**Test Function Document**

|  |  |
| --- | --- |
| Project Name: Assignment 1 Testing | |
| Automation Title: test\_checkout | Version: 1.0 |
| Testing Phase: Phase 1 | Date of Test: 16 November 2023 |
| Module Name: checkout\_and\_payment.py | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Function Title: test\_checkout | | | | Test Designed by: Edvin Eriksson | | | |
| Test Priority (Low/Medium/High): Medium | | | | Test Designed Date: 16 November 2023 | | | |
| Description: The objective of the checkout function is for when a user wants to checkout the products that are in the cart, resulting in the total amount of the products in the cart to be deducted from the user’s wallet. The units of the affected products should also be changed accordingly. | | | | Test Executed by: Edvin Eriksson | | | |
| Test Execution date: 16 November 2023 | | | |
|  | | | | | | | |
| Pre-conditions: None | | | | | | | |
| Dependencies: User, Product, ShoppingCart | | | | | | | |
|  | | | | | | | |
| S. No | Equivalence Class | Test case data | Expected Results | | Actual Results | Status (Pass/Fail) | Notes |
| 1 | VEC1 | product\_list = []  User(name='Kim', wallet='20'),  new\_cart | captured.out.strip() == "Your basket is empty. Please add items before checking out."  user.wallet == 20  len(new\_cart.retrieve\_item()) == 0  len(product\_list) == 0 | | captured.out.strip() == "Your basket is empty. Please add items before checking out."  user.wallet == 20  len(new\_cart.retrieve\_item()) == 0  len(product\_list) == 0 | Pass | Test name: test\_EC1()  Tests with an empty cart.  Uses a fixture new\_cart that is a ShoppingCart(). |
| 2 | VEC2 | product\_list = [Product(name='Orange', price=10, units=3)]  User(name='Kim', wallet=5)  new\_cart.add\_item(product\_list[0]) | captured.out.strip() == f"You don't have enough money to complete the purchase.\nPlease try again!"  user.wallet == 5 len(new\_cart.retrieve\_item()) == 1  len(product\_list) == 1 | | captured.out.strip() == f"You don't have enough money to complete the purchase.\nPlease try again!"  user.wallet == 5 len(new\_cart.retrieve\_item()) == 1  len(product\_list) == 1 | Pass | Test name: test\_EC2()  Tests with insufficient wallet funds.  Uses a fixture new\_cart that is a ShoppingCart(). |
| 3 | VEC3 | product\_list = [Product(name='Orange', price=10, units=3)]  User(name='Kim', wallet=100)  new\_cart.add\_item(product\_list[0]) | expected\_output = f"Thank you for your purchase, {user.name}! Your remaining balance is {user.wallet}"  captured.out.strip() == expected\_output  user.wallet == 90  len(new\_cart.retrieve\_item()) == 0  product\_list[0].units == 2 | | expected\_output = f"Thank you for your purchase, {user.name}! Your remaining balance is {user.wallet}"  captured.out.strip() == expected\_output  user.wallet == 90  len(new\_cart.retrieve\_item()) == 0  product\_list[0].units == 2 | Pass | Test name: test\_EC3  Tests with sufficient wallet funds and a product in cart  Uses a fixture new\_cart that is a ShoppingCart(). |
| 4 | VEC4 | product\_list = [Product(name='Orange', price=10, units=3), Product(name='Apple', price=20, units=2)]  User(name='Kim', wallet=100)  new\_cart.add\_item(product\_list[0])  new\_cart.add\_item(product\_list[1]) | expected\_output = f"Thank you for your purchase, {user.name}! Your remaining balance is {user.wallet}"  captured.out.strip() == expected\_output  user.wallet == 70  len(new\_cart.retrieve\_item()) == 0  product\_list[0].units == 2  product\_list[1].units == 1 | | expected\_output = f"Thank you for your purchase, {user.name}! Your remaining balance is {user.wallet}"  captured.out.strip() == expected\_output  user.wallet == 70  len(new\_cart.retrieve\_item()) == 0  product\_list[0].units == 2  product\_list[1].units == 1 | Pass | Test name: test\_EC4()  Tests with sufficient wallet funds and multiple products in cart  Uses a fixture new\_cart that is a ShoppingCart(). |
