TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Kush Hiren Brahmbhatt
212284IASM

Comparative analysis of selecting a test automation framework for an e-commerce website

Master's thesis

Supervisor: Jekaterina Tšukrejeva MSc

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kush Hiren Brahmbhatt
212284IASM

E-kaubanduse veebisaidi testimise automatiseerimise raamistiku valimise võrdlev analüüs

Juhendaja: Jekaterina Tšukrejeva
MSc

Tallinn 2023

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Kush Hiren Brahmbhatt

08.05.2023

Abstract

Ecommerce websites are growing and there is a need for test automation to mitigate human error caused of manual testing and to provide a seamless experience to their customers shopping online. Playwright is an easy to use open-source test automation framework which will be competitor of the popular open-source framework Selenium Webdriver. This master thesis aims to perform comparative analysis between these two frameworks to make an informed decision when choosing the test automation framework for the ecommerce website and performing end-to-end testing on popular fitness clothing brand Gymshark EU.

After conducting end to end testing. The results suggest that for Gymshark EU website, Selenium Webdriver is 2 times faster than Playwright. The Playwright framework is easy to use, documentation is provided on their website. Scripting in Java using Selenium Webdriver is tedious and has a lot more code to write than Playwright. Playwright locator API and data locator id is superior has an edge over Selenium's find by element method when handing dynamic elements and floating buttons. Playwright has in-built recording libraries taking less hard-disk space then third-party supported recording in Selenium Webdriver. Playwright is a viable option to consider and further improvements in Playwright would improve its execution speed and computational time. Selenium Webdriver has its robust Page Object Model that gives an edge over its completion.

This thesis is written in English and is 51 pages long, including 5 chapters, 10 figures and 5 tables.

Annotatsioon

E-kaubanduse veebisaidi testimise automatiseerimise raamistiku valimise võrdlev analüüs

Poodide veebisaidid kasvavad ja on vaja testimise automatiseerimist, et leevendada käsitsi testimisest põhjustatud inimvigu ja pakkuda klientidele veebis ostlemist sujuvalt. Playwright on hõlpsasti kasutatav avatud lähtekoodiga testimise automatiseerimise raamistik, mis on populaarse avatud lähtekoodiga raamistiku Selenium Webdriver konkurent. Selle magistritöö eesmärk on teha nende kahe raamistiku võrdlev analüüs, et teha teadlik otsus poodide veebisaidi testimise automatiseerimise raamistiku valimisel ja populaarse fitnessrõivabrändi Gymshark EU täieliku testimise läbiviimisel.

Pärast otsast lõpuni testimist. Tulemused näitavad, et Gymsharki EL-i veebisaidil on Selenium Webdriver 2 korda kiirem kui Playwright. Playwrighti raamistikku on lihtne kasutada, dokumentatsioon on nende veebisaidil. Java-skriptimine Selenium Webdriveri abil on tüütu ja selle kirjutamiseks on palju rohkem koodi kui Playwrightil. Dramaturgi asukoha API ja andmelokaatori ID on parem, omavad dünaamiliste elementide ja hõljuvate nuppude üleandmisel eelist Seleeni elementide järgi leidmise meetodi ees. Playwrightil on sisseehitatud salvestusteegid, mis võtavad vähem kõvakettaruumi kui kolmanda osapoole toetatud salvestus Selenium Webdriveris. Playwright on mõistlik võimalus kaaluda ja Playwrighti edasised täiustused parandaksid selle täitmiskiirust ja arvutusaega. Selenium Webdriveril on tugev leheobjekti mudel, mis annab eelise selle valmimise ees.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti
51 leheküljel, 5 peatükki, 10 joonist, 5 tabelit.

List of abbreviations and terms

DPI	Dots per inch
IA	Department of Computer Systems
WWW	World Wide Web
Ecommerce	Electric commerce
SDLC	Software Development Life Cycle
STLC	Software Test Life Cycle
UI	User Interface
QA	Quality Assurance
AUT	Application Under Test
Web App	Web Application
IT	Information Technology
EU	European Union
MB	Megabytes
POM	Page Object Model

Table of contents

List of figures

List of tables

1 Introduction

The global number of websites on the internet has more than doubled from 2015 to 2022. Currently, there are over 1.98 billion active websites on the World Wide Web (WWW), and this number is forecast to increase exponentially in the upcoming years [1]. One of the internet's largest markets, eCommerce is a hugely valuable sector. The eCommerce income projection indicates that in 2024, it will increase to 3.1 billion US dollars [2]. Moreover, as the eCommerce sector keeps on growing, the number of eCommerce websites increases. As the demand for creating an online website increase, focus and emphasis on testing the software becomes critical.

Testing is a vital part of software development life cycle (SDLC) and it is the most time-consuming phase of the life cycle model, consuming approximately 40% to 70% of time in the development phase [3]. Software testing contributes to the improvement and maintenance of software quality, and hence plays a significant role in the software development process. The earlier a failure or bug is detected, the lower the cost of correction and easier it is to update the software [4]. There are various techniques for software testing. Most popular traditional software testing approaches include automated tests in the form of unit level, then followed by integration, and user interface (UI) tests. These tests are performed manually, hence called manual testing. The selection of test cases to evaluate each unit under testing is an important aspect of the testing activity. Such test cases can be developed manually by the tester or automatically via test generators.

An automated testing technique attempts to reduce the process's tedium by utilizing a software tool that creates test cases from the program's specification (black box) or actual text (white box). Automated and manual procedures are frequently regarded to be unique, and they are typically backed by different technologies [5].

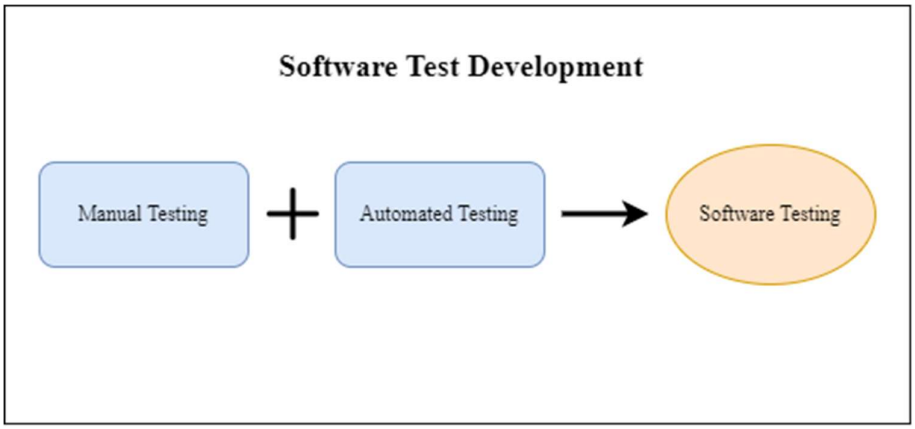

Figure 1. Illustration of software testing

As illustrated in Figure 1, Test development process consists of two main parts. The first half is manual testing performed manually by testers or quality assurance (QA) personal and the other half is automated testing also known as test automation, which is performed by test engineers and test software developers or QA specialized in scripting test automation software.

In recent times, Test automation has helped automating major parts of the testing phase thus reducing the human-error in testing. Test automation frameworks for web applications like Selenium are popular amongst web developers providing a wide range of cross-platform functionalities. Selenium is open-sourced, and it is comprised of two main parts: Selenium IDE (Integrated Development Environment) and Selenium Webdriver. First, Selenium IDE is a Firefox and Chrome plugin feature where any interactions of the browser can be recorded during a bug generation and verification process with the help of assertions that can be used during the test automation suite. Lastly, Selenium Webdriver is a popular web testing framework used in creating and executing test cases in a wide variety of programming languages such as JavaScript (Node.js), Python, Ruby, Java, C# and many more. It is robust, user friendly and a de facto choice of developers and test engineers. Recently, Selenium has launched Selenium Grid that allows Selenium WebDriver scripts to run and test in parallel across multiple remote machines [6].

Microsoft launched Playwright, a new end-to-end testing framework which is open-sourced, and which is cross-platform, cross-language, and cross-browser. The upcoming framework is resilient, provides power tooling and as claimed a faster execution [7].

The aim of the thesis is to provide a detailed overview of major test automation frameworks available in the current software market and perform comparative analysis of those frameworks and run some sample unit tests to help individuals and companies in making an informed decision when selecting an automation framework for their eCommerce web applications.

1.1 Background

Testing automation has been around for the last three to four decades. Predominately, Manual testing was performed during the first couple of decades then in the early 2000s the arrival of test automation tools and frameworks facilitated the growing needs for internet and the world wide web (WWW). Until recent times, the majority of developers preferred to use the tools and frameworks they were comfortable with but that has changed with the arrival of new frameworks and technologies. New open-source platforms like Playwright have entered the web automation industry and are being considered a viable option and potential replacement to the old frameworks and tools developed in the early 2000s.

In the mid-2010s, several new test automation frameworks entered the market and are on a constant rise because of their free to use, reliable and cross-platform features. The de-facto choice for developers and test engineers remains to Selenium frameworks with their constant focus to improve and develop it features and offering to the wider range of user but there is a slow and steady raise with regards to the use-case and popularity of certain frameworks.

The growing eCommerce market is searching for alternatives to improve their website visibility, performance, and user experience.

1.2 Problem

A well thought-out plan for test automation process might not work out owing to a wrong selection made during initial stages of choosing a test automation framework or tool. The selection of a test framework or tool is critical to the testing process and should be done with care.

Before planning, development teams must conduct research on the tools available and assess the available budget. Choosing a popular commercial tool simply because others in the business are does not always work out. A successful evaluation entails identifying tool requirements based on application under test (AUT) testimonies from professionals who have used the tools previously.

No matter how good the testing strategy and methodologies are, they would fail if the tool used to execute them does not meet technical and business criteria. Any automation endeavor is doomed to fail in the absence of the proper tool.

Nowadays, due to enormous growth in the demand of online web applications and software products, software teams are under pressure to produce web apps and testers to run tests quicker. Testers are required to run more tests on websites under various conditions in less time thus creating a constraint. As new versions of software are published, new features must be manually tested, and previous versions must also be evaluated under regression testing [8]. If the same tests must be run repeatedly, automation is preferable than manual execution and thus requires automation testing tool that can be perform parallel testing to save execution time.

There are various important issues in the domain of test automation that must be addressed:

- Making an informed selection and choosing the most suitable test automation framework which is compatible for the project.
- After selecting test automation framework, choosing the appropriate stack of tools and compatible programming language would improve the overall efficiency.
- Automating the test cases and minimizing the use of manual testing.

The eCommerce website that I have selected for my master's thesis is one of the fastest growing gym clothing brands in Europe and it is called Gymshark Europe [9]. As the eCommerce industry is growing rapidly, I was searching for a website which I personally use and frequently purchase online clothing from. The online traffic at Gymshark is amongst

the highest in recent times thus the website would be ideal for performing end to end testing and performance of the selected frameworks would be tested thoroughly. The impact of running automated testing in Selenium Webdriver and Playwright will be observed.

Like any ecommerce platform, the Gymshark EU website is susceptible to encountering various potential issues during the quality assurance phase. There are notable challenges observed from initially using their website. The website contains a lot of media and video elements. The website is updated constantly with new offers and discounts might contain some third-party links to sign up for discount. The overall user experience is improved but there are several test automations challenges and issues that would be addressed in this thesis work. These could include:

- **Functionality issues**: The present study highlights the significance of conducting comprehensive testing of the website's functionality, including the shopping cart, checkout process, and product pages, to ensure optimal performance and user satisfaction.
- **Compatibility issues**: The matter of compatibility is of utmost importance in website development. It is imperative to conduct thorough testing of the website on various browsers and devices to ascertain its seamless functionality across all platforms. This is crucial in ensuring that all users are able to access and utilize the website without any hindrances.
- **Performance issues:** The present study identifies performance issues pertaining to the website, which necessitate optimization measures for enhancing its loading speed and capacity to accommodate high user traffic.
- **Dynamic content:** The Gymshark website comes with a vast and ever-evolving product catalog, characterized by a dynamic nature that is typified by a continuous influx of new products and the continuous removal of outdated ones. The challenge of generating stable and dependable test cases that encompass the entire spectrum of product combinations and variations is a potential issue.
- **Complex user flows:** The Gymshark website exhibits a diverse range of user flows, encompassing activities such as product browsing, cart item addition, and checkout. The comprehensive testing of these various flows necessitates a substantial investment of time and resources and may entail the utilization of specialized software to emulate user actions.

1.3 Thesis Objective

The aim of the thesis is to perform comparative analysis between different test automation frameworks such as popular web-based framework Selenium, new and steady growing framework backed by Microsoft Playwright, in order to identify the best suited test automation framework for eCommerce website.

Many online technology platforms publish their opinions and articles about which automation framework to use when performing web testing, but many articles cover general aspects, but they leak technical viewpoints, lack of actual testing performed on a particular website and their results and data. My thesis aims to cover not only general aspects but to give detail overview by those frameworks, their advantages and disadvantages, their performance and use case on an eCommerce website as well as performing end-to-end testing in selected frameworks Selenium and Playwright to determine which one is better suited for Gymshark webpage and compare the testing process, execution, and ease to program.

The Playwright framework is a new automation framework as compared to other established frameworks. As it is a new framework, it is required to analyze and compare with the existing frameworks and document its competitive advantages and disadvantages. Moreover, the thesis aims to cover the Playwright automation framework and its functionalities. It also aims to highlight the significance of automated testing and manual testing and recommend the optimal tool stack for website testing based on comparative analysis and unit testing.

2 Testing of web applications

The realm of software development is constantly changing and evolving. Over the course of time, information technology (IT) companies and organizations focused on improving the user experience, hence improving their websites, and creating customer friendly user interfaces. With the increase in the demand for creating, managing, and updating web applications resulting in creation of quality assurance division within software department. QA teams are responsible for running the different tests on web applications ensuring the quality of the software product to satisfy the user requirements [10]. Testing a software product is essential and the process of testing is a vital part of SDLC. Testing process contains some fundamental processes of general development life cycle model of defining the scope, strategy, responsibilities, and planned activities.

2.1 Types of Testing

Testing is described as the process of determining whether a certain system satisfies its originally established criteria or requirements [11]. The testing process compasses of validation and verification. The aim of testing the software is to discover any unforeseen bugs, errors, or missing requirements in the developed software product. The general classification of testing based on business requirements is termed as functional testing and it involves testing the software application against the specified business requirements whereas non-functional testing involves those methods that focuses on operational aspects of software product such as security testing and compatibility testing.

