Test Cases

for

ShopPeas

Version 1.0 approved

Prepared by

Eng Yi Xuan
Koh Huei Shan Winnie
Lee Jedidiah
Rachel Tan Min Zhi
Saffron Salmah Yen Lim
Yan Jun Chao

Nanyang Technological University, Team Peas Limited

24/10/2024

Test Cases

1. Black box testing (Equivalence and boundary classes) Transaction

a) TransactionController
   i) checkout

Test id	Description
1.1.1	Valid shopping cart with multiple orders and multiple items without any discrepancies for any fields. Valid authorization key (typical case)
1.1.2	Valid shopping cart with 1 order only without any discrepancies for any fields. Valid authorization key (lower boundary)
1.1.3	Valid shopping cart with 4 orders without any discrepancies for any fields. Valid Authorization key (upper boundary)
1.1.4	Invalid shopping cart with 5 orders without any discrepancies for the field, valid authorization key (> upper boundary of 5)
1.1.5	Invalid shopping cart with 0 items
1.1.6	Valid shopping cart with 1 order with 1 item (lower boundary)
1.1.7	Valid shopping cart with 1 order with 5 items (upper boundary)
1.1.8	Invalid shopping cart with 1 order with no items (below lower boundary)
1.1.9	Invalid shopping cart with 1 order with 6 items (> upper boundary)
1.1.10	Invalid shopping cart with 1 valid order and 1 invalid order (invalid wholesaler/uen)
1.1.11	Invalid shopping cart with an empty string for the wholesaler
1.1.12	Invalid shopping cart where items array contains null values
1.1.13	Invalid cart with a missing uen field
1.1.14	Invalid cart with an invalid wholesaler
1.1.15	Invalid cart with an invalid uen
1.1.16	Invalid order with a missing quantity field in items
1.1.17	Invalid order with quantity = 0

Test id	Boundary Value Testing			Expected output	Actual output
	Inputs				
	No. of orders	No. of items	uen/wholesaler		
1.1.1	2 valid orders	Order 1: 3 valid items Order 2: 2 valid items	Both valid 201936456R 199203796C	201: Checkout successful, shopping cart deleted and part of order history. Transactions status updated	201: Checkout successful, shopping cart deleted and part of order history. Transactions status updated
1.1.2	1 valid order	Order 1: 1 valid item	Valid 201936456R	201: Checkout successful, shopping cart deleted and part of order history. Transactions status updated	201: Checkout successful, shopping cart deleted and part of order history. Transactions status updated
1.1.3	4 valid orders	Order 1: 1 valid item Order 2: 1 valid item Order 3: 1 valid item Order 4: 1 Valid item	Valid 201936456R 199203796C	201: Checkout successful, shopping cart deleted and part of order history. Transactions status updated	201: Checkout successful, shopping cart deleted and part of order history. Transactions status updated
1.1.4	5 valid orders	Order 1: 3 valid items Order 2: 2 valid items Order 3: 1 valid item Order 4: 1 valid item Order 5: 1 valid item	Valid 201936456R 199203796C	400: Checkout failed too many orders in cart, maximum allowed is 4	400: Checkout failed too many orders in cart, maximum allowed is 4
1.1.5	Invalid (order:0)	null	null	400: Checkout failed, Cart cannot be empty	400: Checkout failed, Cart cannot be empty
1.1.6	1 valid order	Order 1: 1 valid item	Valid 201936456R	201: Checkout successful, shopping cart deleted and part of order history. Transactions status updated	201: Checkout successful, shopping cart deleted and part of order history. Transactions status updated
1.1.7	1 valid order	Order 1: 5 valid items	Valid 201936456R	201: Checkout successful, shopping cart deleted and part of order history. Transactions status updated	201: Checkout successful, shopping cart deleted and part of order history. Transactions status updated
1.1.8	Invalid (order: 0)	Order 1: 0 items	Valid 201936456R	400: Checkout failed, each order must contain at least 1 item	400: Checkout failed, each order must contain at least 1 item
1.1.9	1 Valid order	Order 1: 6 valid items	Valid 201936456R	400: Checkout failed, too many items in an order, maximum allowed is 5	400: Checkout failed, too many items in an order, maximum allowed is 5

