VFAST Transactions on Software Engineering
https://vfast.org/journals/index.php/VTSE@ 2025, ISSN(e): 2309-3978, ISSN(p): 2411-6246
Volume 13, Number 1, January-March 2025 pp: 153-165

Integrating Behavior Driven Testing Approach with Cypress and Cucumber

Abdullah[1]*, **Muhammad Usman**[1], **Najeeb Ullah**[1]

[1]Department Of Computer Software Engineering University Of Engineering and Technology Peshawar, Pakistan

Keywords: *Behavior Driven Development, BDD, Cypress, Cucumber, Automated Testing, Smoke Testing, Gherkin, GUI Testing, E-commerce Applications*

Journal Info:
Submitted:
November 11, 2024
Accepted:
March 18, 2025
Published:
March 31, 2025

Abstract Software testing is integral to ensuring the functionality and quality of applications. This study highlights the implementation of Cypress and Node.js by the Nokia RON team to address the challenges of GUI testing. The adoption of Cypress enabled comprehensive and targeted testing, alongside efficient resolution of dependency and third-party package issues through selective installation. By integrating Cypress with Cucumber, an easy-to-use interface was developed to transform smoke test checklists into Gherkin syntax, enhancing readability and adaptability. Additionally, the use of Mochawesome Reporter provided detailed HTML reports, facilitating issue tracking and quick resolutions. This methodology, supported by a structured questionnaire, fostered stakeholder satisfaction and collaboration, resulting in an interactive and effective testing environment. The findings emphasize the role of Behavior Driven Development (BDD) in streamlining automated testing, improving communication among stakeholders, and ensuring higher software quality.

***Corresponding author email address:** abdullah.mkd275@gmail.com
DOI: 10.21015/vtse.v13i1.1969

1 Introduction

The rapid growth of e-commerce has radically transformed how businesses operate, creating a highly competitive market where the efficiency and reliability of online platforms directly influence customer satisfaction and behavior. Online sales are projected to reach $6 trillion by 2024 [1], positioning e-commerce as a critical component of the global economy. As consumer preferences shift towards seamless, personalized, and user-friendly shopping experiences, companies must continuously enhance their digital platforms to meet these expectations.

The stakes are high for e-commerce applications: even minor service disruptions can lead to substantial financial losses and harm a brand's reputation. A study by [2] revealed that a one-second delay in page load time could reduce conversion rates by up to 7%. This highlights the need for efficient testing procedures to ensure that e-commerce applications operate smoothly and reliably in various conditions. Traditionally, manual testing was employed to verify the functionality of software; however, as software development cycles have become faster and more complex, manual testing is no longer sufficient. It is

time-consuming and prone to human error, resulting in inconsistent test results.

Automated testing has emerged as a critical solution to address the challenges posed by rapid development cycles and frequent software updates. Automated testing tools enable teams to quickly validate software functionality, ensuring that new features do not negatively impact existing functionality. Tools like Selenium, Cypress, and Cucumber have become essential in modern software development due to their ability to automate testing processes, increase testing efficiency, and ensure consistent software quality.

Cypress, a JavaScript-based testing framework, has gained significant attention for its unique approach to end-to-end testing. Unlike traditional testing tools, which often function outside the browser, Cypress operates directly within the browser, allowing developers to create fast, reliable, and easily debuggable tests. Its design supports real-time reloading and debugging, making it particularly suitable for agile development environments where frequent iterations are common [3]. Cypress is especially useful for testing dynamic web applications, which are prevalent in e-commerce platforms where user interactions dictate the flow of information.

In addition to Cypress, the concept of Behavior Driven Development (BDD) introduced by Cucumber has further enhanced automated testing practices. BDD emphasizes collaboration among developers, testers, and non-technical stakeholders by using a domain-specific language called Gherkin. This language enables the creation of test scenarios that are simple to understand, regardless of technical expertise [4]. By adopting Gherkin, teams can create precise acceptance criteria and test scenarios based on user stories, ensuring that all key features are verified early in the development cycle.

The combination of Cypress and Cucumber, applying BDD concepts, has created a powerful synergy in the automated testing landscape. By using Gherkin to express business requirements clearly, both technical and non-technical stakeholders can understand project goals, reducing misunderstandings and aligning testing efforts with business objectives. Automated smoke testing, which ensures the core functionality of an application is working correctly, plays a crucial role in this context. Smoke tests validate the most essential features of an application, allowing teams to quickly identify and address critical issues.

However, despite the advantages of using Cypress and Cucumber, there are challenges that teams must overcome. For example, teams unfamiliar with these tools may initially find it difficult to set up the testing environment. Furthermore, continuous collaboration between developers and testers is necessary to ensure that test scripts remain aligned with evolving application features. Nevertheless, the benefits of automated, behavior-driven testing outweigh these challenges. Studies show that organizations that implement automated testing frameworks report improved product quality, shorter regression testing cycles, and increased testing efficiency [5].

