

---

# Testing Document

**SWEN90007**

**<Market Place>**

Team: <Team0>

In charge of: <Team 0 SWEN90007 >

---



---

SCHOOL OF  
**COMPUTING &  
INFORMATION  
SYSTEMS**

---

### Revision History

Date	Version	Description	Author
6/10/2022	01.00	Initial draft	Xi Zhao Wei Hong Lai
6/10/2022	01.00	Initial test setup	Xi Zhao Wei Hong Lai
13/10/2022	02.00-D01	Finish the test of customers and complete test cases document of customers	Xi Zhao
20/10/2022	02.00-D02	Finish the test of sellers and complete test cases document of sellers	Xi Zhao
20/10/2022	02.00-D03	Finish the test of admins and complete test cases document of admins	Xi Zhao
20/10/2022	02.00-D03	Finish the data entry document	Xi Zhao
20/10/2022	02.00-D04	Finish the test of Integration Tests and complete test cases document of Integration Tests	Wei Hong Lai
20/10/2022	02.00-D05	Reviewed	Wei Hong Lai

## 1. Introduction

---

### 1.1 Proposal

The purpose of this document is to define and present the test cases for project Team0 Market Place, covering the test cases for the system use cases.

### 1.2 Target Users

This document is mainly designed for those responsible for executing the test cases in this project Team0 and SWEN90007 teaching team.

### 1.3 Conventions, terms and abbreviations

This section explains the concept of some important terms that will be used throughout this document. These terms are described in the following table, presented in alphabetical order.

Term	Description
LMS	Learning Management System

## 2. Covered Requirements

---

This section lists the system requirements covered in the test cases.

### 2.1 Functional or Product Requirements

Requirement Identifier	Requirement Name
UC001	Customer Test Cases
UC002	Seller Test Cases
UC003	Admin Test Cases

### 3. Test Cases

#### 3.1 UC001: Customer Test Cases

##### 3.1.1 TC001: Customer bids for item - successful

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify if the customer can bid for items they want successfully.	
<b>Setup:</b> <i>The customer is logged on.</i>	
<b>Pre-Conditions:</b> <i>The customer is logged on.</i>	
<b>Notes:</b> [1] As the customer tries to bid for the item bid(2, Money.of(3, Monetary.getCurrency("AUD"))) [2] The money currency for bidding will be AUD	
<b>Time constraint:</b> Minimum: 5 min Maximum: 10 min	

##### 3.1.2 TC002: Customer bids for item which is fixed price - unsuccessful

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify if the customer cannot bid for item which is fixed price	
<b>Setup:</b> <i>The customer is logged on</i>	
<b>Pre-Conditions:</b> <i>The customer is logged on</i>	

<b>Notes:</b>  [1] The item of listing id 2 is fixed price (1, Money.of(3, Monetary.getCurrency("AUD"))) [2] The money currency for bidding will be AUD
<b>Time constraint:</b>  Minimum: 5 min Maximum: 10 min

### 3.1.3 TC003: Customer bids for item which is out of list - unsuccessful

<b>Test Type:</b>  Functional	<b>Execution Type:</b>  Manual
<b>Objective:</b>  Verify if the customer cannot bid for item which is out of list	
<b>Setup:</b>  <i>The customer is logged on</i>	
<b>Pre-Conditions:</b>  <i>The customer is logged on</i>	
<b>Notes:</b>  [1] The item of listing id 300 is out of list (300, Money.of(3, Monetary.getCurrency("AUD"))) [2] The money currency for bidding will be AUD	
<b>Time constraint:</b>  Minimum: 1 min Maximum: 10 min	

### 3.1.4 TC004: Customer bids for item which money is too small- unsuccessful

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify if the customer cannot bid for item when money is too small	
<b>Setup:</b> <i>The customer is logged on</i>	
<b>Pre-Conditions:</b> <i>The customer is logged on</i>	
<b>Notes:</b> [1] The money of 1 is too small for bidding (2, Money.of(1, Monetary.getCurrency("AUD"))) [2] The money currency for bidding will be AUD	
<b>Time constraint:</b> Minimum: 1 min Maximum: 10 min	

