Alejandro Monje Morales

AUTOMATED FRONT-END WEBSITE TESTING WITH CYPRESS

Thesis

CENTRIA UNIVERSITY OF APPLIED SCIENCES

Bachelor of Engineering, Information Technology

September 2023

ABSTRACT

Centria University of Applied Sciences	Date	Author
	September 2023	Alejandro Monje Morales
Degree program		
Bachelor of Engineering, Information Technology		
Name of thesis		
AUTOMATED FRONT-END WEBSITE TESTING WITH CYPRESS		
Centria supervisor		**Pages**
Jari Isohanni		44
Instructor representing commissioning institution or company.		
Tommi Ronkainen		

This thesis explores Cypress as a modern JavaScript-based testing framework for website testing. It aims to examine Cypress's features, pros, and cons compared to other tools, offering practical guidance on test creation, execution, reporting, and its potential for security testing.
The thesis starts by highlighting the importance of software testing and website testing in the software-driven world. It discusses Cypress's architecture, native access features, and its unique testing approach. It also identifies scenarios where Cypress excels in web application testing.

The practical aspects of Cypress are extensively covered, including test syntax, DOM terminology, locator usage, and essential commands and assertions. The thesis discusses test execution, reporting, integration with CI/CD pipelines, and explores Cypress's potential for performance and load testing. Lastly, it delves into security testing using Cypress, identifying techniques for evaluating web application security.

Key words
Cypress, E2E, automated testing, web testing, web application, software testing.

CONCEPT DEFINITIONS

API

APIs serve as mechanisms that facilitate communication between two software components by employing a specific set of definitions and protocols. To illustrate, consider the weather bureau's software system, which stores daily weather data. Through APIs, the weather app on your phone establishes communication with this system and retrieves the latest weather updates, ultimately displaying them on your device. The term "API" stands for Application Programming Interface, where "Application" refers to any software with a distinct purpose. In the context of APIs, an "Interface" can be perceived as a contractual agreement between two applications, determining the rules and methods of communication through requests and responses. (Amazon Web Services 2023.)

CI/CD pipelines

Continuous Integration / Continuous Development. A CI/CD pipeline refers to the complete sequence of operations executed when initiating work on your projects. This pipeline includes workflows that manage and coordinate various tasks, all specified in your project configuration file. (Fosco 2022.)

Code snippet

Code snippets refer to compact sections of code that can be reused and inserted into a code file through various methods such as the right-click context menu or specific keyboard shortcuts. These snippets commonly encompass frequently used code blocks like try-finally or if-else statements, but they can also be utilized to insert entire classes or methods into the code. (Lee et al. 2023.)

CSRF

Cross-Site Request Forgery (CSRF) is an attack method that compels authenticated users to unknowingly send a request to a web application while they are already authenticated with it. (Synopsys 2023a)

DOM

The acronym DOM stands for Document Object Model, which is an interface in programming that enables us to manipulate elements within a document. It provides methods for creating, modifying, or deleting elements in the document. Additionally, the DOM allows us to attach events to these elements, enhancing the interactivity of our webpage. The DOM conceptualizes an HTML document as a hierarchical structure composed of nodes. Each node corresponds to an HTML element. **(**Wilkins 2021.**)**

E2E testing

End-to-end testing, commonly referred to as E2E testing, is a method employed to ensure that applications function as intended and that data flow remains intact for a wide range of user tasks and processes. This testing approach adopts the viewpoint of the end user and replicates real-world scenarios. (Schmitt 2023.)

npm

The term "npm" is an abbreviation for Node Package Manager. It serves as a library and registry for JavaScript software packages. Additionally, npm provides command-line tools that assist in the installation of various packages and the management of their dependencies. npm is freely available and is heavily relied upon by more than 11 million developers globally. (Abramowski 2022.)

SQL

Structured Query Language (SQL) is a universally recognized programming language employed for the purpose of managing relational databases and executing diverse operations on the data stored within them. (Loshin 2022.)

TTD (test driven development)

Test Driven Development (TDD) is a method used in software development where unit tests are created before writing the actual code. It follows an iterative process that involves programming, creating test cases, and refining the code. (Unadkat 2023.)

XSS

Cross-site scripting (XSS) is a form of attack where a malicious actor inserts harmful executable scripts into the code of a trusted website or application. Typically, attackers initiate an XSS attack by sending a deceptive link to a user, enticing them to click on it. (Synopsys 2023b.)

ABSTRACT

CONCEPT DEFINITIONS

CONTENTS

FIGURES

TABLES

CODE SNIPPETS

1 INTRODUCTION

In today's heavily software-dependent landscape, software testing emerges as a corner stone in ensuring the reliability and quality of software products. Within this realm, website testing assumes a pivotal role, given the critical functions that websites serve across a multitude of domains, spanning from e-commerce to communication systems. As websites continue to evolve in complexity and importance, the demand for proficient testing frameworks becomes increasingly urgent.

This thesis squarely focuses its lens on Cypress, a modern JavaScript-based testing framework meticulously crafted for the intricate world of website testing. Cypress arrives with a repertoire of features and advantages that render it an enticing choice for the rigorous evaluation of web applications. This research's cardinal objective is to furnish a comprehensive comprehension of Cypress, unraveling its multifaceted capabilities and illuminating its potential to elevate the testing process to unprecedented heights.

The abstract inaugurates its discourse by underscoring the fundamental significance of software testing, underscoring its pivotal role in the scrutiny of software quality throughout the developmental voyage. Subsequently, it narrows its focus onto the realm of website testing, magnifying the challenges and perils that encapsulate website development, thus accentuating the indomitable necessity for robust testing methodologies.

The pragmatic facets of engaging with Cypress take center stage, the art of composing efficacious tests, and the artistry of executing tests and generating incisive reports. Notably, the abstract casts a bright spotlight on Cypress's potential in the realm of security testing, unraveling the intricacies of techniques and considerations paramount in assessing the security fortitude of web applications.

By plumbing the depths of Cypress and exploring its application in website testing, this thesis aspires to empower software developers and testers with an arsenal of knowledge and pragmatic wisdom, enabling them to harness Cypress's full potential. Ultimately, this research endeavors to not only elevate software quality but also enhance the user experience within the ever-evolving tapestry of web applications, embodying the ongoing quest for digital excellence.

2 INTRODUCTION TO SOFTWARE TESTING

Software testing involves the systematic confirmation and validation of a software or application's freedom from defects, adherence to technical specifications outlined in its design and development, and its ability to meet user requirements effectively and efficiently, including the handling of exceptional and boundary scenarios. Software testing is not solely focused on identifying existing faults but also strives to identify opportunities for enhancing the software in terms of its efficiency, precision, and user-friendliness. Its primary objectives include assessing the compliance, functionality, and performance of a software program or application.

2.1 Definition and importance of software testing

Many of the everyday necessities that our society depends on are highly supported by software. Examples of these are transportation systems, power grids, communications, financial calculation engines, etc. Even normal households have hundreds of running microcontrollers managing activities that for us humans seem extremely simple, and the complexity behind those activities does not even come to mind, e.g., TV's remote control, microwave, router, desktop computer. Many factors affect the quality of the software that is delivered to the users, including meticulous design and flawless process management, that being said, testing is the main way the industry can evaluate software during the development of such. A new increase in agile processes has placed pressure on testing; unit testing is highly regarded, and test-focused development makes testing a fundamental step for the development of functional requirements. There is a clear understanding of how important testing is to guarantee the success of a software product. The good news is the basics of software testing can be used for numerous arrays of software applications. (Ammann , Offutt 2017, 3-4.)

2.2 Objectives and Goals of Software Testing

It is not uncommon that software engineers do not have a clear picture about what is expected from testing. Is it to show issues, preciseness, or something else? To dive into this idea, it is important first to separate two concepts, verification, and validation. Verification refers to the step in which it is analyzed

if the output of a stage of the software development process meets the established demands agreed upon the previous stage of development. In the other hand validation refers to the step in which the software is evaluated at the end of the development process to establish that all the tasks for which the software was developed for in the first place can be performed. Verification is commonly based on technical knowledge about specifications, a by-product of the software and requirements. Validation on the other hand, is related to the knowledge of the purpose for which the software was developed. To make this definition clearer, imagine the software that controls an airplane, such software needs validation from the knowledge of pilots and aerospace engineers who work in the field, but have nothing to do with software implementation. (Amman et al. 2017, 8-10.)