As the software development landscape evolves, so too do the methodologies used to create and test software. The Software Development Life Cycle (SDLC) has undergone significant changes, with the adoption of agile methodologies and DevOps practices. Traditional SDLC approaches, which involve distinct phases of planning, development, testing, and deployment, have shifted towards more iterative and incremental development cycles. Agile development emphasizes delivering smaller, functional software products in shorter time frames, with testing integrated throughout the entire cycle [6].

The transition from manual testing to automated testing has been driven by the need for faster, more reliable testing in agile environments. Manual testing, while effective, is time-consuming and often results in inconsistent outcomes due to human error. In contrast, automated testing enables the execution of large numbers of test cases quickly and consistently, making it an ideal solution for complex software systems like e-commerce applications. In these systems, many critical features—such as inventory management, user authentication, and payment processing—must work together seamlessly. Automated testing can help ensure that these features function correctly across all scenar-

ios.

A report by World Quality [7, 8]highlights that companies adopting automated testing frameworks have seen up to a 75% reduction in testing time and an increase in the accuracy and coverage of tests. This shift has allowed development teams to focus more on improving user experience and functionality while reducing the time spent on repetitive testing tasks. This is particularly important in e-commerce, where maintaining a high-quality user experience is critical to retaining customers.

Cypress is a key player in the automated testing space. Unlike traditional testing tools that operate outside the browser context, Cypress runs directly within the browser, giving developers immediate feedback on their tests. This approach not only enhances the reliability of tests but also simplifies the debugging process, enabling developers to quickly identify and fix issues [9, 10]. Cypress's ability to manage asynchronous processes makes it particularly suited for testing dynamic, user-interactive web applications, which are common in e-commerce environments. Fig. 1 shows evolution testing in e-commerce applications.

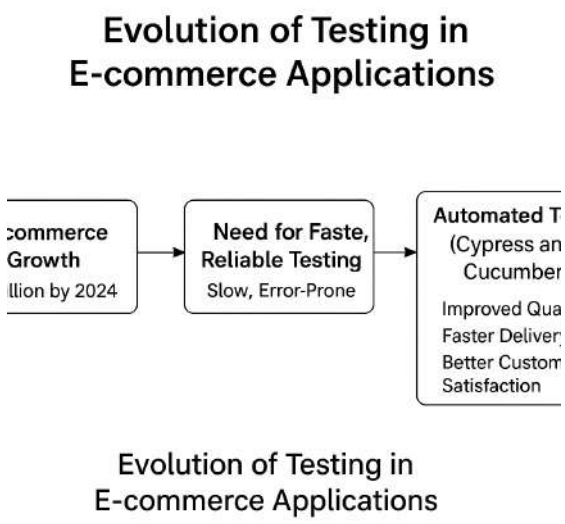

Figure 1. E-commerce evolution Testing

In addition to Cypress, Cucumber has become a popular tool for implementing Behavior Driven Development (BDD). BDD is a testing methodology that promotes collaboration between technical and non-technical stakeholders by using natural language to describe test scenarios. The Gherkin language, used in BDD, enables teams to create test scenarios that are easily understood by all participants, regardless of their technical background [11]. This collaborative approach ensures that all team members, from developers to business stakeholders, are aligned on project objectives and requirements.

When Cypress and Cucumber are used together, they create a powerful framework for automating tests that are both reliable and easy to understand. By using Gherkin to describe business requirements and testing scenarios, teams can bridge the communication gap between developers, testers, and non-technical stakeholders. This collaborative approach improves understanding of project goals and ensures that essential features are tested early in the development cycle. Furthermore, the use of automated smoke tests helps maintain application stability when new features are added, ensuring that core functionality remains intact.

Despite the many benefits of combining Cypress and Cucumber, there are some challenges that teams must address. Setting up the testing environment can be daunting for teams that are new to these technologies. Additionally, maintaining alignment between test scripts and evolving application features requires continuous communication between developers and testers. However, the advantages of using an automated, behavior-driven testing framework far outweigh these challenges. Research has shown that companies using such frameworks experience improved product quality, faster feedback cycles, and better collaboration among stakeholders [12].

This study aims to explore the automation of smoke testing for an e-commerce application using Cypress and Cucumber, with the following primary objectives:

1. To improve communication between technical and non-technical stakeholders by using Gherkin to create test scenarios that are simple and comprehensible for all project participants.
2. To develop an automated smoke test suite that enhances the testing process's usability and ac-

cessibility, resulting in more efficient collaboration and problem-solving.

3. To evaluate the effectiveness and efficiency of the testing framework in terms of stakeholder satisfaction and overall software quality.

By addressing these goals, this research aims to provide valuable insights into automated testing practices and offer practical recommendations for teams considering similar approaches. The findings of this study will help businesses understand the benefits of using automated testing frameworks, such as Cypress and Cucumber, to improve software quality and streamline the testing process.

