Eötvös Loránd University

Faculty of Informatics

Dept. of Software Technology and Methodology

Importance of Web Testing

Supervisor:
Attila Kovacs
Professor

Author:
Yusa Yalcin
Computer Science MSc

Budapest, 2023

Abstract

The internet has become an integral part of our lives and many businesses rely on websites to reach their customers. In order to make sure that these websites are operating effectively and providing a positive user experience, web testing has grown in importance. When tests are performed manually, it can be a slow and inefficient process, leading to incomplete test coverage. This research investigates the importance of web testing and compares the effectiveness of test automation tools and test design techniques. Two different web applications were used to evaluate and compare the tools' effectiveness.

Contents

Chapter 1

Introduction

1.1 Background of the Study

With the digitalization of the world, web applications have become an integral part of our daily lives. In 2018, it was recorded for the first time that 50% of people worldwide were online [1]. This number is expected to continue to rise as more people gain access to the internet. As a result, web applications have become an increasingly important way for businesses to interact with customers to provide services and sell products.

However, many clients wouldn't give you a second opportunity, according to a BCG report. When users face problems while visiting a website, it can have serious consequences for the business. 28% of customers claimed they stopped shopping on that website, 23% of customers claimed they stopped buying at that website, and 6% of customers stated they no longer make purchases at the company's bricks and mortar stores [2]. In such a case, the quality of the products offered to people must be of a high standard. To ensure the quality of web applications, they must be tested well.

In the last decades, there have been significant changes in web technologies. Most of the websites back in those days were static HTML pages, known as Web 1.0. With the emergence of Web 2.0, a social web emerged that aimed to make the web more responsive and interactive. Methodologies such as AJAX became central which allows websites to communicate with the browser behind the scenes and without requiring human interaction [3]. As a result, the complexity of modern web applications has increased, making it more challenging to guarantee their correctness.

1.2 Problem Statement

Software testing is a critical step in the software development process, and it is especially important in the context of web applications. Web applications are complex systems that should function perfectly on a wide range of browsers, operating systems, and devices, which makes them more sensitive to errors and bugs. Software testing helps to identify and fix defects, ensuring that the software meets its functional and non-functional requirements. Despite its importance, software testing is often overlooked or undervalued in the software development process, resulting in costly errors and failures. When working on a software development project, it should be known that errors and faults can occur at any stage of the life cycle. It's important to be aware that the cost of software failure might be very expensive. In case a bug is discovered early on, correcting it will be less expensive. Software testing at an early stage is crucial because it allows for the early detection and correction of errors and bugs in the software before the software product is released. This is especially important in the fast-growing web development world, where rapid deployment is critical to success.

For instance, the UK government first implemented a brand-new, complex system to manage the Child Support Agency's activities back in 2004. The system, known as CS2, was "improperly designed, terribly tested, and badly implemented," according to an internal memo that was leaked. It "had over 1,000 reported faults, of which 400 had no known solution," the agency claimed. It ended up costing an estimated £768 million, significantly more than the budgeted cost of £450 million [4].

Websites can lose users if they are poorly tested. In fact, a website that is not properly tested can lead to a poor user experience, which can result in users leaving the website and not returning. On 17 June 2002, the online betting website www.sportingindex.com experienced an embarrassing outage two days before England versus Brazil World Cup game. Even though the website was only offline for a day, it resulted in significant revenue loss and the loss of customers to rival betting companies [5].

1.3 Overview

Software testing is the practice of examining and validating that a software application performs as it is intended. Software testing is an essential part of the software development life cycle because it allows for the early detection and correction of bugs and errors before the software product is released. The dependability, security, and high performance of a fully tested software application provide time savings, cost-effectiveness, and customer pleasure.

Testing modern and complex web applications is challenging because of several factors such as supporting multiple devices, platforms, and browsers. Another significant challenge is managing the increasing complexity of web technologies, which includes the usage of dynamic web content and APIs. However, the benefits of software testing for web applications cannot be overstated. It can improve the user experience, reduce the risk of defects or security vulnerabilities, and increase the reliability and performance of the application.

There are a variety of testing methods that are commonly used for web applications, including functional testing, performance testing, and security testing. Each of these methods has a specific function and can help ensure the quality of the web application. Functional testing checks that the application meets the requirements and specifications, while performance testing evaluates the application's responsiveness, and speed. Security testing ensures that the application is secure and protected from potential threats.

The tools and techniques used to test web applications have also evolved to meet the changing needs of the industry, and it is important for organizations to stay up to date with these developments to ensure the quality of their web applications. In recent years, automation testing has gained popularity due to its ability to increase efficiency and reduce the risk of human error [6].

1.4 Scope and Objectives

This paper will outline the importance of testing web applications and to evaluate the effectiveness and efficiency of the newest test design techniques and tools for web applications. This research will focus on determining the most useful and important testing techniques for web applications.

The objectives of this research are to:

- Outline the importance of testing web applications and the benefits of using automated test design techniques and automation tools.
- The study will compare the efficiency and effectiveness of three different automation tools (Selenium, testRigor, and Harmony) for functional testing and understand how the test design of a web application affects the effectiveness of automated testing using the Action-state testing technique.
- Analyze the results of the comparison of testing techniques and tools, and document the most useful and effective methods for testing web applications.
- To direct users to more user-friendly automation tools and design techniques that can be easily adopted. Companies will be able to simplify their web testing process and deliver better quality products and services to their customers.

Chapter 2

Literature Review

2.1 Software Quality

Software quality refers to the degree to which a software product meets the needs and expectations of its users [7]. It contains a wide range of characteristics that can be used to measure the software product. According to ISTQB, these are functionality, reliability, usability, performance, maintainability, portability, security, and compatibility.

Software quality is way more important in the context of web applications since they must function perfectly across multiple platforms, browsers, and operating systems. Testing is one of the most important tasks in software development that can be used to ensure the quality of software, and this is especially true for web applications. Web testing is a technique used in software testing to check websites or web applications for potential bugs before publishing them [8].

- Testing is the process of evaluating a software product to identify any bugs and errors that may affect its quality.
- By testing web applications, teams can evaluate the software product against its requirements and ensure that it meets expectations.
- Testing is used to determine places of the software product that can be improved. For example, performance testing can be used to find and fix problems that might affect the website's loading speed.

- It can be used to measure software quality by using metrics such as code coverage and the number of bugs. These metrics can be used to follow the development of software and identify areas where improvements are needed.

So, testing can help to improve software quality over time by detecting and fixing bugs and errors. The user experience for web applications is improving as a result.

2.2 Testing in Agile

The agile software development technique has grown in popularity in recent years because it offers a flexible and iterative method for software development. Continuous testing is essential for identifying bugs early in the development cycle and ensuring the quality of produced software, according to the agile methodology [9].

A sprint is a short, time-limited period in the Agile methodology in which a Scrum team works to finish a specific amount of work. Sprints usually last one to four weeks long. The team works to finish assigned tasks during a sprint. The code that is new or updated in each sprint is tested by testers.

The sprints of Agile can be seen in Figure 2.1 [10].

Figure 2.1: Sprints in Agile

Testing is not an independent phase but rather a necessary phase in the development process. Small iterations and continuous testing should be used throughout the whole development cycle. The aim is to fulfill client needs while delivering high-quality software in an environmentally friendly way. Agile testing emphasizes communication and adaptability and includes collaboration between developers, testers, and stakeholders [11].

Agile testing offers continuous feedback at every stage of the software development lifecycle, allowing teams to make changes early and improve the quality of their work. Also, it enables teams to be adaptable and respond rapidly to customer feedback and changing requirements. Agile testing supports teams in delivering software more quickly by enabling incremental releases of features that have been tested and verified.

On the other hand, agile testing needs qualified testers that can work in an unstable environment that can be difficult due to continuous changes, which means that testers must be flexible and able to modify their technique [12].

2.3 Challenges of Web Application Testing

Website testing is a critical step in the software development cycle because it ensures the quality and functionality of a website before it is released to the public. However, the process of web application testing can be a complex task that can have several difficulties for testers to overcome.