| 5 | VEC5 | product\_list = [Product(name='Orange', price=10, units=1)]  User(name='Kim', wallet=100)  new\_cart.add\_item(product\_list[0])  new\_cart.add\_item(product\_list[0]) | user.wallet == 80  len(new\_cart.retrieve\_item()) == 0  len(product\_list) == 0 | | user.wallet == 80  len(new\_cart.retrieve\_item()) == 0  len(product\_list) == 0 | Pass | Test name: test\_EC5()  Tests with not enough available units than requested by the user  Uses a fixture new\_cart that is a ShoppingCart(). |
| 6 | VEC6 | product\_list = [Product(name='Orange', price=10, units=3)]  User(name='Kim', wallet=-100)  new\_cart.add\_item(product\_list[0]) | expected\_output = f"You don't have enough money to complete the purchase.\nPlease try again!" captured.out.strip() == expected\_output  user.wallet == -100 len(new\_cart.retrieve\_item()) == 1  product\_list[0].units == 3 | | expected\_output = f"You don't have enough money to complete the purchase.\nPlease try again!" captured.out.strip() == expected\_output  user.wallet == -100 len(new\_cart.retrieve\_item()) == 1  product\_list[0].units == 3 | Pass | Test name: test\_EC6()  Tests with negative wallet funds  Uses a fixture new\_cart that is a ShoppingCart(). |
| 7 | VEC7 | product\_list = [Product(name='Orange', price=-10, units=3)]  User(name='Kim', wallet=100)  new\_cart.add\_item(product\_list[0]) | expected\_output = f"Thank you for your purchase, {user.name}! Your remaining balance is {user.wallet}"  captured.out.strip() == expected\_output  user.wallet == 110  len(new\_cart.retrieve\_item()) == 0  product\_list[0].units == 2 | | expected\_output = f"Thank you for your purchase, {user.name}! Your remaining balance is {user.wallet}"  captured.out.strip() == expected\_output  user.wallet == 110  len(new\_cart.retrieve\_item()) == 0  product\_list[0].units == 2 | Pass | Test name: test\_EC7()  Tests with products in cart with negative price  Uses a fixture new\_cart that is a ShoppingCart(). |
| 8 | VEC8 | product\_list = [Product(name='Orange', price=10, units=1)]  User(name='Kim', wallet=100)  new\_cart.add\_item(product\_list[0]) | user.wallet == 90  len(product\_list) == 0 | | user.wallet == 90  len(product\_list) == 0 | Pass | Test name: test\_EC8()  Tests with sufficient funds and a product in cart but with only one unit left  Uses a fixture new\_cart that is a ShoppingCart(). |
| 9 | VEC9 | product\_list = [Product(name='Orange', price=10, units=3)]  User(name='Kim', wallet=100.5)  new\_cart.add\_item(product\_list[0]) | user.wallet == 90.5 len(new\_cart.retrieve\_item()) == 0  product\_list[0].units == 2 | | user.wallet == 90.5 len(new\_cart.retrieve\_item()) == 0  product\_list[0].units == 2 | Pass | Test name: test\_EC9()  Tests with decimal balance in the wallet funds  Uses a fixture new\_cart that is a ShoppingCart(). |
| 10 | VEC10 | product\_list = [Product(name='Orange', price=10, units=0)]  User(name='Kim', wallet=100)  new\_cart.add\_item(product\_list[0]) | user.wallet == 90  len(new\_cart.retrieve\_item()) == 0  product\_list[0].units == -1 | | user.wallet == 90  len(new\_cart.retrieve\_item()) == 0  product\_list[0].units == -1 | Pass | Test name: test\_EC10()  Tests with product with no units  Uses a fixture new\_cart that is a ShoppingCart(). |