Journal of Innovation and Computer Science (JICS)
Vol. 1, No. 2, April 2025
https://journal.itqanpreneurs.com/index.php/jics/index

PHP-Based SIMPORA Application Testing Using the Blackbox Method

Mohamad Firdaus[1]
[1]Universitas Indraprasta PGRI, Jakarta, Indonesia. E-mail: mfirdausmumu@gmail.com

Article History:
Received: 14-01-2025
Revised: 06-04-2025
Accepted: 08-04-2025

Keywords: *Application Testing, Blackbox, PHP, Functionality, Project Monitoring.*

Abstract: *This study uses the black box testing method to test the PHP-based SIMPORA (Project and Budget Monitoring Information System) application's reliability and functionality. This method was chosen because it focuses on validating the application's functions without directly examining the source code, allowing for performance evaluation from the end-user's perspective. The testing process involved a series of test scenarios to assess key features, including project data input, budget monitoring, and result reporting. The testing results indicate that the SIMPORA application successfully meets most of the specified functional requirements, although some anomalies were identified and need to be addressed. This study significantly contributes to ensuring the SIMPORA application's quality and offers insights for further development. Therefore, the results of this research are expected to help improve the efficiency and effectiveness of project and budget management through the application of information technology-based solutions.*

Introduction

Budgeting is a crucial element in running an organization. The ability to predict the budget and its realization is essential to ensure the organization's smooth operation. Organizational activities risk stalling without accurate predictions regarding expenditure and results and will not achieve optimal results. In order to answer these challenges, an online-based management information system is needed that effectively manages the budgeting process and its realization. This system is expected to increase the efficiency of organizational management and facilitate access to information related to the use of funds and monitoring of activities. To improve performance and meet operational targets, Tanjung Pandan Customs and Excise, one of Indonesia's customs offices, developed a unique application. This application is designed to support improving their performance and as part of efforts to win performance competitions between customs offices throughout Indonesia. It is hoped that this application will become a mainstay innovation that will strengthen their activities in the future (Firdaus, 2022).

In software engineering, one of the commonly used methods for software development is the Waterfall model, also known as the Software Development Life Cycle (SDLC). Within the SDLC framework, the software development process begins with the planning phase, followed by analysis and design, and concludes with implementation (Sommerville, 2016). Software testing is a vital process in software development aimed at ensuring the developed product's quality, performance, and security. Over the past few decades, information and communication technology advancements have heightened the demand for reliable and efficient software. Software testing goes beyond merely detecting errors; it also seeks to enhance the overall quality of the software. This includes

improving user experience, ensuring the system can handle expected workloads, and verifying that all software functions operate correctly under various conditions. Consequently, software testing has become an integral part of the software development lifecycle. Software testing is a crucial stage in software development, ensuring the quality of the program's design and control structure. It is an integral aspect of the software development process, functioning as both validation and verification of the software. Additionally, software testing is a key quality assurance component, making it an essential part of the software development lifecycle (Bakti & Firdaus, 2024).

Software testing costs are often significant, sometimes comprising a large portion of the development budget. Literature reviews on software testing indicate that approximately 50% of the total development costs are allocated to testing (Jorgensen, 2014). Automated testing has been proposed as a solution to mitigate these high costs to mitigate these high costs. Various software testing techniques, including blackbox and whitebox methods, can be employed. Blackbox testing focuses on the software's functionality, while whitebox testing examines the program's structure. Each method has its strengths and weaknesses. For instance, black box testing, which emphasizes functionality and input, may face challenges documenting test results if the software specifications are unclear. On the other hand, white box testing, which delves into the program structure, can be resource-intensive when applied to large-scale software and requires testers to be familiar with the code. Structural testing is critical and can be more cost-effective than functional testing (Sulistyanto, H & Azhari, 2014).

Some standard methods in software testing (Wibisono, W., & Baskoro, 2023). First, Unit Testing is testing the smallest part of the software, such as a function or procedure, separately, usually done by the developer and often using automation. Second is Integration Testing, a combined test of several units to ensure the interface between units works appropriately. Third is System Testing, which tests the entire system to ensure that the software meets functional and non-functional requirements. Fourth is acceptance testing, which is the final stage of testing involving users ensuring the system meets their needs, including alpha and beta testing. Fifth is Regression Testing, which is carried out after changes or updates to the software to ensure that no functions are damaged due to the modification. Sixth, Automated Testing, which is the use of automated tools and scripts to run repeated tests efficiently and consistently, especially in large projects that require periodic testing.

