Black Box Testing with Equivalence Partitioning and Boundary Value Analysis Methods (Study Case: Academic Information System of Mataram University)

Putu Ayu Desi Anggara Santi(✉), Royana Afwani, Moh. Ali Albar, Sri Endang Anjarwani, and Ahmad Zafrullah Mardiansyah

Mataram University, Mataram, Indonesia
putuayudesi.as@gmail.com, {royana,mohalialbar,zaf}@unram.ac.id, endang@ti.ftunram.ac.id

Abstract. Mataram University Academic Information System (SIA UNRAM) is software built and managed by UPT PUSTIK to make it easier for users to manage various academic data and academic information at Mataram University. Although the system has been developed, the users still discover the system's lack of features and functions. So, this study aimed to test the SIA system. Software testing is a significant thing to do so that the result of the device is appropriate to the user's needs. Moreover, it can detect and reduce the errors contained in the software. Black box testing is one of the software testings that focuses on functionality. Equivalence partitioning and boundary value analysis are used as the research methods. The final result of this study found various feature errors in 4 SIA actors at Mataram University, including students, lecturers, study program operators, and faculty operators. This test found that the total test cases were 322 scenarios, with 242 test cases passing and 80 defects, so the average test case pass percentage was 75.16%. UPT PUSTIK uses these results to evaluate and improve the SIA software at Mataram University.

Keywords: Academic Information System · Software Testing · Black Box · Equivalence Partitioning · Boundary Value Analysis

1 Introduction

Mataram University Academic Information System (SIA UNRAM) is one of the software products produced by UPT PUSTIK. SIA UNRAM is software designed and built to make it easier for users to manage various data and information related to academics. There are 4 (four) categories of users in SIA UNRAM, including students, lecturers, department operators, and faculty operators. Even though system development has been carried out, feature and function discrepancies can still be found, such as output discrepancies in inputted data and not displaying input data. Based on interviews with UPT PUSTIK, the researcher found that UPT PUSTIK did not conduct in-depth testing of the resulting system, so it was necessary to test the software in UPT PUSTIK.

I G. P. Suta Wijaya et al. (Eds.): MIMSE-I-C 2022, ACSR 102, pp. 207–219, 2022.
https://doi.org/10.2991/978-94-6463-084-8_19

In this study, black box testing was conducted using equivalence partitioning and boundary value analysis methods. Black box testing is important in software testing, which helps validate the system's functionality under test. Black box testing with equivalence partitioning and boundary value analysis methods is useful for detecting functions or features that have defects so that they can be repaired or eliminated by the developer. Black box testing using equivalence partitioning and boundary value analysis methods has resulted in many analyses stating that the resulting software does not always follow user needs based on functional requirements.

Based on the problems and theories described, this study will discuss "Black Box Testing with Equivalence Partitioning and Boundary Value Analysis Methods (Study Case: Academic Information System of Mataram University)." The consideration for choosing SIA UNRAM is because almost the entire academic community always uses the software at Mataram University. This study will test the features of students, lecturers, faculty operators, and department operators. The final result of this study is the extent to which black box testing can be used to find various errors in the SIA UNRAM software. UPT PUSTIK can use these results to evaluate and improve the SIA UNRAM software.

2 Literature Review

2.1 Related Research

In the research on testing the web-based online wedding invitation sales information system using black box testing, the beta test average value is 81%. Testing is done manually using test scenarios and test cases, as well as a questionnaire filled out by five respondents [1].

Research has been carried out on testing the application software for the maintenance of state property using black box testing with equivalence partitioning methods to obtain an application effectiveness value of 93.2% for the functional needs of replacing goods. The test is carried out manually, with five fields from the structure of the replacement table for the goods being tested [2]. In a similar study, the design of a web-based geographic information system (GIS) for the provision of information on facilities and personnel at Lampung University by testing using the equivalence partitioning method obtained a percentage of 100% for questionnaire testing by operators and 94.4% for general users in good opinion, 5.3% in the opinion moderate, and 0.3% think less. The test was carried out manually using a test questionnaire with 20 general user respondents and one operator [3].

