Journal of Computing and Security

Research Article, 2025, Volume 12, Number 2 (pp. 1–23)

http://www.jcomsec.org

A Low-Code Approach for Developing Customizable Teacher Performance Analysis Dashboards for Moodle

Amirhossein Badiei [a] Mansooreh Ezhei [a] Mohammadreza Sharbaf [a,*]

[a] *Department of Software Engineering, University of Isfahan, Isfahan, Iran.*

ARTICLE INFO.

Article history:
Received: 04 April 2025
Revised: 05 July 2025
Accepted: 19 July 2025
Published Online: 03 September 2025

Keywords:
Learning Management System, Moodle LMS, Teacher Performance Analysis Dashboard, Low-Code Development Platform, Educational Quality Improvement.

ABSTRACT

Nowadays, analyzing teacher performance is crucial to improving instruction quality. Evaluating teachers can be effective in designing and delivering a course, managing the class, managing time, and supporting learners. Continuous teacher evaluation helps maintain high educational standards by ensuring that only qualified educators are recruited. In the current educational landscape, learning management systems (LMS) provide an environment for interaction and quality enhancement. Dashboards designed to analyze teacher performance in educational institutions can significantly aid in generating supervisory reports on course quality and improving teaching effectiveness. Additionally, various evaluation criteria—which may differ across institutions—can support these assessments. However, developing performance analysis dashboards for teachers is complex, requiring substantial time and financial investment. Existing research on such dashboards does not fully address the need for diverse dashboard types, highlighting the demand for a new solution. This study employs a low-code development approach to create a customizable platform for teacher performance analysis dashboards in Moodle LMS, tailored to specific evaluation criteria. The platform categorizes teacher performance indicators using a feature model, streamlining dashboard development and reducing time and cost. Its modular architecture enables rapid assembly of dashboard components, while the feature model allows dynamic selection and configuration of relevant metrics. The study was evaluated in two phases. First, three case studies across different subjects were analyzed. Then, usability testing was conducted via an online workshop and questionnaire completed by school administrators and LMS experts. Results demonstrate faster dashboard development and high user satisfaction, confirming the platform's effectiveness.

1 Introduction

In recent years, the adoption of learning management systems has increased, and the number of users is also on the rise. In 2024, the estimated number of users utilizing learning management systems is predicted to reach 73.8 million. Additionally, it is projected that

* Corresponding author.
Email address: m.sharbaf@eng.ui.ac.ir (M. Sharbaf)
https://dx.doi.org/10.22108/jcs.2025.144824.1164
ISSN: 2322-4460

the financial market for learning management systems will grow from $16.2 billion in the period between 2022 and 2023 to $22.4 billion. It is expected that by the end of 2025, this market will increase to $28.1 billion [1]. Therefore, learning management systems continue to grow, with the number of users and market value increasing. Over the past four decades, educational institutions and higher education organizations have come to realize the significant importance of evaluating teacher performance and enhancing the quality of teaching. Additionally, teacher evaluations can be effective in course design and delivery, classroom management, time management, and learner support [2]. This surge underscores the growing demand for customizable analytics dashboards that can evaluate teacher performance and inform teaching improvements.

There is a significant diversity in the components and teaching methods of teachers, and each institution employs different criteria to evaluate their performance [3]. Therefore, there is a need to create customizable analysis dashboards for evaluating teachers' performance. The use of these dashboards enables teachers and institutions to monitor educational courses and allows them to make informed decisions to improve classroom performance and teaching methods using data analysis techniques [4]. These types of learning analysis dashboards are useful for displaying teachers' performance within the learning management system. Additionally, this tool aids in the rapid understanding of information by generating reports and extracting large volumes of data to find relationships among them [5].

Despite the efforts made to design dashboards, research in the field of analyzing these analytical dashboards has been very limited, and analytical dashboards aimed at improving teacher performance evaluation have received less attention. On the other hand, developing analytical dashboards is not straightforward and, in addition to familiarity with teaching concepts and performance evaluation criteria, requires programming knowledge as well as significant time and financial investment, which many educational institutions lack. Therefore, finding a method to accelerate dashboard design would be greatly beneficial [6].

This study introduces a low-code development platform [7] that enables school administrators to automatically generate customized analytical dashboards based on their needs for the Moodle learning management system. This platform, utilizing an intuitive user interface, customizable dashboard features, and requiring no knowledge of designing Moodle learning management system plugins, increases the speed of dashboard production and reduces construction costs.

The proposed approach was evaluated in two phases. In the first phase, three case studies assessed error reduction and platform efficiency to demonstrate its applicability when users develop a targeted analytical dashboard using the proposed platform. In the second phase, the usability of the developed platform was assessed to evaluate ease of use and user satisfaction through an online workshop and a questionnaire completed by school administrators and individuals familiar with the learning management system. The evaluation results indicate user satisfaction and demonstrate significant improvements in the teacher's analytical dashboard development speed, customization flexibility, and user experience.

The rest of this paper is organized as follows. Section 2, introduces the related concepts to provide a deeper understanding of the problem and solution. Section 3, reviews the related studies and discusses the position of this research. The proposed solution is presented in Section 4 and evaluated in Section 5. Finally, Section 6, provides a discussion on the results, and Section 7, concludes the paper and outlines directions for future work.

2 Background

Key concepts for better understanding the proposed approach for evaluating teachers through a customizable platform include Learning Management Systems, Key Performance Indicators (KPIs), Teacher Evaluation Criteria, Model-Driven Engineering, and the Low-Code Development Approach. These concepts are explained in detail in the following sections.

2.1 Learning Management Systems

Learning management systems provide online classrooms for teachers and learners and facilitate learning processes. These systems create an environment for interaction and learner success, allowing learners to enroll in classes, track grades, and view course announcements and notifications [8]. Most learning management systems have features such as course organization, content organization, assignment definition, news posting, as well as facilitating communication and discussion among participants in the course [9]. The Moodle learning management system is one of the most popular and widely used tools for e-learning, providing teachers with the ability to create online courses that allow dynamic interaction with learners. It also simplifies the creation of educational courses, the management of assessments, and participation in forums. The system's user interface is template-based, and utilizing various templates makes the application easier to use. In this type of user interface, predesigned, ready-made templates exist, speeding up the design of the desired page. Additionally, Moodle

employs a role-based mechanism for user allocation, with three primary roles: administrator, teacher, and learner. An individual in the teacher role can also hold the administrator role. The administrator can create and manage courses or assign the teacher role to the relevant user. Furthermore, the addition of plugins is also possible. Plugins help users add new features and capabilities to the system. This flexibility encourages the participation of developers in adding more functionalities to the learning management system [10]. Given that the Moodle learning management system has 213 million users [11] and ranks among the top learning management systems [12], this study utilizes the Moodle learning management system.

2.2 Key Performance Indicators

Key Performance Indicators (KPIs) serve as measurable values to determine the effectiveness of teachers' activities. These indicators are essential for calculating and enhancing the quality of higher education institutions and determining how to achieve key objectives. Institutions use KPIs to ensure that education is on the right track [13]. Additionally, these indicators are used as criteria for measuring teaching quality, and selecting appropriate criteria impacts the usability of the conducted analyses and the success of institutions or teachers [14].

2.3 Evaluation Criteria For Teachers

The criteria and standards used for evaluating teachers include teaching skills, learning environment, classroom management, teaching tools, and learner outcomes. Each evaluation criterion is linked to relevant key performance indicators. To ensure learner progress, teaching tools are utilized. Learning resources, such as slides or discussion forums, are some of the teaching tools in learning management systems. Given that learners' emotions play a role in their learning and teacher behavior affects learner performance, attention to the learning environment, classroom management, and learner outcomes is crucial in evaluating teachers' performance [15].

2.4 Model-Driven Software Engineering

Model-driven engineering is a software development approach that uses models as the primary artifacts. In this method, software is developed based on modeling, and users define the desired system's features in the models. The goal of this approach is to increase quality and productivity by automating the software development stages using modeling. Modeling provides less detail and a higher level of abstraction compared to code written in programming languages. After creating the model, the code is generated automatically. Therefore, human errors are reduced, and software development speed is increased. Another feature of model-driven engineering is the ability to reuse defined models, which reduces production time and improves the maintenance process [16]

2.5 Low-Code Development

The low-code development approach is one of the software development methods that automates the application production process using a graphical interface and drag-and-drop functionality, eliminating reliance on traditional application development methods. The goal of this method is to raise the level of abstraction for application production, reducing or eliminating the need for traditional coding. For example, if an application requires specific programming knowledge, the choice of developer is limited; however, by removing this limitation with the low-code development approach, the application development lifecycle is shortened, allowing more work to be completed in less time [17]. Other features of low-code development platforms include data modeling, use of a graphical user interface, usability by non-experts, support for collaborative development, integration with external services, reusability, scalability, deployment support, cloud computing support, and maintenance and monitoring [7], [18].

3 Related Works

The most relevant studies related to the present research are divided into three categories: student performance analysis dashboards, teacher performance analysis dashboards, and the use of model-driven or low-code approaches for dashboard development.

3.1 Related Research On The Development Of Student Performance Analysis Dashboards

In this section, four studies conducted on the development of student performance analysis dashboards are described.

Kaliisa and Dolonen [19] developed a learning analytics dashboard for the Canvas LMS to assess learner engagement in discussion forums. This dashboard supports instructors in online settings by tracking students' forum activities. It reports on learners' participation and activity in discussions, their message interactions with peers, the sentiments expressed in their posts, and topic and keyword analysis of those messages. Based on the study's results, instructors reacted positively to the dashboard and stressed the need to simplify its information display. Although they overall viewed it favorably, they also offered cri-

tiques. For example, one instructor wasn't convinced by the forum discourse analysis, since it didn't give a clear picture of the discussion's status. Others felt the analysis only showed data on forum conversations and offered little guidance for improving teaching methods. As a result, the proposed dashboard cannot be customized to each instructor's specific needs for analyzing learner activities.