Test id	**Equivalence Partitioning Testing**			Expected output	Actual output
	Inputs				
	No. of orders	No. of items	uen/wholesaler		
1.1.10	2 orders (1 valid and 1 invalid)	Order 1: 1 valid item Order 2: 1 valid item	1 Valid: 201936456R 1 invalid: BK1001	400: Checkout failed, invalid UEN	400: Checkout failed, invalid UEN
1.1.11	1 valid order	Order 1: 1 valid item	Invalid Wholesaler field is left empty	400: Checkout failed, wholesaler cannot be left empty	400: Checkout failed, wholesaler cannot be left empty
1.1.12	1 invalid order	Order 1: null	Valid 201936456R	400: Checkout failed, each order must contain at least 1 item	400: Checkout failed, each order must contain at least 1 item
1.1.13	1 valid order	Order 1: 1 valid item	Invalid Uen field is left empty	400: Checkout failed, UEN cannot be left empty	400: Checkout failed, UEN cannot be left empty
1.1.14	1 valid order	Order 1: 1 valid item	Invalid (Wholesaler does not exist: Mos Burger)	400: Checkout failed, invalid wholesaler	400: Checkout failed, invalid wholesaler
1.1.15	1 valid order	Order 1: 1 valid item	Invalid (UEN does not exist: 12345)	400: Checkout failed, invalid UEN	400: Checkout failed, invalid UEN
1.1.16	1 invalid order	Order 1: 1 invalid item (field such as quantity is missing)	Valid 201936456R	400: Checkout failed, field in items missing	400: Checkout failed, field in items missing
1.1.17	1 invalid order	Order 1: 1 invalid item (quantity = 0)	Valid 201936456R	400: Checkout failed, each order must contain at least 1 item	400: Checkout failed, each order must contain at least 1 item

ii) getTransactionByWholesaler()

Test id	Description
1.2.1	Get all transactions for a specific wholesaler with status of IN-CART
1.2.2	Get all transactions for a specific wholesaler with status of COMPLETED
1.2.3	Get all transactions for a specific wholesaler with status of Pending-acceptance
1.2.4	Attempt to get all transactions of a wholesaler which has the wrong format
1.2.5	Attempt to get all transactions of no wholesalers (wholesaler is null)
1.2.6	Attempt to get all transactions of a specific wholesaler without inputting their status
1.2.7	Attempt to get all transactions of a specific wholesaler when their input status is inaccurate

Test id	Inputs		Expected output	Actual output
	uen	Status		
1.2.1	Valid (199203796C)	IN-CART	200: Returns a list of transactions with status “in-cart”	200: Returned list of transactions where status = IN-CART
1.2.2	Valid (201936456R)	COMPLETED	200: Returns a list of transactions with status “Completed”	200: Returned a list of transactions where status = COMPLETED
1.2.3	Valid (200915506H)	PENDING-ACCEPTANC E	200: Returns a list of transactions with status “Pending-acceptance”	200: Returned a list of transactions where status = PENDING-ACCEPTA NCE
1.2.4	Invalid UEN (12345)	Valid (In-Cart)	400: Invalid UEN	400: Invalid UEN
1.2.5	Invalid UEN (empty/ null)	Valid (In-Cart)	400: Invalid UEN	400: Invalid UEN
1.2.6	Valid (199203796C)	Invalid (Empty/null)	400: Invalid status	400: Invalid Status
1.2.7	Valid (existing uen)	Invalid (completed)	400: Invalid status	400: Invalid Status

