Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Enhancing the Performance of Desk Evaluation: Final Project Web Application through Back-End Maintenance

Naufal Aqil Himawan[1], Tien Fabrianti Kusumasari[2], Nur Ichsan Utama[3]
naufalaqil@student.telkomuniversity.ac.id, tienkusumasari@telkomuniversity.ac.id, nichsan@telkomuniversity.ac.id
[1,2,3] Information System Department, School of Industrial and System Engineering, Telkom University

Article Information

Submitted : 25 Aug 2023
Reviewed: 28 Aug 2023
Accepted : 30 Aug 2023

Keywords

Back-end, Controller, Maintenance, Web Application, Testing

Abstract

Software maintenance resolves issues in development, including web applications. The web application can be split into front-end and back-end perspectives. Performance improvements can be achieved through back-end maintenance. Web applications with MVC (Model-View-Controller) architecture are maintained via code adjustment in the controllers. This research focuses on maintaining the back-end side of the Desk Evaluation, a web application for the university's final project management. The Collaboration Model of Software Development is the method used in this research. The controllers of the Desk Evaluation application that have been maintained are tested using unit and load testing. Unit testing shows that controllers give a proper response. Load testing indicates a 98% success rate and under 30 seconds average load time for all controllers. Then, the code in the controllers is analyzed with SonarQube, earning an A rating for reliability, signifying rule compliance, and minimizing bugs.

A. Introduction

Software Maintenance is one part of the software development cycle [1]. According to [2], Software Maintenance plays an important role in software development. Software Maintenance is a must to prevent problems that can arise so that if ignored, the business cannot grow optimally [3].

Software or applications can be built through the Software Engineering process and can be maintained through the Software Maintenance process. Software Maintenance is a process of Software Engineering that focuses on maintaining previously designed systems in the form of maintaining code writing structures, design maintenance, and maintenance or maintenance of specifications that are not suitable to meet the desired new requirements [4].

According to [1], [5], Software Maintenance is classified into corrective maintenance, adaptive maintenance, perfective maintenance, and preventive maintenance. Corrective Maintenance is the maintenance stage for problems that occur when a user has used the software. Adaptive Maintenance is the stage of maintaining software so that it can still be used when the application environment changes. Perfective Maintenance is a maintenance phase that contains the addition of new features or changes to the code structure to make it better. Preventive Maintenance is a maintenance phase that contains the prevention of errors that might occur in the software and is repaired before the user discovers it.

Software maintenance can also include maintenance of web-based applications. Web-based applications are systems with application components that work on the client side and communicate with application components on a web server to process data [6]. In their design, web-based applications have an architecture in them. This architecture is designed so that content, page design, and navigation on a website can be structured properly so that the goals of website development can be fulfilled [5].

One architecture that can be used to develop a web-based application is the MVC (Model-View-Controller) architecture. MVC is an architecture that divides the application into three interconnected parts. MVC separates the application's internal information from the information that will be displayed to application users. [7].

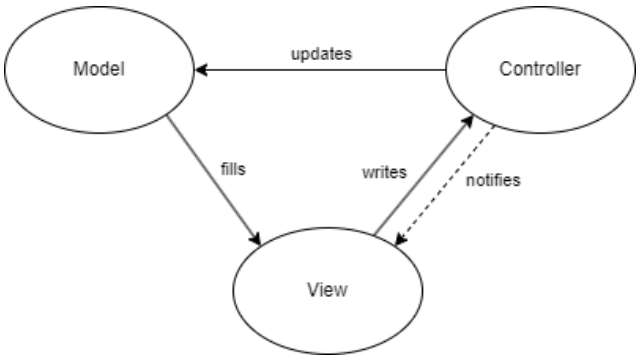

Figure 1. Illustration of Model-View-Controller [5]

The model is the part that manages web-based application data so that later it can be used on the View side. The model communicates directly with the database.