One of the biggest difficulties in website testing is ensuring compatibility across multiple browsers, devices, and operating systems and ensuring that the website works perfectly in all of these environments. In the past, web developers had to deal only with Internet Explorer. With time, the number of browsers increased. Chrome, Firefox, and Safari are some common browsers that are used. In addition, there were only computers that developers were creating products for. With the increase in mobile device and tablet usage, it has become an even more complex challenge, as web applications must be optimized for smaller screens and different resolutions, and touch-based input.

Websites that have dynamic content, such as social media and e-commerce platforms, can be difficult to test due to the always-changing content. This can make

it difficult to create constant and repeatable tests, which are essential for reliable testing.

Many new technologies must be used in web application testing to cover all the web application's components. Asynchronous communication, untyped JavaScript, event handling, timing/latency, browser dependency, and many other factors can make web programs open to errors [13].

Users are more likely to delete a mobile app if it takes longer than three seconds to load, thus speed is a particular problem [14]. Often a web application gets too slow or crashes when the internet traffic increases suddenly. Performance testing is important to ensure that the application works quickly and efficiently. It helps to make sure that using the application does not affect the speed at which tasks are completed.

In addition, the testing of web applications is a crucial topic to manage because of the heterogeneity involved in the numerous languages used, execution environments, technologies, and operating systems [15]. This is because each of these factors can introduce different types of errors and make it difficult to create tests that are both reliable and comprehensive. For example, if a web application is built using a combination of front-end and back-end technologies, testers may need to use a variety of tools and frameworks to test each component effectively.

Despite these challenges, web application testing is an essential step in ensuring the overall quality and reliability of a web application.

2.4 Functional Testing

Functional testing is a type of software testing that is done to see if a system or software application is functioning in accordance with the requirements and expected behavior. A website's functionality is tested using a variety of testing criteria, including user interface, APIs, database, security, client and server, and basic website capabilities [16]. In every project, a document is created that includes functional or required specifications during the planning stage. They don't check the system's intermediate states while carrying out an action; they only check the action's outcome [17]. In short, it is a checklist of what the system should do from the viewpoint of the user.

The main goal of functional testing is to evaluate each software program's function by giving the right input and comparing the results to the functional specifications [18]. In order to make sure that the system performs as expected in all conditions, the tests typically include both positive and negative scenarios. By doing this, the testers can identify any defects or issues in the software's functionality.

Test cases are an essential part of functional testing. Test cases are created to verify that each function or feature of the software system works as expected [19]. The testers use the functional specifications as a reference to create test cases that cover all possible scenarios that a user may experience. Once the test cases are created, they are reviewed to ensure that they cover all possible scenarios and that there are no gaps or duplication. After the review process is done, the test cases are executed manually or through automated testing tools. The use of automated testing tools can accelerate the testing process and minimize the chances of human errors. During the execution, the test cases are run one by one, and the results are recorded. If the actual output matches the expected output, the test case is marked as passed. If there is a difference, the test case is marked as failed.

2.5 Maintenance in Testing

Maintenance testing is an important aspect of software development as it ensures that the application remains functional and relevant over time. In this context, it is essential to have tools that can efficiently adapt to changes in the source code. The efficiency of Selenium and Harmony tools is evaluated in maintenance testing by updating the source code of the Food Ordering web application and measuring the time and effort required to update the corresponding test cases in each tool. The TestRigor tool is not because it is not applicable to local machines. Below, the changes that are made in the source code and their solutions for both test automation tools are given:

Maintenance testing is different from testing new applications. While maintenance testing is performed after the software is released, new application testing is done during the pre-release phase of software development [20]. New application testing is performed when a new software application is being developed. This type of testing involves testing the system from scratch, identifying defects and bugs, and verifying that the application meets the specified requirements as expected.

Maintenance testing, on the other hand, is performed on an existing software application after it has been released into production.

Maintenance in test automation refers to the process of repairing and updating automated tests to ensure they remain up to date with code changes and continue to function correctly [21]. In test automation maintenance, test scripts are updated to reflect modifications to the application being tested. This involves modifying the scripts to adapt to changes in the user interface or underlying software. Test data must also be updated to guarantee that the tests are still relevant and efficient. Finally, it could be necessary to modify the test environment to meet updated requirements or application changes.

2.6 Test techniques

Test techniques are methods and approaches used by software testers to design and execute tests that evaluate the quality and functionality of software applications. The goal of using test techniques is to help detect software errors by providing a set of rules that help testers to perform tests from all aspects to verify that customer requirements and expectations are fully met. They are an important part of the software testing process and help to ensure that applications are tested properly and effectively. Test techniques provide consistency and repeatability, which allows for the possibility of reproducing tests and increasing the number of bugs found [22]. It saves human and time resources and minimizes the number of tests needed to effectively test the product. Additionally, it allows the tracking and tracing of test coverage, which is especially helpful for complex use cases with several actors, choices, and conditions [23]. In the following sections, two test techniques Action-state testing and Model-based testing are examined.

2.6.1 Model-Based Testing

Model-based testing (MBT) is a testing technique that involves using a model of the system under test to generate test cases and test inputs. The model can be used to generate many test cases automatically, which can save time and effort compared with manual test case creation. MBT is based on the idea that it is often easier

to specify the expected behavior of a system using a model rather than writing individual test cases manually.

The MBT involves several steps, including designing a test model, selecting test generation criteria, generating test cases, and executing the test cases [24]. One of the major advantages of model-based testing is that it helps to ensure coverage. All positive and negative paths and combinations are covered in the test cases [25].

Model-based testing can be seen as an efficient way to reduce the efforts and costs of testing [26]. A promising basis for the introduction of model-based testing approaches on a bigger scale is provided by the growing popularity of model-based development processes that lead to a stronger formalization of development artifacts [26]. While software development is in progress, model creation should also be a part of the development. It should be included in the product design during the requirements specification stage.

MBT can be used for both manual and automated testing. In automated testing, the model is used to generate test inputs and expected outputs, which are then used in an automated test execution tool. In manual testing, the model can be used to generate a set of test cases that a human tester can execute manually. In addition, since most model-based testing processes are either fully automated or semi-automated, it promises to significantly reduce the cost of generating and maintaining test artifacts [27].

2.6.2 Action-state Testing

Action-state testing is a software testing technique that models a system as a series of actions and states to test its behavior. It aims to identify potential defects in software systems by analyzing the relationship between actions and states. It is designed to improve code quality, reduce bugs, and decrease the total software development life cycle costs [28].

It can be used to test systems with user interactions, such as web applications, by modeling user actions and system responses. There are several steps in Action-state testing. A user action, a system response, and a test state all make up one step [29]. It breaks down the software into several actions and states and then creates test cases that are covering all possible combinations of actions and states.

Actions in action-state testing are user interactions with the software, such as entering data or clicking buttons. States refer to the different situations of the software, such as the form submitted, or the user logged in. Software testers can detect potential bugs in the software, such as unexpected behavior or wrong error messages, by examining the relationship between actions and states.

The system is in a particular state at any given time, and the transitions between states are triggered by actions performed on the system. It is a unified technique that combines elements of both use case testing and state transition testing [30]. It includes the concept of states, which are not present in use case testing and includes system outputs, which are not included in state transition testing. Action-state testing is a useful tool for finding even the trickiest bugs and ensuring that a system behaves correctly in different states, with smooth and predictable state transitions.

A simple algorithm is used to build abstract steps one at a time in the step-by-step technique. The model is built gradually from the start rather than all at once. Due to the incremental selection and implementation of features in agile, it is typically not possible to create a complete model for the entire program. Action-state testing is in accordance with this idea [31].

Creating test cases one by one is not an efficient solution. This technique also allows for the creation of abstract test cases that can be automatically generated from the Action-state model, like use case models, but using abstract test steps instead of human or system actions.

From the model, abstract test cases should be created. The tool Harmony is very helpful for the next steps. It offers the next steps according to the existing abstract test cases. It highlights the steps which should be extended with new child steps. With this help, the time spent on creating the model is reduced. In addition, it increases the coverage and prevents the repetition of test cases.

2.7 Test Automation and Test Design Automation

Testing can be done in two ways which are manual or automated. Test automation is the use of software or tools to automatically execute test cases to test the functionality of a software application. It is an efficient and cost-effective way to perform testing, as it allows testers to save time and effort by automating repetitive

or time-consuming testing tasks. Test automation can also reduce the risk of human error and improve the consistency and reliability of testing by executing test cases consistently and accurately.