As another example, consider a switch that controls a set of lights in a room. In this case, verification requests if the lighting works according to the specifications. The specifications may say, the lights located in the very front of the room can be controlled independently front the other light in the room, if this is the requirement and lights can't be controlled separately, we can say that the lighting system did not pass verification, because the lighting was implemented, did not meet the specifications. Validation only request user satisfaction. If, specifications are not presented (written down) beforehand, even if the user are extremely disappointed, verification passes, because implementation meets the specification, but validation does not pass, because it does not meet the user's requirements. Therefore, it can be said that validation discloses weaknesses in the specifications. Beizer, explains test with the idea of "test process maturity levels". There are five defined levels, where the first or lowest does not get a number. Maturity levels are explained in detail in TABLE 1. (Amman et al. 2017, 9.)

TABLE 1 Maturity levels. (Knowledge Sharing 2023.)

Level	**Description**
Level 0	"There is no difference between testing and debugging." This is the type of testing used by many undergraduate students from computer sciences majors. In the majority of the classes, the students are assigned a program which they have to compile, then debug with some inputs either given by the teacher or created by the students. The issue with this way of testing is that there is no distinction between an erroneous behavior of the program and an error within the program, and it gives very little help to develop safe or reliable software. (Knowledge Sharing 2023.)

Level 1	"The purpose of testing is to show correctness." Compared to level zero, this is a major step up, the disadvantage being that correctness is very hard to reach or prove. E.g., a few tests are run, and failures are found. What is known after this? Is it possible to assume that we have good software or wrong tests? Due to the fact that correctness is mostly unachievable, testing personnel does not have rigid goal, or a set test technique. When testing personnel are asked how much testing is left to do, they do not have the means to answer that question. Genuinely, testing personnel are in a vulnerable position because they cannot evaluate their work or express it in a qualitative manner. (Knowledge Sharing 2023.)
Level 2	"The purpose of testing is to show that the software does not work." The idea of level 2 is to expose failures. This could be a justifiable achievement, but it is also essentially negative. Level 2, places testers and developers in a colliding position, thus can be harmful for the team itself. However, when the goal is to find failures, the uncertainty about what it will be done if none are found still lingers. Is the software well done, or the testing was poor? Having confidence to know when the test is done is an important achievement for every tester. According to professionals, this level leads the software field. (Knowledge Sharing 2023.)
Level 3	"The purpose of testing is not to prove anything specific, but to reduce the risk of using the software." What leads to level 3, is to realize does not show the absence of failures, but the presence of them. This helps us to know that anytime the software is used, we are not on solid ground and a certain risk must be assumed. The risk can present unremarkable or disastrous consequences depending on the size of the risk. This enables us to understand that the goal of the entire development team is to lower the risk associated with utilizing the software. In this level, testers and developers work as a unit to diminish the risk. (Knowledge Sharing 2023.)
Level 4	"Testing is a mental discipline that helps all IT professionals develop higher-quality software" Once developers and tester are working harmoniously together toward the same goal, the organization advances to the true Level 4 testing. Level 4 reasoning explains testing as an intellectual method that enhances quality. When this way of thinking is embraced, test engineers can turn into the technical head of the project. They possess the main

	duty of calculating and boosting the quality of the software, therefore developers can rely on testers' expertise when help is needed. (Knowledge Sharing 2023.)

2.3 Website Testing

In essence, web testing involves the examination of your web application or site for issues prior to its launch. The purpose of web testing is to thoroughly assess all dimensions of the web application's performance, encompassing the identification of glitches in usability, compatibility, security, and overall functionality. Integrating web testing is an essential component of the development process for any web-based application or website. After dedicating considerable resources, both in terms of time and finances, to develop such a platform, encountering immediate issues upon its release would be an unfortunate scenario that we've witnessed and hope to avoid. (Katalon 2023.)

2.4 Importance of website testing.

Web applications also need to be tested to assure quality, validation, and verification. The process of testing these applications is called web testing. Due to that web-based software application are exposed to different interactions, e.g., all types of background applications, different operating systems, different hardware, testing those applications present a higher level of challenges when it comes down to testing. If issues with the quality of the web application are found, will cause loss of confidence in the software from part of the end user, and also affect the reputation of the company that develops the application. (Jiujiu Yu 2019.)

2.5 Software quality factors

Factors that influence the software are called software factors. Those factors can be widely divided into two groups. The first group includes the factors that can be measured precisely e.g., logical errors and the second group includes those factors that can be measured indirectly e.g., maintainability. In 1977, McCall proposed the classic model of software quality factors which describes 11 factors. These factors are at the same time divided into three groups. The *product operation factors* group which includes

reliability, integrity, correctness, usability, efficiency. Next, we have product revision factors group which includes flexibility, maintainability, testability. Finally, we have the product transition factors group which includes reusability, interoperability, portability. (Tutorials Point 2023.)

2.5.1 Operation factors

Several key factors are crucial in software development. Reliability requirements govern the tolerance for failures, whether they pertain to the entire software system or specific functions within it. Integrity considerations focus on system security, determining access rights and permissions for users, both in terms of reading and writing. Correctness involves the precision and completeness of system outputs, accounting for potential impacts from incorrect calculations or incomplete data, adherence to coding standards, and comprehensive software documentation. Usability addresses the resources required for training staff to operate the software effectively. Lastly, efficiency encompasses the necessary hardware resources, including transmission capabilities, storage, and processor size, required for the software to fulfil its intended functions. These factors collectively shape the quality and performance of software applications. (Tutorials Point 2023.)

2.5.2 Revision factors

Flexibility assesses the effort and capabilities required for software maintenance, including its adaptability to changes and different customer needs without necessitating extensive modifications. This also encompasses the potential to add services that enhance the software's usability in various technical environments. Maintainability, on the other hand, gauges the effort needed for ongoing software maintenance, issue identification, and ensuring their proper resolution. Testability covers aspects of operations and testing, encompassing everything related to testing procedures, intermediate results, automatic diagnostics performed by the software pre-boot, ensuring the system functions correctly, and comprehensive reporting of detected issues. These factors collectively contribute to the overall quality and effectiveness of software. (Tutorials Point 2023.)

2.5.3 Transition factors

Reusability in software development pertains to the extent to which software designed for a particular project can be effectively repurposed for use in new projects still under development. Software that exhibits reusability not only conserves valuable time but also optimizes resource allocation. Interoperability, on the other hand, addresses the software's ability to seamlessly coexist and communicate with other systems or equipment, ensuring compatibility and efficient data exchange. Additionally, portability reflects the software's capacity to adapt to diverse environments, including varying hardware configurations and different operating systems, thereby enhancing its versatility and accessibility across a range of contexts. These factors collectively contribute to the overall utility and effectiveness of software solutions. (Tutorials Point 2023.)

2.6 Challenges and risks associated with website development.

One common challenge which organizations face in the hiring process is a lack of clarity regarding the specific type of specialists required for their projects. Often, stakeholders may not have a comprehensive understanding of the precise skills and expertise needed to achieve their goals. This knowledge gap can lead to misalignment between the job requirements and candidate qualifications, resulting in suboptimal hiring decisions and delayed project progress. Addressing this issue necessitates a thorough needs analysis and collaboration between project managers and subject matter experts to define clear job profiles and skill sets. (CyberCraft 2023.)

Also, identifying and attracting the right talent can be a daunting task for organizations. The challenge often lies in not knowing where to look for candidates who possess the desired qualifications and experience. This can lead to inefficient recruitment processes, increased time-to-fill positions, and missed opportunities. To overcome this challenge, organizations can explore various channels, including online job platforms, industry-specific forums, professional networks, and partnerships with educational institutions, to expand their reach and connect with suitable candidates. Additionally, leveraging employee referrals and building a strong employer brand can also help attract top talent. (CyberCraft 2023.)