View is the part that manages all forms of user interface and can interact directly with users in the form of web pages. All data displayed in the View is data in the Model. The controller is the part that manages all requests made by the user. Requests that occur on the View side such as creating, reading, changing, and deleting data will be conveyed by the Controller to the model so that it becomes a bridge between the View and the Model [8]. MVC architecture illustration can be seen in Figure 1.

According to [9], the perspective of web-based applications is divided into two, namely front-end and back-end. Front-end is a component where users interact with the appearance of a web such as images, colors, and various display designs. Meanwhile, the back-end is a component that manages everything that is invisible to the user, such as the authentication process, database configuration, and management of various requests sent by the user.

This research will focus on the Desk Evaluation application maintenance process. Desk Evaluation is a web-based application to manage student final projects at a university. This application has various main features. From the student side, some of the features are the submission of final project topics, uploading of guidance reports, and uploading of final project reports. From the lecturer's point of view, some of its features are uploading final project topics, approving student topic submissions, and assessing student final project reports.

The Desk Evaluation application still has many shortcomings. For example, there were bugs in some existing features. In addition, there were requests from users of the Desk Evaluation application to add new features. Therefore, this study will discuss the maintenance process for the Desk Evaluation application and will focus on back-end maintenance.

The purpose of this research is to improve the performance of web-based applications through maintenance on the back-end side. According to [10], maintenance on the back-end side aims to manage logic on the server side and improve performance and responsiveness. Thus, this research is expected to contribute to improving the performance of web-based applications.

B. Research Method

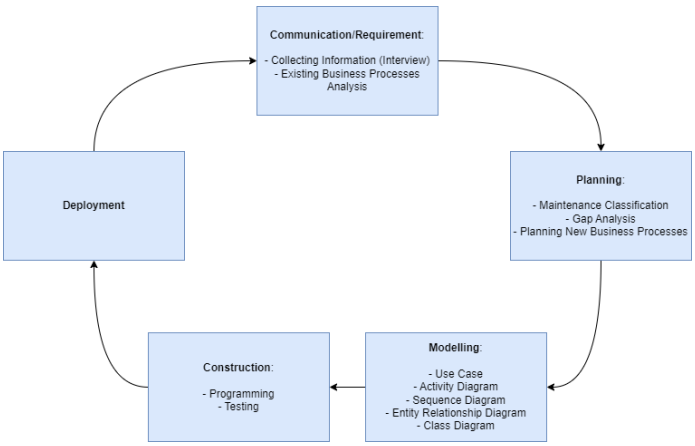

Figure 2. Collaboration Model

This research was conducted using the Collaboration Model of Software Development method. The Collaboration Model is one of several models that use an Agile approach. The phases in the model consist of five generic Software Engineering phases [5], namely communication, planning, modeling, construction, and deployment. Figure 2 shows an illustration of the Collaboration Model used in this study.

According to [11], this model emphasizes the importance of collaboration between customers and developers such as face-to-face communication between users, developers, and stakeholders as well as the collaboration tools used by them.

In the Communication/Requirement phase, information was collected regarding the Desk Evaluation application through face-to-face interviews with all users of the Desk Evaluation application, namely students, lecturers, and staff who act as admins in the Desk Evaluation application. The number of sources obtained were 10 students, 2 lecturers, and 1 admin. The number of sources depends on the completeness of the information provided by the sources [12]. This information will later become maintenance requirements.

In the Planning phase, the Desk Evaluation application requirements have been determined and classified, based on the classification of software maintenance [1], [5]. Figure 3. shows the distribution of determined and classified requirements in the form of a bar chart. As shown in Figure 3, the maintenance is divided into corrective maintenance and perfective maintenance.

