Technological Institute of Costa Rica

Computer Engineering School

Project Case Specification for Dynamic Testing

Software Quality Assurance

I-2024

Professor: Erick Hernández Bonilla

Members:

María Paula Bolaños Apú

Marco Herrera González

Jeffrey Leiva Cascante

Content

Revision Log

Date	Version	Changes
April 20, 2024.	1.0.	Initial draft released for review.

Overview

Introduction

This document details the Test Coverage Items and test cases for Project #1 of the Software Design Course whose objective is the development of a web system for image publications and sales of beauty and skincare products. The objective of the test case specification is to provide the Test Coverage Items and the test cases based on the test models that require dynamic testing.

Scope

This document covers the test case specifications for the test models of the purchase module, necessary for the application of dynamic tests.

References

Internal

1. Test Model Specification for Dynamic Testing.
2. Source Code of the Project

Test Coverage Items for Test Model #1

Name	Identifier	Description	Priority	Traceability
Test Coverage Item – 1	TCI1_TM1	Test coverage for in range product number inputs.	2	Maps to Test Model #1
Test Coverage Item – 2	TCI2_TM1	Test coverage for too low product number inputs.	2	Maps to Test Model #1
Test Coverage Item – 3	TCI3_TM1	Test coverage for too high product number inputs.	2	Maps to Test Model #1
Test Coverage Item – 4	TCI4_TM1	Test coverage for invalid product number inputs.	2	Maps to Test Model #1
Test Coverage Item – 5	TCI5_TM1	Test coverage for generating order with empty cart.	2	Maps to Test Model #1
Test Coverage Item – 6	TCI6_TM1	Test coverage for generating order with products in cart.	2	Maps to Test Model #1
Test Coverage Item – 7	TCI7_TM1	Test coverage for generating order with products and handle an error when registering the order.	2	Maps to Test Model #1

Test Coverage Items for Test Model #2

Name	Identifier	Description	Priority	Traceability
Test Coverage Item – 1	TCI1_TM2	Test coverage for getting a cart with an invalid user.	1	Maps to Test Model #2

Test Coverage Item – 2	TCI2_TM2	Test coverage for getting a cart with valid user.	1	Maps to Test Model #2
Test Coverage Item – 3	TCI3_TM2	Test coverage for getting a cart in valid time (1 to 2 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2
Test Coverage Item – 4	TCI4_TM2	Test coverage for getting a cart in an invalid time (> 12 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2
Test Coverage Item – 5	TCI5_TM2	Test coverage for adding a product with an invalid user.	1	Maps to Test Model #2
Test Coverage Item – 6	TCI6_TM2	Test coverage for adding a product with valid user.	1	Maps to Test Model #2
Test Coverage Item – 7	TCI7_TM2	Test coverage for adding a product in valid time (1 to 2 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2
Test Coverage Item – 8	TCI8_TM2	Test coverage for adding a product in an invalid time (> 12 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2
Test Coverage Item – 9	TCI9_TM2	Test coverage for deleting a product with an invalid user.	1	Maps to Test Model #2
Test Coverage Item – 10	TCI10_TM2	Test coverage for deleting a product with valid user.	1	Maps to Test Model #2
Test Coverage Item – 11	TCI11_TM2	Test coverage for deleting a product in valid time (1 to 2 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2
Test Coverage Item – 12	TCI12_TM2	Test coverage for deleting a product in an invalid time (> 12 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2
Test Coverage Item – 13	TCI13_TM2	Test coverage for updating a product with an invalid user.	1	Maps to Test Model #2
Test Coverage Item – 14	TCI14_TM2	Test coverage for updating a product with valid user.	1	Maps to Test Model #2
Test Coverage Item – 15	TCI15_TM2	Test coverage for updating a product in valid time (1 to 2 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2