Engaging tech advisors or consultants can indeed be costly, particularly for small and medium-sized enterprises with limited budgets. These advisors often bring valuable insights and expertise but may not be financially viable for every project or organization. To address this challenge, businesses can consider

alternative solutions, such as upskilling existing employees, seeking mentorship from industry peers, or exploring open-source communities and resources. Additionally, leveraging freelance specialists on a project-by-project basis can provide cost-effective expertise without the long-term commitment associated with hiring full-time advisors. It is essential to strike a balance between accessing specialized knowledge and managing costs effectively. (CyberCraft 2023.)

If one does not want to find himself in this situation is good to have a clear understanding about what is precisely needed. To get the best results, being specific is a must. No matter what decision is taken concerning which tools are going to be used, endure that idea. If skilled labor is not found locally, try overseas. There are many available talented people around the world. Also, filter the right people. Do not waste time going through a massive number of interviews. Select the ones that meet the requirements the most. (CyberCraft 2023.)

2.6.1 Expenses Cost

The cost to develop the web application is probably the number one concern that a client has. The price will depend on how complex and the kind of the web application to be built, and according to this the price for hiring web designer and developers may change. Also, the geographical location where the personnel are located and how many of them are needed will also affect the price. (CyberCraft 2023.)

Generally, the price can become a colliding point between the client and the team in charge of the development, and sometimes changes in the predefined budget can even trigger a reaction in which the client decides to drop the project entirely. To solve this issue, communication and coordination with a web design service provider is paramount. Everything needs to be documented and must be an agreement among all involved parties concerning how the project will be developed. The services shown in TABLE 2 are considered while developing a web application. (CyberCraft 2023.)

TABLE 2 Considered services while developing a web application. (CyberCraft 2023.)

Service
User experience and planning
Design
Front-end coding
Back-end programming
Question and answer services and after-launch upkeep

2.6.2 To exceed time agreement

Inefficiencies in team management can have a cascading effect on project outcomes. Team members may not be effectively assigned tasks or roles, leading to confusion and duplication of efforts. Additionally, poor communication, inadequate feedback, and a lack of clear goals can hinder productivity. To address this challenge, organizations can implement robust project management methodologies, promote effective leadership and communication, and provide training and development opportunities for team leaders. Encouraging collaboration and fostering a positive team culture can also enhance overall efficiency. (CyberCraft 2023.)

Faulty planning can significantly impact project success. Errors in project planning might involve inaccurate resource estimations, unrealistic timelines, or overlooking critical dependencies. These mistakes can lead to missed deadlines, budget overruns, and a decrease in overall project quality. To mitigate this issue, organizations should invest in comprehensive project planning, including thorough risk assessment and contingency planning. Regular reviews and revisions of project plans based on evolving circumstances can help ensure alignment with project goals and objectives. (CyberCraft 2023.)

Efficiently utilizing granted resources is essential to optimize project outcomes. Failure to allocate resources effectively can result in wastage or insufficient resource availability, hindering project progress. To overcome this challenge, organizations should implement resource management tools and practices that allow for accurate tracking and allocation of resources. This includes monitoring resource utilization, identifying bottlenecks, and reallocating resources as needed to meet project requirements. (CyberCraft 2023.)

Last-minute updates or corrections can disrupt project timelines and introduce unnecessary stress. These changes may arise due to overlooked requirements, evolving stakeholder expectations, or unforeseen issues. To manage this challenge, organizations should emphasize thorough requirements gathering and continuous stakeholder engagement throughout the project lifecycle. Additionally, implementing robust change control processes can help evaluate and prioritize requested updates or corrections, ensuring they are managed systematically and with minimal disruption to the project schedule. (CyberCraft 2023.)

To avoid going through these possible scenarios, it is necessary to ensure that the team's focus is 100% on the project at hand. To avoid getting caught with accumulated work, it is recommended to set timeframes and deadlines for each task. Following this recommendation, it is possible to complete the project within the pre-stablished time frame, no matter how big the tasks are. (CyberCraft 2023.)

2.6.3 Unachieved pre-stablished goals

Having all the requirements beforehand is essential for any web application project. If those are not documented and understood, they will most likely be ignored. The final product might not function the way it was expected. It is common that clients may forget or ignore non-practical requirements as it assumes they don't need to be explained. Therefore, it is necessary to keep in mind that missing requirements can affect the quality of the final product, the timeframe and/or budget. The recommendation is that everybody must be on the same page, so all functions and features of the final product are agreed upon before the development process starts. To have a descriptive task of the client wants to perform with the web application will help to the development of it. (CyberCraft 2023.)

2.6.4 Low-quality final product

This is the worst expected result. A low-quality product will affect not only the business but also the developer's reputation. A poor-quality web application can be the result of a lack of understanding of what the requirements were before the project started to be developed. Also, the culture and language either of the client of web developer can affect such a negative result. A well-established, respected, and professional web developing service provider is always a good way to achieve positive end results. (CyberCraft 2023.)

3 TECHNICAL INTRODUCTION TO HTML

HTML is an acronym for Hyper Text Markup Language, employed for creating web pages through a specialized language. It combines the principles of Hypertext and Markup language, with Hypertext establishing connections between web pages. The use of a markup language is pivotal in delineating the textual content enclosed within tags, consequently structuring web pages. This form of language involves annotating text to make it comprehensible for machines, enabling text manipulation as intended. While many markup languages, including HTML, possess readability for humans, they rely on tags to specify text manipulation instructions. (Geeksforgeeks 2023.)

HTML functions as a markup language harnessed by web browsers for the purpose of formatting text, visuals, and additional elements to exhibit content as desired. Tim Berners-Lee is the originator of HTML, conceiving it in 1991. While the inaugural iteration was HTML 1.0, the first official standard rendition emerged as HTML 2.0, introduced in 1995. (Geeksforgeeks 2023.)

For some automation engineers, one of the main issues is to find web elements. This issue arises due to not having a clear understanding of what is name of each HTML field, and each HTML value is, therefore creates confusion about how to make the right locators when programming the test. To avoid this, it is necessary to have a clear understanding of the structure of the HTML language. (Bondar 2023.)

3.1 HTML page structure

HTML has a structure of tags; these tags are called HTML tag names. Commonly, these tags come in pairs, an opening tag and a closing tag. In the interior of an HTML tag, attributes can be found (they are called attribute name), which in some browsers appear highlighted when the DOM is inspected. These attributes have value and are also highlighted in a different color; in some cases, the attributes do not have any value. (Bondar 2023.)

The foundational framework of an HTML page is outlined as follows, encompassing pivotal constituent elements (namely, the doctype declaration, HTML, head, title, and body elements) that underpin the creation of all web pages. The main HTML tags are named and described in TABLE 3. (Geeksforgeeks 2023.)

TABLE 3. HTML main structure tags. (Geeksforgeeks 2023.)

HTML tag	Description
<!DOCTYPE html>	Referred to as the document type declaration, this component indicates an HTML document's nature, albeit technically not a tag. The doctype declaration maintains case insensitivity.
<html>	Serving as the HTML root element, it serves as the container for all other elements.
<head>	The head tag accommodates background elements of a webpage. Elements housed within the head remain concealed on the frontend.
	These tags are located inside the <head> tag / **Description** <style> / Enabling the integration of styling through CSS, enhancing the visual appeal of webpages. <title> / Displayed atop a browser, it bears the webpage's title and is seen when visiting a site. <base> / Specifies the base URL for relative URLs within a document. <noscript> / Defines an HTML section included when a user's browser disables scripting. <script> / Facilitates the inclusion of functionality via JavaScript. <meta> / Encloses metadata essential for each visit to a website. For instance, metadata charset enables <link> / The 'link' tag is employed to connect HTML, CSS, and JavaScript, and it closes on its own.
<body>	Wraps all visible webpage content, representing what browsers present on the frontend.

3.2 DOM terminology

Ids and classes are attributes names, and they play a special part within the DOM. Id is a unique identifier and class defines the style sheet. Class also has a value, and it is important to have comprehend that when a class has a long value, this is not just one value but multiple. In some cases, a value can be observed in between angle braces, this is regular text value. Therefore, commonly everything that is seen in a website as a text value is located in between angle braces in the DOM. The word 'commonly' was used because sometimes the text can be concealed in the properties of the browser, and to get to this value it is necessary to follow other steps. (Bondar 2023.)

