IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Technical Analysis of Cypress Using Modern Web Application

Mohit Singh[1], Sanjana Gupta[2], Rohit Singh[3],
[1,2] Student, Department of CSE (*AI &ML*), ABES Engineering College, Ghaziabad
[3]Assistant Professor, Department of CSE (*AI &ML*), ABES Engineering College, Ghaziabad, India

Abstract : The increasing demand for automation testing is driven by the unique challenges posed by current software development projects. These projects involve intricate applications with tight development timelines. Many businesses in the sector have turned largely to Selenium as their main functional automation technology to meet these problems and make sure their web applications operate as intended. However, Selenium Limitations, particularly in handling wait times have had a notable impact on test execution and efficiency. To address this issue, this research project is venturing into the realm of a new automation tool, Cypress, which has recently emerged in the market. The primary aim of this research is to evaluate Cypress's capabilities in overcoming the limitations experienced with Selenium. The project involves a comprehensive evaluation of Cypress to assess its effectiveness in script development and test automation for modern web applications. The findings of this study are anticipated to offer valuable insights into the selection of a more suitable tool for automating dynamic, modern web applications. Additionally, it will provide a deeper understanding of Cypress's potential role as the future frontrunner in the field of automation tools.

IndexTerms **- Cypress, Automation Testing Tool, Selenium, Software Testing, Modern web applications, New automation tool.**

INTRODUCTION

Web-based applications that digitize everything from government to retail operations are growing more and more popular in today's technologically advanced society, where information can be accessed with just a click. With more features and dynamic capabilities to offer a richer user experience, web apps have become more sophisticated. This makes testing contemporary online apps to satisfy user expectations extremely difficult. Automated testing is becoming more popular due to complicated web applications and faster product release schedules. In order to decrease execution time and improve testing accuracy, automated testing makes use of tools that simulate the actions of actual users by executing tests independently. Thirty to sixty percent of the software lifecycle is devoted to testing, with a larger percentage going toward the more intricate and crucial parts [1]. Due to lengthy development timelines, complicated websites, and shot lead times, modern web testing needs to be done quickly. As stated [2], when end users try to access 29 out of 40 e-commerce sites, they encounter errors. The majority of businesses in the sector employ Selenium to work on performance verification apps in order to address this issue. Nonetheless, in 2005, when Selenium was developed, webpages were simpler than they are in 2023. The primary drawback of Selenium is the challenge of maintaining current online material, which lowers test success and introduces instability into testing. In order to address the needs of current dynamic online applications, this work provides Cypress as an automation framework. Cypress offers the primary benefit of simple async testing. Within its framework, Cypress defines an automated wait that delays initiating a web search until a DOM element has finished loading or an operation has finished. Automation developers in Selenium are forced to wait for the page to completely load, which is a significant limitation. Experiments can add threads or wait times. To fix this issue, use sleep directives; however, this will impact how the test is executed. Cypress's primary limitation is that it is mostly compatible with the Chrome web browser. Nonetheless, the limitations of browser compatibility will be minimal at this point because, according to Google Trends, 90% of users use Chrome now.

The Kips Health Dynamic HealthCare Web Application is automated by this effort. Kips Health was selected because to its numerous dynamic themes and pictures, as well as the fact that it has had thousands of visits from users worldwide, indicating its dependability.

The following are the objectives of this study, which is centered on Cypress:
i. To use Cypress to write an automation script.
ii. To create regression automation suites for the main patient accounts and report flows workflows at Kips Health.
iii. To use Cypress to compare test execution times.
iv. To contrast Cypress's test coverage and efficiency.

OBJECTIVE

In a Cypress automation project, a variety of tools and techniques are deployed for both development and analysis to ensure the robustness and effectiveness of the testing process. Cypress, as the core testing framework, serves as the foundation for creating and executing end-to-end tests. The Cypress Command Line Interface (CLI) enables test execution and configuration management, while integrated test runners like Mocha or Jasmine help organize and structure test suites for more comprehensive reporting. Git and other version control systems are necessary for code management and collaborative development.

