E2EGit: A Dataset of End-to-End Web Tests in Open Source Projects

Sergio Di Meglio §, Luigi Libero Lucio Starace §, Valeria Pontillo ¶, Ruben Opdebeeck ¶,
Coen De Roover ¶, Sergio Di Martino §
§ Department of Electrical Engineering and Information Technology, University of Naples Federico II, Italy
Email: (sergio.dimeglio, luigiliberolucio.starace, sergio.dimartino)@unina.it
¶ Software Languages (SOFT) Lab, Vrije Universiteit Brussel, Belgium
Email: (valeria.pontillo, ruben.denzel.opdebeeck, coen.de.roover)@vub.be

***Abstract*—End-to-end (E2E) testing is a software validation approach that simulates realistic user scenarios throughout the entire workflow of an application. In the context of web applications, E2E testing involves two activities: Graphic User Interface (GUI) testing, which simulates user interactions with the web app's GUI through web browsers, and performance testing, which evaluates system workload handling. Despite its recognized importance in delivering high-quality web applications, the availability of large-scale datasets featuring real-world E2E web tests remains limited, hindering research in the field.**

To address this gap, we present E2EGit, a comprehensive dataset of non-trivial open-source web projects collected on GITHUB that adopt E2E testing. By analyzing over 5,000 web repositories across popular programming languages (JAVA, JAVASCRIPT, TYPESCRIPT and PYTHON), we identified 472 repositories implementing 43,670 automated Web GUI tests with popular browser automation frameworks (SELENIUM, PLAYWRIGHT, CYPRESS, PUPPETEER), and 84 repositories that featured 271 automated performance tests implemented leveraging the most popular open-source tools (JMETER, LOCUST). Among these, 13 repositories implemented both types of testing for a total of 786 Web GUI tests and 61 performance tests. The dataset is available on ZENODO (DOI: 10.5281/zenodo.14234731).

***Index Terms*—End-to-End Testing, Web GUI Testing, Performance Testing, GitHub-Minining, Web Applications**

I. INTRODUCTION

End-to-End (E2E) testing is essential for ensuring the reliability and quality of software applications, especially for web systems used by millions of people across industries like commerce, banking, and others [1], [2]. By validating end-to-end workflows from the user's perspective, E2E testing ensures that all components of an application work together seamlessly, providing a smooth and reliable user experience [3], [4]. This type of testing includes two main activities: Graphical User Interface (GUI) testing and performance testing.

The former involves simulating user interactions with the application through a browser, such as navigating between pages, submitting forms, or clicking buttons [1]. The latter measures how well a system handles varying workloads. In web contexts, these workloads are defined by user sessions, which consist of a series of network requests (such as HTTP, WSS) made during interactions with the system [5], [6]. While both testing practices can be performed manually, this approach is often prone to errors and less effective in replicating complex, real-world scenarios —especially for performance testing. To overcome these limitations, over the last few years, frameworks and tools for automation have been proposed [7]. For Web GUI testing, popular browser automation frameworks like SELENIUM, CYPRESS, PLAYWRIGHT, and PUPPETEER are used to control browsers and simulate user actions [8], [9]. In the performance testing domain, open-source tools like APACHE JMETER and LOCUST are widely used to simulate web server workloads, which are typically defined by one or more user sessions (e.g., Thread Group in APACHE JMETER and TaskSet in LOCUST) that simulate several concurrent users adopting the same behavior over time [10], [11].

Despite end-to-end (E2E) testing being crucial for ensuring high-quality software and delivering a positive user experience, it is often overlooked in practice [4]. This neglect is mirrored in the literature, where the limited availability of datasets in the context of Web GUI and performance testing has hindered research progress in these areas [12]. Although some Web GUI test datasets do exist, they are often based on small-scale projects such as university assignments, or they only include projects that use a particular framework or a particular programming language. For performance testing, to the best of our knowledge, no dataset has been published so far, probably because companies rarely share such tests publicly [13], [14].