### 3.1.5 TC005: Customer views all orders - successful

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify that the customer can view all the orders successfully	
<b>Setup:</b> <i>The customer is logged on and orders are not null</i>	
<b>Pre-Conditions:</b> <i>The customer is logged on and orders are not null</i>	

**Notes:**

[1] orders are not null

**Time constraint:**

Minimum: 1 min

Maximum: 10 min

**3.1.6 TC006: Customer cancel the order -successful**

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify that the customer can cancel the order successful	
<b>Setup:</b> <i>The customer is logged on and have orders</i>	
<b>Pre-Conditions:</b> The customer is logged on and have orders	
<b>Notes:</b> [1] cancelOrder(1) [2] The order of order id 1 exists	
<b>Time constraint:</b> Minimum: 1 min Maximum: 10 min	

**3.1.7 TC007: Customer cancel the order -unsuccessful**

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify if the customer cannot cancel the order which is not exist	
<b>Setup:</b> <i>The customer is logged on and have orders</i>	
<b>Pre-Conditions:</b> <i>The customer is logged on and have orders</i>	
<b>Notes:</b> [1] cancelOrder(999) [2] The order of order id 300 is not exist in the system	
<b>Time constraint:</b> Minimum: 1 min Maximum: 10 min	

### 3.1.8 TC008: Customer modify the order -successful

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify that the customer can modify the order successful	
<b>Setup:</b> <i>The customer is logged on and have orders</i>	
<b>Pre-Conditions:</b> <i>The customer is logged on and have orders</i>	
<b>Notes:</b> [1] The item for order id 1 and list id 1 exists [2] The quantity modifies to 3	



**Time constraint:**

Minimum: 1 min

Maximum: 10 min

**3.1.9 TC009: Customer check out the order -successful**

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify that the customer can check out the order successful	
<b>Setup:</b> <i>The customer is logged on and have orders</i>	
<b>Pre-Conditions:</b> <i>The customer is logged on and have orders</i>	
<b>Notes:</b> [1] separate situation of order has one item and more than one item	
<b>Time constraint:</b> Minimum: 1 min Maximum: 10 min	

### 3.2 UC002: Seller Test Cases

#### 3.2.1 TC001: Seller Creates a Fixed Price Listing

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify that the seller can creates a Fixed Price Listing for their item successfully	
<b>Setup:</b> <i>The seller is logged on</i>	
<b>Pre-Conditions:</b> <i>The seller is logged on</i>	
<b>Notes:</b> [1] Fixed Price Listing: ListType.Fixed_Price [2] Money currency: Monetary.getCurrency("AUD") [3] LocalDateTime.of(2022, 1, 1, 10, 10) to LocalDateTime.of(2022, 2, 1, 10, 10)	
<b>Time constraint:</b> Minimum: 1 min Maximum: 10 min	

#### 3.2.2 TC002: Seller Creates an Auction Listing

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify that the seller can creates an Auction Listing for their item successfully	
<b>Setup:</b> <i>The seller is logged on</i>	
<b>Pre-Conditions:</b>	

<i>The seller is logged on</i>
<b>Notes:</b>  [1] Fixed Price Listing: ListType.Auction [2] Money currency: Monetary.getCurrency("AUD") [3] LocalDateTime.of (2022, 1, 1, 10, 10) to LocalDateTime.of(2022, 2, 1,10, 10))
<b>Time constraint:</b>  Minimum: 1 min Maximum: 10 min

### 3.2.3 TC003: Seller deletes listing -successful

<b>Test Type:</b>  Functional	<b>Execution Type:</b>  Manual
<b>Objective:</b>  Verify that the seller can delete listing successfully	
<b>Setup:</b>  <i>The seller is logged on and list exists</i>	
<b>Pre-Conditions:</b>  <i>The seller is logged on and list exists</i>	
<b>Notes:</b>  [1] deleteListing(1,1) [2] listing with lising id 1 and group id 1 is exist	
<b>Time constraint:</b>  Minimum: 1 min Maximum: 10 min	

