

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANASANGAMA, BELGAVI-590018



Software Testing Laboratory (21ISL66)
Automated Acceptance Testing Report
on

“E-Commerce Website: ADIDAS Using Selenium Tool”

Submitted in partial fulfilment of the requirements for the 6th Semester

Bachelor of Engineering
in

INFORMATION SCIENCE AND ENGINEERING

Submitted by

ATHARV AMIT GANGRADE

1BI21IS020

SHREYAS RAJ

1BI21IS119

Under the guidance of

Dr. Hema Jagadish

Associate Professor
Department of ISE,
BIT, Bangalore.

Prof. Padmanabha J

Assistant Professor
Department of ISE,
BIT, Bangalore.



DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
BANGALORE INSTITUTE OF TECHNOLOGY
K. R. Road, V.V. Pura, Bengaluru-560004

2023-2024

BANGALORE INSTITUTE OF TECHNOLOGY

K.R. Road, V.V. Pura, Bengaluru -560004

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the implementation of entitled “**SOFTWARE TESTING LABORATORY (21ISL66)**” has been successfully completed by **ATHARV AMIT GANGRADE (1BI21IS020)** and **SHREYAS RAJ (1BI21IS119)** of VI semester B.E. for the partial fulfilment of the requirements for the Bachelor's degree in Information Science & Engineering of the **Visvesvaraya Technological University** during the academic year 2023-2024.

Lab In charge:

Dr. Hema Jagadish

Associate Professor

Dept. of ISE, BIT

Prof. Padmanabha J

Assistant Professor

Dept. of ISE, BIT

HOD:

Dr. Asha T

Professor and Head

Dept. of ISE, BIT

Name of Examiners:

1.

2.

Signature with date

DECLARATION

We, ATHARV AMIT GANGRADE bearing USN (**1BI21IS020**) and **SHREYAS RAJ** bearing USN (**1BI21IS119**) student of VI Semester Bachelor of Engineering in Information Science Engineering, Bangalore Institute of Technology declare that the Automated Acceptance Testing project on “**E-Commerce Website: ADIDAS Using Selenium Tool**” work has been carried out by us and submitted in partial fulfilment of the requirements for the award of the degree of VI Semester degree of Bachelor of Engineering in **Information Science Engineering** of **Visvesvaraya Technological University, Belagavi** during the year 2023-2024.

ATHARV AMIT GANGRADE(1BI21IS020)

SHREYAS RAJ (1BI21IS119)

Place: Bangalore

Date:

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible. So, with gratitude, we acknowledge all those whose guidance and encouragement crowned my effort with success.

It's immense pleasure in thanking principal **Dr. Aswath M.U**, BIT, Bangalore, for providing an opportunity to present this project as a part of my curriculum in the partial fulfilment of the degree.

I express sincere gratitude for **Dr. Asha T**, Head of the Department, Information Science and Engineering for her co-operation and encouragement at all moments of my approach.

It is a pleasant duty to place on record my deepest sense of gratitude to my guides, **Dr. Hema Jagadish**, Associate Professor, Dept of ISE, and **Prof. Padmanabha J**, Assistant Professor, Dept of ISE, for their constant encouragement and valuable help and assistance in every possible way.

I would like to thank all teaching and non-teaching staff of ISE Department for providing their valuable guidance and for being there at all the stages of my world.

ATHARV AMIT GANGRADE(1BI21IS020)

SHREYAS RAJ (1BI21IS119)

ABSTRACT

This report documents the comprehensive testing processes conducted on the Adidas website, leveraging both manual and automated testing methodologies. Manual testing and automation testing are two distinct approaches used in software testing, each serving its own purpose and bringing unique advantages to the testing process.

Manual testing involves the manual execution of test cases, where testers perform actions on the software application, observe the system behaviour, and compare the actual results with the expected results. It relies on human intuition, creativity, and domain knowledge to identify defects and ensure the software meets the specified requirements. It is particularly effective in scenarios where human judgment is required, such as usability testing, ad hoc testing, and exploratory testing.

Automation testing involves the use of automated tools and scripts to execute test cases. It aims to improve efficiency, accuracy, and repeatability, and is especially beneficial for repetitive tasks, regression testing, and performance testing. Automation testing enables the execution of a large number of test cases in a short period, which is useful for projects with frequent updates or releases. Additionally, automation testing helps identify regressions introduced by code changes, ensuring that previously tested functionalities continue to work as expected.

