Software Engineering

Group number: 19

Computer Hardware store

Submission date: 12 / 07 / 2021

Team members:

1. Abdulrahman Adel Ibrahim (Section 3)

Breakdown of individual contributions

1. Abdulrahman Adel

Code:

1. Created market page displaying all products that can be categorized and a shopping cart displaying number of items in your cart.
2. Created product page with its’ corresponding product class displaying all necessary information about product and ability to add to/remove from cart.
3. Cart page with its’ corresponding shopping cart class displaying all items added to the cart with basic information about these products (name, quantity, price).
4. order page with its’ corresponding order class that the user can add the shipping information and payment details and then can confirm the order that displays a message with all information about his/her order
5. Admin page with all necessary information about admin and all his/her functionalities (Add, remove or modify a product all with their respective pages)

Database:

1. Created a Database with all necessary tables for the project (customer, admin, order, product, product list) with all of the necessary information about them

Testing

1. Created unit test for the required function to be tested in the program

1. Project Description

As we are a computer online only store, we do not have a physical store for selling our computer product, so we need software application to display our product and keep the interactive processes between the customer and our store.

When a customer wants to buy any product of our products, he has to call us, or mail us to make sure that the product he need is available, checking its price, the available quantity, and the full description of that product.

these kinds of stuff frustrate us as we have to answer all these questions by checking our store for the availability and the too many excels sheets to check the product information.

As time goes, customers’ questions repeat itself and we have to repeat our answers and that takes a lot of time and unnecessary effort.

And unfortunately, the tragedy doesn’t end here, we also have to update our excel sheets to make it up to date for any upcoming processes after selling any product to delete it from the available products that available to customers to buy it immediately without ordering it from the trader.

Also, when we got new products, we have to add it to our excel sheets and providing all their related information like name, price, technical description.

The purpose of this application is to provide an interactive, easy-to-use management system to allow the client of this product to purchase hardware components and to allow the owner of the store to easily customize his products and to keep record for his clients for the owner to make better decisions.

2. Goals and Constraints

The client for this project wishes to enter the PC-based internet environment. The Store Management System will be PC-base with a internet, allowing Store Clients to search for hardware components and Store staff members to manage the Store inventory and client database. The application will provide the following capabilities:

* The application will be access via an internet on a PC at any place.
* User can view details of the parts without going anywhere.
* It is convenient for users as this system provides accurate cost and description of the system.
* User can view different categories of product of different brands at a single place.
* The products are delivered to the customer in his/her house so this becomes more convenient for them.
* The system calculates bill instantly and user can pay online

Hence the system saves time, efforts and cost.

3. System Requirements

A – Enumerated Functional Requirements

|  |  |
| --- | --- |
| REQ-X | Requirements |
| REQ-1 | Be able to show all products without going anywhere |
| REQ-2 | Being Provided with accurate product description and cost |
| REQ-3 | Product being delivered to his/her house hence a system saves time, cost and effort |
| REQ-4 | A system that calculates bill instantly and allowing the user to pay online. |
| REQ-5 | The system shall allow new users to register an account on the website. |
| REQ-6 | The system should provide a specific Login ID to every Registered user and Admin. |
| REQ-7 | The system should be able to differentiate between the users and Admins while login. |
| REQ-8 | The system shall identify the user based on his login ID and Password. |
| REQ-9 | The system should allow the user to write a feedback. |
| REQ-10 | The admins could add products to the system |
| REQ-11 | The admins will be able to remove/modify products |
| REQ-12 | The admins will be able to see customer’s activity history |
| REQ-13 | The managers will be able to see admin’s activity history |

B- Enumerated Non-functional Requirements

|  |  |
| --- | --- |
| REQ-X | Requirements |
| REQ-14 | User is allowed to change the password and notified when he attempts to change his password. |
| REQ-15 | User is logged, off if he is idle for most of his login duration. |
| REQ-16 | The number of transactions per hour does the system need to be able to handle |
| REQ-17 | The amount of data does the system need to be able to store |

4. functional requirements specification

4.1. Stakeholders

* **Customers:**

1. Customer (visitor): anyone visit our store application
2. Registered customer: the ones who have an account on our store application

* **Admins**:

1. Managers
2. Employees

## 4.2. Actors and Goals:

|  |  |  |  |
| --- | --- | --- | --- |
| Actor | role | type | Goals in case of initiators |
| employee | admin | initiator | Add/remove/modify products/  View activity history |
| Registered customer | customer | initiator | Browse /Buy product(s) |
| visitor | visitor | initiator | Browse product(s) |
| Database | Server Responds to requests | participator | --------------------------- |

## 4.3. use case:

## 4.3.1. use case description:

* Sign up: allow users to create account on the application system to be able to perform the most core operations like purchasing products in case of customers or some operations on products like add/remove/modify and view activity in case of admin.

**That’s satisfying the following requirement(s):**

1. The system shall allow new users to register an account on the website
2. The system should provide a specific Login ID to every Registered user and Admin.

* View products: allow users (including the unregistered ones) to browse our store accessing all products’ information like name, price, and description.

**That’s satisfying the following requirement(s):**

1. Be able to show all products without going anywhere
2. Being Provided with accurate product description and cost.
3. A system that is convenient, flexible and easy to use.

* Log in: allow users (employees/Registered customers) to log by their already created account into the application’s system.

**That’s satisfying the following requirement(s):**

1. The system should provide a specific Login ID to every Registered user and Admin.
2. The system should be able to differentiate between the users and Admins while login.
3. The system shall identify the user based on his login ID and Password.

* Add to cart: allow registered customer to add the selected product to his cart.

**That’s satisfying the following requirement(s**):

1. Registered customers can add products to their cart.

* Remove from cart: allow registered customer to remove the selected product from his cart.

**That’s satisfying the following requirement(s):**

1. Registered customers can remove products from their cart.

* View cart: allow registered customer to view his cart to view the products which is/are in it.

**That’s satisfying the following requirement(s):**

1.registered customers can view their shopping cart.

* Make order: allow registered customer to purchase product(s)

**That’s satisfying the following requirement(s):**

1.allow registered customer to buy product.

2. Be able to show all products without going anywhere

3. A system that is convenient, flexible and easy to use

4. A system that calculates bill instantly and allowing the user to pay online

5. Product being delivered to his/her house hence a system saves time, cost and effort

* Add product: allow admins to add product by providing its name, price and description if there’s a description.

**That’s satisfying the following requirement(s):**

1. allow admins to add/remove/modify products.
2. A system that is convenient, flexible and easy to use

* Remove products: allow admins to remove product by providing some unique information of it like id.

**That’s satisfying the following requirement(s):**

1. allow admins to add/remove/modify products.
2. A system that is convenient, flexible and easy to use

* Edit product: allow admins to edit any information of the selected products by providing unique information of it like id, then enter the new information name or/and, price, or/and description.

