The Indonesian Journal of Computer Science
www.ijcs.net
Volume 13, Issue 5, October 2024
https://doi.org/10.33022/ijcs.v13i5.4351

Enhancing the Functionality of Management Internship and Community Service Through Maintenance of Web Application

Ahmad Faris Hisyam Maulana[1], Tien Fabrianti Kusumasari [2], Ekky Novriza Alam[3]
farishisyam@student.telkomuniversity.ac.id[1], tienkusumasari@telkomuniversity.ac.id[2], ekkynovrizalam@telkomuniversity.ac.id[3]
[1,2,3] Information System Department, School of Industrial and System Engineering, Telkom University

Article Information

Submitted : 17 Aug 2024
Reviewed: 30 Aug 2024
Accepted : 30 Sep 2024

Keywords

Software Maintenance, Iterative Incremental, Requirements Prioritization, Testing.

Abstract

Universities play a crucial role in developing high-quality human resources through Internship and Community Service Programs, which prepare students with professional skills and relevant competencies. However, implementing these programs often faces issues such as inefficiencies in registration, documentation, and progress reporting. To address these challenges, a university department developed and implemented a comprehensive web-based application system. This research aims to improve and refine the application through software maintenance processes, employing an iterative incremental method with phases including planning, requirements, analysis and design, implementation, testing, and evaluation, conducted repeatedly according to the prioritized order of feature improvements. After being maintained, the application is tested through functional testing using black box techniques and automation tools. Results indicate that the maintenance process successfully ensured the application functions according to established scenarios and requirements, thereby enhancing support for administrative processes of internships and community service programs and overcomes previous management challenges.

A. Introduction

Universities play a crucial role in developing human resources and enhancing national competitiveness [1]. Universities strive to produce quality graduates by enhancing learning processes and implementing internship programs [2]. These programs are designed to develop both soft and hard skills, ensuring graduates can perform well in professional environments.

However, several issues are commonly encountered during the execution of internship programs. Inefficient registration and documentation processes often lead to delays and data management errors [3]. Manual and unscheduled monitoring results in progress reporting delays and data inaccuracies [4]. Utilizing websites for internship management can improve information dissemination, registration, and progress monitoring, ensuring data accuracy and accessibility [5].

To address these issues, a university department developed and implemented a comprehensive web-based application system to organize relevant information and data from students, academic mentors, and field mentors involved in the programs. As a new system, it contains initial development bugs and imperfections. Therefore, software maintenance is essential to prevent potential issues and ensure efficient software operation [6]. Effective software maintenance helps in identifying and analyzing opportunities for functionality improvement. It pinpoints components that hinder functionality and enhances the overall system's capabilities [7]. Types of software maintenance include corrective, adaptive, perfective, and preventive maintenance [8].

In the context of web-based applications, which involve client-side and server-side components [9], a structured approach to development and maintenance is required. One such approach is the Model-View-Controller (MVC) architecture pattern, which separates the application into three interconnected components: the model (data logic), the view (user interface), and the controller (business logic) [10]. This separation ensures that data handling, user interaction, and business processes are managed efficiently [11]. Ensuring that all components function correctly and efficiently through regular maintenance is crucial.

Additionally, effective software maintenance also involves prioritizing software requirements. Prioritization helps ensure that the most critical features are addressed first and enhances customer satisfaction [12]. This process varies by organizational perspective and is influenced by factors such as importance, time, risk, cost, value, penalty, and precedence [13]. By prioritizing requirements, development teams can better allocate resources and improve project outcomes, ultimately supporting the ongoing maintenance and enhancement of the system [14].

The purpose of this research is to enhance the functionality of a web-based application for managing internships and community service programs through software maintenance as outlined in the Develop/Build of the Information System Research Framework, as depicted in Figure 1. The expected outcomes include more efficient administrative processes and enhanced service quality in these programs, aligning with the environment's specific needs.