Method

Black Box Testing aims to identify software or information system design errors, such as malfunctioning features or non-working menus. This method is particularly suitable for testing the software's functionality. This testing uses random input data to observe the system's output. If there is an error in the user input, the data will be rejected or not stored in the database. Conversely, correct data will be accepted and stored in the system's database. Several techniques are employed in Black Box Testing: a) Equivalence Analysis, which focuses on identifying external and internal errors and determining minimum and maximum boundary values. b) Fuzzing, which searches for bugs by injecting flawed data into the software. c) Cause-effect graph, which uses graphs to depict cause-effect relationships. d) Orthogonal array testing is used when the input domain is relatively small but sufficiently complex on a large scale. e) All Pair Testing, which designs all pair test cases to test discrete combinations of input parameters. f) State Transition tests machine states and navigation using graphs (Beizer, 1990).

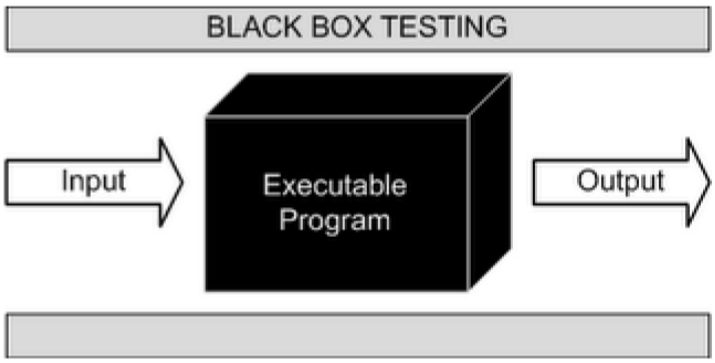

Figure 1. Black Box Testing Software Testing Techniques

Result and Discussion

Results

Black Box testing conducted on the PHP-based SIMPORA application showed positive results related to system functionality (Firdaus, 2022, 2024; Firdaus & Bakti, 2024). This test mainly uses the Equivalence Partitioning technique by grouping input data into several different partitions to evaluate system response. In general, all features and menus in the application function as they should. The system can accept valid data and reject invalid data, indicating that the application has run according to design specifications.

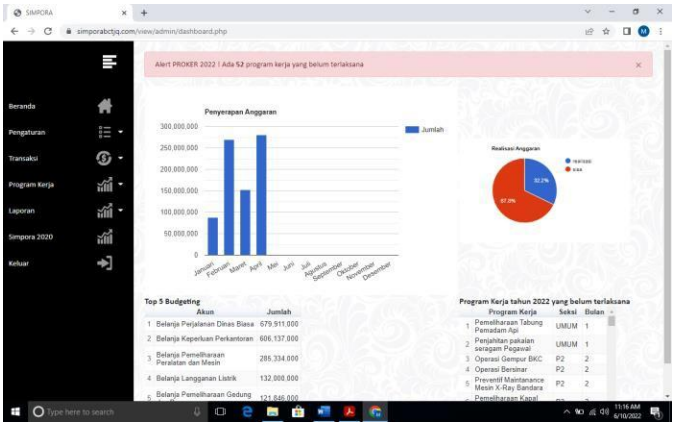

Figure 2. Main page with features

The application demonstrates effective input validation performance in terms of input data handling. Valid data is successfully stored in the database, while invalid data is rejected or generates an error message. This demonstrates the system's ability to handle various types of input data and confirms that the validation mechanism has been implemented correctly. However, several edge cases were successfully identified during testing. Certain combinations of input trigger unexpected system behavior. These issues are generally related to data validation rules that are not fully covered in the initial design.

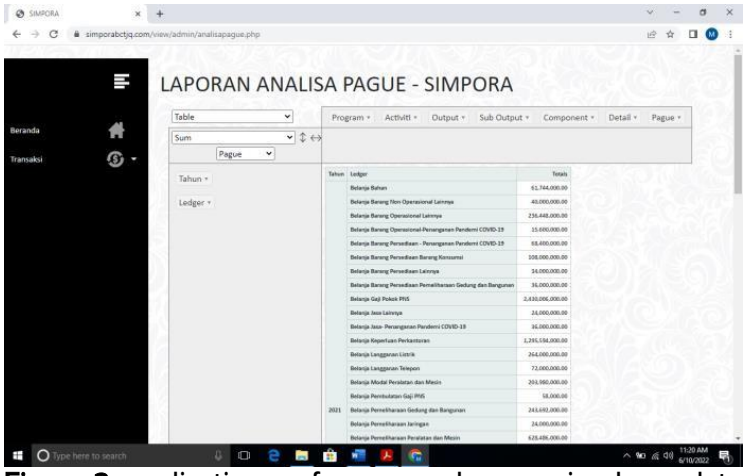

Figure 3. application performance when running large data

However, these issues are considered minor and can be resolved with further adjustments to the validation logic. Overall, the system performance during testing is efficient. No significant obstacles were found regarding speed or stability when running various test scenarios. Thus, this Black Box testing proves that the SIMPORA application has good functionality and is ready for further use, although it still requires improvements in certain aspects.