This paper aims to fill this gap by presenting E2EGit [15], a comprehensive dataset of non-trivial web projects hosted on GITHUB that adopt E2E testing practices. To this end, we analyzed 5,563 non-trivial web repositories developed using at least one of the most popular programming languages for web development, i.e., JAVA, JAVASCRIPT, TYPESCRIPT, and PYTHON. Each repository was processed to identify 1) the presence of Web GUI tests using the four most used browser automation frameworks, i.e., SELENIUM, PLAYWRIGHT, CYPRESS, and PUPPETEER; and 2) the presence of performance tests conducted with the most popular open-source tools, JMETER and LOCUST [13], [16].

At the end of the mining process, we identified 472 repositories containing Web GUI tests, comprising 43,670 test cases, and 84 repositories containing performance tests, totaling 271 test cases. Among these, 13 repositories implemented both

This work has been partially supported by the Italian PNRR MUR project PE0000013-FAIR.

types of testing, contributing 786 Web GUI tests and 61 performance tests, which are included in the respective totals.

II. Research Method

This section describes the mining pipeline shown in Figure 1 for collecting and analyzing repositories from GitHub. In the first step, repositories are collected and filtered to identify web repositories with non-trivial complexity. In the second phase, two separate analyses are performed after locally cloning the repositories to identify GUI and performance tests.

A. Data Collection

For collecting the repository, we used the SEART tool[1], developed by Dabic et al. [17], which is widely used in the literature for mining GitHub repositories. At the time of collection in May 2024, SEART reported a subset of 1,401,170 repositories. Due to the large amount of repositories, we applied a filtering process to exclude "toy" projects. This filtering phase followed well-established criteria from prior research [18]–[20], specifically:

- *Number of commits*: greater than or equal to 2000;
- *Number of contributors*: greater than or equal to 10;
- *Number of stargazers*: greater than or equal to 100;
- *Is fork*: must be False;

These criteria reduced the sample to 14,053 repositories. Since our study refers to web applications, we further refined the selection by including only non-trivial repositories that had at least one dependency on a web framework. However, given the wide variety of web programming technology, our dataset needs to be restricted to the most widely adopted frameworks (e.g., Spring Boot, React and Angular according to [21]; see the online appendix [15] for the full list of considered frameworks) for each of the four most popular programming languages in web development (i.e., JavaScript, TypeScript, Python and Java according to [22]). After this filtering, we are left with 5,563 web applications.

B. Web GUI Test Search

This phase consists of three steps and it was applied to each web application repository to identify Web GUI tests:

1) *Checking dependencies.* We analyzed the projects' dependencies to identify the use of browser automation frameworks, focusing on the most commonly used ones, according to [23], namely Selenium, Playwright, Puppeteer and Cypress. The search was done by extracting dependencies from dependency management files: `pom.xml` and `build.gradle` for projects in Java, `package.json` for JavaScript and TypeScript, and `requirements.txt` for Python.
2) *Finding Web GUI test files.* If the repository includes at least one of the above dependencies, we have identified potential Web GUI tests, i.e., files that import browser automation frameworks.
3) *Validation of Web GUI tests.* We automatically analyzed each candidate GUI test source file to determine whether the identified automation framework dependencies were used for web GUI testing. Specifically, we verified the presence of a test engine alongside the implemented test code. Our focus was on widely-used test engines [24], including JUnit and TestNG for Java; Jest, Mocha, and Jasmine for JavaScript and TypeScript; and PyTest and UnitTest for Python. Each test engine has specific syntax, such as `@Test` in JUnit and TestNG, `it` or `test` in Jest, Mocha, and Jasmine, and `test_` in PyTest and UnitTest.

This process allowed us to identify 472 repositories containing at least one implemented Web GUI test.

C. Performance Test Search

This phase, performed in parallel to the previous one, aims instead to search for performance tests. We focused on Locust and Apache JMeter, as they are recognized as the most representative open-source performance testing tools, widely adopted in both academic research and enterprise contexts [13], [16]. The identification of performance tests involved distinct processes tailored to each tool:

- Locust. As a Python-based framework, Locust must be explicitly imported into the test code. The process begins by checking for the Locust dependency in the `requirements.txt` file. When the dependency is found, the pipeline searches for Python files within the repository that import Locust.
- Apache JMeter. The identification process for JMeter is straightforward, as it does not require dependencies and import analyses. Indeed, JMeter tests are stored as files with a distinctive `.jmx` extension. To identify performance tests created with JMeter, we simply searched for files with the `.jmx` extension.

