Article

Implementation of Distributed Database Replication in Student Performance Assessment System to Improve Effectiveness and Monitoring

Wisnu Uriawan *, Rizka M Imron, Rifqi Syekhi Marsaputra, Salma Khoirunnisa, Wildan Fahtu Rijan, Wiki Nurrohman

Posted Date: 2 July 2024

doi: 10.20944/preprints202407.0091.v1

Keywords: Student Performance Evaluation System; distributed system; Agile methodology; database replication; registration; assessment form creation; grading; statistical analysis; performance testing; efficiency; transparency; higher education

Article

Implementation of Distributed Database Replication in Student Performance Assessment System to Improve Effectiveness and Monitoring

Wisnu Uriawan [1], Rizka M Imron [2], Rifqi Syekhi Marsaputra [3], Salma Khoirunnisa [4], Wildan Fahtu Rijan [5] and Wiki Nurrohman [6]

[1] Informatics Department, UIN Sunan Gunung Djati Bandung, Jawa Barat, Indonesia; wisnu.uriawan@uinsgd.ac.id
[2] Informatics Department, UIN Sunan Gunung Djati Bandung, Jawa Barat, Indonesia; rizkamimron@gmail.com
[3] Informatics Department, UIN Sunan Gunung Djati Bandung, Jawa Barat, Indonesia; rifqisyekhi@gmail.com
[4] Informatics Department, UIN Sunan Gunung Djati Bandung, Jawa Barat, Indonesia; salmakn555@gmail.com
[5] Informatics Department, UIN Sunan Gunung Djati Bandung, Jawa Barat, Indonesia; wildan.fr002@gmail.com
[6] Informatics Department, UIN Sunan Gunung Djati Bandung, Jawa Barat, Indonesia; browik09@gmail.com

Abstract: This research focuses on developing a Student Performance Evaluation System to enhance effectiveness and oversight in higher education. The system aims to address challenges faced by instructors in evaluating student performance using a distributed system approach. Agile methodology is employed in development, and database replication techniques are applied for data synchronization. System features include registration, assessment form creation, assessment submission, statistical visualization, and access to assessment history. The system is designed to improve efficiency and transparency in the assessment process. Key features of the system include registration process, assessment form creation, assessment submission, statistical visualization, and access to assessment history. Its goal is to enhance efficiency and transparency throughout the assessment process. The article details the system workflow and the roles played by Admins, Instructors, and students in system usage. Additionally, it discusses the back-end development model and pathways, as well as the performance testing results. Performance testing indicates that the system achieves high throughput of up to 21.55 requests per second, with an average response time of only 34 milliseconds and an error rate of zero percent. These results affirm that the system efficiently handles user loads and nearly instantaneously responds to requests, providing an optimal user experience in the context of student performance evaluation in higher education.

Keywords: Student Performance Evaluation System; distributed system; Agile methodology; database replication; registration; assessment form creation; grading; statistical analysis; performance testing; efficiency; transparency; higher education

---

1. Introduction

One of the most important techniques in human resource management is performance assessment. It serves as a way to evaluate a person's performance and functions as a basis for making many corporate strategic decisions. The SDM team can generate a comprehensive overview of the contributions and progress of each team member through performance assessments. It helps in determining fair compensation, identifying specialized training needs, career development, and efficient resource allocation.

Performance assessment also allows the company's strategy to be tailored. By understanding the strengths and weaknesses of each individual, organizations can tailor the goals, expectations, and support needed to optimal performance. In this case, performance assessment is not only an evaluation tool but also a powerful tool to guide and inspire individual development.

Not least, performance assessments also have a significant impact on employee motivation and behavior. When a performance assessment system is well designed and transparent, it can provide a clear incentive for employees to do their utmost, continue to grow professionally, and deliver superior results. Thus, performance evaluation is not only about judging the past, but also about building the foundation for future success, both for individuals and organizations as a whole [1].

The rapid development in Information and Communication Technology has had a significant impact on traditional educational systems and learning. Today, E-Learning has become a widely accepted way of learning. With increasing numbers of users, a wide range of learning services and the growth of educational content, E-Learning has emerged as the mode for future learning. However, the volatile user load and massive storage and transfer of rich multimedia content have lead to a need for effective utilization of server side system resources in providing E-Learning services. Cloud computing offers a dynamic provision of virtualized resources, elasticity, scalability, pays as you use and measured service with the ability to dynamically provision and de-provision computing resources as needed. Cloud computing is ideally suited for E-learning systems [2].

Assessment of student learning outcomes in higher education environments has an important role in improving the quality of education. Supervisors play an important role in providing assessment, input and guidance to students. However, the problem of available time and effective supervision and monitoring by supervisors often becomes an obstacle in the process of assessing student learning outcomes. With busy schedules and lots of assignments, it may be difficult for supervisors to give full attention to each student. In addition, due to limited access to necessary information and a lack of integrated systems to support the process, the process of tracking and monitoring student performance often fails. Previous research shows that the use of distributed systems and rating scale methods can effectively facilitate the performance assessment process. However, in the context of supervisors' assessment of student performance, the application of this concept has not been utilized optimally [3].

Lecturers often face significant challenges in managing time and efficiency when monitoring and tracking student performance in the academic environment. The inability to provide adequate attention or carry out effective monitoring can interfere not only with the process of quality assessment, but also affect the important interaction between faculty and students, which should be the foundation for solid academic development.

To address this challenge, the study will focus on the development and implementation of distributed systems specifically designed to evaluate student learning outcomes under the guidance of a lecturer. By adopting a distributed system development approach and following the Software Development Life cycle (SDLC) methodology, the development phase will include in-depth requirements analysis, careful system design, thorough implementation, comprehensive testing, and effective implementation.

It is expected that through this approach, this research can provide practical and measurable solutions to the challenges faced by lecturers in evaluating student performance. By improving the effectiveness of the evaluation process, the proposed system has the potential to improve the overall quality of education, ensuring that each student receives appropriate support and supports them in reaching their full academic potential [4].

2. Related Work

Apart from that, research on teacher performance at SMP Negeri 2 Sukodono showed very good results with a score of 82.43 percent, while the development of a performance assessment system for intern assistants at Muhammadiyah University Gresik used a questionnaire assessed by intern participants [5].

Various research has been carried out to develop a distributed student learning outcomes assessment system. In the first research, web-based SIPENSI was developed to improve the performance of Muhammadiyah Elementary School teachers in Bendo Kulon Progo who obtained an average score of 4.3 and a percentile score of 86.7 percent from expert assessments and management system users [6].

Next, look for a web-based sales performance measurement system at PT. Harsindo Oetama Perkasa produces a system that helps determine daily sales targets and provides printed reports based on date for easy access [7].

Evaluation of the implementation of civil servant performance assessments based on a 360 degree feedback system at the BKN Region VIII Office shows objective, measurable, responsible and inclusive results of job satisfaction and organizational learning. Overall, this system has been proven to be effective and efficient in improving teacher and student performance, as well as facilitating data access, through various research methods including R and D, SWOT analysis, black box testing, quantitative and waterfall development modeling [8].

Apart from that, developing an employee performance evaluation system based on a distributed system at CV. Many blessings. In system development, the SDLC (Software Development Life Cycle) method with the waterfall model is used. The work performance evaluation method applied is the Rating Scale method. The research results show that the application of the rating scale method in a distributed system-based employee performance assessment application can help CV Berkat Abundant carry out performance assessments more easily. With 94 percent of respondents stating that the system can make it easier to see assessment results, the conclusion of this research is that the system is effective in facilitating the employee performance assessment process [9].

3. Methodology

3.1. Agile Method

Distributed System-Based Student Performance Assessment System: Overcoming Availability Problems and Increasing Supervision Effectiveness using the SDLC Agile Method.

Agile is a project management technique that uses short development cycles, otherwise called sprints, which focus on continuous improvement in the development of a product or service. Agile techniques focus on step-by-step development, where software is released incrementally, reducing process effort, producing high-quality code, and involving customers directly in the development process [10].