In conclusion, as e-commerce continues to grow, adopting modern testing techniques like Behavior Driven Development, Cypress, and Cucumber will be essential for businesses to remain competitive. This research seeks to provide a framework that enhances collaboration, improves communication, and ultimately raises the quality of software in the e-commerce industry.

The growing complexity of e-commerce applications makes it increasingly difficult for manual testing to meet the demands of modern software development. As these applications scale and become more sophisticated, the risks of errors and bugs multiply, particularly when handling large volumes of transactions, user interactions, and dynamic content. Manual testing cannot keep up with the speed required in such environments, and relying on it can result in delayed product releases and potentially significant quality issues that impact the end user's experience. Automated testing addresses these concerns by allowing teams to run repeated tests quickly and reliably, ensuring that all functionality, even as it evolves, performs as expected.

Moreover, automated testing frameworks like Cypress and Cucumber integrate seamlessly with Continuous Integration/Continuous Deployment (CI/CD) pipelines. This integration allows teams to execute tests automatically as part of the development lifecycle, ensuring that testing is continuous rather than a one-time or end-of-cycle event. By automatically running tests with each code change, teams can identify and address issues earlier in the development process, reducing the likelihood of defects making it to production. This proactive approach significantly improves the quality of software, helping teams stay ahead of potential issues and ensuring that the final product meets both functional and non-functional requirements.

Ultimately, leveraging automated testing tools, particularly in the context of Cypress and Cucumber, empowers development teams to deliver higher-quality software more efficiently, helping e-commerce businesses maintain competitive advantages in a market where speed, reliability, and customer satisfaction are key to success.

2 Literature Review

The need for strong testing procedures has been highlighted by the quick growth of software development practices. Manual testing is frequently not enough to guarantee quality and performance in software systems that are becoming more sophisticated. Automation testing has become a crucial strategy that helps teams to effectively verify performance and functionality while decreasing human error and boosting coverage [13]. However, while automated testing is essential, it is still evolving, especially in terms of its applicability to complex systems like e-commerce applications, where speed, accuracy, and reliability are paramount.

The industry as a whole has embraced automated testing technologies like Selenium, JUnit, and Testing. Nevertheless, these techniques frequently fail to solve some issues with contemporary online applications, especially those related to performance, dependability, and usability. A research by [14]found that although Selenium and other similar tools have been helpful in automating web testing, they frequently need a lot of setup and can result in brittle tests that are susceptible to UI changes. This issue can be particularly problematic for e-commerce sites, where even small changes to user interfaces can break test scripts, leading to unreliable results.

A more recent option that is especially made for testing modern web apps is Cypress. Tests may be

done directly in the browser thanks to its design, which speeds up execution and enhances debugging capabilities [15]. Because of its emphasis on efficiency and user experience, Cypress has become a popular option for developers, particularly in Agile settings where rapid iterations are essential [16]. Cypress also simplifies handling asynchronous actions, which are common in dynamic, interactive applications like e-commerce platforms, making it an ideal choice for such environments.

An advancement of Test Driven Development (TDD), Behavior Driven Development (BDD) places a strong emphasis on cooperation between developers, testers, and stakeholders who are not technical. It promotes improved communication and comprehension of requirements by encouraging the creation of test scenarios in a language that is understandable to all project participants [17]. In BDD frameworks like Cucumber, stakeholders may describe system behavior in an organized manner using the Gherkin language, which improves alignment and clarity [9]. The capacity of BDD to close the communication gap between team members who are technical and those who are not is a major benefit. According to research by [18], BDD approaches enable everyone to contribute to defining the intended behavior of the software, which improves stakeholder involvement and satisfaction. This cooperative method improves the overall caliber of the finished work in addition to producing a common knowledge of the project's objectives.

A potent combination of testing tools and techniques is represented by the integration of Cypress and Cucumber. Cypress offers a robust environment for executing tests, while Cucumber provides a framework for defining test scenarios in an easily readable format. According to a study by [19], this integration enhances the testing process by allowing teams to write clear, concise tests that reflect user stories and business requirements. By linking user expectations directly with test scenarios, this combination creates a direct feedback loop that benefits both technical and non-technical stakeholders.

The advantages of this connection are further highlighted by research by [10], which points out that it enables quick feedback on application behavior. Because everyone can comprehend and contribute to the test cases, team members may work together more successfully when Gherkin syntax is used. In addition to enhancing communication, this collaboration speeds up testing, enabling more rapid iterations and cutting down on the time it takes to launch new features. This is especially valuable in fast-paced environments such as e-commerce, where quick responses to customer demands are crucial.

Although incorporating Cypress and Cucumber into a BDD framework has several benefits, there is still a dearth of research that focuses on this combination. The majority of current research concentrates on specific tools or approaches, frequently neglecting to investigate how they might be efficiently integrated to improve the software testing procedure. For instance, a thorough analysis of automated testing tools by [20] did not particularly explore the combination of Cypress and Cucumber or its implications for BDD.

