An Implementation of a Web-based Application with Personalized Learning Path and Seamless Single Sign-On for Self-paced Learning

Akira
Faculty of Computer Science
President University
Bekasi, West Java
akira@student.president.ac.id

Cutifa Safitri
Faculty of Computer Science
President University
Bekasi, West Java
cutifa@president.ac.id

Ronny Juwono
Faculty of Computer Science
President University
Bekasi, West Java
ronny@president.ac.id

Abstract—Innovation is what enables humans to thrive. History will repeat itself without innovation, and humanity’s history is far from ideal. One thing humans do have is the choice of a bright future. It is up to us, the future leaders, to earn and achieve this future. This is why education is so important. Education brings more innovations and wisdom to the world. As prospective leaders, it is a must to do whatever it takes to make education more accessible, enjoyable, exciting, and engaging, allowing students to feel the joy of learning as we do. Our research explores methods for the system design of an e-learning platform, focusing on integrating personalized learning path management and the seamless Single Sign-On authentication system. This paper contributes to the advancement of the field of educational technology by proposing a solution aimed at revolutionizing the learning experience. By proposing personalized learning paths and the integration of a seamless Single Sign-On (SSO) system, this paper addresses the critical challenges surrounding conventional educational approaches.

Keywords—learning path, Single Sign-On, Web-based application, learning platform, E-Learning, API

I. INTRODUCTION

Let us start by asking a simple question: How to achieve innovation? The answer is simple: by enhancing the quality and accessibility of education. Conventional education methods often employ a one-size-fits-all approach to teaching materials. This approach has several underlying reasons, each with its own set of consequences.

One of the primary reasons is the challenge of managing diverse student needs. When the study materials were generalized, the depth and complexity of the lessons were compromised compared to tailor-made content for specific students with distinct interests. This compromise often arises due to the students’ disinterest in certain subjects, which can lead to lower overall scores. Educational institutions may become concerned about their students’ performance and, in response, simplify the curriculum to ensure higher scores. Another alternative outcome is that students may invest time and effort in studying a subject they have no future use for, potentially wasting valuable resources that could be better spent pursuing their preferred areas of study.

This paper proposes a solution: implementing a personalized learning path, allowing students to choose their studies according to their interests and goals. It is complemented by a seamless Single Sign-On (SSO) system, enabling access to resources from multiple platforms. Hence, this study contributes to developing a Web-based application for a customized learning path management system built upon Moodle’s course structure and an efficient and user-friendly SSO system, achieved through a custom-made remote login API, connecting both Moodle and the Web-based application. Essentially, this turns Moodle into our Web-based application’s authenticator. This will be discussed more in detail in the next chapter.

While numerous researchers have demonstrated the potential benefits of personalized learning paths and SSO systems in education, their implementation in our current society still needs to be improved. Consequently, students need to take advantage of opportunities for engaging learning environments. Establishing clear learning paths, objectives, and educational goals can drive positive changes, inspire innovation, and improve effectiveness. Additionally, implementing a seamless SSO system streamlines access to educational resources, enhancing user experiences.

II. LITERATURE REVIEW

A. Learning Platform Comparison

A learning platform, sometimes called a Learning Management System (LMS), refers to a digital environment or software application that facilitates the delivery, management, and organization of educational resources and activities. This paper will compare two popular learning platforms, Moodle and Google Classroom.

Moodle is designed to support online teaching and learning, creating and managing online courses, aligning objectives with the curriculum, automated grading, creating discussion forums, providing feedback, and customizable quizzes, assignments, and exams [1]. Moodle also has a rich plugin library supported by developers worldwide. The most significant advantage of Moodle lies in its customization options and open-source nature, empowering educators to personalize the interface, course design, and settings. However, Moodle has drawbacks and limitations, including technical expertise requirements, and complexity in implementing advanced customizations.

Google Classroom offers a range of features, including assignment management, communication tools, content sharing, storage, collaboration capabilities, streamlined
interface, and integration with other Google apps [2]. The biggest downside, however, is the lack of customization available, hence researchers and other educational institutions cannot freely modify it to suit their specific needs.

Moodle was chosen as this study’s learning platform for several reasons. Being an open-source offers extensive customization flexibility, allowing the learning environment to be tailored to the specific research needs. Additionally, Moodle benefits from a large research community, providing valuable resources and support throughout the study.