Research has been carried out on testing the industrial internship information system using the black box testing with boundary value analysis method with a success value of 75% and a success rate of 95%. The test is done manually, with five fields from the apprenticeship table structure in the tested, industrial apprenticeship system [4]. In a similar study, black box testing with boundary value analysis on the submission system application with the test results of each field getting a truth level equal to or more than 75%. The test was carried out manually, with two tables being tested four times of testing sample data in each field [5]. In the black box testing research on online news applications using the boundary value analysis method, the results from 14 test scenarios

have 13 successful tests. Testing is done manually with test scenarios to obtain actual results [6].

In the research on testing the ACC Whistle website using the black box testing method, the results show that there are still bugs in the system being tested. Testing is done manually and using the Katalon Studio application [7]. In a similar study, the functionality test (black box testing) of the batik professional certification agency (SILSP) information system with the AppPerfect Web Test and the user test showed that the test using the application was quite relevant for testing with customized iterations. The application tests were performed ten times on each page [8]. Research has been done to analyze the final project web application test in the network (TADJ) using the black box method and Selenium IDE with the results of testing errors or functions that are wrong or missing in some parts that have inappropriate functions and on interface testing found some features that are missing need to be improved [9].

2.2 Theoretical Basis

Software. The software includes computer programs, documentation, and configuration data that are related and necessary to make the program operate properly. Software is a program that manipulates information with various instructions in certain features or functions so that software can print the data obtained in the virtual form [10].

Information System. An information system is a collection of hardware and software designed to transform data into useful information by collecting, processing, storing, analyzing, and disseminating information for specific purposes. Information systems include many components (humans, computers, information technology, and work procedures), something that is processed (data into information), and intended to achieve a goal or goal [11].

Academic Information System. An academic information system is a system that processes academic data and information, including student or student data, teaching staff data, curriculum, schedule, and various other data in an educational institution. An academic information system is an online-based information system that aims to form a knowledge-based system that can be accessed using the internet [12, 13].

Software Testing. According to the ANSI/IEEE standard, testing is the process of analyzing a software entity to detect differences between the desired conditions (defects/errors/bugs) and evaluating the features of the software entity [14]. Software testing includes various methods, types, and levels or stages of testing. The basic software testing methods include static and dynamic testing, black box testing, gray box testing, and white box testing, as well as manual testing and automated testing [15].

Black Box Testing. Black box testing is a test that only tests the outside of the software [16]. Black box testing is a technique that focuses on the functional requirements of the software based on the software requirements specification [14]. There are several testing methods in black box testing, such as equivalence partitioning, boundary value analysis, cause-effect graph, comparison testing, random data selection, feature tests, all-pair testing, fuzzing, orthogonal array testing, sample testing, robustness testing, behavior testing, performance testing, endurance testing, and others [16, 17].

Equivalence Partitioning. Equivalence partitioning is a test method designed by examining the input and output of data grouped by function, whether valid or invalid

[16]. Equivalence partitioning attempts to define a test case based on evaluating the equivalence of a type or class for an input condition in the form of a numeric value, a range of values, a set of corresponding values , or a Boolean condition [18].

Boundary Value Analysis. Boundary value analysis is a test focusing on boundaries, where the extreme values are chosen. Boundary value analysis is a method that tests the maximum and minimum limits to produce valid values that are considered quite relevant. Boundary value analysis prefers the elements in the equivalent class on the boundary side of the class. Boundary value analysis is a complement to equivalence partitioning [16].

Selenium. Selenium is an open-source tool used to test the automation of various online and real-time applications. Selenium is designed to automate the interface of web browsers, so programming scripts can spontaneously complete similar interactions, some of which are done manually [19]. Test operations using Selenium are very flexible, which allows many options to find user interface elements and compare the expected test results with the actual application behavior [20].

3 Methodology

1. The Study of Literature was conducted by collecting data and information from books, journals, and various readings related to testing.
2. Needs Analysis and Collection to determine the requirements in system testing by conducting interviews, observations, and system inspections.
3. Testing Design is to make a test module of the SIA function of Mataram University comprises students, lecturers, faculty operators, and study program operators.
4. Black Box Testing consists of several stages, including making test cases, testing functionality using equivalence partitioning and boundary value analysis methods, and calculating test results.
5. Test Results Report consists of test documentation, analysis, and discussion of test results (Fig. 1).

.

4 Result and Discussion

Black Box Testing with Equivalence Partitioning and Boundary Value Analysis Methods (Study Case: Academic Information System of Mataram University) was conducted on student actors, lecturers, study program operators, and faculty operators to determine if there were problems in the system. This is useful for getting a large percentage of the success of the specifications provided by the system to the user.