iii) updateTransactionStatus

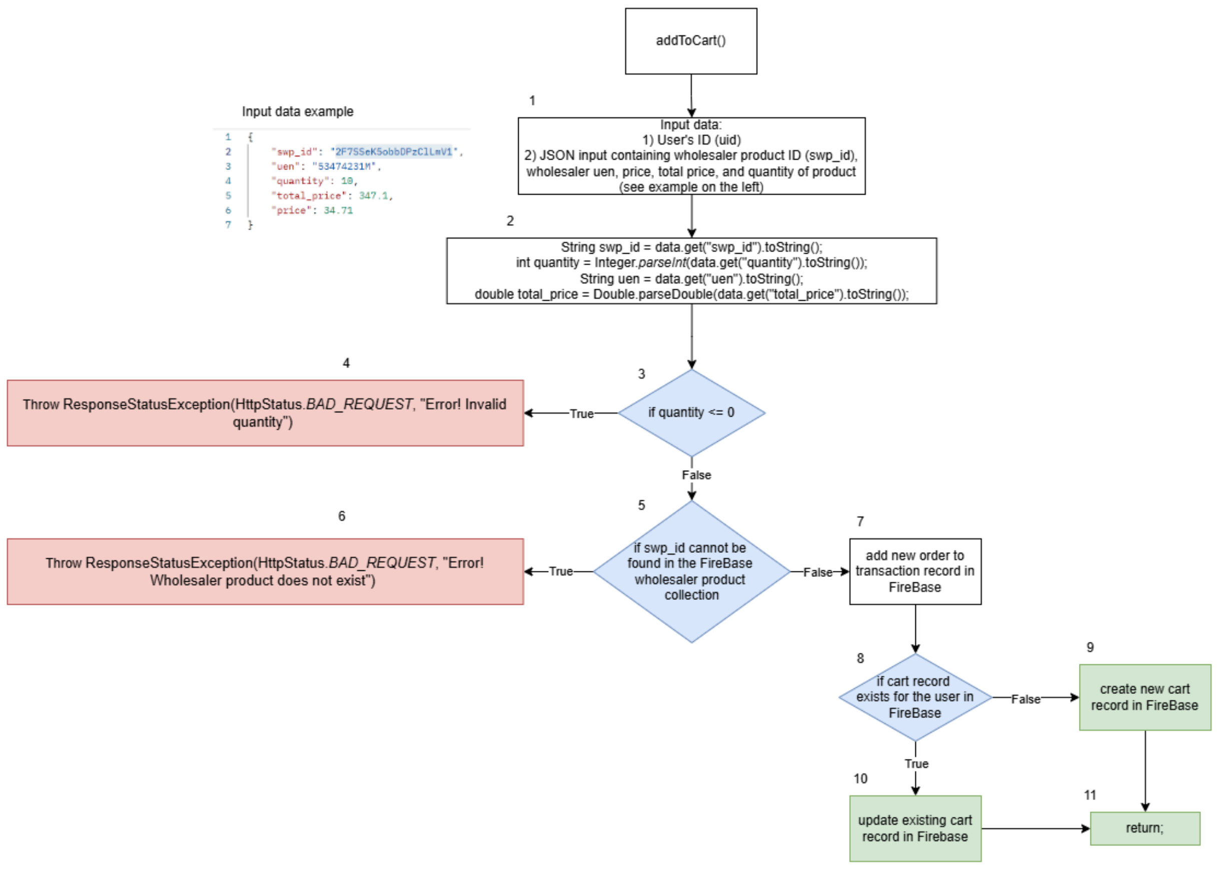
Test id	Description
1.3.1	Update a specific valid transaction to status of COMPLETED
1.3.2	Update a specific valid transaction to status of IN-CART
1.3.3	Update a specific valid transaction to status of PENDING-ACCEPTANCE
1.3.4	Attempt to update a specific transaction to COMPLETED with an invalid transaction id (tid)
1.3.5	Attempt to update a specific transaction to COMPLETED with an empty tid
1.3.6	Attempt to update a specific transaction to COMPLETED without tid field
1.3.7	Attempt to update a specific transaction to an invalid status such as in-cart instead of IN-CART

Test id	Inputs		Expected output	Actual output
	tid	Status		
1.3.1	Valid (IuUU3ZHTTLv7buR14nG0)	COMPLETED	200: Transaction status updated to COMPLETED	200: Transaction status updated to COMPLETED
1.3.2	Valid (n413FkKe0X6G2l9BFyHu)	IN-CART	200: Transaction status updated to IN-CART	200: Transaction status updated to IN-CART
1.3.3	Valid (41Mwt9l7y778UzlegiiS)	Valid (PENDING-ACCEPTANCE)	200: Transaction status updated to PENDING-ACCCEPTANCE	200: Transaction status updated to PENDING-ACCCEPTANCE
1.3.4	Invalid (12345)	Valid (COMPLETED)	400: Invalid tid	400: Invalid tid
1.3.5	Invalid tid (empty)	Valid (COMPLETED)	400: Invalid tid	400: Invalid tid
1.3.6	Invalid tid (null)	Valid (COMPLETED)	400: Invalid tid	400: Invalid tid
1.3.7	Valid (41Mwt9l7y778UzlegiiS)	Invalid format (in-cart)	400: Invalid status	400: Invalid status