Test Coverage Item – 16	TCI16_TM2	Test coverage for updating a product in an invalid time (> 12 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2
Test Coverage Item – 17	TCI17_TM2	Test coverage for finding a product with an invalid user.	1	Maps to Test Model #2
Test Coverage Item – 18	TCI18_TM2	Test coverage for finding a product with valid user.	1	Maps to Test Model #2
Test Coverage Item – 19	TCI19_TM2	Test coverage for finding a product in valid time (1 to 2 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2
Test Coverage Item – 20	TCI20_TM2	Test coverage for finding a product in an invalid time (> 12 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2
Test Coverage Item – 21	TCI21_TM2	Test coverage for registering an order with valid user.	1	Maps to Test Model #2
Test Coverage Item – 22	TCI22_TM2	Test coverage for registering an order with an invalid user.	1	Maps to Test Model #2
Test Coverage Item – 23	TCI23_TM2	Test coverage for registering an order in valid time (1 to 2 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2
Test Coverage Item – 24	TCI24_TM2	Test coverage for registering an order in an invalid time (> 12 seconds) for 998 users and 2 admins at the same time.	1	Maps to Test Model #2

Test Coverage Items for Test Model #3

Name	Identifier	Description	Priority	Traceability
Test Coverage Item – 1	TCI1_TM3	Test coverage for the correct rendering of the user's cart data on the component when the data is queried correctly.	1	Maps to Test Model #3
Test Coverage Item – 2	TCI2_TM3	Test coverage for the rendering of the component when the data is not queried correctly.	1	Maps to Test Model #3
Test Coverage Item – 3	TCI3_TM3	Test coverage for the re-rendering of the component when increasing the quantity of a product on the user's cart when the data is queried correctly.	1	Maps to Test Model #3

Test Coverage Item – 4	TCI4_TM3	Test coverage for the re-rendering of the component when increasing the quantity of a product on the user's cart when there is an error.	1	Maps to Test Model #3
Test Coverage Item – 5	TCI5_TM3	Test coverage for the re-rendering of the component when decreasing the quantity of a product on the user's cart when the data is queried correctly.	1	Maps to Test Model #3
Test Coverage Item – 6	TCI6_TM3	Test coverage for the re-rendering of the component when decreasing the quantity of a product on the user's cart when there is an error.	1	Maps to Test Model #3
Test Coverage Item – 7	TCI7_TM3	Test coverage for the correct functionality of the function that handles the backend responses in case there was an error.	1	Maps to Test Model #3
Test Coverage Item – 8	TCI8_TM3	Test coverage for the correct functionality of the function that handles the backend responses in the case that the result is not an error.	1	Maps to Test Model #3
Test Coverage Item – 9	TCI9_TM3	Test coverage for the correct calculation of the cart's price after querying the user's cart data.	1	Maps to Test Model #3
Test Coverage Item – 10	TCI10_TM3	Test coverage of the reading of the user's cart data in a valid time (0 to 5 seconds).	1	Maps to Test Model #3
Test Coverage Item – 11	TCI11_TM3	Test coverage of increasing the quantity of a product in a valid time (0 to 5 seconds).	1	Maps to Test Model #3
Test Coverage Item – 12	TCI12_TM3	Test coverage of decreasing the quantity of a product in a valid time (0 to 5 seconds).	1	Maps to Test Model #3

Test Coverage Items for Test Model #4

Name	Identifier	Description	Priority	Traceability
Test Coverage Item – 1	TCI1_TM4	Test coverage for the correct rendering of the user's cart data on the component when the data is queried correctly.	1	Maps to Test Model #4
Test Coverage Item – 2	TCI2_TM4	Test coverage for the rendering of the component when the data is not queried correctly.	1	Maps to Test Model #4
Test Coverage Item – 3	TCI3_TM4	Test coverage for the rendering of the correct calculation of the cart's price after querying the user's cart data.	1	Maps to Test Model #4
Test Coverage Item – 4	TCI4_TM4	Test coverage for the correct rendering of the cantons on the component when selecting a province by using the *updateCantons()* internal function.	1	Maps to Test Model #4

Test Coverage Item – 5	TCI5_TM4	Test coverage for the correct rendering of the cantons on the component when selecting a province by using the *updateDistricts()* internal function.	1	Maps to Test Model #4
Test Coverage Item – 6	TCI6_TM4	Test coverage for the correct rendering of the component when the *sendForm*() function is called with all the input needed.	1	Maps to Test Model #4
Test Coverage Item – 7	TCI7_TM4	Test coverage for the correct rendering of the component when the *sendForm*() function is called and there is an error.	1	Maps to Test Model #4
Test Coverage Item – 8	TCI8_TM4	Test coverage for the correct functionality of the function that handles the backend responses in the case that the result is not an error.	1	Maps to Test Model #4
Test Coverage Item – 9	TCI9_TM4	Test coverage for the correct functionality of the function that handles the backend responses in case there was an error.	1	Maps to Test Model #4
Test Coverage Item – 10	TCI10_TM4	Test coverage of the reading of the user's cart data in a valid time (0 to 5 seconds).	1	Maps to Test Model #4
Test Coverage Item – 11	TCI11_TM4	Test coverage of response time of the *updateCantons*() function in a valid time (0 to 5 seconds).	1	Maps to Test Model #4
Test Coverage Item – 12	TCI12_TM4	Test coverage of response time of the *updateDistricts*() function in a valid time (0 to 5 seconds)	1	Maps to Test Model #4
Test Coverage Item – 13	TCI13_TM4	Test coverage of response time of the *sendForm*() function in a valid time (0 to 5 seconds)	1	Maps to Test Model #4