Karademir et al. [20] proposed a learning analytics dashboard to help high school teachers give feedback. Its goals are to present formative assessment results to teachers so they can improve instructional decisions and teaching, turn the dashboard data into the instructional actions the teacher needs, and provide feedback. Since most teacher dashboards are designed to display learner information and raise awareness, this study helps facilitate instructional interventions—such as collecting feedback—and, based on the data presented, recommends the next teaching steps. Interview findings showed that the learning analytics dashboard's feedback system should be designed to minimize the steps needed to give feedback. The results of this study can guide the development of teacher dashboards for formative learner assessment, with features for monitoring progress and providing feedback.

Khelifi et al. [21] introduced an explainable learning analytics dashboard that presents data on learners' performance and engagement. This study aims to make information accessible to teachers and students who may not be well-versed in data analysis. This dashboard lets teachers track learner progress and spot at-risk students, then recommend appropriate interventions based on their status. The dashboard proposed in this study is evaluated on the Blackboard Learn LMS. This dashboard supplies instructors and learners with clear information so they can take steps to improve. Learners can compare their progress with one another, and instructors can spot learners with similar behaviors to offer targeted support. It also helps enhance in-class teaching. The dashboard includes multiple data views, each designed for a specific purpose and to make the results easier to interpret.

Vásquez Bermúdez et al. [22] present the development of a dashboard module for Moodle, aimed at enhancing the tracking of student activities and performance metrics. Utilizing a cascade methodology, the project involved phases of analysis, design, development, testing, implementation, and maintenance. The backend was developed using PHP, while Chart.js was employed for data visualization, allowing for intuitive graphical representations of student progress. MySQL served as the database for efficient data storage and retrieval. Performance metrics indicated a 40% reduction in information access time and a 15% increase in activity completion rates among the experimental group. The dashboard's user-friendly interface promotes personalized learning experiences by providing educators with actionable insights. Overall, the module demonstrates significant potential for optimizing teaching strategies and improving student engagement in educational settings. This work contributes to the ongoing discourse on leveraging technology to enhance educational outcomes.

3.2 Related Research On The Development Of Teacher Performance Analysis Dashboards

In this section, two studies conducted on the development of teacher performance analysis dashboards are described.

Muryjas et al. [14] have investigated the management of remote evaluation for teachers. In this work, they enable the evaluation of teachers using business intelligence tools. The key performance indicators utilized in this study to assess the performance of teachers include the number of enrolled learners in the course, the percentage of overall learner activity, the percentage of interactive learner activity, the number of interactions with course materials, the number of site visits for the course, the site visit rate for the course, the percentage of returning learners, learner engagement rate, the activity-to-visit ratio, attendance percentage, the first-attempt exam pass rate, the average number of exam attempts, learner satisfaction level, average time spent on the site, bounce rate, the number of pages visited per visitor, average teacher time spent, overall teacher performance, the percentage of instructional activities, the percentage of non-instructional activities, the percentage of organizational activities, teacher engagement rate, the number of uploaded materials, teacher response time to learners, and the number of initiated online activities. The dashboard developed in this study is accessible independently of the Learning Management System and utilizes Power BI tools. Additionally, the proposed evaluation criteria are not customizable.

Sugiyanti et al. [23] designed a dashboard for monitoring teachers' performance. This dashboard is based on interviews with high school administrators and is designed around key performance indicators. In this study, a user-centered approach was employed to describe the user interface. This research aims to design a dashboard that assists administrators in monitoring and evaluating teachers' performance. The proposed key performance indicators for designing this dashboard include administrative evaluation, teaching process evaluation, measurement of individual virtues evaluation results, voluntary activity evaluation scores, overall key performance indicator evaluation, and teacher information. Administrative evaluation shows the availability of results of annual and

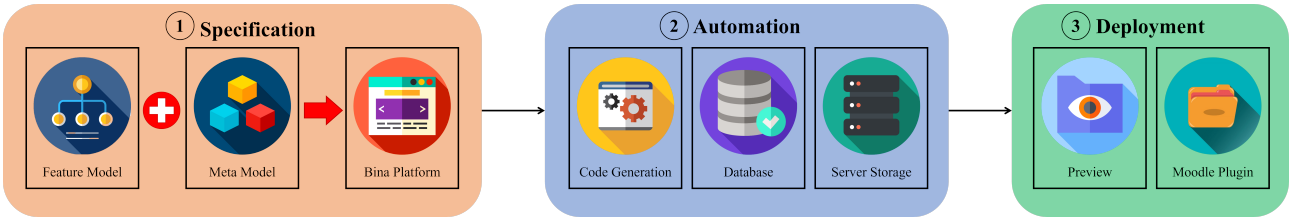

Figure 1. Dashboard Development Process Using A Low-code Platform.

term plans. In the teaching process evaluation, the results of teaching and learning activities during the teaching process are shown. The results of individual virtues indicate the percentage of cultural activities. The overall key performance indicators evaluation can display the percentage of data from the total evaluation of each key performance indicator. The dashboard presented in this study is also accessible independently of the Learning Management System and utilizes Google Data Studio tools. The proposed evaluation criteria are also not customizable.

3.3 Research Related to The Development of Dashboards Using Model-Driven Engineering or Low-Code Development

Pérez-Berenguer et al. [24], utilizing a model-driven engineering approach, presented a domain-specific language for creating customizable dashboards and applying an incremental approach for more efficient dashboard visualization for teachers, aimed at monitoring learners' progress. This language allows learners to be classified into different groups based on their performance levels using logical expressions. Learners using this dashboard can only view data related to their learning progress and some general information about their classmates. For example, a learner can compare their learning status with other learners, but the teacher has access to information about all their students.
Using the provided domain-specific language, teachers can define and customize course content, course evaluations, the sequence of course units, and gamification activities according to their needs. Using a generator engine, the four mentioned types of activities are integrated into a single learning unit for prototyping this learning unit. Additionally, the learning units assist teachers in customizing their dashboards. To construct the dashboard, the required data for customization is stored in JSON format. Then, for each update and dashboard rendering, the rendering information is stored in the MongoDB database. Finally, the generated dashboard is made available as a web-based application. The output dashboard of this study is customizable based on the evaluation criteria of learners' key learning indicators.
Jiang et al. [25] proposed a framework called Mod2Dash for the design and production of customizable dashboards. Using this framework, the desired designs are registered as models, and the dashboards are generated automatically. Additionally, a customization approach based on a graphical user interface has been provided for configuring dashboards and updating the created model. Creating an initial model and generating the dashboard helps in real dashboard evaluation, collecting feedback, and improving the dashboard. Using the customization approach based on the graphical user interface, users can manually edit and adjust their designs. This framework is implemented using a client-server architecture. The generated dashboard is accessible via a website that includes visual and interactive elements. The effectiveness of this framework was evaluated through a case study on 31 dashboards for decision support. These 31 dashboards were gathered and analyzed using various real-world scenarios. Based on the effectiveness evaluation, this framework efficiently records dashboard designs and generates them for different scenarios. This study proposes a method for designing general dashboards with customizable charts that can be used by users with various requirements. However, the output dashboard is not suitable for analytical purposes.

Table 1 presents a comparison of the present study with related research. The current study has been developed using a low-code development method. The user for platform development is the system administrator, and the evaluation audience targeted by the dashboard is the teachers. Using the provided low-code platform, system administrators can customize the dashboard based on desired key performance indicators, teacher evaluation criteria, and chart types.
Using this platform, the user can design and customize a teacher performance analysis dashboard without needing programming knowledge or plugin development requirements. In this platform, instead of existing works [13, 18–24], KPIs for the dashboard can be selected based on instructor evaluation criteria. Additionally, each KPI can be customized based on user type, chart type, and scheduling preferences. Moreover, once the dashboard is created, a preview option is available so that users can review and adjust it if

