Available online at https://ejournal.corespub.com/index.php/ijmsc/index

International Journal of Mathematics, Statistics, and Computing

e-ISSN 3025-0803

Vol. 2, No. 2, pp. 68-75, 2024

Analysis Testing Black Box and White Box on Application To-Do List Based Web

Dede Irman Pirdaus[1*], Rizki Apriva Hidayana[2]

[1]*Faculty of computer science, University of Informatics and business, Bandung, Indonesia*
[2]*Department of Mathematics, Faculty of Mathematics and Natural Sciences, National University of the Republic of Indonesia, Bandung, Indonesia*
**Corresponding author email: dedeirmanpirdaus@gmail.com*

Abstract

The rapid development of information technology has led to the creation of numerous web-based applications designed to assist human activities and work. One such application is the To-Do List, which helps users manage their tasks and increase productivity. This study aims to analyze the quality of web-based To-Do List applications through black box and white box testing. The research focuses on the login and main pages of the application, where various scenarios are tested to ensure that the system functions as intended. The testing process includes designing test scenarios, creating test cases, executing the test cases, and collecting and processing test result data. The study also includes an analysis of the program's source code using flowcharts and flowgraphs to identify the number of independent logic execution paths and design test cases for white box testing. The results of the testing will help identify errors and weaknesses in the application, ensuring that the final product is of high quality.

Keywords: To-Do List Application, Web-based Application, Black Box Testing, White Box Testing.

1. Introduction

The development of information technology is currently growing very rapidly. This is characterized by the large number of web-based applications developed to assist human activities and work. One application that is currently being developed is a to-do list application or a list of activities that must be done. This application really helps users to manage their activities so that users can be more organized. To-do list apps are software designed to help users organize and track their daily tasks more efficiently (Kumar et al., 2015; Syaikhuddin et al., 2018). Using this application, users can create a list of tasks that need to be completed, set priorities, and set deadlines.

To-do list apps generally offer intuitive and easy-to-use interfaces, allowing users to quickly add, edit, or delete tasks. Some common features in these apps include reminders, categories or projects to group tasks, as well as integration with calendars or other apps. The diversity of to-do list apps allows users to choose the one that best suits their needs and preferences, whether that's through ease of access across multiple devices or integration with the rest of the software ecosystem. To-do list applications help increase productivity and help users stay organized in carrying out daily activities (Kato et al., 2014; Merrigan et al., 2021).

In developing a web-based application, testing is a very important part. Testing is carried out to ensure that the application being built runs well and meets user needs. Software testing has a number of advantages and disadvantages that developers and organizations need to consider. One of the main advantages of software testing is its ability to identify and fix potential bugs or errors in an application's source code (Hussain and Singh 2015; Arcuri, 2020; Golian et al., 2022). A good testing process can also improve software security and reliability, and ensure that applications function according to user needs and expectations.

Testing can help reduce the risk of failure and ensure software quality throughout its development life cycle. However, software testing also has its drawbacks. The testing process can take significant time and expense, especially if carried out thoroughly and continuously. Sometimes, security and reliability aspects of software may be overlooked if testing is not done carefully. Additionally, testing cannot guarantee that the software is completely free of all errors or bugs. Especially in complex software projects, there may be some unexpected conditions or usage scenarios that are difficult to identify during testing. There are various kinds of software testing techniques, one of which is black box and white box testing (Krishna Mohan et al., 2010; Praniffa et al., 2023).

Black box and white box testing are two important methods in evaluating and improving software quality. Black box testing involves checking the functionality of software without paying attention to the internal structure of the source code (Qian, 2018). In this testing, the main focus is on the input and resulting output, as well as how the software responds to various conditions. This approach is similar to the external user perspective in that it only pays attention to what can be seen from outside the system. On the other hand, white box testing includes an internal examination of the structure and logic of the software source code. Testers access and analyze source code to ensure that each part of the program functions as intended. White box testing requires a deep understanding of software design and implementation, thus requiring higher technical expertise.

The advantage of black box testing lies in its ability to evaluate applications from the user's perspective, find bugs or functional errors, and identify application performance in various scenarios. Meanwhile, white box testing provides deep insight into the internal structure of the software, allowing testers to identify and fix errors at the code level. Both play an important role in ensuring software security, reliability and optimal performance. Using a combination of black box and white box testing often results in a holistic testing approach, helping to minimize errors and improve the overall quality of the software. In successful software development, black box and white box testing are not simply considered as separate stages, but rather as integral elements of the overall software development life cycle (Gustinov et al., 2023).