### 3.2.4 TC004: Seller deletes listing -unsuccessful

<b>Test Type:</b>  Functional	<b>Execution Type:</b>  Manual
-------------------------------------	--------------------------------------

<b>Objective:</b>
Verify if the seller can not delete listing which is not exist in the system
<b>Setup:</b>
<i>The seller is logged on</i>
<b>Pre-Conditions:</b>
<i>The seller is logged on</i>
<b>Notes:</b>
[1] deleteListing(999,1)
[2] listing with lising id 999 and group id 1 is not exist in the system
<b>Time constraint:</b>
Minimum: 1 min
Maximum: 10 min

### 3.2.5 TC004: Seller views the order

<b>Test Type:</b>	<b>Execution Type:</b>
Functional	Manual
<b>Objective:</b>	
Verify that the seller can view the order successfully	
<b>Setup:</b>	
<i>The seller is logged on and order exists</i>	
<b>Pre-Conditions:</b>	
<i>The seller is logged on and order exists</i>	
<b>Notes:</b>	
[1] viewOrders()	
<b>Time constraint:</b>	
Minimum: 1 min	
Maximum: 10 min	

### 3.2.6 TC004: Seller modify the order

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify that the seller can modify the order successfully	
<b>Setup:</b> <i>The seller is logged on and order exists</i>	
<b>Pre-Conditions:</b> <i>The seller is logged on and order exists</i>	
<b>Notes:</b> [1] modifyOrder(1,1,1,2) [2] order id, group id, listing id and modify the quantity to 2	
<b>Time constraint:</b> Minimum: 1 min Maximum: 10 min	

### 3.2.7 UC005: Seller views seller listings

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify that the seller can view seller listings successfully	
<b>Setup:</b> <i>The customer is logged on and seller listings exists</i>	
<b>Pre-Conditions:</b> <i>The customer is logged on and seller listings exists</i>	

**Notes:**

[1] group id (1)

**Time constraint:**

Minimum: 1 min

Maximum: 10 min

**3.2.8 UC008: Seller views full orders**

<b>Test Type:</b>	<b>Execution Type:</b>
Functional	Manual
<b>Objective:</b>	
Verify that the seller can view full orders successfully	
<b>Setup:</b>	
<i>The customer is logged on and seller listings exists</i>	
<b>Pre-Conditions:</b>	
<i>The customer is logged on and seller listings exists</i>	
<b>Notes:</b>	
[1] viewFullOrder(1)	
[2] orders with group id 1 are exist in the system	
<b>Time constraint:</b>	
Minimum: 1 min	
Maximum: 10 min	

**3.2.9 UC009: Seller cancel the order -successful**

<b>Test Type:</b>	<b>Execution Type:</b>
Functional	Manual
<b>Objective:</b>	
Verify that the seller can cancel the order successfully	

<b>Setup:</b>
<i>The customer is logged on and orders exists in the system</i>
<b>Pre-Conditions:</b>
<i>The customer is logged on and orders exists in the system</i>
<b>Notes:</b>
[1] cancelOrder(1,1) [2] order with order id 1 and group id 1 is exist in the system
<b>Time constraint:</b>
Minimum: 1 min Maximum: 10 min

### 3.2.10 UC0010: Seller cancel the order -unsuccessful

<b>Test Type:</b>	<b>Execution Type:</b>
Functional	Manual
<b>Objective:</b>	
Verify if the seller cannot cancel the order which is not exist in the system	
<b>Setup:</b>	
<i>The customer is logged on</i>	
<b>Pre-Conditions:</b>	
<i>The customer is logged on</i>	
<b>Notes:</b>	
[1] cancelOrder(999,1) [2] order with order id 999 and group id 1 is not exist in the system	
<b>Time constraint:</b>	
Minimum: 1 min Maximum: 10 min	