Test Cases Based on Coverage Items for the Test Model #1

Test Case ID: 1 **Priority**: 2 **Test Coverage Item**: TCI1_TM1 **Purpose**: To verify that the function accepts in range product number inputs.	
Preconditions	A CartDao class stub.
Inputs	Units = 0…5.
Expected outputs	The product is added to the cart.

Test Case ID: 2 **Priority**: 2 **Test Coverage Item**: TCI2_TM1 **Purpose**: To verify that the function rejects too low product number inputs.	
Preconditions	A CartDao class stub.
Inputs	Units = <0.
Expected outputs	Raises an exception.

Test Case ID: 3 **Priority**: 2 **Test Coverage Item**: TCI3_TM1 **Purpose**: To verify that the function rejects too high product number inputs.	
Preconditions	A CartDao class stub.
Inputs	Units = >5.
Expected outputs	Raises an exception.

Test Case ID: 4 **Priority**: 2 **Test Coverage Item**: TCI4_TM1 **Purpose**: To verify that the function rejects invalid products number inputs.	
Preconditions	A CartDao class stub.
Inputs	Unit = “Cinco”.
Expected outputs	Raises an exception.

Test Case ID: 5 **Priority**: 2 **Test Coverage Item**: TCI5_TM1 **Purpose**: To verify that the function rejects generate order with empty cart.	
Preconditions	A CartDao class stub.
Inputs	The function receives the parameters: userId, address, totalPrice and photoPath.
Expected outputs	Raises an exception.

Test Case ID: 6 **Priority**: 2 **Test Coverage Item**: TCI6_TM1 **Purpose**: To verify that the function accepts generate order with products in cart.	
Preconditions	CartDao Stub.
Inputs	The function receives the parameters: userId, address, totalPrice and photoPath.
Expected outputs	The order is generated.

Test Case ID: 7 **Priority**: 2 **Test Coverage Item**: TCI7_TM1 **Purpose**: To verify that the function accepts generate order with products in cart and handle error when registering the order.	
Preconditions	Database stub.
Inputs	The function receives the parameters: userId, address, totalPrice and photoPath.
Expected outputs	Tries to generate the order but raises an exception.

Test Cases based on coverage items for the Test Model #2

Test Case ID: 8 **Priority**: 1 **Test Coverage Item**: TCI1_TM2 **Purpose**: To verify that the function rejects to get a cart with an invalid user.	
Preconditions	A database stub.
Inputs	The function receives an userId.
Expected outputs	Raises an exception.

Test Case ID: 9 **Priority**: 1 **Test Coverage Item**: TCI2_TM2 **Purpose**: To verify that the function accepts to get a cart with valid user.	
Preconditions	A database stub.
Inputs	The function receives an userId.
Expected outputs	The cart is returned successfully.

Test Case ID: 10 **Priority**: 1 **Test Coverage Item**: TCI3_TM2 **Purpose**: To verify that the function accomplishes to get a cart in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	The carts are returned successfully.

Test Case ID: 11 **Priority**: 1 **Test Coverage Item**: TCI4_TM2 **Purpose**: To verify that the function doesn't accomplish to get a cart in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	Raises an exception.

Test Case ID: 12 **Priority**: 1 **Test Coverage Item**: TCI5_TM2 **Purpose**: To verify that the function rejects to add a cart with an invalid user.	
Preconditions	A database stub.
Inputs	The function receives an invalid userId.
Expected outputs	Raises Exception.