With reference to SDLC, defining the scope is the most critical attribute of testing. The scope underlines the size of the tested units of application under test (AUT). Functional testing includes the three main level of testing methods (as shown in the figure 2) [12] as well as acceptance testing as the last layer of verification:

- **Unit testing:** The initial level of testing is unit testing, which is frequently handled by the developers themselves. It is the process of verifying that individual code components of a piece of software are functional and perform as intended. In a test-driven environment, developers often create and execute the tests before passing the program or feature to the testing team. Manual unit testing is possible but automating

the process will shorten delivery cycles and increase test coverage. Unit testing will also make debugging easier since issues detected earlier in the testing process require less time to fix than those discovered later in the testing process.

- **Integration testing:** The second level of testing following comprehensive unit testing, each unit is combined with other units to form modules that are meant to execute certain tasks or activities. These are then evaluated as a group using integration testing to ensure that whole application parts function as intended. Moreover, the interactions between units themselves is tested to make sure that interactions are flawless. User scenarios, such as login into an application or accessing files, are frequently used to structure these tests. Integrated tests are often performed by either developers or independent testers and consist of a combination of automated functional and manual tests.
- **System testing:** The third and final level testing, which is performed as software application. Numerous units combined to form a component; multiple components combined to form a system (i.e., software web application). System testing validates the fully integrated software web application [13].
- **Acceptance testing:** It is regarded as the last phase for functional verification. This layer of testing validates how the system behaves in a production environment, whether this works as intended or not. The system's end users carry out testing [14].

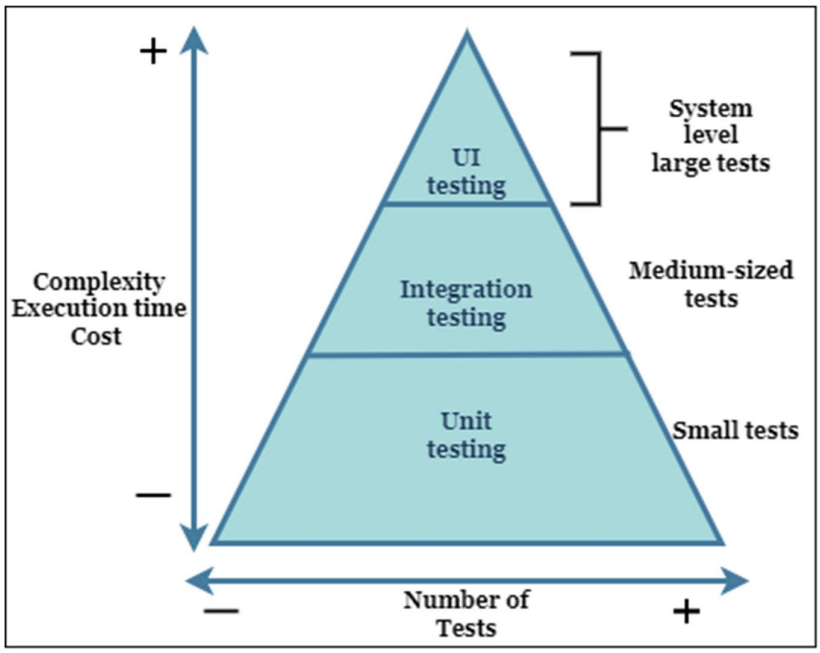

Figure 2. Test automation Pyramid

Figure 2, The testing pyramid is multi-layer test levels. The testing pyramid is an integral part of any software testing approach. Thorough testing is required before releasing an app into the market and a comprehensive test automation plan is required to effectively test an app [15]. Test levels are abstractions that are used to manage time and resources between different stages of software development. Test level definitions vary by company, but test levels commonly relate to three separate software testing tiers [16].

2.2 Testing Techniques and Strategies

Testing is an important part of the software development cycle thus creating a reliable software product requires high level of planning and follows a stepwise structured approach for the development process. Establishing a method of testing or strategy assists in achieving clarity on goals and objectives from the necessary stakeholders, hence benefiting in managing expectations. Selecting an appropriate technique facilitates distinguishing the suitable methodology of approach.

The primary goal of any testing approach is to ensure that the program is bug-free. As a result, it seeks to detect problems in a specific section of the software. Software testing is more dependent on what you want to test in software and how you want to test it; also, this helps to identify the tools needed to complete the test. A test strategy is a brief statement that specifies how the software testing objectives are satisfied. It takes a high-level picture of the test event and focuses on its objectives. It is critical that they prepare their testing methodology at each step of the project and build a framework for testing the project. A testing approach is a method of ensuring that the application being tested operates in a systematic manner. [17].

The three main techniques of testing are as mentioned below [17]:

- Manual or Automated Testing
- Structural or functional testing is also known as **White Box** and **Black Box** testing.
- Static or dynamic testing

For the scope of this master thesis, the main emphasis is on manual and automated testing. Furthermore, structural and functional testing concepts such as black box testing, and white box testing are important and been used in web automation testing. Static and dynamic testing

are out of scope.

2.2.1 Manual and Automated testing

A person or a machine can perform testing. Manual testing occurs when a tester manually conducts a specific set of tests. We refer to Automated testing as the execution of the same set of tests by a machine. In essence, testing is vital for the development process and test automation cannot replace manual testing, but it can allow the tester to perform repetitive testing over a longer period. Furthermore, certain tests are time-consuming to run manually; for example, if you want to automate thousands of operations in a given time frame, it is often impossible to do so by hand.

Manual testing comprises manual tasks such as setting up a test environment, executing test tasks, reporting errors discovered, and reviewing the findings. Manual unit testing has become an essential aspect of modern software development. Manual unit testing frameworks automate the execution of test cases. Hand-written test cases (including input data creation and test result verification) are required. This procedure may be carried out using either a test plan or exploratory testing [18]. It should have a test plan identifier, an introduction, test items, the features to be tested or not, a strategy, item pass/fail criteria, stakeholder information, testing communication, and a timeline, according to IEEE 29119-3 [19].

Automated testing, as opposed to manual testing, automates not just test case execution but also test case generation and test result verification. A completely automated testing system can run software without any user participation. Users do not have control over which test cases are executed when utilizing fully automated testing solutions. Instead, they often supply a test scope: a list of classes to be tested; in other words, they only need to indicate what to test, not how.

Automated testing is the execution of tests without the intervention of a person. However, other prerequisites for automated test execution exist [20]:

- The ability to perform a subset of all tests.
- Automatically configure and record environmental variables.
- Execute the test cases.
- Documenting the outcomes.

- Analyzing actual and predicted results and processing them comprehensively.

Both forms of testing have their pros and cons. It highly depends on the web application under test enables user to choose one testing method over another. Some constraints such as development and testing budget, the defined scope of testing, deadline to release the project are considered when deciding which testing method to select. As the name suggests, automated testing is automated and independent hence faster than the standard manual testing which is time-consuming in nature and is prone to human error. On the budget side of things, manual testing is affordable and easy to follow [21].

Moreover, a tester can arrange the tests to run independently using end-to-end test automation. While tests are running in background, testers may use the extra time to complete any necessary manual testing or create more automated test cases to expand test coverage. Since automated testing allows for concurrent/parallel testing, it saves valuable time. A tester can perform several tests at the same time, but manual tests can only be conducted sequentially [22].

Automation testing offers an advantage over manual testing since manual testing is meticulously prepared and documented, whereas automated testing delivers critical insights such as test execution, length, and failed test cases. Furthermore, automation testing aids in testing large sequences of data and aids in randomizing the search for errors in software, thereby assisting web/based systems in performance reliability testing [17].

To summarize, with a test automation framework and automated testing, software testers can focus on testing the software product (i.e., the web application) instead of worrying about developing the infrastructure needed to support their test environment [21]. During the initial stages of the project, it is recommended to use manual testing and understand its importance in the starting phase of SDLC [23].

To further elaborate on the differences and advantages of manual and automated testing, table comparison is performed refer to the Table 1. The automated testing is reliable and efficient and requires less human attention and the pro of manual testing is manual QA can

give personalized feedback as compared to automation hard type comments and feedback. Furthermore, the cons of manual testing are further described in the Table 1.

Table 1 Comparison between Manual and Automated testing

Manual Testing	**Automated Testing**
Test performed by humans.	Software tools execute all the tests.
Performing manual testing is time-consuming and tedious.	Automated testing is a fast and efficient process. These tests are faster than manual testing.
Prone to human error thus less reliable.	A higher degree of reliability as tests are performed by software program.
Human resources are required.	Software tools and programs are required.
Manual testing is feasible when the test is performed just once or twice during development and testing process.	Automation testing is useful when a repetitive test execution is required over a lengthy period of time.
Manual testing allows for human observation and exploratory testing, which aids in the detection of UX and usability concerns.	Doesn’t allow human observation hence there is less chance of detecting UX and usability concerns.

2.2.2 Black Box and White Box Testing

Structural and functional testing methodologies are also referred to as white box and black box testing techniques, respectively. These two categories of testing methodologies are critical in software testing. **Black box testing** is more commonly referred to as behavioral testing. During this testing approach, the software tester runs a series of tests to see if the application under test (AUT) meets the specifications and requirements [17].

It is referred as a black box because the software tester performs specified tests without any prior knowledge of the software's internal workings; the tester's sole goal is to observe the required outputs in response to the supplied inputs and circumstances. One advantage of

black box testing is the ease with which test cases can be written from the perspective of an end user without understanding of core program logic. As part of this testing technique, it is vital to use test monitoring tools to track the executed tests and avoid repetition.

White box testing is better known as the glass box testing technique. The software tester concentrates on the structure and core of the AUT software program in this testing approach. The test developer is tasked with creating test cases that validate the software program's underlying logic [17]. This approach assists the testing team in determining the efficiency of the software code and its extra activities. One of the key advantages of white box testing is that the tests are performed at the code level, which allows for additional code optimization as well as the detection and removal of unnecessary code or hidden bugs.

Additionally, a new semi-transparent approach of testing is getting popular amongst the developers, QA specialists and software testers in recent times. This testing method is a combination of the black box and white box testing, known as the semi-transparent testing, also known as Grey-box testing. In this testing approach it entails having knowledge of the internal structures of AUT software for creating test design but testing at the block-box level. E.g., Testing the functionality of an online application requires knowledge and an idea of the webpage structure but its functionality is not important to know.

2.3 Phases of Software Test Life Cycle (STLC)

Software testing and STLC is the most important and fundamental part of SDLC. The process of STLC consists of QA specialists, Software testers including manual and automation testing and constant communication with product owner and stakeholders. The step-by-step process of AUT makes sure the quality of software applications. The following are the key phases of STLC and visual illustration of the process of test development refer to Figure 3, each phase of STLC showed in relation with the other, the perquisite or the initial stage where the process of STLC starts is when user requirements are analyzed and the last phase where development ends is shown and called as test closure:

- **Requirement Analysis**: In the first stage of the STLC, The QA team gets acquainted about the requirements, such as what will be tested, throughout this phase. The quality assurance team interacts with the stakeholders to better grasp the specific

knowledge of requirements if anything is missing or not clear.

- **Test Planning:** The most vital step of the software testing life cycle in which all testing strategies are developed is test planning. During this phase, the testing manager and team determine the projected effort and cost for the testing task. When the requirement-gathering phase is finished, this phase begins.
- **Test Case Development:** Upon completion of the test planning phase, the test case development phase is initiated. During this phase, the testing team documents comprehensive test cases. The team responsible for conducting the testing also undertakes the task of generating the necessary test data for the testing process. Upon completion of test case preparation, the QA team conducts a review.
- **Test Environment Setup:** The establishment of a test environment is a crucial component of the STLC. The test environment is a critical determinant of the testing conditions for software. The present activity is deemed independent and may be initiated concomitantly with the development of test cases. The involvement of the testing team is absent in this particular process. The responsibility of creating the testing environment lies with either the developer or the customer.
- **Test Execution:** The phase of test execution commences subsequent to the development of test cases and the establishment of the test environment. During this phase, the testing team proceeds with the execution of test cases that were previously prepared in the preceding step. In the phase of test execution, the software application is subjected to test cases and scripts that were previously designed to detect any potential defects or issues.

The following are some of the key activities taking place during the process of executing tests:

- Initial test execution
- Defect logging
- Test data preparation
- Test environment setup
- Running test cases scripts
- Test result analysis

- Defect retesting
- Test reporting

- **Test closure:** the conclusive phase of the STLC is known as Test Closure, during which all testing-related tasks are finalized and recorded. The primary aim of the test closure phase is to verify the completion of all testing-related tasks and to ascertain the software's readiness for deployment. This phase also includes documentation and report creation such as test summary and test closure reports.

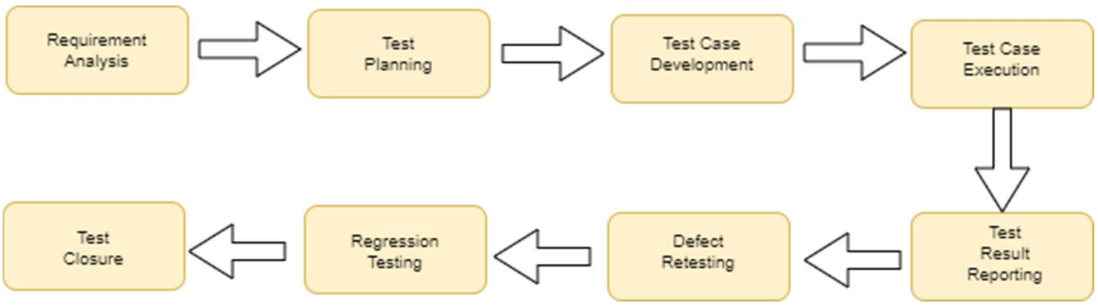

Figure 3 Phases of Software Test Life Cycle (STLC)

2.4 Importance of Test Automation Approach

The utilization of test automation extends beyond the mere execution of test cases. The author of the article [1] mentions that to optimize productivity and efficacy, test engineers must possess knowledge of diverse automated testing methodologies and tools that facilitate testing operations beyond mere test execution.