One of the main benefits of test automation is the ability to run many test cases in a short period of time. Test automation can also be used to test the same functionality across multiple platforms or devices, helping to ensure that the application behaves consistently across different environments. The Selenium tool is a popular example of a test automation tool that is widely used in the software testing industry. It is an open-source tool that can be used to automate web browsers, allowing testers to create and execute automated test cases for web applications. In this research, the Selenium tool will be used to demonstrate the capabilities and benefits of test automation.

On the other hand, test design automation is a software testing technique that involves using tools or techniques to automatically generate test cases or test inputs for testing software. It introduces automation into the testing process at an earlier stage and can be used to create both manual and automated test cases [32]. Test design automation can help testers quickly and efficiently create many test cases that cover a wide range of scenarios and input combinations. This can help to increase test coverage and reduce the number of bugs in the software, resulting in a higher quality product.

It can be particularly useful in cases where it is difficult or time-consuming to create test cases manually, such as when testing complex or highly dynamic systems.It can also be combined with the development and test automation in the same cycle, allowing for a more efficient testing process. Harmony and testRigor are some of examples of test design automation where the tests are created from models. As the complexity of applications increases and time to market decreases, the need for test automation becomes more pressing. However, relying on test automation alone is not sufficient. Test design automation is also necessary to ensure that the testing process is sufficient [33]. Test automation is worthless without proper test design since you will repeatedly run test cases without finding any issues in your software [29].

2.8 Test Design and Automated Test Execution Tools

Test design tools are tools that are used to design and create test cases for testing software. These tools allow testers to specify the steps and inputs for a test, as well as the expected output or result. Test design tools can be used to create manual test cases, which are executed by a human tester, or automated test cases, which are executed by a computer. Frequently, test design tools are built and designed to interact with specific requirements and tools, such as specific requirements management tools [34]. The types of tests required to achieve the specific goal level of coverage and confidence in the system can be determined with the use of test design tools.

As part of the software testing technique known as automation testing, a test case suite is executed with automated testing software tools. At all test levels, automated tests are executed, and outcomes are checked using test execution tools. Automation testing tools are programs that are designed to verify function and/or non-functional requirements via automated test scripts [35].

Using a test execution tool specifically achieves one or more of the following goals:

- To execute more tests.
- To execute more tests at a lower cost and in less time.
- To execute the same test across several environments.
- To execute the tests more repeatable and any time it is wanted.
- To increase the test coverage.

Automated test execution tools typically include features such as scheduling, logging, and reporting to help testers manage and analyze the results of their tests.

The number one problem in test automation is listed as buying the wrong tool because, no matter what kind of process or organization you have, if the tool is not a suitable technical or business match, it cannot be used effectively [36]. In addition, knowing which test automation tools to use and how to use them can increase the coverage in your testing efforts [34]. Because of these reasons, there are multiple test design and automated test execution tools that are used in this paper which are:

2.8.1 Selenium WebDriver

Selenium is an open-source umbrella project for a range of tools and libraries aimed at supporting browser automation [37]. One of its advantages is that it helps users to run the tests automatically in a short time that would take hours manually.

Another advantage of Selenium is its flexibility and compatibility. It can be used together with a lot of programming languages such as Java, Python, and JavaScript. Windows, macOS, and Linux are some of the platforms that are supporting the use of Selenium.

The client (browser) and server (Selenium server) communicate using the WebDriver Wire Protocol, with the server acting as an intermediary between the client and the browser which can be seen in the Figure 2.2 Selenium WebDriver Framework Architecture [38]. WebDriver is designed as a simple and more concise programming interface. It is a compact object-oriented API [39]. Selenium is compatible with most browsers such as Chrome and Firefox. Each browser is supported by a specific WebDriver implementation, also known as a driver, which is an API that determines how the web browsers should behave. The driver manages Selenium's communication with the browser and is responsible for task assignments to the browser. The WebDriver communicates directly with the browser without the use of any middleware. This direct communication allows for more efficient and accurate testing because there is no need for any additional software to translate the commands.

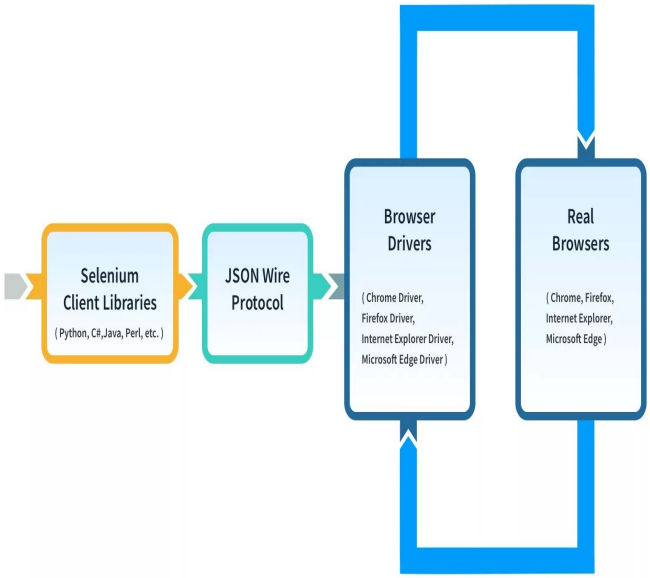

Figure 2.2: Selenium WebDriver Framework Architecture

Another advantage of Selenium is that it has a large and active community of users and developers who contribute to the development of the tool. This means that there are many resources available such as documentation, tutorials, and forums where users can get help and support [40].

It also supports cloud testing platforms such as Sauce Labs and BrowserStack, which allow users to run their tests on multiple devices and browsers without having to set up their own testing infrastructure [41].

The user should only define a line of code according to the browser that wanted to be used. For Chrome browser: WebDriver driver = new ChromeDriver();

The test cases can be run solo or multiple. The user can choose which method they want to run. In addition, in each test execution, a new browser will be started which shows each step of the actions (filling input, clicking button) that are made. On the other hand, it has some limitations. One of its main limitations is maintenance. It is a maintenance-heavy framework, which can be time-consuming and costly [42]. Additionally, there is a need for programming knowledge. This means that if there are no programming skills, it might be way more challenging to use Selenium effectively.

2.8.2 testRigor

testRigor is an AI-driven test automation software [43] that allows users to create and execute automated test cases. It can be reached through its website without requiring any installations. Registration is needed on the website to use testRigor. One of the main features of testRigor is that coding skills are not required to write the test cases. In other words, plain English is used to implement the test cases which makes it easier to understand and implement for non-technical users as well.

It is compatible with most popular browsers such as Chrome and Firefox. It has support for testing desktop and mobile browsers including Safari from MacOS and IOS. This means that users can easily test their web applications on different platforms and devices.

In addition to these features, testRigor offers reporting and analysis features as well. During each test execution, the software saves images of each step of the actions which are saved inside the test cases. This makes it easy for users to review and analyze their test results.

testRigor also offers users to customize their test execution by setting parameters such as test duration, test load, and geographical location of the test environment. This allows users to model real-world testing scenarios and ensure that their application is working properly in different situations. testRigor can easily scale to meet the needs of large organizations with complex testing requirements. The platform supports distributed testing, which means that users can run their tests on multiple machines in parallel and get the results quicker [44].

2.8.3 Harmony

Harmony is a tool that automates tests without requiring coding. It uses the Cypress integration to test web applications quickly and efficiently. The code itself can be written on its website but to execute it, Cypress should be installed. Registration is needed on the website to use Harmony. Test cases should be implemented with the Action-state testing model. Two modeling processes are included in Harmony to implement the Action-state testing model which is an abstract model and a concrete model.

The first step in using Harmony is creating some test cases to make the abstract model. After that, Harmony advises adding more steps that are missing in the test

cases to increase the coverage. In this approach, the test cases that are generated are more reliable, and tricky bugs can be detected easily. Harmony generates JavaScript code for the Cypress Test runner which allows users to easily execute their test cases [28]. Harmony will create the concrete model just by sequentially running the abstract tests. It is compatible only with the Chrome browser. With Harmony, users can run test cases solo or in multiple executions. Additionally, each execution keeps the website next to it with the steps of action which makes it helpful for analyzing and understanding the test results.