Test Case ID: 13 **Priority**: 1 **Test Coverage Item**: TCI6_TM2 **Purpose**: To verify that the function accepts to add a cart with valid user.	
Preconditions	A database stub.
Inputs	The function receives a valid userId.
Expected outputs	The cart is added successfully.

Test Case ID: 14 **Priority**: 1 **Test Coverage Item**: TCI7_TM2 **Purpose**: To verify that the function accomplishes to add a cart in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	The carts are added successfully.

Test Case ID: 15 **Priority**: 1 **Test Coverage Item**: TCI8_TM2 **Purpose**: To verify that the function doesn't accomplish to add a cart in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	Raises an exception.

Test Case ID: 16 **Priority**: 1 **Test Coverage Item**: TCI9_TM2 **Purpose**: To verify that the function rejects to delete a cart with an invalid user.	
Preconditions	A database stub.
Inputs	The function receives an invalid userId.
Expected outputs	Raises an exception.

Test Case ID: 17 **Priority**: 1 **Test Coverage Item**: TCI10_TM2 **Purpose**: To verify that the function accepts to delete a cart with valid user.	
Preconditions	A database stub.
Inputs	The function receives an userId.
Expected outputs	The cart is deleted successfully.

Test Case ID: 18 **Priority**: 1 **Test Coverage Item**: TCI11_TM2 **Purpose**: To verify that the function accomplishes to delete a cart in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	The carts are deleted successfully.

Test Case ID: 19 **Priority**: 1 **Test Coverage Item**: TCI12_TM2 **Purpose**: To verify that the function doesn't accomplish to delete a cart in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	Raises an exception.

Test Case ID: 20 **Priority**: 1 **Test Coverage Item**: TCI13_TM2 **Purpose**: To verify that the function rejects to update a cart with an invalid user.	
Preconditions	A database stub.
Inputs	The function receives an invalid userId.
Expected outputs	Raises Exception.

Test Case ID: 21 **Priority**: 1 **Test Coverage Item**: TCI14_TM2 **Purpose**: To verify that the function accepts to update a cart with valid user.	
Preconditions	A database stub.
Inputs	The function receives an userId.
Expected outputs	The update is completed successfully.

Test Case ID: 22 **Priority**: 1 **Test Coverage Item**: TCI15_TM2 **Purpose**: To verify that the function accomplishes to update a cart in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	All updates are completed successfully.

Test Case ID: 23 **Priority**: 1 **Test Coverage Item**: TCI16_TM2 **Purpose**: To verify that the function doesn't accomplish to update a cart in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	Raises an exception.

Test Case ID: 24 **Priority**: 1 **Test Coverage Item**: TCI17_TM2 **Purpose**: To verify that the function rejects to find a cart with an invalid user.	
Preconditions	A database stub.
Inputs	The function receives an invalid userId.
Expected outputs	Raises an exception.

Test Case ID: 25 **Priority**: 1 **Test Coverage Item**: TCI18_TM2 **Purpose**: To verify that the function accepts to find a cart with valid user.	
Preconditions	A database stub.
Inputs	The function receives an userId.
Expected outputs	The cart is found successfully

Test Case ID: 26 **Priority**: 1 **Test Coverage Item**: TCI19_TM2 **Purpose**: To verify that the function accomplishes to find a cart in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	All carts are found successfully.

Test Case ID: 27 **Priority**: 1 **Test Coverage Item**: TCI20_TM2 **Purpose**: To verify that the function doesn't accomplish to find a cart in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	Raises an exception.

Test Case ID: 28 **Priority**: 1 **Test Coverage Item**: TCI21_TM2 **Purpose**: To verify that the function rejects to register an order with an invalid user.	
Preconditions	A database stub.
Inputs	The function receives an invalid userId.
Expected outputs	Raises an exception.

Test Case ID: 29 **Priority**: 1 **Test Coverage Item**: TCI22_TM2 **Purpose**: To verify that the function accepts to register an order with valid user.	
Preconditions	A database stub.
Inputs	The function receives an userId.
Expected outputs	The order is registered successfully.

Test Case ID: 30 **Priority**: 1 **Test Coverage Item**: TCI23_TM2 **Purpose**: To verify that the function accomplishes to register an order in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	All the orders are registered successfully.

Test Case ID: 31 **Priority**: 1 **Test Coverage Item**: TCI24_TM2 **Purpose**: To verify that the function doesn't accomplish to register an order in valid time (1 to 2 seconds).	
Preconditions	A database stub, 998 users and 2 admins at the same time.
Inputs	None.
Expected outputs	Raises an exception.