### 3.3 UC003: Admin Test cases

#### 3.3.1 TC001: Admin views all users

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify if the admin can view all users successfully	
<b>Setup:</b> <i>The admin is logged on</i>	
<b>Pre-Conditions:</b> <i>The admin is logged on</i>	
<b>Notes:</b> [1] we also separate the situation which one user in the user list and more than one user in the user list	
<b>Time constraint:</b> Minimum: 1 min Maximum: 10 min	

#### 3.3.2 UC002: Admin remove listings -successful

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manua
<b>Objective:</b> Verify if the admin can remove <i>listings that they deem inappropriate successfully</i>	
<b>Setup:</b> <i>The admin is logged on and there are listings in the system</i>	
<b>Pre-Conditions:</b> <i>The admin is logged on and there are listings in the system</i>	



**Notes:**

[1] the listing id (1) is a exist listing

[2] removeListing(1)

**Time constraint:**

Minimum: 20 min

Maximum: 25 min

**3.3.3 TC003: Admin remove listings -unsuccessful**

<b>Test Type:</b>	<b>Execution Type:</b>
Functional	Manual
<b>Objective:</b>	
Verify if the admin cannot remove <i>listings which is not exist</i>	
<b>Setup:</b>	
<i>The admin is logged on</i>	
<b>Pre-Conditions:</b>	
<i>The admin is logged on</i>	
<b>Notes:</b>	
[1] the listing id (0) is not a exist listing	
[2] removeListing(0)	
<b>Time constraint:</b>	
Minimum: 20 min	
Maximum: 25 min	

**3.3.4 TC004: Admin add seller to seller-group - successful**

<b>Test Type:</b>	<b>Execution Type:</b>
Functional	Manual

<b>Objective:</b>
Verify if the admin can add seller to the seller-group successfully
<b>Setup:</b>
<i>The admin is logged on and there are sellers and seller groups in the system</i>
<b>Pre-Conditions:</b>
<i>The admin is logged on and there are sellers and seller groups in the system</i>
<b>Notes:</b>
[1] addSellerToGroup("a",3)
[2] seller group with group name “a” is a exist group
<b>Time constraint:</b>
Minimum: 5 min
Maximum: 10 min

### 3.3.5 TC005 Admin add seller to seller-group -unsuccessful

<b>Test Type:</b>	<b>Execution Type:</b>
Functional	Manual
<b>Objective:</b>	
Verify if the admin cannot add seller to the seller-group which is not exist	
<b>Setup:</b>	
<i>The admin is logged on and there are sellers and seller groups in the system</i>	
<b>Pre-Conditions:</b>	
<i>The admin is logged on and there are sellers and seller groups in the system</i>	
<b>Notes:</b>	
[1] addSellerToGroup("Team0",3)	
[2] seller group with group name “Team0” is a exist group	

**Time constraint:**

Minimum: 5 min

Maximum: 10 min

**3.3.6 TC06: Admin remove seller from seller-group -successful**

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify if the admin can remove seller from the seller-group successfully	
<b>Setup:</b> <i>The admin is logged on and there are sellers in the seller group.</i>	
<b>Pre-Conditions:</b> <i>The admin is logged on and there are sellers in the seller group.</i>	
<b>Notes:</b> [1] removeSellerFromGroup("a",1), seller group with group name “a” is exist in the system	
<b>Time constraint:</b> Minimum: 5 min Maximum: 10 min	

**3.3.7 TC07: Admin remove seller from seller-group -unsuccessful**

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify if the admin cannot remove seller from the seller-group when seller-group is null	
<b>Setup:</b> <i>The admin is logged on and there are sellers in the seller group.</i>	

**Pre-Conditions:**

*The admin is logged on and there are sellers in the seller group.*

**Notes:**

[1] removeSellerFromGroup("Team0",1)

