

**Department of Master of Computer Applications**

**Software Testing (MCAE53)**

**Laboratory Exercises**

**Submitted by:**

**PANKAJ TERON**

**1MS23MC068**

**RAMAIAH INSTITUTE OF TECHNOLOGY**

(Autonomous Institute, Affiliated to VTU)

Accredited by National Board of Accreditation & NAAC with ‘A+’ Grade

MSR Nagar, MSRIT Post, Bangalore-560054

[www.msrit.edu](http://www.msrit.edu/)

**2025**

**1. Problem Description for the Application Date: 06/11/2024**

**Problem Statement**

Online shopping platforms often face issues like poor user experience, inefficient search and filtering, and complex checkout processes leading to cart abandonment. Many systems lack seamless integration with secure payment gateways and struggle with scalability. This project aims to resolve these challenges by offering a **user-friendly, scalable, and secure** e-commerce solution.

**Need for the Project**

* **Improved User Experience:** Traditional platforms have complex navigation and inefficient product discovery mechanisms.
* **Optimized Checkout Process:** Many users abandon their carts due to long and complicated checkout procedures.
* **Enhanced Security:** Secure handling of transactions and user data is essential.
* **Scalability:** The system should be able to handle large traffic volumes and product catalogs.
* **Mobile Responsiveness:** Ensuring a seamless shopping experience on both mobile and desktop devices.

**2. Requirements Specification Date: 07/11/2024**

**System Requirements**

**1. Software Requirements**

* **Operating System**: Windows, Linux, or macOS (based on deployment)
* **Database Management System**: MySQL or Oracle Database
* **Programming Languages**: Java, PHP,
* **Web Technologies**: HTML, CSS, JavaScript for front-end development
* **Frameworks**: Spring Framework (for Java), Laravel (for PHP)
* **Software Tools**: IDEs like Eclipse, Visual Studio; Database Management tools like MySQL Workbench
* **Web Server**: Apache Tomcat, Wamp, or Xamp.
* **Cloud Services**: (Optional) AWS, Google Cloud, or Azure for cloud hosting and scalability.

**2. Hardware Requirements**

* **Server**:
  + Processor: Minimum Intel i5 or AMD Ryzen 5
  + RAM: 8 GB or more
  + Storage: Minimum 500 GB SSD (based on the number of rooms and data volume)
  + Network: High-speed internet connection for data transfer and cloud access
* **Client Computers**: Desktop or laptop computers with basic specifications (e.g., Intel i3 or equivalent, 4 GB RAM) for the staff to access the system.

**3. User Requirements**

* **Customers:** Easy product search, secure checkout.
* **Admins:** Order and inventory management.
* **Developers:** Scalable and maintainable architecture.

**4. Functional Requirements**

* **User Authentication:** Registration, Login, Forgot Password.
* **Product Search & Filters:** Category-wise filtering and keyword-based search.
* **Shopping Cart Management:** Add/remove items, apply coupons, update quantities.
* **Checkout and Payment Gateway Integration:** Multiple secure payment options.
* **Order Tracking:** Live order status updates and estimated delivery time.

**5. Non-Functional Requirements**

* **Performance:** Page response time under 2 seconds.
* **Security:** Encrypted transactions using SSL for secure data transmission.
* **Scalability:** Supports increasing user traffic and large product catalogs.
* **Reliability:** Available 24/7 with minimal downtime.

**3. Preparation of Test Scenarios Date: 13/11/2024**

**Test Scenario 1: User Authentication**

* **Can users register successfully with valid credentials?**
* Yes, users can successfully register with valid credentials.
* **What happens when users enter incorrect login details?**
* No, users entering incorrect login details will see an error notification.
* **Does "Forgot Password" functionality work?**
* Yes, the "Forgot Password" functionality works, and users receive a password reset link.

**Test Scenario 2: Product Search & Filtering**

* **Can users find products using the search bar?**
* Yes, users can find products using the search bar.
* **Are category filters working correctly?**
* Yes, category filters are working correctly.

**Test Scenario 3: Shopping Cart & Checkout**

1. **Can users add/remove items from the cart?**

* Yes, users can add and remove items from the cart.

1. **Does the system update the cart total dynamically?**

* Yes, the system updates the cart total dynamically.

1. **Can users apply discount coupons?**

* Yes, users can apply discount coupons.

1. **Can users complete the checkout process successfully?**

* Yes, users can complete the checkout process successfully.

**Test Scenario 4: Order Management & Tracking**

* **Can users view past and ongoing orders?**
* Yes, users can view past and ongoing orders.
* **Are order tracking details updated in real-time?**
* Yes, order tracking details are updated in real-time.

**Test Scenario 5: Mobile Responsiveness**

* **Does the website function correctly on mobile devices?**
* Yes, the website functions correctly on mobile devices.
* **Does the layout adjust properly on smaller screens?**
* Yes, the layout adjusts properly on smaller screens.

**4. Preparation of Test Cases Date: 14/11/2024**

**4.1 User Sign-Up Module**

| **Test Case ID** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC1 | Enter valid username, email, and password | Username: John, Email: [johndoe@gmail.com](mailto:johndoe@gmail.com),  Password: Test@123 | Redirect to login page | Redirected successfully | Pass |
| TC2 | Enter invalid email format | Username: John, Email: john#gmail.com, Password: John@123 | Error message: "Invalid email format" | Displayed correctly | Pass |
| TC3 | Leave password field empty | Username: John, Email: [john@gmail.com](mailto:john@gmail.com), Password: | Alert message: "Please fill out this field" | Displayed correctly | Pass |

**4. Preparation of Test Cases Date: 14/11/2024**

**4.2 User Login Module**

| **Test Case ID** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC4 | Enter valid email and password | Email: [johndeo@gmail.com](mailto:johndeo@gmail.com), Password: Test@123 | Redirect to homepage | Redirected successfully | Pass |
| TC5 | Enter incorrect password | Email: [john@gmail.com](mailto:john@gmail.com), Password: wrongPass | Error message: "Invalid login credentials" | Displayed correctly | Pass |
| TC6 | Leave both fields empty | Email: , Password: | Alert: "Please fill out all fields" | Displayed correctly | Pass |

**4.3 Shopping Cart Module**

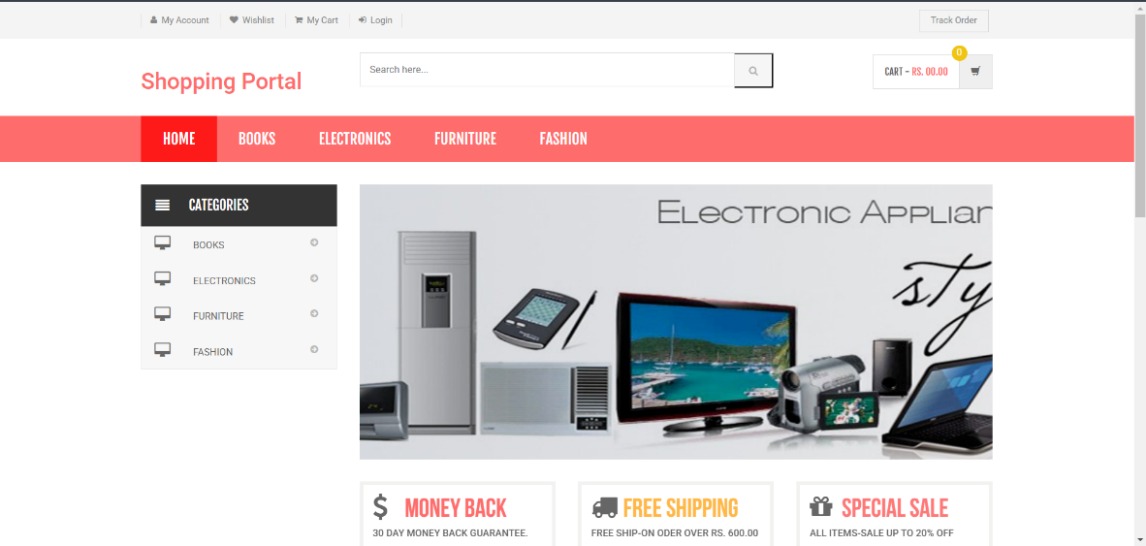
| **Test Case ID** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC7 | Add product to cart | Product: Laptop | Cart updated successfully | Updated correctly | Pass |
| TC8 | Remove product from cart | Product: Laptop | Product removed successfully | Removed correctly | Pass |
| TC9 | Checkout without adding product | No product in cart | Error: "Cart is empty" | Displayed correctly | Pass |