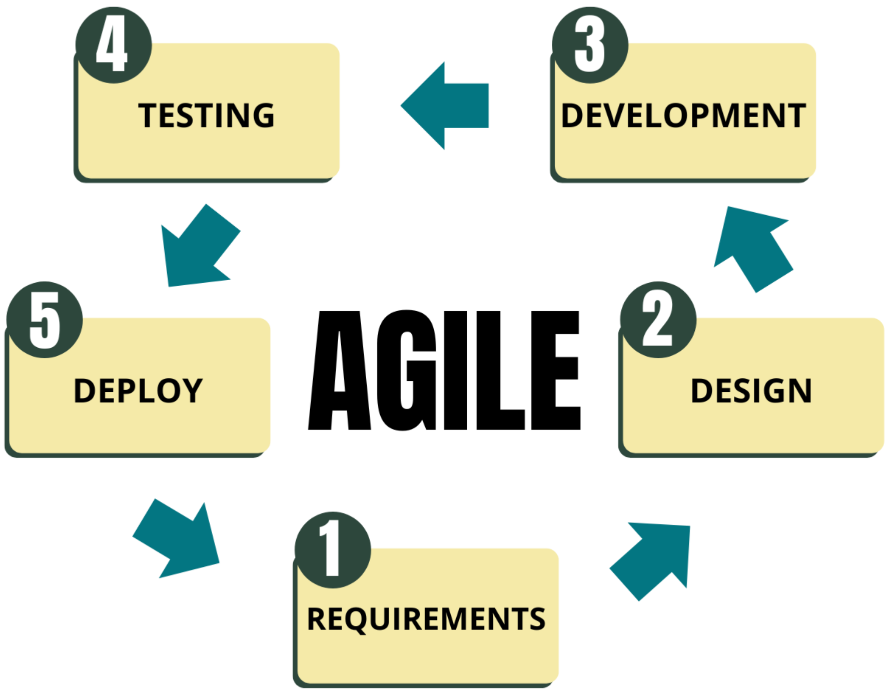


Figure 1. Agile Method

1. Requirements

   This stage involves an assessment of the user to find out in detail the software requirements by the user. This analysis includes the software, hardware, and hosting requirements of the application. The result of this analysis is the software, hardware, and hosting specifications that will be used by the personnel information system [11].

   At this stage too, the team gathers and defines the needs or requirements of the users. The focus is on deeply understanding what the users need and documenting those needs in detail. This process starts with direct interviews with users to identify their needs related to the project or system to be developed.
2. Design

   Once the needs have been gathered, the next stage is to design a solution to meet those needs. This includes the design of the system architecture, user interface, and other technical aspects required for implementation.

   This stage includes architectural design, business process design, and database design. This design uses UML (Unified Modeling Language) and ERD (Entity Relational Diagram) modeling. UML modeling will produce use case diagrams, class diagrams, and schema diagrams [11].
3. Development

   The development stage is the process in which the design that has been created is translated into software code. The development team works to write, test and integrate code to build the features that have been designed.
4. Testing

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2024 doi:10.20944/preprints202407.0091.v1

After development, the software is tested to ensure that all functions run correctly and there are no bugs or errors. This testing can include various types, such as unit testing, integration testing, system testing, and user acceptance testing [11].

In the AGILE method, testing is done using Postman to ensure that all API endpoints function correctly. Postman is used to send requests and receive responses from the server, verify that the output is as expected, and ensure that each application function works according to predefined specifications. The results of these tests are recorded in the form of a table that shows the status of each test and ensures the application is running properly.

5. Deployment This stage is the process of deploying the tested software to a production environment where users can start using it. This can include installation, configuration, and other settings needed to run the software.

   This stage is the process of uploading the application to web hosting so that it can be accessed by users via the internet. Deploy the application is the stage of handing over the software to users to be used in accordance with the system analysis and design. This process involves uploading the application from the local side to the hosting using SSH and Git Version Control [11].

3.2. Database Replication

Database replication is a technique used to copy and distribute data from one database to another and ensure database synchronization. Through replication, data can be distributed to various locations, and users can access it through internet, dial-up, LAN, WAN, or wireless connections.

This process is often performed in distributed database systems to overcome problems such as damage or loss of data in the database. This technique allows data to be accessed in real-time and securely, and automatically updates data on the backup server when one of the networks goes down, so that the system continues to run normally. Replication is essential to protect the availability of accurate data and improve performance [2].

In the context of database replication, the main data distribution or ownership is referred to as the master, while the database that receives the data distribution is referred to as the slave. The purpose of replication is to reduce server latency when accessing data to be sent to other computers and to add and improve database performance. One of the uses of the replication process is when a database goes down, so the data can still be accessed through the backup or slave database. This helps maintain the availability and reliability of the database system.

Database replication techniques are important because they allow copying and distributing data and database objects from one database to another, and synchronizing between databases so that data consistency can be guaranteed. Replication supports data availability at any time and anywhere it is needed, allows multiple locations to store the same data, and supports application performance and physical data deployment according to its use,

3.3. Master-master Replication

The Master-Master replication model allows data to be replicated through distributed writes and reads, which helps distribute the load of writes and reads between the two master nodes. However, due to its loose nature, this model has consistency issues, making it difficult to maintain absolute consistency among replicated servers [12].

MySQL Master-Master replication is referred to as “mirror” because it updates data in real-time to ensure that both servers have the latest version of data. This setup may cost a lot of money, but the benefits include reduced redundancy, easier access, and longer application life.

3.4. Aspects Related to Master-master Replication

1. Topology and burden Distribution

Master-master replication helps divide the load of writing and reading data evenly between the two master nodes, which enables writing and reading data in a distributed manner. This has many key advantages, such as improving performance and availability, and reducing the risk of bottlenecks or points of failure [12].

2. Consistency Issues

   It is very difficult to ensure perfect consistency among duplicated servers due to their flexible structure. The data on the replicated servers is not always the same. This can happen due to concurrent updates or delays in the data replication process.

3. Update Latency Issues

   In addition, Master-Master replication causes update latency issues. Sometimes there is often a delay before changes are actually distributed to all replicas because each update must be approved and synchronized among the nodes involved. This can impact the availability of up-to-date information and the response speed of the system.

4. Lazy Replication

   Lazy replication is often recommended to address update latency issues. In lazy replication, updates to a single replica are applied to multiple nodes before being propagated asynchronously through a batch or background update process. This method reduces the overhead and delay associated with synchronous replication.

5. Eager and Lazy Replication Techniques

   (a) Eager Replication

   Eager replication promotes data integrity by resolving conflicts and synchronizing updates across replicas before transactions are confirmed; however, due to synchronous operation, it can increase communication costs and latency.

   (b) Lazi Replication

   By updating the local copy first and delaying synchronization to other replicas until the transaction is confirmed, lazy replication optimizes performance and reduces live load. However, this method may cause temporary mismatches between replicas.

3.5. *API performance meter*

1. Response Time

   Knowing how long an API takes to process a request is an important part of understanding the basic functionality of an application and its infrastructure. API response time consists of several key components, such as the time it takes for the server to process the request and send the response to the client.

2. Throughput

   Measures the number of requests an API can handle per second or per minute. A high throughput indicates the API's capacity to handle many requests simultaneously.

3. Error Rate

   Knowing the type and number of errors that occur when an API handles a request is an important step in assessing the reliability and stability of an API. It helps in finding issues that affect user experience and API performance, and offers information on areas that need improvement.