In our testing process, we utilized Selenium as the automation tool. Selenium is a comprehensive testing tool that offers a wide range of features for functional testing, GUI testing, data-driven testing, and cross-platform testing. It supports multiple scripting languages, provides extensive object recognition capabilities, and offers robust reporting and debugging functionalities. With Selenium, we achieved increased test coverage, accelerated test execution, and improved overall test efficiency. By automating repetitive test cases, we freed up valuable time for our testers to focus on more exploratory testing and critical areas of the software. The report aims to validate the website's core functionalities, ensuring its reliability, performance, and user experience. The outcomes of this testing endeavour underscore the website's robustness and highlight areas for potential improvement.

TABLE OF CONTENTS

Chapter. No	Title	Page No.
	ABSTRACT	
1	MANUAL TESTING	1
	1.1 Test Scenarios and Test Cases	
2	AUTOMATION TESTING	4
	2.1 Introduction	
	2.2 Installation	
	2.3 Test Scenarios and Test Cases	
	2.4 Test Case Implementation	
	CONCLUSION	13
	REFERENCES	14

LIST OF FIGURES

Figure No.	Description	Page No.
2.1	Checking Node.js version	5
2.2	Installation of Selenium	5
2.3	Installation of Mocha	6
2.4	Installation of WebDriver	6
2.5	Verify Navigation to Adidas Homepage	8
2.6	Verify Clicking on a Category	9
2.7	Verify Filtering of Products	9
2.8	Verify Product Search	10
2.9	The results are logged in an Excel file	10

LIST OF TABLES

Table No.	Description	Page No.
1.1	Functional Requirements	2
1.2	Non - Functional Requirements	2
1.3	Test Cases of Adidas Website	3

Chapter 1

INTRODUCTION TO MANUAL TESTING

Manual testing is a fundamental process in software development, where testers manually execute test cases to validate the functionality, usability, and reliability of a software application. This approach involves following predefined steps to meticulously identify bugs, defects, and inconsistencies, ensuring the application operates as intended and meets the specified requirements. Unlike automated testing, which uses scripts and tools to run tests, manual testing simulates the real-world end-user experience. This human-centric approach allows testers to uncover potential issues that may not be detected through automation, such as usability challenges, subtle UI glitches, and complex navigation problems. Manual testing's ability to adapt to unique and exploratory scenarios makes it an invaluable component in delivering a high-quality user experience.

Adidas, as one of World's premier e-commerce platform, leverages a combination of manual and automated testing to uphold the quality and reliability of its services. The platform's vast range of features, from extensive product catalogues to intricate payment systems, requires a thorough examination to ensure a seamless user experience. Manual testing is crucial in this context, as it involves a detailed inspection of the website's functionalities and user interactions. Testers manually navigate through the website, verifying that each feature operates correctly, from product search and category browsing to the checkout process and customer service interactions. This testing method ensures that users can effortlessly search for products, view detailed descriptions, add items to their cart, and complete purchases without encountering issues. Furthermore, manual testers assess critical aspects like user account management, order tracking, and customer support functionalities, ensuring that users can register, log in, track their orders, and receive assistance when needed. By meticulously checking these elements, manual testing helps to identify and rectify issues that could disrupt the user experience, thus maintaining the platform's reputation for reliability and customer satisfaction. In addition to functionality, manual testing at Adidas focuses on the visual and aesthetic components, ensuring design consistency and responsiveness across devices and browsers. Testers verify that images, fonts, and layouts display correctly and that the platform meets accessibility standards. This meticulous attention to detail ensures a seamless and inclusive user experience, reinforcing Adidas's commitment to quality and customer satisfaction.

1.1 Test Scenarios and Test Cases

➤ Functional Requirements:

Table 1.1 Functional Requirements

S.No	Test Scenario ID	Test Scenario	Pre-conditions	Priority
1	TS001	Navigation to Adidas	Adidas website is accessible	High
2	TS002	Category Selection	Adidas website is accessible	High
3	TS003	Product Sorting	Adidas website is accessible	High
4	TS004	Product Search	Adidas website is accessible	High

➤ Non - Functional Requirements:

Table 1.2 Non - Functional Requirements

S.No	Test Scenario ID	Test Scenario	Pre-conditions	Priority
1	TS005	Performance - Page Load Time	Adidas website is accessible	High
2	TS006	Security - User Authentication	User credentials are valid and securely stored	High
3	TS007	Usability - Navigation and User Interface	Adidas website is accessible	Medium
4	TS008	Scalability - Handling Concurrent User Requests	Adidas website has multiple users accessing simultaneously	High
5	TS009	Compatibility - Browser and Device Compatibility	Adidas website is accessed from various browsers and devices	High
6	TS010	Reliability - Server Uptime	Adidas website server is running and accessible	High

➤ **Test cases:**

Table 1.3 Test Cases of Adidas Website

S.No	Test Scenario	Test Case	Test Scenario	Test Case	Description	Pre-conditions	Expected Result	Actual Result	Status
1	TS001	TC001	Navigation to Adidas	Verify homepage load	Verify that the Adidas homepage loads successfully	Adidas website is accessible	Homepage loads successfully	Homepage loaded successfully	Pass
		TC002	Navigation to Adidas	Verify page title	Verify that the page title is "Adidas Official Website"	Adidas US"	Adidas website is accessible	Page title matches expected title	Page title matches expected title
2	TS002	TC003	Category Selection	Verify category selection	Verify that a category can be selected on Adidas	Adidas website is accessible	Category page loads successfully	Category page loaded successfully	Pass
		TC004	Category Selection	Verify sub-category selection	Verify that a sub-category can be selected after selecting a category	Adidas website is accessible	Sub-category page loads successfully	Sub-category page loaded successfully	Pass
3	TS003	TC005	Product Sorting	Verify product sorting by price	Verify that products can be sorted by price: low to high	Adidas website is accessible	Products sorted by price correctly	Products sorted by price correctly	Pass
		TC006	Product Sorting	Verify product sorting by popularity	Verify that products can be sorted by popularity	Adidas website is accessible	Products sorted by popularity correctly	Products sorted by popularity correctly	Pass
4	TS004	TC007	Product Search	Verify product search	Verify that products can be searched	Adidas website is accessible	Relevant products displayed	Relevant products displayed	Pass
		TC008	Product Search	Verify search suggestions	Verify that search suggestions appear as the user types in the search bar	Adidas website is accessible	Search suggestions displayed correctly	Search suggestions displayed correctly	Pass

Chapter 2

AUTOMATION TESTING

2.1 Introduction

Automation testing involves using software tools and scripts to perform tests and verify the functionality of an application or system. Its primary goal is to enhance testing efficiency, minimize human error, and save time by automating repetitive and monotonous test cases. This approach can be integrated into various stages of the software development lifecycle, including unit testing, integration testing, functional testing, and regression testing. Automation testing facilitates rapid execution of tests, allows parallel testing, and provides detailed reporting, making it an essential part of modern software development practices.

Selenium is a widely-used automation testing tool for web applications, and it is highly suitable for testing the Adidas website. It is an open-source framework offering a comprehensive set of features for automating web browsers. Selenium supports multiple programming languages such as Java, C#, Python, Ruby, and JavaScript, providing flexibility in test automation. It features a user-friendly interface for creating, managing, and executing tests.

Key components of Selenium include:

1. **Selenium WebDriver:** This core component drives the browser and interacts with web elements, offering a robust means of automating browser actions. For the Adidas website, WebDriver can simulate user interactions like searching for products, adding items to the cart, and completing purchases.
2. **Selenium IDE:** This integrated development environment provides a record-and-playback tool for creating test scripts without programming knowledge. It allows quick creation of tests to validate Adidas's user interface and functionality.
3. **Selenium Grid:** This component enables running tests in parallel across different machines and browsers. For Adidas, it helps in testing across various devices and browser versions, ensuring consistent performance and functionality.

Key Features of Selenium for Adidas Testing:

- **Cross-Browser Testing:** Selenium can execute tests across multiple browsers, such as

Chrome, Firefox, Safari, and Internet Explorer. This ensures that the Adidas website behaves consistently regardless of the browser used by customers.

- **Cross-Platform Testing:** Selenium supports different operating systems, including Windows, macOS, and Linux. This feature allows testing of the Adidas website across various platforms to verify compatibility.
- **Support for Multiple Languages:** Test scripts for the Adidas website can be written in several programming languages, providing flexibility and ease of integration with existing projects.
- **Integration with Other Tools:** Selenium integrates with other testing frameworks and tools like TestNG, JUnit, and Maven, enhancing the overall testing process for Adidas.
- **Community Support:** As an open-source tool, Selenium benefits from a large and active community. This community contributes to its development and provides support through forums and online resources.