B. Software and Framework
Creating a robust and efficient web application requires several pieces of software. The study will be using popular softwares such as: Node.js [3], MariaDB [4], and MongoDB [5].

Frameworks are valuable tools in software development that provide structure and efficiency by offering pre-built tools, libraries, and patterns. The study will be using one of the most popular backend frameworks, Express [6], for the study’s requirements. Frameworks save developers time and effort by eliminating the need to write repetitive code and handle everyday tasks from scratch.

C. API
APIs, or Application Programming Interfaces, enable communication and integration between software systems [7]. This study will utilize both Moodle Core API and Moodle Custom API to access the extensive features of the Moodle platform.

1) Core API: The Moodle Core API allows us to seamlessly integrate our application with Moodle’s core functionality [8]. These APIs provide reliability, stability, extensibility, comprehensive documentation, and community support.

2) Custom API: The Moodle Custom API enables us to extend Moodle’s capabilities and customize it for our research objectives [9].

D. Plugins
Plugins expand software applications’ functionality by adding extra features and capabilities, allowing us to extend Moodle’s core features through integration [10]. Moodle plugins provide modular and customizable solutions, enabling us to tailor the platform to our study’s requirements and enhance the learning experience for users. Unmentioned plugins are installed by default.

1) Edwiser course: Edwiser course formats in Moodle offer customizable layouts and design options, improving the visual appeal and user experience of courses and promoting learner engagement and motivation, facilitates progress tracking and completion tracking, enhancing assessment and monitoring of student progress while promoting learner accountability [11].

2) Moodle Adminer: Moodle Adminer grants administrators direct access to the Moodle database, simplifying advanced database management tasks, providing greater control and visibility, and ensuring data integrity and security [12].

III. RESEARCH METHODOLOGY

A. Components
The system has two main components: Moodle, a learning platform with user management and content storage functionalities, and a self-built Web-based application.

1) Moodle: The Moodle component includes the Moodle API, which allows communication between the Web-based application and Moodle for data retrieval. It also utilizes core functions such as the user, course management, and enrollment systems.

2) Web: The Web-based application, referred as the Web, interacts with Moodle to retrieve user information and course data. It uses a MongoDB database for storing its data. The Web component includes features such as user authentication, a learning path system built upon the courses, and data retrieval from Moodle using the Moodle API. It utilizes the MongoDB database for storing user-specific data.

B. Single Sign-On

The main reason for implementing Single Sign-On (SSO) in this study is users’ frequent switching between Moodle and the Web. Requiring users to log in twice would be a significant inconvenience and could result in a disjointed user experience, lacking seamless integration. While other SSO solutions are widely used, such as LDAP and OAuth 2.0 [13], the decision to create a new solution tailored to this study’s unique requirements was made.

Implementing and designing the custom-made API eliminates the need to rely on third-party services, removing the complexities of maintaining LDAP directory structures and managing OAuth configurations. This results in greater control over the authentication process, leading to a more streamlined and efficient solution. Furthermore, the custom SSO solution lets us adapt and customize the authentication process precisely to our needs. This flexibility allows us to implement features and functionality that perfectly fit our study, enhancing its overall effectiveness and user experience.

The proposed SSO process follows a straightforward approach. Instead of involving a third-party authentication service, users from the Web-based application are using Moodle for logging in, utilizing Remote Login, a custom made API that creates both Moodle Session and Token data. By logging in to Moodle and being redirected back to the Web with Token data, users achieve the same result as Single Sign-On, as their authentication status is synchronized between the Web and Moodle. This approach simplifies authentication and ensures a seamless user experience across both platforms. The Token data will be used for API retrieving performed on the Web.

This method transforms Moodle into the authenticator for the Web. Do note, the proposed authenticator does not limit only to the Web. It differs from the typical token retrieval API method as it does not create Moodle Session, only giving a
valid token. This is why developing a custom API is necessary for this study.

Additionally, our method leverages the Check Logged-In API to automatically log in users who already have an active session in Moodle. One-way Single Sign-Off feature is also implemented. Logging out from the Web will destroy both Moodle Session and Web Session, but logging out from Moodle will only destroy Moodle Session. The mechanism is straightforward: using a custom API Remote Logout. This API allows clients to destroy Moodle Session remotely.

C. Learning Path

As for the learning path, the Path, Competency, and Task system will be made. Each Path consists of one or more Competencies, every Competency comprises one or more Tasks, and each Task corresponds to a single Moodle module.