**4.4 Order Placement Module**

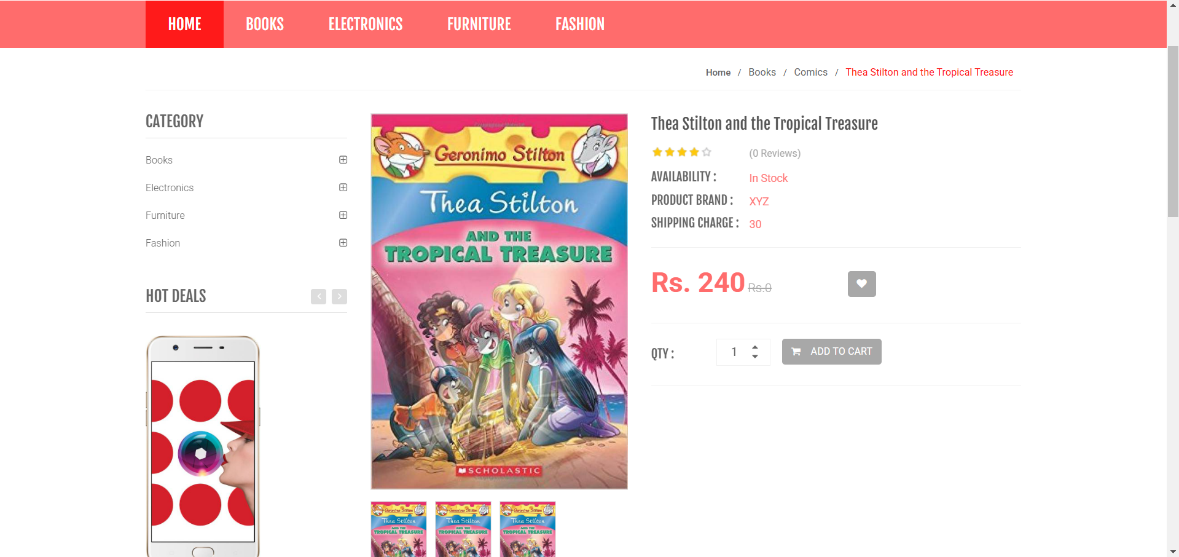
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** |
| TC10 | Select COD as payment method | Click on COD and proceed | Product added to order list successfully | Product added | Pass |
| TC11 | Select Card Payment | Click on Card and proceed | Product added to order list successfully | Product added | Pass |
| TC12 | Select UPI Payment | Click on UPI and proceed | Product added to order list successfully | Product added | Pass |

**6. Development of the Application Date: 27/11/2024**

**Home Page**

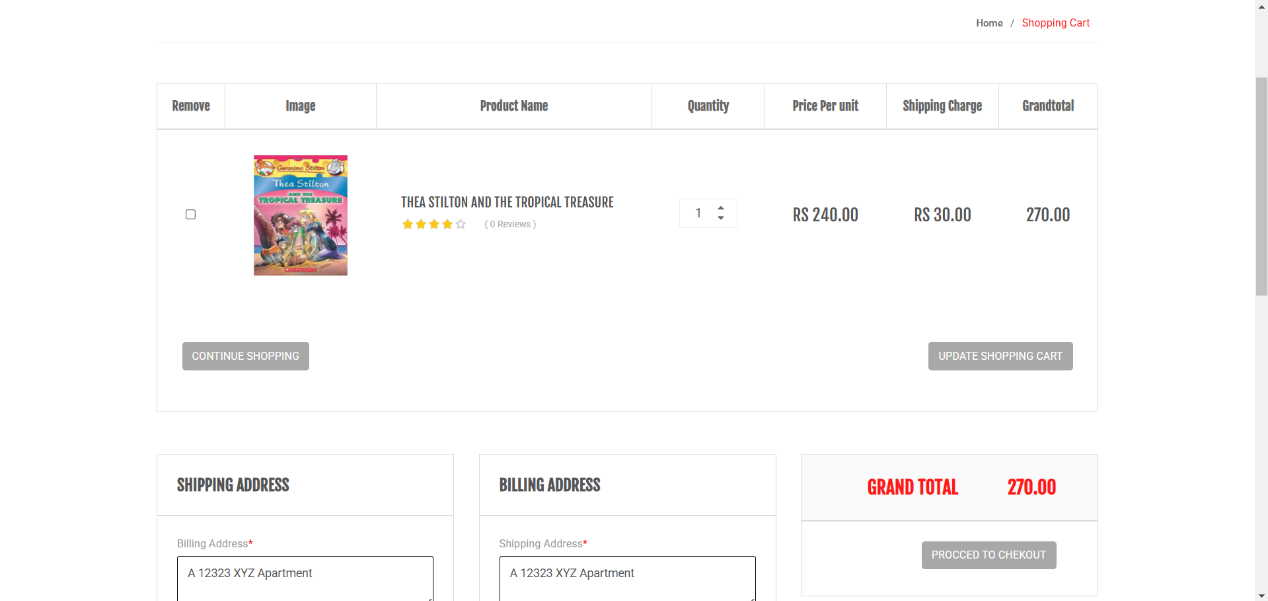
****

**Product**

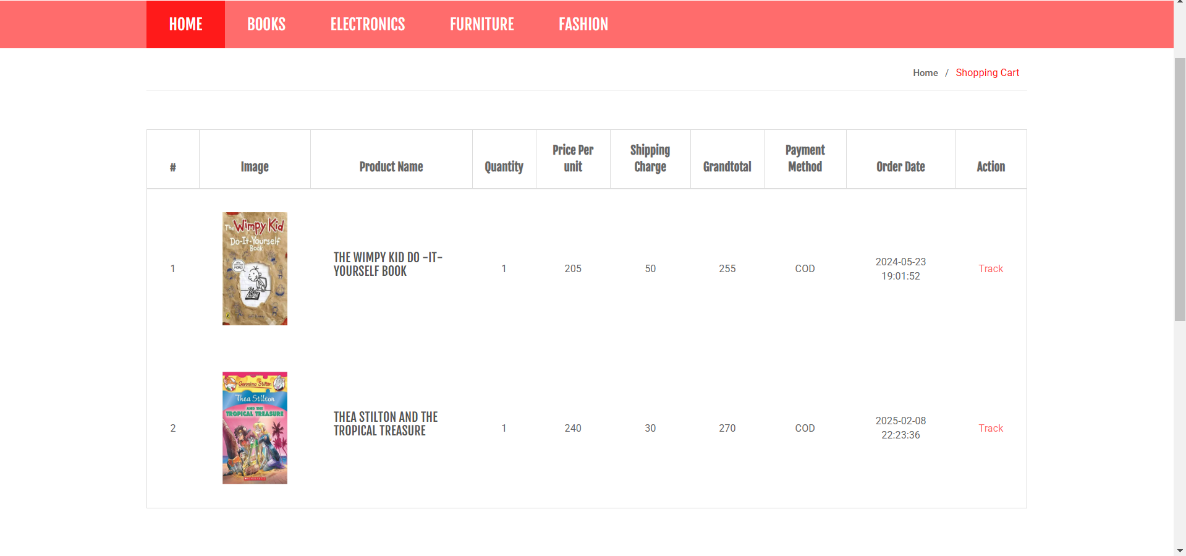
****

**7. Development of the Application Date: 28/11/2024**

**Add to Cart**

****

**Order**

****

**8. Problem description and requirements for the given webpage Date: 04/12/2024**

**Problem Description:**

Active individuals often struggle to find high-quality, stylish, and affordable gym apparel in one convenient location. Existing online retailers may have limited selections, confusing navigation, or lack detailed product information. This project aims to develop an e-commerce platform specializing in gym clothes, offering a wide variety of brands, styles, and sizes, along with a user-friendly shopping experience.