These two types of testing are very important to carry out in order to find errors and deficiencies in the application so that the quality of the application can be guaranteed. In this research, black box and white box testing analysis will be carried out on web-based to-do list applications. It is hoped that the results of this testing can help find errors and weaknesses in the application so that the quality of the application can continue to be improved.

2. Methodology

The first step, Designing test scenarios, involves describing the test scenarios to be carried out. These scenarios are usually written from the user's perspective and include the functionality to be checked. This scenario will then be divided into several test cases. The second step, Designing test cases, involves creating the steps that the tester will follow to test each scenario. These test cases usually include input, procedures, and expected results.

The third step, Test case testing, involves testing each test case that has been created. The tester will execute the specified steps and compare the results with those expected. The fourth step, Test results, involves collecting and processing test result data. This data will be used to determine the success or failure of the software and to identify deficiencies or bugs. For more details, see Figure 1.

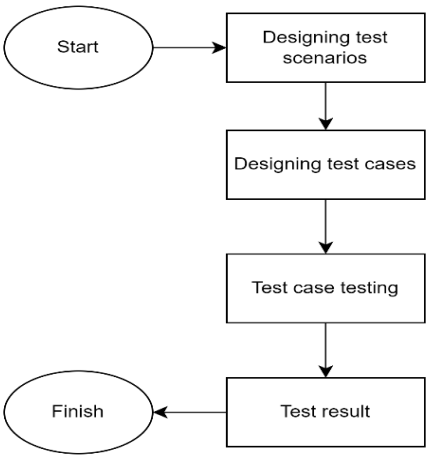

Figure 1: Steps in this Research

In this research stream, it is important to ensure that each step is carried out in a thorough and organized manner. This will help reduce the risk of software failure and improve the quality of the final result. Apart from that, this research flow can be used as a reference in conducting software testing, starting from test scenarios, test case design, testing itself, to test results.

3. Results and discussion

3.1. Black box testing

3.1.1. Program input design

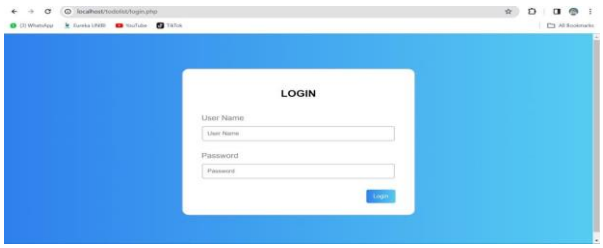

Figure 2: Login page user interface

The test results on the login page will be explained in table 1 as follows:

Table 1: Testing page login

No.	Scenario Testing	Results Testing	The results expected	Information
1.	Username and Passwords Correct	appear message ”Welcome, Name user!”	Entered into page main "Welcome, Name user!”	Valid
2.	Username and Password not in fill Then click Login	Appear message ” Username and Password are required"	The system doesn't Can direct to the yard furthermore And notification " Username and Password are required”	Valid
3.	If only just the username in fill Then click Login	Notification ” Username is required”	The system doesn't Can direct to the yard furthermore And notification " Username is required”	Valid
4.	If only just the password in fill Then click Login	Appear message ”Password is required"	The system doesn't Can direct to the yard furthermore And appear message ” Password is required"	Valid
5.	If username or Wrong Then click login	Appear message ” Incorect Username or password”	The system doesn't Can direct to the yard furthermore And notification " ” Incorect Username or password"	Valid

The following is an explanation of the black box test results on the login page:

1. The test scenario of entering the correct username and password produces the message "Welcome, User Name!" as expected. Valid testing.
2. The test scenario of not entering the username and password and then pressing the login button displays the message "Username and Password are required" as expected. Valid testing.
3. The test scenario of simply entering the username then pressing login displays the message "Username is required" as expected. Valid testing.
4. The test scenario of simply entering the password then pressing login displays the message "Password is required" as expected. Valid testing.
5. The test scenario of entering the wrong username or password then pressing login displays the message "Incorrect Username or Password" as expected. Valid testing.

