间范围的服务

期望输出:

```
{ "errno": "conflict", "errmsg": "Service time conflict. Please choose a different time range." }
```

5. **服务暂停和恢复的测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 商品服务 id: 456
- 请求体:

```
{ "invalid": 1 }
```

期望输出:

```
{ "errno": 0, "errmsg": "成功" }
```

3.DELETE /shops/{did}/product_services/{id} 商户取消服务商在某个地区的商品服务

*****

1. **正常流程测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 商品服务 id: 456

期望输出:

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入:

- 使用顾客的 token 进行请求

期望输出:

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **参数缺失测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 缺失商品服务 id 等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }

4. **服务不存在的测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 不存在的商品服务 id: 789

期望输出:

{ "errno": "not_found", "errmsg": "Service not found" }

4.GET /shops/{did}/products/{id}/region/{rid} 商户通过商品和地区找到服务商

1. **正常流程测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 商品 id: 456
- 地区 id: 789

期望输出:


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "page": 1,
    "pageSize": 10,
    "list": [
      {
        "id": 789,
        "name": "Service Provider 1",
        "product_id": 456,
        "service_id": 123,
        "beginTime": "2023-02-01T00:00:00Z",
        "endTime": "2023-02-28T23:59:59Z",
        "priority": 1000,
        "status": 1
      },
      {
        "id": 790,
        "name": "Service Provider 2",
        "product_id": 456,
        "service_id": 124,
        "beginTime": "2023-02-01T00:00:00Z",
        "endTime": "2023-02-28T23:59:59Z",
        "priority": 1200,
        "status": 1
      }
    ]
  }
}
```


2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:


```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```


输入:

- 使用顾客的 token 进行请求

期望输出:


```
{ "errno": "forbidden", "errmsg": "Forbidden access" }
```


3. **参数缺失测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 缺失商品 id 等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }

4. **服务商不存在的测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 不存在的商品 id: 999
- 地区 id: 789

期望输出:

{ "errno": 0, "errmsg": "成功", "data": { "page": 1, "pageSize": 10, "list": [] } }

5. **地区 id 不存在的测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 商品 id: 456
- 不存在的地区 id: 999

期望输出:

{ "errno": 0, "errmsg": "成功", "data": { "page": 1, "pageSize": 10, "list": [] } }

5.POST /internal/shops/{did}/products/{id}/region/{rid}/serviceOrders 创建服务单

1. **正常流程测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 商品 id: 456
- 地区 id: 789
- 请求体:

```
{
  "type": 0,
  "consignee": {
    "name": "John Doe",
    "mobile": "1234567890",
    "regionId": 789,
    "address": "123 Main Street"
  }
}
```

期望输出:

```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 789,
    "type": 0,
    "address": "123 Main Street",
    "consignee": "John Doe",
    "mobile": "1234567890",
    "serviceSn": 123456,
    "status": "Processing",
    "description": "",
    "maintainerName": "",
    "maintainerMobile": "",
    "result": "",
    "shop_id": 123,
    "customer_id": 456,
    "service_provider_id": 789,
    "service_id": 789,
    "order_item_id": 789
  }
}
```


2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入:

- 使用顾客的 token 进行请求

期望输出:

{ "errno": "forbidden", "errmsg": "Forbidden access" }

3. **参数缺失测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 缺失商品 id 等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }

4. **服务商不存在的测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 不存在的商品 id: 999
- 地区 id: 789
- 请求体:

{ "type": 0, "consignee": { "name": "John Doe", "mobile": "1234567890", "regionId": 789, "address": "123 Main Street" } }

期望输出:

{ "errno": "not_found", "errmsg": "Product not found" }

5. **地区 id 不存在的测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 商品 id: 456
- 不存在的地区 id: 999
- 请求体:

{ "type": 0, "consignee": { "name": "John Doe", "mobile": "1234567890", "regionId": 999, "address": "123 Main Street" } }

期望输出:

{ "errno": "not_found", "errmsg": "Region not found" }

6.GET /shops/{did}/serviceOrder/{id} 店铺根据服务单 id 查询服务单信息

1. **正常流程测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 服务单 id: 789

期望输出:


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 789,
    "type": 0,
    "address": "123 Main Street",
    "consignee": "John Doe",
    "mobile": "1234567890",
    "serviceSn": 123456,
    "status": "Processing",
    "description": "",
    "maintainerName": "",
    "maintainerMobile": "",
    "result": "",
    "shop_id": 123,
    "customer_id": 456,
    "service_provider_id": 789,
```