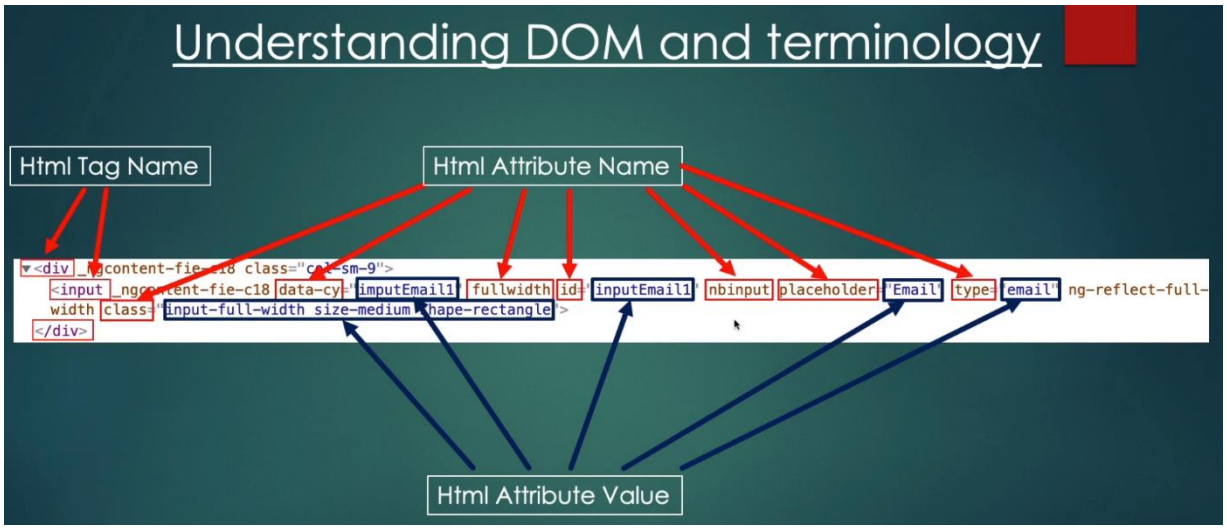

FIGURE 1. DOM terminology. (Bondar 2023.)

There is a relationship among web elements that also needs to be understood. If Figure 2 is observed, the table body HTML tag is the parent element for table row and all the other elements that are found above the table body tag will be parents of this last. All the elements that are found inside the table row are child elements of table row. Elements are considered sibling when are located at the same level of the structure e.g., the td tags. (Bondar 2023.)

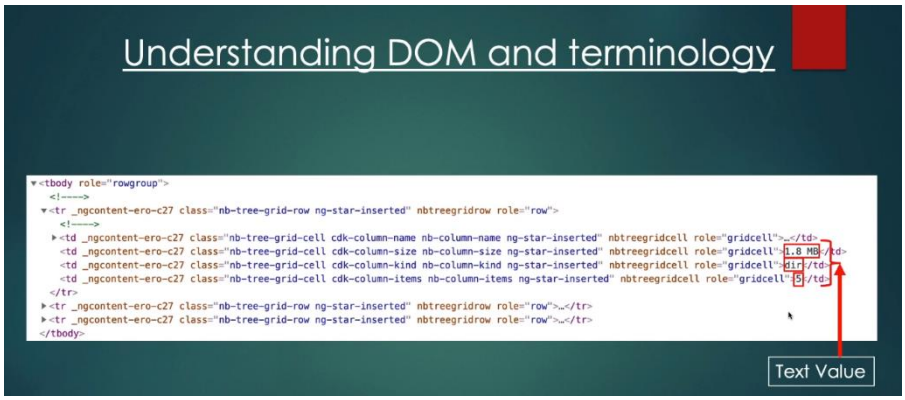

FIGURE 2. Regular text value in the DOM. (Bondar 2023.)

Summarizing, the HTML Document Object Model (DOM) is comprised of elements such as tags, attributes, and values, which form the foundation of web page structure. Among these, Id and Class are notable attributes used to uniquely identify and style elements. Multiple values can be assigned to a single class, provided they are separated by spaces, allowing for versatile styling options. HTML tags conventionally appear in pairs, encompassing both opening and closing tags. Within these tags, plain text serves as the content enclosed between angle brackets (e.g., >example<). In the hierarchy of web elements, parent elements are positioned above the focal web element, while child elements reside within it. Sibling elements, on the other hand, are located at the same hierarchical level, appearing side by side in the DOM structure. (Bondar 2023.)

4 INTRODUCTION TO CYPRESS AS A MODERN JAVASCRIPT-BASED TESTING FRAMEWORK

Cypress is an advanced testing tool that offers web developers and quality assurance experts a comprehensive solution for conducting end-to-end testing of web applications. This tool stands out for its intuitive user interface and real-time testing capabilities, streamlining the process of ensuring the reliability and functionality of web applications. Its distinctive architecture seamlessly combines testing and debugging, making it a potent choice for modern web development needs. (Cypress, 2023a.)

4.1 Architecture

The majority of testing tools, such as Selenium, function by being executed externally to the browser with the use of isolated commands through the network. A Node server process is found in the back of Cypress. The Node process and Cypress are in constant synchronization, communication and carry out responsibilities for one another. Due to that there is access to front and back end, the reactions to the events of the application happen in real time, while simultaneously, actions that need greater privileges are done outside the browser. Cypress can also work through the network by reading and modifying web transit on the go. Cypress can modify the code that is stopping it from automating the browser and also can change everything that is going to and coming from the browser. (Cypress 2023b.)

Cypress can also access the operating system for automation activities since it is installed locally on the computer. This makes it possible to carry out actions like taking screenshots, recording videos, conducting standard file system operations, and carrying out network operations. Finally, Cypress manages the whole automation procedure, which places it in the particular position of being aware about what is going on outside and inside the browser, meaning that Cypress delivers much persistent results than any other testing application. (Cypress 2023b.)

4.2 Native Access

Cypress has direct access to all the components since it works inside the application. It does not matter if it is a DOM element, a function, a window, or any other component Cypress can access it. Every object that the application code has access to, the test code can access them as well. This inherent ability to mirror the access the application has to components serves as a testament of the robustness of Cypress as a testing framework. It ensures that test scenarios can encompass a wide range of interactions and validations, from simple user interface interactions to complex data-driven testing. The ability to tap into the internal components of the application empowers testers to craft comprehensive and realistic test suites, allowing them to replicate real-world user interactions and assess the functionality of the application, performance, and reliability with precision. Moreover, this direct access also streamlines the debugging process. When issues arise during testing, the deep integration of Cypress with the application makes it easier to pinpoint the root causes, as testers can investigate and manipulate application components in real-time. This level of transparency and control contributes to faster issue resolution and ultimately leads to more efficient and effective testing practices. (Cypress 2023b.)

4.3 New type of testing

Having total control over network flow, the application and direct access to all objects among the application, unfolded a new type of testing that was not possible before. Cypress allows to modify any component of the application's functionality. States can be generated artificially, the same is done in unit testing, as opposed to doing slow and expensive tests like creating the state necessary for a certain case. Examples of this is shown in TABLE 4. (Cypress 2023b.)

TABLE 4. Examples of tasks that can be executed with Cypress. (Cypress 2023b.)

Examples
Modifying response status code to 500 will allow to test how the application responds to failures on the server
Elements that are hidden can be forced to appear by modifying the DOM.
Synchronous notifications can be received anytime the application starts to unload or changes to a new page.

Customized event listener to answer the application can be created. It is possible to update application code to act in a different manner while being tested by Cypress.
By making the server send empty responses, is possible to test edge cases such as “empty views”
Expose data stores so the tester can programmatically change the state of the application straight from the test code similar to Redux.

4.4 Shortcuts

With the help of cy.session(), Cypress enables caching of the browser context. In other words, the user only needs to complete authentication once for the full test suit, and you can then restore the previously saved session between tests. This means that each time the test is run, it is not necessary to go to the login page, insert user and password and wait for the page to load or redirect. With cy.session() and, if necessary, cy.origin() this task can be completed once. (Cypress 2023b.)

