Journal of Applied and Research Computer Science and Information Systems
Vol. 3, No. 1, June 2025, pp. 240~254
ISSN: 2988-294X, DOI: 10.61098/jarcis.v3i1.257

White Box Testing with Path Testing on the Web-Based Population and Civil Registration Service (Dukcapil) Submission Status Notification Module in Kuningan Barat Subdistrict

Dimas Abimanyu Panji[1], **Safrizal** [1],
[1]Sistem Informasi, Universitas Pembangunan Jaya, Indonesia
safrizal.abdurrahman@upj.ac.id

Article Info

Article history:

Received May 30, 2025
Revised June 30, 2025
Accepted June 30, 2025

Keywords:

White Box Testing,
Path Testing,
Service Notification,
Software Testing.
DUKCAPIL

ABSTRACT

Software testing is a crucial stage in ensuring the quality and reliability of the system before it is implemented operationally. This study aims to evaluate the accuracy of the program logic flow in the web-based DUKCAPIL service submission status notification module in Kuningan Barat Village using the white box testing method with a path testing approach. This method analyzes the internal structure of the program code to ensure that all logic paths have been thoroughly tested. The testing process begins with the creation of a Control Flow Graph (CFG) to map the program control flow, followed by the calculation of cyclomatic complexity using the McCabe formula, and determining the basis path to form a representative test set. Each test path obtained is then tested manually and/or with the help of debugging tools to ensure the correctness of the logic execution. The results of the study show that there are 5 basis paths formed from the notification module. All of these paths have been successfully tested without any anomalies or logical errors detected. The test shows that the decision-making structure and notification delivery run as expected, both for approved and rejected submission statuses.

Corresponding Author:

Safrizal
Universitas Pembangunan Jaya, Indonesia
Email: safrizal.abdurrahman@upj.ac.id

1. INTRODUCTION

White Box Testing with Path Testing on the Web-Based DUKCAPIL Service Submission Status Notification Module in Kuningan Barat Village is a software testing process by examining the internal structure of the module's code to ensure that all paths in the program have been tested correctly. [1]. The research focused on the login page and the main page of the application, where various scenarios were tested to ensure the system functions as intended. The testing process includes designing test scenarios, creating test cases, executing test cases, and collecting and processing test result data.[2].The disadvantage of Path testing on White Box is that it requires a deep understanding of the internal structure of the code and program logic. Testing the paths in the code, logic paths that should exist but will not be detected if not implemented. [3] The advantages of path testing on White Box testing are that it detects logic errors comprehensively, checks all execution paths, so that it can find logic errors that are not detected by other methods. High test coverage of program control structures Path testing forces the tester to test all branching structures (if, case, loop), thereby increasing code quality assurance [4]. Identifying paths that are never executed (dead code) With path analysis, parts of the program that do not contribute to the output can be found. Improving understanding of algorithm design and program logic because it requires understanding the structure in the code, path testing helps developers understand the logic of the system [5]. Can be automated with modern testing tools, many testing tools support path testing to identify complex paths automatically. [6]. The problem of White Box Testing (Path Testing) in the DUKCAPIL Submission Status Notification Module is that the submission status notification module has many branching

conditions such as pending status, verification process, rejected, or approved, resulting in many execution paths. The notification module usually depends on the results of the submission and verification modules. [7] The problem in this study is the difficulty of identifying Independent Paths in white box testing, especially path testing. There are many decision points and loops, which results in a very complex CFG, making it difficult to determine independent paths manuall [8], [9] The gap analysis in this study is that many studies Testing is done manually without automation tools. There is no official report on path coverage, Error handling only records logs without notification to the admin [10].

White box testing is performed at the unit, integration, and system levels to detect logical errors, debug, and validate programming assumptions. This method includes thorough testing of all code paths, assists in optimization, and provides guidelines for stopping testing. [11]. Path Testing is one of the techniques in white box testing that focuses on testing all logical execution paths in the program code. This will produce a more stable, accurate, and reliable system. [12] White box testing is a software testing method that involves an in-depth understanding of the internal structure of the program. One of the techniques used in White Box Testing is Basic Path Testing, which focuses on testing each execution path in the program. This testing aims to ensure that the program functions properly and can manage the lending process properly [13] White-box testing is done by analyzing the application source code first, determining the files and functions that must be tested, and creating a flow graph from the source code [14] The tests produced from white box testing depend on the flowchart that is transformed into a flowgraph. The white box testing method with the basic path technique produces test tests that are increasingly large along with the size of the system being tested. White box testing is also known as structural testing, where the tester has access to the entire structure or logic of the software being tested [15]. The steps used for path testing are Flowchart/Sourcecode analysis, create Flow Graph, Cyclomatic Complexity, Independent Path Determination, Test case design. Cyclomatic Complexity is a measure of the number of different execution paths in a program. The higher the number, the more complex the code is, and the more difficult it is to test and maintain.

$V(G) = E – N + 2.$

Description:

E = number of edges in the flowgraph

N = number of nodes in the flowgraph,

P = number of predicates nodes in the flowgraph [16].

This study uses White Box testing, namely independent path testing, flow graph development, cyclomatic complexity calculations, and graph matrix development. [17]. The white box testing method prioritizes knowledge of the system structure. In real practice, this method can be applied in testing information system products to ensure their reliability and quality [18]. Research by taking a case study on the inventory information system of goods in a goods delivery service business. This information system testing technique uses the black box testing method, with the Equivalence Partitioning testing technique. [19] White box testing requires a deep understanding of the internal structure of the software and the programming language used. This makes it less suitable for testing teams that do not have adequate technical knowledge or access to the source code. White Box Testing functions as Clear Box Testing or Structural Testing to assess the internal structure of the application and logical functions and programming code. [20].