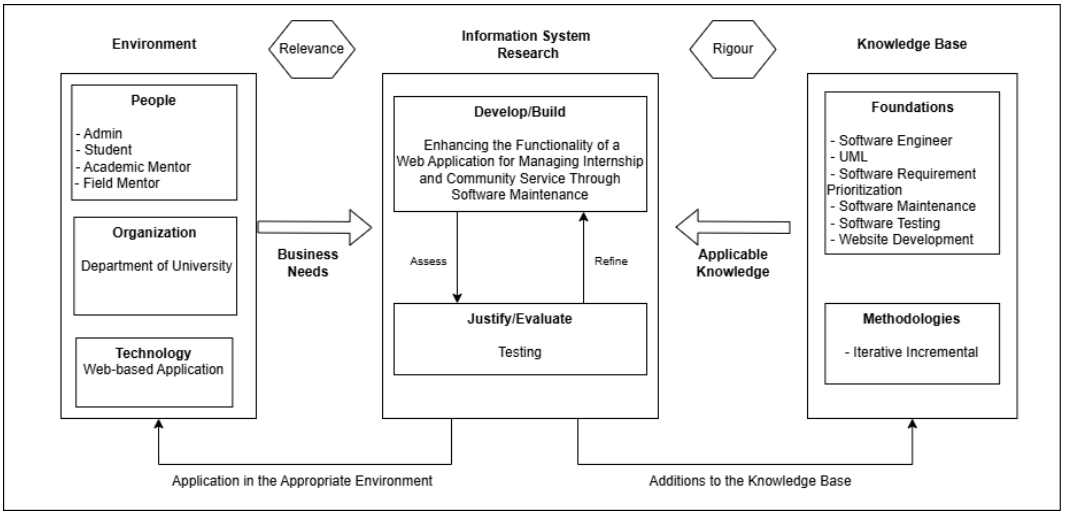

Figure 1. Information System Research Framework [15]

Furthermore, the contribution of this research lies in its detailed approach to maintaining a web-based application for internship and community service management. By leveraging Applicable Knowledge, as depicted in Figure 1, the research employs an iterative incremental method for continuous improvement, prioritizes requirements to address key issues, and integrates various maintenance strategies, such as corrective, adaptive, perfective, and preventive. Hopefully, this study will not only enhance the existing system but also serve as a valuable reference for other institutions looking to improve similar systems while contributing to both practical and theoretical knowledge in the field of web-based application maintenance.

B. Research Method

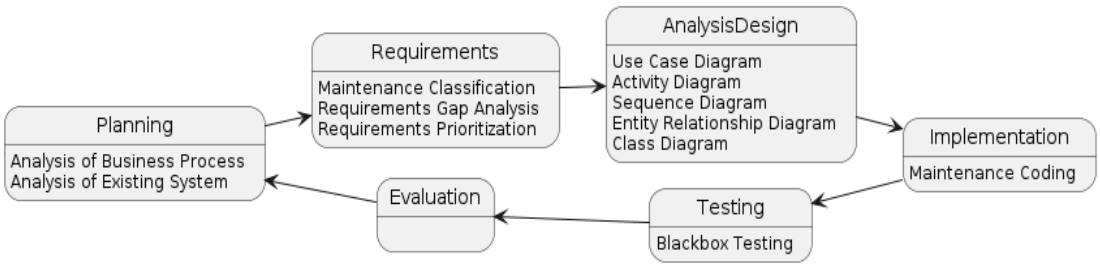

Figure 2. Iterative Incremental

This research employs the Iterative Incremental model, a software development model that builds the product gradually through repeated cycles of planning, requirements, analysis & design, implementation, testing, and evaluation. This approach ensures that each iteration extends the product's functionality by adding new features and refining existing ones. By constructing the product incrementally, the development process allows for early problem identification, quick corrections, and visible results throughout the development phases [16]. This method offers adaptability for handling requirements change with continuous communication for success [17].

The research begins with the planning phase. This involves analyzing the business processes and the existing system. Analyzing business processes is needed to understand the workflow and interactions within the organization. Analyzing the existing system involves documenting issues that need to be fixed or enhanced. This analysis is necessary for defining the elements that require maintenance.

Following the planning phase, the requirements phase involves gathering detailed user and system requirements. This phase includes maintenance classification, requirements gap analysis, and requirements prioritization. The results from the analysis existing system and issues documentation are used to determine the maintenance classification for each identified issue. Subsequently, the details from the maintenance classification are prioritized for improvement or enhancement work using requirements prioritization techniques.

The maintenance classification involves categorizing the identified issues into different types of maintenance. This includes corrective maintenance to fix faults after the software is operational, adaptive maintenance to keep the software usable in a changing environment, perfective maintenance to improve performance or maintainability, and preventive maintenance to detect and correct latent faults before they occur [7]. The following bar chart in Figure 3 illustrates the classification of maintenance tasks by different roles involved in the system.