Recent initiatives, however, have started to fill this knowledge gap. The effects of using Cypress in combination with BDD principles were examined in a research by [21], which emphasized the significance of open communication and stakeholder participation in the testing procedure. Their results provide credence to the idea that combining Cypress and Cucumber enhances test execution efficiency while also promoting a collaborative culture, which is crucial for software development success. Despite the benefits, challenges related to the learning curve, test environment configuration, and continuous integration remain.

Although there are many advantages to combining Cucumber and Cypress, there are drawbacks as well. The initial setup and configuration needed to properly use these technologies is a major obstacle. According to [22], teams may find it challenging to integrate Cypress and Cucumber with their current testing frameworks, particularly if they are switching from more conventional tools like Selenium. Furthermore, as the program develops, maintaining the testing may become more difficult. Team members must continue to collaborate and communicate in order to guarantee that

the test scenarios appropriately represent the application's current state [13]. Notwithstanding these obstacles, the advantages of better communication, quicker feedback cycles, and more test coverage frequently exceed the hurdles of setting up and maintaining these technologies.

Future studies have to concentrate on examining the long-term consequences of combining Cucumber and Cypress in diverse development settings. Case studies that detail how this integration has been implemented in various firms would offer insightful information about possible hazards as well as best practices. Furthermore, comparative research that assesses this integration's efficacy in comparison to alternative testing procedures will enhance the corpus of current knowledge. Investigating training and onboarding tactics for teams implementing Cypress and Cucumber is another exciting research topic. The effectiveness of their deployment may be greatly impacted by knowing how to help teams during the shift, since these tools need a different attitude and approach than standard testing frameworks [20].

With a focus on the integration of Cypress and Cucumber inside a BDD framework, the literature study emphasizes the crucial role that automated testing plays in modern software development. This combination's advantages—such as better communication, quicker feedback, and more test coverage—highlight its applicability in the hectic development settings of today. For companies looking to improve their testing procedures, this strategy is an appealing option due to the possibility of greater cooperation and efficiency, even in the face of certain implementation difficulties.

3 Research Methodology

The research methodology serves as the study's blueprint, guiding the design, data gathering, and analysis procedures. This section outlines the methodology used to examine the incorporation of Cypress and Cucumber in automated testing and Behavior Driven Development (BDD), focusing on study design, participant selection, data collection methods, and analytical approaches to ensure the validity and reliability of the results. Fig 1 illustrates the proposed methodology.

This study follows a mixed-methods approach, combining both quantitative and qualitative techniques. This choice leverages the strengths of each approach to provide a comprehensive understanding of the study topic.

The quantitative aspect involves gathering numerical data to assess the effectiveness of Cypress and Cucumber in automated testing. Surveys and structured questionnaires will be distributed to software development teams who have utilized these tools. The goal is to quantify the perceived benefits, challenges, and overall satisfaction with integrating Cypress and Cucumber into testing procedures.

Quantitative research is valuable for identifying trends, correlations, and variables that may not be apparent through qualitative methods. Through statistical analysis of the data, insights can be gained that help evaluate hypotheses and measure relationships between variables[23].

Qualitative research, on the other hand, allows for a deeper understanding of participants' experiences and the meanings they attach to their actions. This method is particularly relevant for exploring the collaborative aspects of BDD and for fostering open discussions among participants about the challenges and successes of integration[24, 25](Creswell & Poth, 2017).

The study targets software development teams in agile environments that have integrated Cypress and Cucumber. To ensure a broad range of perspectives, participants will be selected from diverse industries, including e-commerce, healthcare, and finance. A purposive sampling strategy will be used to select individuals with specific expertise in Cypress and Cucumber, ensuring participants are knowledgeable and relevant to the study's focus [26].

Recruitment will occur through professional networks, online forums, and industry conferences, ensuring a wide reach. The study will aim for balanced representation across various roles in the software development lifecycle, such as developers, testers, and product managers, to capture different insights on the integration of Cypress and Cucumber.