Selenium is a powerful and versatile automation testing tool that can significantly enhance testing efficiency for the Adidas website. It allows for comprehensive testing to ensure that the website functions correctly across different browsers and platforms, ultimately providing users with a seamless and satisfactory shopping experience.

2.2 Installation

1. Install Node.js:

- Download the latest version of Node.js from the official Node.js website.
- Verify the installation by running `node --version` in your command line.

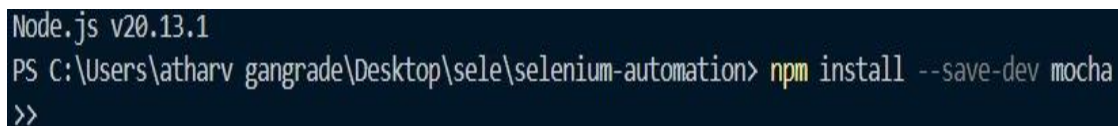


```
PS C:\Users\atharv gangrade\Desktop\sele\selenium-automation> node -v
v20.13.1
```

Figure 2.1 Checking Node version

2. Install Selenium:

- Open a command line and run `npm install selenium` to install the Selenium package.



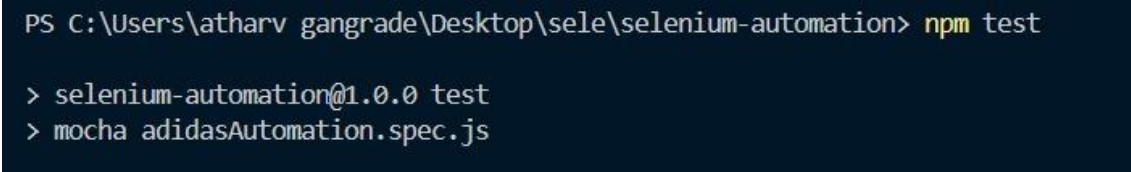
```
Node.js v20.13.1
PS C:\Users\atharv gangrade\Desktop\sele\selenium-automation> npm install --save-dev mocha
>>
```

Figure 2.2 Installation of Selenium

3. Install Mocha:

- In the same terminal, run:

npm install --save-dev



```
PS C:\Users\atharv gangrade\Desktop\sele\selenium-automation> npm test
> selenium-automation@1.0.0 test
> mocha adidasAutomation.spec.js
```

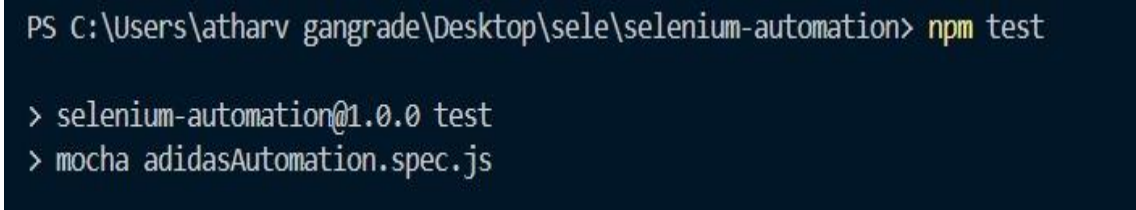
Figure 2.3 Installation of Mocha

4. Download WebDriver:

- Download the appropriate WebDriver executable (e.g., ChromeDriver for Google Chrome).
- Add the WebDriver executable path to the system PATH.

5. Install dependencies:

- dependencies is used to read/write Excel files. Install it by running: npm install



```
PS C:\Users\atharv gangrade\Desktop\sele\selenium-automation> npm test
> selenium-automation@1.0.0 test
> mocha adidasAutomation.spec.js
```

Figure 2.4 Installation of dependencies

2.3 Test Scenarios and Test Cases

In Selenium WebDriver, test scenarios and cases are scripted using WebDriver commands to simulate user interactions and verify outcomes with assertions, while parameterization facilitates testing across different inputs and scenarios.

1. Record and Playback:

- **Selenium IDE:** This tool offers a record and playback feature, enabling testers to

interact with the Adidas website while Selenium IDE captures these actions and generates the corresponding test script. For instance, testers can record actions such as searching for products, adding items to the cart, and completing a purchase. These recorded steps form the basis of test scenarios. Testers can then refine these scripts by adding assertions, parameterization, and other elements to create robust and comprehensive test cases.