In addition to these features, Harmony is a powerful tool for automating software testing, and it offers several features that make it useful for a variety of testing needs. One key advantage of Harmony is that it offers a flexible and customizable testing environment, allowing users to create tests that are specific to their needs. This is possible because Harmony uses the Action-state testing model, which is a testing model that focuses on the interactions between a software application and its environment.

2.9 Related Work

In this section, papers that are related to test automation in web application testing are presented and compared.

2.9.1 Automated Test Code Generation and Execution System for Web

The area of interest for both my thesis work and the comparison paper [45] is the automation of software testing. Both articles recognize the value of software testing in the software development process as well as the necessity of automating testing to reduce costs and save time.

The goal of the proposed thesis study is to create an automated web testing tool that can, without the need for technical knowledge, translate functional instructions written in natural language in software test documents into code segments for automated testing. Additionally, the tool offers beneficial functions like reporting uncovered components in the system being tested and mutant generation for evaluating the efficiency of the test suite.

In my research, there are some comparisons made between the Selenium, Harmony, and testrigor tools based on factors including user-friendliness, usability, the technical knowledge required, amount of test cases developed, and the number of bugs discovered. One of the tools also uses a test design technique Action-state testing to determine how well it finds further bugs.

While both studies aim to improve web testing automation and evaluate the effectiveness of various tools, they differ by using different approaches and focuses. In contrast to the proposed thesis study, which focuses on creating a tool that does not require technical knowledge, my paper compares the performance of various automation tools on various web applications.

2.9.2 The Applicability of Automated Testing Frameworks for Mobile Application Testing: A Systematic Literature Review

This study [46] focuses on a systematic review of the literature on mobile automation testing frameworks. The authors present a taxonomy of mobile automation testing research to organize and combine research works. They also examine the main test concerns in mobile test automation. Additionally, they review the literature that is currently available on test automation frameworks for mobile application testing, determine test difficulties, and suggest an enhanced mobile automation testing framework architecture that resolves these difficulties. My thesis, on the other hand, compares the performance of three different test automation tools on two different web applications, applying functional and maintenance testing to determine which tool is most effective. The test automation frameworks' various metrics are compared. Furthermore, how test design techniques affect bug finding is explored.

The provided paper follows a more theoretical approach and offers an extensive review of the available research on mobile automation testing frameworks, in contrast to my thesis, which focuses on the comparative analysis of certain tools. However, the improvement of test automation effectiveness is a common objective of both works.

2.9.3 The Impacts of Test Automation on Software's Cost, Quality and Time to Market

The presented paper [47] examines the importance of software testing and the advantages of test automation for enhancing software product quality, cost, and time to market. Software testing has been divided into two categories by the authors: manual testing and automated testing. They have explained that automated testing involves the use of standardized software solutions to control the execution of test cases on the Software Under Test, while manual testing involves human testers running the program and examining its behavior.

The authors have also drawn attention to the difficulties that come with test automation, such as the necessity to carefully consider which features need to be tested automatically and the high initial expenses involved with software feature analysis, scripting, tool purchase, and training. They have highlighted the significance of performing an exact Return on Investment analysis of test automation before moving further with it.

The research tried to estimate the impacts of test automation on cost, quality, and time to market software products and offered empirical annotations from other studies on test automation. To determine the effect of test automation on three different software products, the authors conducted tests and created mathematical models. They have discovered that test automation, despite the high implementation and maintenance costs, may produce amazing long-term returns.

My thesis, on the other hand, focuses on a direct comparison of three test automation tools. The features, potential, and restrictions of each tool are investigated. Their efficiency in lowering costs, accelerating time to market, and improving software quality is evaluated. The goal of comparing these technologies is to offer useful advice to software developers and testing experts who are thinking about implementing test automation.

2.9.4 A Critical Analysis of Software Testing Tools

This thesis and the paper [48] that was provided place a strong emphasis on software testing and the advantages of test automation in improving the quality and efficiency of software development. The importance of choosing the right test

automation tool based on a range of factors, including platform compatibility, price, and necessary capabilities, is acknowledged in both papers.

In contrast to the offered article's evaluation of various widely used tools, including Selenium, TestComplete, Ranorex, OpenScript, and Janova, my thesis work particularly analyzes three alternative test automation technologies. While the provided paper offers a more comprehensive overview of the most frequently used tools, my paper focuses more on the specific characteristics and advantages of the three tools that are being compared.

Both papers reach the same conclusion: There is no one ideal tool for software testing, and choosing the best tool for a particular project requires making decisions. The offered paper provides a summary of the chosen tools' characteristics, while my paper provides a complete comparison of the three test automation tools' strengths and weaknesses.

Chapter 3

Methodology

In today's competitive environment, web testing is essential to ensuring that companies can deliver high-quality products and services to their customers. Automated test design techniques and tools in web testing have the potential to provide great benefits to companies. By reducing the number of manual tasks in web testing, automated testing can create cost-effective solutions. Automated web testing can also reduce the possibility of errors caused by manual tasks that result in better quality products and services.

However, some of the existing web test automation solutions require a technical background, which can limit the usability of the tools for users who do not have a development background. This highlights the importance of selecting user-friendly automation tools and design techniques that can be easily adopted by a greater number of users.

In this study, the efficiency of three different automation tools Selenium, testRigor, and Harmony are used to identify the most effective tool for functional testing. Also, the Action-state testing technique is used to understand how the design of a web application affects the effectiveness of automated testing. Additionally, two different web applications are used in testing to ensure that the results are applicable to a range of real-world scenarios.

Comparative analysis is the method used in this study to analyze how well the three automation tools perform in comparison to each other. The main method for gathering data is experimental testing, where test scripts are used to automate the testing of two different web applications. The collected data includes the implementation time, test execution time, number of test cases and number of failing test

cases.

In order to compare the effectiveness, efficiency, and user-friendliness of the three automation systems, the data gathered will be analyzed. The goal is to determine the most efficient functional testing tool and to provide recommendations for the use of test automation tools and design techniques in web testing.

Chapter 4

Results

In this section, the findings of the study are presented and discussed in detail, focusing on the performance of three test automation tools that are Selenium, testRigor, and Harmony. The purpose of this evaluation is to identify the most effective tool for functional testing and analyze how the test design of a web application affects the effectiveness of automated testing.

To achieve these goals, two different web applications were chosen to use, and the Action-state testing technique was utilized to determine if the test design technique had any impact on how well each tool performed in terms of coverage and accuracy. Each tool's usability and user-friendliness were also evaluated, with a focus on how simple it was to use them and integrate them into current testing workflows.

A variety of performance measures were used in the evaluation, including the implementation time, execution time, number of tests executed, and the number of errors found. Moreover, this section covers the maintenance aspect of testing, as updating the source code may affect the test case results. To examine the different advantages and disadvantages of the three tools, these metrics were compared across all three tools.

Three different tools are used to compare the effectiveness and usability of the automated test design techniques and tools. During the evaluations, two web applications are used which are Online Pizza Ordering and Food Ordering System.

4.1 Subject Applications

4.1.1 Online Pizza Ordering

Online Pizza Ordering is a typical example of a dynamic web application that allows users to easily order a pizza online. It serves as an example of a manually created web application. It is formed, like many modern web applications, of an HTML web page with dynamic content and a back-end server layer that handles requests made by the web page and generates and provides the necessary responses to the web page. The web application is very simplified compared to many enterprise web applications, it allows users to perform a set of basic user tasks such as ordering pizza, coke, and beer.

It is a test design web application that is not used as a real online pizza ordering web application. It's used for software design and testing.

The functionalities of the web application are the followings:

- The user can increase the amount of any item one by one.
- The user can delete (set to 0) the ordered number of each item.
- A free beverage is offered to the customer the first time the total price, excluding beverages (coke and beer), exceeds 45 euros. The user may add one free coke or beer to the beverages they have already ordered.
- When a free beverage is requested, the words "Free Coke" or "Free Beer" are displayed.
- It is not possible to switch the free beverage from coke to beer or vice versa.
- The free beverage will no longer be offered if the customer removes food items and the price drops below 45 euros.
- The free beverage can be removed by the customer. If the customer adds the same beverage again while the non-beverage price condition is still true, it stays free.
- The customer can remove any item. Once there is enough food, any free beverage can be chosen.