4. Result and Discussion

4.1. *Functional Requirements*

The student assessment system has several key features designed to ensure that the process of assessing student performance runs smoothly, accurately, and according to standards. These features

doi:10.20944/preprints202407.0091.v1

allow lecturers to register, log into the system, create an assessment form, fill out the form, view assessment statistics, and access the history of assessment results. This system was created to improve the efficiency and transparency of the student performance assessment process. Each functional requirement that has been identified is explained in detail as follows:

1. Register

   In the student assessment system, the registration feature allows admins to create accounts for lecturers. Lecturers are responsible for entering the information required for registration, such as lecturer's full name, NIP, password, and password confirmation. The feature also has an option for forgotten password, which allows lecturers to recover their account if needed. This process results in a new lecturer account being registered in the system. One part of the registration process is validation to ensure that the data entered is completely valid and complete, as well as a password strength check and confirmation that the password matches the password confirmation. Before the lecturer account is activated, the administrator ensures that all data entered is correct.
2. Login

   By utilizing the login feature, registered lecturers can enter the system using the registered email and password. This login process requires input in the form of an email address and password. If authentication is successful, the resulting output is access to the lecturer dashboard. To ensure security, the system performs validation by checking the suitability of the email and password entered with the data registered in the database.
3. Create a student performance assessment form

   This feature allows lecturers to create assessment forms that will be used to assess student performance. This form can include various assessment criteria according to the lecturer's needs. This process requires input in the form of form name, description, and assessment criteria such as attendance, participation, assignments, exams, and others. The output of this process is a new assessment form that is ready to be used to assess students. To ensure that the form is complete and can be used properly, the system will validate the completeness of the information required for each assessment criteria.
4. Fill Out The Performance Assessment Form Statistics

   With this feature, lecturers can fill in student performance assessment forms based on predefined criteria. They can provide scores or comments related to each assessment criteria, ensuring that the evaluation covers various important aspects of student performance. This feature allows lecturers to provide detailed and constructive feedback, ensuring that the assessment process is systematic and structured.
5. Statistik

   This feature provides reports and statistical analysis based on the assessment data that has been entered. Lecturers can view the distribution of grades, grade averages, and overall student performance in the form of graphs or tables. To use this feature, lecturers must enter a selection of time periods or other parameters that are relevant for analysis as input. After the parameters are entered, the system will perform validation to ensure that the parameters entered are appropriate and can be used to generate relevant and accurate reports. The output of this feature is a graph or table displaying student assessment statistics, giving lecturers an in-depth insight into student academic performance within the selected period.
6. Assessment Result History

   This feature allows lecturers to view the history of assessment results that have been given to students. Lecturers can view assessments given at various time periods or for various courses.

To use this feature, lecturers must enter the choice of a certain time period or course as input. After the input is given, the system will display the assessment history in the form of a list or table. The output of this feature is a structured assessment history, giving lecturers easy access to review and analyze assessments that have been given to students in various contexts and periods.

4.2. System and Software Design

The architecture used in this system design consists of three main components. They are the Front-and, API and also the Database. The frontend is used by users to interact with the system, the API allows other applications to interact with the system, and the database stores the data. The system also has a replication component that is used to ensure that data is always available.

1. Architecture Sistem

System architecture is a method used to align the organization's business needs with application needs to support the vision and mission to be achieved. This method is known as Enterprise Architecture or Information Systems and Information Technology Architecture. [13]

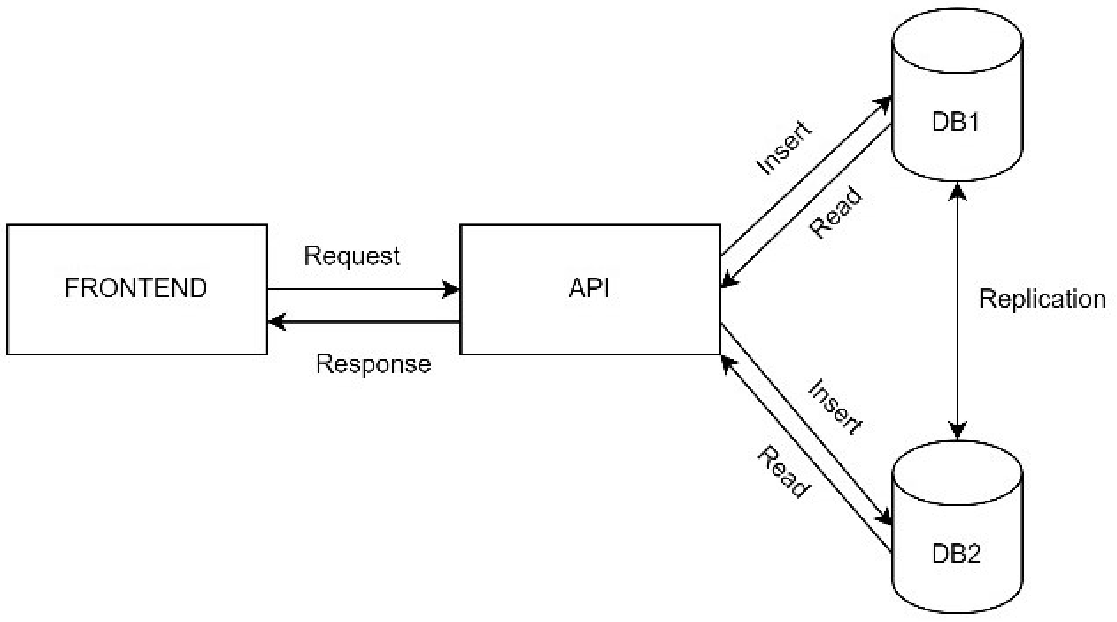


Figure 2. Architecture System

In the figure above regarding the system architecture flow, the user interacts with the front-end to perform an action, such as entering new data or searching for existing data. When the user performs this action, the front-end sends a request to the API. This request contains information about the action the user wants to perform as well as the data required to perform the action. Upon receiving the request from the front-end, the API processes the request and passes it to the database. The database then processes the request by executing the appropriate query, whether it is to retrieve data, store new data, update existing data, or delete data.

The result of this processing is a response containing either the data the user requested or information regarding the status of the action that has been performed. This response is then sent back from the database to the API. After receiving the response from the database, the API sends it back to the front-end. Finally, the front-end displays the response to the user, allowing the user to see the results of their actions, such as the requested data, confirmation that the data has been successfully saved or updated, or an error message if any problems have occurred.

(a) Replication

The system also has a replication component. Replication is the process of copying data from one database to another. Replication is used to ensure that data is always available, even if one of the databases fails. In this system, replication is used to copy data from DB1 to DB2. This ensures that if DB1 fails, the data is still available in DB2.

2. Flow Sistem

A system flow is a sequence of steps performed within a system to transform inputs into desired outputs. It begins with identifying needs and designing the system, followed by collecting the necessary data. This data is then processed according to specific rules or algorithms, resulting in outputs such as information, reports, or physical actions. Subsequently, the results are evaluated and feedback is provided for further improvements, while control mechanisms are implemented to ensure the process operates as intended. All these steps work together in an integrated manner to achieve efficiency and effectiveness within the system.

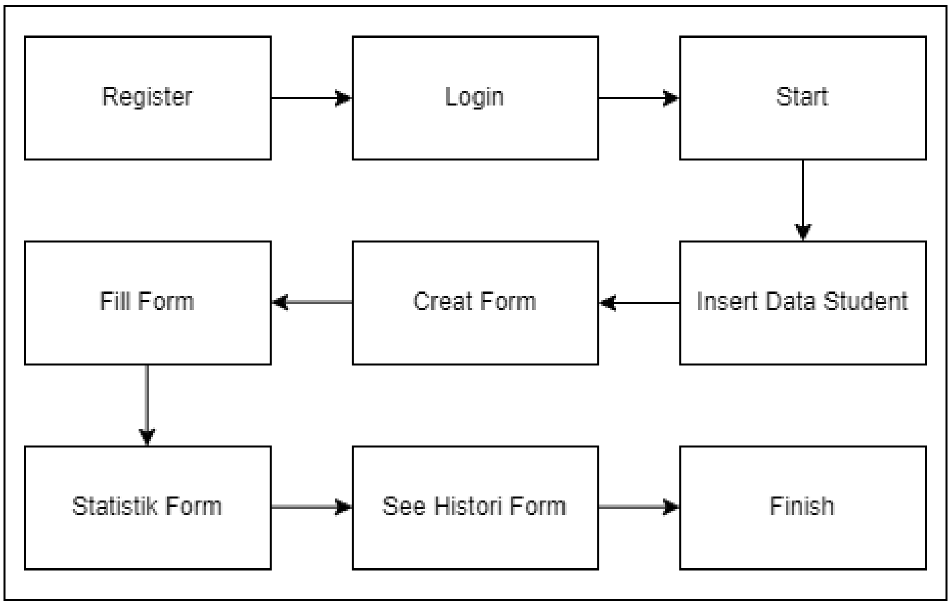


Figure 3. Flow System

(a) Register

At this stage, the admin creates and manages the assessment form in the system, as well as registers lecturers by entering personal information such as name, email address, and password. By registering, the lecturer will have a personal account and can access the system features. The admin is responsible for all information provided by the lecturer during the registration process.

