**Test Function Document**

|  |  |
| --- | --- |
| Project Name: Online Shopping Cart | |
| Automation Title: test\_logout | Version: 1.0 |
| Testing Phase: Phase 1 | Date of Test: 24 Nov 2023 |
| Module Name: Online Shopping Cart | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Function Title: logout | | | | Test Designed by: Lucas Åhl | | | |
| Test Priority (Low/Medium/High):  High | | | | Test Designed Date: 24 Nov 2023 | | | |
| Description: Function to logout a user | | | | Test Executed by: Lucas Åhl | | | |
| Test Execution date: 24 Nov 2023 | | | |
|  | | | | | | | |
| Pre-conditions: csv | | | | | | | |
| Dependencies: cart.items, cart.retrieve\_items, cart.clear\_items | | | | | | | |
|  | | | | | | | |
| S. No | Equivalence Class | Test case data | Expected Results | | Actual Results | Status (Pass/Fail) | Notes |
| 1 | File not found | "no\_csv.csv" | Raises File Not Found Error | | Raises File Not Found Error | Pass |  |
| 2 | Empty csv | (empty\_csv\_file) | captured.out == '[]\n' | | captured.out == '[]\n' | Pass |  |
| 3 | Header only csv | (header\_only\_csv\_file) | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out | | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out | Pass |  |
| 4 | No header csv | (csv\_file\_without\_header) | assert 'Product' not in captured.out  assert 'Price' not in captured.out  assert 'Units' not in captured.out  assert 'Apple' in captured.out  assert 'Banana' in captured.out  assert 'Orange' in captured.out | | assert 'Product' not in captured.out  assert 'Price' not in captured.out  assert 'Units' not in captured.out  assert 'Apple' in captured.out  assert 'Banana' in captured.out  assert 'Orange' in captured.out | Pass |  |
| 5 | Different delimiter csv | (different\_delimiter\_csv\_file) | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out  assert 'Apple' in captured.out  assert '1.0' in captured.out  assert '10' in captured.out | | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out  assert 'Apple' in captured.out  assert '1.0' in captured.out  assert '10' in captured.out | Pass |  |
| 6 | Missing values csv | (csv\_file\_with\_missing\_values) | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out  assert 'Apple' in captured.out  assert 'Banana' in captured.out  assert 'Orange' in captured.out  assert '[]' not in captured.out | | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out  assert 'Apple' in captured.out  assert 'Banana' in captured.out  assert 'Orange' in captured.out  assert '[]' not in captured.out | Pass |  |
| 7 | Special characters csv | (special\_characters\_csv\_file) | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out  assert 'Apple' in captured.out  assert '$5.00' in captured.out  assert '10' in captured.out  assert 'Banana' in captured.out  assert '€2.50' in captured.out  assert '20' in captured.out  assert 'Orange' in captured.out  assert '¥3.00' in captured.out  assert '15' in captured.out | | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out  assert 'Apple' in captured.out  assert '$5.00' in captured.out  assert '10' in captured.out  assert 'Banana' in captured.out  assert '€2.50' in captured.out  assert '20' in captured.out  assert 'Orange' in captured.out  assert '¥3.00' in captured.out  assert '15' in captured.out | Pass |  |
| 8 | Irregular spacing csv | (irregular\_spacing\_csv\_file) | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out  assert 'Banana' in captured.out  assert '0.5' in captured.out | | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out  assert 'Banana' in captured.out  assert '0.5' in captured.out | Pass |  |
| 9 | Quoted values csv | (quoted\_values\_csv\_file) | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out  assert '"Apple"' in captured.out  assert '"2.5"' in captured.out  assert '"10"' in captured.out | | assert 'Product' in captured.out  assert 'Price' in captured.out  assert 'Units' in captured.out  assert '"Apple"' in captured.out  assert '"2.5"' in captured.out  assert '"10"' in captured.out | Pass |  |