2. Keyword-Driven Testing:

- **Custom Solutions and Frameworks:** Although Selenium WebDriver does not natively support keyword-driven testing, it can be implemented using frameworks like Robot Framework or custom setups. In this approach, test scenarios are constructed by defining keywords that represent specific actions or operations. For the Adidas website, keywords could include actions like "login," "search product," "filter results," or "checkout." Testers can create a library of these keywords and combine them to develop various test cases. This method offers flexibility and modularity, simplifying the maintenance and scalability of test scenarios.

3. Scripting:

- **Custom Scripting:** Selenium WebDriver allows testers to generate test scenarios and cases through custom scripts written in supported programming languages like Java, C#, Python, Ruby, or JavaScript. Testers can design complex test scenarios by creating functions, using control structures, and implementing logical operations. For the Adidas website, this could involve scripting tests to handle dynamic content, data-driven scenarios, or specific workflows such as applying filters, verifying product details, and validating checkout processes. Scripting provides full control and flexibility to manage dynamic elements, data manipulation, and advanced test operations.

4. Frameworks:

- **Integration with Testing Frameworks:** Selenium WebDriver can be integrated with various testing frameworks like TestNG, JUnit, and NUnit. These frameworks aid in creating and managing test cases, offering features such as annotations for organizing tests, parallel test execution, data-driven testing, and detailed test reporting. For the Adidas website, these frameworks enable structured and maintainable test scenarios, improving the efficiency of the testing process. They also support running tests in parallel across different browsers and platforms, ensuring comprehensive coverage and consistency.

By focusing on the scripting method, testers can write custom scripts to interact with the Adidas website, perform actions, and validate outcomes. This method is particularly useful for handling complex and dynamic scenarios, making it a powerful approach for testing an e-commerce platform like Adidas.

1. Test Case 1: Navigate to Adidas Homepage

- **Objective:** Ensure that the user can successfully navigate to the Adidas homepage.
- **Steps:**
 1. Launch the browser.
 2. Navigate to the URL: <https://www.adidas.co.in/>
 3. Verify the page title and URL to confirm successful navigation.

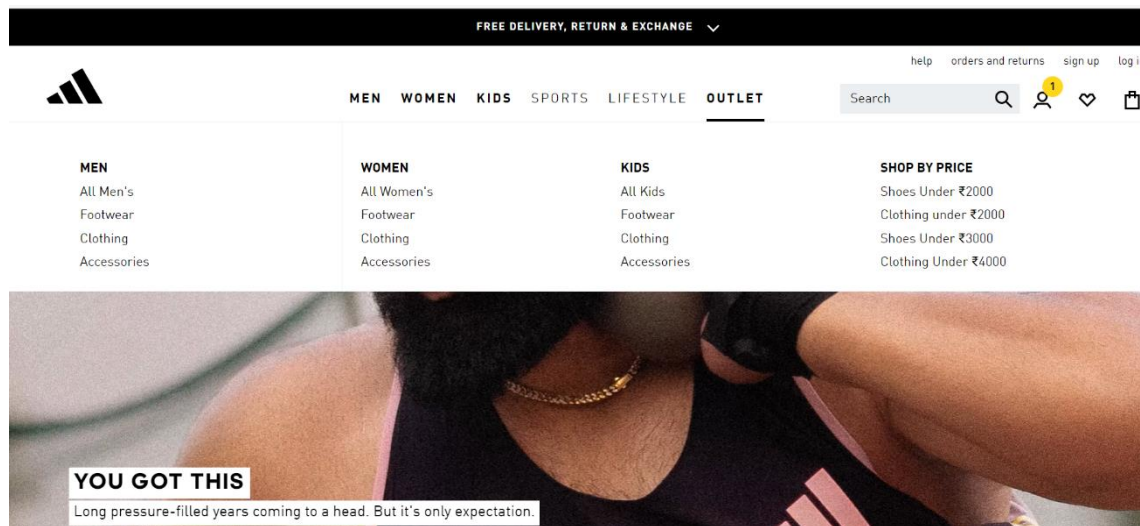


Figure 2.5 Verify Navigation to Adidas Homepage

2. Test Case 2: Interact with Category

- **Objective:** Ensure that the user can click on a category from the homepage.
- **Steps:**
 1. Navigate to the Adidas homepage.
 2. Locate and click on a specific category (e.g., Mobiles & Tablets).