4.1 Testing Implementation

The test was carried out at UPT PUSTIK of Mataram University for 3 (three) days, on 27 May 2022 to 29 May 2022, using Selenium IDE testing tools. Testing is carried out using Selenium IDE tools by following the flow according to the test case that has been

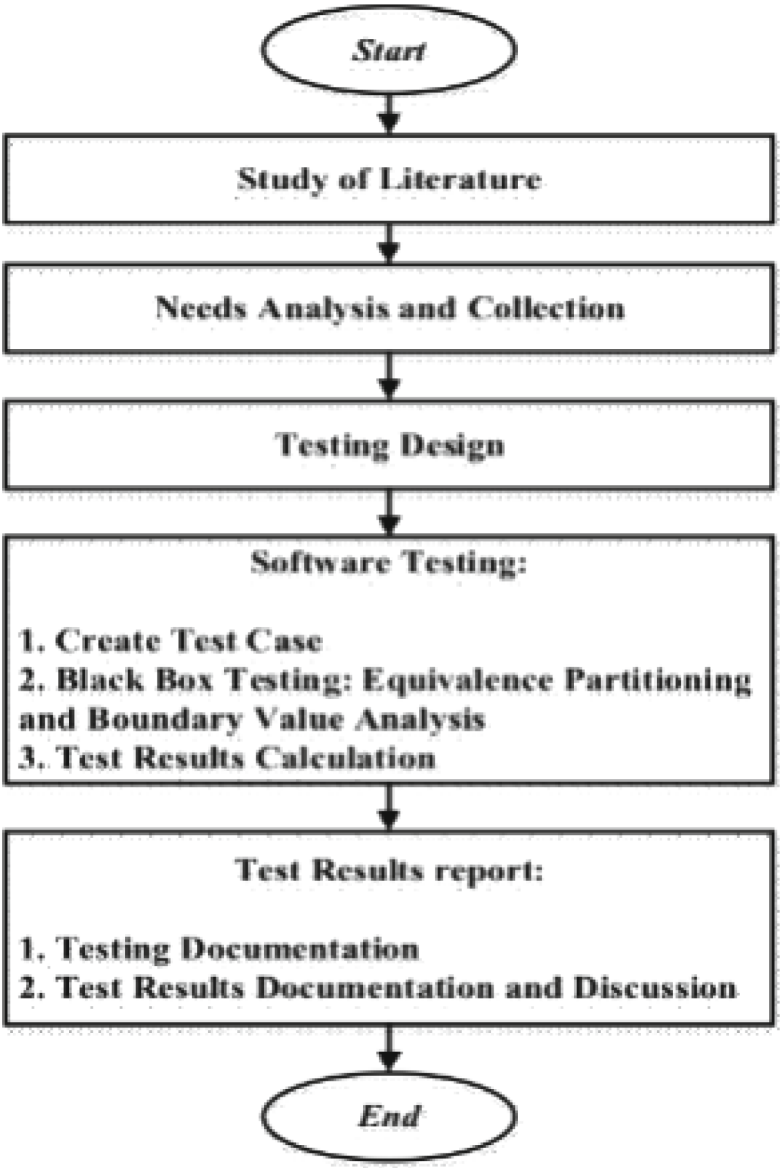

Fig. 1. Research flow chart.

made, by creating a project on the tools according to the module and tests according to the test ID that has been determined in the test case. Then do the record according to the test case that has been made by entering the URL first. Record results are stored and can be used to run tests (Fig. 2).

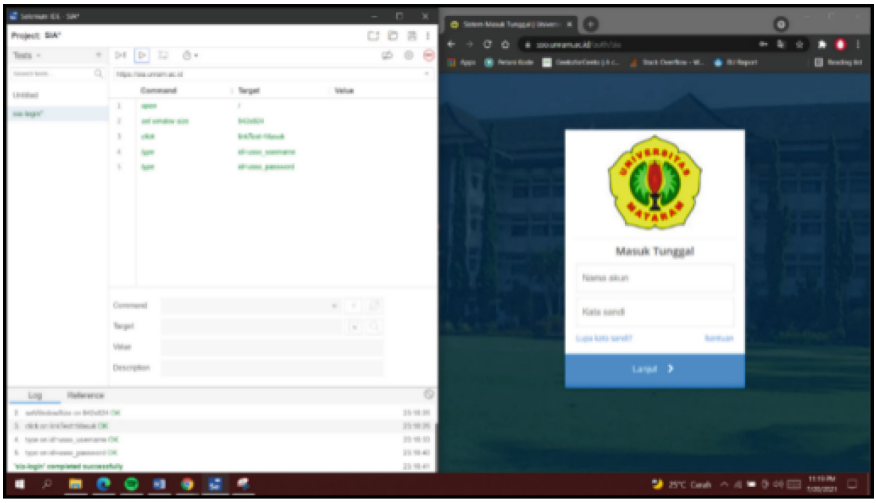

Fig. 2. Selenium IDE implementation.