Nevertheless, the implementation of automation necessitates a judicious approach, as it incurs a cost. The formulation of a test automation strategy comes from the consideration of the consequences of test suite implementation, specifically regarding test maintenance. The significance of continuous integration in agile software development cannot be overstated. It is imperative for software quality assurance professionals to contemplate the integration of automated testing throughout the development, deployment, and delivery phases, extending beyond the mere execution of tests. The optimal utilization of automation in software engineering is contingent upon the familiarity of software engineers with the various test automation strategies and tools at their disposal.

3 Evaluation of testing tools and frameworks

As mentioned, above section 2.2 and 2.3, Choosing the correct testing strategies, tools and framework is the most crucial part of test development. The selection should consider various factors such as client requirements, website compactivity, cost, execution, time duration and so on. In the growing world of start-ups and small-scale business, there are some financial constraints. New and upcoming e-Commerce websites have limited budget and they constantly come up with new ideas and discount campaigns to attract their customers thus hiring third parties or software contractors for regularly conducting web testing is not a solution.

To overcome this challenge, test automation is required which provides a reliable and cost-effective solution to this problem. The present study aimed to automate an e-commerce web application for a startup company, with a focus on open-source tools that are deemed appropriate for enterprise-level deployment. The primary objective is to implement automation of the checkout process across various web browsers, with a particular emphasis on desktop browsers such as Google Chrome and Chromium. The provision of major operating systems, including Linux, Windows, and Mac is important. The present study aims to perform end-to-end automation testing on e-Commerce website based on customer journey from homepage to checkout page by two different selected frameworks and comparing their performance and results.

3.1 Criteria for choosing the test automation frameworks.

The assessment of automation tools can be conducted through the examination of various characteristics or parameters. Therefore, it is imperative to identify the distinct features for the purpose of conducting a comparative analysis.

The present study employed a set of criteria for conducting analysis, derived from a collection of characteristics deemed significant by professionals in the domain of test tool selection. The present study outlines a set of criteria that are deemed essential for the evaluation of a given software system. These criteria include applicability, compatibility,

configurability, cost-effectiveness, cross-platform support, ease of use, expandability, further development, maintenance of test cases and data, performance, popularity, programming skills, and reporting features. The study posits that these criteria are crucial for the effective assessment of software systems and can provide valuable insights into their overall quality and suitability for a given task.

Now, to meet the goal of this study i.e., comparison of software testing tools; features/characteristics are shown in a table below [24, 25, 26, 27] :

Table 2 Comparison of software testing tools [24, 25, 26, 27]

Number	**Parameters / Characteristics**	**Meaning**
1	Cost	Open-sourced (free to use) or license
2	Operating System (OS) compatibility	Cross-OS support
3	Brower compatibility	Cross-browser support
4	Extendibility	The degree to which software can be extend, add functionality
5	Record playback	Ability of tools to record scripts
6	Programming language support	Programming languages used to edit test scripts or for the creation of testing scripts.
7	Ease of learning	How easy the framework/tool is to learn and use
8	Data driven functionalities	The ability of tool to reduce efforts like making it possible to make the scripts access the different sets of input data from external source like data tables, excel sheets
9	Testing functionalities	Different type of testing supported
10	Modifiability	Quality of being modifiable according to user requirement.
11	Report generation	Ability to generate test results

3.2 Comparative analysis of test automation frameworks

Table 2 describes the key characteristics required for comparing software test tools and frameworks. On the basics of the above table 2, A comparative analysis is performed in which the popular frameworks, its tools and features are compared. Key parameters such as OS compatibility, ease of learning and programming language support and so on will analyzed and documented in the upcoming section. For the scope of this comparative analysis four popular frameworks are chosen. To further elaborate on each framework and its features a brief description is provided below:

3.2.1 Cypress

Cypress is a widely adopted end-to-end testing framework utilized in web application development, prioritizing ease of use, expeditiousness, and dependability. The present software artifact is implemented using the JavaScript programming language and is constructed utilizing Mocha and Chai, two widely adopted testing frameworks in the JavaScript domain.

The Cypress framework offers a distinctive architecture that facilitates the composition of test suites that execute within the identical context as the target application under examination. The present architecture obviates the necessity of a distinct Selenium WebDriver and thereby enables expedited and more dependable test executions. Moreover, Cypress incorporates a robust test execution environment that provides functionalities such as automated test retries, parallelization, and optimized test sequencing.

The Cypress testing framework provides a user-friendly syntax that facilitates the creation of tests, incorporating selectors, assertions, and network requests as integral features. The platform offers a diverse array of beneficial application programming interfaces (APIs) that empower software developers to execute intricate testing procedures such as authentication, mocking, and stubbing.

The debugging capabilities of Cypress are a notable feature of the framework. The framework incorporates an integrated test runner, which enables software developers to

perform real-time debugging of tests, establish breakpoints, and execute code line-by-line. Furthermore, Cypress offers a robust extension of browser development tools that empowers developers to scrutinize the Document Object Model (DOM), troubleshoot network requests, and perform other related tasks.

Cypress provides integration support with popular CI tools, such as Jenkins, CircleCI and Travis CI. The framework offers a command-line interface that enables software developers to execute tests in headless mode or on remote machines, thus facilitating seamless integration with continuous integration and continuous delivery pipelines.

3.2.2 Nightwatch

Nightwatch is a software tool that has been developed as an open-source solution for the purpose of facilitating end-to-end testing of web applications. Its primary function is to automate the testing process, thereby reducing the need for manual intervention. The software artifact is implemented in the JavaScript programming language and leverages the robust Selenium WebDriver application programming interface (API) for the purpose of managing web browsers.

The fundamental characteristics of the framework encompass a user-friendly syntax for scripting tests, provision for diverse browser drivers, and an integrated assertion library for validating test outcomes. Nightwatch provides robust reporting functionalities and exhibits seamless integration with Continuous Integration (CI) utilities such as Jenkins, Travis CI, and CircleCI.

Nightwatch's simplicity is a notable advantage. The syntax of the framework is characterized by its simplicity and accessibility, rendering it easily comprehensible even for novice users. The Nightwatch framework employs a rudimentary test runner that is capable of executing tests concurrently, thereby diminishing the aggregate testing duration. Furthermore, the framework facilitates the utilization of page object models (POMs), a design pattern that empowers developers to systematize their test code into self-contained, interchangeable units.

The Nightwatch testing framework offers multi-browser support for a variety of widely used web browsers such as Chrome, Firefox, Safari, and Internet Explorer. Additionally, it enables the execution of tests across diverse operating systems, namely Windows, macOS, and Linux. The framework boasts comprehensive documentation and a thriving developer community that actively contributes plugins, extensions, and other tools to augment its functionality. The plugins offered by Nightwatch facilitate the customization of testing environments by developers, enabling the incorporation of supplementary features such as code coverage, performance testing, and screenshot capture.

3.2.3 Playwright

The Playwright framework is an open-source tool that has been specifically developed to automate the testing of web applications. The software in question was developed by Microsoft and is implemented in TypeScript, with the added capability of interoperability with other programming languages, including but not limited to JavaScript, Python, and Java.

Playwright's notable advantage lies in its provision of multi-browser support, encompassing Chrome, Firefox, Safari, and Edge. The aforementioned feature facilitates the composition of test suites by developers that are capable of being executed on a variety of web browsers and operating systems. Furthermore, Playwright presents a consolidated application programming interface (API) that enables software engineers to compose source code that functions seamlessly across various web browsers.

The Playwright API is a potent tool for engaging with web applications, encompassing navigation, input, and assertions functionalities. The framework incorporates a robust interception capability that empowers developers to alter network requests, append bespoke headers, or even emulate network errors, thereby facilitating the testing of diverse scenarios.

The support for headless testing is a notable characteristic of Playwright. The framework has the capability to execute tests in a headless mode, thereby enabling the execution of tests without the need to launch a browser window. The aforementioned attribute proves to be particularly advantageous in scenarios where tests are executed within a Continuous

Integration (CI) framework, wherein a visual interface is absent.

The Playwright toolset offers a comprehensive testing infrastructure that encompasses support for various testing frameworks such as Jest and Mocha. The framework exhibits seamless integration with widely used Continuous Integration (CI) tools such as Jenkins and Travis CI.

The Playwright framework boasts an integrated video recording functionality that can comprehensively capture the entire test execution process, encompassing screenshots, network traffic, and console logs. The streamlined process facilitates the expeditious review of test outcomes by developers, thereby enabling prompt identification of potential issues.

3.2.4 Selenium Webdriver

Selenium is a test automation framework designed for web applications that is open-source and cross-platform in nature. It was originally created by Jason Huggins in 2004 as a tool to automate tests for a web application. Subsequent to its inception, the aforementioned framework has garnered significant popularity and is presently among the most extensively employed test automation frameworks on a global scale and widely popular amongst senior developers.

Selenium allows QA engineers to automate websites using a wide variety of scripting languages that consists of Java, Ruby, C# and Python. It provides a suite of tools for automating web applications, including Selenium WebDriver, Selenium IDE, and Selenium Grid. One of the most popular and most widely used frameworks in the Selenium suite portfolio is Selenium Webdriver.

The Selenium WebDriver is a widely adopted test automation framework that is utilized for the purpose of testing web applications. The Selenium suite encompasses a tool that offers a programming interface to facilitate interactions with web browsers. With Selenium WebDriver, developers can write scripts that simulate user interactions with web pages, enabling them to automate web application testing.

The Selenium WebDriver tool facilitates the use of various programming languages such

as Java, Python, Ruby, C#, and JavaScript. The feature facilitates the composition of tests by software developers, who are able to utilize their favored programming language. Additionally, Selenium WebDriver can automate tests on multiple browsers, including Chrome, Firefox, Safari, and Edge, making it easier for developers to test their web applications on multiple platforms.

One of the key features of Selenium WebDriver is its ability to interact with elements on a web page. Developers can use WebDriver to find elements by their ID, name, class, or other attributes, and then perform actions on them, such as clicking a button, filling out a form, or verifying text.

The WebDriver software framework is capable of facilitating sophisticated user interactions, including but not limited to the manipulation of graphical user interface elements through drag and drop functionality, as well as the ability to hover the mouse pointer over specific elements.

Selenium WebDriver supports a range of test automation frameworks, including TestNG, JUnit, and NUnit. This makes it easy for developers to integrate their Selenium tests with their existing testing infrastructure.

Additionally, Selenium WebDriver can be integrated with Continuous Integration (CI) tools like Jenkins, allowing tests to be run automatically after every code change. Another advantage of Selenium WebDriver is its support for parallel testing. The concurrent execution of multiple tests by developers can expedite the testing procedure and furnish prompt feedback. The WebDriver tool possesses the capability to execute tests in a headless mode, whereby the tests are performed without the need to launch a browser window. This feature enhances the speed and efficiency of the tests.

Finally, Selenium WebDriver is highly extensible. Custom extensions or plugins can be developed by developers to augment the functionality of WebDriver. The extensibility of the framework enables developers to tailor it to their unique requirements and augment their tests with supplementary features and functionality.

Table 3 Comparison table of test automation frameworks

Parameters / Characteristics	**Cypress** [22]	**Nightwatch** [28]	**Playwright** [29]	**Selenium Webdriver** [6]
Cost	Open source, free to use.	Open source, free to use	Open source, free to use.	Open source, free to use
Operating System (OS) compatibility	Windows, Linux, and macOS 10.9 and above	Windows, Linux, and macOS	Windows, Linux, and macOS	Windows, Linux, and macOS
Brower compatibility	Chrome, Firefox, Edge	Chrome, Safari, Firefox, Edge	Chromium Safari, Firefox, Edge, Internet Explorer	Chrome, Safari, Firefox, Edge, Internet Explorer
Extendibility	Yes, extendible	Up to only certain extent	Up to only certain extent	Limited
Record playback	Available, In-built	Available with extensions	Available, In-built	Available with extensions
Programming language support	JavaScript	JavaScript, TypeScript	JavaScript, Python, C#, .NET, Java	JavaScript, Python, Ruby, Java, C#
Ease of learning	Yes	Yes	Yes	Partially
Data driven functionalities	Available	Basic functionalities	Available	Available
Testing functionalities	E2E and component testing	POM, Parallel testing	E2E testing and more	POM, E2E testing and more
Report generation	Mocha HTML report	HTML, Junit-XML, and JSON reports	HTML, Junit-XML, and JSON reports	TestNG, HTML, Junit-XML, and JSON reports

From the above table 3, all the above frameworks mentioned above, comparative, and personalized choice of framework can be made, Playwright provides the superior testing functionalities, supports most of the popular programming languages thus developers, IT students and tech enthusiast can easily program test automation programs in their preferred programming language. All four frameworks are open sourced, provide extensive support for report generation and provide options to generate innovative report results.

From the figure, a recently published survey ranked Playwright 8th most popular framework amongst JavaScript users. If we compare the year 2020 to the year 2021, awareness for Playwright software has grown from 19% to 34% [30]. Furthermore, Selenium Webdriver is the preferred by experienced QA engineers.

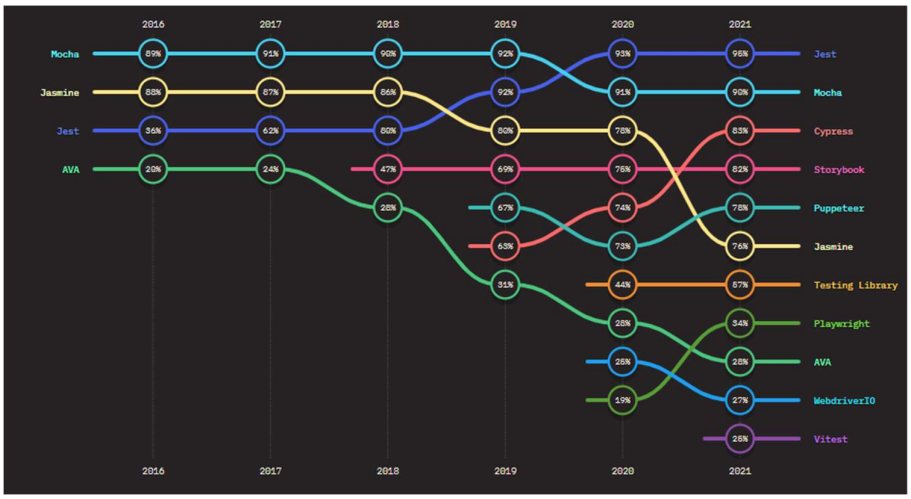

Figure 4 Popular framework amongst JavaScript users. *[30]*

Each test automation framework has its own advantages and disadvantages, Moreover, some individuals have their personal preferences since they have been using frameworks as Selenium for a long time, Hence, transiting to new and improved framework will take some time. To summarize and highlight the features of each test automation framework are listed below:

- Selenium WebDriver is a robust test automation framework that offers a programming interface for web browser interaction. Selenium WebDriver is a highly recommended tool for teams seeking to automate their web application testing due to its capacity to interact with web page elements, compatibility with various programming languages and browsers, seamless integration with popular testing frameworks and CI tools, as well as its support for parallel and headless testing.