4.5 Advantages of Cypress

The Cypress framework offers several noteworthy features that enhance the testing experience. Firstly, it captures screenshots during the test execution, enabling quality assurance professionals or developers to review specific commands within the command log to gain precise insights into each step of the test process. Unlike Selenium, Cypress eliminates the need for explicit or implicit wait directives by automatically handling requests and claims, streamlining the testing workflow. Furthermore, Cypress includes an automatic scrolling function that ensures elements are visible before any actions are taken, improving test reliability. Notably, recent updates have expanded Cypress's browser compatibility beyond Chrome, now offering support for Firefox and Edge browsers. Additionally, Cypress provides real-time execution of instructions as they are typed, accompanied by visual feedback, making it a user-friendly choice for testing, supported by comprehensive documentation (BrowserStack, 2023).

4.6 Cypress's disadvantages

Cypress has certain limitations, as reported by BrowserStack in 2023. It cannot facilitate concurrent operation of two browsers, lacks support for multi-tab functionalities, exclusively permits test case creation in JavaScript, does not currently extend its compatibility to Safari and IE browsers, and offers only restricted support for iFrames. (BrowserStack 2023.)

4.7 When to use Cypress

Even though Cypress and Selenium are made to automate browsers for testing reasons, their architecture and performance are very different. Cypress is a tool that developers may use to learn about test automation rather than just a Selenium alternative. This is why Cypress is one of the automation tools with the greatest rate of growth. Selenium is more all-purpose tool aimed for a wider audience. Cypress is chosen over Selenium when the tester has certain requirements which are mentioned in TABLE 5. (BrowserStack 2023.)

TABLE 5. Cypress is chosen when tester requires. (BrowserStack 2023.)

Tester needs
Test execution capabilities for video recording
One programming language for automation and development (front-end)
Save time by using Cypress's stable, ready-to-use framework rather than creating your own from scratch with Selenium because it comes with everything wrapped up and ready to use
Component, API, e2e, visual, accessible, and performance testing in a single framework
When any test fails, there are meaningful exceptions
Test runner that allows you to time travel through particular steps and attach before and after screenshots to debug errors
The ability to retry actions carried out over items out of the box, which lower flaky tests for Cypress
Capabilities to Mock/Stub Requests and Responses early in Development

4.8 Test structure

Before starting to write the test, the user needs to make sure that the test file has been created in the correct folder. As said before in this text, all tests need to be located in the e2e folder which is inside the cypress folder. The test file has to have extension .spec.js or .cy.js. It is most common to find .cy.js in Cypress newest version, the other extension works as well if the configuration file is set up properly. (Bondar 2023.)

The test starts with the reserved word describe() or context(); these words are interchangeable. For this example and future references the word describe() will be used. The word describe() provides a wide description of what the test is about. The description must be placed within single or double quotes. After this, a callback function is opened with curly braces. Within the body of this function the tests will be created. The key word it('test description') provides the description of the test. Within the describe() scope the number of tests that can be created is unlimited. (Bondar 2023.)

FIGURE 3. Basic structure of the test. (Bondar 2023.)

4.8.1 Locators

Locators are used to find web elements in Cypress. Any Cypress command to be executed starts with the word cy. and the desired method e.g., cy.get(' '). It must be mentioned that on the very top of the file, this line of code is seen ///<reference types ="cypress"/>, this is used so Visual Studio Code will support IntelliSense and it makes it easier to identify Cypress methods. After typing cy.get(' ') method, inside the quotes the locator must be provided. TABLE 6 shows locators used to identify web elements. (Bondar 2023.)

TABLE 6. Locators description and how they are used to identify web elements. (Bondar 2023.)

Description	Usage
Locating element by HTML tag name	cy.get("input")
Locating element by id	cy.get("#inputEmail1")
Locating element by class name	cy.get(".input-full-width")
Locating element by attribute name	cy.get("[placeholder]")
Locating element by attribute name and value	cy.get('[placeholder="Email"]')
Locating element by class value	cy.get('[class="input-full-width size-medium shape-rectangle"]')
Locating element by tag name and attribute with value	cy.get('input[placeholder="Email"]')
Locating element by two different attributes	cy.get('[placeholder="Email"][type="email"]')
Locating element by tag name, attribute with value, ID and Class name	cy.get('input[placeholder="Email"]#inputEmail1.input-full-width')
The most recommended way by cypress (create own attribute)	cy.get('[data-cy="imputEmail1"]')

4.9 Essential Cypress commands and assertions

Cypress, a modern JavaScript-based testing framework, empowers developers and quality assurance professionals with a robust set of commands and assertions. These essential tools are at the heart of Cypress's effectiveness in testing web applications. Cypress commands allow users to interact with web elements, perform actions, and navigate through web pages, while assertions enable the verification of expected outcomes, ensuring that the application behaves as intended. In this brief overview, we will explore some of the key Cypress commands and assertions that play a pivotal role in creating comprehensive and effective tests for web applications. These tools not only streamline the testing process but also contribute to the overall reliability and quality of web-based software products. A list of essential commands and their usage and shown in TABLE 7. (Tutorial Hut 2023.)

TABLE 7 Essential commands used in Cypress. (Tutorial Hut 2023.)

Command	Description	Usage
and	Used as an alias of should and makes assertion	cy.get('input').should('contain', 'hello').and('be.enabled')
as	Creates an alias for future use	cy.get('.text').find('li').first().as('One')
blur	It blurs an element that is in focus	cy.get('.text'). type('element').blur()
check	It marks radio buttons and checkboxes. It is used on elements with input tags.	cy.get('[type="radio"]').check('male')
select	Selects an option within a dropdown menu	cy.get('select').select('peaches').should('have.value', 'sweet')
children	It gets an element's sub elements	cy.get('ul.l2').children()
clear	It deletes the values from a text area or input field	cy.get('.user'). type('Name').clear()
clearCookie	It deletes a specific browser cookie	cy.clearCookie('cookiename')
clearCookies	It deletes browser cookies from an already existent domain and subdomain.	cy.clearCookies()
getCookie	It acquires a specific browser cookie by name	cy.getCookie('cookie_name')
getCookies	Gets all the cookies	cy.getCookies()
click	Click on selected DOM element	cy.get('.button').click()
contains	It gets an element with a specific text. The element can have more than just text and yet be correct	cy.get('.text').contains('Student')
eq	It refers to a specific element in an array of items	cy.get('td').eq(2)
first	It selects the first element from a set of elements	cy.get('td').first()
go	It navigates to the next or previous URL in the browser's history	cy.go('back')
get	The get() method retrieves one or more elements based on the selector input	cy.get('.button')

visit	URL is launched	cy.visit('https://www.centria.fi')
should	It is an alias for .and(), it is also used to generate an assertion	cy.get('.text').should('contain', Hello').and('be.enabled')

4.10 Cypress assertions

Assertions are validations stages that determine whether or not the provided step of the automated test case was successful. In reality, assertions validate the desired state of your test parts, objects, or applications. Assertions, for example, allow the tester to validate whether an element is visible or has a specific attribute, CSS class, or condition. It is always suggested that all automated test cases include assertion phases. Otherwise, validating whether or not the application reaches the required state will be impossible. (Ravinder 2021.)

Cypress incorporates several assertions from various JS assertions libraries, including Chai, JQuery, and others. All these assertions can be divided into two categories based on the subject they can be used, implicit assertions and explicit assertions. (Ravinder 2021.)

Implicit assertion is one that applies to the object provided by the parent chain command. Furthermore, instructions like ".should()" and '.and()' fall under this type of assertions. Because these commands do not stand alone and are always depending on the previously chained parent command, they automatically inherit and operate on the object returned by the preceding command. CODE 1 shows an example of implicit assertion. (Ravinder 2021.)

```
cy.get('.button')
  .should('have.class', 'enabled')
  .and('have.attr', 'active')
```

CODE 1. Implicit Assertion. (Ravinder 2021.)

Because .should('have.class') does not modify the subject and .and('have, attr') is executed against the same element, these implicit assertions are useful when you need to assert many things against a same subject rapidly. Implicit assertions are commonly used when it's desired to make assertions regarding the same topic and to alter the subject previous making the assertion on the element. (Ravinder 2021.)