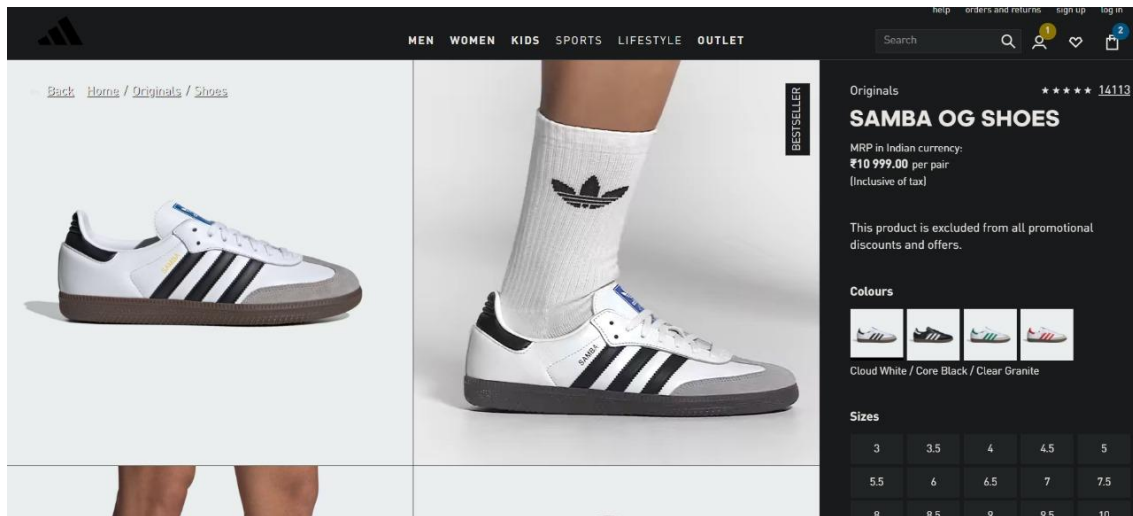


Figure 2.6 Verify Clicking on a Category

3. Test Case 3: Sort Products

- **Objective:** Ensure that the user can sort products by a specific criterion (e.g., price, popularity).
- **Steps:**
 1. Navigate to the Adidas homepage.
 2. Search for a product (e.g., "t-shirt").
 3. Apply a sorting filter (e.g., sort by price).

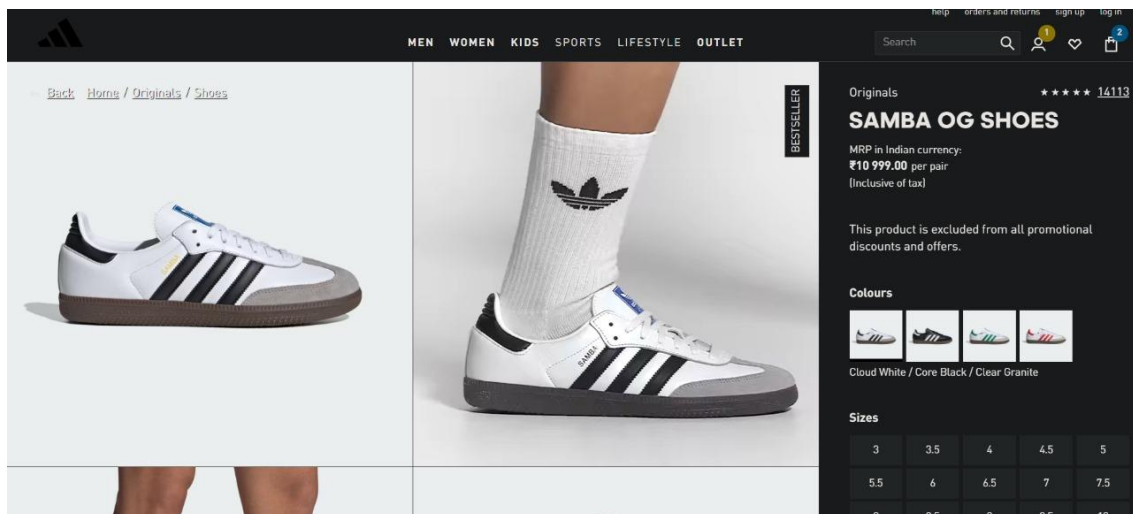


Figure 2.7 Verify Sorting of Products

4. Test Scenario 4: Search for Products

- **Objective:** Ensure that the user can search for products using the search bar.
- **Steps:**
 1. Navigate to the Adidas homepage.
 2. Enter a product name (e.g., "watch") in the search bar and press Enter.
 3. Verify the search results.