**Requirements:**

* 1. **Hardware Requirements**
  + **Server Requirements (for hosting the website):**
* **Processor**: Multi-core processor (e.g., Intel Xeon, AMD Ryzen).
* **RAM**: Minimum 8 GB (16 GB or more for better performance).
* **Storage**: SSD with at least 100 GB of free space (expandable for large-scale applications).
* **Network**: High-speed internet with low latency.
* **Client-Side (for testing): o Processor: Dual-core or higher.**
* **RAM**: 4 GB or higher.
* **Storage**: 20 GB free space.
* **Screen** **Resolution**: Minimum 1366x768.
  1. **Software Requirements**
  + **For Hosting:**
* **Web** **Server**: Apache, Nginx, or IIS.
* **Database**: MySQL, PostgreSQL, or MongoDB.
* **Backend** **Language** **Support**: Python.
* **Operating** **System**: Windows Server.
* **For Testing:**
* **Browser** **Support**: Latest versions of Chrome, Firefox, Safari, and Edge.
* **OS** **Compatibility**: Windows, macOS, and Linux.
* **Device** **Testing**: Android and iOS devices for mobile responsiveness.
  1. **Tools for Automation Testing**
* **Web Testing Tool:**
* **Selenium WebDriver** for browser automation.
  1. **Manual Testing Requirements**
* **Test Plan Documentation:** A document outlining test cases, scenarios, and expected outcomes.
  1. **Dependencies**
* **Frameworks and Libraries:**
* **React.js, Angular, or Vue.js for frontend testing.**
* **Bootstrap or Material UI for UI testing.**
* **Plugins:**
* **Extensions for browser automation, such as Chrome Developer Tools.** 
  1. **Network Requirements**
* **Bandwidth:** Minimum 10 Mbps for smooth access and testing.
* **Connectivity:** Stable connection to avoid disruptions during testing.

**9. Manual Testing - Preparation of Test Cases and Manual Testing for the Application Date: 05/12/2024**

**A. User Authentication (Registration & Login)**

| **Test Case ID** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC1 | Enter valid username, email, and password | Username: John, Email: [johndoe@gmail.com](mailto:johndoe@gmail.com),  Password: Test@123 | Redirect to login page | Redirected successfully | Pass |
| TC2 | Enter invalid email format | Username: John, Email: john#gmail.com, Password: John@123 | Error message: "Invalid email format" | Displayed correctly | Pass |
| TC3 | Leave password field empty | Username: John, Email: [john@gmail.com](mailto:john@gmail.com), Password: | Alert message: "Please fill out this field" | Displayed correctly | Pass |

**B. Product Management**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| TC05 | Add a new product | 1. Login as Admin 2. Navigate to "Add Product" 3. Enter product details and save | Name: Laptop, Price: ₹50,000 | Product added successfully | Product added successfully | Pass |
| TC06 | Edit product details | 1. Login as Admin 2. Edit a product 3. Save changes | Name: Gaming Laptop, Price: ₹55,000 | Product updated successfully | Product updated successfully | Pass |
| TC07 | Delete a product | 1. Login as Admin 2. Delete a product | Product ID: 123 | Product removed successfully | Product removed successfully | Pass |

**C. Shopping Cart & Checkout**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| TC08 | Add product to cart | 1. Open product page 2. Click "Add to Cart" | Product: Laptop | Product added successfully | Product added successfully | Pass |
| TC09 | Remove product from cart | 1. Open cart 2. Click "Remove" | Product: Laptop | Product removed successfully | Product removed successfully | Pass |
| TC10 | Increase quantity of product in cart | 1. Change quantity of an item in the cart | Quantity: 2 | Quantity updated successfully | Quantity updated successfully | Pass |
| TC011 | Checkout with valid details | 1. Click "Checkout" 2. Enter address & payment info 3. Confirm order | Address: XYZ, Payment: Card | Order placed successfully | Order placed successfully | Pass |
| TC012 | Checkout without payment details | 1. Click "Checkout" without entering payment details | - | Error message "Payment details required" | - | Fail |

**D. Order Management**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| TC013 | View order history | 1. Login 2. Navigate to "My Orders" | - | Display user's past orders | Display user's past orders | Pass |
| TC014 | Cancel an order | 1. Open order history 2. Click "Cancel Order" | Order ID: 456 | Order cancelled successfully | - | Fail |

**10. Manual Testing - Preparation of Test Cases and Manual Testing for the Web Page Date:11/12/2024**

**A. User Authentication (Registration & Login)**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| TC001 | Register with valid details | 1. Open registration page 2. Enter valid details 3. Click "Register" | Name: Alex, Email: alex@gmail.com, Password: Fit@123 | Account created successfully | Account created successfully | Pass |
| TC002 | Register with an existing email | 1. Enter an existing email 2. Click "Register" | Email: alex@gmail.com | Error message "Email already exists" | Error message "Email already exists" | Pass |
| TC003 | Login with correct credentials | 1. Enter valid email and password 2. Click "Login" | Email: alex@gmail.com, Password: Fit@123 | Redirect to user dashboard | Redirect to user dashboard | Pass |
| TC004 | Login with incorrect credentials | 1. Enter correct email and wrong password 2. Click "Login" | Email: alex@gmail.com, Password: WrongPass | Error message "Invalid credentials" | Error message "Invalid credentials" | Pass |

**B. Product Management**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| TC005 | View product details | 1. Click on a product | Product: Gym T-shirt | Display product details | Display product details | Pass |
| TC006 | Search for a product | 1. Use search bar to find "leggings" | Search: "leggings" | Display matching results | Display matching results | Pass |
| TC007 | Filter products by category | 1. Select "T-shirts" from filters | Category: "T-shirts" | Display filtered results | Display filtered results | Pass |