Explicit assertion is used when it is necessary to pass an explicit element for the assertion. This kind of assertion contains methods such assert() and expect() which permit the user to pass an explicit object/subject. In general, explicit assertions are used when is desired to apply some personalized logic before asserting the subject and to carry out a number of assertions on the same subject. CODE 2 shows an example of explicit assertion. (Ravinder 2021.)

```
const objec = { name: 'ale' }
expect(objec).to.equal(objec)
expect(objec).to.deep.equal({ name: 'ale' })
// The explicit subject here is the object objec.
```

CODE 2. Explicit Assertion. (Ravinder 2021.)

4.11 Writing effective and maintainable Cypress tests

Writing effective and maintainable Cypress tests is essential for robust web application testing. To achieve this, it's crucial to adhere to best practices that enhance the quality and sustainability of your test suite. Begin by structuring your tests logically with clear names and organized hierarchies, making your test suite more readable and purposeful. Employ custom Cypress commands for repetitive actions and interactions to reduce code duplication and simplify maintenance. Avoid hard waits and utilize Cypress's built-in retry mechanisms for waiting, ensuring tests remain reliable. Separate test data from code using fixtures and embrace the Page Object Model (POM) pattern to encapsulate page-specific interactions, isolating changes to specific pages. Implement custom commands for common assertions to streamline test code and consider data-driven testing for versatile scenario coverage. Leverage Cypress hooks for consistent setup and teardown and use selective testing during development. Integrate Cypress into your CI/CD pipeline to catch issues early and document your tests with comments to enhance comprehension and maintainability. By following these best practices, you can create Cypress tests that not only effectively validate your web application but also remain maintainable and adaptable as your application evolves. (Sthapit 2021.)

4.12 Every test must be singled out

Cypress automatically separates test files at the file level, which means that each test file should ideally not depend on any state or information from a previous test file. However, it is important to know that the test can fail at any step, and each “it” block (test case) is executed independently of the previous “it” block. This means that if one “it” block relies on the state or results from the previous “it” block, it can lead to a chain of failures, making it harder to comprehend the root cause of the issues later on. (Sthapit 2021.)

4.13 Assertion must be brief

When working with Cypress tests, a frequently encountered issue is when someone anticipates a certain condition to be true, resulting in an unclear assertion failure. Cypress provides a comprehensive list of assertions. It is advisable to intentionally cause the test to fail initially and assess whether the error message provides adequate information to determine the reason for the failure. Adopting this approach as a best practice enables the user to verify if the error message is informative enough to understand the cause of the failure. (Sthapit 2021.)

4.14 Avoid code duplication

By isolating and dividing tests into smaller individual tests, we can prevent unwanted cascading failures. However, this approach can result in a significant amount of duplicated code. In larger projects, where there are numerous reused UI components (such as theme-based components), modifying all the tests across different pages becomes undesirable if there are design changes to a single component. To facilitate easier end-to -end testing, we can consider shipping the functionality if all the expected behaviors are functioning correctly. Despite Cypress discouraging the Page Object Model (POM) pattern, employing POM can minimize the need for extensive refactoring in scenarios like the mentioned above. Additionally, it offers visually appealing test cases that are more straight forward to comprehend. (Sthapit 2021.)

5 TEST EXECUTION AND REPORTING

Test execution and reporting with Cypress involves the process of running automated tests on web applications and generating detailed reports about the test outcomes. Cypress facilitates seamless test execution by providing a powerful framework for writing and executing tests in real time, offering features like interactive debugging and time-traveling. After the tests are executed, Cypress generates comprehensive reports that include information about test cases, pass/fail statuses, and any errors encountered during the testing process. These reports aid in identifying issues, tracking progress, and ensuring the quality of the tested application. (Cypress 2023c.)

5.1 How Cypress tests are executed

Unlike most other testing tools, Cypress operates within the browser's environment. This unique architecture ensures that your tests are executed in the same environment as your application, granting Cypress the ability to detect all browser events and providing native access to every element within the tests. This is a great feature that Cypress offers because other tools like Selenium typically run outside of the browser and send remote commands over the network to control the browser. Cypress, on the other hand, functions differently. By running within the browser, Cypress possesses the capability to dynamically read and modify web traffic. It can intercept and manipulate all incoming and outgoing data within the browser. This native access also extends to various browser components such as the window object, document, DOM elements, and service workers. This distinction brings several advantages, such as the ability to directly interact with and modify front-end state libraries like Redux or MobX from your Cypress test. Essentially anything accessible to the browser can also be accessible by Cypress. However, it is important to note that this approach comes with certain trade-offs. (Cypress 2023c.)

5.2 How to generate test reports

Cypress, a test automation tool based on Node.js, has gained popularity due to its developer-friendly features. It supports various types of testing such as Components, API, and E2E testing. The release of Cypress 10 has further increase its traction by introducing advanced features. In addition to conducting

tests, reporting plays a crucial role delivering high-quality software. Without proper reporting, test automation efforts become ineffective. Even if a large number of tests have been covered, if they are presented in a way stakeholders can understand, their value diminishes. (Bhat et al. 2022.)

Therefore, it is essential to comprehensively report test data to maximize its effectiveness. Cypress offers support for multiple reporting options, with HTML reporting being the most useful and reliable. The HTML reporter generates visually appealing reports, that includes test execution status, detailed logs, screenshots, and other relevant information. This enables stakeholders to easily comprehend the test results and extract the maximum value from them. (Bhat et al. 2022.)

5.3 Integrations with continuous integration/continuous deployment (CI/CD) pipelines

Cypress seamlessly integrates with Continuous Integration/Continuous Deployment pipelines, enabling the incorporation of automated testing into your software development workflow. This integration allows for the automated execution of Cypress tests and provides valuable feedback on the application's quality at each stage of the development process. (Cypress 2023d.)

Multiple popular CI/CD tools can be integrated with Cypress, including Jenkins, Travis CI, CircleCI, GitLab CI/CD, Azure Pipelines, and GitHub actions. Jenkins, an open-source automation server, can be easily configured to execute Cypress tests and generate reports. Travis CI, a cloud-based CI/CD platform, integrates with GitHub repositories and facilitates smooth integration with Cypress for automated testing during the continuous integration process. CircleCI, another cloud-based CI/CD platform, offers a straightforward configuration process for running Cypress tests within the CI/CD pipeline. GitLab CI/CD, provided by GitLab, allows Cypress integration by defining specific stages or jobs for executing Cypress tests. Azure pipelines, a CI/CD service by Microsoft Azure, enables the incorporation of Cypress tests for different platforms and browsers as part of the development process. Lastly, GitHub Actions, a robust workflow automation and CI/CD platform, integrates with Cypress, enabling test execution whenever changes are pushed to the repository. These integrations empower the user to automatically trigger Cypress tests, gather tests results, and generate reports within the CI/CD pipelines. By doing so, they ensure that the application remains reliable and thoroughly tested throughout the entire software development lifecycle. (Cypress 2023d.)

5.4 Overview of using Cypress for performance and load testing

While Cypress is widely recognized as a functional test automation tool, it also offers features for conducting performance and load testing. Although Cypress is not specifically tailored for performance testing, it can be employed to simulate multiple virtual users and generate application load. To execute performance and load test using Cypress, its programmable nature can be leveraged by utilizing JavaScript to create custom scripts that mimic, concurrent user actions. By designing test scenarios that emulate real-world usage patterns, the user can evaluate the application's performance under diverse load conditions. (Bahmutov 2020.)

Cypress provides different plugins and libraries that can augment its functionality for performance testing. For instance, the "cypress-real-event" plugin facilitates realistic user interaction, while the "cypress-fail-on-console-error" plugin aids in identifying and managing errors during load testing. It is worth noting that Cypress primarily focuses on end-to-end testing and optimizing developer experience, and it may not offer the same level of sophistication and comprehensive metrics as specialized performance testing tools. However, if the user is already using Cypress for functional testing, harnessing its capabilities for basic performance and load testing can be a convenient choice. (Rubin 2021.)