Continuous Integration (CI) tools such as Jenkins or GitLab CI facilitate automated testing on code commits, ensuring seamless integration with the development pipeline. Moreover, Docker containers are utilized to create consistent and reproducible testing environments. IDEs like Visual Studio Code simplify script development, while the Page Object Model (POM) design pattern enhances code organization. Custom commands and assertions streamline scripting and validation. Data-driven testing leverages test data management tools and reporting and analysis tools such as Mochawesome provide insights into test performance and issues. Debugging tools, linters, formatters, and supplementary services like cross-browser testing and load-testing solutions round out the toolkit. Collectively, these tools and techniques empower Cypress projects to deliver reliable and comprehensive testing, ultimately leading to better software quality.

LITERATURE REVIEW

Notable investigations have already been done in the past. This was done to enhance the effectiveness of web testing with the use of test automation techniques. One of the most popular automation tools in the past was Selenium. Powerful and dependable is the reputation of selenium. For the past ten years, automation tools have been available on the market. But as recent research by [3], [4], [5], as well as automation developers, has demonstrated, Selenium has serious shortcomings when it comes to handling pop-ups, dynamic content generation, and page loading in contemporary online applications. As mentioned in [6], this problem has not been fixed before and affects the stability and performance of Selenium scripts. In light of this, this paper presents the Cypress automation framework that incorporates Test Driven Development (TDD) methods and Page Object model (POM) design. TDD is either a functionality testing method. Developers write automated code for exploratory testing, which is the primary focus of this process. After testing, create functional code. Establish basic guidelines to enhance your measures utilizing TDD, as demonstrated by recent research in [7] and [8]. Control and accuracy go hand in hand, as does precision. [8] adds that TDD lowers automation code and enhances test execution by 21%. The level of difficulty has dropped by 31%. POM is a relatively abstract term that divides web pages at the same time. makes sure that code is reused in all test cases. to Cut down on the quantity of links between It is possible to create web pages and test pages that are separate from one another and to facilitate the reuse of various code segments. Test data recording is also simpler. POM applications are used [3]. Combining the TDD method with POM design, the future has been shortened in proportion to the tests' increased manageability. expenses for upkeep. Automation scripts need to have clean code because it takes a lot of time and effort to fix faulty code [9]. In the meantime, this study's measurement techniques are as follows. Performance-based measurement based on test time. Based on the research conducted by [10], [11], and [12], these indicators are employed in this study. These three factors are regarded as the initial markers of an exam's success. It is possible to maintain the quality of the application under test (AUT) if these three criteria are appropriately managed. Agile methodologies are also employed in the study project. Agile is a process that is incremental and iterative that helps speed up release cycles and increase operational efficiency. Scrum is the most agile approach currently available [13]. Scrum can maximize research and produce high-quality research outcomes, which is a benefit for research projects [14].

PROPOSED METHODOLOGY

Scrum is an agile project management technique used in this study project. Despite Scrum's widespread use in the IT sector, important studies [15–16], [17–18], and [19] have demonstrated that incorporating Scrum into research programs improves the caliber of the research output. Additionally, he produced a study [16] demonstrating its use in construction management.

- **Roles**: Research Project, Research Team, and Supervisor (Scrum Master).
- **Artifacts**: Backlog Analysis and Sprint Backlog.
- **Ceremonies:** Weekly Scrum Meeting, Sprint Review, Sprint Review.

Justification:

The role of the Scrum Master is to oversee the Scrum team, reporting to the manager and finding a workable solution. The target Scrum Master is the person who oversees every issue that arises during the process. Meanwhile, the individual who comprehends the requirements and particulars of this research is known as the Research Owner and is described as the Product Owner. The person performing the research and the research team will be the same since this study is being reviewed individually.

