Technological Institute of Costa Rica

Computer Engineering School

Test Model Specification

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 4, 2024.	1.0.	Initial draft released for review.
April 12, 2024.	2.0.	Includes corrections and additions specified by the Quality Assurance Team members after the review of the document.

Overview

Introduction

This document details the test model specification 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 model specification is to provide a representation of the aspects of the test items, focused on the attributes that define required test coverage.

Scope

This document covers the test model specification for the dynamic tests that are pertinent to the project. Based on the project test plan document, specifically referring to the test item of the purchase module, which consists of four source code files.

References

Internal

1. Project Test Plan V 1.0
2. Source Code of Project

Test Model Specification – Test Model 1

Test Model Unique Identifier

Dynamic Test Model #1

Objective

Ensure the correct functionality and time performance of the functions and methods provided by the *CartAdmin* class. This class provides services related to the management of the user´s shopping cart, including functions to add and delete products from the shopping cart, and send the order to the administrator.

Priority

1

Type of Testing Required (from Test Strategy)

According to the project test plan document, only unit testing will be performed on the test model. Both black box and white box tests must be performed, the techniques that will be used are equivalence partition, boundary value analysis, invalid values, decision tables, and structural tests. The tests are meant to ensure functionality and time performance. All unit tests will be automated using the Jest framework.

Test Model

The test model is defined as follows:

- The Quality Assurance Team is responsible for designing the test cases for the *CartAdmin* functions and methods. Such test cases will be focused on functionality and time performance.
- The test cases will be handed to the Development Team, who is responsible for programming the test cases using the Jest framework. Unit tests will be kept in a separate file in the repository of the project.
- The functions included in this class make use of the *CartDao* functions. The Development Team is responsible for producing a stub class that simulates the communication with the *CartDao* functions and make the proper changes so that the stub class is used under unit testing.
- After the unit tests have been programmed, the Development Team is responsible for executing such tests and report the results to the Quality Assurance Team. The tests will be executed in the environment defined in the test plan.
- The Quality Assurance Team is responsible for adding the results to the project testing report. The report will specify both the unit tests that were successful and those that failed.

Traceability

The test model maps to the CartAdmin.ts file of the source code, located under the “backend” directory, specifically in the “controllers” directory. The source code can be found in the online repository of the project.

Test Model Specification – Test Model 2

Test Model Unique Identifier

Dynamic Test Model #2

Objective

Ensure the correct functionality and time performance of the functions and methods provided by the *CartDao* class. This class provides services related to the management of the user´s shopping cart in the MongoDB database, including queries and procedures to register orders, add and delete products from the shopping cart, and get the data of the shopping cart.

Priority

1

Type of Testing Required (from Test Strategy)

According to the project test plan document, only unit testing will be performed on the test model. Both black box and white box tests must be performed, the techniques that will be used are equivalence partition, boundary value analysis, invalid values, decision tables, and structural tests. The tests are meant to ensure functionality and time performance. All unit tests will be automated using the Jest framework.

Test Model

The test model is defined as follows:

- The Quality Assurance Team is responsible for designing the test cases for the *CartDao* functions and methods. Such test cases will be focused on functionality and time performance.
- The test cases will be handed to the Development Team, who is responsible for programming the test cases using the Jest framework. Unit tests will be kept in a separate file in the repository of the project.
- After the unit tests have been programmed, the Development Team is responsible for executing such tests and report the results to the Quality Assurance Team. The tests will be executed in the environment defined in the test plan.
- The Quality Assurance Team is responsible for adding the results to the project testing report. The report will specify both the unit tests that were successful and those that failed.

Traceability

The test model maps to the CartDAO.ts file of the source code, located under the “backend” directory, specifically in the “daos” directory. The source code can be found in the online repository of the project.

Test Model Specification – Test Model 3

Test Model Unique Identifier

Dynamic Test Model #3

Objective

Ensure the correct functionality and time performance of the functions and methods provided by the *CartView* file. This file provides services related to the management of the user´s shopping cart in the frontend of the system. The file is responsible for functions such as showing the products from the shopping cart and allow to add or delete products from the shopping cart to the user.

Priority

2

Type of Testing Required (from Test Strategy)

According to the project test plan document, only unit testing will be performed on the test model. Both black box and white box tests must be performed, the techniques that will be used are equivalence partition, boundary value analysis, invalid values, decision tables, and structural tests. The tests are meant to ensure functionality and time performance. All unit tests will be automated using the Jest framework.