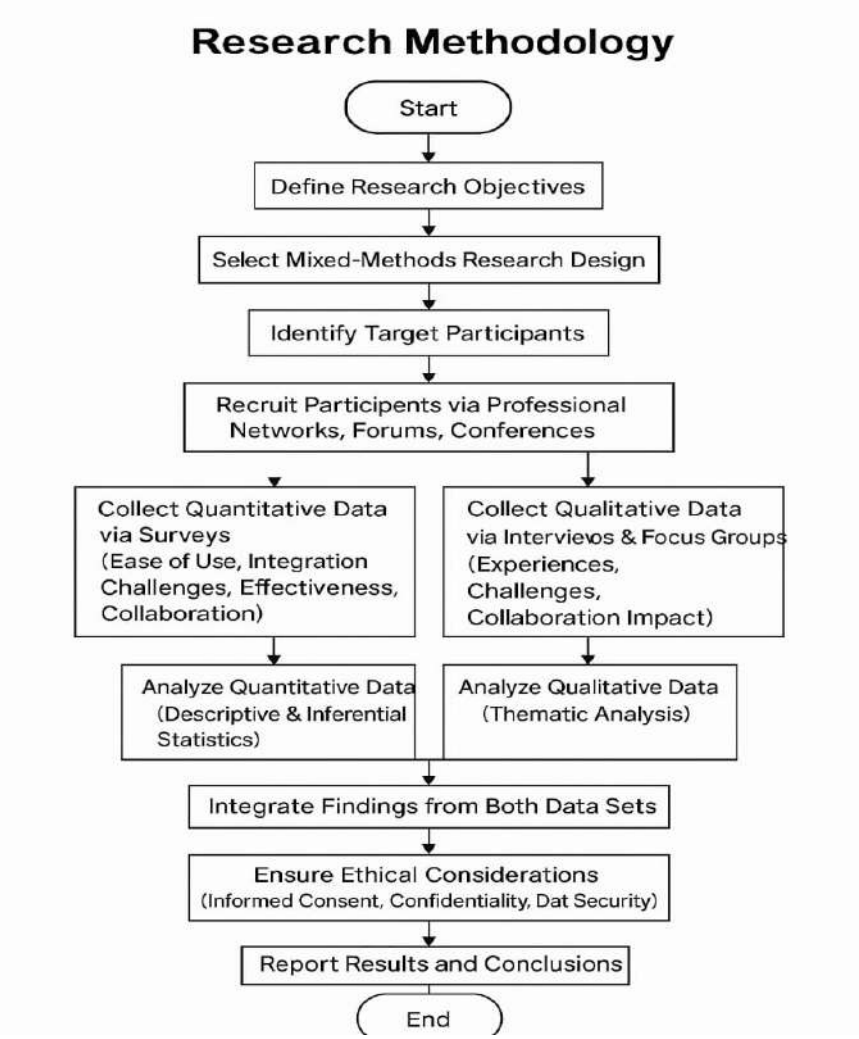

Figure 2. Proposed Methodology Flow Chart

Quantitative data will be gathered through structured surveys that include closed-ended questions covering various aspects of using Cypress and Cucumber:

- **Ease of Use:** Questions will assess the perceived user-friendliness of the tools.
- **Integration Challenges:** Participants will describe difficulties encountered during the integration process.
- **Effectiveness of Testing:** Questions will measure the perceived quality and reliability of tests conducted with these tools.
- **Collaboration and Communication:** Participants will discuss how the integration has affected teamwork and stakeholder engagement.

These surveys will be administered online via platforms like Google Forms or Survey Monkey for ease of access and efficient data collection.

Qualitative data will be collected through semi-structured interviews and focus group discussions. Interviews will feature open-ended questions, allowing participants to share detailed experiences. Sample questions include:

- "Can you describe your experience integrating Cypress and Cucumber into your testing process?"
- "What challenges did you face, and how did you

overcome them?"
- "How has the integration influenced collaboration among team members?"

Focus group discussions will provide an interactive environment for participants to exchange thoughts collectively. This format encourages dynamic conversations, which may reveal insights that individual interviews might not uncover [6, 27]. Both interviews and focus groups will be conducted in person or via video conferencing tools like Zoom, depending on participant preferences and geographical location. All sessions will be recorded with consent for accuracy and analysis.

Quantitative data from surveys will be analyzed using statistical software such as SPSS or R. Descriptive statistics will summarize responses, identifying trends and patterns. Inferential statistics, such as correlation and regression analysis, may be used to explore relationships between variables, such as the correlation between ease of use and perceived effectiveness of testing outcomes, or the influence of team size and industry on the success of integration.

Qualitative data from interviews and focus groups will be transcribed and analyzed through thematic analysis, which identifies and interprets patterns within the data [28, 29]. The analysis process involves the following steps:

1. **Familiarization:** Reading and re-reading transcripts to immerse in the data.
2. **Coding:** Generating initial codes to capture key features relevant to the research questions.
3. **Theme Development:** Organizing codes into broader themes reflecting significant patterns.
4. **Reviewing Themes:** Refining themes to ensure they align with the research objectives.
5. **Reporting:** Presenting findings in a coherent narrative that integrates participants' voices and perspectives.

Ethical considerations will be a top priority throughout the study. Ethical guidelines set by relevant institutional review boards will be followed, ensuring that participants' rights and welfare are protected.

Participants will receive detailed information about the study's purpose, procedures, risks, and benefits. Informed consent will be obtained in writing, ensuring participants understand their right to withdraw from the study at any time without penalty.

Confidentiality will be maintained rigorously. Personal identifiers will be removed from transcripts and data records, and findings will be reported in aggregated form to protect individual identities. Data will be securely stored and accessible only to the research team.