**That’s satisfying the following requirement(s):**

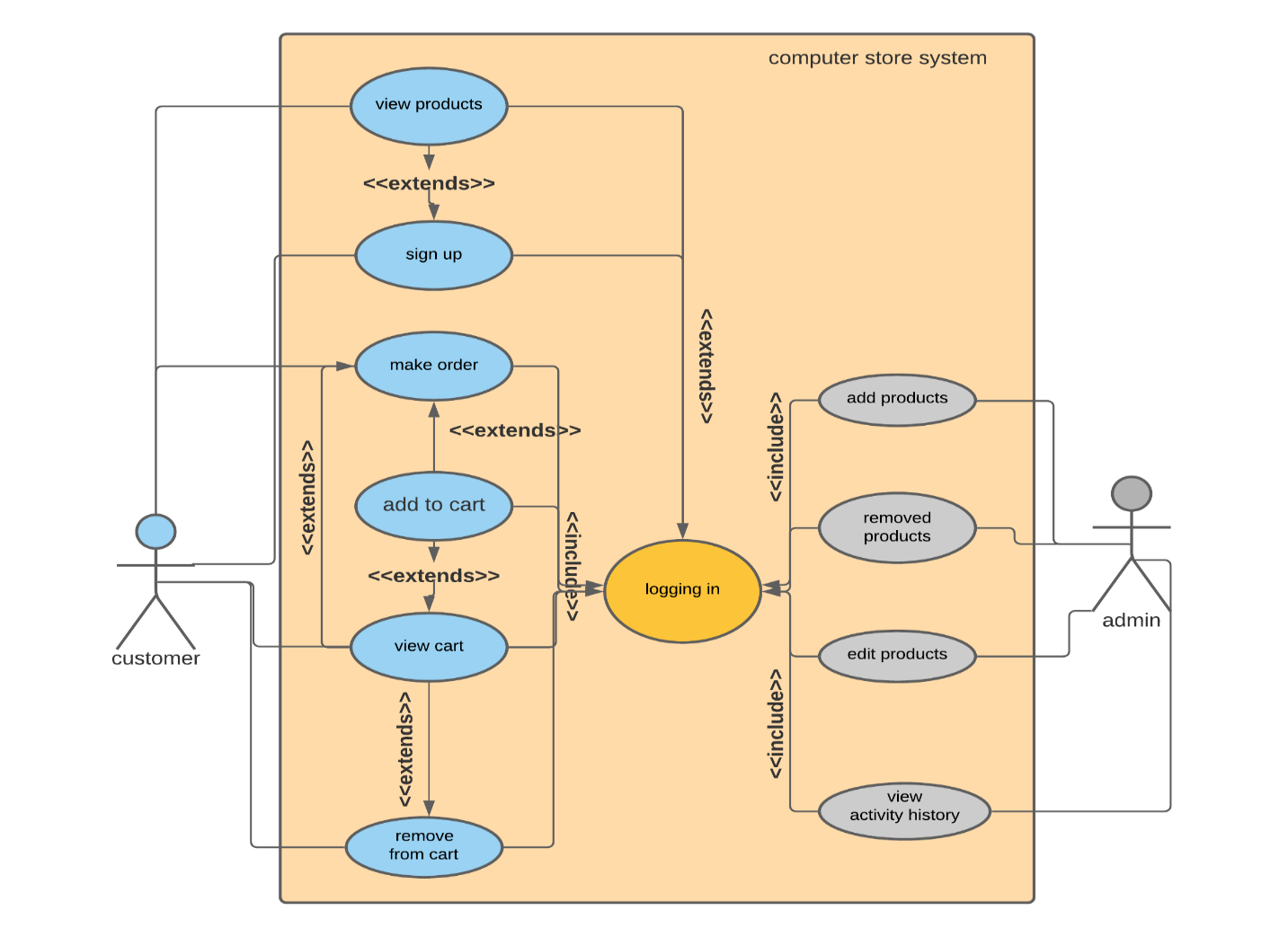
1. allow admins to add/remove modify products.
2. A system that is convenient, flexible and easy to use

* View activity: allow admins to view the activity of their account (activities can’t be deleted). this allows the managers by logging in with any admin’s account to see the activity history of this account.

**That’s satisfying the following requirement(s):**

1.allow managers to keep tracking their employees.

## 4.3.2.use case diagram:



4.4. Black-box test cases

1. Login

|  |
| --- |
| **Main Success Scenario** |
| C: Customer  S: System | |  |  | | --- | --- | | **Steps** | **Description** | | 1 | C: Enter Customer’s ID and Password | | 2 | S: Validate Entered Information | | 3 | S: Allow Account Access | |
| Extensions | |  |  | | --- | --- | | **2a** | **Either Password or Customer’s ID is not valid** | |  | S: System will display a warning message | | **3a** | **Customer ID is not in database** | |  | S: System will ask the visitor to register | |

2. Register

|  |
| --- |
| **Main Success Scenario** |
| C: Customer  S: System | |  |  | | --- | --- | | **Steps** | **Description** | | 1 | C: Enter all the required information | | 2 | S: Validate Entered Information | | 3 | S: Creates an entry in the database | | 4 | S: Move to the login page | |
| Extensions | |  |  | | --- | --- | | **2a** | **Any of these entered information is invalid** | |  | S: System will display a warning message specifying the invalid entry | | **3a** | **There is a duplicate entry in the database** | |  | S: System will inform the customer that his/her (ID/email) is already registered in the system | |  | S: Move to login page | |