**C. Shopping Cart & Checkout**

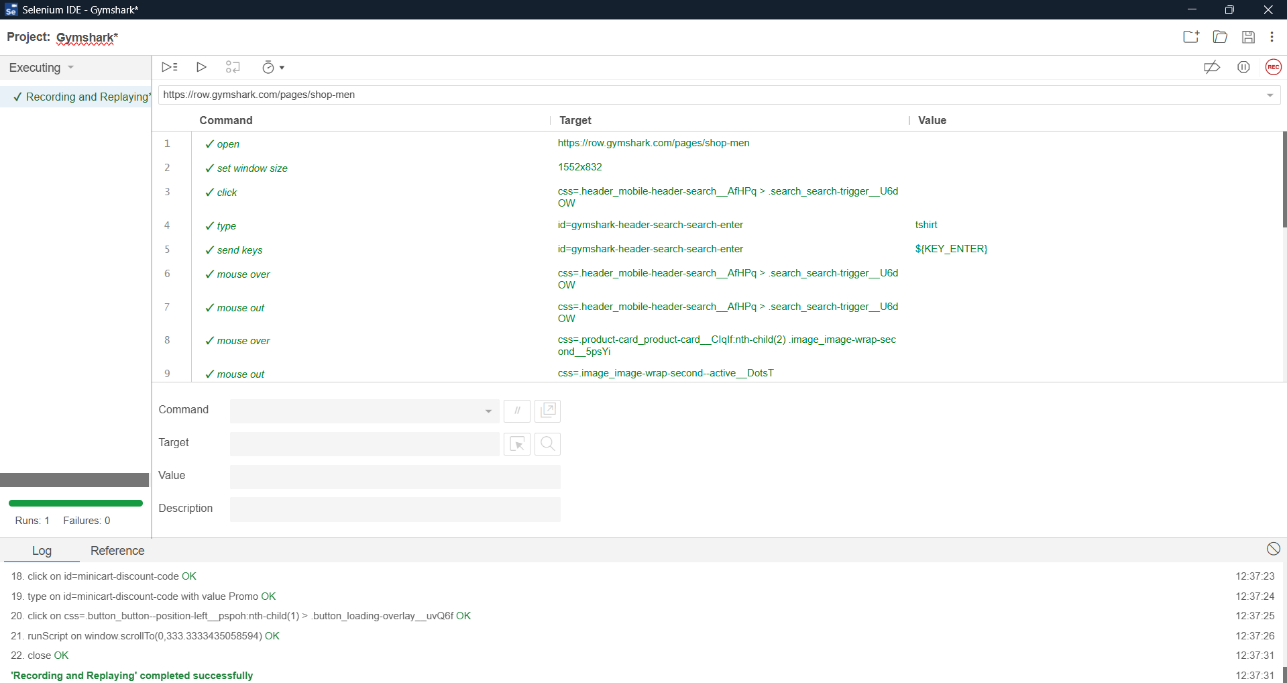
| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| TC008 | Add product to cart | 1. Open product page 2. Click "Add to Cart" | Product: Gym Shorts | Product added successfully | Product added successfully | Pass |
| TC009 | Remove product from cart | 1. Open cart 2. Click "Remove" | Product: Gym Shorts | Product removed successfully | Product removed successfully | Pass |
| TC010 | Update product quantity | 1. Change quantity of an item in the cart | Quantity: 2 | Quantity updated successfully | Quantity updated successfully | Pass |
| TC011 | Checkout with valid details | 1. Click "Checkout" 2. Enter address & payment info 3. Confirm order | Address: XYZ, Payment: Card | Order placed successfully | Order placed successfully | Pass |
| TC012 | Checkout without payment details | 1. Click "Checkout" without entering payment details | - | Error message "Payment details required" | Error message "Payment details required" | Pass |

**D. Payment Processing**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| TC013 | Successful payment | 1. Select payment method 2. Enter valid card details 3. Confirm payment | Card: Valid | Payment successful, order confirmed | Payment successful, order confirmed | Pass |
| TC014 | Payment failure | 1. Enter incorrect card details 2. Try processing payment | Card: Invalid | Payment failure message displayed | Payment failure message displayed | Pass |

* 1. **Selenium IDE: Recording and Replay – For a given web page:**

**Date: 12/12/2024**

****

|  |  |  |
| --- | --- | --- |
| **Command** | **Target** | **Value** |
| open | <https://row.gymshark.com/pages/shop-men> |  |
| set window size | 1552x832 |  |
| click | css=.header\_mobile-header-search\_\_AfHPq > .search\_search-trigger\_\_U6dOW |  |
| type | id=gymshark-header-search-search-enter |  |
| send keys | id=gymshark-header-search-search-enter | ${KEY\_ENTER} |
| mouse over | css=.header\_mobile-header-search\_\_AfHPq > .search\_search-trigger\_\_U6dOW |  |
| mouse out | css=.header\_mobile-header-search\_\_AfHPq > .search\_search-trigger\_\_U6dOW |  |
| mouse over | css=.product-card\_product-card\_\_CIqIf:nth-child(2) .image\_image-wrap-second\_\_5psYi |  |
| mouse out | css=.image\_image-wrap-second--active\_\_DotsT |  |
| mouse over | css=.product-card\_product-card\_\_CIqIf:nth-child(1) .image\_image-wrap-second\_\_5psYi |  |
| click | css=.image\_image-wrap-second--active\_\_DotsT |  |
| mouse over | css=.image-gallery\_gallery--item\_\_OW3UF:nth-child(2) > img |  |
| click | css=.add-to-cart\_section\_\_GygMh:nth-child(5) .size\_size\_\_saJiQ:nth-child(5) |  |
| mouse over | css=div > .product-card\_product-card\_\_CIqIf:nth-child(1) .product-card\_product-title-link\_\_jDI6f |  |
| mouse out | css=div > .product-card\_product-card\_\_CIqIf:nth-child(1) .product-card\_product-title-link\_\_jDI6f |  |
| click | id=minicart-discount-code |  |
| type | id=minicart-discount-code | Promo |
| click | css=.button\_button--position-left\_\_pspoh:nth-child(1) > .button\_loading-overlay\_\_uvQ6f |  |
| run script | window.scrollTo(0,333.3333435058594) |  |
| close |  |  |

**Python Pytest:**

# Generated by Selenium IDE  
import pytest  
import time  
import json  
from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.common.action\_chains import ActionChains  
from selenium.webdriver.support import expected\_conditions  
from selenium.webdriver.support.wait import WebDriverWait  
from selenium.webdriver.common.keys import Keys  
from selenium.webdriver.common.desired\_capabilities import DesiredCapabilities  
class TestRecordingandReplaying():  
 def setup\_method(self, method):  
 self.driver = webdriver.Chrome()  
 self.vars = {}  
 def teardown\_method(self, method):  
 self.driver.quit()  
 def test\_recordingandReplaying(self):  
 self.driver.get("https://row.gymshark.com/pages/shop-men")  
 self.driver.set\_window\_size(1552, 832)  
 self.driver.find\_element(By.CSS\_SELECTOR, ".header\_mobile-header-search\_\_AfHPq > .search\_search-trigger\_\_U6dOW").click()  
 self.driver.find\_element(By.ID, "gymshark-header-search-search-enter").send\_keys("tshirt")  
 self.driver.find\_element(By.ID, "gymshark-header-search-search-enter").send\_keys(Keys.ENTER)  
 element = self.driver.find\_element(By.CSS\_SELECTOR, ".header\_mobile-header-search\_\_AfHPq > .search\_search-trigger\_\_U6dOW")  
 actions = ActionChains(self.driver)  
 actions.move\_to\_element(element).perform()  
 element = self.driver.find\_element(By.CSS\_SELECTOR, "body")  
 actions = ActionChains(self.driver)  
 actions.move\_to\_element(element, 0, 0).perform()  
 element = self.driver.find\_element(By.CSS\_SELECTOR, ".product-card\_product-card\_\_CIqIf:nth-child(2) .image\_image-wrap-second\_\_5psYi")  
 actions = ActionChains(self.driver)  
 actions.move\_to\_element(element).perform()  
 element = self.driver.find\_element(By.CSS\_SELECTOR, "body")  
 actions = ActionChains(self.driver)  
 actions.move\_to\_element(element, 0, 0).perform()  
 element = self.driver.find\_element(By.CSS\_SELECTOR, ".product-card\_product-card\_\_CIqIf:nth-child(1) .image\_image-wrap-second\_\_5psYi")  
 actions = ActionChains(self.driver)  
 actions.move\_to\_element(element).perform()  
 self.driver.find\_element(By.CSS\_SELECTOR, ".image\_image-wrap-second--active\_\_DotsT").click()  
 element = self.driver.find\_element(By.CSS\_SELECTOR, ".image-gallery\_gallery--item\_\_OW3UF:nth-child(2) > img")  
 actions = ActionChains(self.driver)  
 actions.move\_to\_element(element).perform()  
 element = self.driver.find\_element(By.CSS\_SELECTOR, "body")  
 actions = ActionChains(self.driver)  
 actions.move\_to\_element(element, 0, 0).perform()  
 self.driver.find\_element(By.CSS\_SELECTOR, ".add-to-cart\_section\_\_GygMh:nth-child(5) .size\_size\_\_saJiQ:nth-child(5)").click()  
 self.driver.find\_element(By.CSS\_SELECTOR, ".add-to-cart\_button-container\_\_aKAUo:nth-child(6) .button\_success-overlay\_\_aqLb7").click()  
 element = self.driver.find\_element(By.CSS\_SELECTOR, "div > .product-card\_product-card\_\_CIqIf:nth-child(1) .product-card\_product-title-link\_\_jDI6f")  
 actions = ActionChains(self.driver)  
 actions.move\_to\_element(element).perform()  
 element = self.driver.find\_element(By.CSS\_SELECTOR, "body")  
 actions = ActionChains(self.driver)  
 actions.move\_to\_element(element, 0, 0).perform()  
 self.driver.find\_element(By.ID, "minicart-discount-code").click()  
 self.driver.find\_element(By.ID, "minicart-discount-code").send\_keys("Promo")  
 self.driver.find\_element(By.CSS\_SELECTOR, ".button\_button--position-left\_\_pspoh:nth-child(1) > .button\_loading-overlay\_\_uvQ6f").click()  
 self.driver.execute\_script("window.scrollTo(0,333.3333435058594)")  
 self.driver.close()