Fig. 1. Moodle Single Sign-On using Custom API

Once students pick their path, the system automatically enroll them into each of the learning path’s module’s courses, making a clear progression for students to follow. Automatically enrolling students into the corresponding module courses based on their chosen learning path streamlines the process and ensures that they have access to the relevant materials. This automation can save time and effort for both students and administrators, as it eliminates the need for manual enrollment. The proposed system aims to provide a structured and efficient way for students to navigate their learning journey while leveraging Moodle’s existing resources and modules.

D. Experimental Setup

These are the steps to set up the experiment.

1) Set up Moodle: download the Moodle 3.9 package from the official Moodle website and extract it to the desired web server directory. After completing the installation, a functioning Moodle will be ready for use.

2) Set up Edwiser Course Format Plugin: Download the compatible version of the Edwiser Course Formats plugin. Extract the package and install the Edwiser Course Formats plugin as an Admin. Once established, enable and configure the plugin according to the study’s preferences.

3) Upload Students: Upload users as a CSV file as an Administrator at the Site Administration menu to import the students into Moodle.

4) Upload classes: Create desired categories for the courses. Upload courses as a CSV file as an Administrator at the Site Administration menu. To create new courses within the relevant category, provide the course details or upload a CSV containing the required information.

5) Prepare for clone: Within a specific class, create modules such as Assignment 1, Lesson 1, and Quiz 1 with their respective details and instructions. After creating these components in one class, clone them to the other desired classes.

Following these steps will set up Moodle, install the Edwiser Course Formats Plugin, upload students, create classes, and clone activities, allowing for efficient management and organization of the learning environment.

IV. RESULTS

The evaluation demonstrates the system’s effectiveness in meeting objectives, offering strong performance, reliable functionality, and favorable usability. The findings provide valuable feedback for further improvements to enhance performance, expand functionality, and improve the user experience. Please note that the tests will be started with no session in both the WEB and MOODLE in default. Additionally, having a Moodle Session implies being logged into Moodle, while having a Web Session implies being logged into the Web.

A. Forgot Password on Web

In the Forgot Password process on the Web platform, users can reset their passwords securely. The process begins when the CLIENT sends a GET request to WEB ‘/forgot’. Then, the WEB redirects the CLIENT to MOODLE’s ‘/forgot_password.php’. Next, the CLIENT sends a GET request to MOODLE’s ‘/forgot_password.php’. After that, MOODLE send the Forget Password Form Page to CLIENT. The process continues with the CLIENT fills in their username or email address and send POST data to MOODLE. Finally, MOODLE sends Change Password link to the CLIENT. The result shows that users successfully go to MOODLE’s Forget Password Form Page and email function work correctly.

B. Login with the wrong password on the Web

In the Invalid Login process on the Web platform, users will not be granted permission to access. The process begins when the CLIENT got Login Page from the WEB. The CLIENT then sends POST invalid login data to the WEB’s ‘/redirect’. Next, the WEB redirects the CLIENT with the login data to MOODLE’s ‘/remotelogin.php’. Then, MOODLE validates login data, and since the login data is
invalid, MOODLE sends POST login error message to the CLIENT, redirecting the CLIENT to the WEB’s `/login`. Since the WEB doesn’t receive Token data from the redirect, the WEB sends a Login Page with a login error message to the CLIENT. The result shows that an error message displayed in the WEB’s Login Page, displaying “Invalid login, please try again” on the page, indicating that the login attempt failed as intended due to an incorrect password.

C. Login into Web for the first time.

In this New Login to Web process using a new account uploaded from Moodle, which never login to Moodle and forced to change password, users will be redirected to Moodle to change password and redirected back to Web after the users finished changing their passwords. The process begins with the CLIENT got Login Page from the WEB. The CLIENT then sends POST-first timer login data to the WEB’s `/redirect`. Then, the WEB redirects the CLIENT with login data to MOODLE’s `/remotelogin.php`. Next, MOODLE validates login data. While MOODLE validates the login data, MOODLE detects a new user, creates a Moodle Session, and redirects the CLIENT to the WEB’s `/login` with POST Token Data. The WEB receives Token data, creates Web Session, and gives access to the CLIENT, and redirects CLIENT to choose their learning path. The result shows that the WEB successfully redirects users to change their password on MOODLE and back to the WEB. The Learning Path Choice process will be explained in detail later.