Discussion

Black Box Testing methodology, especially with the Equivalence Partitioning technique, has proven to be effective in evaluating SIMPORA applications. Focusing on the system's response to various input types, this method successfully identifies functionality issues related to input data validation and error handling. This testing provides a comprehensive picture of how the system responds to input from the user's perspective without requiring an understanding of the internal structure of the code. This is very helpful in ensuring that the application behaves as expected by the end user.

The effectiveness of the Black Box Testing approach is seen from its ability to comprehensively assess the application's ability to process data correctly, which is critical to maintaining data integrity and system stability. However, some challenges still arise during the testing process. Several issues were identified in certain edge cases where uncommon input combinations triggered system errors. This indicates a gap in the initial system design, especially in the data validation mechanism, which is not yet robust enough to handle all possible inputs. This finding emphasizes the importance of strengthening data validation rules to make the system more resilient to errors.

Based on the test results, improving the input validation logic to address the identified exceptional cases is recommended. In addition, additional testing techniques, such as Fuzzing, can help find vulnerabilities that may not have been detected in the initial Black Box testing. For further testing, the scope of test scenarios should be expanded to involve more complex scenarios and use a wider variety of input types. Automatedols are also recommended to improve efficiency in future system maintenance and updates. The test results show that the SIMPORA application functions well and has met the primary needs. However, continuous input validation and error handling improvement are needed to create a more reliable and user-friendly system.

Conclusion

The testing of the SIMPORA application, developed using PHP with the Black Box Testing method, has successfully identified various aspects of the system's functionality, ensuring that all features and menus operate according to specifications. By applying the Equivalence Partitioning technique, the testing focused on dividing the input data into several partitions, which efficiently identified error classes and reduced the number of required test cases. The results demonstrated that the SIMPORA application correctly handles input data, accepting and storing valid data in the database while rejecting invalid inputs. This confirms that the application meets the defined functional requirements. In conclusion, this testing significantly enhanced the SIMPORA application's quality, ensuring its reliability and providing a better user experience. The Black Box Testing method proved effective in assessing the application's functionality and can serve as a valuable approach for future software testing endeavors.

Acknowledgements

We want to thank the family and all parties who contributed to this research. Support and assistance from various parties are critical in completing this research.

References

Bakti, I., & Firdaus, M. (2024). *Waterfall Metodancangan Software Untuk Pemula*. CV. Media Sains Indonesia. https://doi.org/10.5281/zenodo.11216705

Beizer, B. (1990). *Software Testing Techniques* (2nd ed.). International Thompson Computer Press.

Firdaus, M. (2022). Implementasi IT Project Management Dalam Pembuatan Sistem Informasi Monitoring Pagu Dan Anggaran. *Jurnal Teknologi Informasi (JUTECH)*, *3*(2), 122–130. https://doi.org/10.32546/jutech.v3i2.2031

Firdaus, M. (2024). Perancangan Enterprise Arsitektur Menggunakan TOGAF ADM Di Bea Cukai Tanjung Pandan pada Aplikasi SIMPORA. *Jurnal Informatika*, *3*(1), 6–12. https://doi.org/10.57094/ji.v3i1.1440

Firdaus, M., & Bakti, I. (2024). Perancangan dan Pembuatan Aplikasi SIMPORA Berbasis Online dengan PHP. *Technologia Journal*, *1*(1).

Jorgensen, P. C. (2014). *Software Testing 4th Edition* (4th ed.). Taylor & Francis Group.

Sommerville, I. (2016). *Software Engineering 10th Edition*. Pearson Education Limited.

Sulistyanto, H & Azhari, S. . (2014). Urgensi Pengujian Pada Kemajemukan Perangkat Lunak dalam Multi Perspektif. *KomuniTi*, *1*(1), 65–74.

Wibisono, W., & Baskoro, F. (2023). Pengujian Blackbox Metode Equivalent Partitions pada Aplikasi Karyawan Website Oby Komputer. *Jurnal Sistem Informasi (Teknofile)*, *1*(2). https://jurnal.nawansa.com/index.php/teknofile/article/view/16