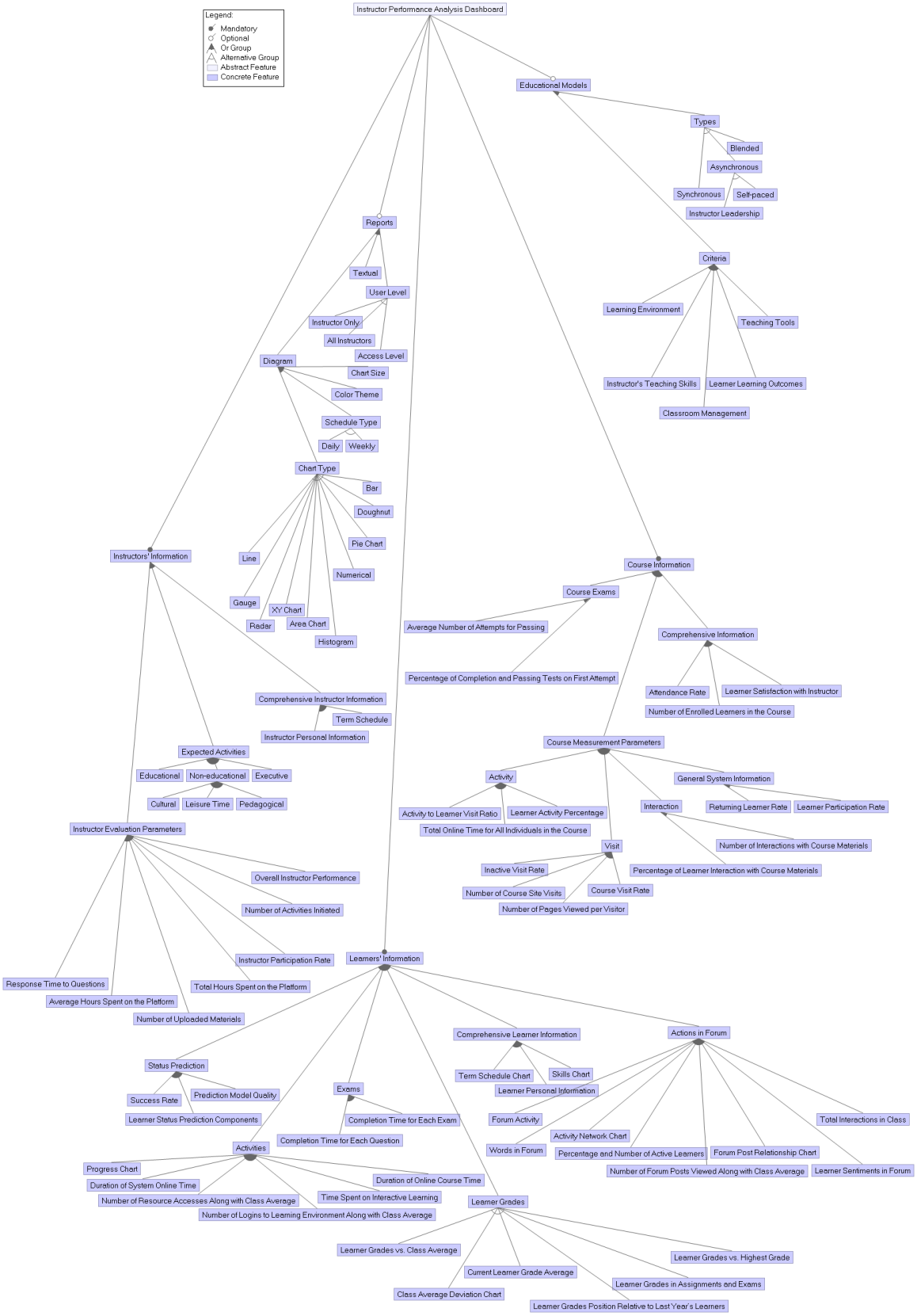

Figure 2. Proposed Feature Model for Performance Analysis Dashboards of Teachers.

Table 1. Comparison of the Present Study With Related Research.

Researcher	Year	Development Methodology	Data Visualization Type	System User	Evaluation Audience	Dashboard Customization			Final Output Type
						Key Performance Indicators	Evaluation Criteria	Chart Customization	
Kaliisa and Dolonen [19]	2023	Cascade Development	Graphical and Text-Based	Teachers	Learners	×	×	×	Canvas LMS Plugin
Karademir and et al. [20]	2024	Agile Development	Graphical	Teachers	Learners	×	×	×	Standalone Website
Khelifi and et al. [21]	2024	Agile Development	Graphical	Teachers	Learners	×	×	×	Output in Blackboard Learn LMS Plugin
Vásquez Bermúdez and et al. [22]	2025	Cascade Development	Graphical	Teachers	Learners	×	×	✓	Moodle LMS Plugin
Pérez-Berenguer and et al. [24]	2020	Model-Driven Development	Graphical	Teachers	Learners	×	Learners' Learning Index	×	Standalone Website
Muryjas and et al. [14]	2021	Agile Development	Graphical and Text-Based	System Administrator	Teachers	×	×	×	Output in Power BI Tool
Sugiyanti and et al. [23]	2022	Agile Development	Graphical and Text-Based	System Administrator	Teachers	×	×	×	Output in Google Data Studio Service
Jiang and et al. [25]	2022	Model-Driven Development	Graphical and Text-Based	Designers, Data Engineers, and Executives	Builds a Public Dashboard	×	×	✓	Standalone Website
The Present Study	2024	Low-Code Development	Graphical and Text-Based	System Administrator	Teachers	✓	Evaluation Criteria for Teachers	✓	Moodle LMS Plugin
User Guide: × No Support ✓Support									

the settings are not satisfactory. Using the provided platform, system administrators can easily implement the instructor performance analysis dashboard and deploy it within the Moodle learning management system. Finally, the generated dashboard is provided as a plugin for the Moodle Learning Management System.

4 Dashboard Development Process Using a Low-Code Platform

The distinguishing feature of the present study compared to related research is the provision of a low-code platform aimed at producing performance analysis dashboards for teachers, customizable based on evaluation criteria and key performance indicators. Developing performance analysis dashboards for teachers typically requires programming knowledge, significant time, and costs, and necessitates specialized personnel for development. However, using the proposed platform, dashboard development is feasible for everyone without requiring programming knowledge, significant time, or costs. Additionally, the visualization of charts for each key performance indicator can be customized. Considering the features and popularity of the Moodle Learning Management System, the generated dashboard can be installed as a plugin on the Moodle LMS. In addition to the mentioned advantages of the proposed platform, it also offers the capabilities of deleting, modifying, and previewing the created dashboard. Users can edit the generated dashboard using the dashboard modification feature and reuse it. In the preview feature, users can view an overview of the created dashboard in an environment similar to

the Moodle system, allowing them to ensure the chart visualization before installing the dashboard on Moodle. Given the characteristics of low-code development platforms, the developed low-code platform features a graphical interface, reusability of the created application, usability by non-experts, and deployment and maintenance capabilities.
The dashboard development process using a low-code platform is shown in Figure 1. This process consists of three sections: Specification, Automation, and Deployment. Each of these sections will be explained in the following. In the Specification process, the proposed platform is presented based on the metamodel and feature model. In the Automation stage, considering the users' choices for creating a customized dashboard, the desired dashboard plugin is generated, and its information is stored in the database and the output files' content on the server. In the Deployment stage, users can upload and install the plugin on their Learning Management System. Additionally, the dashboard can be viewed.

4.1 Specification

In this section, domain modeling is initially performed. For this purpose, related articles and existing dashboard examples were reviewed, and the feature model for this research was then created. Following the feature model and platform process, the metamodel was developed. Finally, using the feature model and metamodel, the Bina low-code platform was developed. In the Bina low-code platform, the user specifies and customizes the teacher analysis dashboard. The components of this section are explained in the following sections.

4.1.1 Feature Model

The first step towards automating and accelerating the process of designing performance analysis dashboards is identifying potential and impactful features in these types of systems. A feature model is a solution for identifying and categorizing capabilities and features within a domain [26]. The creation of a feature model related to performance analysis dashboards provides the necessary conditions for the custom development of performance analysis dashboards using low-code development methods. Before developing the low-code platform, it is first necessary to create its metamodel. Given the absence of a metamodel for developing performance analysis learning dashboards for teachers, the feature model is used to construct the metamodel. The metamodel is shown in Figure 2. Each subset of this feature model will be explained in the following sections.

- ***Features and Categories***
  The proposed features for the teacher performance analysis dashboard are categorized into five types. These features include teacher information, learner information, educational models, course information, and reports. The reports section contains features related to analytical reports and charts. Teacher information, learner information, and course information features are mandatory, while educational models and reports features are optional.
- ***Teachers' Information***
  The features related to teacher information are categorized into three groups: teacher evaluation parameters, expected activities, and comprehensive teacher information. The features related to teacher evaluation parameters include the average hours spent on the platform, total hours spent on the platform, the number of materials uploaded in the course, teacher response time to questions, overall teacher performance, teacher participation rate, and the number of initiated activities. The teacher's response time to questions indicates the duration taken by the teacher to answer learners' questions in the course discussion forum. The overall teacher performance shows the ratio of actual hours spent on the platform to the scheduled hours according to the course timetable. Teacher participation is a value ranging from 1 to 10, calculated and shown based on learners' feedback. The number of initiated activities indicates the number of activities in the course, such as exams, surveys, forums, and assignments. Total hours spent on the platform and average hours spent on the platform display the total and average duration the teacher has spent on the learning management system, respectively. Expected activities include three features: educational, non-educational, and executive. Educational activities encompass teaching activities in the course. Non-educational activities include activities other than teaching. Executive activities encompass tasks unrelated to learners. Non-educational features include activities related to leisure, educational guidance, and cultural affairs. Finally, comprehensive teacher information includes the teacher's personal information and their term schedule.
- ***Learners' Information***
  The features related to learner information are categorized into six groups: comprehensive learner information, tests, learner grades, forum activities, activities, and status prediction.
  In the category of comprehensive learner information, there are three features: the term schedule chart, learner personal information, and learner skills chart.