Web application testing is a software practice designed to ensure quality by verifying that a particular web application functionality is working correctly or according to the specified requirements. White box testing is now the most widely used decision. So, this testing also has some features and can be chosen to be used on a project, but it should be the right choice with an understanding of all the consequences behind it. [21].

This study applies White Box testing with a path testing approach, which includes creating program flow diagrams, assessing cyclomatic complexity, and identifying and testing independent paths in the code. This technique is used to ensure that the entire internal logic structure of the program is thoroughly tested.[22]. This study aims to produce a method for optimizing white box testing on Java programs by utilizing branching and repetition structures using the basis path method [23].

The White box testing method is adapted for transformer-based NLP models. The method includes Mask Neuron Coverage (Mncover) which measures how thoroughly the attention layer in the model is implemented during testing. [24]. White box testing focuses on the control structure in the program, to verify all statements in the program are executed at least once and all logical conditions are executed. Testing is done to verify functional requirements. It includes equivalence partitioning, boundary value analysis, error-based testing, randomized testing, partition testing, causal graph techniques.[25]. This research aims to equip algorithms with explainable AI features to enhance transparency. The findings of this research highlight a promising path for integrating black-box algorithm performance with the need for transparency in critical decision-making domains.[26].

Software testing is an important term for software reliability. Testing provides the original structure and validity to software for efficient performance under operational conditions.[27]. Evolutionary white-box testing is a promising approach for the complete automation of structure-oriented test case generation. Evolutionary testing cannot find valid test data. This can lead to increased efficiency and effectiveness of evolutionary white-box testing.[28] Evolutionary white-box testing has so far not been able to make statements about the existence of unattainable test goals, which can be improved with the help of evolutionary software measurements.[29]. This research shows that this method successfully identifies and fixes errors in programming logic as well as system vulnerabilities, which contribute to a 20% performance improvement, increased stability, and reduced risk to security threats.[30].

White box testing with path-based techniques is applied to the population administration system website, which is a computer application for conducting population administration. It was found that the path of each feature that has been passed has been in accordance with the results of Cyclomatic Complexity and Flowgraph that were created.[31]. This test explains the steps to perform data flow testing and how to design a series of tests that take anomalies into account. This approach includes node-based design, trend discovery coverage, web application comparison, and analytical testing.[32]. This study uses White Box testing, namely Path Testing. The methodology is by conducting Control Flow Graph (CFG) Analysis of the source code of each main function, analyzing the program's logic flow such as branching, repetition, and conditions. Determination of the Basis Path and Independent Path by calculating Cyclomatic Complexity and compiling test cases based on the specified logic path [33].

2. METHOD

White Box Testing with Path Testing on the Web-Based Population and Civil Registration (Dukcapil) Service Submission Status Notification Module in Kuningan Barat Village is applied to the Web-Based DUKCAPIL Service Submission Status Notification Module in Kuningan Barat Village South Jakarta. This Web-Based DUKCAPIL Service Submission is to serve the community in managing population documents such as KTP, Family Card, and other civil registration certificates. From the DUKCAPIL Service Submission Application, the Service Submission Status Notification module is used to be tested using Path Testing on White Box Testing. The research methodology used is to use the Software Testing Life Cycle (STLC) and White Box Testing Stages. Regarding STLC and the White Box Testing Stages, they are explained as follows: Software Testing Life Cycle (STLC) is a systematic process consisting of several stages carried out to ensure the quality and reliability of software through structured testing.

STLC is a series of stages carried out in the software testing process, starting from test planning to test closing, with the aim of finding and fixing bugs, and ensuring that the software works according to the expected specifications. A. Stages in STLC:

a) Requirement Analysis
   The QA (Quality Assurance) team analyzes the requirement document to understand what needs to be tested. Purpose: to identify testing needs and test feasibility.
b) Test Planning
   Develop a testing strategy, determine resources, budget, schedule, and tools to be used.Main output: Test Plan or Test Strategy Document.
c) Test Case Development
   Write test cases and test scripts based on functional and non-functional requirements. Also includes test data preparation.
d) Test Environment Setup
   Prepare the environment needed to run the test, such as servers, databases, and supporting software.
e) Test Execution
   Run test cases and record the results.Bugs or errors found will be reported using a bug tracking tool.
f) Test Cycle Closure
   Evaluate the entire test cycle, document the final results, lessons learned, and submit a final test report.
   Figure 1 shows the stages in the Software Testing Life Cycle (STLC)*)*