- Playwright is a robust test automation framework that provides a consolidated API for engaging with various web browsers, potent interception functionalities, and backing for headless testing.

Playwright is a highly recommended option for teams seeking a streamlined and dependable approach to automating their web application testing, owing to its extensive testing infrastructure and seamless integration with widely used CI tools.

- Nightwatch is a potent and uncomplicated framework for testing web applications that provides a diverse array of functionalities and adaptability. The framework in question is a highly recommended option for teams seeking a user-friendly testing solution that can effectively expedite their testing procedures and enhance the caliber of their web-based applications.

- Cypress represents a contemporary and robust testing framework that presents a diverse array of functionalities aimed at streamlining the testing procedure and enhancing the dependability of web-based applications. Cypress is a highly recommended tool for teams seeking an effective and dependable approach to testing their web applications, owing to its user-friendly syntax, robust debugging functionalities, and integrated test runner.

3.3 Dependencies

To perform end-to-end testing in Selenium Webdriver and Playwright test automation framework. The following are the listed dependencies, tools and its version:

1. **Selenium Webdriver**

   Programming Language: Selenium Java

   List of tools and its versions used:

   - Selenium 3.141.59
   - Webdriver manager 3.8.1
   - TestNG 7.7.0
   - Maven 4.0.0
   - IDE: Eclipse

Framework:

- Pages package, Utilities package, Testcases package and Testdata package.
- pom.xml lists out all Maven dependencies required to execute the tests.
- testng.xml outlines the java classes that contain the test steps.
- Screenshots are saved in the 'Screenshots' folder during test execution.
- Third party screen recorder used for Video recording compactable with Selenium Webdriver.

2. **Playwright framework**

Programming Language: Playwright Javascript

Tools and its versions used:

- Node v16.18.0
- npm 8.19.2
- Playwright 1.33

Framework:

- 'playwright.config.js' file defines the test configurations.
- Tests are written within the 'tests' folder and are executed sequentially based on the Config.
- '@playwright/test' library is implemented to run these tests.
- Generated reports are saved in the 'playwright-report' folder.
- Screenshots captured during test execution are saved in the user-defined folders.
- Video recordings of test execution are stored in the 'test-recordings' folder.

The below figure 5 shows the basic structure and directory of Selenium Webdriver and Playwright. The main program of Selenium Webdrvier *MainTestCases* is in *testcases* folder and for Playwright main program *login.test.ts* is under *tests* folder. The folder contains config files, screenshots, and test results.

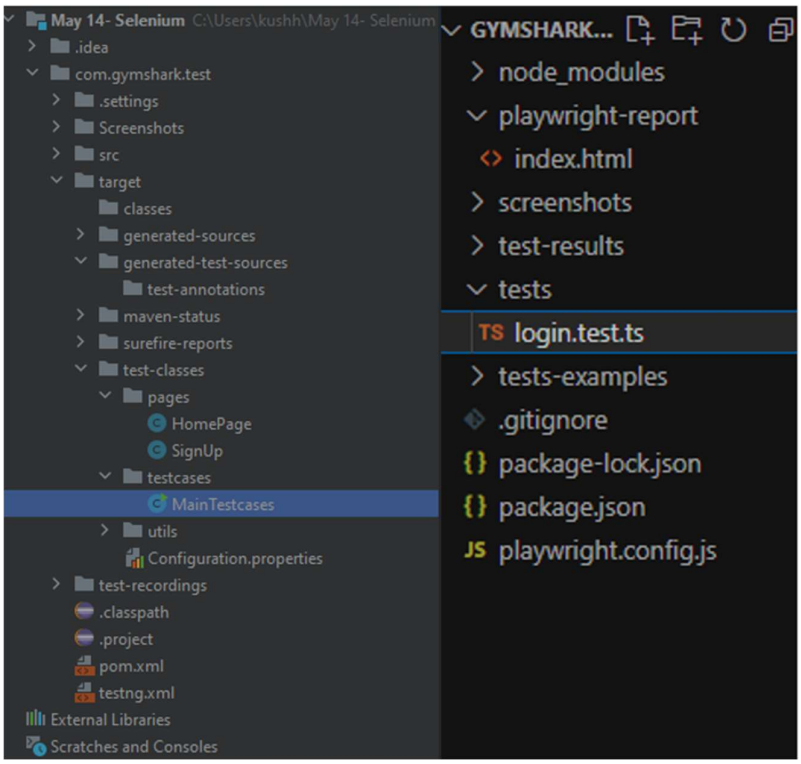

Figure 5 Selenium Webdriver (right) and Playwright folder structure (left)

4 Results

For the scope of this thesis, I have performed end-to-end testing on Gymshark EU website. All the required test cases are taken from Gymshark EU website, and it represents the European sector of Gymshark, a company specializing in fitness apparel and accessories that originated in the United Kingdom in 2012. The Gymshark EU website functions as an electronic commerce platform that provides a diverse selection of fitness apparel, accessories, and equipment tailored for both male and female consumers. The website of Gymshark EU exhibits a contemporary and uncluttered layout, characterized by a preponderance of black and white hues. The primary landing page of Gymshark's website exhibits conspicuous banner visuals that highlight the brand's most recent merchandise and marketing campaigns. The present study observes that the navigation menu located at the uppermost section of the webpage facilitates the customers to conveniently peruse diverse product categories, encompassing men's and women's clothing, accessories, and equipment.

The Gymshark EU website's product pages offer comprehensive details regarding each item, encompassing various product images, a meticulous product description, and sizing and fit information. The product's customer reviews and ratings are available for viewing by prospective customers, providing them with valuable insights from previous purchasers. Figure 6 showcase the landing page of Gymshark website also known as the homepage of the website. From the figure 6, on the top center, men, women and accessories sections are situated by hovering on top of each sections options popup as mentioned and listed under each section in the figure 7.

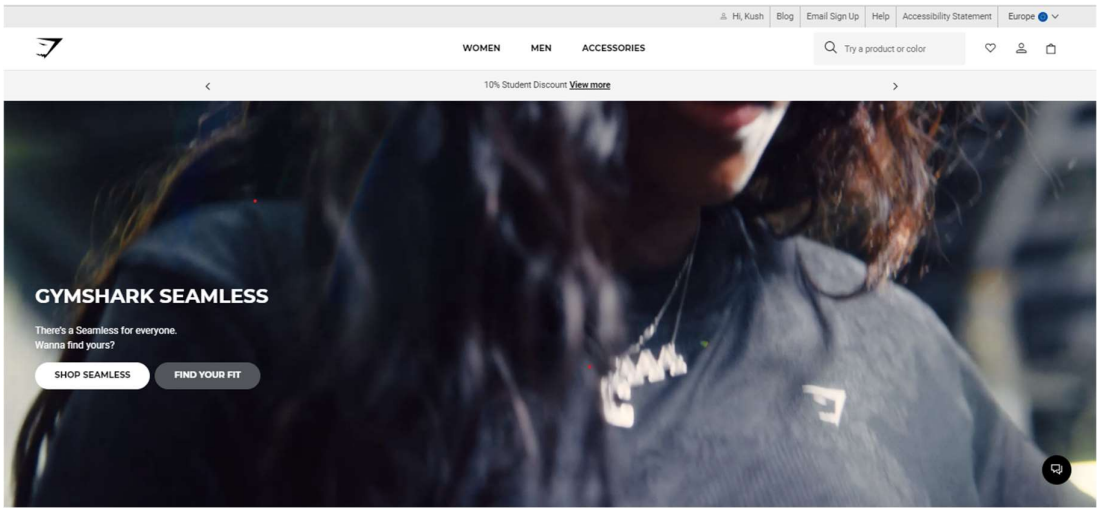

Figure 6 Homepage of Gymshark Website

Moreover, as illustrated in the figure 7 below, the website's mind-map can serve as a basis for the development of manual test plans. The utilization of a mind map is a straightforward technique for ideation, whereby a visual representation is employed to showcase a collection of tasks, terms, concepts, or articles that are associated with a central concept or topic. The non-linear design of the mind map allows for the creation of an intuitive context around a fundamental idea. Prior to initiating the implementation of test automation, it is imperative to first construct a mind-map and document test cases for the purpose of manual test execution of the website in question.

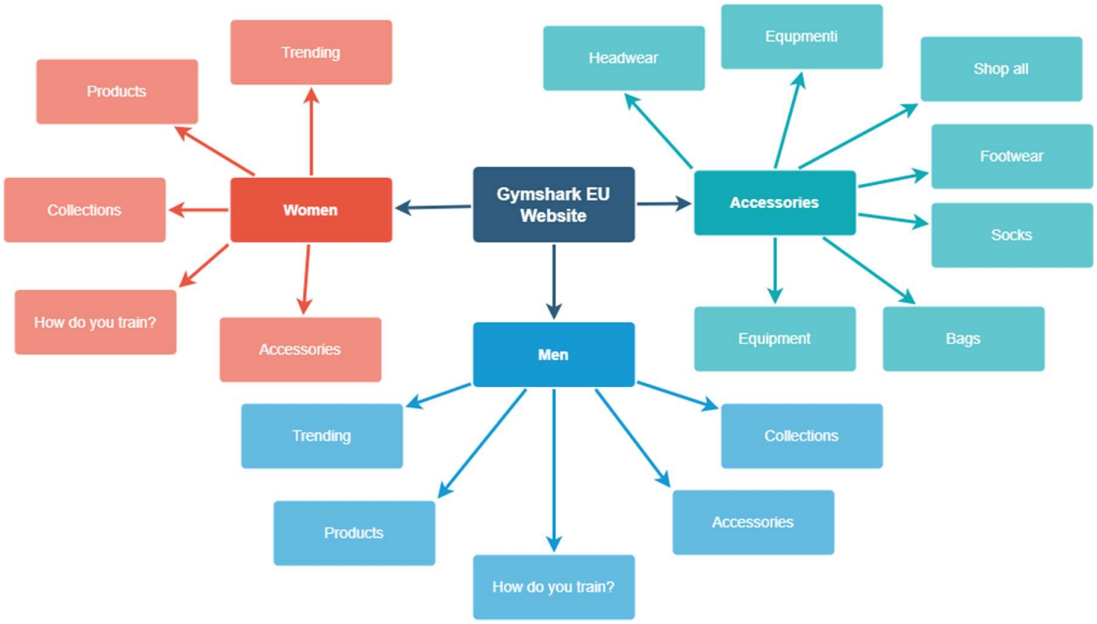

Figure 7 Mind map of Gymshark website

From the above figure 7 the mind-map of the website, I have selected 10 main test cases that identify the journey of a new and existing customer of the Gymshark website. During the process of creating and documenting test cases and their scenarios, the website introduced a special discount offer for students thus I have included it as a special test case that navigates to that page and views its terms and conditions. The following section and table present the documented test cases.

The below table 4, describes 10 main test cases as well as 1 additional test case regarding student discount. The following things are described in detail regarding each test case: test case name, its scenario and perquisite condition and the expected output or the result. The base nomenclature of test case is TC_0XX, where X represents the sequential number. Certain test cases such as TC_002 that checks invalid input during the sign up process are important to check because the invalid or false inputs have the potential to break the website, so it has the utmost importance to detect that website is functioning correctly.

Table 4 Test cases for Gymshark EU

Test Case ID	Test Case Name	Explanation/Test Description	Pre-Condition/ Prerequisite	Expected result
TC_001	Launch the app	To ensure whether the app gets launched successfully.	URL must be available: https://eu.shop.gymshark.com	Browser gets launched.
TC_002	Create a new account with empty fields in the form and invalid password.	To ensure new accounts are not created successfully when there are empty mandatory fields in the form or password invalid.	Home page is open.	Page loads
TC_003	Creating a new account	To ensure that a new account gets created when all required data is entered.	Sign Up page is opened.	All elements are loaded.
TC_004	Log into the website	To verify that the user can login to the website successfully.	Home page is open.	Account' button is located at the top-right corner of the page.
TC_005	Search for a product	To ensure the Search feature is working successfully	1. User is logged in. 2. Home page is open.	Login page is opened.
TC_006	Adding and removing items to and from the cart/bag	To ensure that the user can add and remove items to the shopping bag/cart.	1. User is logged in. 2. Home page is open.	"Sign Up" form is displayed.
TC_007	Student Discount	To ensure that the student discount page is displayed correctly.	1. User is logged in. 2. Home page is open.	"Your password does not match our requirements" message is displayed in red. Also, Grey dot and Green Tick appear below the message.
TC_008	Men's Joggers	To ensure that a particular item can be picked from the list.	1. User is logged in. 2. Home page is open.	Data is populated in the textbox.
TC_009	Wishlist	To verify that the Wishlist feature	1. User is logged in. 2. Home page is open.	"Please fill in this field" message

		works successfully.		appears on email field.
TC_010	Filter and Sort	To ensure that the filter and sort feature works successfully.	1. User is logged in. 2. Home page is open.	Data is populated in the textbox.
TC_011	Logout	To verify that the user can successfully log out from the website.	1. User is logged in. 2. Home page is open.	"Please include a @ in the Email address" message is displayed.

4.1 Test automation implementation

As stated before, the thesis aims to identify and select potential suitable test automation framework for ecommerce websites for the purpose of test automation. The primary objective is to examine the features of both the frameworks, their ease of use, execution time as well as investigate how easy it is to learn the framework and their scripting process.

According to section 3.3, I have installed the listed dependencies and its latest toolset version. Both Selenium Webdriver and Playwright framework provide extensive documentation on the process of installing and configuring the framework system as well as example programs are provided in Java, Javascript and Python. Selenium Webdriver has widely adopted design pattern technique in automated testing known as Page Object Model (POM).

4.1.1 Page Object Model

The Page Object Model (POM) design pattern entails the representation of each webpage as a Java class that encapsulates all the elements and actions associated with the page. The page elements are defined by the class through the utilization of locators such as ID, name, class name, CSS selector, or XPath. The methods offered by each page class enable the execution of various actions on the corresponding elements, including but not limited to button clicking, form filling, and text retrieval.