* 1. **Selenium IDE: Commands: Date: 18/12/2024**

**1. Navigation Commands**

| **Command** | **Description** |
| --- | --- |
| open | Opens a specified URL. |
| goBack | Navigates to the previous page in browser history. |
| goForward | Moves forward in browser history. |
| refresh | Reloads the current page. |

**2. Interaction Commands**

| **Command** | **Description** |
| --- | --- |
| Click | Clicks on an element. |
| clickAt | Clicks on an element at a specific coordinate. |
| doubleClick | Performs a double-click on an element. |
| Type | Enters text into an input field. |
| sendKeys | Simulates keyboard input. |
| Check | Selects a checkbox. |
| Uncheck | Deselects a checkbox. |
| Select | Selects an option from a dropdown. |
| chooseOkOnNextConfirmation | Confirms an alert pop-up. |
| chooseCancelOnNextConfirmation | Cancels an alert pop-up. |

**3. Assertion & Verification Commands**

| **Command** | **Description** |
| --- | --- |
| assertTitle | Verifies the page title (fails test if not matched). |
| assertText | Verifies if an element contains the expected text. |
| assertElementPresent | Ensures an element exists. |
| assertElementNotPresent | Ensures an element does not exist. |
| verifyText | Similar to assertText but does not stop execution on failure. |
| verifyElementPresent | Similar to assertElementPresent but does not stop execution on failure. |
| verifyChecked | Checks if a checkbox is selected. |

**4. Store Commands (Variable Handling)**

| **Command** | **Description** |
| --- | --- |
| store | Stores a value in a variable. |
| storeTitle | Stores the page title in a variable. |
| storeText | Stores text from an element. |
| storeValue | Stores the value of an input field. |
| storeAttribute | Stores an attribute of an element. |

**5. Wait Commands**

| **Command** | **Description** |
| --- | --- |
| waitForElementPresent | Waits for an element to be available on the page. |
| waitForElementNotPresent | Waits until an element is removed. |
| waitForText | Waits until an element contains specific text. |
| waitForValue | Waits until an input field has a specific value. |

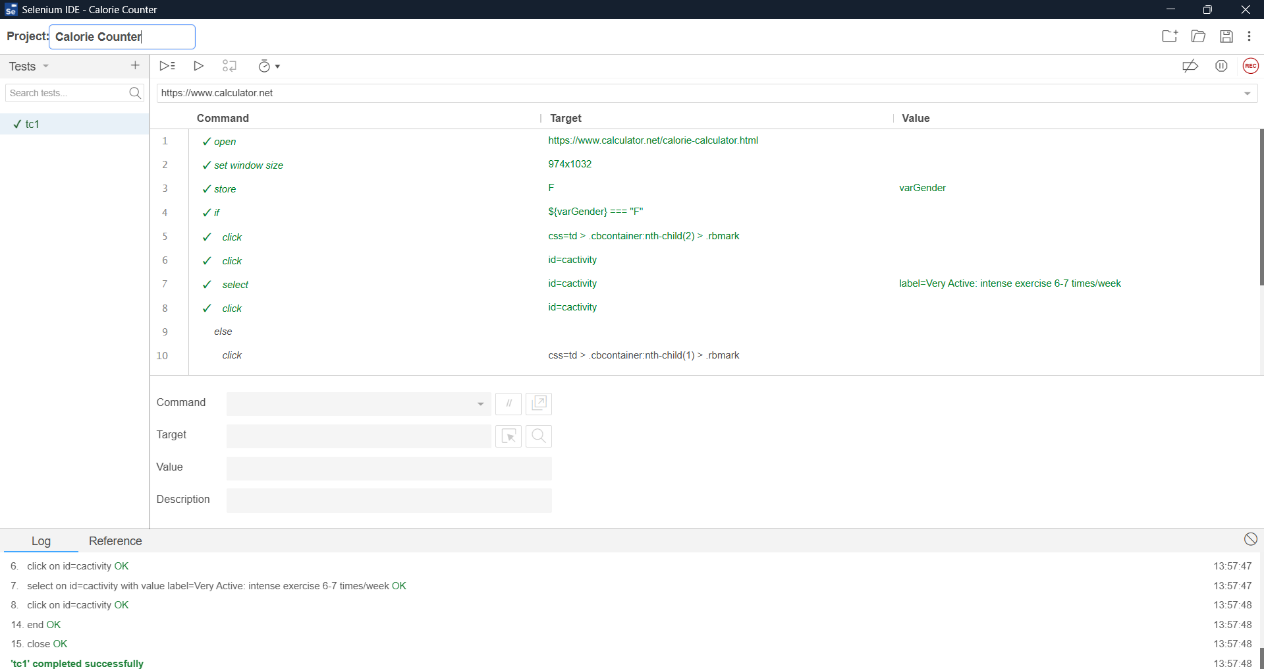
**6. Control Flow Commands**

| **Command** | **Description** |
| --- | --- |
| if...else | Executes a block of commands conditionally. |
| while | Repeats a block of commands while a condition is true. |
| do...repeatIf | Repeats a set of commands until a condition is met. |
| forEach | Iterates over an array of values. |
| break | Exits from a loop. |
| end | Ends an if, while, or forEach block. |

**7. Debugging & Execution Control Commands**

| **Command** | **Description** |
| --- | --- |
| pause | Stops execution for a specified time (in milliseconds). |
| echo | Logs a message in the execution log. |
| setSpeed | Sets the execution speed of Selenium IDE. |
| runScript | Executes JavaScript code on the page. |