D. Login as an Old-Time User on Web

In this Login as an Old-Time User process on the Web platform, users will be granted access to the Web. The process begins when the CLIENT got Login Page from the WEB. The CLIENT then sends valid login data to the WEB’s `/redirect` using POST method. Next, the WEB redirects the CLIENT with the login data to MOODLE’s `/remotelogin.php`. After that, MOODLE validates login data, and since the login data is valid, MOODLE creates a Moodle Session and sends POST Token data for CLIENT, redirecting the CLIENT to the WEB’s `/login`. The WEB then receives the Token data, creates Web Session, and gives access to the CLIENT. The result shows that users can log in with existing credentials and redirected to the Home Page of the Web successfully.

E. Login as an Unexisting User on Web

In this Login as an An Unexisting User process on the Web platform, users will not be granted permission to access. The process begins when the CLIENT receives the Login Page from the WEB. The CLIENT then sends POST data of a non-existent user login credentials to the WEB’s `/redirect`. After that, the WEB redirects the CLIENT with login data to MOODLE `/remotelogin.php`. Next, MOODLE validates login data, and since the login data is invalid, MOODLE sends a POST login error message for the CLIENT to redirect to WEB `/login`. The WEB doesn’t receive Token data and receives the error message, sends Login Page displaying the login error message to the CLIENT. The result shows that users with non-existent login credentials are correctly identified as invalid, by displaying “Invalid login, please try again” error message on the Login Page.

F. Login to Moodle First, then go to Web

In this Login to Moodle and Visit Web process, users will demonstrate the Single Sign-On capabilities by logging into Moodle and visits the Web. The process begins when the CLIENT receives MOODLE’s Login Page. The CLIENT then sends POST login data to MOODLE `/index.php` to validate the login credentials. Since the login credentials are valid, MOODLE creates Moodle Session and redirects the CLIENT to MOODLE’s Home Page, completing the Login to Moodle process. Now let us proceed to the Visit Web process.

The CLIENT sends GET request to the WEB and it will checks if a user is logged in in the WEB. The CLIENT has no Web Session (is not logged in), and the WEB redirects CLIENT to MOODLE’s `/checkloggedin.php`. MOODLE then checks if the CLIENT has a Moodle Session, and since the CLIENT has Moodle Session, MOODLE sends a POST Token data and redirects CLIENT to WEB `/login`. Next, The WEB checks if the request has Token data. Since the WEB receives the Token data, it creates a Web Session and gives access to the CLIENT. The result shows that users experience seamless access to the Web system without requiring additional authentication after logging into MOODLE.

G. Login to Web First, then go to Moodle

In this Login to the Web and Visit Moodle process, users will demonstrate the Single Sign-On capabilities by logging into the Web and visits Moodle. The process begins when the CLIENT got Login Page from the WEB. The CLIENT sends POST login data to WEB `/redirect`. The WEB then redirects CLIENT with login data to MOODLE `/remotelogin.php`. MOODLE validates login data. Since the login data is valid, MOODLE creates Moodle Session, sends POST Token data for CLIENT, and redirect CLIENT to the WEB’s `/login`. The WEB receives Token data and creates Web Session, giving access to the CLIENT and completing the Login to the Web process. Now let us proceed to the Visit Moodle process.

The CLIENT sends a GET request to MOODLE. Since the CLIENT already has a Moodle Session, the CLIENT already logged in and has access. The result shows that users experience seamless access to Moodle without requiring additional authentication after logging into the WEB.

H. Logout Moodle First, then go to the Web

In this Logout from Moodle and Visit Web process, users will demonstrate that the one directional Single Sign-Off does not destroy the Web Session when they log out from Moodle. In this test, the CLIENT has both a Moodle Session and a Web Session. The process begins when the CLIENT sends a GET request to `/logout.php` with sesskey data, logs out from MOODLE, and destroys Moodle Session. The CLIENT can now go to the Web by sending a GET request to the WEB. The WEB will checks if a user is still logged in. The user is still logged in and the CLIENT still has the Web Session. The result shows that users can still access the Web as expected after logging out from Moodle.
I. Logout Web First, then go to Moodle

In this Logout from Web and Visit Moodle process, users will demonstrate that the one directional Single Sign-Off will work, destroying the Moodle Session when they log out from the Web. In this test, the CLIENT has both a Moodle Session and a Web Session. The process begins when the CLIENT sends a GET request to the WEB’s ‘/logout’. The WEB destroys Web Session and redirects the CLIENT to MOODLE ‘/remotelogout.php’. MOODLE then destroys the Moodle Session. The CLIENT can now go to MOODLE by sending a GET request to MOODLE. MOODLE then checks for the CLIENT’s Moodle Session. Since the CLIENT has no Moodle Session, the CLIENT is redirected to MOODLE’s Login Page. The result shows that users are correctly prompted to provide login credentials when accessing MOODLE after logging out from the Web. The one-way Single Sign-Off works.