| 10 | Full csv | ("products.csv") | ("['Product', 'Price', 'Units']\n"  "['Apple', '2', '10']\n"  "['Banana', '1', '15']\n"  "['Orange', '1.5', '8']\n"  "['Grapes', '3', '5']\n"  "['Strawberry', '4', '12']\n"  "['Watermelon', '10', '1']\n"  "['Carrot', '0.5', '20']\n"  "['Broccoli', '1.5', '10']\n"  "['Tomato', '1', '15']\n"  "['Cucumber', '1', '12']\n"  "['Potato', '0.75', '18']\n"  "['Onion', '0.8', '20']\n"  "['Bell Pepper', '1.2', '8']\n"  "['Lettuce', '2', '5']\n"  "['Spinach', '2.5', '7']\n"  "['Milk', '3', '10']\n"  "['Eggs', '2', '24']\n"  "['Cheese', '5', '8']\n"  "['Chicken Breast', '7', '4']\n"  "['Salmon', '10', '2']\n"  "['Ground Beef', '6', '5']\n"  "['Pasta', '1', '15']\n"  "['Rice', '1.5', '10']\n"  "['Bread', '2', '8']\n"  "['Butter', '3', '6']\n"  "['Yogurt', '2', '12']\n"  "['Ice Cream', '4', '6']\n"  "['Chocolate', '2.5', '8']\n"  "['Coffee', '5', '4']\n"  "['Tea', '2', '10']\n"  "['Soda', '1.5', '12']\n"  "['Water', '1', '20']\n"  "['Juice', '3', '8']\n"  "['Chips', '2.5', '10']\n"  "['Cookies', '3', '8']\n"  "['Cereal', '2', '12']\n"  "['Oatmeal', '1.5', '15']\n"  "['Peanut Butter', '3', '6']\n"  "['Jelly', '2', '8']\n"  "['Toothpaste', '1.5', '10']\n"  "['Shampoo', '2', '8']\n"  "['Soap', '1', '12']\n"  "['Toilet Paper', '0.75', '24']\n"  "['Towel', '4', '6']\n"  "['Laundry Detergent', '3.5', '8']\n"  "['Dish Soap', '1.5', '12']\n"  "['Broom', '5', '4']\n"  "['Trash Bags', '2', '10']\n"  "['Light Bulbs', '1', '15']\n"  "['Batteries', '3', '6']\n"  "['Phone Charger', '5', '4']\n"  "['Laptop', '800', '1']\n"  "['Headphones', '50', '1']\n"  "['Bluetooth Speaker', '30', '1']\n"  "['TV', '500', '1']\n"  "['Microwave', '80', '1']\n"  "['Coffee Maker', '40', '1']\n"  "['Toaster', '20', '1']\n"  "['Blender', '30', '1']\n"  "['Vacuum Cleaner', '100', '1']\n"  "['Dumbbells', '20', '2']\n"  "['Yoga Mat', '15', '1']\n"  "['Running Shoes', '60', '1']\n"  "['Backpack', '25', '1']\n"  "['Sunglasses', '10', '1']\n"  "['Hat', '8', '1']\n"  "['Gloves', '5', '1']\n"  "['Umbrella', '7', '1']\n"  "['Notebook', '2', '5']\n"  "['Pens', '0.5', '10']\n"  "['Backpack', '15', '1']\n") = captured.out | | ("['Product', 'Price', 'Units']\n"  "['Apple', '2', '10']\n"  "['Banana', '1', '15']\n"  "['Orange', '1.5', '8']\n"  "['Grapes', '3', '5']\n"  "['Strawberry', '4', '12']\n"  "['Watermelon', '10', '1']\n"  "['Carrot', '0.5', '20']\n"  "['Broccoli', '1.5', '10']\n"  "['Tomato', '1', '15']\n"  "['Cucumber', '1', '12']\n"  "['Potato', '0.75', '18']\n"  "['Onion', '0.8', '20']\n"  "['Bell Pepper', '1.2', '8']\n"  "['Lettuce', '2', '5']\n"  "['Spinach', '2.5', '7']\n"  "['Milk', '3', '10']\n"  "['Eggs', '2', '24']\n"  "['Cheese', '5', '8']\n"  "['Chicken Breast', '7', '4']\n"  "['Salmon', '10', '2']\n"  "['Ground Beef', '6', '5']\n"  "['Pasta', '1', '15']\n"  "['Rice', '1.5', '10']\n"  "['Bread', '2', '8']\n"  "['Butter', '3', '6']\n"  "['Yogurt', '2', '12']\n"  "['Ice Cream', '4', '6']\n"  "['Chocolate', '2.5', '8']\n"  "['Coffee', '5', '4']\n"  "['Tea', '2', '10']\n"  "['Soda', '1.5', '12']\n"  "['Water', '1', '20']\n"  "['Juice', '3', '8']\n"  "['Chips', '2.5', '10']\n"  "['Cookies', '3', '8']\n"  "['Cereal', '2', '12']\n"  "['Oatmeal', '1.5', '15']\n"  "['Peanut Butter', '3', '6']\n"  "['Jelly', '2', '8']\n"  "['Toothpaste', '1.5', '10']\n"  "['Shampoo', '2', '8']\n"  "['Soap', '1', '12']\n"  "['Toilet Paper', '0.75', '24']\n"  "['Towel', '4', '6']\n"  "['Laundry Detergent', '3.5', '8']\n"  "['Dish Soap', '1.5', '12']\n"  "['Broom', '5', '4']\n"  "['Trash Bags', '2', '10']\n"  "['Light Bulbs', '1', '15']\n"  "['Batteries', '3', '6']\n"  "['Phone Charger', '5', '4']\n"  "['Laptop', '800', '1']\n"  "['Headphones', '50', '1']\n"  "['Bluetooth Speaker', '30', '1']\n"  "['TV', '500', '1']\n"  "['Microwave', '80', '1']\n"  "['Coffee Maker', '40', '1']\n"  "['Toaster', '20', '1']\n"  "['Blender', '30', '1']\n"  "['Vacuum Cleaner', '100', '1']\n"  "['Dumbbells', '20', '2']\n"  "['Yoga Mat', '15', '1']\n"  "['Running Shoes', '60', '1']\n"  "['Backpack', '25', '1']\n"  "['Sunglasses', '10', '1']\n"  "['Hat', '8', '1']\n"  "['Gloves', '5', '1']\n"  "['Umbrella', '7', '1']\n"  "['Notebook', '2', '5']\n"  "['Pens', '0.5', '10']\n"  "['Backpack', '15', '1']\n") = captured.out | Pass |  |
| 11 | Invalid int | (1) | Raises Stop Iteration Error | | Raises Stop Iteration Error | Pass |  |
| 12 | Invalid float | (1.1) | Raises Type Error | | Raises Type Error | Pass |  |
| 13 | Invalid list | ([]) | Raises Type Error | | Raises Type Error | Pass |  |