[2] seller group with group name "team 0" is not exist in the system

**Time constraint:**

Minimum: 5 min

Maximum: 10 min

**3.3.8 TC008: Admin create a new seller-group -successful**

<b>Test Type:</b>	<b>Execution Type:</b>
Functional	Manual
<b>Objective:</b>	
Verify if the admin can create a new seller-group successfully	
<b>Setup:</b>	
<i>The admin is logged on</i>	
<b>Pre-Conditions:</b>	
<i>The admin is logged on</i>	
<b>Notes:</b>	
[1] createSellerGroup("Team0")	
[2] seller group with group name "team 0" is not exist in the system	
<b>Time constraint:</b>	
Minimum: 5 min	
Maximum: 10 min	

**3.3.9 TC009: Admin create a new seller-group -unsuccessful**

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify if the admin cannot create a new seller-group when seller group is already existed in the system	
<b>Setup:</b> <i>The admin is logged on and the seller group is already existed in the system which the admin want to create</i>	
<b>Pre-Conditions:</b> <i>The admin is logged on and the seller group is already existed in the system which the admin want to create</i>	
<b>Notes:</b> [1] createSellerGroup("a") [2] seller group with group name “a” is al exist in the system	
<b>Time constraint:</b> Minimum: 5 min Maximum: 10 min	

### 3.3.10 TC0010: Admin remove the listing -successful

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b> Verify that admin can remove the listing successfully	
<b>Setup:</b> <i>The admin is logged on</i>	
<b>Pre-Conditions:</b> <i>The admin is logged on</i>	

**Notes:**

- [1] removeListing(1)
- [2] Listing with listing id 1 is already exist in the system

**Time constraint:**

Minimum: 5 min  
Maximum: 10 min

**3.3.11 TC0011: Admin remove the listing -unsuccessful**

<b>Test Type:</b>	<b>Execution Type:</b>
Functional	Manual
<b>Objective:</b>	
Verify that admin cannot remove the listing which is not exist in the system	
<b>Setup:</b>	
<i>The admin is logged on</i>	
<b>Pre-Conditions:</b>	
<i>The admin is logged on</i>	
<b>Notes:</b>	
[1] removeListing(999)	
[2] Listing with listing id 999 is not exist in the system	
<b>Time constraint:</b>	
Minimum: 5 min Maximum: 10 min	

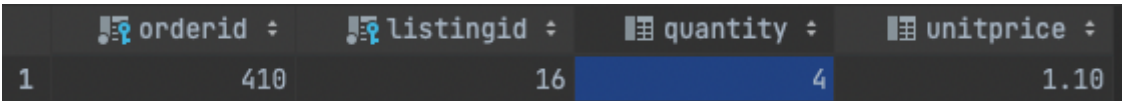
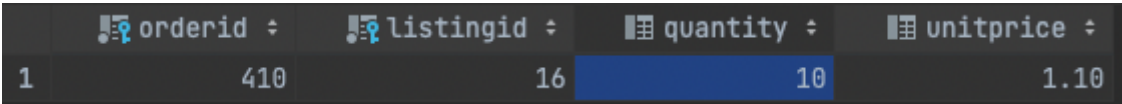
**4. Integration Tests**

All integration tests are run in the following configurations.

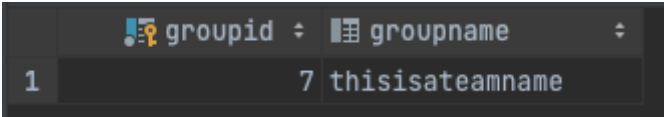
Parameter	Value
-----------	-------

Number of threads (users)	100
Ramp-up period	1
Loop count	2

#### 4.1.1 Modify Order in a multithreaded environment

<b>Test Type:</b> Integration	<b>Execution Type:</b> Automated JMeter
<b>Objective:</b> Ensure that modify order operates correctly in a multithreaded environment	
<b>Setup:</b> The following entry should be present in the database <pre>{listingId: 16, quantity: 4, priceInCents: 110}</pre>	
<b>Pre-Conditions:</b> User 5 is logged into the system	
<b>Notes:</b> <ol style="list-style-type: none"> <li>At the end of the operation, the fields in the database entry should be non-negative.</li> <li>The database should have the following entry           <ol style="list-style-type: none"> <li>{listingId: 16, quantity: 10, priceInCents: 110}</li> </ol> </li> </ol>	
<b>Relevant Evidence:</b>  <p>Screenshot taken after the test</p> 	