3. Add to cart

|  |
| --- |
| **Main Success Scenario** |
| C: Customer  S: System | |  |  | | --- | --- | | **Steps** | **Description** | | 1 | C: Login into the system | | 2 | C: From market Enter on a product page | | 3 | C: Specify the quantity of the desired product | | 4 | C: Press Add to cart button | | 5 | S: Checks available quantity of the product | | 6 | S: Adds the product with the desired quantity to shopping cart | |
| Extensions | |  |  | | --- | --- | | **2a** | **Product is already in shopping cart** | |  | S: System won’t show add to cart button instead will show Update quantity button | | **3a** | **Product quantity is unavailable** | |  | S: System will display a warning message | |

4. Add product

|  |
| --- |
| **Main Success Scenario** |
| A: Admin  S: System | |  |  | | --- | --- | | **Steps** | **Description** | | 1 | A: Login into the system | | 2 | A: Press add product button | | 3 | A: Enter all required information | | 4 | S: Validate Entered Information | | 5 | S: Creates an entry in the database | |
| Extensions | |  |  | | --- | --- | | **2a** | **Any of these entered information is invalid** | |  | S: System will display a warning message | | **3a** | **There is a duplicate entry in the database** | |  | S: System will notify the admin of the already added product | |  | S: Move to admin page | |
|  |  |

5. Modify product

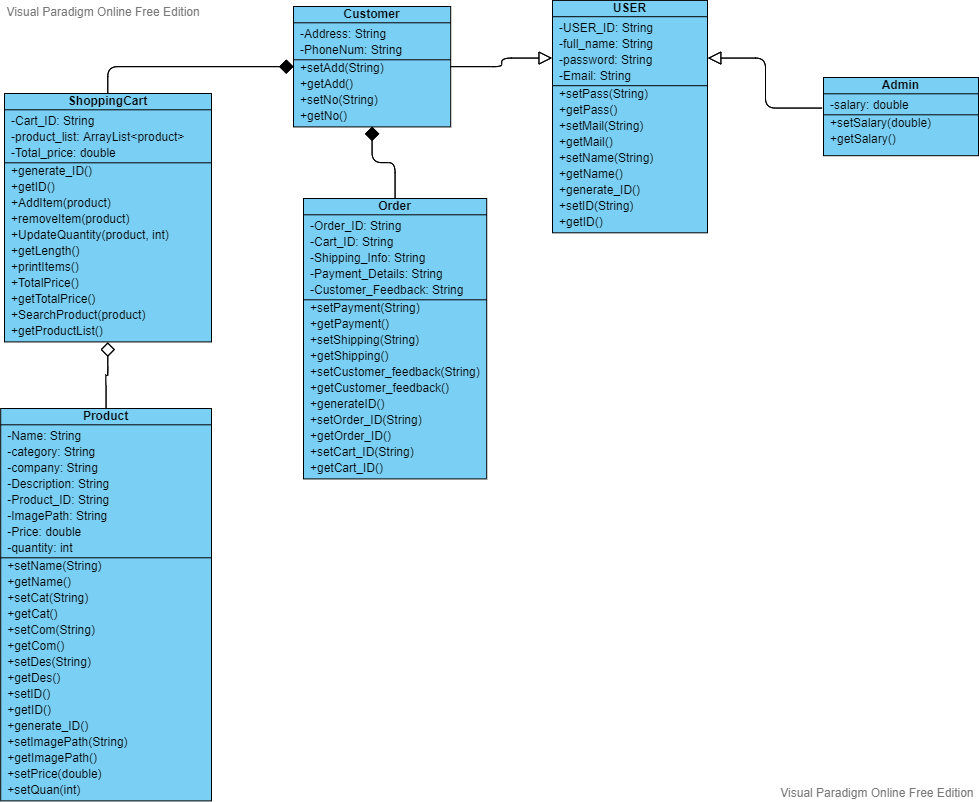
|  |
| --- |
| **Main Success Scenario** |
| A: Admin  S: System | |  |  | | --- | --- | | **Steps** | **Description** | | 1 | A: Login into the system | | 2 | A: Press modify product button | | 3 | A: Enter product ID | | 4 | S: Display all information (Editable) of that product | | 5 | A: Modify all required information | | 6 | S: Validate the entered information | | 7 | S: Update the entry of the product in the database | |
| Extensions | |  |  | | --- | --- | | **2a** | **Any of these entered information is invalid** | |  | S: System will display a warning message | | **3a** | **Entered ID is invalid** | |  | S: System will display a warning message | |