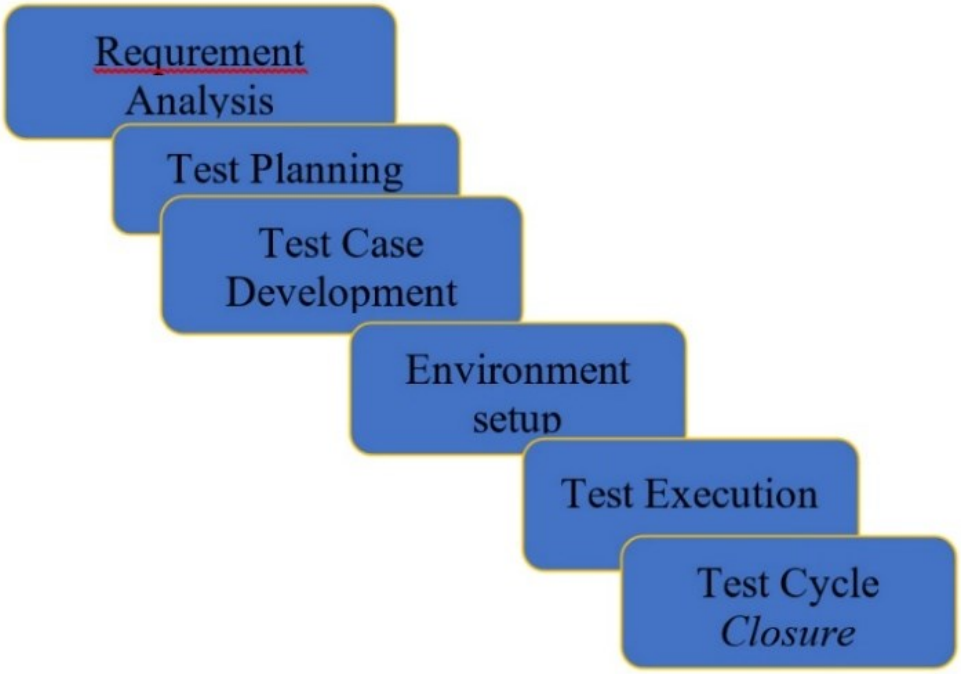

Figure 1. *Software Testing Life Cycle (STLC)*

2.1. White Box Testing Stages

The research methodology used is taken from the Software Testing Life Cycle (STLC) in step 5 of Test Execution. In Test Execution, the White Box testing stage is carried out which contains, among others: Flowchart/Sourcecode analysis, Flow Graph, Cyclomatic Complexity, Independent path, Test Case Design. The testing stages are as illustrated in Figure 2.

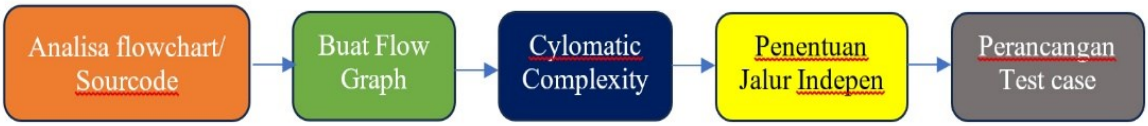

Figure 2. White Box Testing Stages

The explanation of Figure 2 is as follows:

a) Flowchart/Sourcecode analysis, is a graphical representation of a process or logical flow. In software testing, flowcharts are used to: visualize business processes or system logic. useful for understanding how the program flow runs, determining critical points for testing (decision points, loops, input-output).
b) Flow Graph is a directed graph that describes the flow of program execution. Nodes represent blocks of instructions, and edges indicate the direction of program execution.
c) Cylomatic Complexity is a quantitative measure of the number of independent execution paths in a program or module. Introduced by Thomas McCabe in 1976, this metric is used to: Assess the logical complexity of code, Determine the minimum number of test cases required for full path coverage. Assist in the process of code refactoring and code maintenance.
d) An independent path is a path in the program control flow (flow graph) that passes through at least one new edge (new direction) that has not been passed by any other path. Represents a unique combination of branching conditions, loops, or other logical structures. The goal is to: identify all major logical paths in the code. Design minimal test cases that touch the entire program logic.
e) Test Case Design

After determining the independent path from the flow graph above, the next step is to create a test case at least as many independent paths as have been created, ensuring that every possible path that will be passed is made a test case.

3. RESULTS AND DISCUSSION

3.1. White Box Testing Stages

In the diagram, the main activities that can be done by the user include submitting a service request, viewing notifications, checking history, and changing profiles. All of these features begin with the login process, which is a mandatory step to access the system. In addition, users who do not have an account can first register. This diagram shows the relationship between use cases with the <<include>> element, which indicates that login is an important part of every activity carried out by the user in the system. From the Usecase in Figure 3, the description of the use case process above is described, namely, login, registration as a user, submitting a service request, viewing notifications, changing profiles. A detailed and systematic explanation is described below. Figure 3 illustrates a user interacting with the system to access various main features.