Test Model

The test model is defined as follows:

- The Quality Assurance Team is responsible for designing the test cases for the *CartView* functions and methods. Such test cases will be focused on functionality and time performance.
- The test cases will be handed to the Development Team, who is responsible for programming the test cases using the Jest framework. Unit tests will be kept in a separate file in the repository of the project.
- The functions included in this file make use of the backend functions, that is, they communicate with the server. The Development Team is responsible for producing a stub object that simulates the communication with the server functions and make the proper changes so that the stub object is used under unit testing.
- After the unit tests have been programmed, the Development Team is responsible for executing such tests and report the results to the Quality Assurance Team. The tests will be executed in the environment defined in the test plan.
- The Quality Assurance Team is responsible for adding the results to the project testing report. The report will specify both the unit tests that were successful and those that failed.

Traceability

The test model maps to the CartView.js file of the source code, located under the “frontend” directory, specifically in the “pages” directory. The source code can be found in the online repository of the project.

Test Model Specification – Test Model 4

Test Model Unique Identifier

Dynamic Test Model #4

Objective

Ensure the correct functionality and time performance of the functions and methods provided by the *CartPayment* file. This file provides services related to the purchase of the user´s shopping cart in the frontend of the system. The file is responsible for functions such as requesting the user for the direction of the order and the payment ticket, showing a summary of the shopping cart data (products and final price), and allowing to user to confirm the order and sent it to the administrator.

Priority

1

Type of Testing Required (from Test Strategy)

According to the project test plan document, only unit testing will be performed on the test model. Both black box and white box tests must be performed, the techniques that will be used are equivalence partition, boundary value analysis, invalid values, decision tables, and structural tests. The tests are meant to ensure functionality and time performance. All unit tests will be automated using the Jest framework.

Test Model

The test model is defined as follows:

- The Quality Assurance Team is responsible for designing the test cases for the *CartPayment* functions and methods. Such test cases will be focused on functionality and time performance.
- The test cases will be handed to the Development Team, who is responsible for programming the test cases using the Jest framework. Unit tests will be kept in a separate file in the repository of the project.
- The functions included in this file make use of the backend functions, that is, they communicate with the server. The Development Team is responsible for producing a stub object that simulates the communication with the server functions and make the proper changes so that the stub object is used under unit testing.
- After the unit tests have been programmed, the Development Team is responsible for executing such tests and report the results to the Quality Assurance Team. The tests will be executed in the environment defined in the test plan.
- The Quality Assurance Team is responsible for adding the results to the project testing report. The report will specify both the unit tests that were successful and those that failed.

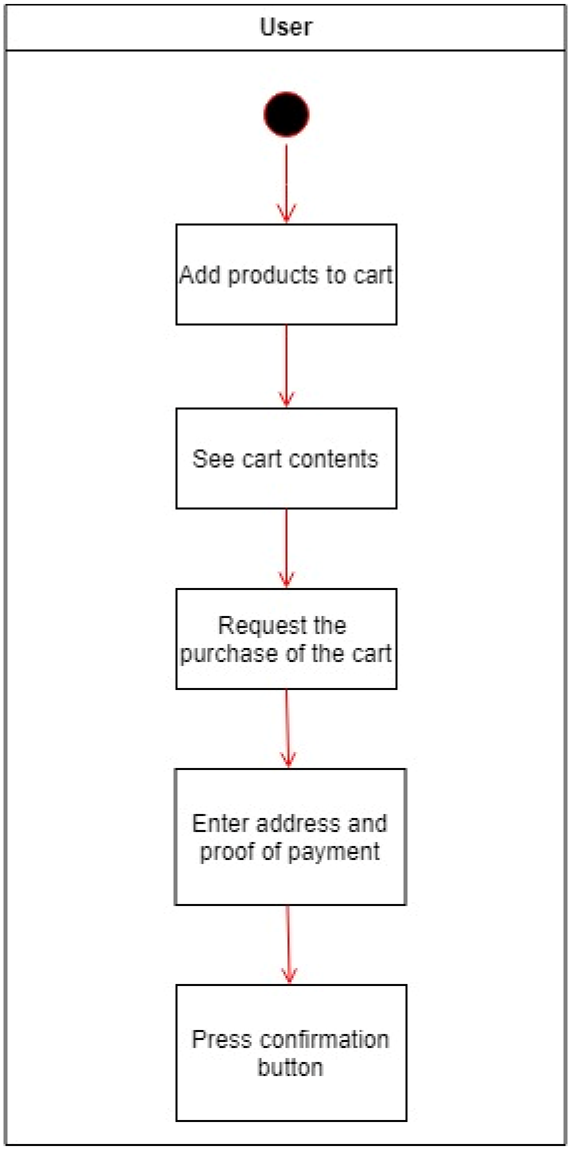
Traceability

The test model maps to the CartPayment.js file of the source code, located under the “frontend” directory, specifically in the “pages” directory. The source code can be found in the online repository of the project.