(b) Login

Registered lecturers can access the assessment form system with their email address and password. The system will verify the lecturer's credentials to ensure that they have authorized access. After login, lecturers can access various features related to form management.

(c) Start

After logging in, lecturers can fill in the assessment form created by the admin. Lecturers can start by clicking the "Fill in the Assessment Form" button or selecting the appropriate form management option. The system will display the assessment form interface, where lecturers can fill in the structure and content that has been determined by the manager.

(d) Fill Student Data

Lecturers can choose to use an external database to enter student data or manually select data from a list to allow the assessment form to connect with existing student data.

(e) Create form

Admin will create an assessment form in this section. According to the lecturer's needs, this form can include various assessment criteria, such as form name, description, and assessment criteria, such as attendance, participation, assignments, exams, etc. The system will collect user input and create the form structure and store it in the database. After the assessment form is created, the manager will give it to lecturers to assess their students.

(f) Fill Form

Once the admin creates and assigns an assessment form, lecturers can access it through a shared link, an embedded form on the website, or a direct form URL. Lecturers can fill out the form by entering data into the specified fields.

(g) Statistic Form

To gain an understanding of the usage and performance of the assessment form, lecturers can view the statistics of the assessment form. These statistics are displayed in tables or graphs and include grade distribution, average grade, and overall student performance.

(h) assessment history

To monitor the use of previous assessment forms, lecturers can view the history of assessment form submissions. This also allows lecturers to view the results of assessments that have been given to students at different time periods or for various courses.

(i) finish

Once the process of creating and managing assessment forms is complete, managers can continue to create, edit, fill, and check forms as required, and then send them to lecturers. [13]

3. Entity Relationship Diagram

To design a database, an Entity Relationship Diagram (ERD) is used as a structural diagram that describes the data to be stored in the system and its boundaries. ERD helps in visualizing the relationship between data entities and how they interact with each other, providing a clear picture of the database structure to be created. [14]

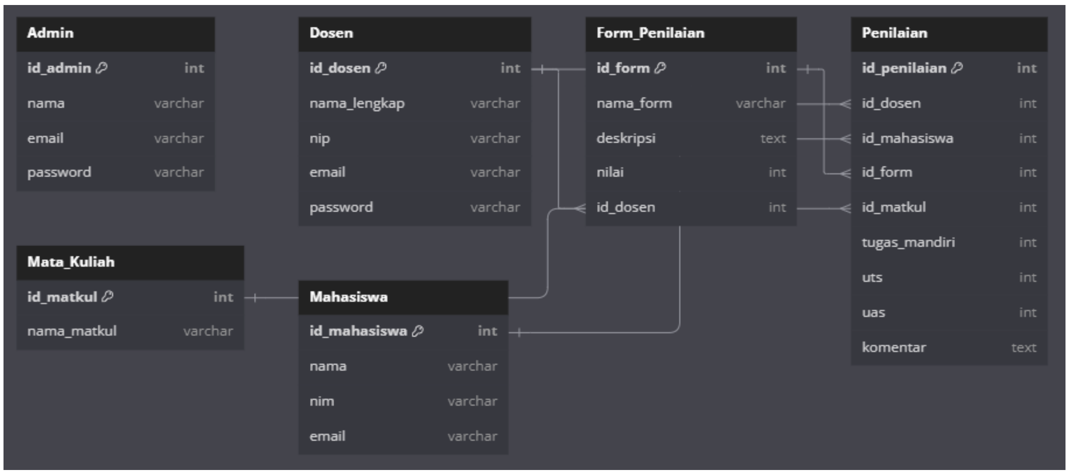


Figure 4. Entity Relationship Diagram

(a) Admin Table

The Admin table stores the data of the admins who use the system. All admins have a separate identity known as the Admin ID; their name, email address, and password are used to verify their identity when they log into the system.

doi:10.20944/preprints202407.0091.v1

(b) Lecturer Table

The Lecturer table stores information about lecturers registered in the system. Lecturer ID is a unique identity for each lecturer. Full name, NIP, email address, and password are used to authenticate lecturers when logging into the system.

(c) Table Form Assessment

Form Table The assessment table stores information about the assessment forms used in the system. The form ID is a unique identity for each assessment form. The form name and description are used to identify the assessment form.

(d) Assessment Table

The Assessment table stores information about assessments made by lecturers to students. The assessment ID is a unique identity for each assessment. Value, comment, lecturer ID, student ID, assessment form ID, and course ID are used to record assessment results.

4. Use Case Scenario

(a) Prerequisite

Users need to have the right credentials in order to access the system.

(b) Admin For Use Case

At this stage, the admin creates and manages assessment forms in the system, as well as registers lecturers by entering personal information such as name, email address, and password. By registering, the lecturer will have a personal account and can access the system features. The admin is responsible for all information provided by the lecturer during the registration process. The admin will also create an assessment form, which may include various assessment criteria such as form name, description, and assessment criteria such as attendance, participation, assignments, exams, etc. The system will collect user inputs and create the form structure and store it in the database. Once the assessment form is created, the admin will provide it to lecturers to assess their students. After the process of creating and managing the assessment form is completed, the admin can continue to create, edit, fill, and check the form as needed, and then send it to the lecturer.

- Registration: The system administrator creates an account.
- Login: Using their login credentials, administrators access the system. Enter Student Data: Administrators update the student database.
- Create Assessment Form: Administrators create forms to evaluate student performance.
- View Assessment Statistics: The administrator checks the statistical data for the student's overall assessment.
- Finish: Administrators complete their task.

(c) lecturers for Use Cases

Registered lecturers can login with their email and password to access the form management feature. After login, lecturers can fill in the assessment form created by the admin by clicking "Fill in Assessment Form" and selecting student data from an external database or manual list. The form is accessed through a link, website, or direct URL, and filled in as per the specified fields. Lecturers can view form statistics in tables or graphs to understand student performance and grade distribution, as well as review assessment form submission history for various courses and time periods.

- Registration: Lecturer creates a system account.
- Login: Using their login credentials, lecturers access the system.
- Enter Student Data: Lecturers enter new student information into the database.
- Complete Assessment Form: Lecturers complete the assessment form created by the admin by providing relevant details about the student's performance.
- View Assessment Statistics: Lecturers check student assessment statistics.
- View Assessment History: The lecturer reviews the student's assessment form history.
- Finish: Lecturers end their assignment.

(d) Student for Use Case

Students do not have a direct role in this use case scenario, because they only act as objects of assessment in forms filled out by lecturers and created by admins. However, the results of the assessment filled in by the lecturer will affect their academic performance which can be seen through form statistics or assessment history.

(e) Addition

- Invalid login information: Users are asked to enter their credentials again by the system.
- Data validation error: The user receives a notification from the system to correct the error.
- Incomplete form completion: The user was prompted to complete the form by the system.