4.1.2 Food Ordering System

Food Ordering System is a web-based application that is developed with the Python-based web framework Django for the back end. For the front end, HTML and JavaScript are used. It is not an existing web application. It must be run from the local machine.

This Food Ordering System helps the food store's customers to easily order their desired foods. The project has two different user interfaces which are one for the admin and one for customers. These two different user roles provide different functionalities and features to both user types.

The application requires the users to log in to gain access to features and functionalities. Without a login, users cannot see and order any food. For this, there are registration and login pages. After the successful registration or login, customers can start to order different foods that the shop is offering. Users will be redirected to the food categories. The categories on this web application are pizza, burger, salad, pasta, and platter. The customers can choose different sizes for each food which are small and large. In addition, customers can select up to three different toppings for each ordered pizza. After the food selection is done, users can go to the cart to either checkout or delete the orders.

On the other hand, there is an admin page where the admin could manage food categories, and add, edit, or delete items as needed. The admin can also access the customer web application with the same functionalities that users have. They have access to an extra page which is the Order List Page the admin can mark the pending orders to deliver.

4.2 Test Case Creation

For Online Pizza Ordering and Food ordering System, two different test scenarios are created. With these test scenarios, test cases are created. During the test creation process, various tools are used through testing of the website which are Selenium, testRigor, and Harmony. Additionally, in Harmony, the test technique Action-state testing is used to generate additional test cases and covers a wider range of scenarios. By utilizing these tools and techniques.

4.2.1 Test Case Creation for Online Pizza Ordering

couldSince this website only has one HTML page, the limited scope of its functionalities limits the variety of test cases that can be created. The test scenario was implemented according to the given functionalities of the website. A variety of test cases can be created to thoroughly test the website with the specified functionalities.

The initial test cases focused on increasing the number of each item and making sure that the right total price is calculated, considering the free beverage when applicable. The next test cases involved deleting an ordered item and ensuring that the total price is updated accordingly. Also, it is important to test the specific conditions under which the free beverage is provided, including adding and removing products from the order to ensure that the same free beverage is correctly added and removed in each case. Additionally, it is important to test the restriction on changing the complimentary beverage's type as well. Ensuring that the beverage is still free if it is introduced again while the non-beverage price condition is still valid. Finally, testing the ability to remove any item and the ability to select any unrestricted beverage once there is enough food.

The test cases are created based on this test scenario by using different pizza and beverage inputs.

4.2.2 Test Case Creation for Food Ordering System

Several test cases are constructed to ensure that the website is functioning properly and has all the functions listed. The test cases include a range of functionalities, including login, logout, navigation, authentication, adding various foods with different sizes, checkout, and deleting orders.

The first test case involves checking that a user can successfully log in with the right credentials and log out. The following test cases verify that unauthorized users cannot access specific pages of the website to evaluate the authentication functionality. The functionality which is tested the most is adding various foods. In the test cases, the cart is added with a variety of foods in a range of sizes to make sure that the pricing and total amounts are calculated correctly. The checkout procedure is also tested by checking that the user can complete the order and that the right information is provided. Testing the ability to delete orders from the cart is also necessary. The admin functionality which includes marking the orders as delivered

is tested as well. Finally, checking the website's navigation is also a significant component of the test scenario. This can be performed by clicking on buttons, visiting every link and page on the website, and making sure that the user is taken to the relevant page and that all functionalities work as expected.

By testing each of these functionalities, the website can be ensured to work as intended and provide a positive user experience.

4.3 Results of Test Case Executions

4.3.1 Test Case Execution Results of Online Pizza Ordering

Functional test cases are tested with Selenium and testRigor tools. The test cases are first implemented on Selenium and then the same is implemented on testRigor. It took 1 hour to create the test scenario which was used in Selenium and testRigor. It took 1.5 hours to implement them on Selenium. For testRigor, it took 2 hours for the implementation which also includes the learning of the tool that is 1 hour.

During the testing of the web application, some bugs were discovered. In total, there are 18 passing test cases and 8 failing test cases.

After this, to have better test coverage, the Action-state testing method was used for test case implementation. The Harmony tool was used for this. With the Action-state testing method, out of 50 test cases; 35 passed and 15 failed. It took 1 hour to learn the Action-state testing and Harmony tool. The Action-state testing took 2 hours to complete.

Below, a small part of the state transition graph of the Action-state model is given which is automatically created by the Harmony tool. The example is as follows: To place an order on the website, the user should log in, add salads and platters to their cart, and then either proceed to checkout or remove the items from the order. If the order has been checked out, the admin has the option to mark it as delivered. However, if the order has been deleted, it is considered canceled.

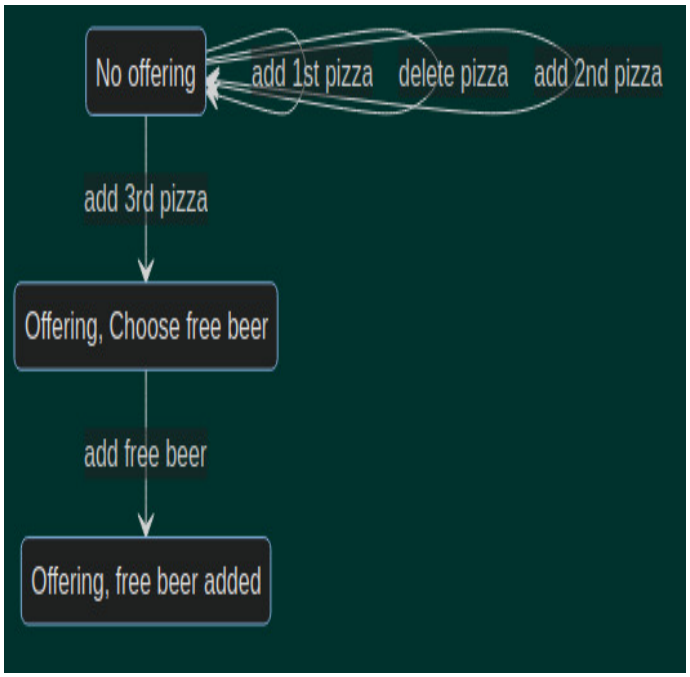

Figure 4.1: Action-state transition graph for Online Pizza Ordering

4.3.2 Test Case Execution Results of Food ordering System

The functional test cases were implemented using Selenium tools like the first web application. Initially, the test cases were designed and implemented on Selenium, which took approximately 2 hours. During the testing of the web application, a total of 61 test cases were executed, out of which 51 passed and 10 failed, uncovering several bugs.

After Selenium, the same test cases were supposed to be tried on testRigor but an error was encountered. However, an error was encountered when trying to connect to the Food Ordering System web application running on the local machine. It was found that testRigor does not support the testing of local machine applications that are not exposed to the internet. Due to this limitation, the Food Ordering System could not be tested on testRigor.

To see if using a test design technique is effective, the Action-state testing technique was utilized, and the Harmony tool was used for test case implementation. It took about 1 hour to implement the test cases. Out of the 43 test cases executed, 26 passed, and 17 failed, revealing additional defects in the web application that were not discovered from the other two tools.

Below, a small part of the state transition graph of the Action-state model is given which is automatically created by the Harmony tool. The example is as follows: To

place an order on the website, the user should login, add salads and platters to their cart, and then either proceed to checkout or remove the items from the order. If the order has been checked out, the admin has the option to mark it as delivered. However, if the order has been deleted, it is considered as canceled.