4.2 Test Cases

In a test case, a test case scenario forms the basis for testing.

In Tables 1, 2, 3 and 4, the total test cases belonging to each actor are described. In Table 1 the total test cases for the student module are 127 scenarios; in Table 2, the total

Table 1. Total student module test case.

ID	Testing Module	Test Case
A001	Log in	13
A002	Account Settings	7
A003	Students Bio	99
A004	Message	8
Total		127

Table 2. Total lecturer module test case.

ID	Testing Module	Test Case
B001	Log in	10
B002	Account Settings	7
B003	Lecturers Bio	27
B004	Grades	8
Total		52

Table 3. Total department operator module test case.

ID	Testing Module	Test Case
C001	Log in	10
C002	Account Settings	4
C003	Department Profile	40
C004	Lecturers	8
C005	Students	6
C006	Course	8
C007	Grades	6
C008	KRS/KHS/KSP Settings	6
Total		88

Table 4. Total faculty operator module test case.

ID	Testing Module	Test Case
D001	Log in	10
D002	Account Settings	2
D003	Faculty Operators	21
D004	Department Operators	8
D005	Department List	8
D006	Announcement	6
Total		55

test cases for the lecturer module are 52 scenarios; in Table 3, the total test cases for the operator module of the study program are 88 scenarios; in Table 4 the total test cases for the faculty operator module are 55 scenarios.

4.3 Black Box Testing with Equivalence Partitioning and Boundary Value Analysis Methods

- Students Module
  Table 5 is a table of the overall results of student module testing, which consists of 4 modules with a total of 127 test scenarios and 17 defects.
  Figure 3 is a defect that displays HTTP ERROR 500, meaning the server does not recognize a problem when the character in the student's place of birth is more than 255 characters. The display on this defect is a display that UPT PUSTIK has customized, but it would be better if the user were given information regarding the problems.
- Lecturers Module
  Table 6 is a table of the overall results of the lecturer module testing, which consists of 4 modules with a total of 52 test scenarios and 10 defects.

Table 5. Student's module testing results

ID	Testing Module	Total Testing	Total Defect
A001	Log in	13	1
A002	Account Settings	7	0
A003	Students Bio	99	16
A004	Message	8	0
Total		127	17

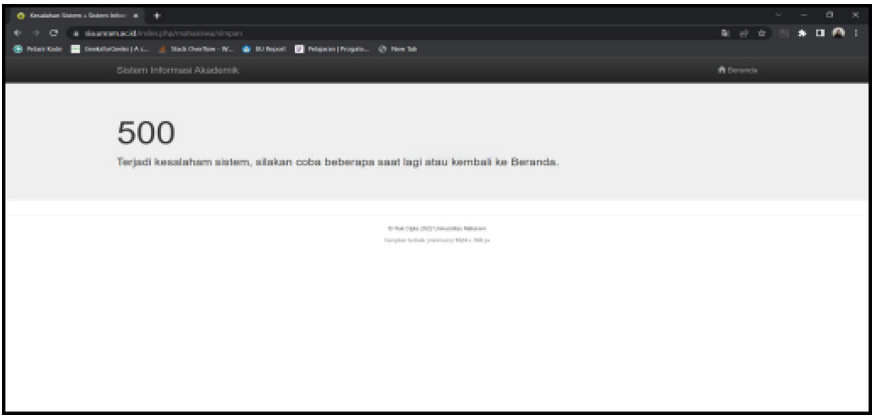

Fig. 3. Students' module defect.

Table 6. Lecturer's module testing results.

