**Cashier System**

**Test Case Approach and solution**

**Elaborated by: Octavio Restrepo**

|  |  |  |
| --- | --- | --- |
| **TC Id** | **TC001** |  |
|  |  |  |
| Description | Given a user with 0 items in the Cart  When the user verifies the empty Cart  Then the Cart should be empty  And the Price should be £0.00 |  |
| Test env | local |  |
| Test Data | priv/assets/products.yml  priv/assets/rules.yml |  |
| Preconditions | The products should exist in the storage/  The select products should not candidate for any discount.  / The payment component should be implemented |  |
| Browser / Version | Default |  |
| Test Steps | **Step** | **Expected outcome** |
|  | There should not be items added to the cart | Total Price: £0.00 |
|  |  |  |
| **TC Id** | **TC002** |  |
| Description | Scenario: Verify total price calculation for a single product without discounts  Given a product with a price of £3.11  When the user adds the product to the Cart  Then the Cart should display a total price of £3.11 |  |
| Test env | local |  |
| Test Data | priv/assets/products.yml |  |
| Preconditions | The products should exist in the storage/  The select products should not candidate for any discount.  / The payment component should be implemented |  |
| Browser / Version | Default |  |
| Test Steps | **Step** | **Expected outcome** |
|  | Add an existing product to the cart | The product is added to the cart after systems verifies the stock |
|  | Press the pay button | A notification with the value of the product is displayed |
|  | User selects the payment method | The system loads the payment information |
|  | User completes the payment | System registers the item payment |
|  |  |  |
|  |  |  |
| **TC Id** | **TC003** |  |
| Description | Scenario: Verify "Buy one get one free" discount rule  Given a user with two identical products in the Cart  When the discount rule "Buy one get one free" is applied  Then the Cart should display the discounted price for one product  And the Cart should display the quantity of the discounted product as 1 |  |
| Test env | local |  |
| Test Data | priv/assets/products.yml  priv/assets/rules.yml |  |
| Preconditions | The products should exist in the storage/  The select products should not candidate for any discount.  / The payment component should be implemented |  |
| Browser / Version | Default |  |
| Test Steps | **Step** | **Expected outcome** |
|  | Add GR1 (Green Tea), GR1 (Green Tea) to the cart | The product is added to the cart after systems verifies the stock |
|  | Press the pay button | Total Price: £3.11 (only one Green Tea is charged, the other is free) |
|  | User selects the payment method | The system loads the payment information |
|  | User completes the payment | System registers the item payment |
|  |  |  |
|  |  |  |
| **TC Id** | **TC004** |  |
| Description | Scenario: Verify "Buy > N products, pay X price per product" discount rule  Given a user with a Cart containing N+1 products  And the discount rule "Buy > N products, pay X price per product" is defined as buying more than N products at a price of X per product  When the user adds N+1 products to the Cart  Then the Cart should display the total price as N\*X for all N+1 products |  |
| Test env | local |  |
| Test Data | priv/assets/products.yml  priv/assets/rules.yml |  |
| Preconditions | The products should exist in the storage/  The select products should not candidate for any discount.  / The payment component should be implemented |  |
| Browser / Version | Default |  |
| Test Steps | **Step** | **Expected outcome** |
|  | Add SR1 (Strawberries), SR1 (Strawberries), SR1 (Strawberries) to the cart | The products are added to the cart after systems verifies the stock |
|  | Press the pay button | Total Price: £13.00 (Strawberries priced at £5.00 each and with discount it turns to 4.3333 each) |
|  | User selects the payment method | The system loads the payment information |
|  | User completes the payment | System registers the items payment with discount |
|  |  |  |
|  |  |  |
| **TC Id** | **TC005** |  |
| Description | Scenario: Verify "Buy > N products, pay X% of the original price" discount rule  Given a user with a Cart containing N+1 products  And the discount rule "Buy > N products, pay X% of the original price" is defined as buying more than N products at X% of the original price  When the user adds N+1 products to the Cart  Then the Cart should display the total price applying the corresponding discount  N = 5  X% = 20% |  |
| Test env | local |  |
| Test Data | priv/assets/products.yml  priv/assets/rules.yml |  |
| Preconditions | The products should exist in the storage/  The select products should not candidate for any discount.  / The payment component should be implemented |  |
| Browser / Version | Default |  |
| Test Steps | **Step** | **Expected outcome** |
|  | Add CF1 (Coffee), CF1 (Coffee), CF1 (Coffee), CF1 (Coffee), CF1 (Coffee) to the cart | The products are added to the cart after systems verifies the stock |
|  | Press the pay button | Total Price: £44.92 (20% of the original price, £8.984, for each Coffee) |
|  | User selects the payment method | The system loads the payment information |
|  | User completes the payment | System registers the items payment with discount |
|  |  |  |
|  |  |  |
| **TC Id** | **TC006** |  |
| Description | Scenario: Verify combined discount rules  Given a user with a Cart containing 9 products  And the discount rule "Buy > 5 products, pay 20% of the original price" is applied  And the discount rule "Buy one get one free" is applied  When the user adds 9 products to the Cart  Then the Cart applies the discounts  And the Cart should display the total price as the original price for the remaining 1 product  N = 5  X% = 20% |  |
| Test env | local |  |
| Test Data | priv/assets/products.yml  priv/assets/rules.yml |  |
| Preconditions | The products should exist in the storage/  The select products should not candidate for any discount.  / The payment component should be implemented |  |
| Browser / Version | Default |  |
| Test Steps | **Step** | **Expected outcome** |
|  | Add GR1 (Green Tea), GR1 (Green Tea), GR1 (Green Tea), GR1 (Green Tea), GR1 (Green Tea), SR1 (Strawberries), SR1 (Strawberries), SR1 (Strawberries), CF1 (Coffee) to the cart | The products are added to the cart after systems verifies the stock |
|  | Press the pay button | Total Price: £28.67 (discounts applied for Green Tea pay 5 and got 20% discount and Strawberries pay 1 get one free, normal price for Coffee) |
|  | User selects the payment method | The system loads the payment information |
|  | User completes the payment | System registers the items payment with discount |
|  |  |  |
|  |  |  |
| **TC Id** | **TC007** |  |
| Description | Scenario: Verify handling of invalid product codes  Given a user with a Cart  When the user tries to add a product with an invalid code to the Cart  Then the Cart should display an error message indicating the invalid product code |  |
| Test env | local |  |
| Test Data | priv/assets/products.yml  priv/assets/rules.yml |  |
| Preconditions | The products should exist in the storage/  The select products should not candidate for any discount.  / The payment component should be implemented |  |
| Browser / Version | Default |  |
| Test Steps | **Step** | **Expected outcome** |
|  | XX1 (Invalid Product) | A notification indicating that one of the product codes is not valid. |
|  | Press the pay button | The systems is unable to process payment |
|  |  |  |
|  |  |  |
| **TC Id** | **TC008** |  |
| Description | Scenario: Verify combining handling of missing product in YAML file and ehandle a valid product  Given a user with a Cart  When the user tries to add a product with an invalid code to the Cart  And The user adds a product with a valid code  Then the Cart should display an error message indicating the invalid product code  And The cart should process the valid product |  |
| Test env | local |  |
| Test Data | priv/assets/products.yml  priv/assets/rules.yml |  |
| Preconditions | The products should exist in the storage/  The select products should not candidate for any discount.  / The payment component should be implemented |  |
| Browser / Version | Default |  |
| Test Steps | **Step** | **Expected outcome** |
|  | GR1 (Green Tea), PR1 (Product Not Defined) | The products are added to the cart after systems verifies the stock.  A notification indicating that one of the product codes is not valid. |
|  | Press the pay button | Total Price: £3.11 (ignores missing product, calculates price for available products) |
|  | User selects the payment method | The system loads the payment information |
|  | User completes the payment | System registers the items payment with discount |
|  |  |  |
|  |  |  |
| **TC Id** | **TC009** |  |
| Description | Scenario: Verify handling of missing or invalid discount rules in YAML file  Given a user with a Cart  When user adds a product with a valid code  And there is no YAML rules file added  Then The cart should process the valid product without discounts |  |
| Test env | local |  |
| Test Data | priv/assets/products.yml |  |
| Preconditions | The products should exist in the storage/  The select products should not candidate for any discount.  / The payment component should be implemented |  |
| Browser / Version | Default |  |
| Test Steps | **Step** | **Expected outcome** |
|  | APP1 (Apple) | The products are added to the cart after systems verifies the stock. |
|  | Press the pay button | Total Price: £3.00 (no discounts applied, calculates normal price) |
|  | User selects the payment method | The system loads the payment information |
|  | User completes the payment | System registers the items payment with discount |
|  |  |  |