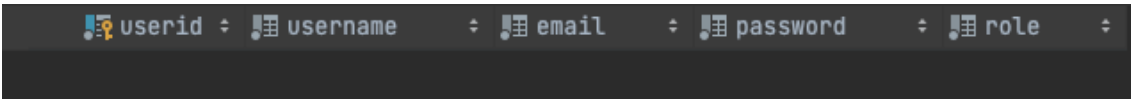
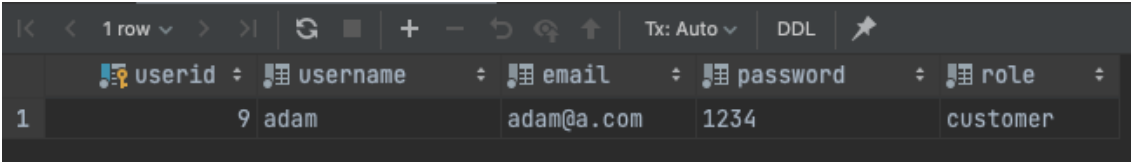
#### 4.1.2 Add Seller Group

<b>Test Type:</b>	<b>Execution Type:</b>
Integration	Automated JMeter
<b>Objective:</b>	
Ensure that all sellergroups are unique	
<b>Setup:</b>	
The following entry should not be present in the database  {groupName: thisIsATeamName}	
<b>Pre-Conditions:</b>	
The admin is logged into the system	
<b>Notes:</b>	
3. The database should have the following entry <ul style="list-style-type: none"> <li>a. {groupName: thisIsATeamName, groupId: &lt;something&gt;}</li> </ul> 4. There should be no duplicate group names in the database	
<b>Relevant Evidence:</b>	
Query: SELECT * FROM sellergroups WHERE groupName="thisIsATeamName";	
<pre>marketPlaceDB.public&gt; SELECT * FROM sellergroups WHERE groupname="thisIsATeamName" [2022-10-20 09:40:33] [42703] ERROR: column "thisIsATeamName" does not exist [2022-10-20 09:40:33] Position: 44</pre>	
Screenshot before the test	
	

#### 4.1.3 Register

<b>Test Type:</b>	<b>Execution Type:</b>
-------------------	------------------------



Integration	Automated JMeter
<b>Objective:</b> Ensure that no duplicate emails are used during registration	
<b>Setup:</b> The following entry should not be present in the database {username: adam, email: adam@a.com, password: 1234}	
<b>Pre-Conditions:</b>	
<b>Notes:</b> 5. The database should have the following entry a. {username: adam, email: adam@a.com, password: 1234} 6. Only one such entry should be present in the User table	
<b>Relevant Evidence:</b> Screenshot taken before the test  Screenshot taken after the test 	

#### 4.1.4 Register

<b>Test Type:</b>	<b>Execution Type:</b>
Integration	Automated JMeter
<b>Objective:</b> Ensure that sellers only belong to one seller group	

**Setup:**

The following entry should not be present in the database

{userid: 1, groupid: <groupid for group with groupname='something'}

**Pre-Conditions:**

1. User with userid 1 is an existing seller

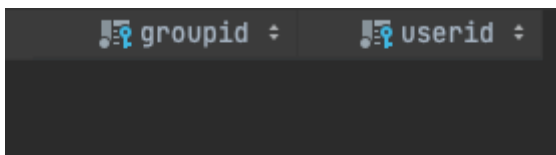
**Notes:**

7. The database should have the following entry
  - a. {userid: 1, groupid: <groupid for group with groupname='somethings'}
8. Only one such entry should be present in the groupmembership table
9. In our run, 'somethings' has a groupid of 3.