5.5 Strategies for simulating and measuring load using Cypress

Cypress offers a comprehensive array of capabilities for advanced testing and analysis, enhancing the overall quality assurance process. Firstly, it facilitates load testing by allowing the execution of multiple tests concurrently to simulate heavy loads, and it supports parallelization techniques and external libraries to distribute tests across various machines or browsers, ensuring scalability. Additionally, Cypress empowers users with network traffic monitoring tools, enabling the interception and analysis of network requests and responses, thus facilitating the measurement of performance and the identification of optimization opportunities. The framework also supports performance metrics tracking through integration with tools like Lighthouse or Chrome DevTools, enabling the generation of performance reports and the ability to wait for specific network conditions before proceeding with tests. While Cypress does not include built-in load generation capabilities, it seamlessly integrates with tools such as Gatling or Apache JMeter for more realistic load testing scenarios. Moreover, Cypress's support for custom plugins empowers users to extend its functionality to simulate specific load scenarios, generate reports, or integrate seamlessly with third-party tools, providing unparalleled flexibility and customization options. These

advanced features collectively empower testers to conduct thorough and meaningful evaluations of web applications' performance and scalability. (Cypress 2023a.)

6 SECURITY TESTING

Security testing is a critical aspect of ensuring the robustness and integrity of web applications in today's digital landscape. Cypress, a powerful JavaScript-based testing framework, offers valuable capabilities for conducting security testing of web applications. This involves evaluating the application's vulnerability to various security threats and verifying that it can withstand potential attacks. By leveraging Cypress's features and integration with security testing tools, organizations can enhance their web application's defense against potential security breaches. In this context, security testing with Cypress becomes an integral component of the development and quality assurance process, safeguarding sensitive data and user trust. (Cypress 2023a.)

6.1 Potential for security testing with Cypress

Security testing plays a crucial role in the software development process as it helps identify potential vulnerabilities and weaknesses in web applications that could be exploited by attackers. However, conducting security testing can be challenging due to the need for specialized knowledge and tools. Nevertheless, it is possible to perform security testing using Cypress. (Patel 2023.)

Understand the web application’s functionality and identify potential security vulnerabilities. This involves researching and comprehending the various types of attacks that can target the application, such as across-site scripting (XSS), cross-site request forgery (CSRF), and SQL injection. Create a test plan that includes all the necessary tests cases to check for the identify vulnerabilities. The test plan cover should cover positive and negative scenarios, as well as edge cases and unexpected behavior. (Patel 2023.)

Write automated test for the identified tests cases. Cypress enables developers to create tests that simulate user interactions with the web application, emulating real user behavior. This allows for tests that verify security measures like user input sanitization, encryption, and description of sensitive data and proper handling of session and cookies. Execute the tests and analyze the results. Cypress provides a robust debugging tool that allows developers to step through tests, inspect variables and elements, and even pause and modify the tests execution in real-time. This facilitates the identification and resolution of issues with tests, as well as the discovery of potential security vulnerabilities. (Patel 2023.)

Integrate Cypress with other security testing tools and frameworks to enhance testing capabilities. Cypress.io can be combined with tools such as OWASP ZAP, Burp Suite, and Nessus to augment the security testing process. Regularly perform security testing, as it is an ongoing endeavor. Ensuring the web application's security remains up to date is crucial for safeguarding it against potential threats. (Patel 2023.)

The given example demonstrates how Cypress.io can be used to check for XSS, CSRF, and SQL injection vulnerabilities in a web application. The code includes tests that verify the prevention of script execution from user input, the blocking of unauthorized POST requests, and the absence of errors when handling SQL injection payloads. Cypress is a powerful and versatile testing framework suitable for security testing. Its ability to simulate user interactions and provide robust debugging capabilities makes it a valuable choice for teams aiming to enhance their security testing workflows. (Patel 2023.)

6.2 Relevant Cypress techniques for security testing

All web browsers strictly adhere to the same-origin policy as a default security measure. This principle dictates that browsers will inhibit communication between <iframes> if the origins of these elements are incompatible. The same-origin policy plays a vital role in safeguarding against potential security threats by limiting a script or document's ability from one origin to interact with resources from another origin. While Cypress necessitates continuous direct communication with your web applications, it operates entirely within the confines of the browser. Unfortunately, the default browser settings can obstruct Cypress's functionality in this regard. To circumvent this restriction and align with the same-origin policy's guidelines, Cypress employs various techniques, including JavaScript code execution, manipulation of the browser's internal APIs, and network proxying. These strategies allow Cypress to automate your application comprehensively without necessitating any code modifications on your part. (Hamid 2023.)

One specific challenge Cypress tackles involves managing cookies within cross-origin iframes. To maintain consistent behavior, Cypress ensures that your web app's URL is considered the top URL when running tests. This is facilitated through a behind-the-scenes proxy, which facilitates the seamless attachment and setting of cookies. Additionally, Cypress takes measures such as injecting document.domain into text/html pages, proxying all HTTP/HTTPS traffic, and modifying the hosted URL of the application being tested. These efforts enable Cypress to overcome the initial hurdles posed by the same-origin policy, allowing your application to function as expected. Cypress also goes the extra mile to

support testing of HTTPS sites. It manages and stubs network-level traffic and assigns and manages browser certificates for real-time traffic modification. While Chrome may issue an 'SSL certificate does not match' warning due to Cypress functioning as its own CA authority, this warning is legitimate and applies only to the superdomain being tested. (Hamid 2023.)

Despite Cypress's extensive capabilities, there are still some limitations to be aware of. For instance, all URLs accessed during a single test must share the same superdomain, which can lead to errors if multiple superdomains are accessed within a single test. Cypress strives to enforce this restriction, but there may be scenarios where your application bypasses it, potentially causing issues. Cypress is a powerful tool for web application testing, adeptly handling cross-origin challenges, managing cookies, and supporting HTTPS sites. However, understanding its limitations and applying workarounds when necessary is crucial for effective testing. By doing so, you can leverage Cypress's native access to various browser objects and its stability mechanisms to create robust and reliable tests for your applications. (Hamid 2023.)

7 CHALLENGES OR LIMITATIONS OF USING CYPRESS FOR WEBSITE TESTING

Cypress, a JavaScript-based end-to-end testing framework for web applications, offers unique capabilities but also comes with specific trade-offs. Some permanent trade-offs include Cypress not being a general-purpose automation tool, Cypress commands running exclusively inside a browser, the lack of support for multiple browser tabs, and the inability to drive two browsers simultaneously. Temporary trade-offs include issues that Cypress plans to address in the future, such as the absence of a cy.hover() command and native mobile events support. Cypress excels at end-to-end testing of web applications in the development phase, providing developers and QA engineers with native access to everything within the application under test. However, it requires workarounds for back-end communication and doesn't support multiple tabs or multiple browsers simultaneously, aiming to encourage efficient testing practices that focus on the essential behaviors of the application. (Cypress, 2023e.)

8 CONCLUSION

In conclusion, this thesis has provided a comprehensive exploration of Cypress as a modern JavaScript-based testing framework for website testing. The research has highlighted the importance of software testing and the specific significance of website testing in today's software-driven world. Through an in-depth analysis of Cypress, its architecture, and its unique features, this thesis has shed light on the advantages and disadvantages of using Cypress compared to other testing tools, particularly Selenium. The practical aspects of Cypress, including installation, test writing, execution, reporting, and its potential for security testing, have been thoroughly examined.

The insights and knowledge presented in this thesis aim to empower software developers and testers to leverage Cypress effectively in their testing processes. By adopting Cypress, organizations can enhance the quality, reliability, and security of their web applications, ensuring a positive user experience and mitigating the risks associated with website development. As the field of website testing continues to evolve, Cypress stands as a powerful tool that offers developers and testers a modern and efficient approach to testing web applications. By embracing Cypress and leveraging its capabilities, software professionals can stay ahead in the rapidly changing landscape of software development, delivering high-quality websites that meet user expectations.

This research opens avenues for further exploration and experimentation with Cypress, encouraging continuous improvement and innovation in the field of website testing. By embracing new technologies and methodologies, software professionals can adapt to the ever-growing complexity of web applications and ensure that they are robust, secure, and user-friendly. Furthermore, is important to consider these challenges and limitations when using Cypress for website testing and explore appropriate strategies and solutions to overcome them effectively. There are other challenges that can be found along the way, but despite these limitations, Cypress remains a popular and powerful testing framework for many web applications, offering numerous advantages in terms of ease of use, speed, and reliability.