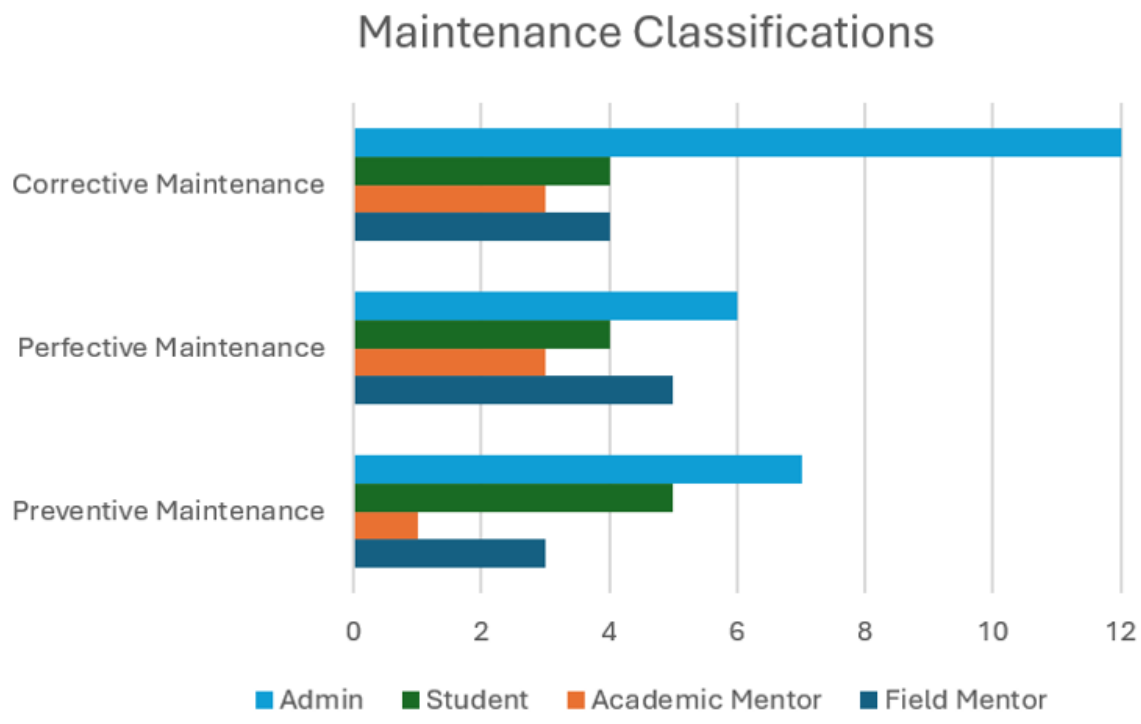

Figure 3. Maintenance Classifications

Factors for software requirement prioritization that serve as a basis for determining priorities have given rise to techniques used in prioritizing needs from multiple perspectives. One such technique is Numerical Assignment, which divides priority needs into a numerical scale. An example is using a scale of 1 to 5 to represent the significance of a requirement [11]. The scatter plot in Figure 4 shows the priority scores assigned to various maintenance codes. The assigned priority scores indicate that 1 means top priority, and 5 signifies lowest priority. The higher the priority, the sooner the issue will be addressed.