In the final step, *Performance Test Validation*, we verified whether the identified test artifacts were genuinely intended for performance testing. To this end, we ensured that each test included at least one `ThreadGroup` or `TaskSet` and at least one network request associated with it. For JMeter, this validation was conducted automatically by parsing and analyzing the `.jmx` test files. For Locust, the analysis was performed manually by the first two authors of this paper. After validation, we identified 84 repositories containing at least one implemented performance test.

III. E2EGit Dataset

The E2EGit dataset comprises 43,670 Web GUI tests from 472 repositories and 271 performance tests from 84 repositories. Notably, the dataset also includes 13 repositories that integrate both types of testing, featuring 786 Web GUI tests and 61 performance tests. On average, repositories with Web GUI testing contain 92.5 tests, while those with performance testing feature an average of 5 tests. Table I provides an overview of the dataset based on the analyzed frameworks and tools.

[1] https://seart-ghs.si.usi.ch/

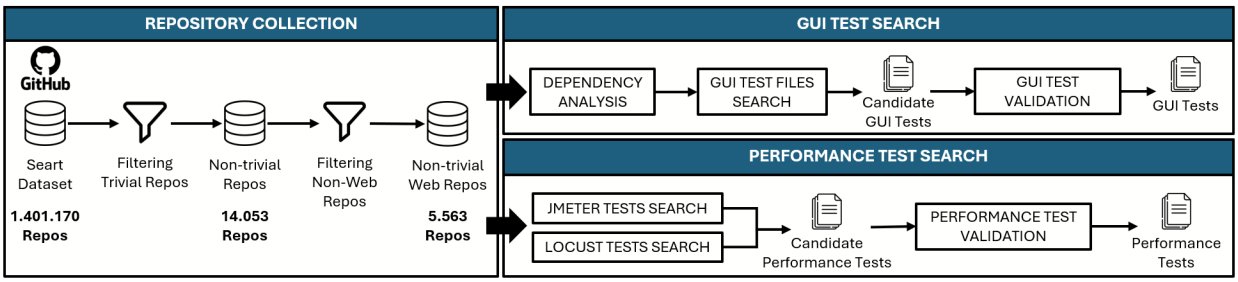

Figure 1: Overview of the mining pipeline

Table I: Overview of repositories and tests collected in E2EGit, categorized by the frameworks and tools used for Web GUI testing and performance testing. Notably, some web applications adopt multiple frameworks for Web GUI tests, which results in overlapping counts across categories.

Web GUI Testing		
Browser Automation Framework	**Num. of Repos**	**Num. of Tests**
SELENIUM	87	10,464
PLAYWRIGHT	197	18,175
PUPPETEER	20	302
CYPRESS	187	14,733
Performance Testing		
Load Generator Tool	**Num. of Repos**	**Num. of Tests**
APACHE JMETER	72	244
LOCUST	13	27

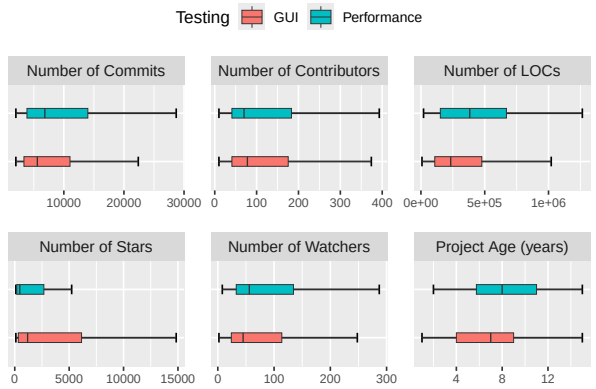

Figure 2: Distribution of Repository Characteristics for Projects with Web GUI or Performance Testing

The repositories exhibit diverse characteristics, as highlighted by the boxplots in Figure 2. Project age ranges from newly initiated to well-established, decade-old ones, while the lines of code per project vary from a few thousand to over a million. Activity indicators such as the number of commits and contributors also differ significantly. Some projects are highly active and collaborative, involving hundreds of contributors, while others exhibit minimal maintenance.