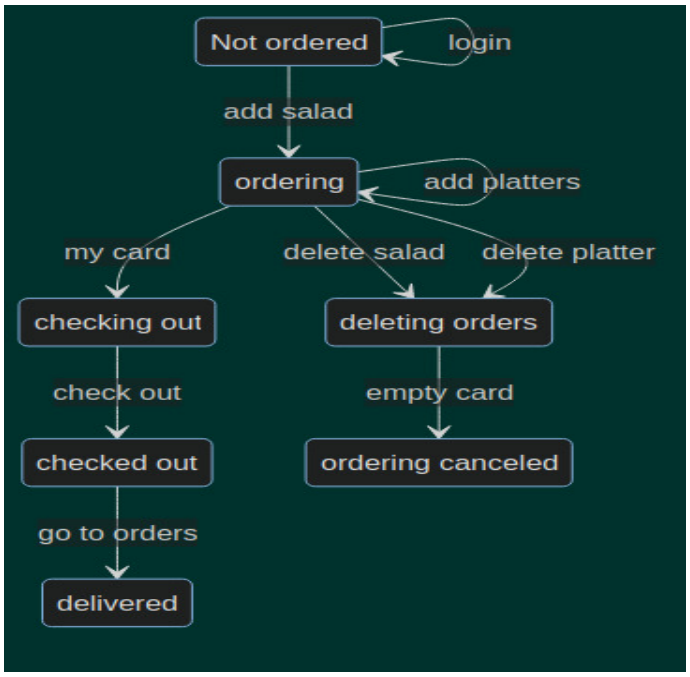

Figure 4.2: Action-state transition graph for Food Ordering System

4.4 Comparisons of the Test Automation Tools

To perform comprehensive testing of the website's functionalities, three different test automation tools were used which are Selenium, testRigor, and Harmony. These tools were selected based on their effectiveness and efficiency in performing functional testing. There were obvious differences in terms of their usability, adaptability, and capability. To find the tool that will best meet testing needs of the tested web applications, the advantages and disadvantages of the tools will be outlined.

Features	Selenium	testRigor	Harmony
Free/open source	Free and open source	Has free option. Tests results are public.	Free
Platform	Automation for web browsers	Cloud-based test automation tool	Cloud-based test automation tool
Installation	Used programming language installation is needed	Not needed	Cypress and Node.js installation is needed
Compatibility	Is compatible with most of the browsers	Is compatible with most of the browsers	Is compatible only with the Chrome browser
Technical knowledge	Programming language knowledge is needed	Plain English knowledge	Action-state testing knowledge is needed
Difficulty for learning	Hardest to learn between the mentioned three tools	Easiest to learn between the three tools	Harder than testRigor, easier than Selenium
Built-in reporting and Analytics	Does not provide	Provides	Provides

Table 4.1: Features of the test automation tools

The implementation of each tool is compared as well.

The following three figures below were implemented for Online Pizza Ordering. They show a single test case that is implemented using three different tools which are Selenium, testRigor, and Harmony. The test case starts by adding coke to an order, followed by adding three pizzas to the same order. The expected outcome is that the coke should be free which resulted in a total price of 48 instead of 45. The test case doesn't work as expected because the coke is not returned for free, which means the total price stays at 48 instead of 45.

```
1
2   @Test
3   public void firstBug() throws InterruptedException {
4         MainPage mainPage = new MainPage(this.driver);
5         mainPage.cokeIncrease();
6         mainPage.pizzaPolloIncrease();
7         mainPage.pizzaPolloIncrease();
8         mainPage.pizzaPolloIncrease();
```

```
9
10          System.out.println(mainPage.getBodyText());
11
12          Assert.assertTrue(mainPage.getBodyText().contains("Total
                price: 45"));
13  }
```

Code 4.1: Selenium Source Code of Online Pizza Ordering

```
1
2 Click on the 7th "+"
3 Click on the 3rd "+"
4 Click on the 3rd "+"
5 Click on the 3rd "+"
6 Check that page contains "Total price: 45"
```

Code 4.2: testRigor Source Code of Online Pizza Ordering

```
1     add coke => 1 coke in the cart STATE No offering, beverage
          added
2          add pizza => 1 pizza and 1 coke in the cart STATE No
               offering, beverage added
3               add pizza => 2 pizza and 1 coke in the cart STATE No
                    offering, beverage added
4                   add pizza => 3 pizza and free 1 coke in the cart
                        STATE Offering, coke converted
```

Code 4.3: Harmony Source Code for Online Pizza Ordering

Below, the test cases for the Food Ordering System can be seen as implemented in each of the tools: Selenium, and Harmony. It could not be implemented for testRigor because it does not support web applications running on local machines. The test cases consist of a series of steps that include logging in, navigating to the pizza page, ordering a pizza, going to the cart page, and deleting the pizza.

```
1
2   @Test
3   public void deleteOrder() throws InterruptedException {
4         MainPage main = new MainPage(this.driver);
5         DashboardPage dashBoardPage = login();
6
7         Food food = main.openFood();
8         food.orderPizza();
```

```
9
10          main.goToCart();
11          main.delete();
12          driver.switchTo().alert().accept();
13
14          Assert.assertTrue(dashBoardPage.getBodyText().contains("0")
                ;
15      }
```

Code 4.4: Selenium Source Code of Food Ordering System

```
1
2 login => Logged in STATE Not ordered
3     add pizza => 1 pizza added STATE ordering
4         delete pizza => empty card  STATE checked out
5             check count => the count is 0 STATE ordering
```

Code 4.5: Harmony Source Code of Food Ordering System

4.5 Maintenance Test Results

Maintenance testing is an important aspect of software development as it ensures that the application remains functional and relevant over time. In this context, it is essential to have tools that can efficiently adapt to changes in the source code. The efficiency of Selenium and Harmony tools is evaluated in maintenance testing by updating the source code of the Food Ordering web application and measuring the time and effort required to update the corresponding test cases in each tool. The maintenance testing results are as follows:

- Updating a food price: Just need to update the total price for Selenium and Harmony.
- Deleting a food option from the menu:
  - Selenium: Either replace it with another option (which requires a new method and XPath) or delete the test cases that have the deleted option
  - Harmony: It cannot see the deleted option. It asks the user to select a new option from the menu, and then just requires the total price to be updated

- Updating an element ID:
  - Selenium: It requires adding a new XPath to the code
  - Harmony: It gives a warning when executing the code that it can't find the element and directs the tester to choose the correct element.
- Change in the location of a button:
  - Selenium: The XPath of the button needs to be updated to match its new location.
  - Harmony: The action that corresponds to the button needs to be updated to match its new location.
- Name change in a food:
  - Selenium: Update the text check where the tester verifies if the order contains the ordered food.
  - Harmony: Update the changed food name in the action.

Both Harmony and Selenium have their strengths and weaknesses when it comes to maintenance testing. In scenarios where there is a need to update food prices, change the location of a button, or change the name of a food item, both tools perform equally well. However, when it comes to deleting items from the menu, Harmony's ability to recognize and prompt the user to select the correct option makes it more convenient to use. On the other hand, when there is a change to an element ID, Selenium's need to add a new XPath makes it a more time-consuming process.

4.6 Evaluation

The results of the testing process give important information about the quality of the web application under evaluation. These results must be evaluated to figure out whether the application fulfills the expected requirements. This section will analyze the results of the web testing process using three different test automation tools: Selenium, testRigor, and Harmony. The goal is to evaluate the effectiveness and efficiency of each tool and to present an informed suggestion for the selection of the most suitable tool for future web testing tasks.

The evaluations of the three tools that were used to test two web applications are compared in the following two tables below:

Comparisons	**Selenium**	**testRigor**	**Harmony**
Implementation time	1.5 hours	1 hours	2 hours
Number of test cases	18	18	50
Number of failing test cases	8	8	15
Execution time	2:45 mins	18:30 mins	2:47 mins

Table 4.2: Comparisons of the results for Online Pizza Ordering

In the Online Pizza Ordering web application, for Selenium and testRigor tools, the same test scenario is used. In this case, comparing the number of test cases and the number of failing test cases comparison is not valid because they are the same. When it comes to implementation time, it seems that testRigor is the most efficient option because it took the least amount of time but, Harmony has almost three times more test cases and only takes twice as long to implement. In addition, Harmony also found more failures compared to other tools, which was made possible by the Action-state testing technique. Lastly, testRigor was inefficient in execution time. Although Selenium and Harmony have almost the same execution time, Harmony's execution time was the most efficient one because it has more test cases.