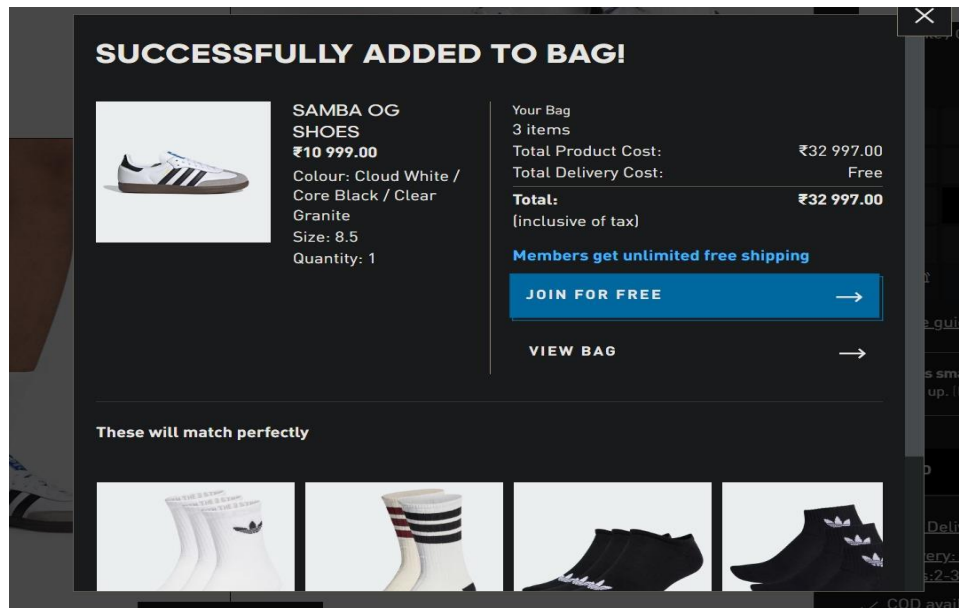


Figure 2.8 Verify Product Search

Test Report:

	A	B	C	D
1	Test Case	Status	Details	Timestamp
2	test_navigate	PASSED	Navigated to Adidas	2024-07-24 15:09:36
3	test_category	PASSED	Category clicked	2024-07-24 15:09:45
4	test_sortproduct	PASSED	Product sorted	2024-07-24 15:09:55
5	test_searchproduct	PASSED	Product searched	2024-07-24 15:10:13