2. White box testing for Cart Functions

2.1 Add to cart() : Add new order to Cart

2.1.1 Control Flow Graph & Valid Input Data


addToCart()
1
Input data:
1) User's ID (uid)
2) JSON input containing wholesaler product ID (swp_id), wholesaler uen, price, total price, and quantity of product
(see example on the left)
Input data example
2
3
if quantity <= 0
True
4
Throw ResponseStatusException(HttpStatus.BAD_REQUEST, "Error! Invalid quantity")
False
5
if swp_id cannot be found in the FireBase wholesaler product collection
True
6
Throw ResponseStatusException(HttpStatus.BAD_REQUEST, "Error! Wholesaler product does not exist")
False
7
add new order to transaction record in FireBase
8
if cart record exists for the user in FireBase
False
9
create new cart record in FireBase
True
10
update existing cart record in Firebase
11
return;

```
{
    "swp_id": "2F7SSeK5obbDPzClLmV1",
    "uen": "53474231M",
    "quantity": 10,
    "total_price": 347.1,
    "price": 34.71
}
```

```
String swp_id = data.get("swp_id").toString();
int quantity = Integer.parseInt(data.get("quantity").toString());
String uen = data.get("uen").toString();
double total_price = Double.parseDouble(data.get("total_price").toString());
```

Valid JSON input example

{
“swp_id”: “example_swp_id”,
“uen”: “example_uen”,
“quantity”: x,
“total_price”: yy.yy,
“price”: zz.zz
}

Description: - Only one item is added to the cart at a time. - “swp_id” refers to the wholesaler product ID of the item, “uen” refers to the wholesaler’s UEN, “quantity” refers to the number of the item being added to the cart,

"total_price" refers to the total price of the items being added to the cart and "price" refers to the unit price of the item being added to the cart.

2.1.2 Test Case Descriptions

Test id	Description
2.1a	1. Valid JSON input that represents the addition of one product to cart without any discrepancies for any fields. ○ The product must have a positive quantity. ○ Valid authorization key (consumer ID, typical case). 2. A cart record for the consumer does not exist in FireBase (i.e., this is the first time the consumer is adding a product to cart). 3. After calling addToCart(), a new transaction record in FireBase will be created with the status "IN-CART". ○ The transaction record will reflect the new product added. 4. After calling addToCart(), a new shopping cart record in FireBase will be created. ○ The shopping cart record will reflect the new product added. 5. HTTP Status of 201 will be received
2.1b	1. Valid JSON input that represents the addition of one product to cart without any discrepancies for any fields. ○ The product must have a positive quantity. ○ Valid authorization key (consumer ID, typical case). 2. A cart record for the consumer already exists in FireBase (i.e., the consumer has added a product to cart before). 3. After calling addToCart(), the transaction records in FireBase for the consumer will be updated. ○ The transaction record will reflect the new product added. 4. After calling addToCart(), the shopping cart record in FireBase will be updated. ○ The shopping cart record will reflect the new product added. 5. HTTP Status of 201 will be received
2.1c	1. Invalid JSON input that represents the addition of one product to cart with a discrepancy in the "quantity" field only. ○ The product has a negative or zero quantity. 2. HTTP Status of 400 will be received 3. No updates will be made to the consumer's shopping cart and transaction record in FireBase
2.1d	1. Invalid JSON input that represents the addition of one product to cart with a discrepancy in the "swp_id" field only. ○ The wholesaler product represented by the swp_id does not exist. 2. HTTP Status of 400 will be received 3. No updates will be made to the consumer's shopping cart and transaction record in FireBase

2.1.3 JUnit Test Results

	Test Script	backend/src/test/java/com/peaslimited/shoppeas/controller/addToCartTest.java		
Test id	Basis Path	**Basis Path Testing** Inputs	Expected output	Actual output
		JSON Input Data		

2.1a	1,2,3,5,8,9,11	{
		"swp_id": "3q1VqrKuNAkpFxIvziTd",
		"uen": "53474231M",
		"quantity": 10,
		"total_price": 304.6,
		"price": 30.46
		}
		● Default correct JSON input (shown above)
		● Cart record does not exist for the user
			201
			Cart and transaction records in FireBase reflect the new product added.
				201
				Cart and transaction records in FireBase reflect the new product added.