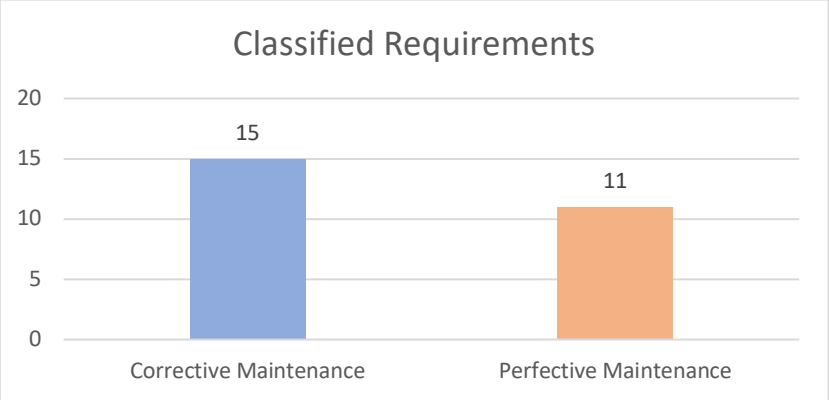

Figure 3. Classified Requirements

In the Modeling phase, UML (Unified Modeling Language) had been designed to help describe how the system works. UML is a standard language used to create software blueprints such as visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. One of them is use case diagrams. Use case diagrams are the center of modeling the behavior of a system, a sub-system, or a class [13]. Figure 4. shows a use case diagram for the Desk Evaluation application. The yellow use case diagram is a new use case that is proposed for maintenance in this study.