J. Upload Path, Competency, and Task Data as Admin on Web

In this Upload Path, Competency, and Task Data process performed on the Web platform as an Administrator, users will demonstrate the data processing capabilities in creating the Path, Competency, and Task system using the Upload functionality. The process begins when the CLIENT logs in as Administrator. Then, the CLIENT sends a GET request to WEB ‘/upload’. The WEB will check if user has the role of Administrator. Since CLIENT is logged in as an Administrator, the WEB response with the Upload Page.

In the next process will be the test of Upload functionality. First, handle the Task. The CLIENT uploads the “Task.csv” file and processes the data, preparing and sending the POST request to WEB ‘/upload’. The WEB then checks if data is valid. Since the data is valid, the WEB updates the database and sends a successful response back to the CLIENT. The CLIENT page shows updated data status. Repeat for Competency and Path. The result shows that users with the Administrator role can successfully upload Path, Competency, and Task data.

K. Create Path, Competency, and Task Manually on Web

In this Create Path, Competency, and Task Data Manually process performed on the Web platform as an Administrator, users will demonstrate the data processing capabilities in creating the Path, Competency, and Task system using the Create functionality. The process begins when the CLIENT logs in as Administrator. Then, the CLIENT sends a GET request to WEB ‘/manual’. The WEB will check if user has the role of Administrator. Since CLIENT is logged in as an Administrator, the WEB response with the Create Page.

The next process will be testing the Create functionality. First, process the Task. The CLIENT fills in the form. The CLIENT then process the POST request and sends the Path data to WEB ‘/manual’. Next, the WEB checks if data is valid. Since the data is valid, the WEB updates the database and responds with a successful response back to the CLIENT. The CLIENT page shows updated data status. Repeat for Competency and Path. The result shows that users with the Administrator role can successfully create Path, Competency, and Task data manually.

L. Select Path for New Users on Web

In this Path Selection for New Users process performed on the Web platform, new users who have not select their learning path will be able to select their learning path. The process begins after the CLIENT performs a login. The WEB then checks if a user is in MongoDB WEB database (different from MariaDB MOODLE database). Since the user is new, the client’s login is not found in the MongoDB database (the learning path of user is not recorded), hence redirected to WEB GET ‘/selectpath’. Next, the WEB responds with Major Page with a list of Majors to the CLIENT. After that, the CLIENT sends the selected Major’s ID to POST WEB ‘/selectpath’. The WEB responds with Learning Path Page with a list of Paths to CLIENT. Then, the CLIENT sends the selected Path’s ID to POST WEB’s ‘/progress’. The WEB processes the data and saves the Student’s Learning Path Selection to MongoDB WEB Database. The WEB then processes the Enrolled Courses Data to the CLIENT. The CLIENT can see all the Enrolled Courses. This results shows that all new users can successfully enroll in a learning path.

M. Unenroll User Path on Web

In this Enenroll Students’ Learning Path process, Administrators will control to unenroll any selected users from the learning path they previously chose. The process begins with the CLIENT logs in as an Administrator in the WEB. The CLIENT sends a GET request to the WEB’s ‘/manageupath’. The WEB will check if the user has the role of Administrator. Since the CLIENT is logged in as an Administrator, the WEB responds with Manage Path Page with list of Paths to the CLIENT. The CLIENT then sends the selected Path’s ID on a GET request to the WEB’s ‘/manageupath/id’ and the WEB will responds with list of Students enrolled in that particular Path to the CLIENT. Next, the CLIENT sends the selected Student’s ID on a GET request to the WEB’s ‘/unenroll/id’. After that, the WEB removes the Student’s record from MongoDB WEB database, and responds with “Completed” to CLIENT. The result shows that Administrators can unenroll specific users from a Learning Path.

