Test case design

Test number	Test name	Description	Test data	Expected result
1	Valid token	Check that given token results in the right permissions.	rep, storekeeper, shipper	rep, storekeeper, shipper
2	Invalid token	Check that when given an invalid token. No rights are given.	Dhslkjh or some other random string	«» (empty string)
3	Model filter	Retrieve a list of skis with model filter	URI has query: ?model=redline	2 skis. One with size 140 and the other with size 150
4	Grip-system filter	Retrieve a list of skis with grip-system filter	URI has query: ?grip=wax	3 skis. Frogger with size 150 and 2 Reline skis of sizes 140 and 150
5	Add new ski	Add a new ski to the database	[“model”: “Redline” “size”: 150, “weight”: “40-50”]	The new model exists in the database. Endpoint returns the added ski and returns the status code 200.

6	Add invalid ski	Trying to add an invalid resource to the database	[“model”: “Redline”]	Code 400.
7	Get production plan	Check that you get the production plan you request	GET request to URI: “customer/production-plan/3”	An array with keys result, id, start_date, end_date, plans and status. Status = 200. ID = 3

8	Create valid order	Checking that an order is created properly	<pre>{
			"customer_id": 1,
			"types": [
			{
			"weight": "20-30",
			"size": 140,
			"model": "Redline",
			"quantity": 3
			},
			{
			"weight": "20-30",
			"size": 145,
			"model": "Fisher",
			"quantity": 1
			}
			]
			}</pre>
				<pre>{
				"order_number": 4,
				"total_price": 16300,
				"customer_id": 1
				}</pre>Status = 201

9	Create invalid order	Check correct response for an invalid order creation	<pre>{
			"customer_id": 1,
			"types": [
			{</pre>
				Status = 400

			"weight": "20-30",
			"size": 140,
			"model": "Red",
			"quantity": 1
			},
			{
			"weight": "20-30",
			"size": 145,
			"model": "Fisher",
			"quantity": 1
			}
			]
			}	

10	Delete order	Check that an order has been deleted properly	DELETE request to URI: "customer/order/2"
			Body:
			{
			"customer_id": 1
			}
				Status = 200

11	Delete non existing order	Check proper response for a delete request of a non-existing order	DELETE request to URI: "customer/order/4"
			Body:
			{
			"customer_id": 1
			}
				Status = 403
				It is irrelevant for the user to know that it is non-existent (404). Regardles of the orders existence; the user is unauthorized to delete it.

12	Retrieve new orders	Request to customer rep endpoint for new orders	GET request to URI: “rep/orders?state=new”	3 results. Order numbers 1, 2 and 3. Status = 200.
13	Change state	Customer rep set state of order from new to open	PATCH request to URI: “rep/order/open/1”	State = open id = 1 Status = 200
14	Retrieve available skis	Customer rep get skis with state skis-available	GET request to URI: “rep/orders?state=skis-available”	Status = 200 0 results
15	Retrieve skis in a non-existent state	Customer rep retrieve skis with state dsfghk	GET request to URI: “rep/orders?state= dsfghk”	Status = 400
16	Set shipment state to shipped	Shipper endpoint set state of a shipment to shipped	PATCH request to URI: “shipper/ship/2”	Status = 200 Returns number = 2, state = shipped
17	Set shipment state of non-existent shipment	Check for proper response when trying to set state of non-existent shipment	PATCH request to URI: “shipper/ship/4”	Status = 404