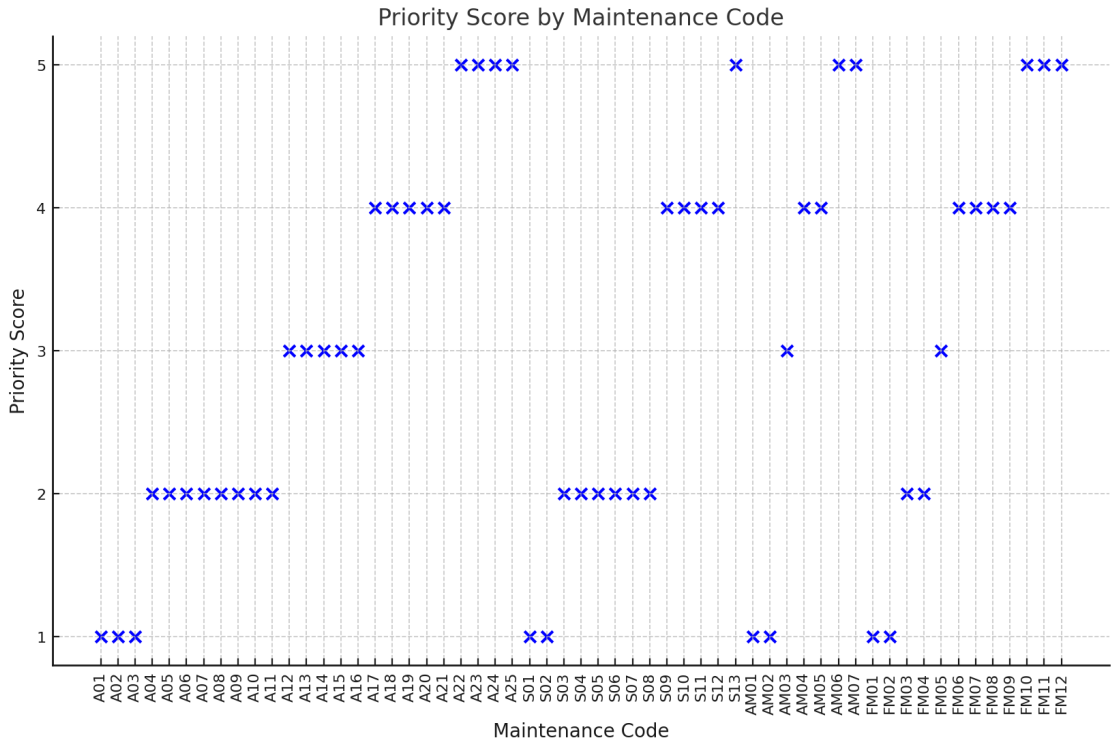

Figure 4. Requirement Priority Scores

In the analysis and design phase involves identifying system components and designing the system to be implemented. System diagrams are created using UML (Unified Modeling Language) to visualize the system. Use case diagrams illustrate the primary functions of a system and show how external users interact with these internal functions [18]. Use case diagrams assist in illustrating how users interact with a system's features to accomplish their objectives and clarify the distinctions between various system behaviors and the perspectives of different stakeholders [19]. The following use case diagram in Figure 5 provides an overview of the interactions within the web application system, highlighting the roles of different users such as Admin, Student, Academic Mentor, and Field Mentor.

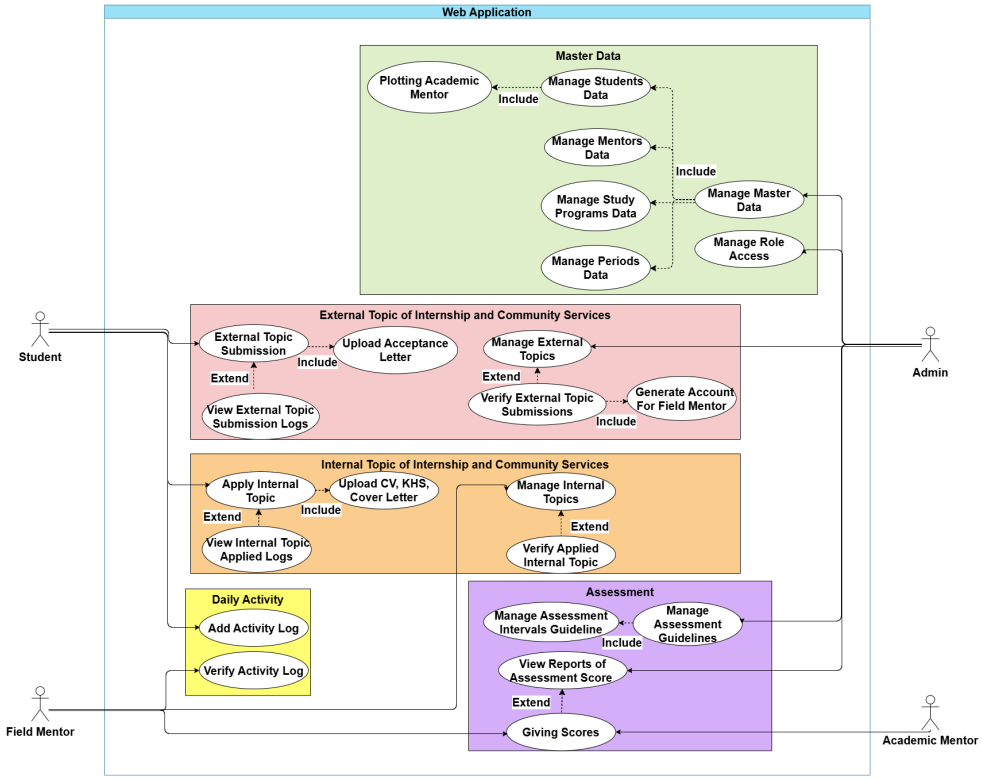

Figure 5. Use Case Diagram

During the Implementation stage, maintenance was carried out for each type of task according to the assigned priority score order, while the Testing stage ensured that the developed system met the requirements through black box testing, which verified the system's functionality.

C. Result and Discussion

The web application was initially built using the Laravel framework, which follows the MVC pattern to separate business logic from presentation [20]. The maintained function of the MVC architecture is depicted in Figure 6, highlighting the components that have undergone maintenance. The View, Controller, and Model sections represent the list of key elements that have been maintained to ensure the system's proper functionality and efficiency.