ID	Testing Module	Total Testing	Total Defect
B001	Log in	10	1
B002	Account Settings	7	2
B003	Lecturers Bio	27	7
B004	Grades	8	0
Total		52	10

Figure 4 is a defect found in each module, one of which is the lecturer module. In this defect, the system can accept input symbols (′, &, +, ′, ", <, >, ?, \, and /) at login so that actors can still enter the system. This defect proves that even though the input values for the username and password have been set to remove non-alpha-numeric

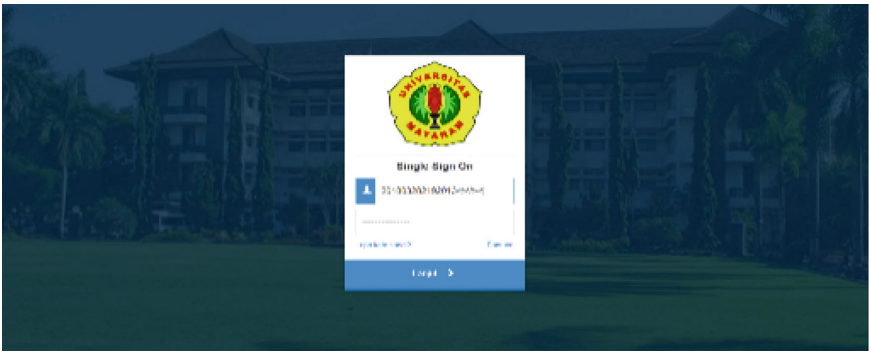

Fig. 4. Lecturer's module defect.

Table 7. Department operator's module testing results.

ID	Testing Module	Total Testing	Total Defect
C001	Log in	10	1
C002	Account Settings	4	1
C003	Department Profile	40	19
C004	Lecturers	8	2
C005	Students	6	3
C006	Course	8	3
C007	Grades	6	2
C008	KRS/KHS/KSP Settings	6	1
Total		88	32

characters or character symbols in SQL, there are still symbols that are not removed so that the user cannot log in.

- Department Operators Module
  Table 7 is a table of the overall results of the department operator module testing, which consists of 8 modules with a total of 88 test scenarios and 32 defects.

  Figure 5 is a defect that displays HTTP ERROR 500, meaning the server does not recognize the problem when the study program operator saves the data for the added courses. The display on this defect is a display that UPT PUSTIK has customized, but it would be better if the user were given information regarding the problems.
- Faculty Operators Module
  Table 8 is the overall table of the test results for the faculty operator module, which consists of 6 modules with a total of 55 test scenarios and 21 defects.

  Figure 6 is a defect found on the faculty operator's announcement page, in which all the features on this page do not work, including new features, search, select category, next, previous, announcement title, delete, and edit.

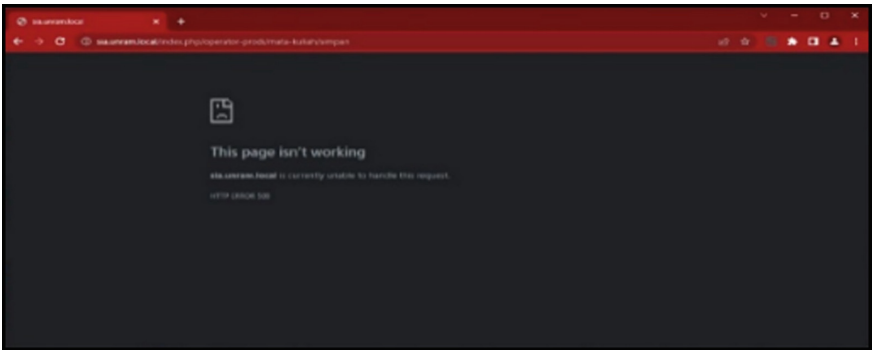

Fig. 5. Department operator's module defect.

Table 8. Faculty operator's module testing results.

ID	Testing Module	Total Testing	Total Defect
D001	Log in	10	2
D002	Account Settings	2	0
D003	Faculty Operators	21	7
D004	Department Operators	8	3
D005	Department List	8	4
D006	Announcement	6	6
Total		55	21