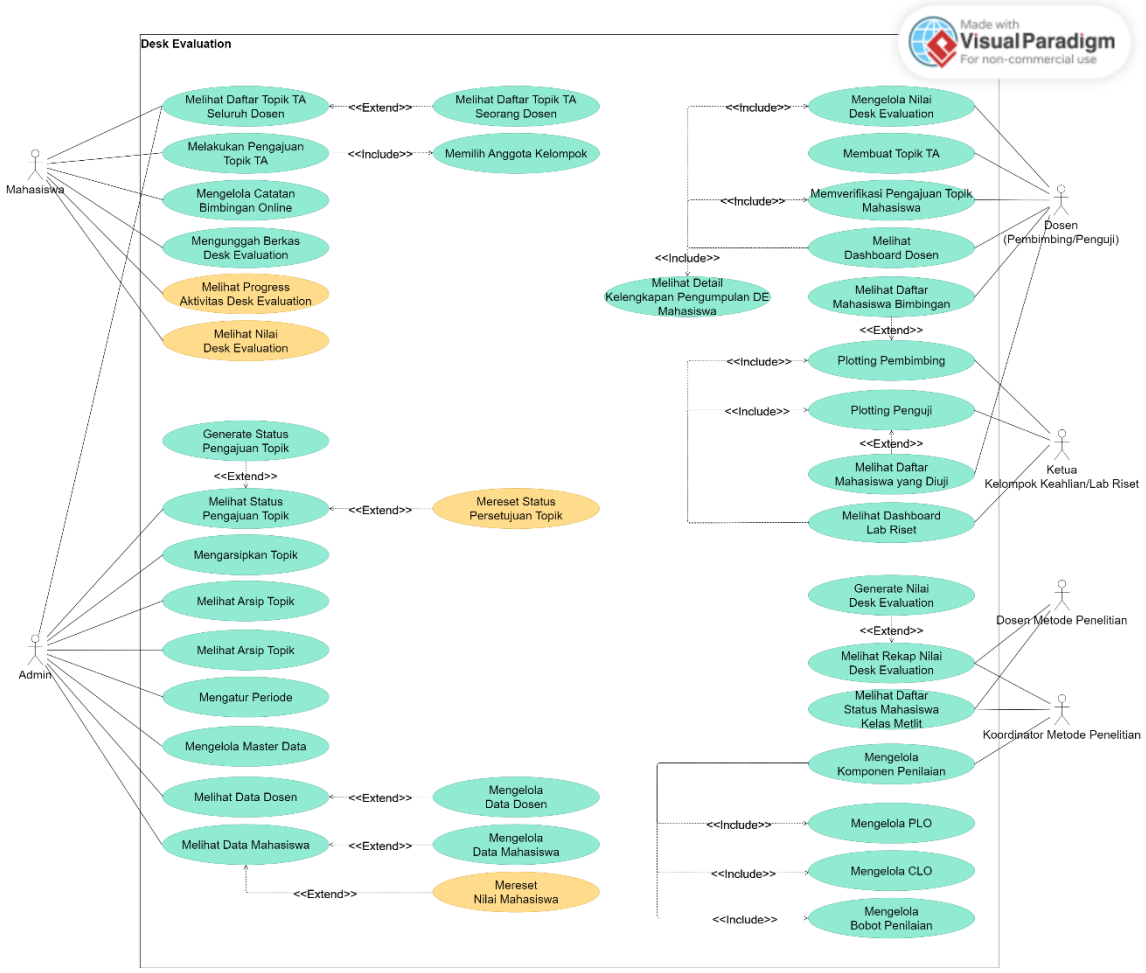

Figure 4. Use Case Diagram for Desk Evaluation

In the Construction phase, programming and testing had been carried out on the back-end side. Maintenance is done by creating and repairing code logic on the Desk Evaluation application controllers. After the code logic in the controllers had been maintained, the functionality and loading of the controllers were tested.

C. Result and Discussion

The Desk Evaluation application was previously built using Sails as a framework on the back-end side. Sails is a back-end framework based on the JavaScript programming language. Sails implement an MVC architecture that already supports API requirements [14]. Figure 5 shows the models and controllers of the MVC architecture in the Desk Evaluation application that was being maintained in this study.