It is utilized for the purpose of systematically organizing and managing web pages and their constituent elements in a structured and coherent manner. The utilization of the Page Object Model (POM) design pattern has been shown to enhance the maintainability and scalability of automated testing procedures through the segregation of test code from page-specific code.

I have used the features of POM as well as utilized the element locators' option of finding the element by XPath. Here is an example line 1 of finding an element in Gymshark website and clicking on it. Line 2 is an example code of class *HomePage* where all the XPath of elements of homepage are stored and further called in the main class *MainTestCases*.

```
driver.findElement(By.xpath("//a[contains(text(),'t shirt')]")).click(); //line 1
Public static By homePageSearchButton =
By.xpath("//header/div[1]/div[2]/div[1]/button[1]"); //line 2
```

For the playwright, the process of creating test automation script was straightforward. The website was easier to access and navigate from the code. One of the key advantages Playwright had over Selenium Webdriver was it uses Locator API for locating all the web elements in my case website had some dynamic elements with variable paths and locating them with Playwright was easier and efficient. I followed the simple steps provided by Playwright writing test documentation with regards to creating test scripts using Playwright framework in Visual Studio. [29]

4.2 Result analysis and challenges

The 10 main testcases covering the most important scenarios are listed below and these names mentioned below are used as function names in the program.

- Launch testcase
- SignupWithEmptyFields testcase
- Signup testcase
- Login testcase

- SearchItem testcase
- ShoppingCart testcase
- StudentDiscount testcase
- MensJogger testcase
- Wishlist testcase
- FilterAndSort testcase
- Logout testcase

Initially, For Selenium Webdriver out of the 10 testcases, 4 kept failing randomly due to the behavior of the website. The website uses dynamic elements and XPaths that make test automation process challenging as well as the special test case of *StudentDiscount* consist of third-party link that sometimes block the access of the page. These complications were never encountered in Playwright.

The Gymshark website itself poses some challenges for running test automation especially in Selenium Webdriver framework. The website consists of media snippets playing during the execution of the program. Hence, Making it time-consuming and for some parts of testing. I had to add a bit of delay or wait time in order to move ahead.

For Playwright, creating the test automation code was straightforward and step by step guidelines of its functions and its use case were written in the Playwright website's documentation section. I encounter no issues while navigating and clicking using JavaScript in Playwright. The visual studio environment is integrated well with the Playwright framework. Below is an example screenshot of a successful test case of implementing invalid password credentials for the signup process in Playwright.

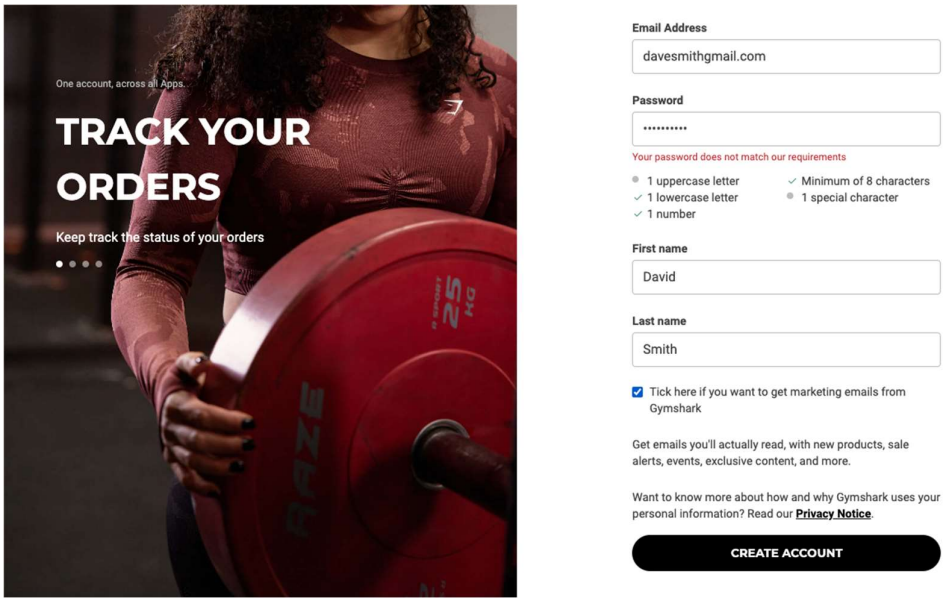

Figure 8 Screenshot passed test case for invalid signup using Playwright framework

For Playwright, creating the test automation code was straightforward and step by step guidelines of its functions and its use case were written in the Playwright website's documentation section. I encounter no issues while navigating and clicking using JavaScript in Playwright. The visual studio environment is integrated well with the Playwright framework.

The above figure 8 is an example screenshot of a successful test case of implementing invalid password credentials for the signup process in Playwright. As described Table 4 description section, checking with help of test automation if the website detects invalid, empty or null empty and returns back an error message. As seen on figure 8, red error message under password section is visible.

All the test cases were successfully carried out in Playwright. The biggest advantage of Playwright was that I was able to use data locator id to locate elements easily whereas with Selenium locating a dynamic element was difficult. Moreover, while using Selenium Webdriver and Chrome browser the website was constantly asking for access to cookies hence initially results in failed test cases whereas using Playwright and Chromium browser there were no issue with cookies only had to accept them once. Playwright also has an in-

built video recording option providing easy access to screen recording the progress and test cases whereas I had to use third-party add on feature for Selenium Webdriver.

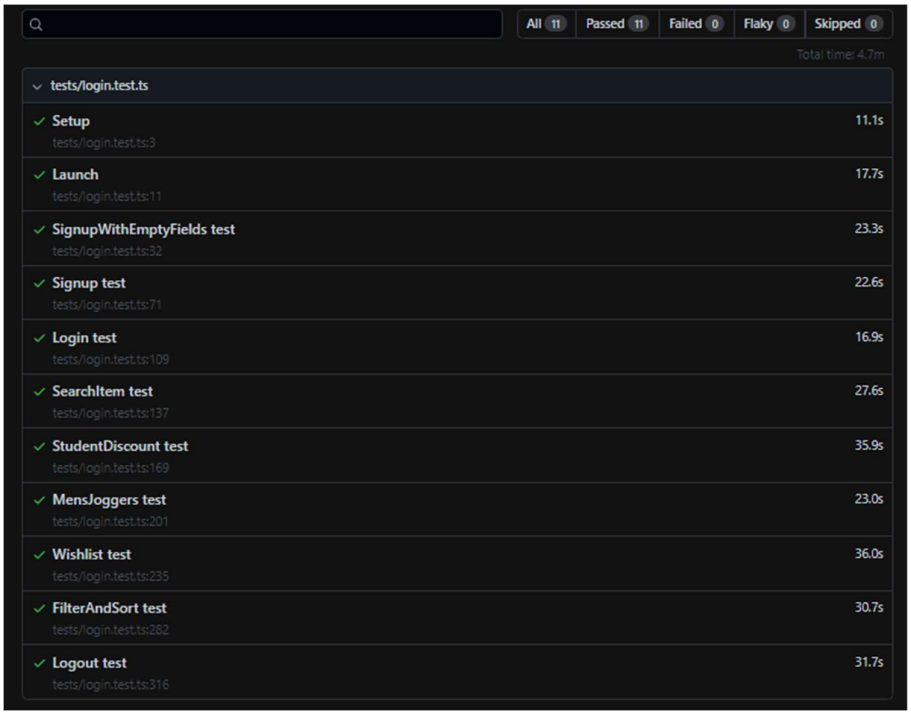

Figure 9 Playwright test results

Class	Method	Start	Time (ms)
Suite			
Test — passed			
testcases.MainTestcases	filterSortTest	1684044523620	6691
	launch	1684044404878	15257
	login	1684044434919	16548
	logoutTest	1684044530312	14245
	mensJoggersTest	1684044498093	2702
	searchItem	1684044451469	14951
	shoppingCart	1684044466424	7194
	signUpWithEmptyFields	1684044420150	14765
	studentDiscountTest	1684044473621	24471
	wishlistTest	1684044500798	22821

Figure 10 Selenium Webdriver test results

Figure 9 and 10, display the successful execution of test automation program. Both Playwright and Selenium Webdriver were able to run all test cases successfully. The initial issue in Selenium framework regarding some test cases failing was solved after making

adjustment in the code and changing to XPaths certain elements. To further evaluate and compare both the frameworks table is created below to compare the successful test cases and its execution time.

Table 5 Comparison of execution time between Selenium Webdriver and Playwright

Test Case ID	Test Case	Test function name	Selenium Webdriver (Execution time in seconds)	Playwright (Execution time in seconds)
TC_001	Launch the app	Launch test	15.3 seconds	17.1 seconds
TC_002	Create a new account with empty fields in the form and invalid password.	SignupWithEmptyFields test	14.8 seconds	23.3 seconds
TC_003	Creating a new account	Signup test	Skipped	22.6 seconds
TC_004	Log into the website	Login test	15.5 seconds	16.9 seconds
TC_005	Search for a product	SearchItem test	14.9 seconds	27.6 seconds
TC_006	Adding and removing items to and from the cart/bag	ShoppingCart test	7.2 seconds	
TC_007	Student Discount	StudentDiscount test	24.5 seconds	35.9 seconds
TC_008	Men's Joggers	MensJogger test	2.7 seconds	23 seconds
TC_009	Wishlist	Wishlist test	22.8 seconds	36 seconds
TC_010	Filter and Sort	FilterAndSort test	6.7 seconds	30.7 seconds
TC_011	Logout	Logout test	14.2 seconds	31.7 seconds
Total time			**2.3 minutes**	**4.6 minutes**

From the above table 5 and result discussed before, Comparison between two frameworks can be made. At first, Playwright automation script was run in Visual Studio and the results were stored as test report with the help of in-built report library in Playwright as a html file named *index.html* under *playwright-report* as shown in figure 5. The screenshot of *index.html* shown on figure 9. The total execution time was calculated with the help Playwright in built report library.

Furthermore, for Selenium Webdriver with the help of TestNG [31] to create test report, a framework for testing based and inspired by Junit with additional features and plugin options. Similarly, *index.html* and *emailable-report.html* are stored under *surefire-reports* as shown in the figure *5* top right side.

The Selenium Webdriver is a robust framework and well-liked by QA and testing specialists whereas the upcoming ease-to-use Playwright framework well integrated with Microsoft's visual studio is also a viable option. For Gymshark EU end to end testing Selenium proved to faster than Playwright. The total execution time for Playwright was 4.6 minutes on the other hand Selenium Webdriver is twice as fast as the total time was 2.3 minutes. During script development, several challenges were encountered with Selenium Webdriver. Also, locating the elements in Selenium specially for those websites who uses dynamic elements and *iframe* tags is very difficult.

Playwright locates the elements more effectively with the help of data locator and by simply searching by name as well as Scripting process is very easy. In Selenium Webdriver, there were more line of codes as compared to Playwright Javascript coding. Debugging in Selenium framework took a long time whereas debugging in Playwright with visual studio was very easily.

In Selenium Webdriver, there was an instance were automating a click event on a floating button was challenging. However, in Playwright, the button was located without any difficulty and was used in the program without any trouble.

The video recording and taking screenshot in Playwright is easier and takes less Megabytes (MB) to store about 20-30 MB whereas in Selenium Webdriver with third party support, it takes more than 100 MB. Hence, Playwright consumes less hard-drive space with the help of in-built recording libraries.

In conclusion, both Selenium and Playwright have their strengths and weaknesses, and the choice between them depends on your type of website and its specific requirements. If there is a requirement for support of multiple languages and browsers, it is plausible that opting for Selenium Webdriver would be a best. If one is seeking a more user-friendly framework with superior performance capabilities, it may be advisable to consider Playwright as a more viable alternative.

With the comparative analysis and end to end testing conducted in this Master thesis, it would be helpful for new and existing ecommerce websites on how to select the correct framework for themselves as well as the explore multiple options in the market not only the de-facto frameworks existing for decades. Furthermore, importance user experience, searching for framework with in-built features such as Playwright which is open source helps companies and individuals in starting the entrepreneurship journey on budget providing the best customer experience with the help of new and robust open-source testing framework.

Future research can be conducted further, testing the cross compatibility of the website with different types of electronic devices like mobile phones, tablets, and smartwatches. Cross browser compatibility test can be performed further to examine the website performance under different platforms. Moreover, end to end testing right from user lands at the homepage to checkout and payment can be test further with Cypress another popular test automation framework mentioned in the comparative analysis. Hybrid testing can be perform using combination of frameworks like Playwright and Cypress or Selenium Webdriver with Playwright and comparing the results, speed and execution time with the result published in this thesis work to compare efficiency.

5 Summary

Ecommerce websites are ever evolving and the need to automate the test processes is rising day by day. Web test automation provides a viable alternative to the tedious manual testing process which is prone to human error. New test automation technologies are coming up in the market. Playwright is a new upcoming open-sourced test automation framework which is an alternative to the established Selenium framework. Selenium Webdriver is a widely adopted test automation framework that is utilized for the purpose of testing web applications.

I performed comparative analysis to make informed decision while choosing between Selenium Webdriver and Playwright highlighting their pros and cons as well as performing end-to-end testing on popular fitness clothing brand Gymshark EU. Creating test cases on basics of website mind map and customer journey later performing test automation and scripting them in Java and Javascript respectively.

After conducting test automation and executing the programs for Selenium Webdriver and Playwright. The following are the major findings:

- The Playwright framework is easy to use, and detailed documentation is provided on their website.
- For Gymshark EU website, Selenium Webdriver is 2 times faster than Playwright.
- Scripting in Java using Selenium Webdriver is tedious and has a lot more code to write than Playwright.
- Playwright locator API and data locator id is superior has an edge over Selenium's find by element method when handing dynamic elements and floating buttons.
- Playwright functions and features are easy to learn and implement.
- Playwright has in-built recording libraries taking less hard-disk space then third-party supported recording in Selenium Webdriver.

Playwright is a viable option to consider and further improvements in Playwright would improve its execution speed and computational time. Selenium Webdriver has its robust Page Object Model that gives an edge over its completion.

References

[1] O. Djuraskovic, "FirstSiteGuide," 28 09 2022. [Online]. Available: https://firstsiteguide.com/how-many-websites/.

[2] Statista, 2021. [Online]. Available: https://www.statista.com/forecasts/480797/e-commerce-revenue-forecast-in-the-world.