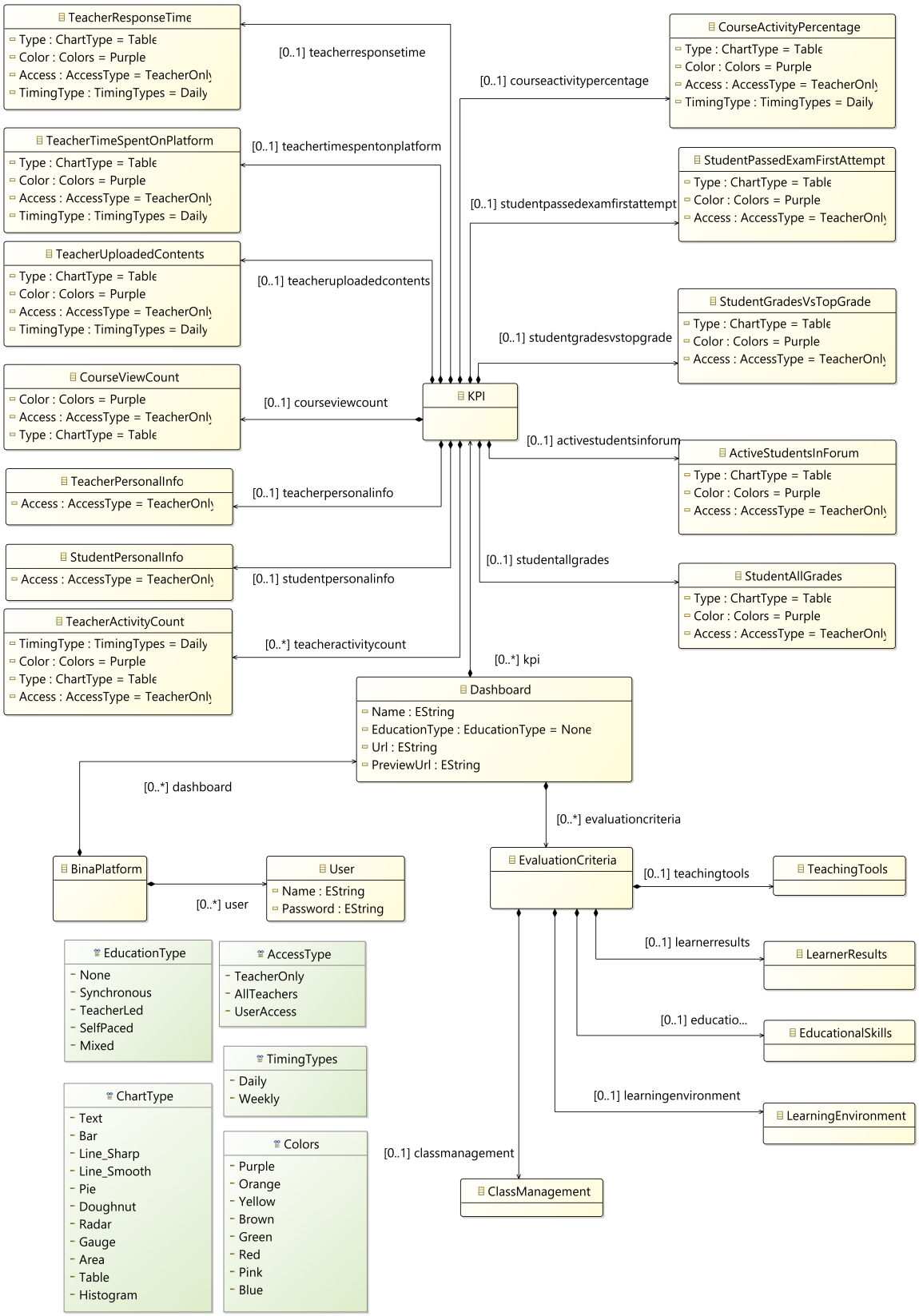

Figure 3. Metamodel of the Proposed Platform.

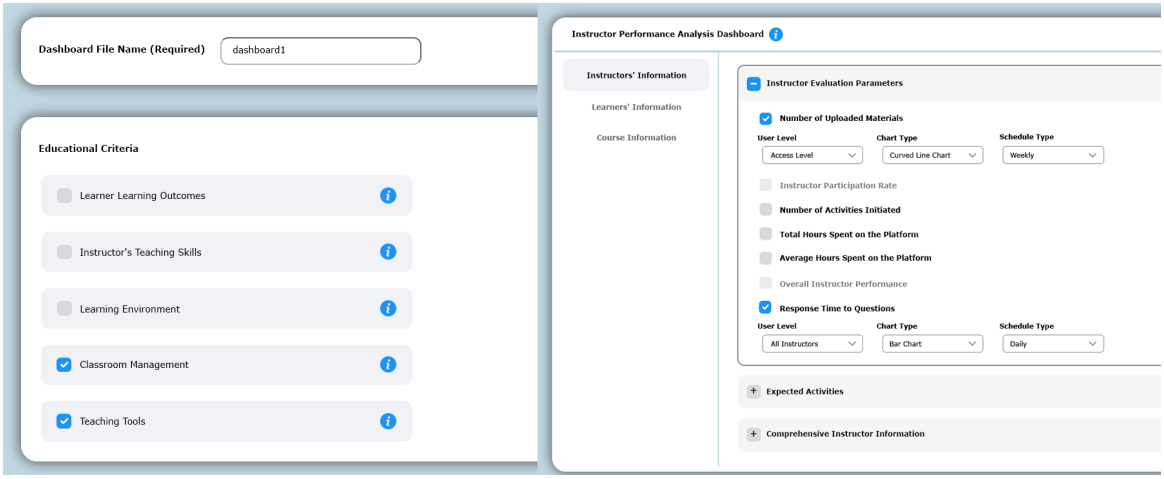

Figure 4. A Section of the Plugin File Name and Educational Criteria Selection Pages and Key Performance Indicators Selection.

```php
<?php
namespace Models;

trait SanitizerTrait
{
    public static function sanitizeinput ($input) {
        if (is_array( value: $input)) {
            return array_map( callback: [self::class, 'sanitizevalue'], array: $input);
        }
        else {
            return self::sanitizevalue( value: $input);
        }
    }

    private static function sanitizevalue ($value) {
        $sanitizedValue = trim( string: $value);
        $sanitizedValue = filter_var( value: $sanitizedValue, filter: FILTER_SANITIZE_STRING);
        $sanitizedValue = htmlspecialchars( string: $sanitizedValue, flags: ENT_QUOTES, encoding: "UTF-8");

        return $sanitizedValue;
    }
}
```

Figure 5. User Input Filtering Class.

In the tests section, there are two features: the time taken to complete each question and the time taken to complete each test. In the learner grades section, the features related to learner grades are presented in six categories. These features include learner grades compared to the highest grade, the current status of the learner's average grades, the status of the learner's grades compared to learners from the previous year, learner grades compared to the class average, learner grades in assignments and tests, and the deviation chart in the class average.

In the forum activities section, the actions and activities of learners in the forum are included. This information comprises the activity network chart, learner sentiments in the forum, total interactions in the class, the number of forum posts viewed along with the class average, the percentage and number of active learners, forum post interaction chart, words used in the forum, and forum activities. In the activity network chart, the node size indicates the number of participations, and the branch thickness indicates the number of interactions with that learner. Learner sentiments in the forum can be positive, negative, or neutral.

The total interactions in the class show the total interactions with posts, such as responding to posts by learners. The forum post interaction chart is a tree diagram that displays the relationships between forum posts based on keywords. The words used in the forum display the keywords used in the posts. Finally, in the forum activity chart, learner activities in the forum are depicted based on the number of posts made, the number of responses to posts, and the overall

participation percentage.

In the activities section, the features related to learner activities are included. These features comprise the time spent online in the learning management system, the time spent online in courses, the number of accesses to learning resources, along with the class average, the number of logins to the learning environment in the learning management system along with the class average, the time spent on interactive learning, and the progress chart. In the activity chart, the progress of activities performed by the learner is shown sequentially with different color codes. The time spent on interactive learning also displays the total time spent reading learning materials provided by the teacher.

In the status prediction section, the parameters for predicting learner status are included, comprising three parameters: prediction model quality, success rate, and learner status prediction components. The learner status prediction components include predictions of the learner's current status, such as deviation percentage, and the prediction of future assignment and test scores based on previous learners. The prediction quality chart indicates the quality of predictions to display the model's predictive power, and a lower score for this feature indicates higher accuracy. The success rate chart shows the predicted status of passing the current term courses for the learner.

- ***Course Information***

  Course information is divided into three categories: comprehensive information, course measurement parameters, and exams.

  In the comprehensive information section, there are three features: learner satisfaction with the teacher, the number of learners enrolled in the course, and learner attendance percentage. Learner satisfaction with the teacher is a value between 1 and 10 that represents satisfaction with the course teacher. The exam feature is divided into two categories: the average number of attempts to pass the exams and the percentage of completion and passing the exams on the first attempt by the learners.

  Course measurement parameters are divided into four categories: general site information, visits, interaction, and activity. In the general site information section, there are two parameters: learner participation rate and returning learner rate. The learner participation rate is calculated based on the ratio of total online activities performed to the number of learners enrolled in the course. The returning learner rate is determined based on the ratio of the number of learners returning to the course site within a unit of time to the total number of learners enrolled in the course. In the visits section, the parameters related to learners' visits to the course are included, such as course site visit rate, inactive visit rate, number of pages visited per visitor, and number of visits to the course site.

  In the interaction feature, the attributes related to learner interaction with course materials are included. These attributes encompass the number of interactions with course materials and the percentage of learner interactions with course materials. The percentage of learner interactions with course materials is calculated based on the ratio of the number of learners who have interacted at least once with online course materials to the total number of learners enrolled in the course. In the activities section, information on learners' activities on the site, such as email, discussion, and forum posts, is included. This section includes three attributes: the percentage of learner activity, the ratio of activity to learner site visits, and the total time online by all individuals in the course. The percentage of learner activity is calculated based on the ratio of the total activities of each learner to the number of learners enrolled in the course. The activity-to-visit ratio is calculated based on the ratio of the number of activities performed during visits to the total number of visits.

- ***Educational Models***

  Educational model features are optional and can be categorized into two groups: types and criteria. The types feature includes various educational models for teaching. In the criteria feature, the aspects related to the evaluation criteria for teachers are described. These criteria were detailed in the background section. The various educational models for teaching that can be implemented through learning management systems are classified into three categories: asynchronous, synchronous, and blended.

  In the asynchronous method, learners use the system according to their schedule without the need for real-time interaction with their teacher during the class. Utilizing lectures, forums, and audio discussions can be suitable for enhancing the learning experience in asynchronous courses. Asynchronous learning can be implemented through two methods: self-paced and teacher-led. In the self-paced method, learners follow the course material at their own pace. However, in the teacher-led method, learning is conducted according to the teacher's schedule and expected process. In the synchronous method, learners interact with the teacher and other learners at least once a

week. The teacher conducts the teaching using a dedicated agenda. In this method, the teacher doesn't need to lead the lectures, and learners can also lead the discussions themselves. Both synchronous and asynchronous methods use the learning management system to facilitate learning for the learners. The blended learning approach provides an environment where learners can simultaneously use both asynchronous and synchronous methods [8].