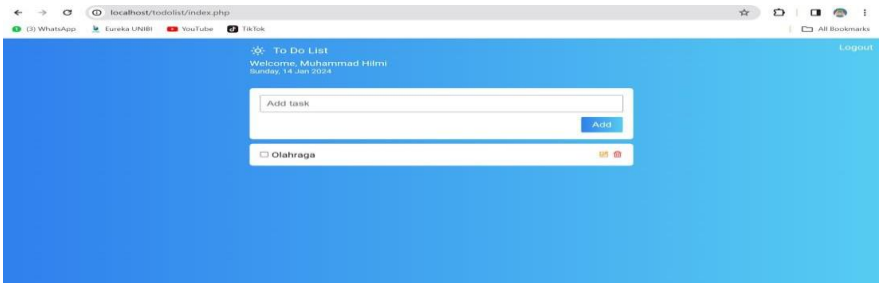

Figure 3: Main page user interface

The test results on the main page will be explained in table 2 as follows:

Table 2: Test results

No.	Scenario Testing	Results Testing	The results expected	Information
1.	When column add tasks are not filled in And click Add	Notification ” Column no can blank"	Can not fill in the data and Appear message ” Column No can blank"	Valid
2.	When column tasks in fill with, for example sports And click Add	Input data go to tasks label	Input data go to tasks labels and databases todolist	Valid
3.	When click checkbox on tasks label	Checkboxes become ter- uncheck And Tasks status become 'close'	Checkboxes become ter- uncheck And Task status become 'close' on databases	Valid
4.	When you click the button edit on tasks label	Directed to edit page and input fields editable by user in accordance need	User directed back to main page with that change applied	Valid
5.	When you click the button delete on task label	Notification ” Are you sure to delete this data!"	Data tasks label succeed erased on databases	Valid
6.	When click log out	Notification ” Are you sure you want to logs out?”	If select yes Directed back to login page, If choose cancel will remain is at on page main	Valid

The following is an explanation of the black box test results on the main page:

1. Test scenarios where the add task column is empty and then click add displays the notification "Column cannot be empty" as expected. Valid testing.
2. Test scenario by filling in the task column then clicking add, the data is successfully entered into the task label and database. Valid testing.
3. Test scenario by clicking the checkbox on the task label, the checkbox status and the task status in the database change as expected. Valid testing.
4. Test scenario by clicking the edit button on the task label, the edit page opens and the user can change the data. After returning to the main page, the changes take effect. Valid testing.
5. Test scenario by clicking the delete button on the task label, a deletion confirmation appears. If selected yes, the task data is deleted from the database. Valid testing.
6. Test scenario by clicking logout, confirmation appears. If yes, return to the login page. Otherwise, stay on the main page. Valid testing.

3.2. White Box Testing

3.2.1. Code program web to-do-list

```
<?php
session_start();
include "database.php";
if (isset($_POST['uname']) && isset($_POST['password'])) {
function validate($data){
}
$data = trim($data);
$data stripslashes($data);
$data = htmlspecialchars($data); return $data;
$uname = validate($_POST['uname']); $pass = validate($_POST['password']);
if (empty($uname) && empty($pass)) {
header("Location: login.php?error=Username and Password are required"); exit();
}else if(empty($uname)) {
header("Location: login.php?error-Username is required"); exit();
```

```
}else if(empty($pass)){
header("Location: login.php?error-Password is required"); exit();
}else{
$sql = "SELECT * FROM users WHERE username='$uname' AND password='$pass'";
$result = mysqli_query($conn, $sql);                                                    (1)
if (mysqli_num_rows($result) = 1) {
$row = mysqli_fetch_assoc($result);
if ($row['username'] $uname 66 $row['password'] = $pass) {
$_SESSION['username'] = $row['username'];
$_SESSION['name'] = $row['name']; $_SESSION['id'] = $row['id'];
echo "<script> alert('Welcome,
$_SESSION['name'] . "!');
window.location.href="index.php';</script>";
exit();
}else{
}else{
header("Location: login.php?error-Incorect User name or password"); exit();
header("Location: login.php?error-Incorect Username or password"); exit();
}else{
header("Location: login.php");
exit();
```

3.2.2. Flowchart web to-do-list

Flowcharts aim to facilitate understanding of the overall application workflow, especially in conducting white box testing. Flowcharts are useful for analyzing the cyclomatic complexity of program code.