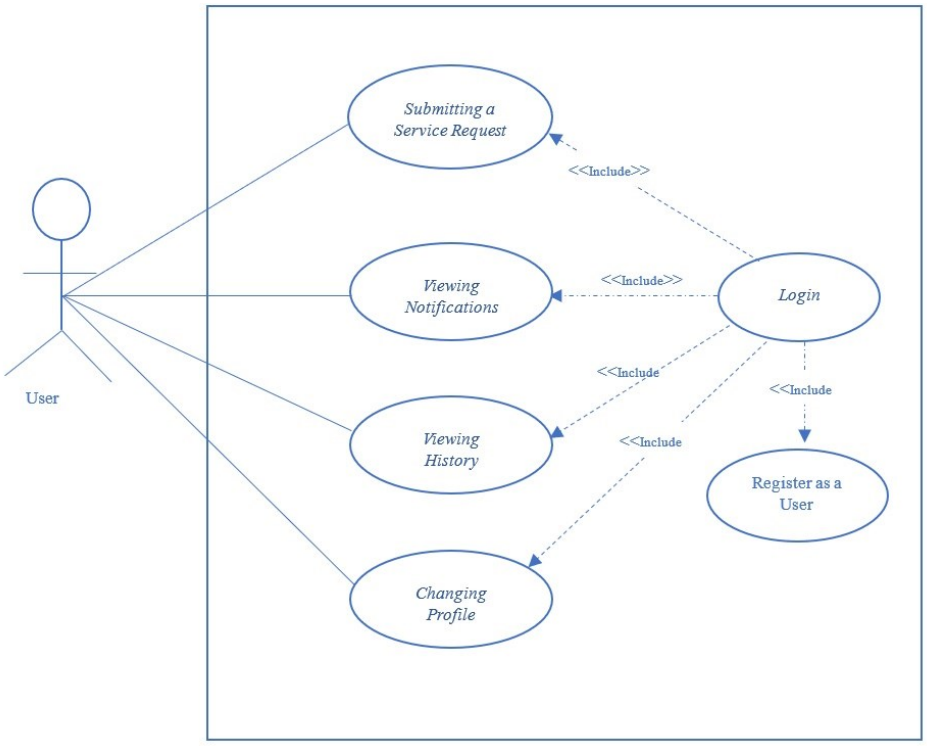

Figure 3 Usecase for Submitting Service Requests

1. Login

Table 1. Description of Login Usecase Diagram

Use Case Name: Login	**ID:** UC01	**Priority:** High
Description	*Users, Admins and Super Admins must log in first before using the application.*	
Actor	*User, Admin* and *Super admin*	
Trigger	*User, Admin and Super admin enter username/NIK and password on the login page.*	
Pre-Condition	The account is already registered in the system.	
Post-Condition	The system allows access according to the user's role.	
Normal Flow	*1. User, Admin and Super admin access the login page.* *2. User, Admin and Super admin fill in the username and password.* *3. The system verifies the credentials.* *4. The system checks the role (User, Admin, or Super admin) to determine access.*	
Subflows	-	

Exceptional Flow	The system displays an error message if the username/NIK or password is incorrect.

2. Register as a User

Table 2. Description of Usecase Diagram Registration as User

Use Case Name: Register as a User	**ID:** UC02	**Priority:** High
Description	*Users must register as users to apply for services.*	
Actor	*User*	
Trigger	*User selects the registration option.*	
Pre-Condition	The system has an active registration module.	
Post-Condition	User data is saved and account is successfully created.	
Normal Flow	*1. User opens the registration page.* *2. User fills out the registration form.* *3. The system saves the new user data.*	
Subflows	-	
Exceptional Flow	The system displays an error message if the registration data is incomplete or has already been used.	

3. Submitting a Service Request

Table 3 Description of Usecase Diagram Submitting a Service Request

Use Case Name: Submitting a Service Request	**ID:** UC03	**Priority:** High
Description	*Registered users can submit service requests according to their needs.*	
Actor	*User*	
Trigger	*The user is in the user dashboard.*	
Pre-Condition	*The user has logged into the system.*	
Post-Condition	Application data is saved and notification is provided.	
Normal Flow	1. The system displays the types of services available. 2. The user selects the type of service desired. 3. The system displays the application form. 4. The user fills out the form (if the application is for a Letter of Transfer), uploads the required files and submits the application.	
Subflows	1. The user selects the type of service desired such as: KK, KTP, KIA, or Transfer Letter. 2. The system saves the type of service along with the request details.	
Exceptional Flow	The system displays a message if the application data is incomplete or invalid.	

3. Viewing Notifications

Table 4 Description of Usecase Diagram Viewing Notifications

Use Case Name: Melihat Notifikasi	**ID:** UC04	**Priority:** Medium
Description	*Users will receive a notification regarding the status of their application in the notification menu.*	
Actor	*User*	
Trigger	*User selects the notification menu.*	
Pre-Condition	Notifications are already available in the system.	
Post-Condition	Notifications are marked as read.	
Normal Flow	*1. User opens the notification list.*	

	2. The system displays all notifications.
Subflows	-
Exceptional Flow	The system displays a message if no notifications are available.

4. Viewing History

Table 5 Description of Usecase Diagram Viewing History

Use Case Name: ***Viewing History***	**ID:** UC05	**Priority:** Medium
Description	User request activity is stored in the history menu.	
Actor	User	
Trigger	Users select the history menu to view the history of previous activities or requests.	
Pre-Condition	The system has historical data of user activity.	
Post-Condition	Users can view detailed history of activities or requests that have been submitted.	
Normal Flow	1. User selects the history menu. 2. The system displays a list of activity history. 3. User can view the entire history data along with additional messages and files that have been uploaded.	
Subflows	-	
Exceptional Flow	The system displays a message if no activity history is available.	

5. Changing Profile

Table 6 Description of Usecase Diagram Changing Profile

Use Case Name: Mengubah *Profile*	**ID:** UC06	**Priority:** Medium
Description	Users can change their profile photo, username and password and can also delete their data permanently.	
Actor	User	
Trigger	User selects a menu to change profile data.	
Pre-Condition	The user has logged into the system.	
Post-Condition	New User profile data is saved in the system.	
Normal Flow	1. User opens the profile menu. 2. User edits the desired profile data. 3. The system saves the data changes.	
Subflows	1. User can change one or both of profile photo and username. 2. User can change password. 3.User can delete user data permanently from database.	
Exceptional Flow	The system displays an error message if there is invalid data or if it fails to save.	

From Figure 3, a user interface is created regarding the login page, Dashboard, Upload application files, History page, Notifications. The following describes the User Interface of each of the above, namely:

6. Desktop Version User Interface Display

Figure 4 shows the user interface for Login.