- ***Reports***
  Dashboard report features are classified into three categories: textual, graphical, and display level. The graphical section includes various types of charts. The display level section comprises three categories: teacher only, all teachers, and access level. If the display level is set to teacher only, only the teacher can view the key performance indicators, and they are not visible to the system administrator, with only the teacher's information being displayed. If it is set to all teachers, both the teacher and the system administrator can view all teachers' information. Finally, if the access level is selected, the system administrator can view all teachers' information, while the teacher can only view their information.

4.1.2 Metamodel

Figure 3 illustrates the metamodel of the proposed platform. This metamodel has been utilized in the implementation of the Bina low-code platform. To use this metamodel, a sample of the BinaPlatform metaclass must first be created. This metaclass includes the User and Dashboard metaclasses. The User metaclass pertains to user information, while the Dashboard metaclass relates to dashboard information. After creating a sample of the Dashboard metaclass, the Evaluation Criteria metaclass must be used to determine the teacher evaluation criteria. With this metaclass, the five types of evaluation criteria metaclasses can be utilized, which include teaching tools, classroom management, learning environment, teaching skills, and learner learning outcomes. To create key performance indicators (KPIs), a sample of the KPI metaclass must first be created to enable the use of KPIs. Each key performance indicator has its customization features.

4.1.3 Platform Overview

Figure 4 shows a section of the plugin file name selection page and educational criteria, as well as the key performance indicators (KPIs) selection page for customization. On the plugin file name selection and educational criteria page, the desired name for the dashboard is chosen. Additionally, the teacher evaluation criteria, based on which the evaluation KPIs are automatically selected, are specified on this page. On the KPIs selection page, users can specify the teacher performance evaluation indicators and how they are displayed. This page is categorized according to the feature model. By selecting each KPI, the user type, display type, and scheduling type features for dashboard customization will be shown. In the user type feature, the access level for chart display on the dashboard can be configured, with display levels including teacher only, all teachers, and access level. Once these selections are confirmed, the system enters the automation phase.

4.2 Automation

In the automation section, a transformation engine has been implemented to generate the desired dashboard codes based on user selections. For the implementation of the transformation engine, a model-to-code conversion method has been used. The model is stored in JSON format and includes the identifiers of the key performance indicators selected by the user, along with the customization settings for that dashboard. In the proposed platform, templates have been written to extract data from the model and generate code using the constructed model. All templates for the key performance indicators are written in PHP files and are selected based on the user's choices to create their desired dashboard.

Additionally, the template invocation codes and the overall plugin configuration are also generated, and finally, the plugin folder is made available as a compressed file.

To implement the transform engine, 10 PHP classes were used. The first PHP class manages the database connection to the low-code platform and contains the necessary commands for database access. The second class is responsible for data filtering, defining rules for refining user input values. This class includes a function for filtering user input based on array input and variable input. For variable input, extra spaces and special HTML characters are removed, and the output is presented in HTML format for better user readability. For array filtering, each element in the array undergoes the same filtering process. The user input filtering class code is shown in Figure 5.

Three classes have been designated for storing templates and functions related to the plugin. These classes are named *"TeacherAnalytics.php"*, *"StudentAnalytics.php"*, and *"CourseAnalytics.php"*, corresponding to KPIs for teachers, students, and courses, respectively. Additionally, three other classes are used to store all functions related to the preview display.

The classes are named *"TeacherPreview.php"*, *"StudentPreview.php"*, and *"CoursePreview.php"*, and

```
function kpi_name ($user, $plot, $timing):
    $user_level = get user access level
    if $plot == "table":
        if $user == "user":
            $query = sql query for table and current teacher data
            $array = run query and convert result to array
            create table with html tags
            add rows, using $array data
        else if $user == "all"
            $query = sql query for table and all teachers data
            $array = run query and convert result to array
            create table with html tags
            add rows, using $array data
        else if $user == "admin_all_user_self"
            if $user_level == "admin":
                kpi_name ("all", $plot, $timing);
            else if $user_level == "user":
                kpi_name ("user", $plot, $timing);
        end if;
    else
        $queryResultArray = null;
        // $timing data could be daily or weekly
        if $user == "user":
            $query = sql query for current teacher data (with $timing input if exist)
            $queryResultArray = run query and convert result to array
        else if $user == "all"
            $query = sql query for all teachers data (with $timing input if exist)
            $queryResultArray = run query and convert result to array
        else if $user == "admin_all_user_self"
            if $user_level == "admin":
                kpi_name ("all", $plot, $timing);
            else if $user_level == "user":
                kpi_name ("user", $plot, $timing);
        end if;
        $x_data = create data for x axes use $queryResultArray
        $y_data = create data for y axes use $queryResultArray
        $chart = null;
        if $plot == "bar"
            $chart = create bar chart
        else if $plot == "pie"
            $chart = create pie chart
        else if $plot == "doughnut"
            $chart = create pie chart with set_doughnut(true);
        else if $plot == "line_sharp"
            $chart = create line chart
        else if $plot == "line_smooth"
            $chart = create line chart with set_smooth(true);
        end if;
        set $x_data and $y_data to $chart
    end if;
    return data;
end function;
```

Figure 6. Pseudo-code for KPI Functions.

they are used to manage the functions and templates for dashboard previews of KPIs for teachers, students, and courses, respectively. During the execution of the transform engine that converts models to text, these templates are used to select KPIs based on user choices and generate the plugin code and preview. Figure 6 shows the pseudo-code for implementing each KPI.

According to Figure 6, the overall implementation of KPIs for all six mentioned classes is based on customizable criteria. As explained in the metamodel section, customizable properties are provided as inputs to the function of each KPI. These inputs include access level type, chart type, and chart scheduling type. Some KPIs may not include properties such as scheduling type or chart type.

The most important classes for implementing the transform engine are *"FileReader.php"* and *"CreateFiles.php"*. The *FileReader* class extracts templates based on the KPIs selected by the user by reading the files related to plugin generation and preview. The *CreateFiles* class is responsible for generating the necessary files for plugin creation and preview. It uses the templates extracted by the *FileReader* class to build new files. Due to the large amount of code written for implementing the transform engine, pseudo-codes for the *FileReader* and *CreateFiles* classes are provided. The pseudo-code for FileReader.php is shown in Figure 7.

According to Figure 7, the data related to the user's selections is first converted into a JSON file. This file is then used to implement the transformations. After

```
convert selected kpi to json
initialize folders for plugin and preview files
initialize array of includes that contains include lines in main page of dashboard
initialize array for teachers, students and course kpis that contains plugin function texts of each selected kpi
initialize array of includes that contains include lines in main page of preview
initialize array for teachers, students and course kpis that contains preview function texts of each selected kpi
read json file
foreach selected kpi that has id name:
    if selected kpi starts with teachers:
        add "include_once classes/TeachersAnalytics.php" text to array of includes only one time
        add "include_once classes/TeachersPreview.php" text to array of preview includes only one time
        if name of kpi found in TeachersAnalytics.php class:
            add current line to array of teachers functions
            do:
                add next lines to array of teachers functions
            while current line contains return
        end if;
        if name of kpi found in TeachersPreview.php class:
            add current line to array of teachers preview functions
            do:
                add next lines to array of teachers preview functions
            while current line contains return
        end if;
    end if;
    if selected kpi starts with students:
        add "include_once classes/StudentsAnalytics.php" text to array of includes only one time
        add "include_once classes/StudentsPreview.php" text to array of preview includes only one time
        if name of kpi found in StudentsAnalytics.php class:
            add current line to array of students functions
            do:
                add next lines to array of students functions
            while current line contains return
        end if;
        if name of kpi found in StudentsPreview.php class:
            add current line to array of students preview functions
            do:
                add next lines to array of students preview functions
            while current line contains return
        end if;
    end if;
    if selected kpi starts with courses:
        add "include_once classes/CoursesAnalytics.php" text to array of includes only one time
        add "include_once classes/CoursesPreview.php" text to array of preview includes only one time
        if name of kpi found in CoursesAnalytics.php class:
            add current line to array of courses functions
            do:
                add next lines to array of courses functions
            while current line contains return
        end if;
        if name of kpi found in CoursesPreview.php class:
            add current line to array of courses preview functions
            do:
                add next lines to array of courses preview functions
            while current line contains return
        end if;
    end if;
end foreach;
create instance of CreateFiles class and pass arrays of includes and arrays of plugin function texts
```

Figure 7. Pseudo-code for FileReader Class.

that, two arrays are created: one for the *"include_once"* code strings and another for the template content strings selected by the user for plugin generation.

Two additional arrays are also created for the *"include_once"* code and template content used in the preview section. Using the *"include_once"* statement in PHP, project files can be included, and it's enough to call a file just once with this command. The selected text strings, which are templates for key performance indicators (KPIs), are stored in an array for easier use. Next, the saved JSON file is read. Each KPI in the low-code platform has a unique ID, and the IDs of the selected KPIs are placed in an array called id. A loop is then used to iterate through this array and populate the arrays related to the include_once code strings and the KPI functions used for the plugin and its preview. If the beginning of the selected ID name is teachers, the process moves to selecting the plugin code strings for teacher KPIs from the *"TeacherAnalytics.php"* class. If it starts with a student, it selects the KPI plugin code strings for learners from the *"StudentAnalytics.php"* class.

Finally, if the ID starts with course, the code strings are selected from the CourseAnalytics.php class. At the beginning, the statement *"include_once classes/TeacherAnalytics.php"* is added to the corresponding array to load the newly generated TeacherAnalytics.php file. This file will be created in the classes folder. Then, in the main *"TeacherAnalytics.php"* class, the code is searched based on the KPI identifier. Once the identifier is found, the corresponding code block—up to the end of its function—is added to the template array. A similar process is followed for selecting preview code, except that for teachers, the function search is performed in the *"TeacherPreview.php"* class, and the result is added to the preview array for teachers.