The dataset is stored as an SQLite database and its schema is reported in Figure 3. The **repository** table contains information on 1.5 million repositories collected using SEART in May 2024, with 34 fields detailing repository characteristics, including the number of commits, number of contributors, and last commit on which our mining pipeline was executed. The subset **non_trivial_repository** table includes repositories that passed the two filtering stages described in Section II. For each repository, we specify the use of frameworks for JAVA, JAVASCRIPT, TYPESCRIPT, or PYTHON. For repositories using multiple frameworks, the corresponding fields (e.g., `is_web_java`) are set to `true`, and the `web_dependencies` field lists all the detected web frameworks.

Details about the detected web GUI tests are reported in the **gui_testing_test_details** table, providing information at the test file level. Each row represents a test file detailing the file path, the browser automation framework used, the test engine employed, and the number of tests implemented in the file. For ease of analysis, the final dataset also includes a **gui_testing_repo_details** (not shown in Figure 3 for brevity), aggregating web GUI test data at repository level. For each repository with web GUI tests, this table summarizes the number of test files using each considered framework, the employed test engines, and the total number of identified tests.

Concerning performance testing, the **performance_testing_test_details** table consists of 410 rows, one per each identified `ThreadGroup` or `TaskSet`. This table includes details such as the file path, whether the test uses APACHE JMETER or LOCUST, and extracted parameters like the number of thread groups, concurrent users, and requests. It is worth noting that some fields may be absent. For example, external files (e.g., CSVs defining workloads) might be unavailable, or in the case of LOCUST tests, parameters like duration and concurrent users may be specified via the command line rather than within the test files themselves. Moreover, some fields (e.g., `number_of_users`) may contain parametric expressions of the form `${foo}`, referring to specific environment variables which we were not able to determine during the mining process. In this case, the field contains the parametric expression as a string.

IV. POSSIBLE USES OF THE E2EGIT DATASET

The E2EGit dataset is particularly valuable because it contains a large number of tests from non-trivial repositories with an extensive commit history that make use of diverse web and

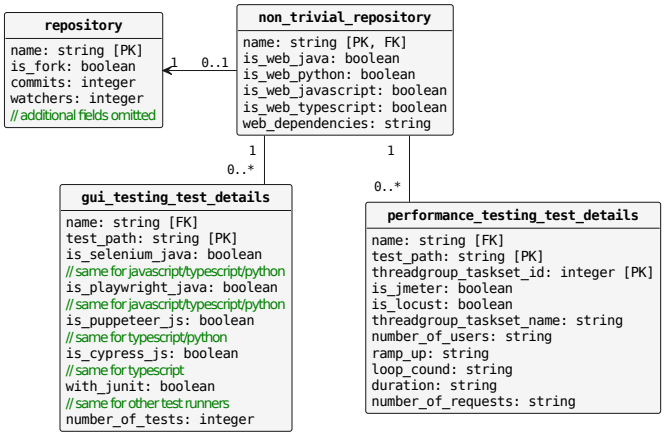

Figure 3: E2EGit dataset schema

automation frameworks. This makes it suitable for a variety of purposes.

One key area of exploration is test quality and maintenance. Researchers can investigate the presence of "test smells" [25], such as redundancy, fragility, or poor structure, to assess the overall quality of tests. This analysis can guide the application of refactoring techniques to improve test design, maintainability, and robustness. For instance, the dataset could enable studies on how systematic refactoring of tests can mitigate identified smells, reduce maintenance overhead, and enhance the stability of test suites over time. Additionally, the dataset could support studies on the impact of application code changes on tests, revealing patterns in how code modifications affect test stability and reliability.

Furthermore, the dataset could also be a valuable resource for creating specialized datasets to train machine learning models and Large Language Models (LLMs). This is particularly relevant for advancing research in areas like automatic test generation and repair, which are still in their early stages [26]. Furthermore, E2EGit can aid research requiring systems with multiple versions and identified bugs for evaluation, such as test case prioritization and regression testing. Existing studies often rely on outdated applications [12], making E2EGit a more current and robust alternative.

From an industry perspective, E2EGit could provide valuable insights for performance testing, offering examples of how workloads are structured for different types of applications. This information can help refine performance testing strategies for similar projects [27]. Finally, the dataset can be leveraged for benchmarking testing tools, enabling comparisons of browser automation frameworks and load generation tools to evaluate their performance in real-world applications.