Comparisons	**Selenium**	**Harmony**
Implementation time	2 hours	1 hours
Number of test cases	61	43
Number of failing test cases	10	17
Execution time	8:37 mins	4:49 mins

Table 4.3: Comparisons of the results for Food Ordering System

On the other hand, for the Food Ordering System, the comparisons were made for the two tools Selenium and Harmony. The reason it took 2 hours for Selenium is because of the need for a lot of classes and methods. The Food Ordering System is a more complex web application than Online Pizza Ordering. Web applications that are more complex require more classes and methods. Even though Selenium has

more test cases, the number of failed test cases is more for the Harmony tool. The reason behind this is that Harmony prohibits having test cases that are repeated. Moreover, the Action-state testing technique recommends users add new steps in Harmony that are not covered. Therefore, the number of failures is more for Harmony than for Selenium. Lastly, Harmony's execution time was more efficient than Selenium's.

When comparing the test cases implemented in each tool, it is obvious that Selenium's implementation is significantly longer than that of Harmony. This is mainly because Selenium requires more code to be written for each test case. Specifically, in Selenium, there are separate classes defined such as MainPage and Food, each with its own set of methods that need to be defined and implemented.

In contrast, Harmony requires less code to be written. Harmony is a codeless tool that allows users to create test cases by simply dragging and dropping predefined elements. This makes test case implementation significantly faster and more efficient than in Selenium.

Despite the longer implementation time in Selenium, it is worth noting that the tool does offer a wide range of customization options that may be useful for more complex testing scenarios. In addition, the use of classes and methods in Selenium allows for greater code reusability, which can be beneficial in larger projects.

These results show the importance of choosing the appropriate tool for the task. While Selenium and testRigor were able to find bugs in the web application, the Action-state testing approach with Harmony produced more test cases that passed and had higher test coverage.

This demonstrates the importance of careful investigation and evaluation of several testing methods and methodologies to achieve the best results. The Harmony tool was more effective than Selenium and testRigor on the web applications that were used. The reasons for this are the followings:

- Test design and model are not necessary for Selenium and testRigor. Everything is up to the tester who will create the test cases. The tester just thinks about what and why to test. On the other hand, to use the Harmony tool, an Action-state testing model is needed. With this model, the tool is advising new test cases to add which is increasing the coverage.

- Harmony allows users to test the state of the web application after the action, while Selenium is mostly focused on automating UI interactions. This can help to find bugs with the web application's state that is not visible on the UI.

- There is a huge difference in test case repetition. Harmony is designed to minimize repetition in testing. As such, it provides warnings to the user if it detects any repetition in the test cases. This helps to ensure that tests are efficient. It minimizes the chances of missing important scenarios. On the other hand, Selenium does not have any built-in library to detect test case repetition. This can lead to longer and potentially redundant test cases, which can be time-consuming to create and maintain. While it is still possible to create efficient test cases with Selenium, it requires more attention and discipline from the tester to ensure that repetition is kept to a minimum.

- Harmony's approach suggesting additional steps in test cases to improve coverage is a notable advantage over other tools. By providing suggestions for additional steps, Harmony can identify potential gaps in test coverage that other tools may miss. With this strategy, test cases are more detailed and can catch a greater number of possible mistakes.

- Compared to Selenium, implementing Harmony took less time. One of the reasons behind this is that Harmony is a codeless tool. Unlike Selenium, there is no need to manually find XPath for element locations when implementing test cases with Harmony. Moreover, Selenium often requires implementing a significant number of custom methods to perform specific functionalities, which can be time-consuming.

- Although testRigor is a codeless testing solution like Harmony that enables users to automate tests without having a deep understanding of programming, one of the biggest disadvantages of TestRigor is that the tool cannot be used on local machines, which can be a significant limitation for web applications that are not deployed on the internet or that need to be tested before being released.

- Harmony has been found to be more time-efficient and convenient than Selenium based on maintenance testing done in the Food Ordering System.

Overall, the effectiveness of test automation efforts can be significantly increased by using a well-defined test design technique, such as Action-state testing. Two small-scale web applications were evaluated using various test automation tools. In comparison to Selenium and testRigor, the results showed that Harmony was the most effective tool, providing functional and maintenance testing that was less time-consuming and more convenient. However, it should be noted that Harmony might not be the best option for large-scale projects due to its limited customization options. Therefore, while selecting a test automation tool, the environment and requirements of the current project should always be taken into consideration.

Chapter 5

Conclusion and Future Work

In conclusion, high-quality websites are essential for the success of any company or group that wants to stay competitive in the current digital era. It is impossible to overestimate the value of testing because it is essential to ensure that websites function as intended, are reliable, and meet users' needs.

Manual testing can be time-consuming and open to errors, which can cause releases to be delayed and consumers to become dissatisfied. Test automation can help with these problems by offering a more reliable and consistent testing approach. It has become an important part of the software development life cycle since it may significantly save the time and effort needed for testing, making the process more efficient and effective.

In this project, three test automation tools and one test design technique were evaluated for their effectiveness in testing small-scale web applications. The web application called Online Pizza Ordering has been made public and is available online. This was tested by the three test automation tools. On the other hand, the other web application called Food Ordering System is not published on the internet. It must be executed on the local machine. This was tested by two of the test automation tools because testRigor is not providing support for the local machine. Also, the test design technique Action-state testing is used only with the Harmony tool.

For both web applications, test scenarios were created first. The functional test cases were built based on the scenarios. After that, the test cases were executed, and the results were noted for each of the tools. The initial comparisons were focused on the features offered by the tools, such as whether they are free, the platforms

they might be used, and whether programming language knowledge is required. The comparisons of the tools that were based on the results were implementation time, test execution time, number of test cases, and number of failing test cases. In addition, maintenance testing is done for Selenium and Harmony to see how efficient the tools are in adapting to changes in the source code.

At the beginning of this study, some objectives were set which outline the importance of testing web applications and the benefits of using automated test design techniques and automation tools. Web application testing is crucial to ensuring the application's quality and functionality. Developers can do testing more quickly and efficiently while saving time and effort by using automated test design techniques and automation tools. The right test automation tool should be chosen depending on the environment and requirements of that specific project. In addition, the Harmony tool was found to be the most effective at performing functional and maintenance testing that was less time-consuming and user-friendly compared to Selenium and testRigor. Action-state testing is the primary contributing element to these results. It helped in the creation of more test cases and cover a wider range of scenarios that were not covered at the beginning of the test scenario.

However, it is important to remember that the selection of a test automation tool should be made based on the requirements and environment of the project. Two small-scale web applications were used in this study, and this condition largely affected the results that were obtained. For small-scale web applications, Harmony with Action-state testing can be the best option.

The following section outlines the planned future works:

- Expansion to larger scale projects: Harmony proved to function well for simple web applications, but further studies may be needed to see how well it works for larger, complex projects.

- When compared to other tools: There are many different test automation tools available, but Selenium, Harmony, and testRigor were the three that were compared in this study. Future research could add alternative tools to see which is best for a specific project.

- Integration with other test design techniques: Although action-state testing was used in this study, other test design techniques might be integrated with test automation tools to produce more thorough testing. Future research may

examine how to combine test automation tools with other test design techniques.

- Testing across different platforms: In this study, testing was done on web applications. Future research may test automation tools on other platforms, such as mobile applications, desktop applications, or embedded devices.

Acknowledgements

I would like to express my deepest gratitude and appreciation to my thesis advisor, Kovacs Attila, for their essential guidance and support throughout my research. His expert knowledge, feedback, and advice have been critical to the completion of this thesis.

Bibliography

[1] D. O'Halloran. "4 ways the web has changed our lives – and will shape our future". In: *World Economic Forum* (Mar. 2019). URL: http://www.weforum.org/agenda/2019/03/four-ways-the-web-has-changed-our-lives-and-will-shape-our-future/.

[2] Paul Gerrard and Neil Thompson. *Risk Based E-Business Testing*. USA: Artech House, Inc., 2002. ISBN: 1580533140.

[3] Daniel Nations. "What is web 2.0?" In: *Lifewire* (Apr. 2022). URL: http://www.lifewire.com/what-is-web-2-0-p2-3486624.

[4] Daniel Martin. "11 of the most costly software errors in history". In: *Raygun Blog* (Jan. 2022). URL: https://raygun.com/blog/costly-software-errors-history/.