* 1. **Selenium IDE: Conditional Statements: Date: 26/12/2024**

****

**Python Pytest:**

# Generated by Selenium IDE

import pytest

import time

import json

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.action\_chains import ActionChains

from selenium.webdriver.support import expected\_conditions

from selenium.webdriver.support.wait import WebDriverWait

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.desired\_capabilities import DesiredCapabilities

class TestNew1():

def setup\_method(self, method):

self.driver = webdriver.Chrome()

self.vars = {}

def teardown\_method(self, method):

self.driver.quit()

def test\_new1(self):

self.driver.get("https://www.calculator.net/calorie-calculator.html")

self.vars["varGender"] = "F"

if self.driver.execute\_script("return (arguments[0]===\"F\")", self.vars["varGender"]):

self.driver.find\_element(By.CSS\_SELECTOR, "td>.cbcontainer:nth-child(2)>.rbmark").click()

self.driver.find\_element(By.ID, "cactivity").click()

dropdown = self.driver.find\_element(By.ID, "cactivity")

dropdown.find\_element(By.XPATH, "//option[. = 'Very Active: intense exercise 6-7 times/week']").click()

self.driver.find\_element(By.ID, "cactivity").click()

else:

self.driver.find\_element(By.CSS\_SELECTOR, "td>.cbcntainer:nth-child(1)>.rbmark").click()

self.driver.find\_element(By.ID, "cactivity").click()

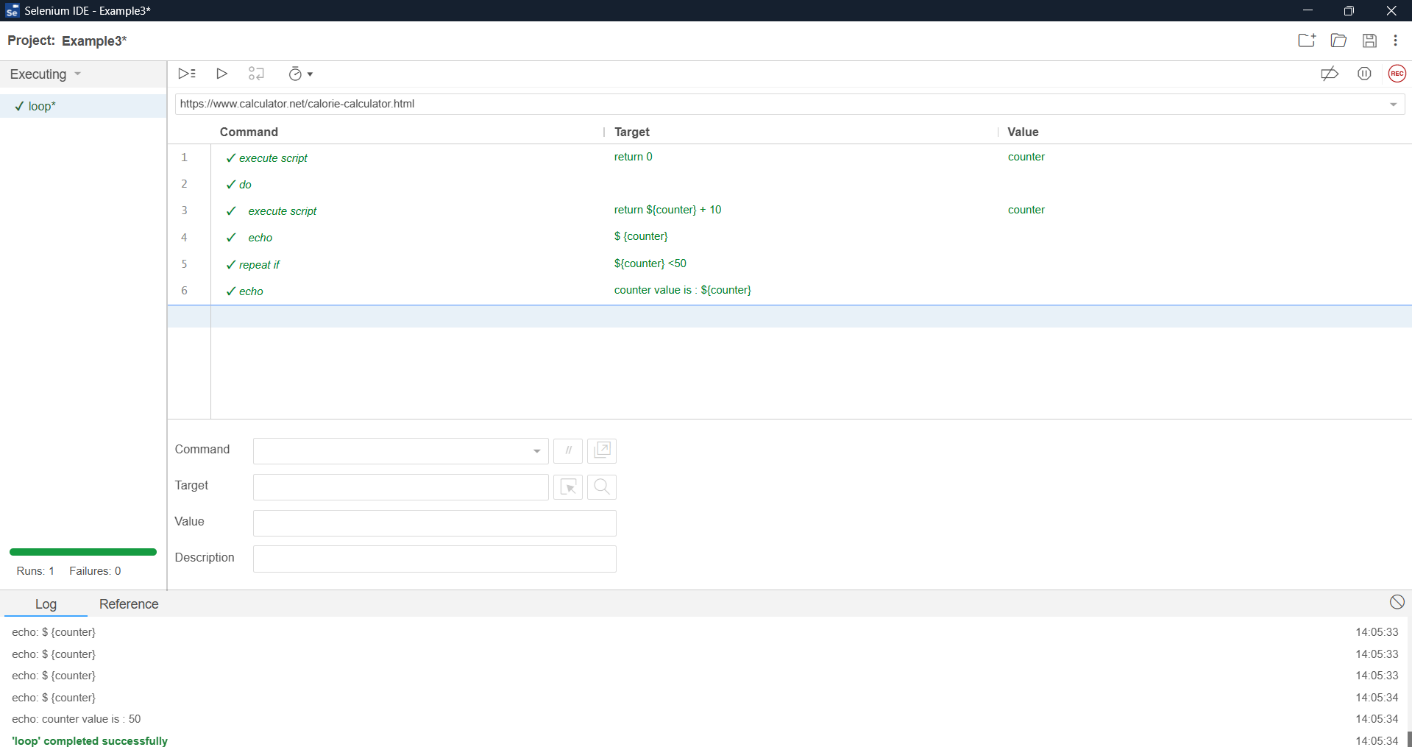
dropdown = self.driver.find\_element(By.ID, "cactivity")

dropdown.find\_element(By.XPATH, "//option[. = '1']").click()

self.driver.find\_element(By.ID, "cactivity").click()

self.driver.close()

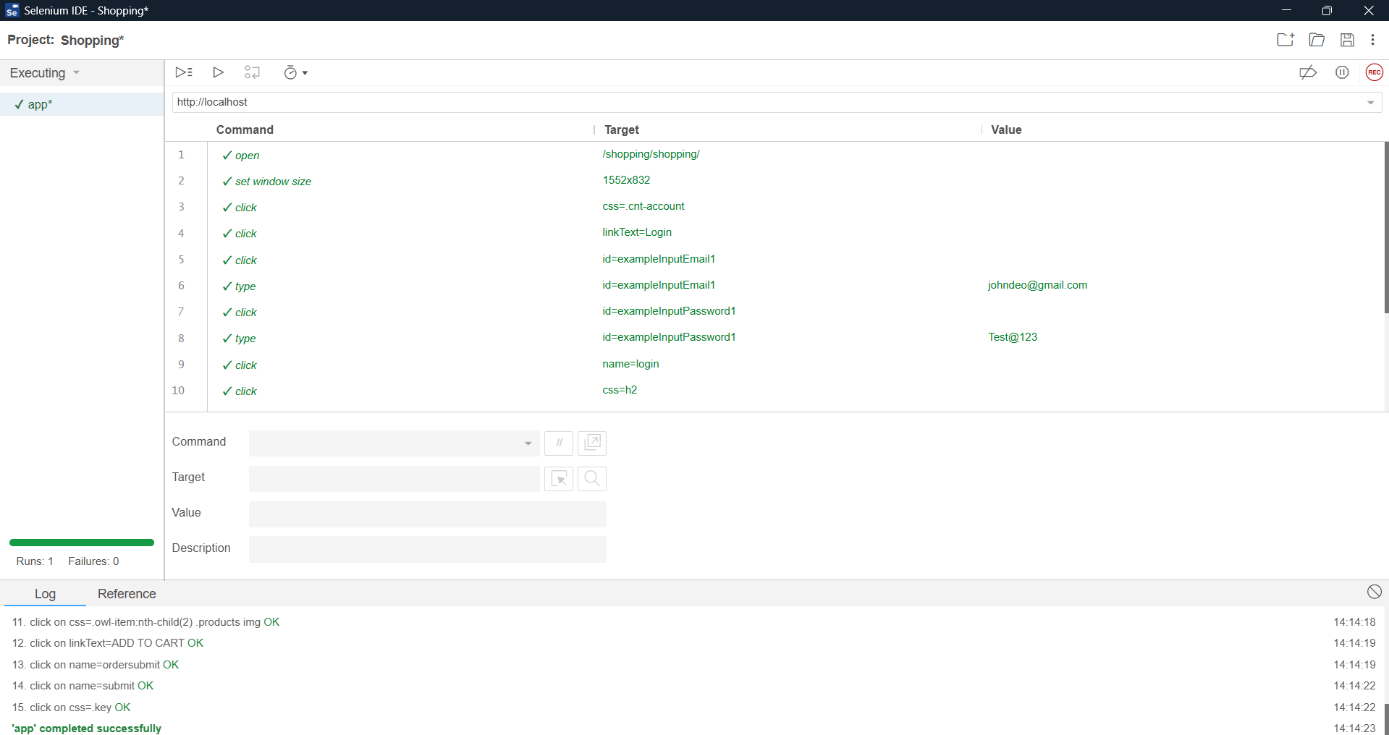
* 1. **Selenium IDE: Loops: Date: 01/01/2025**