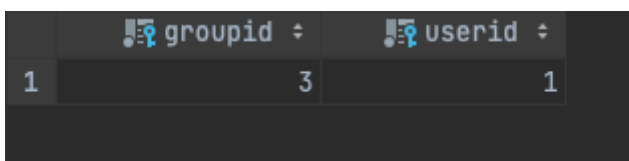
**Relevant Evidence:**

**Query : SELECT \* FROM groupmembership where userid=1 and groupid=3**

Screenshot taken before the test



Screenshot taken after the test

**4.1.5 Checkout**

<b>Test Type:</b>	<b>Execution Type:</b>
Integration	Automated JMeter
<b>Objective:</b>	
Ensure that the listings have sufficient quantity to allow orders to be checked out	

**Setup:**

The following entry should not be present in the database

{listingId: 21, quantity: 40, priceInCents: 110}

**Pre-Conditions:**

2. Every request has to have a valid JWT token
3. Every request submits this order
  - a. {listingId: 21, quantity: 4, priceInCents: 110}

**Notes:**

10. The database should have the following entry if more than 50 users checkout the same listing
  - a. {listingId: 21, quantity: 0, priceInCents: 110}

**Relevant Evidence:**

**Query : SELECT \* FROM listings where listingid=16**

Screenshot taken before the test

	listingid	groupid	type	title	description	quantity	price	starttime	endtime
1	21	1	FIXED_PRICE	a	a	40	10.00	2022-09-14 23:17:00.000000	2022-09-14 23:17:00.000000

Screenshot taken after the test

	listingid	groupid	type	title	description	quantity
1	21	1	FIXED_PRICE	a	a	0

## 5. Entry Data

### 5.1 Bid: <Bid Data Set>

*Description:*

< The entry data that will be used for bidding for customer >

Field	Value
1. Bid bid1	(1, 1, Money.of(10, Monetary.getCurrency("AUD"))) )
2. Bid bid2	( 2, 2, Money.of(20, Monetary.getCurrency("AUD")))

## 5.2 User: <User Data Set>

### Description:

< The entry data that will be users that exist in the system>

Field	Value
1. User user1	("a", "a", "a", 1, UserRoles.CUSTOMER.toString())
2. User user1	("b", "b", "b", 2, UserRoles.SELLER.toString())

## 5.3 GroupMembership: <GroupMembership Data Set>

### Description:

< The entry data that will be group memberships that exist in the system>

Field	Value
1.GroupMembership gm1	(1,1)
2. GroupMembership gm2	(2,2)

## 5.4 Listing: <Listing Data Set>

### Description:

< The entry data that will be listing data that exist in the system>

Field	Value
1.Listing listing1	(1,1, ListingTypes.FIXED_PRICE, "a", "a", 5,Money.of(1, Monetary.getCurrency("AUD")), LocalDateTime.now(), LocalDateTime.now())
2. Listing listing2	(2,2, ListingTypes.ASUCTION, "b", "b", 2,Money.of(2, Monetary.getCurrency("AUD")), LocalDateTime.now(), LocalDateTime.now())

## 5.5 SellerGroup: <SellerGroup Data Set>

**Description:**

*< The entry data that will be Seller Groups that exist in the system >*

Field	Value
1.SellerGroup sg1	(1,"a")
2. SellerGroup sg2	(2,"b")

**5.6 Order: <Order Data Set>****Description:**

*< The entry data that will be orders that exist in the system >*

Field	Value
1.Order order	(1,1,"a")
2. SellerGroup sg2	(2,2,"b")

**5.7 OrderItem: <OrderItem Data Set>****Description:**

*< The entry data that will be orderitems that exist in the system >*

Field	Value
1. <i>Order order</i>	<i>(1, 1, 1, Money.of(1, Monetary.getCurrency("AUD")) )</i>
2. <i>SellerGroup sg2</i>	<i>(2, 2, 2, Money.of(2, Monetary.getCurrency("AUD")) )</i>