[3] R. Angmo and M. Sharma, "Performance Evaluation of Web based Automation Testing Tools," *IEEE,* no. 5th International Journal of Computer Science and Information Technologie, pp. 908-912, 2014.

[4] B. Boehm and V. Basili, "Software Defect Reduction," *IEEE,* vol. 34, no. 1, pp. 135-137, 2001.

[5] A. Leitner, I. Ciupa and B. Meyer, "Reconciling Manual and Automated Testing: the AutoTest Experience," no. Proceedings of the 40th Hawaii International Conference on System Sciences, 2007.

[6] Selenium, Software Freedom Conservancy, [Online]. Available: https://www.selenium.dev/.

[7] Playwright , "Playwright by Microsoft," [Online]. Available: https://playwright.dev/.

[8] M. Limaye, Software Testing – Principles, Techniques and tools, Tata MCGraw-Hill Education PVT Ltd , 2009.

[9] Gymshark, "Gymshark Europe," [Online]. Available: https://eu.gymshark.com/.

[10] E. Vila et al., "Automation Testing Framework for Web Applications with Selenium WebDriver: Opportunities and Threats," *ICAIP,* no. Association for Computing Machinery, 2017.

[11] M. Jamil et al., "Software Testing Techniques: A Literature Review," *6th International Conference on Information and Communication Technology for The Muslim World (ICT4M), Jakarta, Indonesia,* no. IEEE, pp. 177-182, 2016.

[12] The International Software Testing Qualifications Board, ISTQB, 2017. [Online]. Available: https://www.istqb.org/.

[13] A. Husen, "Maintainable Test Suite Design using Page Object Model in Selenium Webdriver," no. Taltech, 2020.

[14] Smartbear, "Software Testing Methodologies," [Online]. Available: https://smartbear.com/learn/automated-testing/software-testing-methodologies/.

[15] M. Tufano, D. Drain, A. Svyatkovskiy and N. Sundaresan, "Generating accurate assert statements for unit test cases using pretrained transformers," no. IEEE, pp. 54-64, 2022.

[16] T. Virtanen, "Literature Review of Test Automation Models in Agile Testing," 2018. [Online]. Available: https://trepo.tuni.fi/bitstream/handle/123456789/25869/Virtanen.pdf?sequence=4.

[17] M. Kaur, Software testing and quality assurance, New Delhi: Excel books private limited, 2017.

[18] J. Húska, "Automated Testing of the Component-based Web Application User," 2017.

[19] ISO/IEC/IEEE International Standard, "Software and systems engineering – Software testing –Part 3: Test documentation," Vols. ISO/IEC/IEEE 29119-3, no. IEEE, p. 1/138, 2013.

[20] D. Hoffman, "Test Automation Architectures: Planning for Test Automation," 1999.

[21] A. Cervantes, "Exploring the use of a test automation," no. IEEE Aerospace conference, pp. 7-14, 2009.

[22] Cypress, "Cypress," [Online]. Available: https://learn.cypress.io/testing-foundations/manual-

vs-automated-testing.

[23] R. Dahiya and S. Ali, "Importance of Manual and Automation Testing," 2019.

[24] A. S. Gadwal, "Comparative review of the literature of automated testing tools," *ResearchGate,* 2020.

[25] M. Monier, "International, Evaluation of automated web testing tools," 2015.

[26] P. Raulamo-Jurvanen, "Practitioner Evaluations on Software Testing tools," 2019.

[27] E. S. Architecture, "Gorton, Ian," *Springer-verlag berlin Heidelberg,* vol. second edition, 2011.

[28] Nightwatch, "Nightwatch," 15 05 2023. [Online]. Available: https://nightwatchjs.org/.

[29] P. w. Tests, "Playwright," [Online]. Available: https://playwright.dev/docs/writing-tests. [Accessed 15 05 2023].

[30] "State of javascript 2021," 2021. [Online]. Available: https://2021.stateofjs.com/en-US/libraries/testing/. [Accessed 08 May 2023].

[31] TestNG, "TestNG," [Online]. Available: http://testng.org/doc/. [Accessed 15 05 2023].

Appendix 1 – Non-exclusive license for reproduction and publication of a graduation thesis[1]

I Kush Hiren Brahmbhatt

1. Grant Tallinn University of Technology free license (non-exclusive license) for my thesis "Comparative analysis of selecting a test automation framework for an e-commerce website", supervised by Jekaterina Tšukrejeva.
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive license.
3. I confirm that granting the non-exclusive license does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

08.05.2023

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Test cases written in Javascript using Playwright framework and Chromium web browser

```
import { chromium, test } from "@playwright/test"

test("Setup", async () => {
   const browser = await chromium.launch({
      headless: false
   });
   const context = await browser.newContext();
   const page = await context.newPage();
});

test("Launch", async ({page}) => {
   await page.goto("https://eu.shop.gymshark.com/");
   await page.waitForTimeout(4000);

   //await page.screenshot({ path: 'screenshots/001_launch.png' });
   console.log("Website launched...");
   console.log("Cookies close button clicked...");

   await page.click('[data-locator-id="storeSelector-confirm-select"]');
   await page.waitForTimeout(4000);
   //await page.screenshot({ path: 'screenshots/002_HomePage.png' });
   console.log("Location Confirm button clicked...");

   await page.waitForTimeout(4000);
});

test("SignupWithEmptyFields test", async ({page}) => {
   await page.goto("https://eu.shop.gymshark.com/");
   await page.waitForTimeout(4000);

   //await page.screenshot({ path: 'screenshots/Testing.png' });
   await page.click('[data-locator-id="storeSelector-confirm-select"]');
   await page.click('[data-locator-id="secondaryLinks-account-select"]');
   console.log("Home page Account button clicked...");
   await page.waitForTimeout(8000);

   await page.click('[id="tab-2"]');
   //await page.screenshot({ path: 'screenshots/003_SignUpPage.png' });
   console.log("Sign Up Tab clicked...");

   await page.click('[id="signup-email"]');
   console.log("Sign Up Email textbox clicked...");
   await page.locator('[id="signup-email"]').fill('davesmithgmail.com');
```

```
    console.log("Entered an invalid email in the textbox...");
    await page.click('[id="new-password"]');
    await page.locator('[id="new-password"]').fill('asdfg12345');
    console.log("Entered an invalid password in the textbox...");

    await page.click('[id="signup-firstname"]');
    await page.locator('[id="signup-firstname"]').fill('David');
    await page.click('[id="signup-lastname"]');
    await page.locator('[id="signup-lastname"]').fill('Smith');
    await page.click('[id="signup-email_optin"]')
    //await page.screenshot({ path: 'screenshots/004_SignUpFormFilled.png' });

    await page.click('[id="btn-signup"]');
    await page.waitForTimeout(6000);
});

test("Signup test", async ({page}) => {
    await page.goto("https://eu.shop.gymshark.com/");
    await page.waitForTimeout(4000);

    await page.click('[data-locator-id="storeSelector-confirm-select"]');

    await page.click('[data-locator-id="secondaryLinks-account-select"]');
    console.log("Home page Account button clicked...");
    await page.waitForTimeout(8000);

    await page.click('[id="tab-2"]');
    //await page.screenshot({ path: 'screenshots/005_SignUpPage.png' });
    console.log("Sign Up Tab clicked...");

    await page.click('[id="signup-email"]');
    console.log("Sign Up Email textbox clicked...");
    await page.locator ('[id="signup-email"]').fill('sraj17916@gmail.com');
    console.log("Entered a valid email in the textbox...");
    await page.click('[id="new-password"]');
    await page.locator('[id="new-password"]').fill('A$dfg12345');
    console.log("Entered a valid password in the textbox...");

    await page.click('[id="signup-firstname"]');
    await page.locator('[id="signup-firstname"]').fill('John');

    await page.click('[id="signup-lastname"]');
    await page.locator('[id="signup-lastname"]').fill('Mason');
    await page.click('[id="signup-email_optin"]');

    //await page.screenshot({path:'screenshots/006_SignUpFormFilledValid.png' });

    await page.click('[id="btn-signup"]');
```

```
    await page.waitForTimeout(6000);
});

test("Login test", async ({page}) => {
    await page.goto("https://eu.shop.gymshark.com/");
    await page.waitForTimeout(4000);
    await page.click('[data-locator-id="storeSelector-confirm-select"]');

    await page.click('[data-locator-id="secondaryLinks-account-select"]');
    console.log("Home page Account button clicked...");
    await page.click('[id="tab-1"]');
    //await page.screenshot({ path: 'screenshots/007_LogInPage.png' });
    console.log("Log In Tab clicked...");
    await page.click('[id="login-email"]');
    console.log("Log In Email textbox clicked...");
    await page.locator('[id="login-email"]').fill('kushbrahm3@gmail.com');
    console.log("Entered a valid email in the textbox...");
    await page.click('[id="current-password"]');
    await page.locator('[id="current-password"]').fill('A$dfg12345');
    console.log("Entered a valid password in the textbox...");

    //await page.screenshot({ path: 'screenshots/008_LogInFormFilledValid.png' });
    await page.click('[id="btn-login"]');
    console.log("Clicked on the Login Button...");

    await page.waitForTimeout(5000);

});

test("SearchItem test", async ({page}) => {
    await page.goto("https://eu.shop.gymshark.com/");
    await page.waitForTimeout(4000);

    await page.click('[data-locator-id="storeSelector-confirm-select"]');
    await page.waitForTimeout(6000);

    //await page.screenshot({ path: 'screenshots/Testing.png' });

    await page.click('[data-locator-id="header-searchContainer-read"]');
    console.log("Clicked on the Search Button...");

    await page.waitForTimeout(4000);
    await page.click('[data-locator-id="search-search-enter"]');
    //await page.screenshot({ path: 'screenshots/009_SearchBox.png' });
    await page.locator('[data-locator-id="search-search-enter"]').fill('shorts');
    await page.keyboard.press('Enter');
    console.log("Entered Search term in the Search textbox...");
```

```
    await page.waitForTimeout(4000);
    console.log(await page.textContent('[data-locator-id="plp-productCount-read"]'));
    //await page.screenshot({ path: 'screenshots/010_SearchResults.png' });

    await page.waitForTimeout(4000);
});

test("StudentDiscount test", async ({page}) => {
    await page.goto("https://eu.shop.gymshark.com/");
    await page.waitForTimeout(4000);

    await page.click('[data-locator-id="storeSelector-confirm-select"]');
    await page.waitForTimeout(4000);

    await page.click('[title="10% Student Discount"]');
    console.log("Clicked on the View More link...");
    //await page.screenshot({ path: 'screenshots/011_ViewMoreClicked.png' });

    await page.waitForTimeout(6000);
    await page.goto("https://connect.studentbeans.com/v4/gymshark-
eu?stb_offer_path=https%3A%2F%2Feu.shop.gymshark.com%2Fpages%2Fstudentbeans
&validate_iframe=true");
    //await page.screenshot({ path: 'screenshots/012_TAndCCookies.png' });

    await page.waitForTimeout(5000);
    //await page.keyboard.press('Enter');
    await page.click('[id="onetrust-accept-btn-handler"]');
    await page.waitForTimeout(4000);
    await page.locator('text="See Terms & Conditions"').click();
    await page.waitForTimeout(4000);

    //await page.screenshot({ path: 'screenshots/013_TermsAndConditions.png' });

    console.log("Clicked on the Terms and Conditions link...");
    await page.waitForTimeout(4000);

});

test("MensJoggers test", async ({page}) => {
    await page.goto("https://eu.shop.gymshark.com/");
    await page.waitForTimeout(4000);

    await page.click('[data-locator-id="storeSelector-confirm-select"]');
    await page.waitForTimeout(4000);

    await page.hover('[title="Men"]');
    //await page.screenshot({ path: 'screenshots/Testing.png' });
```

```
    await page.hover('[href="https://eu.shop.gymshark.com/collections/joggers/mens"]');
    //await page.screenshot({ path: 'screenshots/Testing2.png' });

    await page.click('[href="https://eu.shop.gymshark.com/collections/joggers/mens"]');
    //await page.screenshot({ path: 'screenshots/014_MenJoggers.png' });

    await page.waitForTimeout(4000);
    //await page.click('[data-locator-id="plp-size-m-select"]');
    await page.getByRole('button', { name: 'm' }).nth(1).click();

    await page.waitForTimeout(4000);
    //await page.screenshot({ path: 'screenshots/Testing3.png' });

    await page.getByRole('button', { name: 'l' }).nth(1).click();
    //await page.click('[data-locator-id="plp-size-l-select"]');
    //await page.screenshot({ path: 'screenshots/Testing3.png' });
    await page.waitForTimeout(4000);

    console.log(await page.textContent('[class="summary_summary-info-
wrapper__UvYyZ"]'));
    //await page.screenshot({ path: 'screenshots/015_MenJoggersCart.png' });

});

test("Wishlist test", async ({page}) => {
    await page.goto("https://eu.shop.gymshark.com/");
    await page.waitForTimeout(4000);

    await page.click('[data-locator-id="storeSelector-confirm-select"]');
    await page.waitForTimeout(4000);

    await page.click('[data-locator-id="header-searchContainer-read"]');
    console.log("Clicked on the Search Button...");
    await page.waitForTimeout(4000);

    await page.click('[data-locator-id="searchModal-trendingTerm-t_shirt-select"]');
    //await page.screenshot({ path: 'screenshots/016_TrendsTShirt.png' });
    //await page.click('[title="Geo Seamless T-Shirt"]');
    await page.click('[class="wishlist-button_icon-container__mFQoT"]');
    await page.waitForTimeout(2000);

    //await page.screenshot({ path: 'screenshots/017_AddedToWishlist.png' });
    await page.click('[data-locator-id="loginPrompt-login-select"]');

    await page.click('[id="login-email"]');
    await page.locator('[id="login-email"]').fill('rsrajeshs@gmail.com');
    await page.click('[id="current-password"]');
    await page.locator('[id="current-password"]').fill('A$dfg12345');
```

```
    await page.click('[id="btn-login"]');
    await page.waitForTimeout(4000);
    await page.click('[class="header_action-bar-item-wrap__m_Rhz"]');
    await page.waitForTimeout(4000);
    //await page.screenshot({ path: 'screenshots/018_WishlistConfirmation.png' });
    console.log(await page.textContent('[data-locator-id="plp-productCount-read"]'));

});

test("FilterAndSort test", async ({page}) => {
    await page.goto("https://eu.shop.gymshark.com/");
    await page.waitForTimeout(4000);
    await page.click('[data-locator-id="storeSelector-confirm-select"]');
    await page.waitForTimeout(4000);

    await page.hover('[title="Men"]');
    await page.hover('[href="https://eu.shop.gymshark.com/collections/sport/mens"]');
    //await page.screenshot({ path: 'screenshots/Testing2.png' });
    await page.waitForTimeout(4000);
    //await page.screenshot({ path: 'screenshots/019_MensSport.png' });

    await page.click('[href="https://eu.shop.gymshark.com/collections/sport/mens"]');
    await page.waitForTimeout(4000);
    console.log(await page.textContent('[data-locator-id="plp-productCount-read"]'));

    await page.click('[data-locator-id="plp-filterButton-select"]');
    await page.waitForTimeout(2000);
    //await page.screenshot({ path: 'screenshots/020_FilterAndSort.png' });
    await page.click('[data-locator-id="filters-filterCategory-PRICE-select"]')

    await page.click('[data-locator-id="filters-filterOption-20_30-select"]');
    await page.click('[data-locator-id="filters-seeProducts-select"]');
    await page.waitForTimeout(4000);
    console.log(await page.textContent('[data-locator-id="plp-productCount-read"]'));

});

test("Logout test", async ({page}) => {
    await page.goto("https://eu.shop.gymshark.com/");
    await page.waitForTimeout(4000);

    await page.click('[data-locator-id="storeSelector-confirm-select"]');
    await page.waitForTimeout(4000);

    await page.click('[data-locator-id="secondaryLinks-account-select"]');
    console.log("Home page Account button clicked...");
```

```
    await page.waitForTimeout(4000);
    await page.click('[id="tab-1"]');
    await page.click('[id="login-email"]');
    await page.locator('[id="login-email"]').fill('rsrajeshs@gmail.com');
    await page.click('[id="current-password"]');
    await page.locator('[id="current-password"]').fill('A$dfg12345');

    await page.click('[id="btn-login"]');
    await page.waitForTimeout(6000);

    await page.click('[data-locator-id="secondaryLinks-account-select"]');
    await page.waitForTimeout(4000);
    //await page.screenshot({ path: 'screenshots/050_AccountStatus.png' });


    await page.click('[data-locator-id="account-logoutButton-select"]');
    await page.waitForTimeout(4000);
    //await page.screenshot({ path: 'screenshots/051_LoggedOutSuccess.png' });

});
```