Test Cases based on Test Coverage Items for the Test Model #3

Test Case ID: 32 **Priority**: 1 **Test Coverage Item**: TCI1_TM3 **Purpose**: To verify that the name of the products of the data is rendered correctly in the component when queried from the backend in a correct manner.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The user's products names are present in the component as expected.

Test Case ID: 33 **Priority**: 2 **Test Coverage Item**: TCI1_TM3 **Purpose**: To verify that the units of each product in the user's cart data is rendered correctly in the component when queried from the backend in a correct manner.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	Each product units are present in the component as expected.

Test Case ID: 34 **Priority**: 2 **Test Coverage Item**: TCI1_TM3 **Purpose**: To verify that the individual price for each product of the user's cart data is rendered correctly in the component when queried from the backend in a correct manner.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The price for each product is present in the component as expected.

Test Case ID: 35 **Priority**: 1 **Test Coverage Item**: TCI1_TM3 **Purpose**: To verify that the accumulated price for each product of the user's cart data based on individual price and quantity is rendered correctly in the component when queried from the backend in a correct manner.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The accumulated price for each product based on its units and individual price is present in the component as expected.

Test Case ID: 36 **Priority**: 1 **Test Coverage Item**: TCI2_TM3 **Purpose**: To verify that the component doesn't render any product names when there is an error on the backend response.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	There user's product names are not present in the component after rendering.

Test Case ID: 37 **Priority**: 2 **Test Coverage Item**: TCI2_TM3 **Purpose**: To verify that the component doesn't render any product units when there is an error on the backend response.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The user's product units are not present in the component after rendering.

Test Case ID: 38 **Priority**: 2 **Test Coverage Item**: TCI2_TM3 **Purpose**: To verify that the component doesn't render any product individual price when there is an error on the backend response.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The user's product individual prices are not present in the component after rendering.

Test Case ID: 39 **Priority**: 2 **Test Coverage Item**: TCI2_TM3 **Purpose**: To verify that the component doesn't render each product accumulated price when there is an error on the backend response.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The user's product accumulated prices are not present in the component.

Test Case ID: 40 **Priority**: 41 **Test Coverage Item**: TCI3_TM3 **Purpose**: To verify that when the user adds a product to the cart, the component updates the quantity of said product by reloading the window.	
Preconditions	A backend stub.
Inputs	We make a click on the button to add a product, the component makes a request to the backend stub.
Expected outputs	The window is reloaded to increase the products quantity.

Test Case ID: 42 **Priority**: 2 **Test Coverage Item**: TCI4_TM3 **Purpose**: To verify that when the user tries to add a product to the cart and the response has an error, the component changes its state and renders the error via its modal.	
Preconditions	A backend stub.
Inputs	We make a click on the button to add a product, the component makes a request to the backend stub.
Expected outputs	The state of the component must have changed, and the error message is rendered in it.

Test Case ID: 43 **Priority**: 2 **Test Coverage Item**: TCI5_TM3 **Purpose**: To verify that when the user deletes a product from the cart, the component updates the quantity of said product by reloading the window.	
Preconditions	A backend stub.
Inputs	We make a click on the button to delete a product, the component makes a request to the backend stub.
Expected outputs	The window is reloaded to decrease the products quantity.

Test Case ID: 44 **Priority**: 2 **Test Coverage Item**: TCI6_TM3 **Purpose**: To verify that when the user tries to delete a product from the cart and the response has an error, the component changes its state and renders the error via its modal.	
Preconditions	A backend stub.
Inputs	We make a click on the button to decrease a product, the component makes a request to the backend stub.
Expected outputs	The state of the component must have changed, and the error message is rendered in it.

Test Case ID: 45 **Priority**: 1 **Test Coverage Item**: TCI7_TM3 **Purpose**: To verify that when the user makes a query to the backend that returns an error, the function that handles the response sets the error state variable correctly.	
Preconditions	A backend stub.
Inputs	We make a click on the button to decrease a product, the component makes a request to the backend stub.
Expected outputs	The error variable state is set to true in the component, the message “Ha ocurrido un error” is shown in the modal of the component.