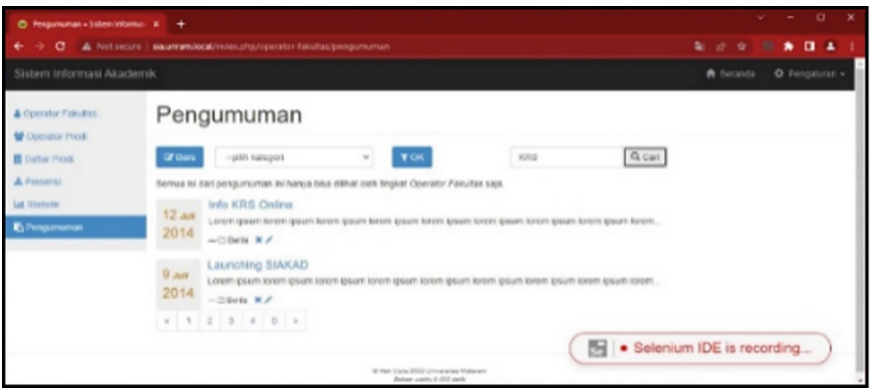

Fig. 6. Faculty operator's module defect.

Table 9. SIA UNRAM testing results.

Testing Module	Total Testing	Total Test Case Pass	Total Defect
Students	127	110	17
Lecturers	52	42	10
Department Operators	88	56	32
Faculty Operators	55	34	21
Total	322	242	80

4.4 Test Results Calculation

Based on the tests that have been carried out, the next step is to calculate the percentage of tests using the test case pass formula to find out the percentage of valid or successful test cases as in Eq. (1) and use the test case failed formula to determine the percentage of invalid test cases such as in the following Eq. (2) [21]:

$$Test\ Case\ Pass = \left(\frac{Test\ Case\ Passed}{Total\ test\ case}\right) \times 100\% \tag{1}$$

$$Test\ Case\ Failed = \left(\frac{Find\ Test\ Case\ Failed}{Total\ Test\ Case}\right) \times 100\% \tag{2}$$

The following are the results of calculations using Eqs. (1) and (2) on the results of the black box test using the equivalence partitioning and boundary value analysis methods performed at SIA UNRAM (Table 9).

The valid value obtained on the total test results of SIA UNRAM is 242 out of 322 total test scenarios, then the calculation using Eqs. (1) and (2) test case pass of the total test results of SIA UNRAM is as follows:

$$Test\ Case\ Pass = \left(\frac{242}{322}\right) \times 100\% = 75.16\%$$

$$Test\ Case\ Failed = \left(\frac{88}{322}\right) \times 100\% = 24.84\%$$

Based on these calculations, the percentage of SIA UNRAM test case passes was 75.16% and the test case failed was 24.84%. So, the results can conclude that the test success scale is in the "Appropriate" category with a success scale of 61% to 80% and a value of 75.16% and there is still a mismatch of functions or features of 24.84%, so improvements are needed on some features or functions of Mataram University SIA by the parties. UPT PUSTIK.

4.5 Feedback from UPT PUSTIK

The researcher gave documentation of the results of this test to UPT PUSTIK on 15 June 2022, then UPT PUSTIK provided feedback regarding the results of the tests that have

been carried out. The UPT PUSTIK said that the test results were sufficient and useful for developing SIA UNRAM. The UPT PUSTIK told that the total defects found were following the number of features belonging to each actor, such as the number of defects found in the study program operator module due to the many features found in the study program operator module and in the faculty operator module many defects were found because the operators rarely used it by the faculty operator so that UPT PUSTIK rarely makes improvements to the features of the faculty operator module.

5 Conclusion

Based on the results of Black Box Testing with the Equivalence Partitioning and Boundary Value Analysis Methods (Study Case: Academic Information System of Mataram University), the researcher can draw the following conclusions:

1. Black Box testing using equivalence partitioning and boundary value analysis methods can find defects in the invalid category in the functions or features contained in the software based on the test cases that have been made. The total defects found in the test were 80 defects from 322 test cases with percentage values including defects found in the log-in feature of all actors, HTTP ERROR 500 defects when saving user data, and GUI incompatibility with inputs.
2. The percentage of valid test scores based on calculations using the test case pass formula is 75.16% and the percentage of valid test scores based on analyses using the test case failed formula is 24.84%. This states that the test success scale is in the "appropriate" category with a success scale of 61% to 80% and there are still some inconsistencies in functions or features, so it is necessary to improve some features or functions of the SIA UNRAM by UPT PUSTIK.
3. The UPT PUSTIK said that the results of the tests were good and useful for the improvement and development of the next SIA UNRAM.