5. Activity Diagram

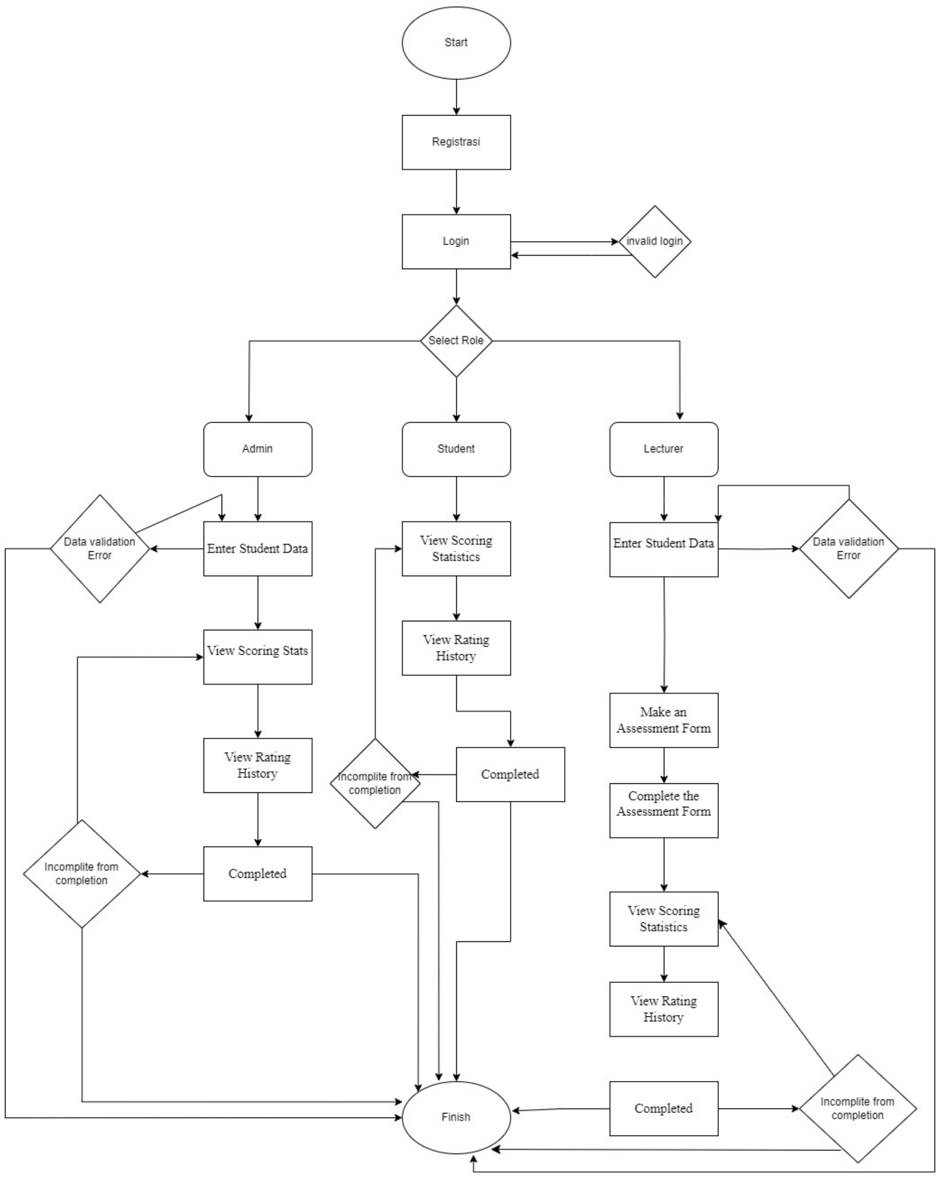


Figure 5. Diagram Activity

This diagram illustrates the activity flow of the three main roles in the system. Admin, Student, and Lecturer, with each role having specific tasks and activities. Each role has steps to follow and if there are any errors or incomplete processes, they must be corrected or completed before reaching completion.

(a) Start

The process starts from this point, signaling the beginning of the user's interaction with the system.

(b) Registration

New users can register for an account by filling out the registration form provided by the system. This form usually contains personal information such as name, address, email, and phone number, as well as relevant academic data such as NIM, major, and class.

The system will verify the information entered to ensure its validity and completeness. If the data entered is valid, the system will create a new user account and send a confirmation email to the registered email address. Users can immediately login to the system using the username and password that was created during registration.

This registration process is important to ensure that only authorized and interested users can access the system and use its features.

(c) Login

After completing the registration process, users can access the system using their login credentials, namely the username and password that have been created previously. Users need to enter their username and password on the login page provided by the system.

The system will verify the login credentials entered. If the username and password are valid, the user will be granted access to the system and directed to the main page according to their role (administrator, student, or lecturer).

However, if the login credentials entered are invalid (for example, incorrect username or password), the user will be redirected back to the login page and an error message will be displayed explaining the cause of the login failure. Users will need to correct their login credentials and try logging in again.

This login process is important to ensure that only authorized and interested users can access the system and protect the data and information contained therein.

(d) Select Role

Upon successful login, users will be prompted to select their role in the system. The system provides three main roles, namely:

- Administrator
  Has full access rights to manage the system and data. Administrators can add, delete, or edit users, set tasks and classes, and access all data within the system.
- Student
  Students can view information related to their assignments and assessments. Students can view assignment descriptions, submission deadlines, grades earned, and provide comments on assignments.
- Lecturer
  Lecturers can manage assignments, assessments, and student data. Lecturers can create and edit assignments, give assessments to assignments done by students, and manage student data such as grades and attendance.

(e) Activity Flow by Role

1) Administrator

The administrator enters the student's personal and academic data information into the system, where the system then validates the data to ensure its completeness and correctness. If there are errors in the validation, the administrator will be redirected to correct the data. Furthermore, the administrator can access and view overall student assessment statistics, which include average scores, score distribution, and performance comparison between students.

Lecturers have the task of creating assessment forms to evaluate student performance in specific assignments or courses. These forms include grading criteria, weights, and specific questions relevant to the course.

Administrators can also view each student's assessment history, including completed assessment forms and comments from lecturers. This information is helpful in monitoring student learning progress and identifying areas for improvement.

The administration process is considered complete once the administrator has completed all the required tasks. After that, the administrator can exit the system or return to the "Select Role" step to perform other actions.

2) Student

Students can view their own assessment statistics, such as average score, score distribution, and comparison with classmates. This information helps students understand their performance and identify areas for improvement. In addition, students can also view their previous assessment history, including completed assessment forms and comments from lecturers. This information is useful for students to track their learning progress and see how their performance changes over time.

The process for students is considered complete once they view their assessment information and assessment history. After that, students can exit the system or return to the "Select Role" step to access other available features.

3) Lecturer

Lecturers enter student personal and academic data information into the system. The system then validates the entered data to ensure its completeness and correctness. If there is an error in validation, the lecturer will be redirected to correct the data.

Next, lecturers fill in the assessment form for each student that has been created by the admin by providing scores and comments based on the results of their assessment. This information is important to provide constructive feedback to students and assist them in improving their understanding.

Lecturers can also view the assessment statistics of the students they teach, such as the average score, score distribution, and comparison between students. This information helps lecturers in analyzing overall class performance and identifying areas for improvement in learning.

In addition, lecturers can view each student's assessment history, including completed assessment forms and comments from other lecturers.

6. Finish

   The workflows of three different types of users-Admin, Student, and Lecturer-and how they interact with the system to complete their work are shown in this diagram. Although the flow of each function is different, they all lead to a "Done" status and, ultimately, a "Completed" procedure. Users are directed to complete or correct the process if there are any errors.

4.3. Development System

At this stage, programmers implement system development. System development uses the Agile method, which includes writing code, integrating components, testing to ensure functionality, and documentation to support use and maintenance. In the Agile approach, programmers work in short iterations and collaborate with other teams to ensure all technical and functional aspects of the system are implemented correctly and on schedule.

1. Back-end Developer

   This student assessment form project uses JavaScript programming language with Node.js as the runtime environment. The main framework used is Express.js which all shows the creation of APIs and web applications. This project also uses Postman for API testing. Sensitive configuration settings, such as database connections or API keys, are managed through env files. In addition, this project will be deployed using Vercel, a platform for deploying front-end and back-end applications. Dependencies management is done using the Node Package Manager (NPM).