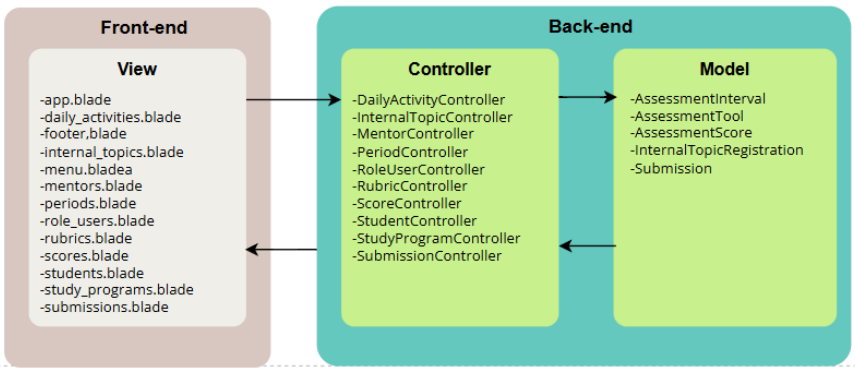

Figure 6. Maintained MVC

Additionally, In Figure 7, the yellow-highlighted sections within the database indicate the areas that were corrected. These corrections involved adding attributes and changing data types within the database to ensure more accurate data management.

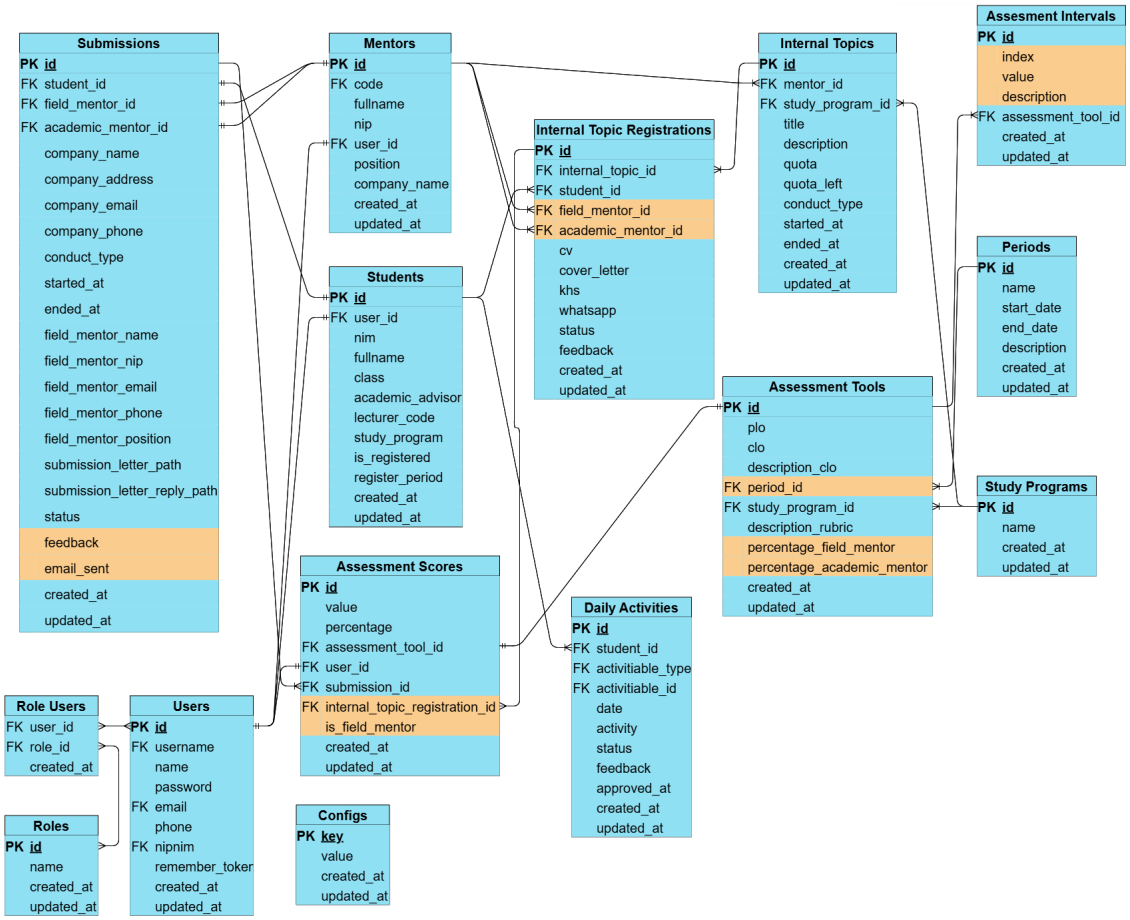

Figure 7. Maintained Database Structure

To provide a clearer understanding of the maintenance activities performed, Table 1 presents a detailed breakdown of each task during the research. This table lists the specific maintenance codes and their corresponding descriptions, offering insight into how each implementation of maintenance contributed to the overall improvement functionality of the system.