To select the templates and functions related to key performance indicators (KPIs) for learners and courses, the same process used for teachers is applied. The necessary code is chosen from their corresponding classes. Next, an instance of the *"CreateFiles.php"* class is created, and its main functions are used to generate the core files and complete the plugin and

```
function initializePluginFiles (inludes, functions, calls, userid, fileName):
    initialize sub folders for plugin and preview files
    if includes contains TeacherAnalytics.php:
        create new TeacherAnaytics.php file in folder with classes name
        write $teacher = TeacherAnalytics(); in main.php file of plugin
        write functions of teacher analytics from functions variable to created TeacherAnalytics.php file
    end if;
    if includes contains StudentAnalytics.php:
        create new StudentAnaytics.php file in folder with classes name
        write $student = new StudentAnalytics(); in main.php file of plugin
        write functions of student analytics from functions variable to created StudentAnalytics.php file
    end if;
    if includes contains CourseAnalytics.php:
        create new CourseAnalytics.php file in folder with classes name
        write "$course = new CourseAnalytics();" in main.php file of plugin
        write functions of course analytics from functions variable to created CourseAnalytics.php file
    end if;
    add selected kpi function inputs from calls variable such as access level, chart type or timing type
    add plugin config settings to plugin folder
end function;


function addCalls (callsArray):
    $outputText = "";
    if callsArray[0] != null:
        $teacherArray = split callArray[0] by space;
        foreach item in $teacherArray:
            outputText += $result .= $teacher -> item;
        end for;
    end if;
    if callsArray[1] != null:
        $studentArray = split callArray[1] by space;
        foreach item in $studentArray:
            outputText += $result .= $student -> item;
        end for;
    end if;
    if callsArray[2] != null:
        $courseArray = split callArray[2] by space;
        foreach item in $courseArray:
            outputText += $result .= $course -> item;
        end for;
    end if;
    return $outputText;
end function;
```

Figure 8. Pseudo-code for CreateFiles Class.

preview setup. As shown in Figure 8, the *CreateFiles* class uses two main functions to produce the required code. The most important function is *initializePluginFiles*, which takes five inputs. The first input is the includes array, which holds the list of files to include using *"include_once"*. The second input is the string containing the KPI template code. The third input, called calls, stores the customization settings for the selected KPIs.

The fourth and fifth inputs are the user ID and the chosen name for the plugin file. These two inputs will be explained further in the next section. In this function, subfolders needed for plugin and preview generation—such as the classes folder—are created first. Input array element 0 always corresponds to teachers, while elements 1 and 2 are always related to learners and courses, respectively. If the includes array contains the entry *"TeacherAnalytics.php"*, a new *"TeacherAnalytics.php"* file is first created in the classes folder. After writing the include statements from the corresponding array, the line *"$teacher = new Teacher();"* is added to the *"main.php"* file.

In the *"main.php"* file, all calls needed to use the newly generated *Analytics* classes are made to run the dashboard plugin and display the desired dashboard in the Moodle system. Then, in the generated TeacherAnalytics.php file, the KPI code is selected using the function's input. In this class, a new *"TeacherPreview.php"* file is also created to build the preview, and the selected code is written into it. The same process is followed for creating new classes for learners and courses.

The second function, called *addCalls*, handles assigning input values to functions based on the custom KPI selections. This function is used within the *initializePluginFiles* function. First, a string variable is created to store all the function calls. Based on the array index number, the inputs are assigned to either teachers, learners, or courses. In each section, the input string is split by spaces and stored in an array. Then, for each element in that array, the corresponding function call string for teachers, learners, or courses is added to the main string variable.

For example, the expression *"$teacher-¿item"* is added for teachers. The variable used to access the *"TeacherAnalytics.php"* class is named *$teacher*, and the item variable contains the selected KPI identifier along with its related input values. Finally, the *addCalls* function returns the *outputText* string. This output string is used in the main dashboard file to write the necessary function calls. After the strings are written into their respective files, the plugin configuration files are added to the file set. Finally, the plugin output is packaged as a compressed file, ready

```
{
  "dashboard_name":"dashboard1",
  "hidden_edit":"",
  "ltype":["manage_class","teaching_tools"],
  "id":[
    "teacher_uploaded_content",
    "teacher_time_response",
    "student_active_students",
    "course_student_avg_attempts_to_pass"
  ],
  "teacher_uploaded_content":["admin_all_user_self","line_smooth","weekly"],
  "teacher_activities":["user","table","daily"],
  "teacher_times_pent":["user","total","daily"],
  "teacher_average_times_pent":["user","total","daily"],
  "teacher_time_response":["all","bar","daily"],
  "teacher_main_info":["user"],
  "student_quiz_time":["user","table"],
  "student_main_info":["user"],
  "student_all_grades":["user","table"],
  "student_grades_to_top_grade":["user","table"],
  "student_grades_to_avg_other":["user","table"],
  "student_active_students":["admin_all_user_self","line_sharp"],
  "student_login_count":["user","table"],
  "student_interaction_time":["user","table"],
  "student_daily_time_spent":["user","table"],
  "student_course_time_spent":["user","table"],
  "course_student_avg_attempts_to_pass":["user","table","daily"],
  "course_student_passed_percentage_first_attempt":["user","table","daily"],
  "course_student_participants":["user"],
  "course_student_activity_to_view_percentage":["user","table","daily"],
  "course_student_activity_percentage":["user","table","daily"],
  "course_view_count":["user","total","daily"],
  "course_student_view_without_activity_rate":["user","table","daily"],
  "course_student_view_rate":["user","table","daily"],
  "create_button":"\u0627\u062f\u0627\u0645\u0647"
}
```


Figure 9. Sample Output JSON File.

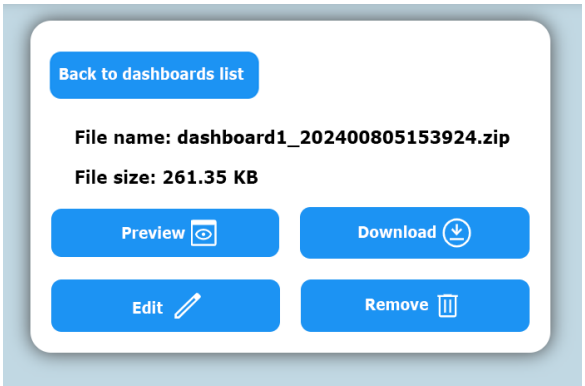

Figure 10. Dashboard Output Page.

for upload. The preview files are also saved on the server for display purposes.

The plugin configuration files assist in the easier installation of the dashboard on the Moodle system and configure the dependencies required for better dashboard performance. These files include *settings.php*, *lib.php*, and *version.php* files. The *version.php* file contains the name and version of the plugin, the minimum Moodle version for installation, and the required plugins for better performance. The *lib.php* file includes the navigation settings of the plugin after installation to help improve accessibility.

Finally, the *settings.php* file contains the overall settings for running the plugin. All files generated by the transformation engine include the compressed plugin file for installation in the Moodle system, a JSON file for storing dashboard settings, and the necessary files for dashboard preview. When editing the generated dashboard, the JSON file loads the dashboard information into the platform, enabling changes. The programming language used for implementing the transformation engine is PHP, and all files required for plugin generation and preview are also written in PHP. An example of the JSON file is shown in Figure 9.

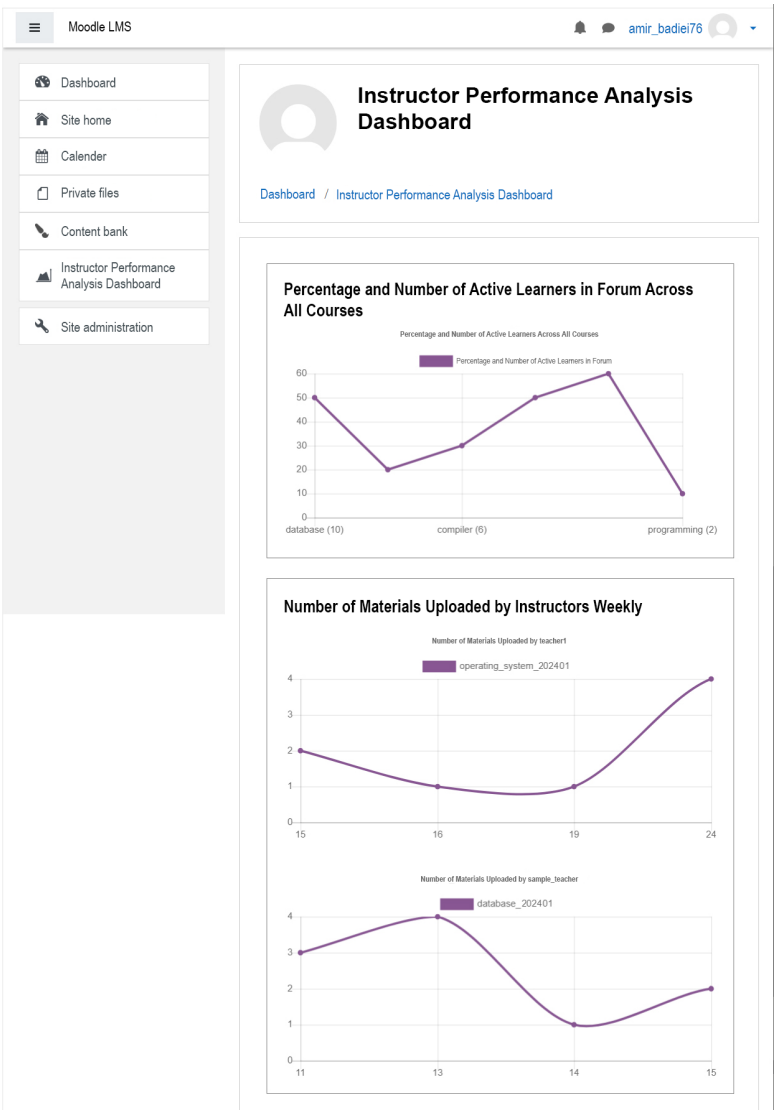

Figure 11. View of the Generated Dashboard Preview.