In this project, there are also several models and routes. for more details, it will be explained as follows:

(a) Model

- Admin.js
  The 'Admin.js' model will define the Admin table structure using Sequelize, an ORM for Node.js. This model imports the data types from Sequelize and the database configuration. Then, the Admin model is defined with the id_admin column as the primary key which is of type integer and auto-increment, and the name, email, and password columns which are all of type string. In the model options, the table name is set as "Admin" and the timestamp setting is disabled, so the createdAt and updatedAt columns will not be created automatically. This model is exported so that it can be used in other parts of the application to perform CRUD operations on the Admin table.
- Dosen.js
  The above 'Dosen.js' model defines the structure of the 'Teacher' table using Sequelize, an ORM (Object-Relational Mapping) for Node.js. The model imports the data types from Sequelize and links the database configuration from the 'database.js' file. In the 'Dosen' model definition, there are several columns: 'id_dosen' as the primary key which is an integer and auto-increment, 'name_full' which is a string to store the lecturer's full name, 'nip' which is also a string to store the employee identification number, and 'email' and 'password' which are strings to store the lecturer's email and password respectively. The model options set the table name as "Lecturer" and disable automatic generation of 'createdAt' and 'updatedAt' columns by setting 'timestamps: false'. This model is exported to allow its use in other parts of the application, so that CRUD (Create, Read, Update, Delete) operations can be performed on the 'Teachers' table via Sequelize.
- FormPenilaian.js
  The FormPenilaian.js model defines the structure of the FormPenilaian table using Sequelize. This model imports data types and database configurations, as well as the Lecturer model to organize relationships between tables. In the FormPenilaian model, there is an id_form column as an auto-increment primary key, form_name (string) for the form name, description (text) for the form description, and value (integer) for the assessment value. The id_dosen (integer) column functions as a foreign key that refers to the id_dosen column in the Lecturer table. The table name is set as "Form_Assessment" and the automatic creation of createdAt and updatedAt columns is disabled with timestamps: false. This model is exported to allow CRUD operations on the FormPenilaian table as well as manage the relationship with the Lecturer table.
- Mahasiswa.js
  The 'Mahasiswa.js model is an integral part of a Node.js application that uses Sequelize as an ORM to interact with a relational database. The model represents a student entity with key attributes such as student_id as a primary key that is automatically incremented, as well as name, nim, and email fields to store basic student information. By disabling the automatic timestamp option, this model allows applications to efficiently perform CRUD (Create, Read, Update, Delete) operations on student data without additional createdAt and updatedAt columns. By being exported through module.exports, Student.js not only provides an organized data structure, but also facilitates effective management of student data in Node.js applications.
- MataKuliah.js
  The MataKuliah.js model in the Node.js application uses Sequelize as an ORM for interaction with relational databases. The model represents a course entity with the

primary attributes id_matkul, an auto-increment INTEGER type primary key, and name_matkul, a STRING field for the course name. Additional configurations include tableName: "Mata_Kuliah" to set the table name and timestamps: false to disable the automatic createdAt and updatedAt columns. By exporting this model, the application can perform CRUD (Create, Read, Update, Delete) operations against course data efficiently, ensuring organized data management.

- Penilaian.js

  The Penilaian.js model in the Node.js application uses Sequelize to manage interactions with the database. This model represents the assessment entity with the main attribute id_assessment as an auto-increment primary key of type INTEGER, and several other INTEGER columns such as id_dosen, id_miswa, id_form, and id_matkul which refer to the Lecturer, Student, FormAssessment, and Course models. In addition, there are columns of assignment_self, uts, uas to store the value of the assessment component, as well as comments of type TEXT. With tableName configuration: "Assessment" and timestamps: false, this model disables the automatic createdAt and updatedAt columns. The model is exported via module.exports, allowing applications to perform CRUD operations on assessment data efficiently.

(b) Routes

On the back-end there are files and folders containing Routes that govern how the back-end application responds to HTTP requests. In this context, the routes file is used to define endpoints or paths that clients (such as front-end applications or other tools like Postman) can access to interact with the server.

- admin.js

  The admin.js file defines CRUD (Create, Read, Update, Delete) routes for Admin entities using Express. At the beginning, Express and the Admin model are imported and a router from Express is created. The POST/ route is used to create a new admin with the data from the req.body, returning a status of 201 Created if successful or 400 Bad Request if failed. The GET / route returns all admin data from the database with a status of 200 OK or 400 Bad Request if it fails. The GET /:id route retrieves admin data by id, returning 200 OK if found, 404 Not Found otherwise, or 400 Bad Request if there is an error. The PUT /:id route updates admin data by id with new data from req.body, returning the updated data with a status of 200 OK, or 404 Not Found if the admin is not found, and 400 Bad Request if it fails. The DELETE /:id route deletes the admin by id, returning status 204 No Content if successful or 404 Not Found if not found, and 400 Bad Request if failed. This router is exported for use in the main application.
- Dosen.js

  The Dosen.js file defines the CRUD route for Lecturer entities that use Express. At first, Express and the Lecturer model are imported, and a router from Express is created. The POST / route is used to create a new lecturer with data from the req.body, returning a status of 201 Created if successful or 400 Bad Request if failed. The GET / route returns all lecturer data from the database with a status of 200 OK or 400 Bad Request if an error occurs. The GET /:id route retrieves lecturer data by id, returning 200 OK if found, 404 Not Found otherwise, or 400 Bad Request if an error occurs. The PUT /:id route updates lecturer data based on id with new data from req.body, returning the updated data with a status of 200 OK if successful, or 404 Not Found if the lecturer is not found, and 400 Bad Request if an error occurs. The DELETE /:id route deletes lecturers by id, returning status 204 No Content if successful or 404 Not Found if the lecturer is not found, and 400 Bad Request if an error occurs. This router is then exported for use in the main application.
- FormaPenilaian

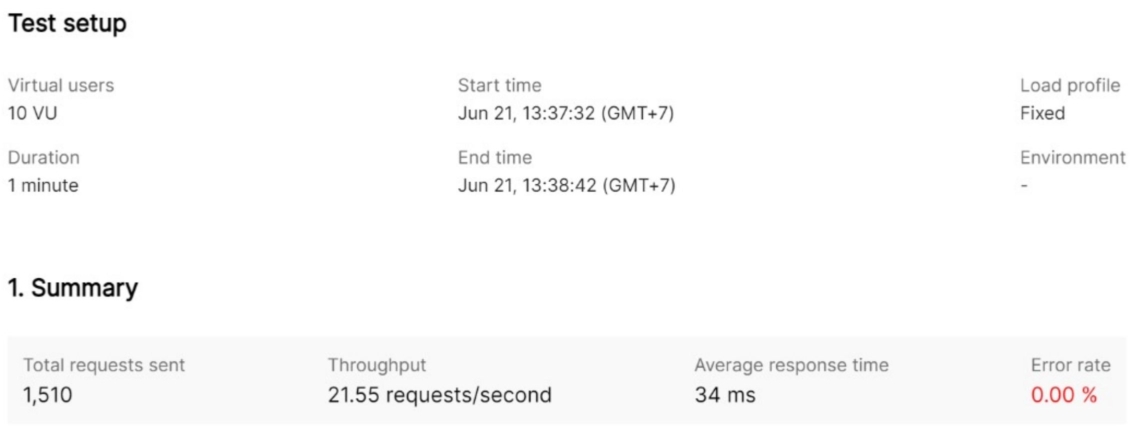
The FormaPenilaian.js module uses Express to manage CRUD operations on the FormaPenilaian model. The module imports Express and the FormaPenilaian model, and creates a router object to define various API routes. The POST/ route is used to create a new FormaPenilaian entity with data from the req.body, returning it with status 201 or sending a 400 error if it fails. The GET / route is used to read all FormaPenilaian entities from the database, returning them in JSON or sending error 400 if there is a problem. The GET /:id route is used to read one FormaPenilaian entity by id, returning it if found or sending a 404 error if not found. The PUT /:id route is used to update the FormaPenilaian entity by id, returning the updated data if successful or sending a 404 error if not found. Finally, the DELETE /:id route is used to delete the FormaPenilaian entity based on id, returning status 204 if successful or sending a 404 error if not found. Each of these CRUD operations is wrapped in a try-catch block to handle errors and send the appropriate response.