Appendix 3 – Test cases in Java using Selenium Webdriver framework and Chrome web browser

```
// Homepage.class contains xpaths of all the homepage elements used in MainTestCases

package pages;

import org.openqa.selenium.By;

public class HomePage {
 public static By homePageAccountButton =
By.xpath("//body[1]/div[1]/header[1]/div[1]/div[4]/a[1]");
 public static By homePageTopLeftLogo =
By.xpath("//body[1]/div[1]/header[1]/div[1]/span[1]/a[1]/span[1]");
 public static By homePageWomenNav =
By.xpath("//header/div[1]/div[1]/div[2]/div[2]/a[2]");
 public static By homePageMenNav =
By.xpath("//body[1]/div[1]/header[1]/div[1]/div[1]/div[2]/div[2]/a[4]");
 public static By homePageAccessoriesNav =
By.xpath("//header/div[1]/div[1]/div[2]/div[2]/a[2]");
 public static By homePageSearchButton =
By.xpath("//header/div[1]/div[2]/div[1]/button[1]");
 public static By homePageWishlistButton =
```

```
By.xpath("//body[1]/div[1]/header[1]/div[1]/div[3]/a[1]");
 public static By homePageCart = By.xpath("//header/div[1]/div[5]");
 public static By homePageSearchTextbox =
By.xpath("//body/div[6]/div[1]/div[1]/div[1]/div[1]/div[1]/input[1]");
 public static By homePageSearchCloseButton =
By.xpath("//body/div[6]/div[1]/div[1]/div[1]/div[1]/button[1]");
 public static By homePageSearchFirstArticle =
By.xpath("//body[1]/div[1]/main[1]/div[1]/section[1]/div[1]/article[1]/div[1]/a[1]");
 public static By homePageSearchViewAll =
By.xpath("//body/div[6]/div[1]/div[1]/div[1]/div[3]/div[2]/a[1]");
 public static By homePageCookiesLabel = By.xpath("//h2[@id='onetrust-policy-title']");
 public static By homePageCookiesClose = By.xpath("//body/div[@id='onetrust-consent-
sdk']/div[@id='onetrust-banner-sdk']/div[1]/div[2]/button[1]");
 public static By homePageAreYouInTheRightPlace = By.xpath("//h5[contains(text(),'are
you in the right place?')]");
 public static By homePagePleaseSelectAStore = By.xpath("//label[contains(text(),'Please
select a store')]");
 public static By homePageRightPlaceConfirmButton =
By.xpath("//button[contains(text(),'confirm')]");
 public static By homePageRightPlaceCloseButton =
By.xpath("//body/div[3]/div[1]/div[1]/button[1]");
 public static By homePageChooseLocation =
By.xpath("//body/div[5]/div[1]/div[1]/div[1]/div[1]/div[2]/select[1]");
 public static By homePageChooseLocationLabel =
By.xpath("//label[contains(text(),'Please select a store')]");
 public static By homePageLocationEurope =
By.xpath("//option[contains(text(),'Europe')]");
 public static By homePageLocationConfirmButton =
By.xpath("//button[contains(text(),'confirm')]");
 public static By homePageLocationCloseButton =
By.xpath("//body/div[5]/div[1]/div[1]/button[1]");
 public static By homePageDiscountLink =
By.xpath("//body[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[1]/d
iv[1]/a[1]");
 public static By studentbeansCookies = By.xpath("//button[@id='onetrust-accept-btn-
handler']");
 public static By studentDiscountTerms = By.xpath("//p[contains(text(),'See Terms &
Conditions')]");
 public static By JoggersFirstItem =
By.xpath("//body[1]/div[1]/main[1]/div[1]/section[1]/div[1]/article[1]/div[1]/div[1]/a[1]/d
iv[1]/span[1]");
 public static By quickAddSizeM =
By.xpath("//body[1]/div[1]/main[1]/div[1]/section[1]/div[1]/article[1]/div[1]/div[1]/div[1]
/div[1]/div[2]/div[1]/button[3]");
 public static By quickAddSizeL =
By.xpath("//body[1]/div[1]/main[1]/div[1]/section[1]/div[1]/article[1]/div[1]/div[1]/div[1]
/div[1]/div[2]/div[1]/button[4]");
 public static By youMightLikeSizeM =
By.xpath("//body[1]/div[8]/div[1]/div[1]/div[1]/div[2]/div[1]/section[1]/div[1]/article[1]/d
iv[1]/div[1]/div[1]/div[1]/div[2]/div[1]/button[4]");
```

```
 public static By cartCloseButton =
By.xpath("//body[1]/div[8]/div[1]/div[1]/div[2]/div[1]/button[1]");
 public static By trendTShirt =
By.xpath("//body[1]/div[6]/div[1]/div[1]/div[1]/div[2]/div[1]/div[1]/div[1]/div[4]/a[1]");
 public static By geoSeamlessTShirt = By.xpath("//h4[contains(text(),'Geo Seamless T-
Shirt')]");
 public static By geoSeamlessTShirtWishlist =
By.xpath("//body[1]/div[1]/main[1]/div[1]/section[1]/div[1]/article[2]/button[1]/span[1]/di
v[1]/i[1]");
 public static By menSport =
By.xpath("//body[1]/div[1]/header[1]/div[1]/div[1]/div[2]/div[2]/section[2]/div[1]/ul[1]/li[
3]/ul[1]/li[6]/a[1]");
 public static By filterAndSort =
By.xpath("//body[1]/div[1]/main[1]/section[1]/button[1]");
 public static By filterPrice =
By.xpath("//body[1]/div[12]/div[1]/div[1]/div[2]/details[7]");
 public static By filterPrice2030 =
By.xpath("//body[1]/div[12]/div[1]/div[1]/div[2]/details[6]/ul[1]/li[2]/input[1]");
 public static By seeFilterResults =
By.xpath("//body[1]/div[12]/div[1]/div[1]/div[3]/button[1]");
 public static By liftingFirstItem =
By.xpath("//body[1]/div[1]/main[1]/div[1]/section[4]/div[2]/div[1]/article[1]/div[1]/div[1]
/a[1]");
 public static By SearchResultsGrid = By.className("class=\"product-card_product-
card__gB8_b\"");

 public HomePage() {
 }
}
```

```
// SignUp.class contains xpaths of all the signup elements used in MainTestCases

package pages;

import org.openqa.selenium.By;

public class SignUp {
 public static By logInTab = By.xpath("//body[1]/div[1]/div[2]/div[1]/div[2]/button[1]");
 public static By logInEmailTextbox = By.xpath("//input[@id='login-email']");
 public static By logInPasswordTextbox = By.xpath("//input[@id='current-password']");
 public static By logInButton = By.xpath("//button[@id='btn-login']");
 public static By signUpTab =
By.xpath("//body[1]/div[1]/div[2]/div[1]/div[2]/button[2]");
 public static By signUpEmailTextbox = By.xpath("//input[@id='signup-email']");
 public static By signUpPassword = By.xpath("//input[@id='new-password']");
 public static By signUpFirstname = By.xpath("//input[@id='signup-firstname']");
 public static By signUpLastname = By.xpath("//input[@id='signup-lastname']");
```

```
 public static By signUpMarketingCheckbox = By.xpath("//input[@id='signup-
email_optin']");
 public static By signUpButton = By.xpath("//button[@id='btn-signup']");
 public static By signUpGreyDot =
By.xpath("//body/div[1]/div[2]/div[1]/div[3]/div[2]/form[1]/div[2]/div[1]/ul[1]/li[1]/div[1
]");
 public static By signUpGreenTick =
By.xpath("//body/div[1]/div[2]/div[1]/div[3]/div[2]/form[1]/div[2]/div[1]/ul[1]/li[2]/span[
1]");
 public static By signUpPasswordRequirements = By.xpath("//span[@id='for-
password']");
 public static By signUpLogout = By.xpath("//a[contains(text(),'Log out')]");
 public static By signUpYourOrders = By.xpath("//h3[contains(text(),'Your orders')]");

 public SignUp() {
 }
}
```

```
// Selenium Webdriver main class – MainTestCases

package testcases;

import io.github.bonigarcia.wdm.WebDriverManager;
import java.io.File;
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import org.apache.commons.io.FileUtils;
import org.monte.screenrecorder.ScreenRecorder;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Keys;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import org.testng.annotations.AfterSuite;
import org.testng.annotations.Test;
import pages.HomePage;
import pages.SignUp;
import utils.ConfigFileReader;
```

```
import utils.ScreenRecorderUtil;

public class MainTestcases {
 public static WebDriver driver;
 public static ScreenRecorder screenRecorder;
 public static String emailAddressSignUp =
ConfigFileReader.getConfigPropertyVal("emailAddressSignUp");
 public static String emailAddressLogIn =
ConfigFileReader.getConfigPropertyVal("emailAddressLogIn");
 public static String password = ConfigFileReader.getConfigPropertyVal("password");

 public MainTestcases() {
 }
 public static void main(String[] args) throws Exception {
 System.out.println("Main function...");
 MainTestcases object = new MainTestcases();
 object.launch();
 }


  @Test(priority = 0)
  public void launch() throws Exception {
  ScreenRecorderUtil.startRecord("launch");
  ChromeOptions chromeOptions = new ChromeOptions();
  WebDriverManager.chromedriver().setup();
  driver = new ChromeDriver(chromeOptions);
  driver.manage().window().maximize();
  driver.get("https://eu.shop.gymshark.com");
  System.out.println("Website is launched...");
  takeSnapShot(driver, "Screenshots/001_launch.png");
  ScreenRecorderUtil.stopRecord();
  }


  @Test( priority = 1, enabled = true )
  public void signUpWithEmptyFields() throws Exception {
  System.out.println("In signUpWithEmptyFields method...");
  ScreenRecorderUtil.startRecord("signUpWithEmptyFields");
  if (((WebElement)(new WebDriverWait(driver,
 30L)).until(ExpectedConditions.presenceOfElementLocated(HomePage.homePageCooki
```

```
esLabel))).isDisplayed()) {
 driver.findElement(HomePage.homePageCookiesClose).click();
 driver.manage().timeouts().implicitlyWait(3L, TimeUnit.SECONDS);
 driver.findElement(HomePage.homePageRightPlaceConfirmButton).click();
 System.out.println("Location submitted...");
 }

 if
(driver.findElement(By.xpath("//header/div[1]/div[1]/div[2]/div[3]/ul[1]/li[1]/a[1]")).get
Attribute("text").contains("account")) {
 System.out.println("User has not logged in yet.");
 driver.findElement(HomePage.homePageAccountButton).click();
 driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
 driver.findElement(SignUp.signUpTab).click();
 driver.findElement(SignUp.signUpEmailTextbox).sendKeys(new
CharSequence[]{"davesmith123gmail.com"});
 driver.findElement(SignUp.signUpPassword).sendKeys(new
CharSequence[]{"asdfg12345"});
 driver.manage().timeouts().implicitlyWait(2L, TimeUnit.SECONDS);
 takeSnapShot(driver, "Screenshots/002_PasswordInvalid.png");
 if (driver.findElement(SignUp.signUpGreyDot).isDisplayed()) {
 Assert.assertTrue(true, "\"Please include an @ in the Email address\" message is
displayed.");
 Assert.assertTrue(true, "Password does not match the requirements.");
 }

 driver.findElement(SignUp.signUpFirstname).sendKeys(new
CharSequence[]{"David"});
 driver.findElement(SignUp.signUpButton).click();
 takeSnapShot(driver, "Screenshots/003_SignUpButton.png");
 }

 else {
 driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
```

```
System.out.println("User already logged in...");
driver.findElement(HomePage.homePageAccountButton).click();
driver.manage().timeouts().implicitlyWait(3L, TimeUnit.SECONDS);
takeSnapShot(driver, "Screenshots/004_accountButton.png");
driver.findElement(SignUp.signUpLogout).click();
System.out.println("Logout link clicked.");
driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
System.out.println("User logged out.");
driver.findElement(HomePage.homePageTopLeftLogo).click();
driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
driver.findElement(HomePage.homePageAccountButton).click();
driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
driver.findElement(SignUp.signUpTab).click();
driver.findElement(SignUp.signUpEmailTextbox).sendKeys(new
CharSequence[]{"davesmith123gmail.com"});
driver.findElement(SignUp.signUpPassword).sendKeys(new
CharSequence[]{"asdfg12345"});
driver.manage().timeouts().implicitlyWait(2L, TimeUnit.SECONDS);
takeSnapShot(driver, "Screenshots/005_PasswordInvalid.png");
if (driver.findElement(SignUp.signUpGreyDot).isDisplayed()) {
Assert.assertTrue(true, "\"Please include an @ in the Email address\" message is
displayed.");
Assert.assertTrue(true, "Password does not match the requirements.");
}

driver.findElement(SignUp.signUpFirstname).sendKeys(new
CharSequence[]{"David"});
driver.findElement(SignUp.signUpButton).click();
takeSnapShot(driver, "Screenshots/006_SignUpButton.png");
}
ScreenRecorderUtil.stopRecord();
}

@Test( priority = 2, enabled = false)
```

```
public void signUp() throws Exception {
System.out.println("In signUp method...");
ScreenRecorderUtil.startRecord("signUp");
driver.get("https://eu.shop.gymshark.com");
driver.findElement(HomePage.homePageAccountButton).click();
driver.manage().timeouts().implicitlyWait(3L, TimeUnit.SECONDS);
driver.findElement(SignUp.signUpTab).click();
driver.findElement(SignUp.signUpEmailTextbox).sendKeys(new
CharSequence[]{"tottygreen123@gmail.com"});