N. User Score Something

In this User Score process, users will be able to access Moodle’s Incomplete Modules from the Web platform, finishes them, and see the updated score result in the Web. The lecturer’s grading process will also be included, but note that this is not necessary if the Module student is working on has an automatic scoring system (No essays, only multiple choice). This process begins with the CLIENT logs in as a Student in the WEB. The CLIENT will send a GET request to WEB ‘/dashboard’ and the WEB will responds with list of Incomplete Enrolled Course Modules to the CLIENT. The CLIENT then go to the MOODLE’s Module by the Module’s Link. After that, the MOODLE responds with Module Page, showing the quiz, assignment, or a lesson to the CLIENT. The CLIENT then uploads or finishes an assignment or quiz.
Now let us proceed with the Lecturer’s Grading process. The CLIENT logs in as a Lecturer in MOODLE. The CLIENT goes to the Module’s link. The CLIENT then assigns scores to the Student’s quiz or assignment. The CLIENT logs back in as a Student in the WEB.

The next step is for the CLIENT to send a GET request to the WEB’s /score. The WEB processes the data, retrieves the score data and responds with the Score Page filled with Score Data to the CLIENT. The result shows that users can see the update on assignments, quizzes, and lessons’ scores.

V. DISCUSSION

A. Remarks

Based on the conducted tests, it can be concluded that the Single Sign-On and Authentication system performs effectively and provides a seamless user experience. The system handles various scenarios such as forgotten passwords, first-time user setup, existing user login, and account validation. It also demonstrates the ability to maintain separate login sessions between Moodle and the Web platforms, allowing smooth transitions between them.

The discussion findings highlight the advantages of using Moodle as the Authenticator for Single Sign-On, eliminating the need for third-party authentication systems and simplifying the login process.

Administrators can also upload, create, and manage learning path, competency, and task data within the system, unenroll users, and assign scores or grades. The Path Selection successfully automate the enrolling of students into the corresponding courses based on their chosen learning path. This approach can streamline the enrollment process and provide students with a structured and personalized learning experience.

Overall, the system functions as intended, ensuring secure access and efficient management of user accounts and learning resources.

B. Comparison with Existing System

Several notable differences can be observed when comparing the developed system with existing systems. Firstly, the system offers a more streamlined and user-friendly interface, allowing for more straightforward navigation and accessibility. It provides enhanced features such as personalized learning paths, competency tracking, and automated grading.

There is still room for improvement, particularly in feature completeness and extensive user testing, which some established systems may have already achieved. Overall, the developed system showcases advancements in user experience, innovative features, and the potential for further enhancements to provide a more comprehensive learning management solution compared to existing systems.

C. Limitations and Constraints

The system faces several limitations and constraints that need to be addressed.

1) Privacy and Data Protection: First, privacy and data protection regulations must be followed to ensure user data remains secure. This can be achieved through data encryption, access controls, and anonymization techniques.

2) Integration: Integrating the system with external tools and databases may pose challenges due to differences in data formats and compatibility issues, requiring thorough testing and robust integration strategies.

3) Scalability: Scalability issues may arise as the user base and data grow, but optimizing infrastructure, database management, and load-balancing techniques can mitigate these challenges.

4) The Lack of User Testing: The absence of extensive user testing can result in usability issues and functionality gaps, emphasizing the need for comprehensive testing sessions. Additionally, user feedback and market research should guide the development of new features to meet specific user requirements and expectations.

VI. CONCLUSION AND FUTURE WORKS

In conclusion, the proposed system design has achieved its completeness, performance, and functionality objectives. The validation process has provided valuable insights and highlighted areas for improvement, such as the need for further user testing and the potential integration of machine learning. Overall, this paper contributes to the understanding and development of an efficient and user-friendly system, paving the way for future advancements and enhancements in the field.

Future works focus on the importance of security measures and robust authentication methods to protect user accounts and data. Feedback from users is crucial for identifying areas to enhance the learning experience through new features.

Furthermore, applying machine learning algorithms for student performance analysis can provide personalized recommendations and optimize the learning experience by analyzing data and tailoring content based on individual strengths and weaknesses. These advancements align with research on data-driven approaches to improve learning outcomes and student success.

In conclusion, the identified future works align with existing literature, focusing on improving security, expanding features, automating processes, leveraging machine learning, and incorporating feedback to enhance the system’s effectiveness and usability in the educational domain. The plan aligns with current research and best practices by incorporating these advancements.

REFERENCES