Before commencing the research, ethical approval from the relevant institutional review board will be sought to ensure the study meets ethical standards and protects participants' interests.

In conclusion, this research methodology outlines a mixed-methods approach to investigating the integration of Cypress and Cucumber in automated testing. By combining both quantitative and qualitative methods, the study will provide a comprehensive understanding of participants' experiences, perceptions, and the overall impact of this integration on software development practices. Careful participant selection, data collection, and analysis will ensure the reliability and validity of the findings, contributing valuable insights to the field of software testing and development.

4 Results and Discussion

The results and discussion section presents the outcomes of implementing Cypress and Cucumber within the context of Behavior Driven Development (BDD) and automated testing. This section outlines the steps taken during implementation, highlights the challenges encountered, and evaluates the impact of the integration on the software development process. Table 1 Shows summary of Results discussion.

Before starting the implementation, the team conducted a comprehensive review of the existing technology stack used in their software development projects. The compatibility of Cypress and Cucumber with the current frameworks and tools was assessed to ensure a smooth integration. The selected stack included:

- **Programming Languages:** JavaScript for Cypress and Gherkin syntax for Cucumber.

Table 1. Summary of Results and Discussion

Aspect	Details
Technology Stack	• JavaScript for Cypress, Gherkin for Cucumber • Node.js with Express.js backend • Git for version control
Development Environment Setup	1. Installed Node.js 2. Initialized project using `npm` 3. Installed Cypress and Cucumber preprocessor 4. Configured preprocessor 5. Organized directory structure
Feature Files and Step Definitions	• Feature files written in Given-When-Then format • Step definitions implemented in JavaScript using Cypress commands
Test Execution Modes	• Interactive mode using Cypress Test Runner • Headless mode for CI/CD pipelines
Challenges Encountered	• Compatibility issues between Cypress and Cucumber preprocessor • Learning curve for new team members • Maintenance of test scripts with application updates
Performance Metrics	• Test Coverage • Execution Time • Defect Detection Rate
Outcomes	• Improved collaboration through BDD approach • Increased testing efficiency through automation • Enhanced quality assurance with higher defect detection rate

- **Frameworks:** Node.js as the runtime environment, with Express.js for backend development.
- **Version Control:** Git for source code management and collaboration among team members.

This careful selection ensured that the team could leverage existing skills while minimizing the learning curve associated with adopting new tools.

Once the technology stack was finalized, the development environment was configured. This process involved the following steps:

1. Installation of Node.js: Node.js was installed on all development machines to provide the necessary runtime for both Cypress and Cucumber.
2. Project Initialization: A new project was initialized using npm [30](Node Package Manager), which created a package.json file to manage project dependencies.
3. Installing Cypress and Cucumber: The team installed Cypress and the Cucumber preprocessor by running the following commands:

```
npm install cypress --save-dev
npm install @cypress/cucumber-preprocessor
--save-dev
```

4. Configuring Cucumber Preprocessor: The Cucumber preprocessor was configured in the cypress/plugins/index.js file to allow Cypress to understand and execute Gherkin feature files.

5. Creating Directory Structure: A clear directory structure was established to organize test scripts, feature files, and supporting utilities, with separate folders for:
    - Feature files (e.g., cypress/integration/features)
    - Step definitions (e.g., cypress/integration/steps)
    - Support files (e.g., cypress/support)

This structure facilitated the development and management of automated tests.

The next step was to write Gherkin feature files that outlined the desired behavior of the application. These feature files used the Given-When-Then format to clearly define test scenarios. For example, a feature file for login functionality might look like this:

```
Feature: User Login
  Scenario: Successful login with valid credentials
    Given the user is on the login page
    When the user enters valid credentials
    Then the user should be redirected to the dashboard
```

These feature files served as the foundation for automated testing, ensuring alignment between developments, testing, and business requirements.

The team then developed step definitions to map the Gherkin scenarios to executable code in Cypress. Each step definition was written in JavaScript, using Cypress commands to interact with the application and assert outcomes. For instance, the step definition for the login scenario might look like this:

```
Given('the user is on the login page', () => {
   cy.visit('/login');
});
When('the user enters valid credentials', () => {
   cy.get('input[name="username"]').type('validUser');
   cy.get('input[name="password"]').type('validPassword');
   cy.get('button[type="submit"]').click();
});
Then('the user should be redirected to the dashboard', () => {
   cy.url().should('include', '/dashboard');
});
```

With the feature files and step definitions in place, the team executed the automated tests using Cypress. The tests could be run in two modes:

- **Interactive Mode:** Developers could run tests interactively in the Cypress Test Runner, enabling them to observe the tests in real time and debug issues.
- **Headless Mode:** For continuous integration (CI), tests could be executed in headless mode, running without a graphical interface. This was particularly useful for CI/CD pipelines.