V. Threats to Validity

Internal Validity. The main internal threat is the selection of web frameworks and E2E testing tools, which may have led to the exclusion of some repositories or tests. However, we included the four most popular programming languages [22], as well as the 15 widely used web frameworks for each [21], and the most prominent browser automation and load generation tools [13]. We are confident the dataset reflects the current state of E2E testing adoption on GitHub.

External Validity. A key threat is the focus on open-source projects from GitHub, which represents only a subset of the broader software ecosystem. As a result, the findings may not generalize to other contexts, such as industrial projects. To address this, all materials have been made publicly available to encourage further research and validation in diverse settings [15].

VI. Related Work

This section examines the literature on dataset availability for Web GUI and performance testing in web applications, highlighting significant gaps. For performance testing, to the best of our knowledge, no studies have focused on creating public datasets, likely due to companies keeping such tests private [13], [14].

In contrast, some efforts have been made for Web GUI testing. Christophe et al. [28] in a dated study investigated the prevalence of Web GUI testing written with Selenium in open-source Java projects, making the data available. Gortazar et al. [29] proposed collecting GUI test datasets from GitHub using 12 simple naming criteria, resulting in 114 projects. However, their methodology lacked explicit verification steps, risking false positives. Sanchez et al. [30] built a small dataset with six bugs and a few GUI tests across three student-developed applications. Fuad et al. [31] introduced the WebEV dataset, comprising Web GUI tests from 100 GitHub projects. Their work, however, focused solely on the Cypress framework, the dataset is currently unavailable, as the repository remains empty at the time of writing. Our dataset aims to fill this gap by offering a comprehensive collection of repositories with automated tests for Web GUI and performance testing.

VII. Conclusion

End-to-end testing is crucial for ensuring software quality, especially in web applications. However, the lack of comprehensive datasets in this area has hindered progress in both academic research and industry practices. To address this gap, we released the E2EGit dataset, which includes 472 non-trivial web repositories with 43,670 Web GUI tests, and 84 repositories featuring 271 performance tests. Among these, 13 repositories implemented both types of testing, with 786 Web GUI tests with 90 performance tests [15].

We believe E2EGit serves as a valuable resource for researchers and practitioners. It could support studies on the evolution and maintenance of E2E tests, their fragility, the relationship between code and test modifications, and patterns in performance test workloads. Additionally, the dataset has the potential to drive advances in test automation, such as automatic test generation and repair, and to aid in the training of machine learning models and domain-specific LLMs.

As part of our future agenda, we plan to expand and update the dataset by including new repositories and supporting additional frameworks and tools.

References

[1] M. Leotta, B. García, F. Ricca, and J. Whitehead, "Challenges of end-to-end testing with selenium webdriver and how to face them: A survey," in *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, 2023, pp. 339–350.

[2] A. Corazza, S. Di Martino, A. Peron, and L. L. L. Starace, "Web application testing: Using tree kernels to detect near-duplicate states in automated model inference," in *Proceedings of the 15th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2021, pp. 1–6.

[3] S. Di Meglio and L. L. L. Starace, "Towards predicting fragility in end-to-end web tests," in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 387–392. [Online]. Available: https://doi.org/10.1145/3661167.3661179

[4] M. Leotta, D. Clerissi, F. Ricca, and P. Tonella, "Approaches and tools for automated end-to-end web testing," in *Advances in Computers*. Elsevier, 2016, vol. 101, pp. 193–237.

[5] D. A. Menascé, "Load testing of web sites," *IEEE internet computing*, vol. 6, no. 4, pp. 70–74, 2002.

[6] A. van Hoorn, M. Rohr, and W. Hasselbring, "Generating probabilistic and intensity-varying workload for web-based software systems," in *Performance Evaluation: Metrics, Models and Benchmarks*, S. Kounev, I. Gorton, and K. Sachs, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 124–143.

[7] Z. M. Jiang and A. E. Hassan, "A survey on load testing of large-scale software systems," *IEEE Transactions on Software Engineering*, vol. 41, no. 11, pp. 1091–1118, 2015.

[8] B. García, M. Gallego, F. Gortázar, and M. Munoz-Organero, "A survey of the selenium ecosystem," *Electronics*, vol. 9, no. 7, p. 1067, 2020.

[9] B. García, J. M. del Alamo, M. Leotta, and F. Ricca, "Exploring browser automation: A comparative study of selenium, cypress, puppeteer, and playwright," in *International Conference on the Quality of Information and Communications Technology*. Springer, 2024, pp. 142–149.