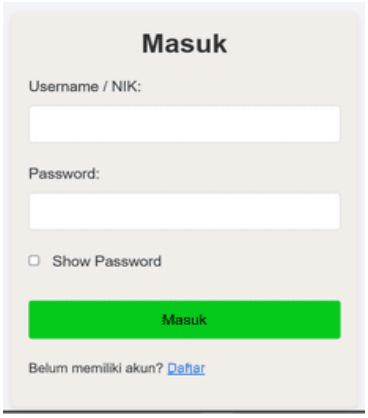

Figure 4. Login Page

7. User Dashboard

Figure 5 shows the user interface for the Dashboard.

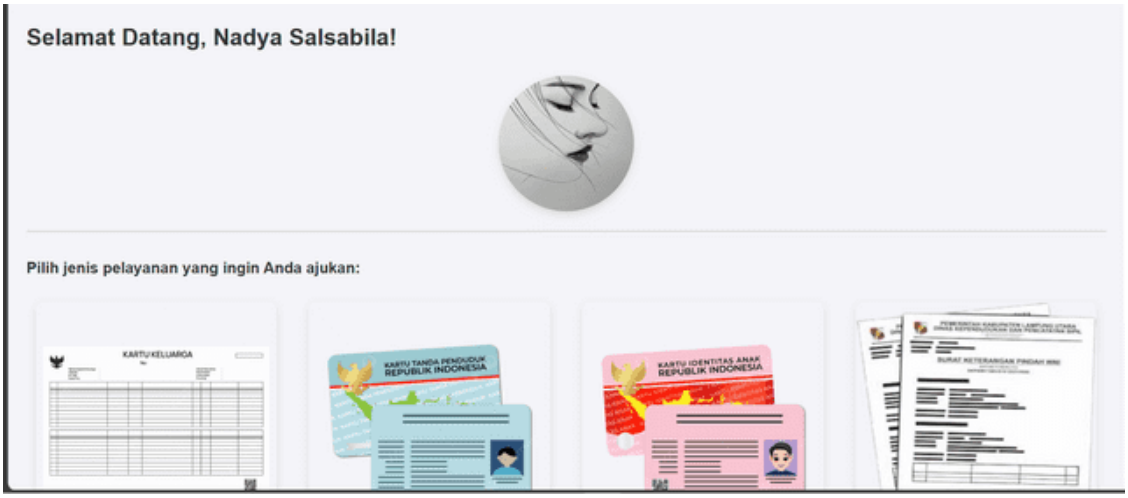

Figure 5 User Dashboard

8. Submission of Application

Figure 6 shows the User Interface for Uploading Application Files.

Figure 6 Submission of Application

9. ***User History***

Figure 7 shows the user interface regarding Application History

Riwayat Permohonan

Total Permohonan: 4

No.	Tanggal Permohonan	Jenis Pelayanan	Nama Berkas	Status	Pesan
1	07-12-2024 23:14	Kartu Identitas Anak (KIA)	Akta Lahir Foto Anak Foto Copy KK	Ditolak	-
2	07-12-2024 23:14	Kartu Tanda Penduduk (KTP)	Surat Kehilangan Kepolisian KTP Foto Copy KK	Pending	
3	07-12-2024 23:13	Kartu Keluarga (KK)	KK Asli/Foto Copy KK Akta Kawin/Cerai	Pending	Harry anggara : Gol. Darah : O, Pekerjaan : Kary. Swasta Hanna Michella : Gol. Darah : tidak tahu, pekerjaan : ibu rumah tangga Gabriel baschin : pekerjaan : belum bekerja, pendidikan : sd kelas 4 Anastashia angelica : pekerjaan : belum bekerja, pendidikan

Figure 7 User History

10. ***User Notification***

Figure 8 shows the user notification user interface.

Figure 8 User Notification

3.2. Flowchart analysis

From Usecase image 3, a use case is taken about viewing notifications, from that use case a flowchart is made, from that flowchart a flow diagram is made, Cyclomatic Complexity is calculated then an independent path is determined and from the independent path a test case design is made. Those are the stages of making a testing path from white box testing. These stages are described below.

3.2.1. Flowchart analysis

From the use case in Figure 3, the Service Request Submission Use case, the flowchart is made as follows:

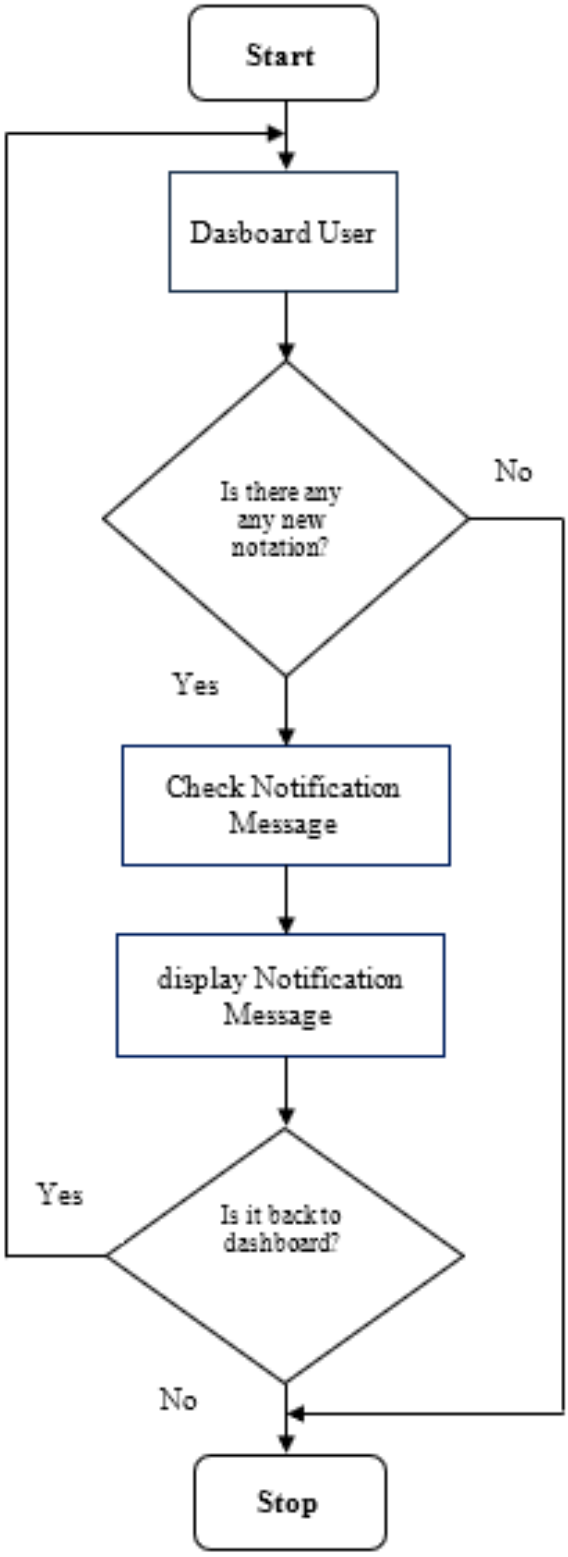

Figure 9. Notification Module Flowchart

The explanation of the flowchart in figure 9 is as follows: Enter the user dashboard, after entering the dashboard, enter the notification menu. In the notification menu ask if there is a notification. If there is a notification, check the notification message, display the notification message If there is no notification then the process is complete. After completing the notification stage. The next stage displays the menu whether to return to the Dashboard. If yes, return to the Dashboard menu. If not, the process is complete.

3.2.2. Flow Graph

The creation of a Flow Graph from the Flowchart above is as follows:

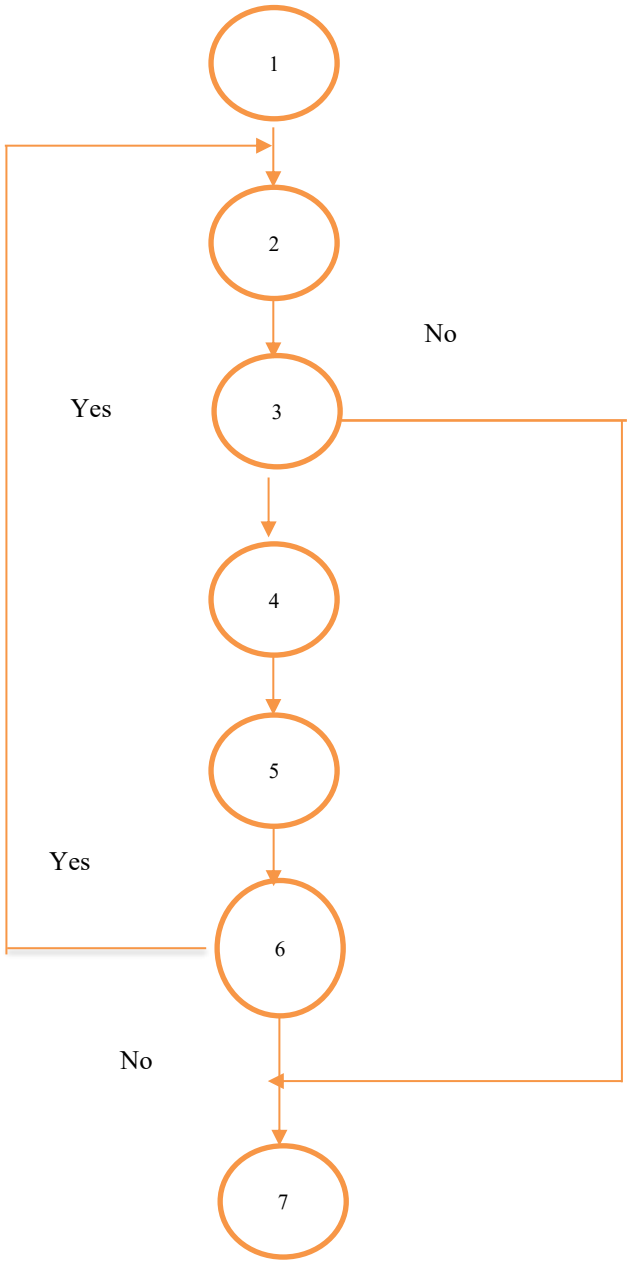

Figure 10 Flow Graph Notification Module

The explanation of figure 10 is as follows, from figure 9, the flow graph is made start as node 1 which is symbolized by a circle and written with number 1. Then enter the user dashboard menu called node 2 which is symbolized by a circle and written with number 2. From node 2, the next stage is node 3, which is symbolized by a circle and written with number 3. Node 3 shows the question menu, namely whether there is a new notification if there is a new notification then continue to node 4. Node 4 shows check notification messages. After going to node 4, continue to node 5. Node 5 shows display notification messages. Then go to node 6. Node 6 shows the question back to the dashboard, if the answer is yes then the next process is to node 2. If at node 6, the answer is no then the next process is to node 7. Node 7 shows the process is complete. If the

question at node 3 is there a question there is a new notification, if the answer is no there is a new notification then the process is complete, namely directly to node 7. Node 7 shows the process is complete.