4. Remove product

|  |
| --- |
| **Main Success Scenario** |
| A: Admin  S: System | |  |  | | --- | --- | | **Steps** | **Description** | | 1 | A: Login into the system | | 2 | A: Press remove product button | | 3 | S: System will display an input message to input the required product ID | | 4 | A: Enter product ID | | 5 | S: Validate the ID | | 6 | S: Delete this product entry from the database | |
| Extensions | |  |  | | --- | --- | | **2a** | **Entered ID is invalid** | |  | S: System will display a warning message | |

4. Architectural Design

5.Detailed Design



**1- Administrator**

A child class that inherits from “User” class that is for admins and managers.

**Operations:**

* **Add new product()**

The admin can add a new product by adding the following information

* ID
* Name
* Price
* Description
* **Modify a product()**

Only administrators can modify product’s data, all products can be modified.

* **Remove a Product()**

Only administrators can remove products, all products can be removed

* **ViewActivity()**

Only administrators can view user’s activity history, all the previous purchases.

**Attributes:**

* AdminName: String 🡪 The name of each admin
* email: String 🡪 The email if each admin
* UserID: String 🡪 A unique number for each admin
* password: String 🡪 password to protect the account
* Salary: double 🡪salary of the employee determined by managers.

**2- Customer**

A child class that inherits from “User” class that is for potential customers.

**Operations:**

* **Register**

New customers must sign up in order to complete a purchase by filling out their personal information and creating an account.

* **Login**

Admins and customers can log in to the system by their Name/ID or email and password.

**Attributes:**

* CustomerName: String 🡪 The name of the customer (filled while registering)
* email: String 🡪 the email of the customer (filled while registering)
* UserID: String 🡪 A unique number for each customer
* password: String 🡪 a password to protect the account (filled while registering)
* Address: String 🡪 the current place of residence (filled while registering)
* PhoneNum: String 🡪 the phone number of the customer

**3- User**

A class for existing customers

**Operations:**

* Update Personal Information
* Verfiy if the customer is logged in

**Attributes:**

* UserID: String
* password: String
* Full\_name: String
* Email: String

**4- Shopping Cart**

A class for placing all desired products that the customer wants to purchase.

**Operations:**

* +AddItem() a customer/user can add items to the cart
* +removeItem() a customer/user can remove items from the cart
* +UpdateQuantity() a customer/user can select the quantity of each product
* +ViewDetails() a customer/user can view the full details of the cart
* +CheckOut() only a user can checkout the finish shopping and a customer would be asked to register or login if he’s already a user

**Attributes:**

* CartID: int 🡪 A unique number for each cart
* ProductID: int 🡪 a unique number for each product
* Quantity: int 🡪 the quantity of each product
* DateAdded: String 🡪 the date that the cart is added at.

**5- Order**

A class that specifies placing an order

**Operations:**

* ConfirmOrder() a user can confirm the order after completing his/her shipping information, calculating the total price and payment details.
* GiveFeedback() a user can give feedback and a review of the process.

**Attributes:**

* OrderID: int 🡪 A unique number of the order
* CartID: int 🡪 A unique number for shipping information.
* UserID: int 🡪 the ID of the user that is ordering
* OrderDetails: String 🡪 the complete details of the order (total price – shipping info – shipping cost – user info)

**6- Product**

A class for products

**Attributes:**

* Name: String 🡪 name of the product
* productID: String 🡪 A unique number for each product.
* company: String 🡪 the manufacturing company of the product
* category: String
* Description: String
* Price: double
* Quantity
* ImagePath: String 🡪 the image of the product path in this computer.