- Mahasiswa.js
  The mahasiswa.js module uses Express to manage CRUD operations on the Student model. The module imports Express and the Student model, and creates router objects to define various API routes. The POST/ route is used to create a new Student entity with data from the req.body, returning it with status 201 or sending status 400 if an error occurs. The GET / route is used to read all Student entities from the database and return them in JSON, or send status 400 if there is a problem. The GET /:id route is used to read one Student entity based on the id given in the URL parameter, returning it if found or sending a 404 status if not found, as well as sending a 400 status if an error occurs. The PUT /:id route is used to update the Student entity based on the id with data from the req.body, returning the updated data if successful or sending a 404 status if not found, as well as sending a 400 status if an error occurs. Finally, the DELETE /:id route is used to delete the Student entity based on id, returning status 204 if successful or sending status 404 if not found, as well as sending status 400 if an error occurs. Each of these CRUD operations is wrapped in a try-catch block to handle errors and send the appropriate response.
- matakuliah.js
  The matakuliah.js module uses Express to manage CRUD operations on the Courses model. The module imports Express and the Courses model, and creates a router to define various API routes. The POST route / creates a new Course entity with data from the req.body, returning it with status 201 or status 400 if an error occurs. The GET / route reads all Course entities from the database and returns them in JSON, or sends status 400 if there is a problem. The GET /:id route reads one Course entity by id, returning it if found or sending a 404 status if not found, as well as a 400 status if an error occurs. The PUT /:id route updates the Course entity by id with the data from the req.body, returning the updated data if successful or a 404 status if not found, as well as a 400 status if an error occurred. The DELETE /:id route deletes the Courses entity by id, returning status 204 if successful or status 404 if not found, as well as status 400 if an error occurs. Each operation is wrapped in a try-catch block to handle errors and send the appropriate response.
- penilaian.js
  The penilaian.js module uses Express to manage CRUD operations on the Assessment model. The POST route / creates a new entity using Assessment.create(req.body), returning data with status 201 on success, or status 400 with an error message on failure. The GET / route retrieves all Assessment entities from the database and returns them in JSON, or sends status 400 if there is a problem. The GET /:id route reads one Assessment entity by id, returning it if found or a 404 status with the message "Assessment not found" otherwise, as well as a 400 status if an error occurs. The PUT

/:id route updates an Assessment entity by id, returning the updated data if successful, or a 404 status with the message "Assessment not found" if not found, as well as a 400 status if an error occurred. The DELETE /:id route deletes the Assessment entity by id, sending status 204 if successful, or status 404 with the message "Assessment not found" if not found, as well as status 400 if an error occurs. Each operation is wrapped in a try-catch block to handle errors and send the appropriate response.

4.4. Performance Testing

Performance testing is the process of evaluating the ability of a system or application to handle an expected workload. The main objective of performance testing is to ensure that the application is capable of providing fast and reliable services to end users even under peak load conditions. In this test, we will focus on four key metrics that are critical to understanding system performance: Total Requests, Throughput, Response Time, and Error Rate.



Test setup

Virtual users 10 VU	Start time Jun 21, 13:37:32 (GMT+7)	Load profile Fixed
Duration 1 minute	End time Jun 21, 13:38:42 (GMT+7)	Environment -

1. Summary

Total requests sent	Throughput	Average response time	Error rate
1,510	21.55 requests/second	34 ms	0.00 %

Figure 6. Performance Testing

1. Test Set up

   This test was conducted using 10 virtual users. The test started on June 21 at 13:37:32 (GMT+7) and lasted for 1 minute, ending at 13:38:42 (GMT+7). The load profile used was "Fixed", which means that the number of virtual users was fixed throughout the test. Information about the test environment was not mentioned (marked with a minus sign "-").
2. Total requests sent

   The total number of requests sent during the test was 1,510. This shows how many interactions were tested on the system or application within the specified test duration (1 minute).
3. Throughput

   Throughput describes the rate of requests that the system can handle per second. In this case, there were an average of 21.55 requests/second: requests successfully processed by the system every second during the test.
4. Average response time

   The average time taken by the system to respond to each request. The average response time is 34 milliseconds, which indicates that the system responds very quickly.
5. Error rate

   Percentage of errors during the test. A value of 0.00% means that no requests failed or experienced errors during the test. All requests sent were successfully processed by the system.

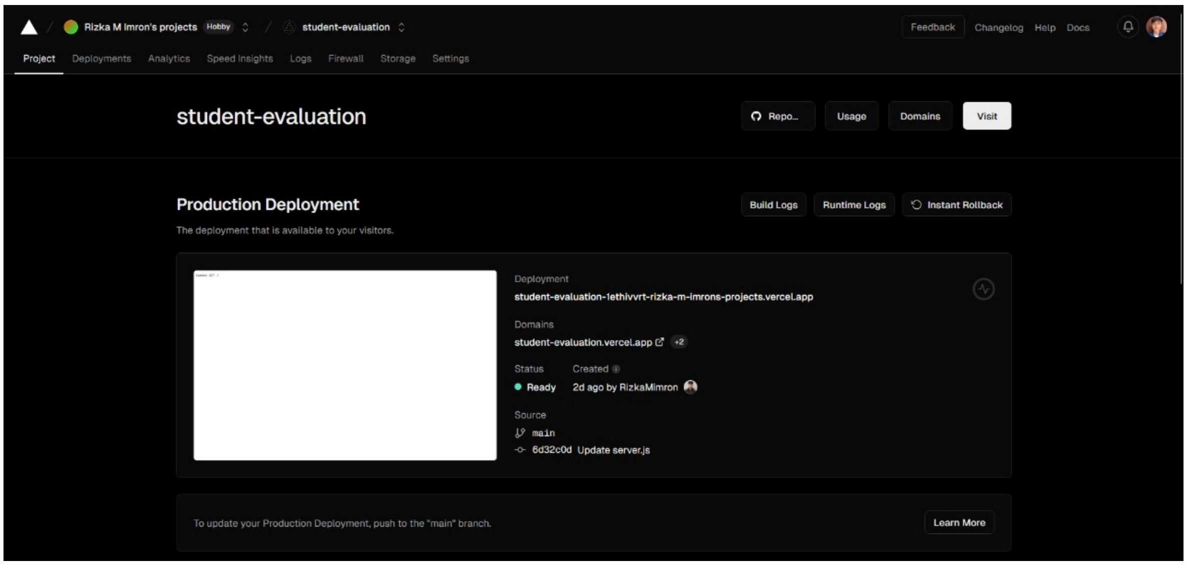
It can be concluded that the tested system has excellent performance. The system is able to handle a large number of requests, namely 1,510, in a very short time, namely 1 minute. The high throughput, at 21.55 requests per second, shows that the system can process requests quickly and efficiently. The very low average response time of 34 milliseconds shows that the system responds almost instantly. The zero error rate (0.00%) indicates that the system functions flawlessly without any failed requests. Overall, this summary shows that the tested system can handle the user load very well, responds to requests quickly, and did not experience any errors during the test.

4.5. Deployment

This stage is the process of deploying the tested software to a production environment, where users can start using it. This can include installation, configuration, and other settings required to run the software.

This stage is the process of uploading the application to web hosting so that users can access it via the internet. The stage of handing over the software to the users for use according to the system analysis and design is known as deploying the application. This process includes uploading the application from the local side to the hosting by using Git version control and SSH, as well as additional settings required to run the program.

Deployment is the stage following development where an application is implemented into the actual operational environment. This process involves placing the application or system into the production or operational environment, which typically differs from the testing environment.



The deployment that is available to your visitors.

Deployment
student-evaluation-1ethivvrt-rizka-m-imrons-projects.vercel.app

Domains
student-evaluation.vercel.app +2

Status: Ready
Created: 2d ago by RizkaMimron

Source
main
6d32c0d Update server.js

To update your Production Deployment, push to the "main" branch.