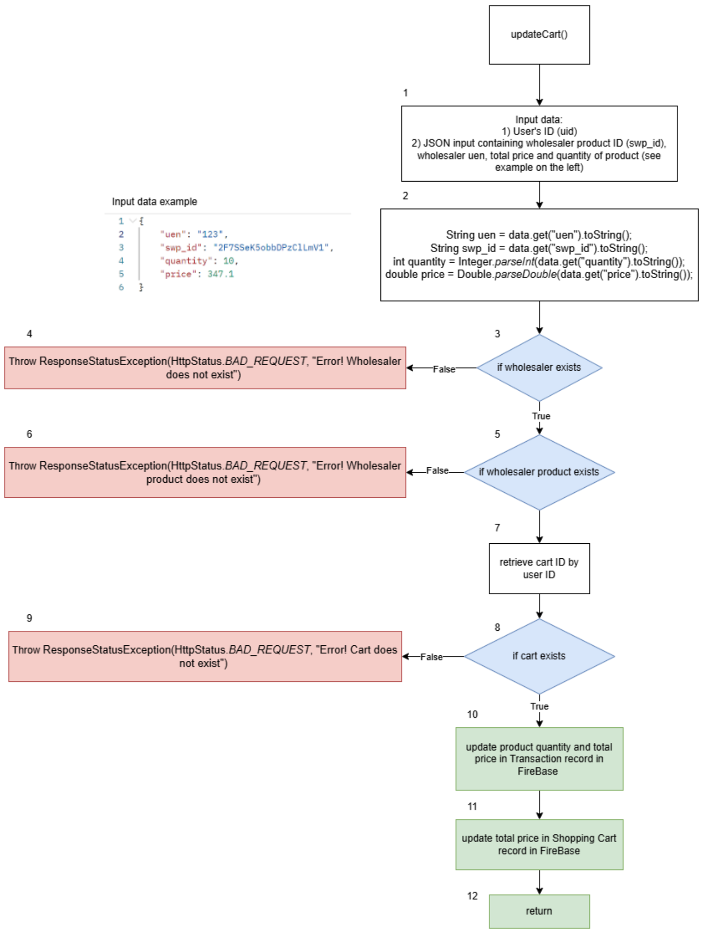
2.1b	1,2,3,5,8,10,11	{
		"swp_id": "3q1VqrKuNAkpFxIvziTd",
		"uen": "53474231M",
		"quantity": 10,
		"total_price": 304.6,
		"price": 30.46
		}
		● Default correct JSON input (shown above)
		● Cart record already exists for the user
			201
			Cart and transaction records in FireBase reflect the new product added.
				201
				Cart and transaction records in FireBase reflect the new product added.

2.1c	1,2,3,4	{
		"swp_id": "3q1VqrKuNAkpFxIvziTd",
		"uen": "53474231M",
		"quantity": -10,
		"total_price": 304.6,
		"price": 30.46
		}
		● Invalid JSON input (shown above)
		● Quantity of product is less than or equal to zero
			400
			Throws Response Status Exception: Error! Invalid quantity
				400
				Throws Response Status Exception: Error! Invalid quantity

2.1d	1,2,3,5,6	{
		"swp_id": "123",
		"uen": "53474231M",
		"quantity": -10,
		"total_price": 304.6,
		"price": 30.46
		}
		● Invalid JSON input (shown above)
		● swp_id is invalid (wholesaler product does not exist)
			400
			Throws Response Status Exception: Error! Wholesaler product does not exist
				400
				Throws Response Status Exception: Error! Wholesaler product does not exist

2.2 updateCart():

2.2.1 Control Flow Graph and Input Data


updateCart()
1
Input data:
1) User's ID (uid)
2) JSON input containing wholesaler product ID (swp_id), wholesaler uen, total price and quantity of product (see example on the left)
2
3
if wholesaler exists
False
4
Throw ResponseStatusException(HttpStatus.BAD_REQUEST, "Error! Wholesaler does not exist")
True
5
if wholesaler product exists
False
6
Throw ResponseStatusException(HttpStatus.BAD_REQUEST, "Error! Wholesaler product does not exist")
7
retrieve cart ID by user ID
8
if cart exists
False
9
Throw ResponseStatusException(HttpStatus.BAD_REQUEST, "Error! Cart does not exist")
True
10
update product quantity and total price in Transaction record in FireBase
11
update total price in Shopping Cart record in FireBase
12
return