Test Case ID: 46 **Priority**: 1 **Test Coverage Item**: TCI7_TM3 **Purpose**: To verify that when the user makes a query to the backend that returns an error, the function that handles the response sets the showModal state variable correctly.	
Preconditions	A backend stub.
Inputs	We make a click on the button to decrease a product, the component makes a request to the backend stub.
Expected outputs	The showModal variable state is set to true in the component, the message “true” is present in the modal of the component.

Test Case ID: 47 **Priority**: 1 **Test Coverage Item**: TCI8_TM3 **Purpose**: To verify that when the user makes a query to the backend that returns correctly, the function that handles the response sets the error state variable correctly.	
Preconditions	A backend stub.

Inputs	We make a click on the button to decrease a product, the component makes a request to the backend stub.
Expected outputs	The error state variable is set to false in the component, the message “false” is present in the component’s modal.

Test Case ID: 48 **Priority**: 1 **Test Coverage Item**: TCI9_TM3 **Purpose**: To verify that when the user’s cart data is queried correctly, the function that calculates the price does so correctly, and the price is present in the component.	
Preconditions	A backend stub.
Inputs	The component makes a call to the backend stub.
Expected outputs	The cartPrice state variable is set to the correct value according to the product’s prices and their quantity, the total price is present in the component.

Test Case ID: 49 **Priority**: 1 **Test Coverage Item**: TCI10_TM3 **Purpose**: To verify that when querying for a user’s cart data, it does so in a valid time (0 to 5 seconds)	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub upon rendering.
Expected outputs	The response is returned in 0 to 5 seconds for each user.

Test Case ID: 50 **Priority**: 1 **Test Coverage Item**: TCI11_TM3 **Purpose**: To verify that when a user increments the quantity of a product in the cart, it does so in a valid time (0 to 5 seconds)	
Preconditions	A backend stub.
Inputs	We make a click on the button to increase the product, and the component makes a request to the backend stub.
Expected outputs	The response is returned in 0 to 5 seconds.

Test Case ID: 51 **Priority**: 1 **Test Coverage Item**: TCI12_TM3 **Purpose**: To verify that when a user decrements the quantity of a product in the cart, it does so in a valid time (0 to 5 seconds)

Preconditions	A backend stub
Inputs	We make a click on the button to decrease the product, and the component makes a request to the backend stub.
Expected outputs	The response is returned in 0 to 5 seconds.

Test Cases based on the Test Coverage Items for Test Model #4

Test Case ID: 52 **Priority**: 1 **Test Coverage Item**: TCI1_TM4 **Purpose**: To verify that the name of the products of the data is rendered correctly in the component when queried from the backend in a correct manner.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The user's products names are present in the component as expected.

Test Case ID: 53 **Priority**: 2 **Test Coverage Item**: TCI1_TM4 **Purpose**: To verify that the units of each product in the user's cart data is rendered correctly in the component when queried from the backend in a correct manner.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	Each product units are present in the component as expected.

Test Case ID: 54 **Priority**: 2 **Test Coverage Item**: TCI1_TM4 **Purpose**: To verify that the accumulated price for each product of the user's cart data based on individual price and quantity is rendered correctly in the component when queried from the backend in a correct manner.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The accumulated price for each product based on its units and individual price is present in the component as expected.

Test Case ID: 55 **Priority**: 2 **Test Coverage Item**: TCI1_TM4 **Purpose**: To verify that when the component renders, the provinces are present in it for the user to select.	
Preconditions	Existence of the provinces in a previously created file.
Inputs	None.
Expected outputs	The provinces names are present in the rendered component.

Test Case ID: 56 **Priority**: 2 **Test Coverage Item**: TCI2_TM4 **Purpose**: To verify that the component doesn't render any product names when there is an error on the backend response.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	There user's product names are not present in the component after rendering.

Test Case ID: 57 **Priority**: 1 **Test Coverage Item**: TCI2_TM4 **Purpose**: To verify that the component doesn't render any product units when there is an error on the backend response.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The user's product units are not present in the component after rendering

Test Case ID: 58 **Priority**: 1 **Test Coverage Item**: TCI2_TM4 **Purpose**: To verify that the component doesn't render the accumulated price of each product based on units and individual price when there is a backend error	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The user's product accumulated prices are not present in the component.

Test Case ID: 59 **Priority**: 1 **Test Coverage Item**: TCI3_TM4 **Purpose**: To verify that the component render's the total price of the cart without the extra fee for delivery after successfully querying the user's cart data.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The total price of the cart without the delivery fee is present in the component.