driver.findElement(SignUp.signUpPassword).sendKeys(new
CharSequence[]{password});
driver.manage().timeouts().implicitlyWait(2L, TimeUnit.SECONDS);
driver.findElement(SignUp.signUpFirstname).sendKeys(new
CharSequence[]{"Kush"});
driver.findElement(SignUp.signUpLastname).sendKeys(new
CharSequence[]{"Smith"});
driver.manage().timeouts().implicitlyWait(2L, TimeUnit.SECONDS);
takeSnapShot(driver, "Screenshots/007_ValidCredentials.png");
driver.findElement(SignUp.signUpButton).click();
driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
if (driver.findElement(HomePage.homePageWishlistButton).isDisplayed()) {
Assert.assertTrue(true, "User Account created successfully.");
takeSnapShot(driver, "Screenshots/008_AccountCreated.png");
}
ScreenRecorderUtil.stopRecord();
}

@Test( priority = 3, enabled = true)
public void login() throws Exception {
System.out.println("In Login method...");
ScreenRecorderUtil.startRecord("login");
driver.get("https://eu.shop.gymshark.com");
driver.manage().timeouts().implicitlyWait(10L, TimeUnit.SECONDS);
if
```

```
(driver.findElement(By.xpath("//header/div[1]/div[1]/div[2]/div[3]/ul[1]/li[1]/a[1]")).get
Attribute("text").contains("account")) {
 System.out.println("User has not logged in yet.");
 driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
 driver.findElement(HomePage.homePageAccountButton).click();
 driver.manage().timeouts().implicitlyWait(2L, TimeUnit.SECONDS);
 driver.findElement(SignUp.logInTab).click();
 driver.manage().timeouts().implicitlyWait(2L, TimeUnit.SECONDS);
 driver.findElement(SignUp.logInEmailTextbox).sendKeys(new
CharSequence[]{emailAddressLogIn});
 driver.findElement(SignUp.logInPasswordTextbox).sendKeys(new
CharSequence[]{password});
 driver.manage().timeouts().implicitlyWait(2L, TimeUnit.SECONDS);
 takeSnapShot(driver, "Screenshots/009_LoginButton.png");
 driver.findElement(SignUp.logInButton).click();
 driver.manage().timeouts().implicitlyWait(20L, TimeUnit.SECONDS);

System.out.println(driver.findElement(By.xpath("//header/div[1]/div[1]/div[2]/div[3]/ul[
1]/li[1]/a[1]")).getAttribute("text").toString());
 driver.navigate().refresh();

Assert.assertTrue(driver.findElement(By.xpath("//header/div[1]/div[1]/div[2]/div[3]/ul[1
]/li[1]/a[1]")).getAttribute("text").contains("Hi"), "User successfully logged in.");
 System.out.println("Exiting Login method now...");
 } else {
 Assert.assertTrue(true, "User is already logged in.");
 System.out.println("User is already logged in.");
 driver.findElement(HomePage.homePageAccountButton).click();
 System.out.println("Account button clicked.");
 driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
 driver.findElement(SignUp.signUpLogout).click();
 System.out.println("Logout link clicked.");
 takeSnapShot(driver, "Screenshots/010_LoggedOut.png");
```

```
driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
System.out.println("User logged out.");
driver.findElement(HomePage.homePageTopLeftLogo).click();
System.out.println("Logo clicked...");
driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
driver.findElement(HomePage.homePageAccountButton).click();
System.out.println("Account button clicked...");
driver.manage().timeouts().implicitlyWait(10L, TimeUnit.SECONDS);
driver.findElement(SignUp.logInTab).click();
driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
driver.findElement(SignUp.logInEmailTextbox).sendKeys(new
CharSequence[]{emailAddressLogIn});
driver.findElement(SignUp.logInPasswordTextbox).sendKeys(new
CharSequence[]{password});
driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
takeSnapShot(driver, "Screenshots/011_LogInButton.png");
driver.findElement(SignUp.logInButton).click();
driver.manage().timeouts().implicitlyWait(10L, TimeUnit.SECONDS);
driver.findElement(HomePage.homePageTopLeftLogo).click();
driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);

Assert.assertTrue(driver.findElement(By.xpath("//header/div[1]/div[1]/div[2]/div[3]/ul[1
]/li[1]/a[1]")).getAttribute("text").contains("Hi"), "User successfully logged in.");
System.out.println("Exiting Login method now...");
}
ScreenRecorderUtil.stopRecord();
}
@Test( priority = 4, enabled = true )
public void searchItem() throws Exception {
System.out.println("In searchItem method...");
ScreenRecorderUtil.startRecord("searchItem");
driver.get("https://eu.shop.gymshark.com");
driver.manage().timeouts().implicitlyWait(5L, TimeUnit.SECONDS);
```

```
 driver.findElement(HomePage.homePageSearchButton).click();
 System.out.println("Search button clicked...");
 driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
 driver.findElement(HomePage.homePageSearchTextbox).sendKeys(new
CharSequence[]{"shorts"});
 takeSnapShot(driver, "Screenshots/012_SearchTerm.png");
 driver.findElement(HomePage.homePageSearchTextbox).sendKeys(new
CharSequence[]{Keys.ENTER});
 driver.manage().timeouts().implicitlyWait(10L, TimeUnit.SECONDS);
 if
(driver.findElement(By.xpath("//body[1]/div[1]/main[1]/div[1]/section[1]/div[1]/article[
1]/div[1]/div[1]/a[1]")).isDisplayed()) {
 System.out.println("Search items are displayed...");
 String results =
driver.findElement(By.xpath("//body[1]/div[1]/main[1]/section[1]/div[1]/span[2]")).getT
ext();
 takeSnapShot(driver, "Screenshots/013_ResultsCount.png");
 System.out.println("The search returned " + results);
 }
 ScreenRecorderUtil.stopRecord();
 }
 @Test( priority = 5, enabled = true, dependsOnMethods = {"searchItem"})
 public void shoppingCart() throws Exception {
 System.out.println("In shoppingCart method...");
 ScreenRecorderUtil.startRecord("shoppingCart");
 driver.get("https://eu.shop.gymshark.com");
 driver.manage().timeouts().implicitlyWait(5L, TimeUnit.SECONDS);
 driver.findElement(HomePage.homePageSearchButton).click();
 System.out.println("Search button clicked...");
 driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
 driver.findElement(HomePage.homePageSearchTextbox).sendKeys(new
CharSequence[]{"shorts"});
 driver.findElement(HomePage.homePageSearchTextbox).sendKeys(new
```

```
CharSequence[]{Keys.ENTER});
 driver.manage().timeouts().implicitlyWait(10L, TimeUnit.SECONDS);
 ScreenRecorderUtil.stopRecord();
 }
 public static void takeSnapShot(WebDriver webdriver, String fileWithPath) throws
IOException {
 TakesScreenshot scrShot = (TakesScreenshot)webdriver;
 File SrcFile = (File)scrShot.getScreenshotAs(OutputType.FILE);
 File DestFile = new File(fileWithPath);
 FileUtils.copyFile(SrcFile, DestFile);
 }
 @Test( priority = 6, enabled = true)
 public void studentDiscountTest() throws Exception {
 System.out.println("In studentDiscountTest method...");
 ScreenRecorderUtil.startRecord("studentDiscountTest");
 driver.get("https://eu.shop.gymshark.com");
 driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
 WebDriverWait waitDiscount = new WebDriverWait(driver, 20L);
waitDiscount.until(ExpectedConditions.elementToBeClickable(HomePage.homePageDis
countLink));
 driver.findElement(HomePage.homePageDiscountLink).click();
 System.out.println("Clicked on View more...");
 driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);

driver.get("https://connect.studentbeans.com/v4/gymshark/us?stb_offer_path=https%3A
%2F%2Fus.shop.gymshark.com%2Fpages%2Fstudentbeans&validate_iframe=true");
 driver.findElement(By.id("onetrust-accept-btn-handler")).click();
 driver.manage().timeouts().implicitlyWait(3L, TimeUnit.SECONDS);
 driver.findElement(By.className("_1eizlsp")).click();
 System.out.println("Clicked on Terms and Conditions...");
 Assert.assertTrue(true, "Clicked on Student Discount Terms and Conditions...");
 driver.navigate().back();
 driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
```

```
ScreenRecorderUtil.stopRecord();
driver.navigate().back();
}
@Test( priority = 7,enabled = true )
public void mensJoggersTest() throws Exception {
System.out.println("In mensJoggersTest method...");
ScreenRecorderUtil.startRecord("mensJoggersTest");
driver.findElement(HomePage.homePageTopLeftLogo).click();
driver.manage().timeouts().implicitlyWait(3L, TimeUnit.SECONDS);
WebElement ele = driver.findElement(HomePage.homePageMenNav);
Actions action = new Actions(driver);
action.moveToElement(ele).perform();
driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
takeSnapShot(driver, "Screenshots/017_MensMouseHover.png");
WebElement joggerEle =
driver.findElement(By.xpath("//body[1]/div[1]/header[1]/div[1]/div[1]/div[2]/div[2]/sect
ion[2]/div[1]/ul[1]/li[2]/ul[1]/li[3]/a[1]"));
action.moveToElement(joggerEle).perform();
driver.findElement(By.xpath("//body[1]/div[1]/header[1]/div[1]/div[1]/div[2]/div[2]/sect
ion[2]/div[1]/ul[1]/li[2]/ul[1]/li[3]/a[1]")).click();
driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
takeSnapShot(driver, "Screenshots/018_Joggers.png");
System.out.println("Clicked on Joggers...");
ScreenRecorderUtil.stopRecord();
}
@Test( priority = 8, enabled = true )
public void wishlistTest() throws Exception {
System.out.println("In wishlistTest method...");
ScreenRecorderUtil.startRecord("wishlistTest");
driver.get("https://eu.shop.gymshark.com");
driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
driver.findElement(HomePage.homePageSearchButton).click();
driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
```

```
WebDriverWait waitTrend = new WebDriverWait(driver, 30L);
WebElement Trend = driver.findElement(By.xpath("//a[contains(text(),'t shirt')]"));
waitTrend.until(ExpectedConditions.elementToBeClickable(Trend));
takeSnapShot(driver, "Screenshots/Testing.png");
driver.findElement(By.xpath("//a[contains(text(),'t shirt')]")).click();
driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
driver.findElement(By.xpath("//body/div[@id='__next']/main[@id='MainContent']/div[1
]/section[1]/div[1]/article[1]/div[1]/a[1]")).click();
ScreenRecorderUtil.stopRecord();
}

@Test(priority = 9, enabled = true )
public void filterSortTest() throws Exception {
System.out.println("In filterSortTest method...");
ScreenRecorderUtil.startRecord("filterSortTest");
driver.get("https://eu.shop.gymshark.com");
driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
WebElement ele = driver.findElement(HomePage.homePageMenNav);
Actions action = new Actions(driver);
action.moveToElement(ele).perform();
driver.manage().timeouts().implicitlyWait(3L, TimeUnit.SECONDS);
WebElement sportEle = driver.findElement(HomePage.menSport);
action.moveToElement(sportEle).perform();
driver.findElement(HomePage.menSport).click();
driver.manage().timeouts().implicitlyWait(4L, TimeUnit.SECONDS);
takeSnapShot(driver, "Screenshots/021_MenSport.png");
System.out.println("Clicked on Sport...");
ScreenRecorderUtil.stopRecord();
System.out.println("Filter applied...");
}

@Test( priority = 10, enabled = true )
public void logoutTest() throws Exception {
```

```
System.out.println("In logoutTest method...");
ScreenRecorderUtil.startRecord("logoutTest");
driver.get("https://eu.shop.gymshark.com");
driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
driver.findElement(HomePage.homePageAccountButton).click();
driver.manage().timeouts().implicitlyWait(2L, TimeUnit.SECONDS);
WebElement element =
driver.findElement(By.xpath("//*[@id=\"MainContent\"]/div[1]/h1[1]"));
String text = element.getText();
System.out.println(text);
Assert.assertEquals(text, "YOUR GYMSHARK ACCOUNT");
System.out.println(text);
String highlightScript = "arguments[0].style.border='3px solid red'";
takeSnapShot(driver, "Screenshots/023_AccountStatus.png");
Actions actions = new Actions(driver);
actions.moveToElement(element).perform();
((JavascriptExecutor)driver).executeScript(highlightScript, new Object[]{element});
driver.findElement(SignUp.signUpLogout).click();
System.out.println("Logout link clicked.");
driver.manage().timeouts().implicitlyWait(6L, TimeUnit.SECONDS);
takeSnapShot(driver, "Screenshots/024_LoggedOut.png");
System.out.println("User logged out.");
ScreenRecorderUtil.stopRecord();
}
@AfterSuite
public void QuitBrowser() throws InterruptedException, IOException {
Thread.sleep(5000L);
System.out.println("Quitting the Browser now...");
driver.quit();
}
}
```