```
    "service_id": 789,
    "order_item_id": 789
  }
}
```

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```

输入:

- 使用顾客的 token 进行请求

期望输出:

```
{ "errno": "forbidden", "errmsg": "Forbidden access" }
```

3. **参数缺失测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 缺失服务单 id 等参数

期望输出:

```
{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }
```

4. **服务单不存在的测试用例:**

输入:

- 用户 token（商户管理员）: "valid_merchant_token"
- 商铺 id: 123
- 不存在的服务单 id: 999

期望输出:

```
{ "errno": "not_found", "errmsg": "Service order not found" }
```

7.PUT /adminusers/{aid}/maintainers/{mid}/account 平台管理员审核服务商开户*

1. **正常流程测试用例:**

输入:

- 用户 token（平台管理员）: "valid_platform_admin_token"
- 管理员 id: 123
- 服务商 id: 456

期望输出:

```
{ "errno": 0, "errmsg": "成功" }
```

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```

输入:

- 使用商户管理员的 token 进行请求

期望输出:

{ "errno": "forbidden", "errmsg": "Forbidden access"

3. **参数缺失测试用例:**

输入:

- 用户 token（平台管理员）: "valid_platform_admin_token"
- 缺失服务商 id 等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameter: mid" }

4. **服务商不存在的测试用例:**

输入:

- 用户 token（平台管理员）: "valid_platform_admin_token"
- 管理员 id: 123
- 不存在的服务商 id: 999

期望输出:

{ "errno": "not_found", "errmsg": "Maintainer not found" }

8.PUT /adminusers/{aid}/maintainers/{mid}/modify 平台管理员审核服务商变更*

1. **正常流程测试用例:**

输入:

- 用户 token（平台管理员）: "valid_platform_admin_token"
- 管理员 id: 123
- 服务商 id: 456

期望输出:

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入:

- 使用商户管理员的 token 进行请求

期望输出:

{ "errno": "forbidden", "errmsg": "Forbidden access"

3. **参数缺失测试用例:**

输入:

- 用户 token（平台管理员）: "valid_platform_admin_token"
- 缺失服务商 id 等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameter: mid" }

4. **服务商不存在的测试用例:**

输入:

- 用户 token（平台管理员）: "valid_platform_admin_token"
- 管理员 id: 123
- 不存在的服务商 id: 999

期望输出:

{ "errno": "not_found", "errmsg": "Maintainer not found" }

9.PUT /adminusers/{aid}/services/{sid}/service 平台管理员审核服务*

1. **正常流程测试用例:**

输入:

- 用户 token（平台管理员）: "valid_platform_admin_token"
- 管理员 id: 123
- 服务 id: 456

期望输出:

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入:

- 使用服务商管理员的 token 进行请求

期望输出:

{ "errno": "forbidden", "errmsg": "Forbidden access"

3. **参数缺失测试用例:**

输入:

- 用户 token（平台管理员）: "valid_platform_admin_token"
- 缺失服务 id 等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameter: sid" }

4. **服务不存在的测试用例:**

输入:

- 用户 token（平台管理员）: "valid_platform_admin_token"
- 管理员 id: 123
- 不存在的服务 id: 999

期望输出:

{ "errno": "not_found", "errmsg": "Service not found" }

10.GET /maintainers/{mid}/serviceOrders/{id}服务商根据服务单 id 查询服务单信息*

1. **正常流程测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务单 id: "serviceOrder456"

期望输出:

```
{
  "errno": 0,
  "errmsg": "string",
  "data": {
    "id": 0,
    "type": 0,
    "address": "string",
    "consignee": "string",
    "mobile": "string",
    "serviceSn": 0,
    "status": "string",
    "description": "string",
    "maintainerName": "string",
    "maintainerMobile": "string",
    "result": "string",
    "shop_id": 0,
    "customer_id": 0,
    "service_provider_id": 0,
    "service_id": 0,
    "oeder_item_id": 0
  }
}
```

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```

输入:

- 使用平台管理员的 token 进行请求

期望输出:

```
{ "errno": "forbidden", "errmsg": "Forbidden access"
```

3. **参数缺失测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 缺失服务单 id 等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }

4. **服务单不存在的测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 不存在的服务单 id: "serviceOrder999"

期望输出:

{ "errno": "not_found", "errmsg": "Service order not found" }

11.PUT /maintainers/{mid}/serviceOrders/{id}/receive 服务商收到寄出商品

1. **正常流程测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务单 id: "serviceOrder456"
- 请求体:

jsonCopy code

{ "accepted": true, "result": "Received the products in good condition." }

期望输出:

jsonCopy code

{ "errno": {}, "errmsg": "string" }

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入:

- 使用平台管理员的 token 进行请求

期望输出:

{ "errno": "forbidden", "errmsg": "Forbidden access"

3. **参数缺失测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务单 id 缺失等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }

4. **服务单不存在的测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"

- 不存在的服务单 id: "serviceOrder999"
- 请求体:

{ "accepted": true, "result": "Received the products in good condition." }

期望输出:

{ "errno": "not_found", "errmsg": "Service order not found" }

5. **无效的请求体测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务单 id: "serviceOrder456"
- 无效的请求体（缺失 accepted 字段）:

{ "result": "Received the products in good condition." }

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required field: accepted" }

12.PUT /maintainers/{mid}/serviceOrder/{id}/finish 服务商完成服务单

1. **正常流程测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务单 id: "serviceOrder456"
- 请求体:

{ "result": "Service completed successfully." }

期望输出:

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入:

- 使用平台管理员的 token 进行请求

期望输出:

{ "errno": "forbidden", "errmsg": "Forbidden access"

3. **参数缺失测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务单 id 缺失等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }

4. **服务单不存在的测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 不存在的服务单 id: "serviceOrder999"
- 请求体:

```
{ "result": "Service completed successfully." }
```

期望输出:

```
{ "errno": "not_found", "errmsg": "Service order not found" }
```

5. **无效的请求体测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务单 id: "serviceOrder456"
- 无效的请求体（缺失 result 字段）:

期望输出:

```
{ "errno": "bad_request", "errmsg": "Missing required field: result" }
```

13.PUT /maintainers/{mid}/serviceOrders/{id}/cancelOrder 服务商取消服务单

1. **正常流程测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务单 id: "serviceOrder456"
- 请求体:

```
{ "result": "Service canceled by the maintainer." }
```

期望输出:

```
{ "errno": 0, "errmsg": "成功" }
```

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```

输入:

- 使用平台管理员的 token 进行请求

期望输出:

```
{ "errno": "forbidden", "errmsg": "Forbidden access"
```

3. **参数缺失测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"

- 服务单 id 缺失等参数

期望输出:

```
{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }
```

4. **服务单不存在的测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 不存在的服务单 id: "serviceOrder999"
- 请求体:

```
{ "result": "Service canceled by the maintainer." }
```

期望输出:

```
{ "errno": "not_found", "errmsg": "Service order not found" }
```

5. **无效的请求体测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务单 id: "serviceOrder456"
- 无效的请求体（缺失 result 字段）:

期望输出:

```
{ "errno": "bad_request", "errmsg": "Missing required field: result" }
```

14.POST /maintainers/{mid}/categories/{id}/service 服务商定义在某个或多个地区为某种商品的服务*

1. **正常流程测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 商品类别 id: "category456"
- 请求体:

```
{
  "name": "Express Delivery",
  "description": "Fast and reliable express delivery service",
  "type": 1,
  "region": {
    "id": 123,
    "name": "City A"
  }
}
```

期望输出:

```
{
```

```
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 789,
    "name": "Express Delivery",
    "description": "Fast and reliable express delivery service",
    "type": 1,
    "service_area": {
      "id": 123,
      "name": "City A"
    }
  }
}
```


2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:


```
{ "errno": "unauthorized", "errmsg": "Unauthorized access" }
```


输入:

- 使用普通用户的 token 进行请求

期望输出:


```
{ "errno": "forbidden", "errmsg": "Forbidden access"
```


3. **参数缺失测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 商品类别 id 缺失等参数

期望输出:


```
{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }
```


4. **无效的请求体测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 商品类别 id: "category456"
- 无效的请求体（缺失 name 字段）:


```
{
  "description": "Fast and reliable express delivery service",
  "type": 1,
  "region": {
    "id": 123,
    "name": "City A"
  }
}
```


期望输出:

{ "errno": "bad_request", "errmsg": "Missing required field: name" }

15.PUT /maintainers/{did}/services/{id}/changeRegion 服务商修改某服务的服务范围(region 放 body 里，responses 结果详情)*

1. **正常流程测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务 id: "service789"
- 请求体:

{ "id": 456, "name": "New Region" }

期望输出:

{ "errno": 0, "errmsg": "成功" }

2. **权限测试用例:**

输入:

- 不提供用户 token

期望输出:

{ "errno": "unauthorized", "errmsg": "Unauthorized access" }

输入:

- 使用普通用户的 token 进行请求

期望输出:

{ "errno": "forbidden", "errmsg": "Forbidden access"

3. **参数缺失测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务 id: "service789"
- 请求体缺失等参数

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required parameter: id" }

4. **无效的请求体测试用例:**

输入:

- 用户 token（服务商）: "valid_maintainer_token"
- 服务商 id: "maintainer123"
- 服务 id: "service789"
- 无效的请求体（缺失 name 字段）:

{ "id": 456 }

期望输出:

{ "errno": "bad_request", "errmsg": "Missing required field: name" }

3. 软件需求测试结论

售后模块：

编号	API	测试结论
1	POST /order/{oid}/orderitem/{id}/aftersales	通过测试，功能正常
2	GET /aftersales	通过测试，功能正常
3	GET /aftersales/orderitems	通过测试，功能正常
4	GET /aftersales/{id}	通过测试，功能正常
5	PUT /aftersales/{id}	通过测试，功能正常
6	DELETE /aftersales/{id}	通过测试，功能正常
7	POST /aftersales/{id}/arbitrations	通过测试，功能正常
8	DELETE /arbitration/{aid}	通过测试，功能正常
9	GET /aftersales/{id}/express	通过测试，功能正常
10	PUT /shops/{shopid}/aftersales/{id}/confirm	通过测试，功能正常
11	PUT /shops/{shopid}/aftersales/{id}/receive	通过测试，功能正常
12	DELETE /shops/{shopid}/aftersale/{id}/cancel	通过测试，功能正常
13	PUT /shops/{shopid}/arbitrations/{id}/response	通过测试，功能正常
14	PUT /shops/{shopid}/arbitrations/{id}/accept	通过测试，功能正常
15	PUT /shops/{shopid}/arbitrations/{id}/close	通过测试，功能正常
16	GET /shops/{shopid}/aftersales	通过测试，功能正常
17	GET /shops/{shopid}/aftersales/{id}	通过测试，功能正常

18	GET /shops/{shopid}/arbitrations	通过测试，功能正常
19	GET /shops/{shopid}/arbitrations/{id}	通过测试，功能正常

服务模块：

编号	API	测试结论
1	POST /shops/{did}/products/{id}/maintainers/{mid}/service/{sid}	通过测试，功能正常
2	PUT /shops/{did}/product_services/{id}	通过测试，功能正常