Finally, this thesis has emphasized the significance of Cypress in website testing, providing valuable insights and practical guidance. By adopting Cypress, software professionals can embark on a journey towards enhanced software quality and improved user satisfaction in the dynamic world of web development.

9 REFERENCES

Abramowski Nicole, 2022. What is NPM? A Beginner's Guide. [Online]. Career Foundry. Last Updated: 28 November 2022. Available at: https://careerfoundry.com/en/blog/web-development/what-is-npm/#:~:text=npm%20stands%20for%20Node%20P [Accessed 3 July 2023].

Amazon Web Services, 2023. What Is An API (Application Programming Interface)?. [Online]. Amazon Web Services. Last Updated: 2023. Available at: https://aws.amazon.com/what-is/api/#:~:text=API%20stands%20for%20Application%20Programming,other%20u [Accessed 3 July 2023].

Ammann P., Offutt J, 2017. *Introduction To Software Testing*. 2nd ed. United States of America: Sheridan Books, Inc. pp.27-39.

Bahmutov G, 2020. Cypress for Load Testing: How to Get Started. [Online]. Cypress. Last Updated: 2020. Available at: https://www.cypress.io/blog/2020/05/06/cypress-for-load-testing-how-to-get-started/ [Accessed 30 June 2023].

Bhat Priyanka & Hegde Ganesh, 2022. Understanding Cypress HTML Reporter. [Online]. BrowserStack. Last Updated: 03 October 2022. Available at: https://www.browserstack.com/guide/cypress-html-reporter [Accessed 28 June 2023].

Bondar Artem, 2023. Cypress: Web Automation Testing from Zero to Hero. [Online]. Udemy. Last Updated: 2023. Available at: https://www.udemy.com/course/cypress-web-automation-testing-from-zero-to-hero/learn/lecture/18179436 [Accessed 22 June 2023].

CyberCraft, 2023. Top 5 web development issues and how companies can avoid them. [Online]. Cyber Craft. Available at: https://cybercraftinc.com/top-5-web-development-issues-and-how-companies-can-avoid-them/ [Accessed 7 June 2023].

Cypress, 2023a. Why Cypress. [Online]. cypress.io. Last Updated: 2023. Available at: https://docs.cypress.io/guides/overview/why-cypress [Accessed 19 September 2023].

Cypress, 2023b. Key Differences. [Online]. Cypress. Last Updated: 2023. Available at: https://docs.cypress.io/guides/overview/key-differences [Accessed 7 June 2023].

Cypress, 2023c. Cypress runs in the browser. [Online]. Cypress. Last Updated: 2023. Available at: https://learn.cypress.io/cypress-fundamentals/cypress-runs-in-the-browser [Accessed 28 June 2023].

Cypress, 2023d. Running Cypress in Continuous Integration (CI). [Online]. Cypress. Last Updated: 2023. Available at: https://learn.cypress.io/advanced-cypress-concepts/running-cypress-in-ci [Accessed 28 June 2023].

Cypress. 2023e. Trade-offs. [Online]. cypress.io. Available at: https://docs.cypress.io/guides/references/trade-offs [Accessed 19 September 2023].

Fosco Molly, 2022. What is a CI/CD pipeline? [Online]. Circle CI. Last Updated: 06 October 2022. Available at: https://circleci.com/blog/what-is-a-ci-cd-pipeline/?utm_source=google&utm_medium=sem&utm_campaign=se [Accessed 3 July 2023].

GeeksforGeeks, 2023. HTML Introduction. [Online]. Geeksforgeeks. Last Updated: 08 May 2023. Available at: https://www.geeksforgeeks.org/html-introduction/ [Accessed 29 August 2023].

GeeksforGeeks, 2022. Software Testing | Basics. [Online]. GeeksforGeeks. Last Updated: 28 June 2022. Available at: https://www.geeksforgeeks.org/software-testing-basics/ [Accessed 3 September 2023].

Hamid Akhtar, 2023. Maximizing Web Security with Cypress: A Step-by-Step Guide. [Online]. browserstack. Last Updated: 10 March 2023. Available at: https://www.browserstack.com/guide/cypress-web-security [Accessed 19 September 2023].

Jiujiu Yu, 2019. Exploration on Web Testing of Website. Journal of Physics: Conference Series. 1176(2). [Online]. Available at: https://www.researchgate.net/publication/331904112_Exploration_on_Web_Testing_of_Website [Accessed 2 June 2023].

Kalaton, 2023. What is Web Testing? Types of Web Application Testing. [Online]. Kalaton. Last Updated: August 2023. Available at: https://katalon.com/resources-center/blog/types-of-web-testing [Accessed 29 August 2023].

Lee Terry G. et al., 2023. Code snippets. [Online]. Microsoft. Last Updated: 10 March 2023. Available at: https://learn.microsoft.com/en-us/visualstudio/ide/code-snippets?view=vs-2022 [Accessed 3 July 2023].

Loshin Peter et al., 2022. Structured Query Language (SQL). [Online]. Tech Target. Last Updated: February 2022. Available at: https://www.techtarget.com/searchdatamanagement/definition/SQL [Accessed 3 July 2023].

Noor E. Alam Robin, 2023. Software Testing Goals Based on Test Process Maturity. [Online]. Knowledge Sharing. Available at: https://ksharing.info/software-testing-goals-based-on-test-process-maturity/ [Accessed 31 May 2023].

Patel Parita, 2023. Security Testing using Cypress. [Online]. Linkedin. Last Updated: 29 January 2023. Available at: https://www.linkedin.com/pulse/security-testing-using-cypress-parita-patel/ [Accessed 3 July 2023].

Ravinder Singh, 2021. Cypress Assertions. [Online]. Tools QA. Last Updated: 23 August 2021. Available at: https://www.toolsqa.com/cypress/cypress-assertions/ [Accessed 28 June 2023].

Rubin Oren, 2021. Performance Testing with Cypress. [Online]. Cypress. Last Updated: 2021. Available at: https://www.cypress.io/blog/2021/05/24/performance-testing-with-cypress/ [Accessed 30 June 2023].

Schmitt Jacob, 2023. What is end-to-end testing? [Online]. Circle CI. Last Updated: 06 April 2023. Available at: https://circleci.com/blog/what-is-end-to-end-testing/#c-consent-modal [Accessed 3 July 2023].

Sthapit Abiral, 2021. Maintainable Tests — Cypress.io. [Online]. Medium. Last Updated: 01 March 2021. Available at: https://abrialstha.medium.com/maintainable-tests-cypress-io-df2f859c846c [Accessed 28 June 2023].

Synopsys, 2023a. Cross Site Request Forgery. [Online]. Synopsys. Last Updated: 2023. Available at: https://www.synopsys.com/glossary/what-is-csrf.html#:~:text=Definition,which%20they%20are%20currentl [Accessed 3 July 2023].

Synopsys, 2023b. Cross-Site Scripting (XSS). [Online]. Synopsys. Last Updated: 2023. Available at: https://www.synopsys.com/glossary/what-is-cross-site-scripting.html#:~:text=Cross%2Dsite%20scripting [Accessed 3 July 2023].

Tutorial Hut, 2023. Basic Commands used is Cypress. [Online]. Tutorials Hut. Last Updated: 2023. Available at: https://tutorialshut.com/basic-commands-used-in-cypress/ [Accessed 28 June 2023].

Tutorials Point, 2023. Software Quality Management. [Online]. Tutorials Point. Available at: https://www.tutorialspoint.com/software_quality_management/software_quality_management_factors.htm# [Accessed 2 June 2023].

Unadkat Jash, 2023. What is Test Driven Development (TDD)?. [Online]. BrowserStack. Last Updated: 14 June 2023. Available at: https://www.browserstack.com/guide/what-is-test-driven-development#:~:text=Driven%20Development%20(T [Accessed 3 July 2023].

Wilkins Jessica, 2021. What is the DOM? Document Object Model Meaning in JavaScript. [Online]. freeCodeCamp. Last Updated: 27 September 2021. Available at: https://www.freecodecamp.org/news/what-is-the-dom-document-object-model-meaning-in-javascript/ [Accessed 3 July 2023].