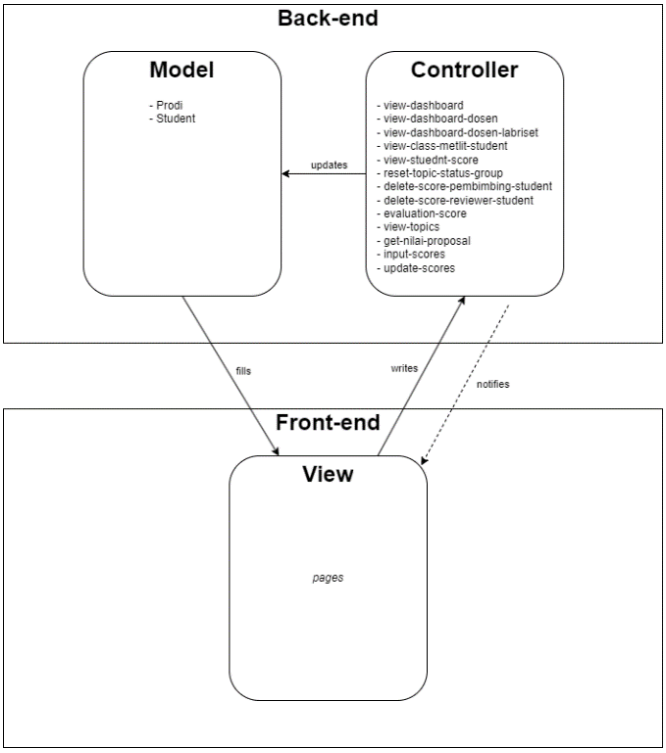

Figure 5. Maintained MVC of Desk Evaluation

The controllers that were maintained are divided into two parts according to their function, namely views and non-views. The controller that has the prefix view name is the controller that functions to return data from the model to the view. The data format returned by the controller with the view prefix name is JSON (JavaScript Object Notation), while non-view controllers do not return values but update or delete commands for existing data in the model.

```
const supertest = require("supertest");

describe("view-dashboard controller", function () {
  it("should redirect to /student/dashboard and return data to it", function (done) {
    const agent = supertest.agent(sails.hooks.http.app);
    agent
      .post("/entrance/login")
      .send({
        userIdVal: sails.config.custom.nimStudent,
        password: sails.config.custom.passwordStudent,
        userType: sails.config.custom.isStudent,
      })
      .expect(302)
      .expect("location", "/")
      .end(function (err, res) {
        if (err) return done(err);
        agent
          .get("/student/dashboard")
          .expect(200)
          .expect("Content-Type", "application/json; charset=utf-8")
          .end(function (err, res) {
            if (err) return done(err);
            console.log("Response: ", res.body);
            console.log("Status: ", res.status);
            done();
          });
      });
  });
});
```

Figure 6. Unit Testing on view-dashboard Controller

Figure 7. Testing Output of view-dashboard Controller

Controllers that had been maintained were tested using unit testing and load testing. Unit testing is the testing of a function performed by a developer who has an understanding of the structure of the application code so that it is included in the white box testing technique. Load testing is testing to ensure that the application can handle high loads. This test is included in the gray box testing because the tester has a limited understanding of the application, but tests the application with a tool designed to be able to find out the internal structure of the code [15].

Figure 6 shows the script of the unit testing that was performed on the view-dashboard controller. This unit testing was done using Mocha as the testing framework for the JavaScript program. Figure 7 shows the testing output of the view-dashboard controller, where the controller returns the desired JSON response and returns the status code 200.

The unit testing results of controllers that had been maintained are shown in Table 1. It shows that the unit functions on each controller can run well. Controllers that are functioning properly will send a response according to what was requested on the Desk Evaluation application page and marked with a check (✓) in the pass column.

Table 1. Unit Testing of Maintained Controllers of Desk Evaluation

No	Controller	Pass
1	view-dashboard	✓
2	view-dashboard-dosen	✓
3	view-pengumpulan-de	✓
4	view-dashboard-dosen-labriset	✓
5	view-class-metlit-student	✓
6	view-student-score	✓
7	reset-topic-status-group	✓
8	delete-score-pembimbing-student	✓
9	delete-score-reviewer-student	✓
10	evaluation-score	✓
11	view-topics	✓
12	get-nilai-proposal	✓
13	input-scores	✓
14	update-scores	✓

As a comparison, Table 2 shows the results of load testing or load testing on several existing Desk Evaluation application features that had not been maintained. The endpoints being tested are existing endpoints that will later undergo maintenance.

Table 2. Load Testing of Existing Controllers of Desk Evaluation

No	Controller	Virtual Users (VUs)	Average Request Time	Success Rate
1	view-dashboard	1000 VUs	6 requests/s	100%
2	view-class-metlit-student	400 VUs	3 requests/s	100%

Table 3. Load Testing of Maintained Controllers of Desk Evaluation

No	Controller	Virtual Users (VUs)	Average Request Time	Success Rate
1	view-dashboard	1000 VUs	6 requests/s	100%
2	view-dashboard-dosen	400 VUs	3 requests/s	100%
3	view-pengumpulan-de	400 VUs	5 requests/s	100%
4	view-dashboard-dosen-labriset	40 VUs	2 requests/s	100%
5	view-class-metlit-student	40 VUs	3 requests/s	100%
6	view-student-score	40 VUs	1 request/2s	98,4%
7	get-nilai-proposal	20 VUs	1 request/s	100%

The results of load testing is shown in Table 3. There are differences in the number of virtual users tested in each controller. This difference is because the number of students, lecturers, and admins does not have the same number so the number of virtual users is adjusted to the number of each role. It shows that all endpoints of controllers that had been maintained have an average request duration of under 30 seconds, which means that they have met the standard web application loading time [16]. The success rate on all load testing cases is between 98 and 100 percent so the probability of a server experiencing a load failure is less than 2 percent. Load Testing on Table 2 and Table 3 was done on the local server. Table 4 shows the specifications of the local server.