4.3 Deployment

In the deployment section, the dashboard output page is displayed on the platform. Similar to the dashboard list page, this page provides options to delete, edit, upload, and preview the dashboard.

By clicking the download button, the compressed file is downloaded to the system and can be used for deployment in the Moodle learning management system. By clicking the preview button, the generated dashboard with sample data is displayed in an environment resembling the Moodle learning management system. By clicking the edit dashboard button, the user is directed to the name and evaluation criteria selection page, where the generated dashboard can be edited.

To display the dashboard charts in Moodle and the preview section, the ChartJS library has been used. This library is implemented by default in Moodle for chart display and can be used based on the desired data. The dashboard output page is shown in Figure 10.

The plugin can only be installed by the system administrator on the Moodle learning management system. The generated plugin performs better on Moodle version 3.10. To install the Moodle plugin, first, extract the compressed file, then place the teachers_dashboard folder in the moodle/local directory. Next, access the Moodle URL to initiate the installation process. Utilizing the configuration files aids in the easy installation of the software, allowing users to use the plugin without needing to generate these files themselves.

The dashboard's installation requirement is the logstore_lanalytics plugin, which must be installed first during the installation process. Using the logstore_lanalytics plugin, user activity reports are recorded. After the generated plugin is installed in the Moodle system, the generated dashboard becomes viewable. Upon deployment and installation of the

plugin in the Moodle learning management system, data is updated and displayed every time the dashboard page is accessed. A preview of the generated dashboard output is shown in Figure 11.

5 Evaluation

The evaluation of the present study is conducted in two parts. In the first part, three case studies are presented for different purposes, through which the utility of the proposed platform for various requirements is assessed. In the second part, a usability evaluation is conducted to assess ease of use and user satisfaction in utilizing the low-code development platform. The research questions are as follows:

(1) How much time and effort is required to learn and work with this platform compared to the traditional method of plugin development?
(2) To what extent does this platform support key performance indicators for creating various dashboards?

5.1 Utility Assessment

The subject of the first case study is the development of a dashboard aimed at reviewing the teacher's teaching results. The second case study focuses on the development of a dashboard for analyzing the use of teaching resources and tools. The goal of the third case study is to present a dashboard for teacher-learner interactions.
The objective of the dashboard developed for the first case study is to review the teacher's teaching results. The key performance indicators (KPIs) for this dashboard include students' grades compared to the class average, the percentage of students completing and passing on the first attempt, and the average number of attempts to pass the exams. In this case study, the KPI for students' grades compared to the class average is in the form of a bar chart. The KPI for the percentage of students completing and passing on the first attempt is displayed as a table, and the KPI for the average number of attempts to pass the exams is shown as a line chart. The design of this dashboard using the platform took approximately three minutes. Users do not need to be familiar with programming to utilize the proposed platform. The number of lines of code generated was 1,424 lines. The estimated time required to implement this plugin by a user with intermediate programming knowledge, without utilizing a low-code platform, is approximately three hours.
The dashboard for the second case study was developed to analyze the teacher's use of teaching resources and tools. The selected key performance indicators (KPIs) for this case study include the number of uploaded materials and the number of activities initiated in the course. For the KPI of the number of uploaded materials in the course, the user type is set to all teachers, and its display type is chosen as a table. The user type for the KPI of the number of activities initiated in the course is set to access level. The display type and scheduling type are selected as a pie chart and weekly, respectively. In the weekly scheduling type, the week number during which an activity was performed is considered. The design of this dashboard using the platform took approximately two minutes. The number of lines of code generated was 1,172. Implementing this dashboard through programming would require approximately two to three hours.
The objective of the dashboard developed for the third case study is to analyze the teacher's interaction with learners. The key performance indicators (KPIs) associated with this subject include the teacher's response time to questions in the forum, the percentage and number of active learners in the forum, and the percentage of learner activity. For the KPI of the teacher's response time to questions in the forum, the user type is set to all teachers, and its display type is set to total to show the total response time of all teachers to learners' questions in the forum. For the KPI of the percentage and number of active learners, the user type is set to access level, and its display type is chosen as a donut chart. Regarding the percentage of learner activity, the user type is set to teacher only, allowing each teacher to view the learner activity in their courses. Its display type is chosen as a curved line chart with a daily scheduling type. The number of lines of code needed to implement this platform was 1,472 lines. The design of this dashboard using the proposed platform takes three minutes, whereas implementing it through the traditional method for a user with intermediate programming knowledge would take approximately four hours.

5.2 Usability Evaluation

In this section, the usability evaluation and user satisfaction in using the proposed low-code development platform are addressed. To this end, an online workshop was held. The workshop included training on using the platform, implementing the dashboard, customizing it, and viewing its preview. In this section, the usability of the Bina platform for generating performance analysis dashboards of teachers is assessed using a questionnaire evaluated by 37 participants, and the results are reported. For the platform evaluation, the online workshop was conducted with system administrators from various educational institutions and individuals familiar with the operation of learning management systems. To accomplish this, we shared

Table 2. Platform Evaluation Results.

Group	Evaluation Question	Very High	High	Moderate	Low	Very Low
Efficiency	Extent of Simplification in Accelerating Dashboard Development	12	20	5	0	0
	Understanding the Dashboard Using Preview Functionality	18	16	3	0	0
Aesthetics	Utilization of Platform Features Based on the User Interface	12	15	10	0	0
	Suitability of Color Schemes and Fonts	15	16	6	0	0
Ease of use	Comprehensibility of Various Platform Sections	8	17	10	2	0
	Ease of Using the Platform for Dashboard Creation	14	18	5	0	0
	Ease of Editing and Recreating Dashboards	16	15	4	2	0
Guides and documentation	Clarity of the Platform's Documentation	17	10	10	0	0
	Time Required to Review the Platform's Documentation	0	6	11	13	7
	Utilization of Help Pages for Platform Usage	9	19	7	2	0
	Time Required to Install the Plugin in the Moodle LMS	0	10	16	7	4

the workshop content with the Isfahan teaching community, as well as students from the Faculty of Education at the University of Isfahan.

In this workshop, participants were asked to first watch an instructional video [2] on how to use the platform. Then, using the implemented platform, they created the teacher performance analysis dashboard and finally evaluated the platform by filling out an assessment form [3]. The evaluation questions were defined based on ISO 9241 standards [27]. These questions were categorized into four groups: efficiency, aesthetics, ease of use, and guides and documentation. The evaluation results [4] of the platform are shown in Table 2. Additionally, the assessment form was shared on social media and sent by professors to the Faculty of Education to answer the platform evaluation questions. The workshop participants included 27 men and 10 women, aged 21 to 45. Five of the evaluators were system administrators in different educational institutions. Fifteen evaluators were undergraduate students, seventeen were master's students, and five were doctoral students which are all of them educational students.

In general, the time required to create a dashboard using the proposed low-code platform ranges between two to eight minutes. The duration of dashboard design through traditional methods varies depending on the volume of key performance indicators. Based on measured times for traditional dashboard development, an average of approximately one hour is required for every 300 lines of code. Given that the number of lines of code generated by participants ranged from 700 to 6,000 lines of PHP code, around two to twenty hours of programming would be required. Therefore, based on participant feedback and the measured times compared to traditional development methods, using the proposed low-code platform helps reduce the time required to develop teacher performance analysis dashboards.

5.3 Threats To Validity

Construct Validity: Threats to construct validity are related to the accuracy of observations in connection with the hypothesis of the present study. In this research, a feature model, metamodel, and low-code platform were proposed for the development of teacher performance analysis dashboards with customization capabilities for the Moodle learning management system. To create the low-code platform, domain feature analysis was conducted, which was used to design the feature model. Subsequently, the metamodel and, finally, the low-code platform were implemented. The stages of dashboard development using the implemented low-code platform included modeling, automation, and deployment. We argue that there are no threats to construct validity.

Internal Validity: Internal validity threats refer to factors that might influence the research

[2] The instructional video on how to use the platform is available at the link https://youtu.be/n7mpEsITCrs.

[3] The platform evaluation form can be accessed via the link https://forms.gle/6kxrZKBgCKgXZd7F8.

[4] The files created by the participants and the three case studies, along with additional instructions for installing Moodle and the plugin, can be accessed via the link https://github.com/amirbadiei76/Bina.

results. The selection of topics for the three case studies could be one potential threat. For each case study, the most relevant key performance indicators (KPIs) were selected; however, in future developments, more KPIs will be added to the low-code development platform. Another threat pertains to validation constraints to prevent the incorrect generation of dashboard codes. To avoid errors in code generation, the codes are selected from pre-written templates. Each template related to KPIs is reviewed and tested before being added to the platform.

Reliability Validity: Based on the conducted case studies, three case studies were carried out with the topics of designing a dashboard to review the teacher's teaching results, a dashboard for analyzing the use of teaching resources and tools, and a dashboard for teacher-learner interaction. However, it cannot be claimed that all dashboards that could potentially be created are supported by the proposed platform. By generalizing the feature model and developing additional templates for other key performance indicators (KPIs), it becomes possible to create various dashboards with a wider range of topics. Additionally, by enabling the extensibility of KPIs, the platform allows users to create dashboards with their desired KPIs. KPI extensibility is one of the future tasks aimed at enhancing the capabilities of the low-code development platform.

External Validity: Threats to external validity relate to the generalizability of the results. The Bina low-code platform was evaluated through three case studies and subsequent user surveys conducted via an online workshop. To generalize the platform evaluation, additional case studies with different topics should be utilized alongside the existing three case studies. In the conducted online workshop, five of the evaluators were system administrators, and it is possible that the selected target population for evaluating the platform was insufficient and did not yield accurate results. Nevertheless, for further evaluation of the usability and effectiveness of the platform in producing teacher performance analysis dashboards, additional studies should be conducted with system administrators and educational institutions.