Table 1. Detail of Each Maintenance

No	Maintenance Code	Description
1	A01	Fixes access to the assessment guideline
2	A02	Fixes access to the assessment interval guideline
3	A03	Fixes access to the assessment score
4	A04, A05, A11	Prevents schedule confusion due to date input errors
5	A06, A09	Prevents incomplete data input for create assessments
6	A07, A10	Prevents incomplete data input for create intervals
7	A08	Prevents incomplete data input for create study programs
8	A12	Prevents student data addition errors
9	A13	Fixes inability to view mentor assessment details

10	A14, A15	Fixes inaccuracies in student assessment unweighted scores
11	A16	Fixes errors in updating the sent email column
12	A17	Enhances access to assessment data
13	A18	Enhances access to external topic data
14	A19, S10, FM06	Prevents user misunderstanding of date interpretation
15	A20, S11, FM08	Reduces navigation confusion for performing operations
16	A21, S12, AM05, FM09	Reduces navigation confusion for menu selection
17	A22	Prevents user misunderstanding of text interpretation
18	A23	Prevents navigation confusion after menu selection
19	A24. AM06, FM11	Prevents user discomfort when accessing pages
20	A25, S13, AM07, FM12	Prevents user confusion about system identity
21	S01, AM02, FM02	Prevents unauthorized access to sensitive data
22	S02	Prevents sensitive data leaks
23	S03	Fixes the ability for students to apply for internal topics
24	S04	Prevents incomplete data input
25	S05, S06	Prevents file uploads that do not match the format
26	S07	Prevents activity logging date errors
27	S08	Prevents invalid external topic date periods
28	S09	Enhances access to activity logs
29	AM01	Fixes access to the assessment score
30	AM03	Fixes inaccuracies in student assessment values
31	AM04	Enhances access to assessment data
32	FM01	Fixes access to the assessment score
33	FM03, FM04	Prevents invalid data input for date
34	FM05	Fixes inaccuracies in student assessment scores
35	FM07	Enhances access to assessment data
36	FM10	Prevents user misunderstanding of text interpretation

It is essential to outline the specific application features impacted by the maintenance activities to provide a more comprehensive view of the system enhancements. After describing each maintenance task, it is important to highlight the features that were directly affected by these tasks. This approach helps illustrate how the targeted maintenance efforts have contributed to improving the functionality of the system, ensuring it effectively meets user needs. Table 2 summarizes the features impacted by each maintenance activity.

Table 2. Features Impacted

No	Maintenance Code	Features Impacted
1	A01	View Assessment Guidelines
2	A02	View Assessment Interval Guidelines
3	A03	View Assessment Score
4	A04, A05, A11	Create Period Data
5	A06, A09	Create Assessment Guidelines
6	A07, A10	Create Assessment Interval Guidelines
7	A08	Create Study Program Data
8	A12	Create Student Data
9	A13	View Field Mentor Assessment Details
10	A14, A15	View Assessment Reports
11	A16	View External Topic Details
12	A17	View Assessment Report Data
13	A18	View External Internship Data
14	A19, S10, FM06	Date Display in Table
15	A20, S11, FM08	Show, Edit, and Delete Actions in Table
16	A21, S12, AM05, FM09	View Sidebar Menu
17	A22	Rubrics Display

18	A23	Sidebar Display
19	A24. AM06, FM11	Footer Display
20	A25, S13, AM07, FM12	Title Name
21	S01, AM02, FM02	Master Data
22	S02	Export External Internship Data
23	S03	Apply Internal Topic
24	S04	Internal Topic Submission
25	S05, S06	Internal and External Topic Submission
26	S07	Create Activity Log
27	S08	Create External Topic Data
28	S09	View Activity Logs
29	AM01	Academic Mentor Assessment Score
30	AM03	Calculation of Academic Mentor Assessment
31	AM04	View Academic Mentor Assessment
32	FM01	Field Mentor Assessment Score
33	FM03, FM04	Create Internal Topic
34	FM05	Calculation of Field Mentor Assessment
35	FM07	View Field Mentor Assessment
36	FM10	Display of Internal Topic Submission Details

After implementing the maintenance, the application undergoes testing to ensure it meets all requirements and achieves high-quality assurance [21]. Among various testing techniques, black box testing focuses on the functional specifications of software programs [22], which is a valuable approach for improving software quality and user experience [23]. To support this, Laravel Dusk as an automated testing tool, is used to enhance quality, reliability, and performance by optimizing time efficiency, boosting accuracy, and allowing for repeatable testing [24]. Table 3 summarizes the testing results for each maintenance case with the maintenance code related to the case.