References

1. U. Salamah and F. Nidaul Khasanah.: Pengujian Sistem Informasi Penjualan Undangan Pernikahan Online Berbasis Web Menggunakan Black Box Testing. Inf. Manag. Educ. Prof 2(1), 35–46 (2017).
2. N. W. Rahadi and C. Vikasari.: Pengujian Software Aplikasi Perawatan Barang Milik Negara Menggunakan Metode Black Box Testing Equivalence Partitions. Infotekmesin 11(1), 57–61 (2020).
3. K. Muludi, A. R. Irawati, and E. Priyanto.: Perancangan Sistem Informasi Geografis (SIG) Berbasis Web untuk Penyediaan Informasi Fasilitas dan Personalia di Universitas Lampung. J. Komputasi 1(2), 167–172 (2013).
4. C. Vikasari.: Pengujian Sistem Informasi Magang Industri dengan Metode Blackbox Testing Boundary Value Analysis. Syntax J. Inform 7(1), 44–51 (2018).
5. L. A. A. Ma'ruf, C. Kartiko, and C. Wiguna.: Black Box Testing Boundary Value Analysis pada Aplikasi Submission System. J. Edik Inform 6(2), 15–22 (2020).

6. A. Ijudin and A. Saifudin.: Pengujian Black Box pada Aplikasi Berita Online dengan Menggunakan Metode Boundary Value Analysis. J. Inform. Univ. Pamulang 5(1), 8–12 (2020).
7. Y. D. Leksanti.: Pengujian Website ACC Whistle Menggunakan Metode Black Box Testing Program Studi Informatika. Universitas Atma Jaya (2020).
8. D. Febiharsa, I. M. Sudana, and N. Hudallah.: Uji Fungsionalitas (BlackBox Testing) Sistem Informasi Lembaga Sertifikasi Profesi (SILSP) Batik Dengan AppPerfect Web Test Dan Uji Pengguna. JOINED J 1(2), 117–126 (2018).
9. I. Lucyda.: Analisa Uji Coba Aplikasi Web Tugas Akhir Dalam Jaringan (TADJ) menggunakan Metode Black-Box dan Selenium IDE. Institut Teknologi Bandung (2015).
10. L. Setiyani.: Rekayasa Perangkat Lunak [Software Engineering]. no. May. CV. Jatayu Catra Internusa, Karawang (2018).
11. A. Kadir.: Pengenalan Sistem Informasi Edisi Revisi. Ed. II. Andi Offset, Yogyakarta (2014).
12. J. Chandra W.: Implementasi Sistem Informasi Akademik (Studi Kasus : SMP Negeri 20 Bandung). J. Teknol. dan Inform 1(1), 17–26 (2013).
13. N. Nuari.: Perancangan Aplikasi Layanan Mobile Informasi Administrasi Akademik Berbasis Android Menggunankan Webservice (Studi Kasus Reg. B Universitas Tanjungpura). J. Sist. dan Teknol. Inf 1(0), 1–7 (2014).
14. Romeo.: Testing dan Implementasi Sistem, Edisi I. Ed. I. STIKOM, Surabaya (2003).
15. M. Sharma and R. Angmo.: Web based Automation Testing and Tools. Int. J. Comput. Sci. Inf. Technol 5(1), 908–912 (2014).
16. R. Parlika, T. A. Nisaa, S. M. Ningrum, and B. A. Haque.: Studi Literatur Kekurangan dan Kelebihan Pengujian Black Box. Teknomatika 10(2), 131–140 (2020).
17. M. S. Mustaqbal, R. F. Firdaus, and H. Rahmadi.: Pengujian Aplikasi menggunakan Black Box Testing Boundary Value Analysis (Studi Kasus: Aplikasi Prediksi Kelulusan SNMPTN). JITTER 1(3), 31–36 (2015).
18. Ayuliana.: Testing dan Implementasi: Black Box Testing. 1–6 (2009).
19. J. Devi, K. Bhatia, and R. Sharma.: A Study on Functioning of Selenium Automation Testing Structure. Int. J. Adv. Res. Comput. Sci. Softw. Eng 7(5), 855–862 (2017).
20. N. Uppal and V. Chopra.: Design and Implementation in Selenium IDE with Web Driver. Int. J. Comput. Appl 46(12), 8–11 (2012).
21. I. G. S. Aryandana, A. E. Permanasari, and T. B. Adji.: Comparing method equivalence class partitioning and boundary value analysis with study case add medicine module. IOP Conf. Ser. Mater. Sci. Eng 732(1), 1–8 (2020).