[5] Pete Swabey. "The 10 worst web application failures". In: *Information Age* (Feb. 2006). URL: https://www.information-age.com/the-10-worst-web-application-failures-22592/.

[6] Kasper Siig. "Test automation: Benefits and use cases". In: *SystemsDigest* (Jan. 2023). URL: https://systemsdigest.com/posts/test-automation-benefits-and-use-cases.

[7] In: *ISO 25000* (n.d). URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010.

[8] "Web application testing guide: How to test a website". In: *Software Testing Help* (2023). URL: https://www.softwaretestinghelp.com/web-application-testing/.

[9] "Best Practices for Agile Testing". In: *Global App Testing* (n.d). URL: https://www.globalapptesting.com/the-ultimate-guide-to-agile-testing.

[10] J. Adam. "Sprints in Agile". In: *K&C* (Nov. 2022). URL: https://kruschecompany.com/agile-software-development/.

[11] Thomas Hamilton. "What is Agile Testing? Process and Life Cycle". In: *Guru99* (Mar. 2023). URL: https://www.guru99.com/agile-testing-a-beginner-s-guide.html.

[12] "What Is Agile Software Testing?" In: *Micro Focus* (n.d). URL: https://www.microfocus.com/en-us/what-is/agile-testing.

[13] Miodrag Zivkovic and Tamara Zivkovic. *Challenges in Testing of Web Applications*. Jan. 2018, pp. 91–96. DOI: 10.15308/Sinteza-2018-91-96.

[14] James McLaughlin. "10 common web and app testing issues and how to avoid them". In: *Digivante* (July 2022). URL: https://www.digivante.com/blog/10-common-web-and-app-testing-issues-and-how-to-avoid-them/.

[15] Shivangi Bhardwaj. "Analytical Review of User Perceived Testing Techniques". In: *International Journal of Advanced Research in Computer Science and Software Engineering* (Oct. 2012).

[16] Thomas Hamilton. "Web Application Testing: How to Test a Website?" In: *Guru99* (2023). URL: https://www.guru99.com/web-application-testing.html.

[17] S. Pittet. "The different types of testing in software". In: *Atlassian* (n.d). URL: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing.

[18] Thomas Hamilton. "What is functional testing? types & examples". In: *Guru99* (Mar. 2023). URL: https://www.guru99.com/functional-testing.html.

[19] Thomas Hamilton. "Functional Vs Non-Functional Testing – Difference Between Them". In: *Guru99* (Mar. 2023). URL: https://www.guru99.com/functional-testing-vs-non-functional-testing.html.

[20] B. Rajkumar. "Maintenance testing guide: You should know". In: *Software Testing Material* (Jan. 2023). URL: https://www.softwaretestingmaterial.com/maintenance-testing/.

[21] "What Is Test Maintenance? A Guide To Keeping Tests Healthy". In: *Perfecto* (Feb. 2022). URL: https://www.perfecto.io/blog/test-maintenance.

[22] "Overview of Test Design Techniques in Software Development". In: *Test Automation Resources* (Nov. 2018). URL: https://testautomationresources.com/software-testing-basics/software-test-design-techniques/.

[23] Thanh Huynh and Rachit Zambre. "Test case design 101: A beginner's guide". In: *AskTester* (Apr. 2022). URL: https://www.asktester.com/test-case-design-101-beginner-guide/.

[24] D. Sanjeev, N. Kumar, and S. Saini. *MODEL BASED TESTING CONSIDERING STEPS, LEVELS, TOOLS & STANDARDS OF SOFTWARE QUALITY*. Apr. 2011. URL: https://www.rroij.com/open-access/model-based-testing-considering-steps-levels-tools-and-standards-of-software-quality-148-151.pdf.

[25] G Ukkuru. In: *Why use model-based testing?* (Oct. 2022). URL: https://www.linkedin.com/pulse/why-use-model-based-testing-george-ukkuru/?trk=pulse-article_more-articles_related-content-card.

[26] I. Schieferdecker and A. Hoffmann. In: *Model-Based Testing* (2010). URL: www.ischieferdecker.bplaced.net/publications/2010/Schieferdecker2010d%20-%20Model%20Based%20Testing.pdf.

[27] Muhammad Nouman Zafar et al. "Model-based testing in practice: An industrial case study using GraphWalker". In: *14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference)* (2021). DOI: 10.1145/3452383.3452388.

[28] In: *Harmony* (Sept. 2022). URL: http://www.harmony.ac/.

[29] Istvan Forgacs. "A deeper insights into test design". In: *LambdaTest* (Aug. 2022). URL: https://www.lambdatest.com/blog/insight-on-test-design/.

[30] "Action-state testing". In: *Site of Software Test Design* (Nov. 2021). URL: https://test-design.org/action-state-testing/.

[31] Attila Kovács and Istvan Forgács. "Action-state testing". In: *Paradigm Shift in Software Testing*. 2021, 35–35.

[32] Daniel Howard. "Test Design Automation". In: *Bloor Research* (Jan. 2023). URL: http://www.bloorresearch.com/technology/test-design-automation/.

[33] "Test Automation and Automated Test Design". In: *Site of Software Test Design* (Oct. 2019). URL: https://test-design.org/test-automation-and-automated-test-design/.

[34] In: *Certified Tester Advanced Level Syllabus* (n.d). URL: https://istqb-main-web-prod.s3.amazonaws.com/media/documents/ISTQB_CTAL-TA_Syllabus_v3.1.2.pdf.

[35] In: *katalon.com* (n.d). URL: https://katalon.com/resources-center/blog/automation-testing-tools.

[36] R. Rice. *Surviving the Top Ten Challenges of Software Test Automation*. 2003. URL: https://www.cmcrossroads.com/sites/default/files/article/file/2012/XDD6506filelistfilename1_0.pdf.

[37] In: *Selenium* (n.d). URL: http://www.selenium.dev/documentation/.

[38] J. Unadkat. "Selenium WebDriver Tutorial: Getting Started with Test Automation". In: *BrowserStack* (2023). URL: https://www.browserstack.com/guide/selenium-webdriver-tutorial.

[39] In: *Selenium* (n.d). URL: https://www.selenium.dev/documentation/webdriver/.

[40] "Why Should You Choose Selenium? 10 Benefits Of Selenium That You Must Know About". In: *Appsiearra* (n.d). URL: https://www.appsierra.com/blog/selenium-benefits.

[41] A. McPeak. "Selenium 101: Running your test against two browsers at once". In: *Smartbear* (Feb. 2018). URL: https://smartbear.com/blog/run-test-multiple-browsers-parallel-selenium/.

[42] "Limitations of Selenium Webdriver". In: *GeeksforGeeks* (Apr. 2020). URL: https://www.geeksforgeeks.org/limitations-of-selenium-webdriver/.

[43] In: *GetApp* (n.d). URL: http://www.getapp.com/it-management-software/a/testrigor/.

[44] A. Golubev. "Can testRigor run tests in parallel?" In: (May 2022). URL: https://testrigor.com/blog/faq-can-testrigor-run-tests-in-parallel/.

[45] Süleyman Fatih İşler. *Automated test code generation and execution system for Web*. 2015. URL: https://etd.lib.metu.edu.tr/upload/12618415/index.pdf.

[46] Natnael Gonfa Berihun, Cyrille Dongmo, and John Andrew Van der Poll. "The Applicability of Automated Testing Frameworks for Mobile Application Testing: A Systematic Literature Review". In: *Computers* 12.5 (May 2023), p. 97. ISSN: 2073-431X. DOI: 10.3390/computers12050097.

[47] Divya Kumar and Krishn Mishra. "The Impacts of Test Automation on Software's Cost, Quality and Time to Market". In: *Procedia Computer Science* 79 (Dec. 2016), pp. 8–15. DOI: 10.1016/j.procs.2016.03.003.

[48] F. Okezie, I. Odun-Ayo, and S. Bogle. "A critical analysis of software testing tools". In: *Journal of Physics: Conference Series* 1378.4 (2019), p. 042030. DOI: 10.1088/1742-6596/1378/4/042030.

List of Figures

List of Tables

List of Codes