Input data example

```
{
    "uen": "123",
    "swp_id": "2F7SSeK5obbDPzClLmV1",
    "quantity": 10,
    "price": 347.1
}
```

```
String uen = data.get("uen").toString();
String swp_id = data.get("swp_id").toString();
int quantity = Integer.parseInt(data.get("quantity").toString());
double price = Double.parseDouble(data.get("price").toString());
```

Valid JSON input example

{
“swp_id”: “example_swp_id”,
“uen”: “example_uen”,
“quantity”: x,
“price”: yy.yy
}

Description: - Only one type of item in the cart is updated at a time. - “swp_id” refers to the wholesaler product ID of the item, “uen” refers to the wholesaler’s UEN, “quantity” refers to the number of the item being added to the cart, and “price” refers to the total updated price of the additional items being added

2.2.2 Test Case Descriptions

Test id	Description
2.2a	1. Valid JSON input that represents the update of one product in cart without any discrepancies for any fields. ○ Product quantity and price can be positive or negative. ■ A positive product quantity/ price represents the additional quantity/ price of the additional products being added to cart. ■ A negative product quantity/ price represents the quantity/ price of products being subtracted from existing products in cart. ○ Valid authorization key (consumer ID, typical case). 2. After calling updateCart(),the respective transaction and shopping cart records will be updated and will reflect the new changes. 3. HTTP Status of 204 will be received.
2.2b	1. Invalid JSON input that represents the update of one product to cart with a discrepancy in the “uen” field only. ○ The wholesaler with the corresponding uen does not exist. 2. HTTP Status of 400 will be received. 3. No updates will be made to the consumer’s shopping cart and transaction record in FireBase.
2.2c	1. Invalid JSON input that represents the update of one product to cart with a discrepancy in the “swp_id” field only. ○ The wholesaler product with the corresponding swp_id does not exist. 2. HTTP Status of 400 will be received 3. No updates will be made to the consumer’s shopping cart and transaction record in FireBase
2.2d	1. Valid JSON input that represents the update of one product in cart without any discrepancies for any fields. 2. Incorrect authorization key (consumer ID) ○ The user with the corresponding uid does not have a cart record in FireBase (i.e., there is nothing that can be updated). 3. HTTP Status of 400 will be received 4. No updates will be made to the consumer’s shopping cart and transaction record in FireBase

2.2.3 JUnit Test Results

	Test Script	backend/src/test/java/com/peaslimited/shoppeas/controller/updateCartTest.java		
Test id	Basis Path	**Basis Path Testing**	Expected output	Actual output
		Inputs		

		JSON Input Data		

2.2a	1,2,3,5,7,8,
	10,11,12
		{
		"uen": "53474231M",
		"swp_id": "2F7SSeK5obbDPzClLmV1",
		"quantity": 15,
		"price": 520.65
		}
		
		● Default correct JSON input (shown above)
			204
			
			Cart and transaction records in FireBase reflect the updated product.
				204
				
				Cart and transaction records in FireBase reflect the updated product.

2.2b	1,2,3,4	{
		"uen": "123",
		"swp_id": "2F7SSeK5obbDPzClLmV1",
		"quantity": 15,
		"price": 520.65
		}
		
		● Invalid JSON input (shown above)
		● uen is invalid (wholesaler does not exist)
			400
			
			Throws Response Status Exception: Error! Wholesaler does not exist
				400
				
				Throws Response Status Exception: Error! Wholesaler does not exist

2.2c	1,2,3,5,6	{
		"uen": "53474231M",
		"swp_id": "123",
		"quantity": 15,
		"price": 520.65
		}
		
		● Invalid JSON input (shown above)
		● swp_id is invalid (wholesaler product does not exist)
			400
			
			Throws Response Status Exception: Error! Wholesaler product does not exist
				400
				
				Throws Response Status Exception: Error! Wholesaler product does not exist

2.2d	1,2,3,5,7,8,9	{
		"uen": "53474231M",
		"swp_id": "2F7SSeK5obbDPzClLmV1",
		"quantity": 15,
		"price": 520.65
		}
		
		● Default correct JSON input (shown above)
		● Incorrect authorization key (consumer ID)
		○ The user with the corresponding uid does not have a cart record in FireBase
			400
			
			Throws Response Status Exception: Error! Cart does not exist
				400
				
				Throws Response Status Exception: Error! Cart does not exist