Test Case ID: 60 **Priority**: 1 **Test Coverage Item**: TCI3_TM4 **Purpose**: To verify that the component render's the total price of the cart with the extra fee for delivery after successfully querying the user's cart data.	
Preconditions	A backend stub.
Inputs	The component makes a request to the backend stub.
Expected outputs	The total price of the cart with the delivery fee is present in the component.

Test Case ID: 61 **Priority**: 1 **Test Coverage Item**: TCI4_TM4 **Purpose**: To verify that the component renders the correct cantons when selecting a province via the *updateCantons*() function.	
Preconditions	Existence of the locations in a predefined file.
Inputs	We select a province of the ones provided by the component.
Expected outputs	The cantons state variable is set to the selected province's corresponding cantons and are displayed on the component.

Test Case ID: 62 **Priority**: 1 **Test Coverage Item**: TCI5_TM4 To verify that the component renders the correct districts when selecting a canton via the *updateDistricts*() function.	
Preconditions	Existence of the locations in a predefined file. Select a province of the ones provided in the component.
Inputs	We select a canton of the ones provided by the component

Expected outputs	The districts state variable is set to the canton's corresponding districts and the districts are displayed on the component.

Test Case ID: 63 **Priority**: 1 **Test Coverage Item**: TCI6_TM4 **Purpose**: To verify that after calling the *sendForm*() function, the component renders the success message on its modal.	
Preconditions	Backend stub
Inputs	The component makes a call to the backend stub.
Expected outputs	The component renders the success message on its modal.

Test Case ID: 64 **Priority**: 1 **Test Coverage Item**: TCI7_TM4 **Purpose**: To verify that after calling the *sendForm*() function with an error, the component renders the error message on its modal.	
Preconditions	A backend stub.
Inputs	The component makes a call to the backend stub.
Expected outputs	The component renders the error message on its modal.

Test Case ID: 65 **Priority**: 1 **Test Coverage Item**: TCI8_TM4 **Purpose**: To verify that when the user makes a query to the backend that returns correctly, the function that handles the response sets the error state variable correctly.	
Preconditions	A backend stub.
Inputs	We make a click on the button to send a form, the component makes a request to the backend stub.
Expected outputs	The error state variable is set to false in the component, the message "false" is present in the component's modal.

Test Case ID: 66 **Priority**: 1 **Test Coverage Item**: TCI9_TM4 **Purpose**: To verify that when the user makes a query to the backend that returns an error, the function that handles the response sets the error state variable correctly.	
Preconditions	Backend stub
Inputs	We make a click on the button to send a form, the component makes a request to the backend stub.
Expected outputs	The error state variable is set to false in the component, the message “true” is present in the component’s modal.

Test Case ID: 67 **Priority**: 1 **Test Coverage Item**: TCI10_TM4 **Purpose**: To verify that when querying for a user’s cart data, it does so in a valid time (0 to 5 seconds)	
Preconditions	Backend stub
Inputs	The user’s ID for the query.
Expected outputs	The response is returned in 1 to 5 seconds.

Test Case ID: 68 **Priority**: 1 **Test Coverage Item**: TCI11_TM4 **Purpose**: To verify that when the *updateCantons*() function is called, it does the job in a valid time (0 to 5 seconds)	
Preconditions	File with the locations.
Inputs	We select a province of the ones provided by the component.
Expected outputs	The rendering of the cantons takes between 0 and 5 seconds.

Test Case ID: 69 **Priority**: 1 **Test Coverage Item**: TCI12_TM4 **Purpose**: To verify that when the *updateDistricts*() function is called, it does the job in a valid time (0 to 5 seconds)	
Preconditions	Existence of the locations in a predefined file. Select a province of the ones provided in the component.
Inputs	We select a canton of the ones provided by the component.
Expected outputs	The rendering of the districts takes between 0 and 5 seconds.

Test Case ID: 70 **Priority**: 1 **Test Coverage Item**: TCI13_TM4 **Purpose**: To verify that when the *sendForm*() function is called, it does the job in a valid time (0 to 5 seconds)	
Preconditions	A backend stub.
Inputs	The component makes a call to the backend stub.
Expected outputs	The rendering of the success message takes between 0 and 5 seconds.