Table 3. Testing

No	Case	Maintenance Code Related	Status
1	Manage Students Data	A12, A20, A21, A25	Pass
2	Manage Mentors Data	A20, A21, A25	Pass
3	Manage Study Programs Data	A08, A12, A20, A21, A25	Pass
4	Manage Periods Data	A04, A05, A11, A19, A20, A21, A23, A25	Pass
5	Manage Role Access	A19, A20, A21, A25	Pass
6	Manage External Topics	A18, A19, A20, A21, A25	Pass
7	Verify External Topic Submissions	A18, A19, A20, A21, A25	Pass
8	Manage Assessment Guidelines	A01, A06, A09, A20, A21, A22, A24, A25	Pass
9	Manage Assessment Intervals Guideline	A01, A02, A07, A10, A20, A21, A24, A25	Pass
10	View Reports of Assessment Score	A03, A13, A14, A15, A17, A21, A24, A25	Pass
11	External Topic Submission	S05, S06, S08, S11, S12, S13	Pass
12	View External Topic Submission Logs	S02, S10, S11, S12, S13	Pass
13	Apply Internal Topic	S03, S04, S05, S06, S10, S11, S12, S13	Pass
14	View Internal Topic Applied Logs	S10, S11, S12, S13	Pass
15	Add Activity Log	S07, S09, S10, S11, S12, S13	Pass
16	Manage Internal Topics	FM03, FM04, FM06, FM08, FM09, FM12	Pass
17	Verify Applied Internal Topic	FM06, FM08, FM09, FM10, FM12	Pass
18	Verify Activity Log	FM06, FM08, FM09, FM12	Pass
19	Giving Scores (Field Mentor)	FM01, FM05, FM07, FM09, FM11, FM12	Pass
20	Giving Scores (Academic Mentor)	AM01, AM03, AM04, AM05, AM06, AM07	Pass

The testing of the application revealed that all 20 test cases passed successfully, confirming that the maintained application performed well according to the established scenarios and requirements. Each test case validated key aspects such as data management and every activity that related to the actors involved, ensuring optimal system functionality. Laravel Dusk facilitated comprehensive automated testing, which verified that the system meets all criteria and maintains high standards of quality. These results demonstrate the effectiveness of the maintenance strategies and affirm the system's readiness for continued use.

D. Conclusion

This research successfully enhanced the functionality of a web-based application for managing internship and community service programs. Through comprehensive software maintenance, key issues such as inefficiencies in registration, documentation, and progress reporting were addressed. These enhancements streamlined administrative tasks, improving the efficiency and effectiveness of managing internships and community service activities.

Software maintenance was carried out for each type of task according to the assigned priority score order. These efforts addressed issues for key system actors, including Admin, Student, Academic Mentor, and Field Mentor. Maintenance targeted the application's components built using the Laravel MVC framework, with specific modifications made to the View, Controller, and Model elements, as well as adjustments to the underlying database structure to optimize system functionality and ensure more accurate data management. Corrective Maintenance resolved 12 issues affecting Admin, 4 issues for Students, 3 for Academic Mentors, and 4 for Field Mentors. Perfective Maintenance addressed 6 issues for Admin, 4 for Students, 3 for Academic Mentors, and 5 for Field Mentors. Preventive Maintenance handled 7 issues for Admin, 5 for Students, 1 for Academic Mentors, and 3 for Field Mentors.

To validate the effectiveness of these maintenance activities, rigorous black box testing was conducted using the Laravel Dusk automation tool. The testing validated 20 cases of testing functionality in the system, confirming that the application met the established scenarios and requirements from the user's perspective, thereby demonstrating the success of the maintenance efforts.

E. References

[1] M. Arifin, “Strategi Manajemen Perubahan dalam Meningkatkan Disiplin di Perguruan Tinggi,” *EduTech: Jurnal Ilmu Pendidikan dan Ilmu Sosial*, vol. 3, no. 1, 2017.

[2] C. Suharyanti, “Pengaruh Proses Pembelajaran dan Program Kerja Praktek Terhadap Pengembangan Soft Skills Mahasiswa,” *Jurnal Pendidikan Administrasi Perkantoran Universitas Sebelas Maret*, vol. 4, no. 1, p. 118291, 2015.

[3] A. K. Rasyid, N. P. Dewi, B. Said, and U. Ubaidi, “Sistem Informasi Manajemen Kerja Praktek dan Tugas Akhir di Prodi Informatika Universitas Madura Berbasis Web,” *Insand Comtech: Information Science and Computer Technology Journal*, vol. 7, no. 2, 2023.

[4] N. Nurhasanah, S. D. Budiwati, and T. D. Tambunan, "Aplikasi Monitoring Peserta Kerja Praktek Berbasis Web di PT. Industri Telekomunikasi Indonesia (PERSERO)," *eProceedings of Applied Science*, vol. 3, no. 2, 2017.

[5] I. Ruslianto and S. Al Rasyid, "Aplikasi Monitoring Kerja Praktik Mahasiswa Program Studi Sistem Komputer," *SEMIRATA 2015*, vol. 5, no. 1.