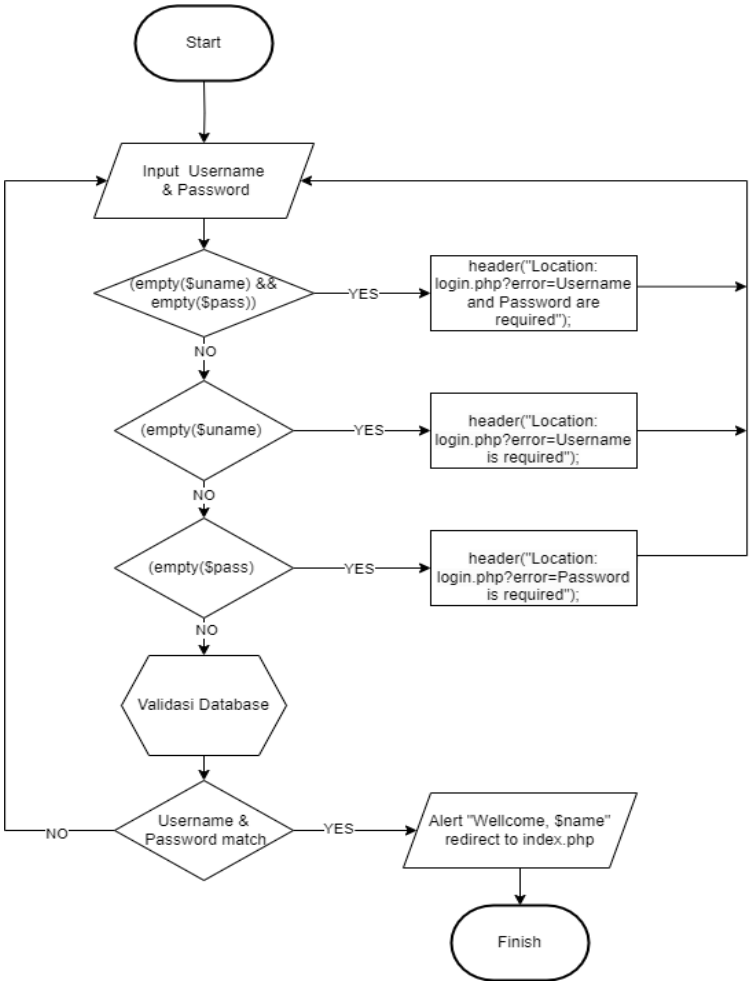

Figure 4: Flowchart

3.2.3. Flow graph web to-do-list

Flowgraph aims to analyze and model all possible logic paths of a program/source code. The following web to-do-list flowgraph can be seen in Figure 5.