[10] S. D. Meglio and L. L. L. Starace, "Evaluating performance and resource consumption of rest frameworks and execution environments: Insights and guidelines for developers and companies," *IEEE Access*, pp. 1–1, 2024.

[11] S. Di Meglio, L. L. L. Starace, and S. Di Martino, "Starting a new rest api project? a performance benchmark of frameworks and execution environments." in *IWSM-Mensura*, 2023.

[12] S. Balsam and D. Mishra, "Web application testing—challenges and opportunities," *Journal of Systems and Software*, vol. 219, p. 112186, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121224002309

[13] M. Yenugula, R. Kodam, and D. He, "Performance and load testing: Tools and challenges," *International Journal of Engineering in Computer Science*, vol. 1, pp. 57–62, 2019.

[14] E. Battista, S. D. Martino, S. Di Meglio, F. Scippacercola, and L. L. Lucio Starace, "E2e-loader: A framework to support performance testing of web applications," in *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, 2023, pp. 351–361.

[15] "E2EGit: A Dataset of End-to-End Tests in Web Open Source Projects — zenodo.org," https://zenodo.org/records/14221860, [Accessed 26-11-2024].

[16] S. Shrivastava and S. Prapulla, "Comprehensive review of load testing tools," *International Research Journal of Engineering and Technology*, vol. 7, no. 3392-3395, p. 43, 2020.

[17] O. Dabic, E. Aghajani, and G. Bavota, "Sampling projects in github for msr studies," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, 2021, pp. 560–564.

[18] M. M. N. Fuad and K. Sakib, "Webev: A dataset on the behavior of testers for web application end to end testing," in *2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC)*, 2023, pp. 79–83.

[19] M. Biagiola, A. Stocco, F. Ricca, and P. Tonella, "Diversity-based web test generation," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 142–153.

[20] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in github," in *Proceedings of the 36th international conference on Software engineering*, 2014, pp. 356–366.

[21] M. Repository, "Java Web Frameworks," https://mvnrepository.com/open-source/web-frameworks, [Accessed 19-09-2024].

[22] RapidAPI, "State of APIs 2022: Rapid developer survey results," https://stateofapis.com/, 2023, online; accessed 2023-06-23.

[23] "Popular Test Automation Frameworks: How to Choose — BrowserStack — browserstack.com," https://www.browserstack.com/guide/best-test-automation-frameworks, [Accessed 03-12-2024].

[24] "State of JavaScript 2023: Testing — 2023.stateofjs.com," https://2023.stateofjs.com/en-US/libraries/testing/, [Accessed 19-09-2024].

[25] A. Deursen, L. M. Moonen, A. Bergh, and G. Kok, "Refactoring test code," CWI (Centre for Mathematics and Computer Science), NLD, Tech. Rep., 2001.

[26] M. Leotta, H. Z. Yousaf, F. Ricca, and B. Garcia, "Ai-generated test scripts for web e2e testing with chatgpt and copilot: A preliminary study," in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 339–344. [Online]. Available: https://doi.org/10.1145/3661167.3661192

[27] M. Curiel and A. Pont, "Workload generators for web-based systems: Characteristics, current status, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1526–1546, 2018.

[28] L. Christophe, R. Stevens, C. De Roover, and W. De Meuter, "Prevalence and maintenance of automated functional tests for web applications," in *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 141–150.

[29] F. Gortázar, M. Maes-Bermejo, M. Gallego, and J. Contreras Padilla, "Looking for the needle in the haystack: End-to-end tests in open source projects," in *Quality of Information and Communications Technology*, A. C. R. Paiva, A. R. Cavalli, P. Ventura Martins, and R. Pérez-Castillo, Eds. Cham: Springer International Publishing, 2021, pp. 40–48.

[30] Ó. Soto-Sánchez, M. Maes-Bermejo, M. Gallego, and F. Gortázar, "A dataset of regressions in web applications detected by end-to-end tests," *Software Quality Journal*, pp. 1–30, 2022.

[31] M. M. N. Fuad and K. Sakib, "Webev: A dataset on the behavior of testers for web application end to end testing," in *2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC)*. IEEE, 2023, pp. 79–83.