****

**Python Pytest:**

# Generated by Selenium IDE  
import pytest  
import time  
import json  
from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.common.action\_chains import ActionChains  
from selenium.webdriver.support import expected\_conditions  
from selenium.webdriver.support.wait import WebDriverWait  
from selenium.webdriver.common.keys import Keys  
from selenium.webdriver.common.desired\_capabilities import DesiredCapabilities  
class TestLoop():  
 def setup\_method(self, method):  
 self.driver = webdriver.Chrome()  
 self.vars = {}  
 def teardown\_method(self, method):  
 self.driver.quit()  
 def test\_loop(self):  
 self.vars["counter"] = self.driver.execute\_script("return 0")  
 condition = True  
 while condition:  
 self.vars["counter"] = self.driver.execute\_script("return arguments[0] + 10", self.vars["counter"])  
 print("$ {counter}")  
 condition = self.driver.execute\_script("return (arguments[0] <50)", self.vars["counter"])  
 print("counter value is : {}".format(self.vars["counter"]))

* 1. **Selenium IDE: For the Application: Date: 02/01/2025**

****

**Python Pytest:**

# Generated by Selenium IDE  
import pytest  
import time  
import json  
from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.common.action\_chains import ActionChains  
from selenium.webdriver.support import expected\_conditions  
from selenium.webdriver.support.wait import WebDriverWait  
from selenium.webdriver.common.keys import Keys  
from selenium.webdriver.common.desired\_capabilities import DesiredCapabilities  
class TestApp():  
 def setup\_method(self, method):  
 self.driver = webdriver.Chrome()  
 self.vars = {}  
 def teardown\_method(self, method):  
 self.driver.quit()  
 def test\_app(self):  
 self.driver.get("http://localhost/shopping/shopping/")  
 self.driver.set\_window\_size(1552, 832)  
 self.driver.find\_element(By.CSS\_SELECTOR, ".cnt-account").click()  
 self.driver.find\_element(By.LINK\_TEXT, "Login").click()  
 self.driver.find\_element(By.ID, "exampleInputEmail1").click()  
 self.driver.find\_element(By.ID, "exampleInputEmail1").send\_keys("johndeo@gmail.com")  
 self.driver.find\_element(By.ID, "exampleInputPassword1").click()  
 self.driver.find\_element(By.ID, "exampleInputPassword1").send\_keys("Test@123")  
 self.driver.find\_element(By.NAME, "login").click()  
 self.driver.find\_element(By.CSS\_SELECTOR, "h2").click()  
 self.driver.find\_element(By.CSS\_SELECTOR, ".owl-item:nth-child(2) .products img").click()  
 self.driver.find\_element(By.LINK\_TEXT, "ADD TO CART").click()  
 self.driver.find\_element(By.NAME, "ordersubmit").click()  
 self.driver.find\_element(By.NAME, "submit").click()  
 self.driver.find\_element(By.CSS\_SELECTOR, ".key").click()

* 1. **Selenium Web driver – Locating web element – id, name, link\_text, partial\_ link\_text – Application: Date: 08/01/2025**

from selenium import webdriver

from selenium.webdriver.common.by import By

driver = webdriver.Chrome()

driver.get("http://localhost/shopping/shopping/")

driver.maximize\_window()

element\_by\_id = driver.find\_element(By.ID, "lightbox")

print("Element found by ID")

element\_by\_name = driver.find\_element(By.NAME, "product")

print("Element found by Name")

element\_by\_link\_text = driver.find\_element(By.LINK\_TEXT, "My Account")

print("Element found by Link Text")

element\_by\_partial\_link\_text = driver.find\_element(By.PARTIAL\_LINK\_TEXT, "Account")

print("Element found by Partial Link Text")

driver.quit()

* 1. **Selenium Web driver – Locating web element – id, name, link\_text, partial\_ link\_text – Webpage: Date: 09/12/2025**

from selenium import webdriver

from selenium.webdriver.common.by import By

driver = webdriver.Chrome()

driver.get("https://row.gymshark.com/pages/shop-men")

driver.maximize\_window()

element\_by\_id = driver.find\_element(By.ID, "carousel-heading")

print("Element found by ID")

element\_by\_name = driver.find\_element(By.NAME, "description")

print("Element found by Name")

element\_by\_link\_text = driver.find\_element(By.LINK\_TEXT, "Accessibility Statement")

print("Element found by Link Text")

element\_by\_partial\_link\_text = driver.find\_element(By.PARTIAL\_LINK\_TEXT, "Statement")

print("Element found by Partial Link Text")

driver.quit()

* 1. **Selenium Web driver – Locating web element – class name and tag name – Application: Date: 15/01/2025**

from selenium import webdriver

from time import sleep

import time

from selenium.webdriver import Keys

from selenium.webdriver.common.by import By

driver = webdriver.Chrome()

driver.get("http://localhost/shopping/shopping/")

driver.maximize\_window()

sleep(1)

driver.find\_element(By.CLASS\_NAME, "search-field").click()

driver.find\_element(By.CLASS\_NAME, "search-field").send\_keys("Shoes")

driver.find\_element(By.CLASS\_NAME, "search-field").send\_keys(Keys.ENTER)

element\_by\_tag\_name = driver.find\_element(By.TAG\_NAME, "p")

print("\nElement found by Tag Name\n")

driver.quit()

* 1. **Selenium Web driver – Locating web element – class name and tag name – Webpage Date: 16/01/2025**

from selenium import webdriver

from time import sleep

import time

from selenium.webdriver import Keys

from selenium.webdriver.common.by import By

driver = webdriver.Chrome()

driver.get("https://row.gymshark.com/pages/shop-men")

driver.maximize\_window()

time.sleep(1)

driver.find\_element(By.CLASS\_NAME, "header\_mobile-header-search\_\_AfHPq").click()

time.sleep(1)

driver.find\_element(By.ID, "gymshark-header-search-search-enter").send\_keys("running")

driver.find\_element(By.ID, "gymshark-header-search-search-enter").send\_keys(Keys.ENTER)

element\_by\_tag\_name = driver.find\_element(By.TAG\_NAME, "p")

print("\nElement found by Tag Name\n\n")

driver.quit()

* 1. **Selenium Web driver – Locating web element – CSSSelector – Application:**

**Date: 22/01/2025**

from selenium import webdriver

import time

from selenium.webdriver import Keys

from selenium.webdriver.common.by import By

driver = webdriver.Chrome()

driver.get("http://localhost/shopping/shopping/")

driver.maximize\_window()

time.sleep(1) #Class Selector

driver.find\_element(By.CSS\_SELECTOR, "a .fa-sign-in").click()

time.sleep(3) #ID Selector

driver.find\_element(By.CSS\_SELECTOR, '#exampleInputEmail1').send\_keys("johndeo@gmail.com")

time.sleep(2) # Attribute Selector

driver.find\_element(By.CSS\_SELECTOR, "input[name='password']").send\_keys("Test@123")

time.sleep(2) # Combinator Selector

driver.find\_element(By.CSS\_SELECTOR, 'form button[name="login"]').click()

driver.find\_element(By.LINK\_TEXT, "Shopping Portal").click()

time.sleep(2)print("Login Successful\n")

driver.quit()