3.2.3. Cyclomatic Complexity

Cylomatic Complexity is calculated from the flowchart analysis in the first stage

V(G)= (E-N) +2

V(G)= Number of Regions

E= Number of edges determined by arrow images

N= Number of graph nodes (nodes) with circle images

VG = (8-7) + 2

VG = 3

3.2.4. Independent path

From the Flowchart Analysis, the Independent path is determined. The independent path is obtained as follows:

Path 1 = 1,2,3,4,5,6,7

Path 2 = 1,2,3,7

3.2.5. Test Case Design

After determining the independent path from the flow graph above, the next step is to create a test case of at least the number of independent paths that have been created, ensuring that every possible path that will be passed has a test case created.

Table 7. Test cases from the created independent path

Path	1
Track	1,2,3,4,5,6,7
Scenario	1. Start 2. Dasboard User 3. Is there any any new notation?, if no 4. Check Notification Message 5. display Notification Message 6. Is it back to dashboard? If no 7. Stop
Test Results	Succeed
Path	2
Track	1,2,3,7
Scenario	1. Start 2. Dasboard User 3. Is there any any new notation?, if yes 7. Stop
Test Results	Succeed

4. CONCLUSION

Based on the results of white box testing with a path testing approach on the web-based DUKCAPIL service submission status notification module in Kuningan Barat Village, it can be concluded that all logical paths contained in the notification module were successfully identified and tested using the control flow graph method and cyclomatic complexity. The notification module is able to send service submission status information to users accurately and according to the designed program flow. For future system development and improvement, the following are recommended: Implementation of Advanced Testing: In addition to white box testing, black box testing and user acceptance test (UAT) need to be carried out to ensure that the system also functions according to end user expectations. Automated Testing: The use of automated testing tools is recommended to speed up the testing process for future system updates. Expansion of the Notification Module can be further developed with personalization features and integration into other communication media such as SMS or WhatsApp to reach more users.

ACKNOWLEDGEMENTS
"The author expresses his deepest appreciation and gratitude to the Institute for Research and Community Service (LPPM) of Pembangunan Jaya University for all forms of support, both moral and material, which have made possible the publication of this scientific paper."

REFERENCE

[1] Roger S.Pressman ,"Software Engineering: A Practitioner's Approach." [Online]. Available: www.mhhe.com/pressman. MC Graw Hill, Higher Education.

[2] D. I. Pirdaus and R. A. Hidayana, "Analysis Testing Black Box and White Box on Application To-Do List Based Web," *International Journal of Mathematics, Statistics, and Computing*, vol. 2, no. 2, pp. 68–75, 2024.

[3] *Katlon*. (2025.). Katlon,2025, White Box Testing: All You Need To Know, https://katalon.com/resources-center/blog/what-is-white-box-testing

[4] Ian. Sommerville, *Software engineering*. Pearson, 2011. Pearson Education Limited Edinburgh Gate ISBN 10: 1-292-09613-6 ISBN 13: 978-1-292-09613-1

[5] S. Bardin, O. Chebaro, M. Delahaye, and N. Kosmatov, "An all-in-one toolkit for automated white-box testing," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2014, pp. 53–60. doi: 10.1007/978-3-319-09099-3_4.

[6] P. C. Jorgensen, "Software Testing Fourth Edition A Craftsman's Approach.",2018, https://doi.org/10.1201/b15980

[7] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge University Press, 2016. doi: 10.1017/9781316771273.

[8] C. Kaner, "Measurement Issues & Software Testing Measurement Issues and Software Testing 1," 2001. [Online]. Available: www.kaner.com

[9] E. Viglianisi, M. Dallago, and M. Ceccato, "RESTTESTGEN: Automated Black-Box Testing of RESTful APIs," in *Proceedings - 2020 IEEE 13th International Conference on Software Testing, Verification and Validation, ICST 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 142–152. doi: 10.1109/ICST46399.2020.00024.

[10] M. Ghibran and A. L. Khamaeni, "Implementasi White Box Testing Berbasis Path Pada Aplikasi Berbasis Web," *Jurnal Siliwangi*, vol. 9, no. 1, p. 2023.

[11] M. Kumar, A. Professor, S. Kumar Singh, R. K. Dwivedi, and A. Professor, "International Journal of Advance Research in Computer Science and Management Studies," *International Journal of Advance Research in Computer Science and Management Studies*, vol. 3, no. 10, 2015, [Online]. Available: www.ijarcsms.com

[12] H. Gusdevi *et al.*, "Pengujian White-Box Pada Aplikasi Debt Manager Berbasis Android. NARATIF (Jurnal Ilmiah Nasional Riset Aplikasi dan Teknik Informatika) Vol. 04 No. 01 Juni 2022,P-ISSN: 2656-7377 || E-ISSN: 2714-846

[13] A. Dimas Saputro and A. Azzahra Narwastika, "Implementasi White Box Testing Dengan Teknik Basis Path Pada Pengujian Form Peminjaman Sistem Aplikasi Perpustakaan." Prosiding Seminar Nasional Teknologi Informasi dan Bisnis (SENATIB) 2023, e-ISSN 2962-1968

[14] M. A. Nurwicaksono, I. N. Lisa, A. R. Tiara, and R. Sidik, "Optimasi Sistem Informasi Konsultasi Hukum melalui Pendekatan Pengujian Kombinasi White-box dan Black-box," *Jurnal Manajemen Informatika (JAMIKA)*, vol. 14, no. 1, pp. 1–15, Nov. 2023, doi: 10.34010/jamika.v14i1.10110.