**Shift-Left Approach**

**Requirements Analysis:**

* Verify the provided requirements for the Cashier System:
  + product list
  + discount rules
  + YAML file configuration.

**Test Case Design:**

* Design test cases to cover different scenarios, including pricing calculations, discount rules, product combinations, and error handling.
* Ensure the test cases provide good coverage of the system's functionality and address both positive and negative scenarios.

**Unit Testing:**

* Developers should perform unit tests for individual components and functions of the Cashier System.
* Verify that the unit tests provide sufficient code coverage.

**Test Environment Setup:**

* Set up a test environment with the necessary infrastructure, including the Cashier System, the YAML files for products and rules, and any required dependencies.
* Ensure the test environment accurately reflects the production environment.

**Automated Testing:**

* Develop automated tests to cover the designed test cases.
* Implement test scripts or frameworks using suitable automation tools (Selenium, JUnit, Cypress) to execute the automated tests.

**Integration Testing:**

* Perform integration testing to ensure the Cashier System correctly interacts with other components or systems (if applicable).

**Regression Testing:**

* When new features or changes are introduced to the Cashier System, execute regression tests to ensure that existing functionalities continue working as expected.
* Verify that the modifications do not impact any previous scenarios.

**Continuous Integration/Continuous Delivery (CI/CD) Pipeline Integration:**

* Integrate the automated tests into the CI/CD pipeline to execute the tests automatically with each code commit or deployment.

**Documentation:**

* Maintain updated documentation to ensure clear communication and knowledge sharing within the team.

**Continuous Improvement:**

* Regularly evaluate the testing process, identify areas for improvement, and implement necessary changes.
* Leverage metrics and feedback to optimize the testing strategy, increase efficiency, and enhance the quality of the Cashier System.

**METRICS NEEDED TO ASURE A COMPLETE TEST COVERAGE**

1.Number of functions covered by tests / Total number of functions

2. Number of conditions covered by tests / Total number of conditions