Tests were executed by running the following command:

```
npx cypress open
```

This command opened the Cypress Test Runner, displaying all available test files for execution.

During test execution, Cypress provided detailed logs and visual feedback, helping the team monitor the status of each test. Test results were categorized as passed or failed, with additional information available for troubleshooting failed tests. Cypress also captured screenshots and videos automatically, which proved invaluable for debugging.

Despite the successful setup and integration of Cypress and Cucumber, the team faced several challenges during implementation:

- **Compatibility Issues:** The team initially encountered compatibility issues between the versions of Cypress and the Cucumber preprocessor, leading to occasional errors. By regularly consulting the documentation and community forums, the team adjusted configurations to ensure compatibility.
- **Learning Curve:** As Cypress and Cucumber were new tools for some team members, there was a learning curve. To address this, the team organized training sessions and workshops, providing hands-on experience and promoting collaboration in writing tests.
- **Maintaining Tests:** As the application evolved, keeping the automated tests up to date became challenging. Changes in the application's behavior required updates to feature files and step definitions. The team implemented regular reviews of the test suite to identify outdated tests and ensure alignment with the latest application requirements.

To evaluate the success of the implementation, the team established several performance metrics, including:

Table 2. Performance Metrics after Implementation of Cypress and Cucumber

Metric	Description	Result/Observation
Test Coverage	Measures the percentage of application features covered by automated tests.	High Coverage Achieved
Execution Time	Measures the total time required to execute the full suite of automated tests.	Significantly Reduced
Defect Detection Rate	Compares the number of defects identified through automated testing versus manual testing.	Higher Detection Rate

- **Test Coverage:** The percentage of application features covered by automated tests.
- **Execution Time:** The time taken to run the entire suite of automated tests.
- **Defect Detection Rate:** The number of defects identified through automated testing compared to manual testing.

Table 2 highlights the performance metrics Feedback sessions after the implementation revealed several positive outcomes:

- **Improved Collaboration:** The BDD approach improved communication among developers, testers, and stakeholders, leading to a shared understanding of requirements.
- **Increased Testing Efficiency:** Automation significantly reduced the time spent on manual testing, allowing the team to focus on higher-value tasks.
- **Enhanced Quality Assurance:** Automated testing led to a higher defect detection rate, improving the overall quality of the software.

In conclusion, the implementation of Cypress and Cucumber demonstrated the effectiveness of integrating these tools within a software development process. Despite challenges, the outcomes highlighted significant improvements in collaboration, testing efficiency, and quality assurance. These results provide valuable insights into the role of automated testing and BDD in modern software development practices.

5 Conclusion

In conclusion, the implementation of a user-friendly environment for smoke testing, alongside a comprehensive questionnaire, has received significant appreciation and positive feedback from both users and stakeholders. This approach has enhanced engagement and accessibility, fostering active participation and improving communication between technical and non-technical project members. By translating smoke test scenarios into the easily readable Gherkin language and integrating Cypress and Cucumber tools, the testing process has become more transparent and understandable for all stakeholders. The use of Mochawesome to generate clean HTML reports has further streamlined the process, making test results clear and easily viewable. This solution has bridged communication gaps and optimized the overall testing workflow, as reflected in the overwhelmingly positive responses. The user-friendly design and well-constructed questionnaire have created an interactive atmosphere, driving higher levels of interest and engagement. The affirmation from stakeholders highlights the success of this approach in enhancing the efficiency and effectiveness of smoke testing for web applications.

Author Contributions

Abdullah: Conceptualization, Methodology, Software **Muhammad Usman**:Data curation, Writing- Original draft preparation , Editing. **Najeeb Ullah**: Software, Validation. Writing- Reviewing and Supervison.

Compliance with Ethical Standards

It is declared that all authors don't have any conflict of interest. It is also declare that this article does not contain any studies with human participants or animals performed by any of the authors. Furthermore, informed consent was obtained from all individual participants included in the study.

References

[1] Statista, "E-commerce worldwide - statistics & facts," 2021. Retrieved from.

[2] A. Hassanien and A. Naim, "The impact of web performance on e-commerce: A review," *International Journal of Web Engineering and Technology*, vol. 15, no. 4, pp. 401–417, 2020.

[3] Cypress, "Writing your first test," 2021. Accessed 23.2.2021.

[4] Cucumber, "Gherkin language reference," 2021. Retrieved from.

[5] R. Mobaraya and S. Ali, "Comparative analysis of selenium and cypress for automation testing," *Journal of Software Engineering and Applications*, vol. 12, no. 8, pp. 274–284, 2019.

[6] S. Barab and K. Squire, "Design-based research: Putting a stake in the ground," *The Journal of the Learning Sciences*, vol. 13, no. 1, pp. 1–14, 2004.

[7] W. Q. Report, "World quality report 2020-21," 2020. Retrieved from.

[8] W. Q. Report, "World quality report 2020-21," 2020. Retrieved from.