[15] S. Y. Rini and A. Kusmaya Putri, "Implementasi Blackbox Testing Dan Whitebox Testing Pada Pengujian Form Profil Toko Admin Sistem Aplikasi Raja Ongkir Berbasis Website." Prosiding Seminar Nasional Teknologi Informasi dan Bisnis (SENATIB) 2023, e-ISSN 2962-1968

[16] Enrique. Peñalosa, Fernando. Votero, Paloma. Balencia, and Roi. Barreras, *Licuadoras luminosas*. La Jaula Publicasiones, 2019.

[17] M. M. Syaikhuddin, C. Anam, A. R. Rinaldi, and M. Conoras, "Conventional Software Testing Using White Box Method. Kinetik : Game Technology, Information System," *Computer Network, Computing, Electronics, and Control*, vol. 3, no. 1, pp. 67–74, 201

[18] Mega & Safrizal, 2025, Implementasi Pengujian White Box Menggunakan Path Testing untuk Meningkatkan Keandalan Sistem Presensi Berbasis Web di PT XYZ. In *JITUS: Journal Information Technology for Urban Society* (Vol. 1). Maret.https://ojs.upj.ac.id/index.php/JITUS

[19] Rahman Abdillah, Rudi Hermawan, Wawan Hermawansyah, Ibnu Adkha, and Heri Arifin, "Pengujian Perangkat Lunak Sistem Informasi Inventori pada Usaha Jasa Pengiriman Paket," *Polygon : Jurnal Ilmu Komputer dan Ilmu Pengetahuan Alam*, vol. 2, no. 4, pp. 166–175, Jul. 2024, doi: 10.62383/polygon.v2i4.199.

[20] Safrizal et al,2024,Testing Dan Implementasi, PT Mafy Media Literasi Indonesia , https://www.academia.edu/119966136/Buku_TESTING_DAN_IMPLEMENTASI_2024, 978-623-8638-29-1.

[21] N. Golian, V. Golian, and I. Afanasieva, "Black And White-Box Unit Testing For Web Applications," *Bulletin of National Technical University "KhPI". Series: System Analysis, Control and Information Technologies*, no. 1 (7), pp. 79–83, Jul. 2022, doi: 10.20998/2079-0023.2022.01.13.

[22] S. Izzat and N. N. Saleem, "Software Testing Techniques and Tools: A Review," *Journal of Education and Science*, vol. 32, no. 2, pp. 31–40, Jun. 2023, doi: 10.33899/edusj.2023.137480.1305.

[23] R. Andrian Ibrahim and G. Saktian Laksito, "Optimization of White Box Testing by Utilizing Branching and Repeating Structures in Java Programs Using Base Path," *International Journal of Mathematics, Statistics, and Computing*, vol. 2, no. 2, pp. 85–89, 2024.

[24] A. Sekhon, Y. Ji, M. B. Dwyer, and Y. Qi, "White-box Testing of NLP models with Mask Neuron Coverage," May 2022, [Online]. Available: http://arxiv.org/abs/2205.05050

[25] P. M. Chawan, W. : Www, S. Thakare, S. Chavan, and M. Chawan, "Software Testing Strategies and Techniques International Journal of Emerging Technology and Advanced Engineering Software Testing Strategies and Techniques," 2012. [Online]. Available: www.ijetae.com

[26] B. Žlahtič, J. Završnik, H. Blažun Vošner, and P. Kokol, "Transferring Black-Box Decision Making to a White-Box Model," *Electronics (Switzerland)*, vol. 13, no. 10, May 2024, doi: 10.3390/electronics13101895.

[27] T. Sheakh, T. Hussain, and S. Singh, "International Journal of Allied Practice, Research and Review A Comparative Study of Software Testing Techniques Viz. White Box Testing Black Box Testing and Grey Box Testing," *IJAPRR International Peer Reviewed Refereed Journal*, vol. II, pp. 1–08, 2015, [Online]. Available: https://www.researchgate.net/publication/276028491

[28] F. Lammermann and S. Wappler, "Benefits of Software Measures for Evolutionary White-Box Testing." [Online]. Available: https://www.researchgate.net/publication/327691019

[29] A. Caniço and A. Santos, "Witter: A Library for White-Box Testing of Introductory Programming Algorithms," in *SPLASH-E 2023 - Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E, Co-located with: SPLASH 2023*, Association for Computing Machinery, Inc, Oct. 2023, pp. 69–74. doi: 10.1145/3622780.3623650.

[30] A. Suryanta, L. Wahyu Widianti, dan Mirsya Ashari, P. Kodiklatad, and S. Jakarta STI, "Pengujian White Box Terhadap Sistem Informasi Akademik (Siakad) Di Politeknik Angkatan Darat," JATI (Jurnal Mahasiswa Teknik Informatika),Vol. 9 No. 2, April 2025.

[31] G. W. Sasmito, "White Box Testing with Basis Path Technique in the Demography Administration Website," in *ICSECC 2020 - 2nd International Conference on Sustainable Engineering and Creative Computing, Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2020, pp. 86–92. doi: 10.1109/ICSECC51444.2020.9557428.

[32] A. H. Ali and N. N. Saleem, "Data Flow Testing and Tools Review," *Journal of Education and Science*, vol. 32, no. 2, pp. 51–60, Jun. 2023, doi: 10.33899/edusj.2023.137611.1315.

[33] D. Madhavi, "A White Box Testing Technique in Software Testing: Basis Path Testing", [Online]. Available: www.journalforresearch.org