However, as daily stand-up meetings are impractical for research projects, weekly meetings might occasionally be more frequent than daily ones because of various software development research projects and the supervisor's need to complete ten or more tasks at once.

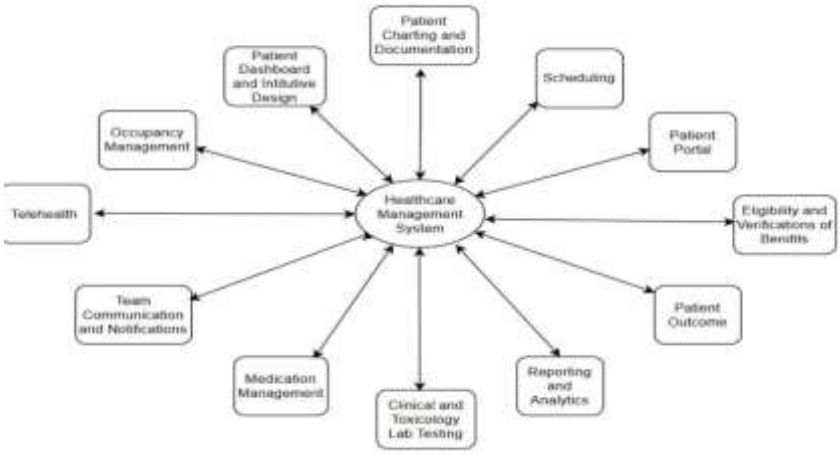

Fig 1. DFD LEVEL 1 DIAGRAM

Fig 1: Shows the various components and functionalities of the HealthCare web application..

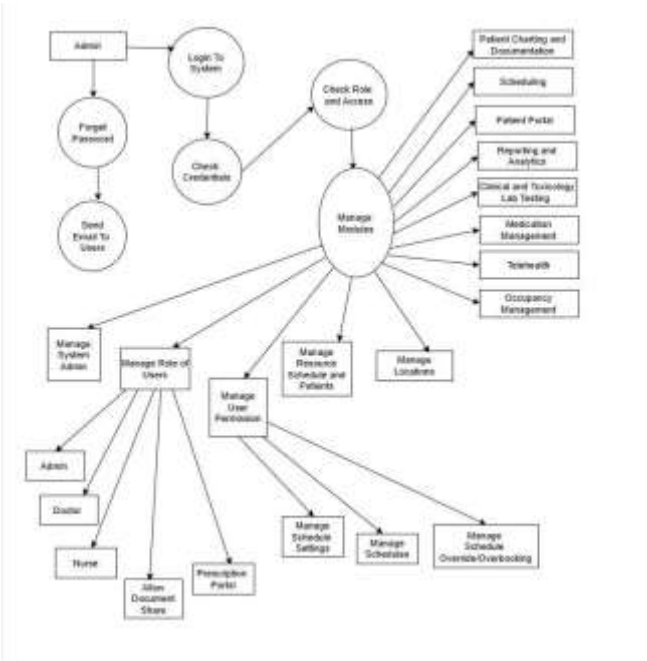

Fig 2. DFD LEVEL 2 DIAGRAM

Fig 2 represents-:

1. Request to Sign Up or Login.
2. Request login information or prompt details for a new user.
3. Provide information or credentials.
4. Check the login credentials of the current user.
5. Store the login credentials for the new user
6. Verification of credentials.
7. A successful login.
8. New users inquire about roles and access.
9. The user interface for current users displays management modules.
10. Keeping track of updated user permissions.
11. Saved preferences.
12. The home page appears.
13. Use the search bar to find any related functionality.
14. Get Patient information.
15. Create a Patient Appointment.
16. Manage Resource Schedule.
17. Make changes in the communication for appointments.
18. Choosing Patient Medication based on suggestions.
19. History is maintained for improved system accuracy and feedback.