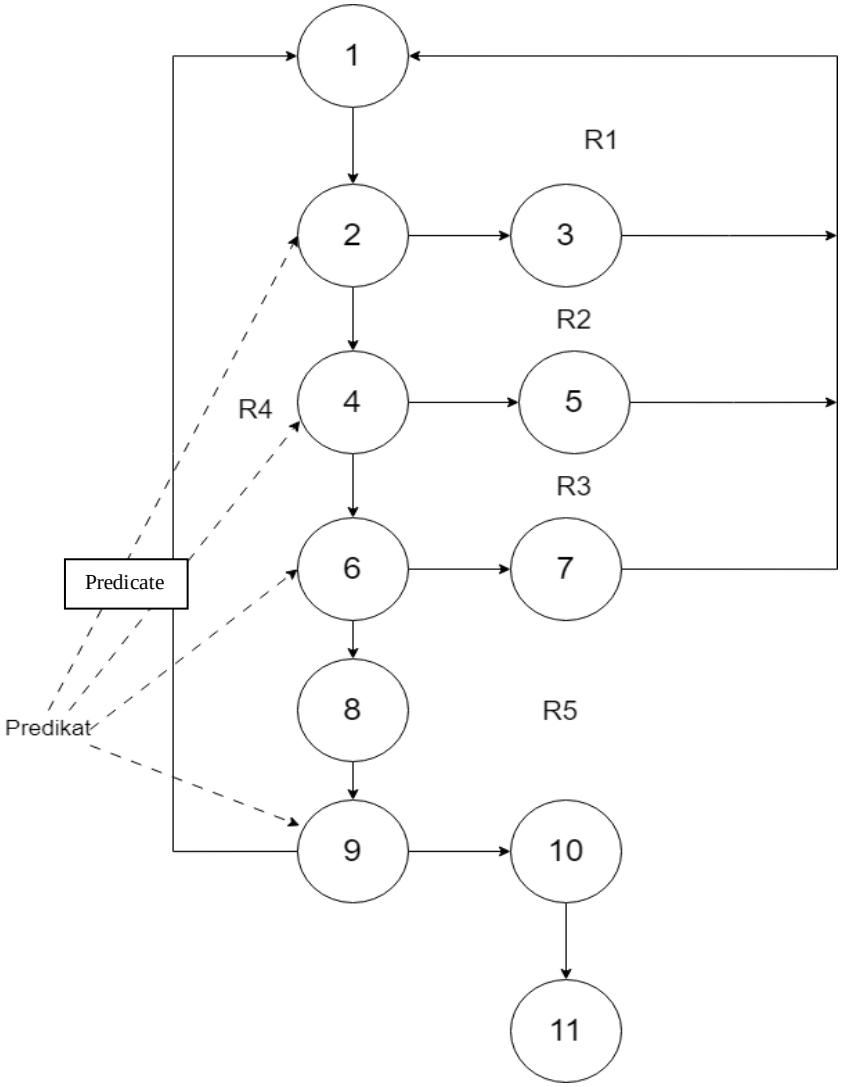

Figure 5: The Flowgraph

From the flowgraph results above, it consists of 11 nodes which represent the number of statements in the program, 14 edges which show the control flow between statements, 5 regions which are independent paths of the program, and 4 predicate nodes as decision nodes. From the flowgraph it can be seen that the program has 5 independent logic execution paths. The basic path of the program is shown by region 1 (R1) with a node sequence of 1-2-3-1. Then there are 4 control branches, namely R2 1-2-4-5-1, R3 1-2-4-6-7-1, R4 1-2-4-6-8-9-1, and R5 1-2 -4-6-8-9-10-11. R5 is the most complex path with more nodes.

V(g) = E – N + 2	V(G)= P+1	V(G) = R	(2)
V(G) = 14 – 11 + 2	V(G)= 4 + 1	V(G) = 5	
V = 5	V(G) = 5		

It can be concluded that the program in this example has 5 cyclomatic complexities. This means that there are 5 independent paths that must be tested in the program to achieve optimal white box testing coverage. It can be concluded that the base path is:

Path 2 = 1-2-4-5-1
Path 3 = 1-2-4-6-7-1
Path 4 = 1-2-4-6-8-9-1
Path 5 = 1-2-4-6-8-9-10-11

Next, graph matrix testing will be carried out which aims to help design test cases in white box testing.

Table 3: Graph matrix testing

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Results
1		1													0
2			1	1											1
3	1														0
4					1	1									1
5	1														0
6							1	1							1
7	1														0
8									1						0
9	1									1					1
10											1				0
11															0
Qty															5

Based on results testing Which has done obtained results total from 5 track Which There is there is 1 track Which passes, 4 track Which is condition For catch wrong, and 0 lines the file.

4. Conclussion

Based on a series of tests carried out on the To-Do List Application web-based, it can be concluded that the system has passed a series of tests try it with satisfactory results. The trial consisted of two main approaches, viz White-Box Testing And Black-Box Testing. In Black Box Testing, plan the detailed tests cover the various scenarios on the page login, page main and edit pages.

References

Arcuri, A. (2020). Automated black-and white-box testing of restful apis with evomaster. *IEEE Software*, *38*(3), 72-78.

Golian, N. V., Golian, V. V., & Afanasieva, I. V. (2022). *Black and white-box unit testing for web applications.*

Gustinov, M. D., Azani, N. W., Al Ghani, R., Auliani, S. N., Maharani, S., Hamzah, M. L., & Rizki, M. (2023). Analysis of Web-Based E-Commerce Testing Using Black Box and White Box Methods. *International Journal of Information System and Innovation Management (IJISIM), 1*(1), 20-31.

Hussain, T., & Singh, S. (2015). A comparative study of software testing techniques viz. white box testing black box testing and gray box testing. *(IJAPRR), ISSN*, 2350-1294.

Kato, J., Sakamoto, D., Igarashi, T., & Goto, M. (2014, October). Sharedo: to-do list interface for human-agent task sharing. *In Proceedings of the second international conference on Human-agent interaction* (pp. 345-351).

Krishna Mohan, K., Verma, A. K., Srividya, A., & Papic, L. (2010). Integration of black-box and white-box modeling approaches for software reliability estimation. *International Journal of Reliability, Quality and Safety Engineering*, *17*(03), 261-273.

Kumar, M., Singh, S. K., & Dwivedi, R. K. (2015). A comparative study of black box testing and white box testing techniques. *International Journal of Advanced Research in Computer Science and Management Studies, 3*(10).

Merrigan, K. K. A., Giraud, E. G., & Greene, C. (2021). The Critical To-Do List for Organic Agriculture: 46 Recommendations for the President.

Praniffa, A. C., Syahri, A., Sandes, F., Fariha, U., Giansyah, Q. A., & Hamzah, M. (2023). Pengujian Sistem Informasi Parkir Berbasis Web Pada UIN SUSKA RIAU Menggunakan White Box dan Black Box Testing. *Jurnal Testing Dan Implementasi Sistem Informasi, 1*(1), 1-6.

Qian, Z. Z. J. (2018). Test Suite Augmentation via Integrating Black-and White-Box Testing Techniques. *International Journal of Performability Engineering*, *14*(6), 1324.

Syaikhuddin, M. M., Anam, C., Rinaldi, A. R., & Conoras, M. E. B. (2018*). Conventional software testing uses the white box method.* Kinetics: game technology, information systems, computer networks, computing, electronics, and control, 65-72.