[9] Cypress, "Cypress documentation," 2021. Retrieved from.

[10] M. Rojas, D. Pacheco, and J. Vargas, "Improving software quality with cypress and cucumber integration: An empirical study," *Software Quality Journal*, vol. 29, no. 2, pp. 443–467, 2021.

[11] Cucumber, "Gherkin language reference," 2021. Retrieved from.

[12] R. Mobaraya and S. Ali, "Comparative analysis of selenium and cypress for automation testing," *Journal of Software Engineering and Applications*, vol. 12, no. 8, pp. 274–284, 2019.

[13] B. Meyer, *Introduction to Testing Software*. Cambridge University Press, 2018.

[14] N. Absharina, R. Mardiana, and R. Putra, "The role of automation testing in software development life cycle," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 6, pp. 65–72, 2020.

[15] Cypress, "Cypress documentation," 2021. Retrieved from.

[16] M. Liikka, "Fast and reliable testing with cypress: A case study," *Journal of Software Engineering Research and Development*, vol. 9, no. 1, pp. 27–34, 2021.

[17] M. Fowler, *Domain Specific Languages*. Addison-Wesley, 2011.

[18] R. Biddle, J. Noble, and S. Marshall, "The role of stakeholders in behavior-driven development," in *Proceedings of the International Conference on Agile Software Development*, vol. 4, pp. 43–55, 2017.

[19] M. López, R. Martinez, and S. Garcia, "Enhancing test automation using cypress and cucumber: A collaborative approach," *Journal of Computer Science and Technology*, vol. 35, no. 4, pp. 654–667, 2020.

[20] H. Khalil, H. Khattak, and M. Khan, "A review of automated testing tools: Comparative study and future directions," *Software Testing, Verification & Reliability*, vol. 30, no. 10, p. e2210, 2020.

[21] A. Gupta, R. Tiwari, and S. Sharma, "Exploring the integration of cypress with bdd principles for enhanced software testing," *Journal of Software Engineering and Applications*, vol. 14, no. 8, pp. 305–317, 2021.

[22] R. Sykes, D. Franklin, and A. Thompson, "Transitioning to cypress: Challenges and solutions," *International Journal of Information Technology and Management*, vol. 18, no. 1, pp. 53–66, 2019.

[23] S. Avasarala, *Selenium WebDriver Practical Guide*. Birmingham B3 2PB, UK: Packt Publishing, 2014.

[24] N. Chanana and S. Goele, "Future of e-commerce in india," *International Journal of Computing & Business Research*, 2021.

[25] X. Zhang and Y. Liu, "Exploring the effectiveness of bdd in agile software development: A case study," *Agile Processes in Software Engineering and Extreme Programming*, vol. 35, no. 1, pp. 21–30, 2020.

[26] S. Ashmore and K. Runyan, *Introduction to Agile Methods*. Crawfordsville, Indiana, United States: Pearson Education, Inc., 2015.

[27] B. Morelli, "How to test javascript with mocha - the basics," 2017. Accessed 23.2.2021.

[28] RedHat, "What is application lifecycle management (alm)?," 2021. Accessed 13.2.2021.

[29] K. K. S. A. Bhat and J. M. Khan, "A review paper on e-commerce," *Asian Journal of Technology & Management Research*, vol. 6, no. 1, 2016.

[30] Node.js, "Introduction to node.js," 2021. Accessed 29.3.2021.

[31] T. Point, "Sdlc – overview," 2014. Accessed 3.2.2021.

[32] C. Solis and X. Wang, "A study of the characteristics of behaviour driven," in *EUROMICRO Conference on Software Engineering and Advanced Applications*, vol. 37, 2011.

[33] V. K. Chauhan, "Smoke testing," *Int. J. Sci. Res. Publ*, vol. 4, no. 1, pp. 2250–3153, 2014.

[34] Guru99, "What is alm? application lifecycle management," 2021. Accessed 13.2.2021.

[35] T. QA, "What is test design? or how to specify test cases?," 2021. Accessed 27.2.2021.

[36] Wikipedia, "Regression testing," 2021. Accessed 27.3.2021.

[37] A. Khetarpal, "What is cypress: Introduction and architecture?," 2020. Accessed 22.3.2021.

[38] M. . Company, "How to improve software delivery performance," 2020. Retrieved from.

[39] Cucumber, "Gherkin language reference," 2021. Retrieved from.

[40] R. Mobaraya and S. Ali, "Comparative analysis of selenium and cypress for automation testing," *Journal of Software Engineering and Applications*, vol. 12, no. 8, pp. 274–284, 2019.

[41] M. D. V. M. Rangel, "Post-pandemic consumer behavior towards e-commerce and retail stores in united states," *Revista Venezolana de Gerencia: RVG*, vol. 26, no. 6, pp. 47–64, 2021.

[42] M. T. Taky, "Automated testing with cypress," 2021.