Figure 7. Deployment

Deployment of the Student Assessment application named "Student Evaluation" to Vercel has been successfully carried out. The application is now accessible via the URL student-evaluation.vercel.app2. Important information from this deployment includes the deployment name "student-evaluation," source "main," and commit message "Update server.js." Users can monitor application performance through analytics, speed insights, and logs. In addition, firewall, storage, and application configuration settings can also be accessed through Vercel.

Please note that this deployed system does not have a front-end interface, but only covers the back-end. It is hoped that the "Student Evaluation" application can run smoothly and provide significant benefits to its users.

4.6. Discussion

The development of the Student Performance Evaluation System discussed in this research demonstrates significant potential for enhancing effectiveness and oversight in higher education. By employing a distributed system approach and Agile methodology, the system effectively addresses various challenges faced by educators in assessing student performance. Implementing database replication techniques for data synchronization is also crucial in ensuring data integrity and consistency within the system.

One of the system's main strengths lies in its comprehensive features, including registration, assessment form creation, grading, statistical analysis, and assessment history access. These features not only streamline the assessment process but also enhance transparency. Administrators have the capability to register and manage faculty, create assessment forms, and view student assessment statistics. Meanwhile, faculty members can fill out assessment forms, view statistics, and evaluate student performance. However, it's important to note that students do not have a direct role in this system, which may be an area for future development.

Performance testing results indicate that the system achieves high throughput of 21.55 requests per second, with an average response time of 34 milliseconds and an error rate of 0.00%. These metrics highlight the system's efficiency and reliability in handling large volumes of requests error-free. Its near-instantaneous performance under user load suggests readiness for real-world implementation in higher education environments.

Nevertheless, there are aspects worthy of further consideration. For instance, while the system does not currently involve students directly, there may be benefits in actively engaging them in the assessment process, such as providing feedback or accessing their assessment results directly. Additionally, further research could evaluate the system's impact on educational quality and user satisfaction among both faculty and students.

Overall, this research makes a significant contribution to the field of student performance assessment, demonstrating that with the right approach, an efficient and transparent assessment system can be successfully developed and implemented.

5. Conclusions

This research has successfully developed a Student Performance Evaluation System aimed at enhancing effectiveness and oversight in higher education. By utilizing a distributed system approach and Agile methodology, the system effectively addresses challenges faced by educators in evaluating student performance. Implementation of database replication techniques ensures efficient data synchronization.

The system is equipped with various features such as registration, assessment form creation, grading, statistical analysis, and assessment history access, all designed to improve efficiency and transparency in the assessment process. Performance testing indicates that the system achieves a high throughput of 21.55 requests per second, with an average response time of 34 milliseconds and an error rate of 0.00%, demonstrating its ability to handle large volumes of requests quickly and efficiently without errors.

The article also outlines the workflow and system development process, detailing the roles of Administrators, Students, and Faculty, as well as the backend development model and routes used. Administrators can register and manage faculty, create assessment forms, and view student assessment statistics, while faculty members can fill out assessment forms, view statistics, and evaluate student performance. Students do not have a direct role in this system.

Overall, the system exhibits excellent performance and reliability in managing user loads and responding almost instantly to requests, thereby expected to enhance student performance assessment processes in higher education environments.

Acknowledgments: We would like to thank the Department of Informatics Engineering of UIN Sunan Gunung Djati Bandung for supporting this research. We would also like to thank our advisor Dr. Wisnu Uriawan S.T., M.Kom, for his invaluable guidance, valuable inputs, and constant encouragement. Without his expertise and patience, this work would not have been possible. In addition, our gratitude goes to our family and friends for their tireless support and understanding during this journey. Their confidence is a source of strength and motivation for us. Finally, we would like to thank all the team members who have worked together with dedication and passion to complete this research. Our unity and cooperation have enriched this experience tremendously.

References

1. Murphy, K.R.; Cleveland, J.N. *Understanding performance appraisal: Social, organizational, and goal-based perspectives*; Sage, 1995.
2. Aljenaa, E.; Al-Anzi, F.; Alshayeji, M. Towards an efficient e-learning system based on cloud computing. Proceedings of the Second Kuwait Conference on e-Services and e-Systems, 2011, pp. 1–7.
3. Fadhilah, A.M.I.; Nurdiawan, O.; Basyisyar, F.M. PENGEMBANGAN SISTEM INFORMASI BERBASIS WEB SMART CONTRACT PADA BLOCKCHAIN BERBASIS NFT. *JATI (Jurnal Mahasiswa Teknik Informatika)* **2023**.
4. Rifan, A. Sistem Monitoring Program Kerja untuk Mengevaluasi Kinerja Mahasiswa pada Organisasi Kerukunan Mahasiswa Pinrang Berbasis Web. *Buletin Sistem Informasi dan Teknologi Islam* **2023**.
5. Pralaya, G.; Setiawan, H. Sistem Penilaian terhadap Kinerja Guru Berbasis Web (Studi Kasus: SMP Negeri 2 Sukodono). *Physical Sciences, Life Science and Engineering* **2024**, *1*, 15–15.
6. Utami, D.; Devi, P.A.R. Sistem Penilaian Kinerja Asisten Praktikum Prodi Teknik Informatika Berbasis Web (Studi Kasus: Universitas Muhammadiyah Gresik). *Media Jurnal Informatika* **2022**, *14*, 55.
7. Kurniawan, M.R.; Nurdiana, D.; Andriyan, A.; Pangestu, D.A.; Putro, S.W.A. SISTEM PENILAIAN KINERJA SALES BERBASIS WEB PADA PT. HARSINDO OETAMA PERKASA. *JATI (Jurnal Mahasiswa Teknik Informatika)* **2023**, *7*, 1041–1046.
8. Wibawa, A. EVALUASI PENERAPAN PENILAIAN KINERJA KERJA PEGAWAI NEGERI SIPIL BERBASIS SISTEM 360 DERAJAT FEEDBACK-APPRAISAL DI KANTOR REGIONAL VIII BKN. *Jurnal Borneo Humaniora* **2022**, *5*, 11–28.
9. Yasa, I.G.W.; Syahrir, M.S.; Azwar, M.; others. Pengembangan Aplikasi Penilaian Kinerja Karyawan Berbasis Sistem Terdistribusi pada CV. Berkat Melimpah dengan Menggunakan Metode Rating Scale. *Journal of Millenial Informatics* **2023**, *1*, 17–26.
10. Simatupang, K.O.; Pakpahan, A.F. Metode Agile Dalam Perancangan Sistem Informasi Reservasi Fasilitas Universitas Advent Indonesia. *Journal of Information System Research (JOSH)* **2022**, *3*, 608–617. https://doi.org/10.47065/josh.v3i4.1816.
11. Suhari, S.; Faqih, A.; Basysyar, F.M. Sistem Informasi Kepegawaian Mengunakan Metode Agile Development di CV. Angkasa Raya. *Jurnal Teknologi dan Informasi* **2022**, *12*, 30–45. doi:10.34010/jati.v12i1.6622.
12. Haile, K.; Mekuria, T.; Fantahun, A.; Belay, A. Avoiding consistency and update latency problem in lazy master-master replication using a coordinator architecture. 2017 IEEE AFRICON. IEEE, 2017. https://doi.org/10.1109/afrcon.2017.8095607.
13. Ardiansyah, S.; Setiorini, A.; Atrinawati, L.H.; Fiqar, T.P. Perancangan Arsitektur Sistem dan Teknologi Informasi Menggunakan Togaf ADM (Studi Kasus Dinas Perhubungan Kota Balikpapan). *MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer* **2019**, *19*, 70–79. doi:10.30812/matrik.v19i1.481.
14. Togatorop, P.R.; Simanjuntak, R.P.; Manurung, S.B.; Silalahi, M.C. PEMBANGKIT ENTITY RELATIONSHIP DIAGRAM DARI SPESIFIKASI KEBUTUHAN MENGGUNAKAN NATURAL LANGUAGE PROCESSING UNTUK BAHASA INDONESIA. *Jurnal Komputer dan Informatika* **2021**, *9*, 196–206. https://doi.org/10.35508/jicon.v9i2.5051.