[6] Mohammed AL Mashhadani and Dr. Prof. Ali Yazici, "Software Maintenance Process Towards Cloud Environment: A Review Study," *International Journal For Multidisciplinary Research*, vol. 4, no. 6, Dec. 2022, doi: 10.36948/ijfmr.2022.v04i06.1184.

[7] C.-P. Bezemer and A. Zaidman, "Performance Optimization of Deployed Software-as-a-Service Applications," *Journal of Systems and Software*, vol. 87, pp. 87–103, 2014, doi: https://doi.org/10.1016/j.jss.2013.09.013.

[8] R. Pressman and B. Maxim, *Software Engineering: A Practitioner's Approach 9th Edition*. 2019.

[9] N. R. Dissanayake and G. Dias, "Web-Based Applications: Extending the General Perspective of the Service of Web," 10th International Research Conference of KDU (KDU-IRC 2017), 2017.

[10] I. H. Sarker and K. Apu, "Mvc architecture driven design and implementation of java framework for developing desktop application," *International Journal of Hybrid Information Technology*, vol. 7, no. 5, pp. 317–322, 2014.

[11] P. Ouyang, W. Cao, M. Wu, C. Gan, and F. Wang, "Design of Intelligent Drilling System Software Framework and Data Architecture Based on MVC Pattern," in *2019 Chinese Control Conference (CCC)*, IEEE, 2019, pp. 7075–7078.

[12] A. Hudaib, R. Masadeh, M. H. Qasem, and A. Alzaqebah, "Requirements Prioritization Techniques Comparison," *Mod Appl Sci*, vol. 12, no. 2, p. 62, Jan. 2018, doi: 10.5539/mas.v12n2p62.

[13] I. Olaronke, I. Rhoda, and G. Ishaya, "An Appraisal of Software Requirement Prioritization Techniques," *Asian Journal of Research in Computer Science*, pp. 1–16, Apr. 2018, doi: 10.9734/ajrcos/2018/v1i124717.

[14] P. Berander and A. Andrews, "Requirements Prioritization," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 69–94. doi: 10.1007/3-540-28244-0_4.

[15] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS quarterly*, pp. 75–105, 2004.

[16] A. P. Izzulhaq, R. Fauzi, S. Suakanto, A. K. H. Disina, H. Mahdin, and I. A. Salihu, "Development of User Management in Ihya Digital Ecosystem Using Iterative Incremental Method," in *2022 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS)*, 2022, pp. 1–6. doi: 10.1109/ICADEIS56544.2022.10037391.

[17] S.-T. Lai, "A Maintainability Enhancement Procedure for Reducing Agile Software Development Risk," *International Journal of Software Engineering & Applications*, vol. 6, no. 4, pp. 29–40, Jul. 2015, doi: 10.5121/ijsea.2015.6403.

[18] E. R. Aquino, P. de Saqui-Sannes, and R. A. Vingerhoeds, "A Methodological Assistant for UML and SysML Use Case Diagrams," 2021, pp. 298–322. doi: 10.1007/978-3-030-67445-8_13.

[19] O. Kautz, B. Rumpe, and L. Wachtmeister, "Semantic Differencing of Use Case Diagrams.," *The Journal of Object Technology*, vol. 21, no. 3, p. 3:1, 2022, doi: 10.5381/jot.2022.21.3.a5.

[20] D. Jain, *Ultimate Laravel for Modern Web Development: Build Robust and Interactive Enterprise-Grade Web Apps using Laravel's MVC, Authentication, APIs, and Cloud Deployment (English Edition)*. Orange Education Pvt Limited, 2024.

[21] S. Joshi and I. Kumari, "Analyses of Software Testing Approaches," in *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, IEEE, Nov. 2022, pp. 1276–1281. doi: 10.1109/IIHC55949.2022.10060147.

[22] T. Hidayat and M. Muttaqin, "Pengujian Sistem Informasi Pendaftaran dan Pembayaran Wisuda Online menggunakan Black Box Testing dengan Metode Equivalence Partitioning dan Boundary Value Analysis," 2018. [Online]. Available: www.ccssenet.org/cis

[23] F. C. Ningrum, D. Suherman, S. Aryanti, H. A. Prasetya, and A. Saifudin, "Pengujian Black Box pada Aplikasi Sistem Seleksi Sales Terbaik Menggunakan Teknik Equivalence Partitions," *Jurnal Informatika Universitas Pamulang*, vol. 4, no. 4, p. 125, Dec. 2019, doi: 10.32493/informatika.v4i4.3782.

[24] S. K. Alferidah and S. Ahmed, "Automated Software Testing Tools," in *2020 International Conference on Computing and Information Technology (ICCIT-1441)*, IEEE, Sep. 2020, pp. 1–4. doi: 10.1109/ICCIT-144147971.2020.9213735.