EXPECTED OUTCOMES

The expected outcome of a Cypress project is to provide a structured, efficient, and reliable way to automate end-to-end testing for web applications. As the project grows, Cypress offers a project structure out of the box, accommodating different files that are introduced. It aims to keep all tests and all branches in sync with development, making it easier to do continuous delivery.

One of the key principles of Cypress is that readability is the most important decision-maker. If a failed test does not provide enough information on what happened or why it happened, then the test did not do its job effectively. Therefore, when writing tests, test readability should guide all test design decisions.

Another expected outcome is that testing will improve delivery speed. Slow debugging and slow testing mean slow delivery, so test automation must be optimized for speed.

Finally, human time is more valuable than machine time. It's important to pick your battles and ensure that creating test automation actually saves you time. Building anything requires maintenance, and it's especially important to consider this when choosing between building a tool and paying for a solution.

REFERENCES

[1] Dautov, R., & Parıltı, N. (2020). Automated Web Testing with Selenium and Cypress. International Journal of Intelligent Systems and Applications in Engineering, 8(2), 50-54.
[2] Agarwal, A., & Garg, S. (2022). Cypress.io: An Evaluation of Its Capabilities and Limitations for End-to-End Web Testing. Software Quality Journal, 30(2), 539-562.
[3] Venkatraman Rajaram, G.N. Shah, Ronald Mueller, "Cypress Web Automation: Its Expanding Role in Microsoft's Web App Development Projects.
[4] Rafi, D. M., Moses, K. R. K., Petersen, K., & Mäntylä, M. V. (2019). Benefitting from Selenium and Cypress: A case study exploring automated testing of web applications. (pp. 241-248).
[5] Zhong, M., & Zhang, X. (2020). A Comparative Study of Selenium and Cypress for Automated Web Testing.Engineering and Technology, (pp. 515-519).
[6] Bajpai, N., & Jalali, S. (2021). Selenium vs Cypress: A Comprehensive Evaluation for Automated Web Testing (pp. 201-208).
[7] Zhu, J., & Li, P. (2020). Exploring the Use of Selenium and Cypress for Cross-Browser Testing. (pp. 119-126).
[8] Nguyen, B. N., Robbins, J. E., Banerjee, I., & Memon, A. (2022). Selenium or Cypress? An Empirical Study of Automated Web Testing Tools. (pp. 1-20.)
[9] Singh, S., & Kaur, G. (2020). A Comparative Analysis of Selenium and Cypress for Automated Web Testing. (pp 232-238.)[10] Jyolsna Thekkan Othayoth, Syahid Anuar," Modern Web Automation with Cypress.Io",
[11] Fernández-Sánchez, C., Garbajosa, J., & Yagüe, A. (2020). A Study on the Use of Cypress.io for End-to-End Testing of Web Applications. Journal of Software: Practice and Experience, 50(10), 1785-1806.
[12] Rossi, G., & Russo, B. (2020). Migrating from Selenium to Cypress.io: A Case Study in Test Automation for Web Applications. Journal of Software: Evolution and Process, 32(7), e2290.
[14] Bose, S., & Roy, A. (2021). Cypress: Redefining end-to-end testing for the web. IEEE Software, 38(4), 58-64.
[15] Vaish, R., & Gupta, M. (2020). Evaluating Cypress for end-to-end testing of modern web applications. (pp. 155-162).
[16] Parra, A., Font, J., & Méndez, D. (2020). Exploring the Benefits of Combining Cypress.io and Docker for Automated Web Testing. Journal of Web Engineering, 19(5-6), 885-910.
[17] Kumar, A., & Bansal, R. (2022). Cypress: A comprehensive evaluation for end-to-end testing. (pp. 217-224).
[18] Chaudhary, M., & Singh, Y. (2021). Cypress: A comprehensive evaluation for end-to-end web testing. IEEE Transactions on Software Engineering, 47(9), 1949-1967.