Table 4. Local Server Specification

Description	Specification
Operating System	Windows 11
CPU	AMD Ryzen 5 5600H
RAM	16 GB

After the maintenance was done, the source code of Desk Evaluation was analyzed using SonarQube. SonarQube is a tool that can check the source code. SonarQube can check the lines of code and adapt them to the programming rules according to the programming language [17].

As a comparison, Figure 8 shows the analysis of the source code before maintenance from the previous research and Figure 9 shows the analysis of the source code after maintenance using SonarQube, where the Desk Evaluation application's code reliability post-maintenance, which improved to an A rating from its initial C rating, as shown in Figure 8. This indicates that the lines of code written

are according to the applicable programming rules to minimize errors in the Desk Evaluation application.

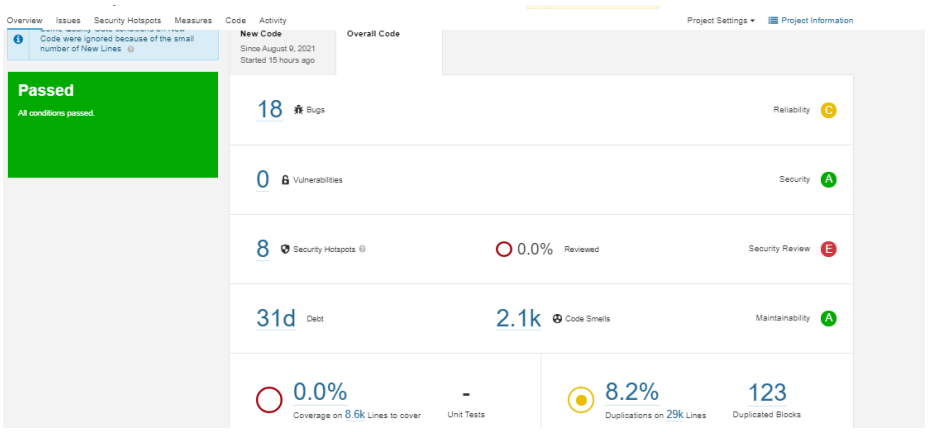

Figure 8. Analysis of The Source Code Before Maintenance [18]

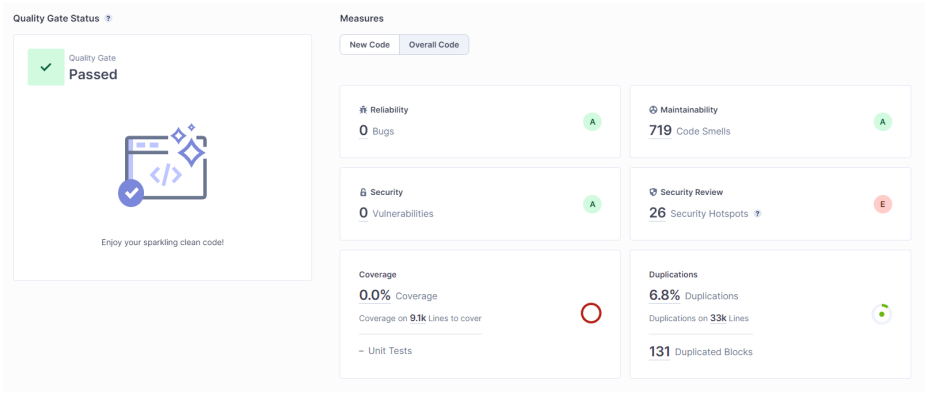

Figure 9. Analysis of The Source Code After Maintenance

D. Conclusion

This study concludes that it has succeeded in improving the performance of the Desk Evaluation application through maintenance on the back-end side. Maintenance includes changes to the logic side of models and controllers. In the model, attributes are added to add columns to tables in the database. Meanwhile, in the controller, there have been many changes including the creation of a new controller. Changes to the old controller aim to fix incorrect code and add missing code while creating a new controller aims to serve views or pages as well as new features proposed.