4. Testing Report

1- Register information test case

Function to be tested 🡪 Check\_info();

* It checks all entered information about the customer when registering
* It returns a Boolean (true if all entries are correct, false if any in incorrect)

Information to be checked:

* Full Name 🡪 has to be alphanumeric string
* Phone number 🡪 11 digits
* Password 🡪 Password must contain Minimum eight characters, at least one uppercase letter, one lowercase letter, one number and one special character.
* Address
* Email 🡪 a valid email format.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input | Expected output | Result | Test case status |
| TC1 | All information are placed correctly | True | True | Passed |
| TC2 | One of the entries is missing | False,  Warning message | False,  Warning message | Passed |
| TC3 | The two passwords don’t equal each other | False,  Warning message | False,  Warning message | Passed |
| TC4 | Invalid Email entry | False,  Warning message | False,  Warning message | Passed |

2- Cart Test

Functions to be tested 🡪 Addproduct(); , removeProduct(); , searchproduct();

* Shopping cart class stores the purchased product in an ArrayList.
* These functions perform operations on that list.
* AddProduct(); 🡪 adds product to the list and displays a message “product added”
* removeProduct(); 🡪 removes a product from the list (if already exists)
* searchProduct(); 🡪 boolean function to search for a product in the list.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input | Expected output | Result | Test case status |
| TC1 | Create a new product,  Add it to the cart and search for that product in the list | True,  Message “product added” | True,  Message ”product added” | Passed |
| TC2 | Create two products, add one and search for the other | False | False | Passed |
| TC3 | Add two products, remove one of them and search for it | False | False | Passed |

3- Add product test case

Function to be tested 🡪 check\_info();

* It checks all entered information about the product when added by an administrator
* It returns a Boolean (true if all entries are correct, false if any in incorrect)

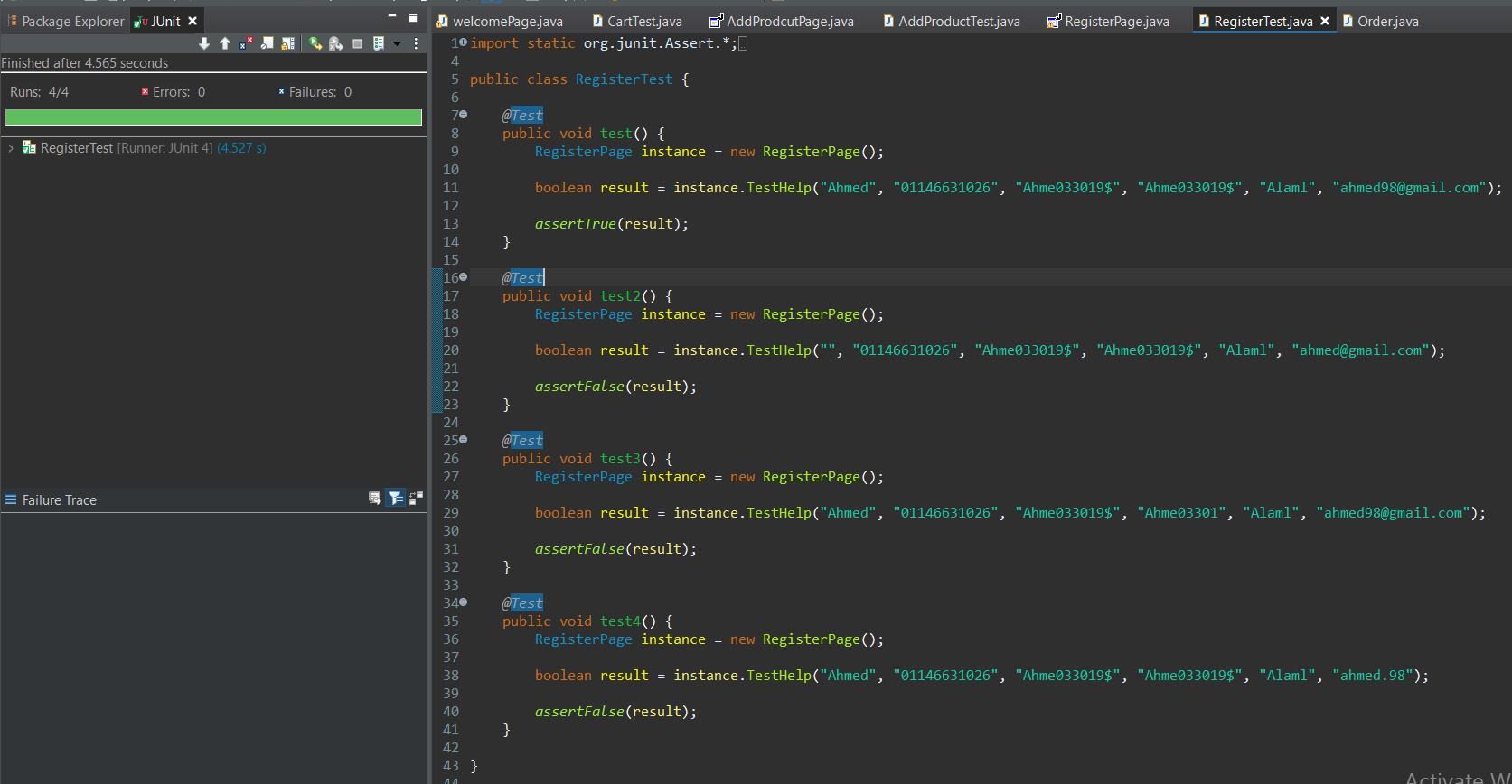
Information to be checked:

* Name 🡪 has to be a string
* category 🡪 has to be a string
* company
* Image path 🡪 has to be a valid image path format (compared to a regex)
* Price 🡪 float number.
* Description 🡪 String

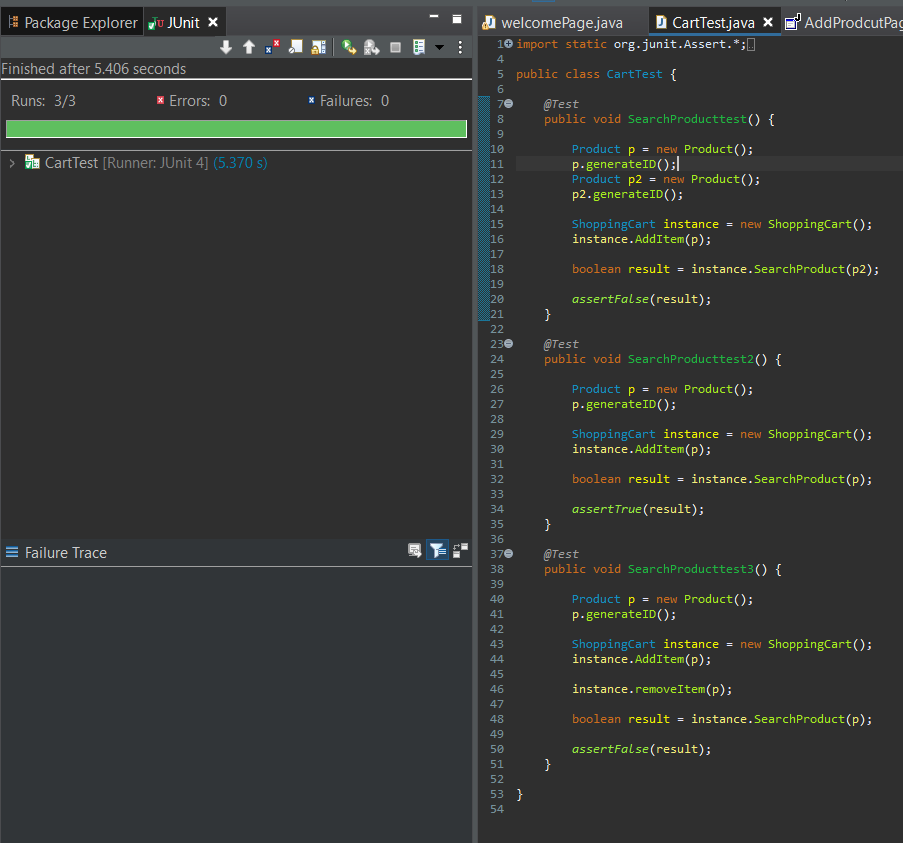
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input | Expected output | Result | Test case status |
| TC1 | All information is placed correctly | True | True | Passed |
| TC2 | Invalid price  (not a float) | False,  Warning message | False,  Warning message | Passed |
| TC3 | All entries are missing | False,  Warning message | False,  Warning message | Passed |

2- Unit testing

1- Register information



2- Cart test



3- Add product test

صورة تحتوي على نص

تم إنشاء الوصف تلقائياً