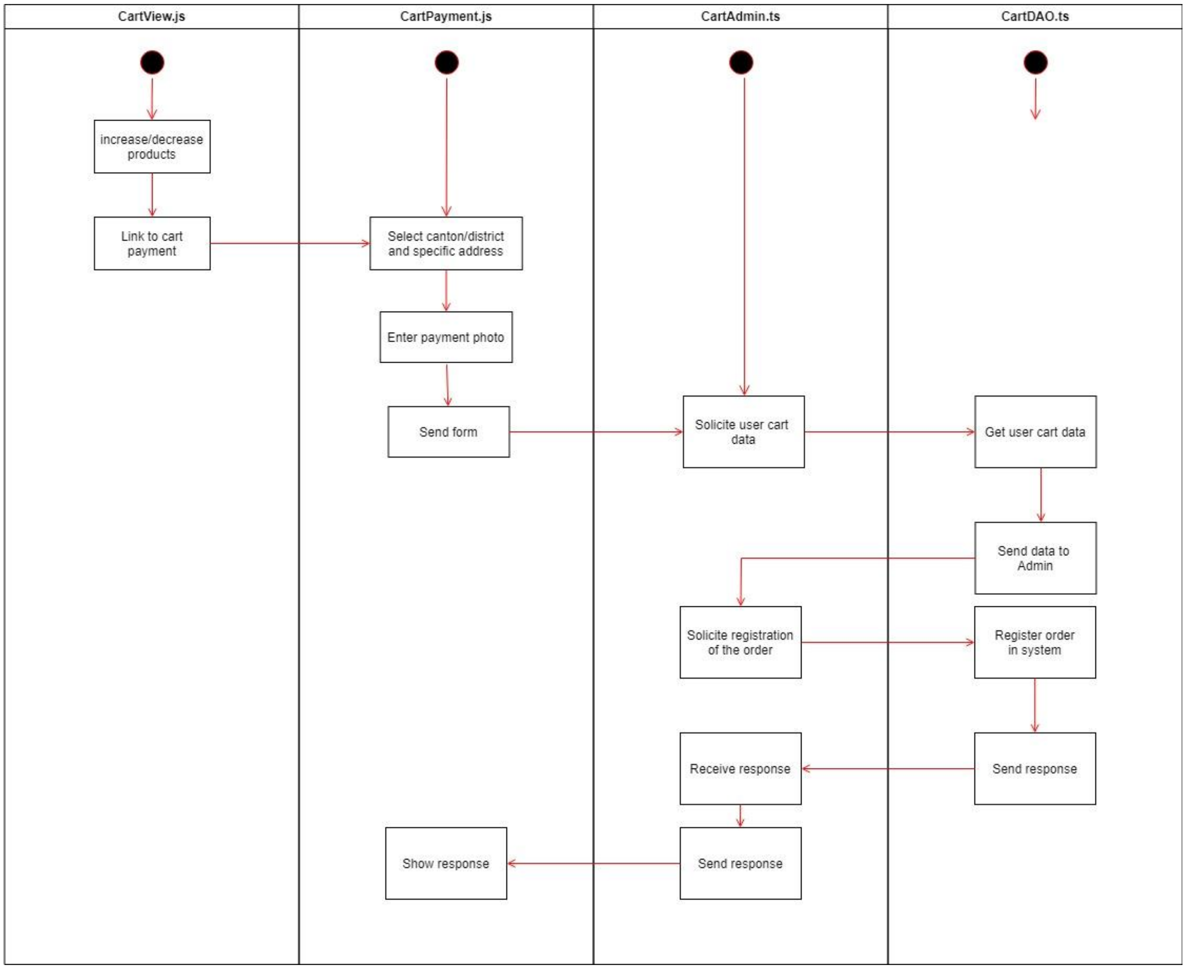
Diagrams for the Purchase Process

The following diagrams show the process of the purchase of a shopping cart in the system. One diagram specifies the process from the user´s perspective, and the other diagram specifies the process of how the system manages the process of purchasing a shopping cart.

Activity Diagram for the Purchase Process from the User´s Perspective


User
Add products to cart
See cart contents
Request the purchase of the cart
Enter address and proof of payment
Press confirmation button

Activity Diagram for the Purchase Process from the Module's Perspective


CartView.js
CartPayment.js
CartAdmin.ts
CartDAO.ts
increase/decrease products
Link to cart payment
Select canton/district and specific address
Enter payment photo
Send form
Solicite user cart data
Get user cart data
Send data to Admin
Solicite registration of the order
Register order in system
Send response
Receive response
Send response
Show response