Testing of features that have been repaired shows good results. Unit testing on the controller gives pass results on all controllers, which means the controllers are running properly and according to the needs of the view or page.

The results of testing the processing duration of the Desk Evaluation application after being repaired showed satisfactory results. Endpoints that were tested have an average request duration of under 30 seconds and all endpoints had loading rates above 98 percent, indicating that the controllers have met the standard

average request duration [16] and the probability of the controller failing to cope with the request was extremely low.

The source code from the Desk Evaluation application that has undergone maintenance is checked using SonarQube, which is a tool that can check lines of code that have been maintained with compliance with programming rules according to the programming language used [17]. The results show that the source code of the Desk Evaluation application does not indicate any errors.

E. References

[1] A. Masrat, H. Gawde, and M. A. Makki, "Software Maintenance Models and Processes: An Overview," 2021. [Online]. Available: https://ssrn.com/abstract=3913486

[2] R. Rani and A. K. Cheema, "A Review on Software Maintenance Cost Evaluation," no. 09, 2015, [Online]. Available: www.ijspr.com

[3] B. Tarika, "A Review on Importance of Maintenance in Software Engineering," vol. 3, pp. 2617–4596, 2020, doi: 10.31058/j.mana.2020.31002.

[4] I. Sommerville, *Software engineering*. Pearson Education, 2016.

[5] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed., vol. 9. Boston: McGraw-Hill Education, 2020.

[6] N. Dissanayake and K. Dias, *Web-based Applications: Extending the General Perspective of the Service of Web*. 2017.

[7] S. J. Patel and P. D. Pancholi, "Implementation and Comparison of MVC Model in ASP.net Frameworkand PHP Framework," 2018. [Online]. Available: http://ijrar.com/

[8] M. S. Singh, "MVC Framework: A Modern Web Application Development Approach and Working," *International Research Journal of Engineering and Technology*, 2020, [Online]. Available: www.irjet.net

[9] J. Shetty, D. Dash, and A. K. Joish, "Review Paper on Web Frameworks, Databases and Web Stacks," *International Research Journal of Engineering and Technology*, 2020, [Online]. Available: www.irjet.net

[10] K. P. Hathwar and R. Ravishankar, "IJARCCE Back-End Web-Application Development and the Role of an Admin," *International Journal of Advanced Research in Computer and Communication Engineering ISO*, vol. 3297, no. 9, 2007, doi: 10.17148/IJARCCE.2017.6911.

[11] T. F. Kusumasari, I. Supriana, K. Surendro, and H. Sastramihardja, "Collaboration model of software development," in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics, ICEEI 2011*, 2011. doi: 10.1109/ICEEI.2011.6021769.

[12] A. Heryana, "Informan dan Pemilihan Informan dalam Penelitian Kualitatif," Dec. 2018.

[13] Grady. Booch, James. Rumbaugh, and Ivar. Jacobson, *The unified modeling language user guide*. Addison-Wesley, 1999.

[14] I. Nathan and M. McNeil, *Sails.js in Action*. Manning, 2017.

[15] N. Anwar and S. Kar, "Review Paper on Various Software Testing Techniques & Strategies," *Global Journal of Computer Science and Technology*, pp. 43–49, May 2019, doi: 10.34257/gjcstcvol19is2pg43.

[16] J. D. Meier, C. Farre, P. Bansode, S. Barber, and D. Rea, "Performance Testing Guidance for Web Applications," 2007.
[17] V. Lenarduzzi, F. Lomio, H. Huttunen, and D. Taibi, "Are SonarQube Rules Inducing Bugs?," Jun. 2019, [Online]. Available: http://arxiv.org/abs/1907.00376
[18] Y. Gianinda, "PENAMBAHAN MODUL PADA APLIKASI FINAL PROJECT MANAGEMENT (TA-PROPOSAL)," 2020.