* 1. **Selenium Web driver – Locating web element – CSSSelector – Webpage:**

**Date: 23/02/2025**

from selenium import webdriver

import time

from selenium.webdriver import Keys

from selenium.webdriver.common.by import By

driver = webdriver.Chrome()

driver.get("https://row.gymshark.com/pages/shop-men")

driver.maximize\_window()

time.sleep(1)

driver.find\_element(By.CSS\_SELECTOR, ".header\_action-bar-item-wrap\_\_m\_Rhz .icon-user").click()

time.sleep(2) #Class Selector

driver.find\_element(By.CSS\_SELECTOR, "input.input.cec61d1fe.c54babdaf").click()

time.sleep(2) #ID Selector

driver.find\_element(By.CSS\_SELECTOR, "#username").send\_keys("john@gmail.com")

time.sleep(2) #Attribute Selector

driver.find\_element(By.CSS\_SELECTOR, "input[id='password']").click()

time.sleep(2) #Combinators

driver.find\_element(By.CSS\_SELECTOR,"input[type='password'][autocomplete='current-password']").send\_keys("Test!@3")

time.sleep(2)

driver.find\_element(By.CSS\_SELECTOR, ".ulp-button-icon").click()

time.sleep(2)

driver.quit()

* 1. **Selenium Web driver – Locating web element – xpath – Application:**

**Date: 24/01/2025**

from selenium import webdriver

import time

from selenium.webdriver import Keys

from selenium.webdriver.common.by import By

driver = webdriver.Chrome()

driver.get("http://localhost/shopping/shopping/")

driver.maximize\_window()

time.sleep(3)

driver.find\_element(By.XPATH, "/html/body/header/div[2]/div/div/div[2]/div/form/div/input").send\_keys("Phone")

time.sleep(3)

driver.find\_element(By.XPATH, "/html/body/header/div[2]/div/div/div[2]/div/form/div/button").click()

time.sleep(3)

# product.send\_key(Keys.ENTER)

driver.find\_element(By.XPATH, "/html/body/div[1]/div/div[1]/div[2]/div[2]/div/div/div/div/div/div/div/div[1]/div/a/img").click()

time.sleep(3)

driver.find\_element(By.XPATH,"/html/body/div[2]/div/div[1]/div[2]/div[1]/div[2]/div/div[6]/div/div[3]/a").click()

time.sleep(3)

driver.find\_element(By.XPATH, "/html/body/div[2]/div/div[1]/div/div[4]/table/tbody/tr/td/div/button").click()

time.sleep(3)

driver.quit()

* 1. **Selenium Web driver – Locating web element – xpath – Webpage:**

**Date: 29/01/2025**

from selenium import webdriver

import time

from selenium.webdriver import Keys

from selenium.webdriver.common.by import By

driver = webdriver.Chrome()

driver.get("https://row.gymshark.com/pages/shop-men")

driver.maximize\_window()

time.sleep(1)

driver.find\_element(By.CSS\_SELECTOR, ".header\_mobile-header-search\_\_AfHPq > .search\_search-trigger\_\_U6dOW").click()

time.sleep(1)

driver.find\_element(By.CSS\_SELECTOR, "body")

time.sleep(1)

driver.find\_element(By.ID, "gymshark-header-search-search-enter").send\_keys("running")

time.sleep(2)

driver.find\_element(By.ID, "gymshark-header-search-search-enter").send\_keys(Keys.ENTER)

time.sleep(2)

driver.find\_element(By.CLASS\_NAME, "product-card\_product-title-link\_\_jDI6f").click()

time.sleep(3)

driver.find\_element(By.CSS\_SELECTOR, ".add-to-cart\_section\_\_GygMh:nth-child(5) .size\_size\_\_saJiQ:nth-child(5)").click()

time.sleep(3)

driver.find\_element(By.CSS\_SELECTOR, ".add-to-cart\_button-container\_\_aKAUo:nth-child(6) .button\_success-overlay\_\_aqLb7").click()

time.sleep(4)

driver.find\_element(By.XPATH, "/html/body/div[6]/div/div/div[2]/div[3]/div/div[4]/div/div/fieldset/input").click()

time.sleep(2)

driver.find\_element(By.CLASS\_NAME, "input\_input\_\_iz\_H4").send\_keys("PROMO")

time.sleep(2)

driver.find\_element(By.XPATH, "/html/body/div[6]/div/div/div[2]/div[3]/div/div[4]/div/div/button/span[2]").click()

time.sleep(2)

print("\nOrder Placed Successfully")

driver.quit()

* 1. **Application commands and Browser commands: Date: 30/01/2025**

from selenium import webdriver

driver = webdriver.Chrome()

import time

#application command

driver.get("http://localhost/shopping/shopping/")

time.sleep(3)

print("Title:", driver.title)

time.sleep(1)

current\_url = driver.current\_url

print("Current URL:", current\_url)

time.sleep(1)

#browser command

driver.maximize\_window()

time.sleep(3)

driver.minimize\_window()

time.sleep(3)

driver.maximize\_window()

time.sleep(3)

driver.quit()

* 1. **Conditional Commands and Navigation Commands. Date: 31/01/2025**

from selenium import webdriver

from selenium.webdriver.common.by import By

import time

driver = webdriver.Chrome()

driver.get("http://localhost/shopping/shopping/")

# Conditional Commands

element = driver.find\_element(By.TAG\_NAME, "body")

if element.is\_displayed():

print("Element is displayed")

if element.is\_enabled():

print("Element is enabled")

if element.is\_selected():

print("Element is selected")

time.sleep(3)

# Navigation Commands

driver.get("https://row.gymshark.com/pages/shop-men")

time.sleep(3)

print("Navigated to Application")

driver.back()

time.sleep(3)

print("Navigated back")

driver.forward()

time.sleep(4)

print("Navigated forward")

driver.refresh()

print("Page refreshed again")

driver.quit()

* 1. **Difference between find\_element and find\_elements: Date: 05/02/2025**

from selenium import webdriver

from selenium.webdriver.common.by import By

import time

driver = webdriver.Chrome()

driver.get("http://localhost/shopping/shopping/")

element = driver.find\_element(By.TAG\_NAME, "a")

print("Single element found by tag name:", element.text)

elements = driver.find\_elements(By.TAG\_NAME, "p")

print("\n\nMultiple elements found by tag name:")

for e in elements:

print(e.text)

driver.close()

* 1. **Difference between text and get\_attribute: Date: 06/02/2025**

from selenium import webdriver

from selenium.webdriver.common.by import By

import time

driver = webdriver.Chrome()

driver.get("http://localhost/shopping/shopping/")

element = driver.find\_element(By.XPATH, "/html/body/header/div[2]/div/div/div[1]/div/a")

print("Element text:", element.text)

print("Element href attribute:", element.get\_attribute("href"))

driver.quit()

**28. Wait Commands Date: 07/02/2025**

from selenium import webdriver

import time

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

driver = webdriver.Chrome()

driver.get("http://localhost/shopping/shopping/")

driver.maximize\_window()

wait = WebDriverWait(driver, 10)

product\_input = wait.until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "input[name=product]")))

product\_input.send\_keys("Book")

time.sleep(3)

print("Entered 'Book' in the product input field.")

search\_button = wait.until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "button[name=search]")))

search\_button.click()

print("Clicked the search button.")

driver.quit()