Figure 2.9 The results are logged in an Excel file

2.4 Test Case Implementation

Here are the test cases implemented using Selenium and JavaScript, with results logged in an Excel file:

```
const { Builder, By, Key, until } = require('selenium-webdriver');
const assert = require('assert');
```

```
const { describe, it, beforeEach, afterEach } = require('mocha');

describe('adidas_automation', function() {
  this.timeout(60000);
  let driver;
  let vars;

  beforeEach(async function() {
    driver = await new Builder().forBrowser('chrome').build();
    vars = { };
  });

  afterEach(async function() {
    await driver.quit();
  });

  async function performSearch(searchTerm) {
    await driver.findElement(By.css("\input_1f3oz_13")).click();
    await driver.findElement(By.css("\input_1f3oz_13")).sendKeys(searchTerm);
    await driver.findElement(By.css("\input_1f3oz_13")).sendKeys(Key.ENTER);
  }

  async function applyFilter() {
    await driver.findElement(By.css(".gl-cta--secondary > .gl-icon__wrapper")).click();
    await driver.findElement(By.css(".gl-cta--secondary > .gl-cta__content")).click();
    await driver.findElement(By.css(".sorting_option___1AgyS:nth-child(1)")).click();
    await driver.findElement(By.css(".sidebar_filter___36lIx:nth-child(3) .gl-label--1")).click();
    await driver.findElement(By.css(".open___3TMo3:nth-child(3) li:nth-child(4) .gl-checkbox")).click();
    const element = await driver.findElement(By.css(".gl-checkbox--checked"));
    await driver.actions({ bridge: true }).moveToElement(element).clickAndHold().perform();
    const filterElement = await driver.findElement(By.name("Sweat Wicking"));
    await driver.actions({ bridge: true }).moveToElement(filterElement).release().perform();
  }

  async function selectProduct() {
    await driver.executeScript("window.scrollTo(0,216)");
```

```
await driver.findElement(By.css(".grid-item:nth-child(2) .product-card-image:nth-child(2)").click());
const noChatButton = await driver.findElement(By.css(".no-chat-button"));
await driver.actions({ bridge: true }).doubleClick(noChatButton).perform();
await driver.executeScript("window.scrollTo(0,0)");
const labelElement = await driver.findElement(By.css(".gl-label:nth-child(9)"));
await driver.actions({ bridge: true }).moveToElement(labelElement).perform();
await driver.findElement(By.css(".gl-label:nth-child(13)").click());
}

async function addToCart() {
  await driver.executeScript("window.scrollTo(0,382.3999938964844)");
  await driver.executeScript("window.scrollTo(0,482.3999938964844)");
  await driver.findElement(By.css("#add-to-bag .gl-cta__content")).click();
  const overlayElement = await driver.findElement(By.css(".gl-modal__overlay"));
  await driver.actions({ bridge: true }).moveToElement(overlayElement).perform();
  await driver.findElement(By.css(".gl-cta:nth-child(10) > .gl-cta__content")).click();
  await driver.executeScript("window.scrollTo(0,47.20000076293945)");
  await driver.executeScript("window.scrollTo(0,446.3999938964844)");
  await driver.executeScript("window.scrollTo(0,887.2000122070312)");
  await driver.executeScript("window.scrollTo(0,916)");
  await driver.executeScript("window.scrollTo(0,0)");
}

it('adidas_filter', async function() {
  await driver.get("https://www.adidas.co.in/");
  await driver.manage().window().setRect({ width: 811, height: 816 });
  await performSearch("t-shirt");
  await applyFilter();
});

it('adidas_search', async function() {
  await driver.get("https://www.adidas.co.in/");
  await driver.manage().window().setRect({ width: 1552, height: 832 });
  await performSearch("shoes");
  await selectProduct();
  await addToCart();
});
});
```

CONCLUSION

In conclusion, testing is pivotal to maintaining the quality, reliability, and performance of the Adidas website. Both manual and automation testing play critical roles in a thorough testing strategy. Manual testing offers a human-centric approach, allowing testers to utilize their domain knowledge, intuition, and exploratory skills to identify defects and usability issues that automated tests might overlook. This type of testing is especially valuable in the early stages of development, during user acceptance testing, and for evaluating the overall user experience. Manual testing provides insight into how the website performs from an end-user perspective, assesses the user interface, and facilitates ad-hoc testing scenarios to capture nuances that automated tests may miss.

Conversely, automation testing brings substantial benefits in terms of efficiency, repeatability, and scalability. For the Adidas website, automation testing is particularly effective for executing repetitive test cases, performing regression testing, and conducting load testing. Automation tools can rapidly execute a multitude of tests, reducing human error and providing swift feedback. This approach enables parallel testing across different platforms and configurations, which is crucial for verifying the website's performance across various devices and browsers. By automating routine test cases, teams can concentrate on more complex testing areas such as performance, security, and API testing. Moreover, automation supports continuous integration and delivery practices, facilitating rapid and reliable software updates.

Manual and automation testing are complementary approaches that together ensure a comprehensive and effective testing strategy for the Adidas website. While manual testing adds a layer of human insight and flexibility, automation testing delivers speed, consistency, and extensive coverage. Leveraging both methods based on project needs enables Adidas to achieve thorough test coverage and deliver a high-quality user experience, meeting and exceeding customer expectations.

REFERENCES

- [1] **Selenium Official Documentation:** <https://www.selenium.dev/documentation/>
- [2] **Nodejs Official Website:** <https://www.nodejs.org/>
- [3] **Visual Studio Code:** <https://www.codecademy.com/article/visual-studio-code>
- [4] **ChromeDriver:** <https://sites.google.com/chromium.org/driver/>
- [5] **Selenium :** <https://www.javatpoint.com/selenium-tutorial>
- [6] **Pytest:** <https://www.tutorialspoint.com/pytest/index.html>

