软件测试大纲

组　名	图灵基组
组　别	2-7 组
学　院	信息学院
专　业	软件工程
成　员	吴乐言、苏一涵、张瀚文、庄子鲲、张愔
日　期	2025 年 12 月 27 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。

1.3 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

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。

2.2 测试机构和人员

测试机构：2-7 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 /orderitems/{id}/aftersales	顾客提交售后申请	TestAftersale1	成功提交售后
			TestAftersale2	申请信息无效（缺字段/数量≤0/type非法）
			TestAftersale3	信息越界（数量过大、字符串超长）
			TestAftersale4	用户无权限（token 越权/无 token）
2	GET /shops/{id}/aftersales	店铺查询售后列表（可按类型、状态、时间筛选）	TestAftersale5	默认成功查询
			TestAftersale6	根据售后状态成功查询
			TestAftersale7	用户无权限（店铺越权）
3	POST /shops/{id}/aftersales/{afterSaleId}/confirm	店铺审核售后（同意/拒绝）	TestAftersale8	成功同意审核（状态流转，触发远端）
			TestAftersale9	成功拒绝审核（状态=拒绝，不触发远端）
			TestAftersale10	用户无权限（店铺越权）
			TestAftersale11	状态不符/参数缺失（confirm 缺失或非待审核状态）
4	PATCH /aftersales/{afterSaleId}/cancel	取消售后	TestAftersale12	成功取消（可取消状态，生成或透传 requestId）
			TestAftersale13	已取消幂等（changed=false，不落库）
			TestAftersale14	用户无权限/操作者越权
			TestAftersale15	状态不允许取消（如状态=2）
			TestAftersale16	operatorType 非法或参数缺失

2.3.2 售后模块测试用例

1.POST /orderitems/{id}/aftersales 顾客提交售后申请

1.1 正常流程测试用例

输入:


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


用户 token: "valid_token"（合法用户，departId 对应店铺，userLevel=1）

订单明细 id（id）: 456

请求体:

输出:


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 1,
    "aftersaleSn": "生成的售后编号",
    "type": 1,
    "reason": "Product damaged",
    "conclusion": "处理中",
    "quantity": 2,
    "status": 0
  }
}
```


说明：实际返回不含 product_id 等字段；aftersaleSn 为系统生成；status=0 为新建。

1.2 异常流程测试用例

输入:


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

无效用户 token: "invalid_token"

订单明细 id: -1

不完整的请求体（缺少 consignee/type）：

输出:

{ "errno": 503, "errmsg": "token invalid or expired" }

或字段校验失败时：

{ "errno": 503, "errmsg": "type不能为空" }

1.3 边界值测试用例

输入:

最小/非法 id: 0

最大 id: 2147483647

请求体边界：quantity=1，reason 空/超长，mobile 非法长度等。

输出:

非法 id 期望资源不存在或越界错误；字段越界期望 FIELD_NOTVALID。

1.4 权限测试用例

输入:

不提供用户 token

输出:

{ "errno": 401, "errmsg": "Unauthorized access" }

输入:

使用不同用户的 token 访问不属于其的 orderItem

输出:

{ "errno": 503, "errmsg": "RESOURCE_ID_OUTSCOPE" }

2.1 正常流程测试用例

输入:


```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "page": 1,
    "pageSize": 10,
    "list": [
      {
        "id": 1001,
        "aftersaleSn": "SN123",
        "type": 0,
        "status": 1,
        "quantity": 1
      }
    ]
  }
}
```


店铺管理员 token: "shop_token"

店铺 id（id）: 1

查询参数：page=1&pageSize=10&type=0&state=1&beginTime=2025-01-01 00:00:00&endTime=2025-12-31 23:59:59

输出:

{ "errno": 0, "errmsg": "成功 ", "data": { "page": 1, "pageSize": 10, "list": [{ "id": 1001, "aftersaleSn": " SN123", "type": 0, "status": 1, "quantity": 1 }] }}

说明：列表按 gmtCreate 倒序；当筛选条件存在时，返回记录均满足 type/status/时间窗。

2.2 异常流程测试用例

输入:

{ "errno": 400, "errmsg": "invalid time format" }

非法时间格式：beginTime=bad-time

其他参数同正常

输出:

{ "errno": 400, "errmsg": "invalid time format" }

2.3 边界值测试用例

输入:

页码/页大小边界：page=1&pageSize=1；pageSize 超过上限（如 >50）

时间窗为空（不传 beginTime/endTime）

输出:

合法边界返回 200 且 list 条数符合 pageSize

超上限 pageSize 期望被截断或报 400（根据配置）；若有最大值校验则返回 FIELD_NOTVALID

2.4 权限测试用例

输入:

不提供 token

店铺 token 与路径 shopId 不一致（departId=2，路径 id=1）

输出:

{ "errno": 401, "errmsg": "Unauthorized access" }

（若网关未放行匿名）

{ "errno": 503, "errmsg": "RESOURCE_ID_OUTSCOPE" }

（越权访问时）

3. POST /shops/{id}/aftersales/{afterSaleId}/confirm 店铺审核售后

3.1 正常流程测试用例（同意）

输入:

{

"confirm": true,

"conclusion": "agree",

"type": 0

}

店铺管理员 token: "shop_token"（departId=店铺 id）

店铺 id: 1

售后单 id: 2001（状态=0，新建，shopId=1，type=0 退货）

请求体:

{ "confirm": true, "conclusion": "agree", "type": 0}

输出:

{
"errno": 0,
"errmsg": "成功",
"data": {
"afterSaleId": 2001,
"currentStatus": 2, // 退货：待用户退货
"type": 0,
"oldStatus": 0,
"conclusion": "agree"
}
}

说明：type=1（换货）期望 currentStatus=5；type=2（维修）期望 currentStatus=3。远端调用按类型触发对应运单/服务单。

3.2 正常流程测试用例（拒绝）

输入:

{ "confirm": false, "conclusion": "reject" }

同上，状态=0

请求体:

{ "confirm": false, "conclusion": "reject" }

输出:

{ "errno": 0, "data": { "afterSaleId": 2001, "currentStatus": 7, // 拒绝 "conclusion": "reject" }}

说明：拒绝不触发远端调用。

3.3 异常流程测试用例

售后不存在：afterSaleId 不存在 → errno=RESOURCE_ID_NOTEXIST

状态非法（非 0）：如 status=2 → errno=STATENOTALLOW

请求体缺失 confirm 字段：按实现，视为 null→走拒绝/或校验失败；若框架校验则 400 FIELD_NOTVALID

类型不符/缺失 type：当前实现未强制校验 type 与单据一致，通常仍可通过；如需严格，可期望 400/业务异常

3.4 边界值测试用例

conclusion 为空字符串：可接受但应记录；若有长度校验则 400

type 边界：0/1/2 合法，其他（99）期望 400/业务异常

3.5 权限测试用例

不提供 token：期望 401/403

越权：token.departId≠shopId → errno=RESOURCE_ID_OUTSCOPE（HTTP 403）

shopId 与售后单所属店铺不一致 → 同样 RESOURCE_ID_OUTSCOPE

4. PATCH /aftersales/{afterSaleId}/cancel 取消售后

4.1 正常流程测试用例

输入:

{

"operatorType": "USER",

"operatorId": 10,

"cancelReason": "no need"

}

操作者 token：可为用户或商户，取决于状态与规则。示例（用户取消）："user_token"，customerId 匹配单据。

售后单 id：3001（状态=0/1/3/4 可取消；示例用状态=1，shopId=1，customerId=10）

Header：X-Request-Id: "req-123"

请求体:

{ "operatorType": "USER", "operatorId": 10, "cancelReason": "no need"}

输出（:

{

"errno": 0,

"errmsg": "成功",

"data": {

"afterSaleId": 3001,

"currentStatus": 6, // 已取消

"cancelTime": "2025-01-01T10:00:00",

"operatorType": "USER",

"cancelReason": "no need",

"requestId": "req-123",
"changed": true
}
}

4.2 异常流程测试用例

售后不存在：afterSaleId 不存在 → errno=RESOURCE_ID_NOTEXIST

状态不可取消：status=2（待用户退货）或其他未列入可取消集合 → errno=STATENOTALLOW

操作者类型非法：operatorType="OTHER" → errno=FIELD_NOTVALID

越权（用户/商户 id 不匹配单据）→ errno=RESOURCE_ID_OUTSCOPE

商户尝试取消状态 0（默认仅用户可取消）→ errno=RESOURCE_ID_OUTSCOPE

4.3 边界值测试用例

重复取消（已是状态 6）→ errno=0，changed=false，不应落库。

状态 5（待商家换货）：用户尝试取消 → 越权；商户匹配取消 → 成功。

cancelReason 空串或超长：若有长度校验则 FIELD_NOTVALID；当前实现未加长度校验，可记录为同意取消。

4.4 权限测试用例

不提供 token（若网关要求登录）→ 401/403；若网关放行则按 operatorType/id 继续校验，可能仍返回越权。

不同店铺商户取消他店售后 → errno=RESOURCE_ID_OUTSCOPE

不同用户取消他人售后 → errno=RESOURCE_ID_OUTSCOPE

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	PUT /service-orders/{id}/assignee	指派服务单处理人员	TestAssign1	成功指派员工（状态流转为DISPATCHED）
			TestAssign2	服务单不存在或员工不存在
			TestAssign3	员工不属于该服务商（越权指派）
			TestAssign4	服务单状态不允许（已完成/已取消）
			TestAssign5	用户无权限（非服务商管理员）
2	DELETE /service-orders/{id}/cancel	取消服务单	TestCancelOrder1	成功取消（状态流转为CANCELLED）
			TestCancelOrder2	幂等性测试（重复取消不报错）
			TestCancelOrder3	状态不允许取消（如已完成）
			TestCancelOrder4	服务单不存在
			TestCancelOrder5	用户无权限（非本人/非对应商户）
3	PUT /service-orders/{id}/complete	完成服务单	TestComplete1	成功完成上门服务单（状态流转为COMPLETED）
			TestComplete2	成功完成寄修服务单（触发物流网关创建退货单）
			TestComplete3	非指派员工操作（越权）
			TestComplete4	状态不允许（未开始/已取消）
4	PUT /service-contracts/{id}/cancellation	取消服务合同	TestCancelContract1	成功取消（商户终止/到期/服务商退出）
			TestCancelContract2	存在活跃服务单（禁止取消）
			TestCancelContract3	未到期尝试以“过期”原因取消

			TestCancelContract4	无效的原因代码
			TestCancelContract5	幂等性测试（已终止再次取消）
5	POST /internal/shops/{did}/products/{id}/region/{rid}/service eOrders	创建服务单	TestCreate1	成功创建
			TestCreate2	用户无权限
			TestCreate3	参数缺失
			TestCreate4	服务商不存在
			TestCreate5	地区id不存在
6	GET /shops/{did}/serviceOrder/{id}	店铺根据服务单id查询服务单信息	TestSearch1	成功查询
			TestSearch2	用户无权限
			TestSearch3	参数缺失
			TestSearch4	服务单不存在

2.3.4 服务模块测试用例

1. PUT /service-orders/{id}/assignee 指派服务单处理人员

1.1 正常流程测试用例

输入:用户 token: "provider_admin_token"（服务商管理员，departId=100）

服务单 id (id): 1001（状态=UNHANDLED/ACCEPTED，shopId=1，serviceProviderId=100）

请求体:

```
{
 "staffId": 2001,
 "message": "请尽快处理"
}
```

输出:

```
{
 "errno": 0,
 "errmsg": "成功",
 "data": {
  "id": 1001,
  "status": 2, // DISPATCHED
  "maintainerId": 2001,
  "maintainerName": "张三"
 }
}
```

说明: 成功指派后，服务单状态变更为 DISPATCHED (2)，并记录维修人员信息。

1.2 异常流程测试用例

输入:服务单不存在 (id=9999) 或 员工不存在 (staffId=9999)

输出:

{ "errno": 504, "errmsg": "RESOURCE_ID_NOTEXIST" }

输入:服务单状态不允许（如已完成 status=3）

输出:

{ "errno": 507, "errmsg": "STATENOTALLOW" }

1.3 边界值测试用例

输入:staffId 为负数或 0

输出:期望返回 FIELD_NOTVALID (503) 或 RESOURCE_ID_NOTEXIST (504)。

1.4 权限测试用例

输入:员工不属于当前服务商（staffId=3001，归属 departId=200）

输出:

{ "errno": 505, "errmsg": "RESOURCE_ID_OUTSCOPE" }

输入:非服务商管理员操作（如普通用户 token）

输出:

{ "errno": 505, "errmsg": "RESOURCE_ID_OUTSCOPE" }

2. DELETE /service-orders/{id}/cancel 取消服务单

2.1 正常流程测试用例

输入:用户 token: "user_token"（服务单所属用户）或 "merchant_token"（所属商户）

服务单 id (id): 1002（状态=UNHANDLED/ACCEPTED）

请求体:

```
{
 "reasonCode": "USER_CANCEL",
 "reason": "不需要服务了"
}
```

输出:


```
{
 "errno": 0,
 "errmsg": "成功",
 "data": {
  "id": 1002,
  "status": 4, // CANCELLED
  "cancelReason": "不需要服务了"
 }
}
```


2.2 异常流程测试用例

输入:服务单不存在 (id=9999)

输出:


```
{ "errno": 504, "errmsg": "RESOURCE_ID_NOTEXIST" }
```


输入:状态不允许取消（如已完成 status=3）

输出:


```
{ "errno": 507, "errmsg": "STATENOTALLOW" }
```


2.3 边界值测试用例

输入:重复取消（服务单已是 CANCELLED 状态）

输出:


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


说明: 幂等性设计，不报错，状态保持不变。

2.4 权限测试用例

输入:其他用户尝试取消非本人的服务单

输出:


```
{ "errno": 505, "errmsg": "RESOURCE_ID_OUTSCOPE" }
```

3. PUT /service-orders/{id}/complete 完成服务单

3.1 正常流程测试用例（上门服务）

输入:用户 token: "staff_token"（被指派的维修人员）

服务单 id (id): 1003（类型=上门服务，状态=DISPATCHED）

请求体:


```
{
 "result": "维修成功，更换零件A",
 "status": "COMPLETED"
}
```


输出:


```
{
 "errno": 0,
 "errmsg": "成功",
 "data": {
  "id": 1003,
  "status": 3, // COMPLETED
  "result": "维修成功，更换零件A"
 }
}
```


3.2 正常流程测试用例（寄修服务）

输入:服务单 id (id): 1004（类型=寄修服务，状态=RECEIVED）

请求体同上

输出:


```
{
 "errno": 0,
 "errmsg": "成功",
 "data": {
  "id": 1004,
```

```
  "status": 3,
  "returnExpressNo": "SF123456789" // 自动生成的退货物流单号
 }
}
```

说明：系统自动调用物流网关创建退货物流单。

3.3 异常流程测试用例

输入:状态不允许（如未指派 status=0）

输出:

```
{ "errno": 507, "errmsg": "STATENOTALLOW" }
```

3.4 权限测试用例

输入:非当前指派的维修人员尝试完成

输出:

```
{ "errno": 505, "errmsg": "RESOURCE_ID_OUTSCOPE" }
```

4. PUT /service-contracts/{id}/cancellation 取消服务合同

4.1 正常流程测试用例

输入:用户 token: "merchant_token"（合同甲方）

合同 id (id): 2001（状态=VALID）

请求体:

```
{
 "reasonCode": "MERCHANT_TERMINATE",
 "reason": "业务调整，终止合作"
}
```

输出:

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

```
  "id": 2001,
  "status": 4, // ENDED
  "reasonCode": "MERCHANT_TERMINATE"
 }
}
```

4.2 异常流程测试用例

输入:存在活跃服务单（该合同下有未完成的服务单）

输出:

```
{ "errno": 507, "errmsg": "Active orders exist" }
```

输入:未到期尝试以“过期”原因取消（reasonCode="EXPIRED"，但 endTime > 当前时间）

输出:

```
{ "errno": 507, "errmsg": "Contract not expired yet" }
```

输入:无效的原因代码（reasonCode="INVALID_CODE"）

输出:

```
{ "errno": 503, "errmsg": "Invalid reason code" }
```

4.3 边界值测试用例

输入:幂等性测试（合同已是 ENDED 状态）

输出:

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

说明: 状态保持不变，不抛出异常。

4.4 权限测试用例

输入:非合同相关方（第三方用户）尝试取消

输出:

```
{ "errno": 505, "errmsg": "RESOURCE_ID_OUTSCOPE" }
```

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

5.1 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
  }
}
```


5.2 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" }

5.3 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" }

5.4 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" }

5.5 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

6.1 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
  }
}
```

6.2 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" }

6.3 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" }

6.4 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" }

3．软件需求测试结论

售后模块：

编号	API	测试结论
1	POST /orderitems/{id}/aftersales	通过测试，功能正常
2	GET /shops/{id}/aftersales	通过测试，功能正常
3	POST /shops/{id}/aftersales/{afterSaleId}/confirm	通过测试，功能正常
4	PATCH /aftersales/{afterSaleId}/cancel	通过测试，功能正常

服务模块：

编号	API	测试结论
1	PUT /service-orders/{id}/assignee	通过测试，功能正常
2	DELETE /service-orders/{id}/cancel	通过测试，功能正常
3	PUT /service-orders/{id}/complete	通过测试，功能正常
4	PUT /service-contracts/{id}/cancellation	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

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。