3	DELETE /shops/{did}/product_services/{id}	通过测试，功能正常
4	GET /shops/{did}/products/{id}/region/{rid}	通过测试，功能正常
5	POST /internal/shops/{did}/products/{id}/region/{rid}/serviceOrders	通过测试，功能正常
6	GET /shops/{did}/serviceOrder/{id}	通过测试，功能正常
7	PUT /adminusers/{aid}/maintainers/{mid}/account	通过测试，功能正常
8	PUT /adminusers/{aid}/maintainers/{mid}/modify	通过测试，功能正常
9	PUT /adminusers/{aid}/services/{sid}/service	通过测试，功能正常
10	GET /maintainers/{mid}/serviceOrders/{id}	通过测试，功能正常
11	PUT /maintainers/{mid}/serviceOrders/{id}/receive	通过测试，功能正常
12	PUT /maintainers/{mid}/serviceOrder/{id}/finish	通过测试，功能正常
13	PUT /maintainers/{mid}/serviceOrders/{id}/cancelOrder	通过测试，功能正常
14	POST /maintainers/{mid}/categories/{id}/service	通过测试，功能正常
15	PUT /maintainers/{did}/services/{id}/changeRegion	通过测试，功能正常

4. 评价

4.1 软件能力

通过编写测试用例对各模块，各 API 进行测试，可以验证程序在处理正常逻辑，边界值处理，内部数据结构，出错的错误处理方面是否能正确执行，得到预期结果；可以验证程序在性能上能否达到要求，同时，软件具有良好的用户体验和易用性，能够为用户提供良好的购物体验和服务。

4.2 缺陷和限制

4.2.1 局限性和缺陷：

a. 在一些极端情况下，如并发访问、异常数据输入等情况下，软件可能出现性能下降或崩溃等问题。这是由于在极端情况下，软件所面对的压力和负载超出了系统设计和测试的范围，导致软件无法正常工作。为减少这类局限性和缺陷，可以考虑加强性能测试和容错测试，模拟更多的场景和数据情况，提高软件的容错能力和稳定性。

b. 软件的安全性还有待进一步加强，尤其是在支付系统和用户信息管理方面，需要更加严格的权限控制和防护机制。这是由于随着互联网应用的普及，黑客和攻击者的技术手段变得越来越复杂和隐蔽，要求软件具备更高的安全性和防护能力。为解决这类问题，可以考虑引入更加严格的安全审计机制和漏洞扫描工具，及时发现和修复潜在的安全问题，同时加强用户权限控制和密码强度要求等措施。

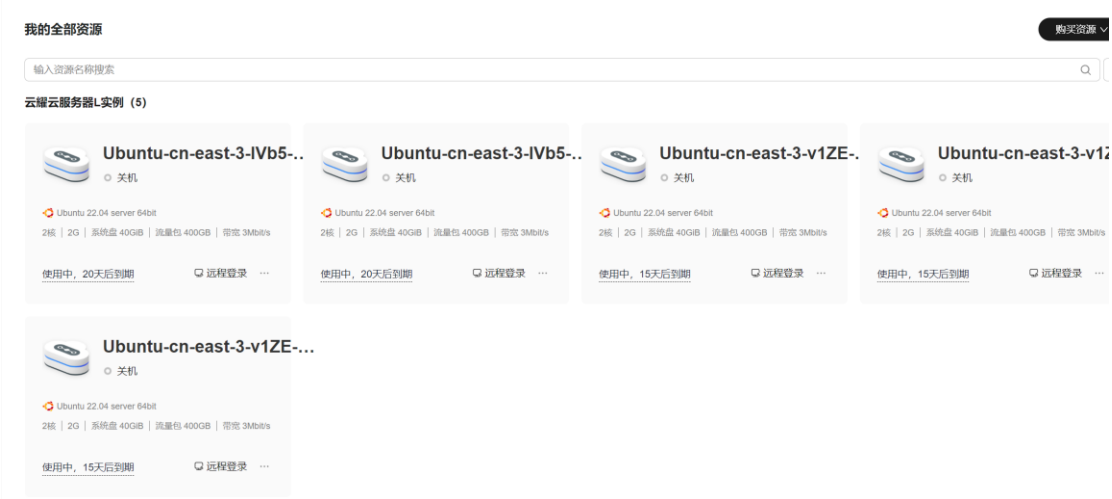
c. 在一些特殊场景下，如跨平台兼容性、国际化支持等方面，软件还存在一些局限性和不足。这是由于不同平台和语言环境之间的差异导致软件在部分情况下无法适应。为解决这类问题，可以考虑增加更多的测试用例和功能模块，以覆盖更多的场景和语言环境，同时对软件进行本地化和国际化的优化，提高软件的适配性和兼容性。

4.2.2 限制：

时间限制：设计开发时间共一学期

软件限制：使用 MySQL 作为数据库存储工具，Java 作为编程语言

硬件限制：在 Linux Ubuntu 服务器上运行，服务器数量有限，提供的性能支持有限



4.3 建议

a. **软件性能优化和稳定性测试：**

进行负载测试，模拟高并发情况下的用户访问和交易操作，评估系统的性能表现。

进行压力测试，验证系统在长时间运行和大量数据处理的情况下是否稳定，并检查是否存在内存泄漏或性能瓶颈等问题。

进行持续集成和自动化测试，确保每次代码提交后都进行自动化测试，及时发现和解决潜在的问题。

进行异常情况测试，如网络中断、服务器故障等，评估系统在异常情况下的容错能力和恢复能力。

b. **安全性测试：**

进行身份验证测试，验证用户身份认证的准确性和安全性。

进行数据加密测试，验证数据在传输和存储过程中是否得到适当的加密保护。

进行安全审计测试，检查系统是否具备完善的日志记录和审计功能，以便追踪和

分析安全事件。

进行漏洞扫描和渗透测试，发现系统中的潜在漏洞和安全弱点，并提供修复建议。

5. 跨平台兼容性和国际化支持测试：

进行跨平台兼容性测试，验证软件在不同操作系统、浏览器和设备上的兼容性。

进行国际化测试，检查软件是否能够正确处理不同语言、时区和地域设置，并确保界面和文本显示的适配性。

进行本地化测试，验证软件是否符合当地的法规要求和用户习惯。

4.4 测试结论

经过测试，软件在大部分需求上都能够满足预期要求，但仍然存在一些局限性和不足。建议在后续开发过程中，继续完善和优化软件功能和性能，提高用户体验和安全性。