6 Discussion

This article introduces a low-code development platform for designing teacher performance analysis dashboards. The purpose of this platform is to minimize manual coding, reduce development costs and time, and enable non-expert users to utilize it. Most analytical dashboards are designed for evaluating learners, and despite the significance of teacher performance as a key element in the educational process, little attention has been given to the development of dashboards capable of evaluating teacher performance. Furthermore, most existing dashboards are external to learning management systems, requiring the end user to rely on third-party software for performance analysis and displaying analytical charts. This, along with the high cost of custom development for these systems, has resulted in limited adoption of teacher performance analysis dashboards by educational institutions. Therefore, the use of the proposed low-code platform facilitates the development of teacher performance analysis dashboards for educational institutions. However, as the platform scales to larger datasets and more concurrent users, performance tuning and efficient resource management become critical to maintain responsiveness. Our platform is designed with a modular architecture that allows it to handle increasing volumes of teacher data and a growing number of dashboards without significant performance degradation. This is achieved by leveraging Moodle's underlying robust database and employing efficient data retrieval mechanisms, ensuring that the dashboards remain responsive even as the institution's size and data grow. Handling sensitive teacher performance data also demands robust privacy safeguards—such as encryption at rest, strict access controls, and audit logging—to protect personal information. Our platform adheres to Moodle's established security protocols and allows for granular access control, ensuring that only authorized personnel can view specific dashboards and data. Integration of diverse plugins may face compatibility issues that require extra configuration or precise version matching. As a Moodle-integrated solution, our platform is developed to be highly compatible with existing Moodle plugins and functionalities. This ensures a seamless user experience and minimizes potential conflicts with other tools educational institutions may already be using. Finally, although the system is aimed at non-expert users, there is still an initial learning curve as they familiarize themselves with the interface, configuration of data sources, and visualizations.

This article begins by addressing the concepts necessary to understand teacher performance analysis dashboards and various learning management systems. Subsequently, related research and a comparison of the present study with those studies are discussed. In the solution section, the low-code platform is explained, and in the evaluation section, the present study is assessed in two parts. Based on the usability evaluation results, three case studies with different objectives were created. Then, to evaluate usability, the proposed low-code platform was assessed by 37 participants. According to the evaluation results, the

low-code platform facilitates faster dashboard development. Finally, the two questions posed at the beginning of the evaluation section must be addressed.
Response to the first question: Given the availability of a graphical interface and guide pages for various sections of the platform, users can quickly learn how to use it. Furthermore, additional features, such as the inclusion of an intelligent assistant and improved user interface, will be incorporated into the platform in future developments.
Response to the second question: Based on the studies conducted to design key performance indicators (KPIs) relevant to teacher performance evaluation, this research sought to collect all the necessary KPIs to enable the creation of all the dashboard configurations required by users.

7 Conclusions

Nowadays, the use of learning management systems (LMS) has increased significantly. This has made the analysis of teacher performance essential, considering the diverse components of teaching. In the developed platform, users can customize and generate teacher performance analysis dashboards without requiring programming knowledge or plugin development expertise. Within this platform, key performance indicators (KPIs) for the dashboard can be selected based on teacher evaluation criteria. Additionally, each KPI can be customized according to the user type, chart type, and scheduling type. Furthermore, after creating the dashboard, a preview feature allows users to review the settings and make adjustments if needed. Using the proposed platform, system administrators can easily implement teacher performance analysis dashboards and deploy them within the Moodle learning management system.
To enhance and expand the proposed platform and improve its usability, several areas can be addressed. These include improving customization parameters, providing generated dashboards for other learning management systems (LMS), intelligent platform enhancement, improving the performance of the generated charts, and expanding the extensibility of key performance indicators (KPIs). In the proposed platform, available customization features include teacher evaluation criteria, KPIs, user types for displaying the desired KPI, and the chart type and scheduling. In future work, additional chart types can be introduced. The provided plugin file is currently only installable on the Moodle LMS. In future efforts, by researching other LMSs and gathering the requirements for plugin development for those systems, the generated plugin could be made applicable to other LMSs. To create a better user experience, an intelligent assistant can be incorporated. Adding an intelligent assistant to the platform will help users make better use of the platform and meet their needs more quickly. Since the output plugin operates within the Moodle LMS, the ChartJS library, available in Moodle, is used to display charts. This library only supports line, pie, donut, and bar charts. The low-code platform was developed based on the proposed metamodel and feature model, and the ability to add custom KPIs has not been implemented. To advance the platform's capabilities, we can extend multi-LMS compatibility by developing adaptive plugins for systems like Canvas and Blackboard using standardized API integrations, and implement a modular KPI framework supporting dynamic formulas, third-party extensions, and AI-driven metric suggestions.

References

[1] Renata Hryniewicz. Learning management system statistics: 2024 trends and facts. https://samelane.com/blog/lms-statistics/, 2024. Accessed: 29/11/2024.

[2] Zamzam Amhimmid Mare. Evaluating the performance of teaching staff members, a step towards teaching performance development of higher education in sebha university. *Journal of Pure & Applied Sciences*, 20:21–26, 2021. doi:10.51984/jopas.v20i3.1080.

[3] Sri Susanti, Pratiwi Pujiastuti, and Arif Purnanto. Students' perception of the assessment transparency based google sheet. *AL-ISHLAH: Jurnal Pendidikan*, 13(2):1269–1277, 2021. doi:10.35445/alishlah.v13i2.648.

[4] Tobias Rohloff. *Learning analytics at scale*. doctoralthesis, Universität Potsdam, 2021.

[5] Yeonjeong Park and Il-Hyun Jo. Development of the learning analytics dashboard to support students' learning performance. *JUCS - Journal of Universal Computer Science*, 21(1):110–133, 2015. doi:10.3217/jucs-021-01-0110.

[6] Ngoc Buu Cat Nguyen. Evaluating a teaching analytics dashboard in adult education: Lessons learned. In *Technology Enhanced Learning for Inclusive and Equitable Quality Education*, pages 235–240. Springer Nature Switzerland, 2024. doi:10.1007/978-3-031-72312-4_33.

[7] Alexander C Bock and Ulrich Frank. Low-code platform. *Business & Information Systems Engineering*, 63(6):733–740, 2021. doi:10.1007/s12599-021-00726-8.

[8] Vaughn Malcolm Bradley. Learning management system (LMS) use with online instruction. *International Journal of Technology in Education*, 4 (1):68–92, December 2021. doi:10.46328/ijte.36.

[9] Kim Mahoney and Leanne Cameron. An introduction to learning management systems. In *Readings in Education and Technology: Proceedings of the 8th ICICTE*, pages 314–323. University of Fraser Valley, 2008.

[10] K Syamala Devi, V Vijaya Lakshmi, and M Aparna. Moodle -an effective learning management system for 21st century learners. *Alochana Chakra Journal*, 9, 2020.

[11] Lambda-2022. [updated] moodle™ continues to be the world's leading lms software. https://www.lambdasolutions.net/blog/moodle-continues-to-be-the-worlds-leading-lms-software, 3 2022. Accessed: 02/12/2024.

[12] elearning-2024. The top learning management systems. https://elearningindustry.com/directory/software-categories/learning-management-systems#the-best-learning-management-systems-top-list, 3 2024. Accessed: 02/12/2024.

[13] Mohammed Badawy, A Abd El-Aziz, and Heshan Hefny. Exploring and measuring the key performance indicators in higher education institutions. *International Journal of Intelligent Computing and Information Sciences*, 18(1):37–47, 1 2018. doi:10.21608/ijicis.2018.15914.

[14] Piotr Muryjas, Monika Wawer, and Magdalena Rzemieniak. Managing the process of evaluation of the academic teachers with the use of data mart and business intelligence. *European Research Studies Journal*, 24:127–140, 2021. doi:10.35808/ersj/2196.

[15] Roghaye Abbasi, Mostafa Ghaderi, and Hasan Malaki. Meta-analysis of the evaluation of teachers' performance and providing a suitable model for it. *Training & Learning Researches*, 17:73–85, 2022. doi:10.22070/tlr.2021.13559.1021.

[16] Davide Di Ruscio, Dimitris Kolovos, Juan de Lara, Alfonso Pierantonio, Massimo Tisi, and Manuel Wimmer. Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*, 21(2):437–446, 4 2022. doi:10.1007/s10270-021-00970-2.

[17] Hourieh Khalajzadeh and John Grundy. Accessibility of low-code approaches: A systematic literature review. *Information and Software Technology*, 177, 1 2025. doi:10.1016/j.infsof.2024.107570.

[18] Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio, and Alfonso Pierantonio. Supporting the understanding and comparison of low-code development platforms. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 171–178, 2020. doi:10.1109/SEAA51224.2020.00036.

[19] Rogers Kaliisa and Jan Arild Dolonen. CADA: a teacher-facing learning analytics dashboard to foster teachers' awareness of students' participation and discourse patterns in online discussions. *Technology, Knowledge and Learning*, pages 937–958, 9 2022. doi:10.1007/s10758-022-09598-7.

[20] Onur Karademir, Daniele Di Mitri, Jan Schneider, Ioana Jivet, Jörn Allmang, Sebastian Gombert, Marcus Kubsch, Knut Neumann, and Hendrik Drachsler. I don't have time! but keep me in the loop: Co-designing requirements for a learning analytics cockpit with teachers. *Journal of Computer Assisted Learning*, 5 2024. doi:10.1111/jcal.12997.

[21] Tesnim Khelifi, Nourhène Ben Rabah, Bénédicte Le Grand, and Ibtissem Daoudi. EX-LAD: Explainable learning analytics dashboard in higher education. In *EPiC Series in Computing*, volume 97, pages 38–51. EasyChair, 2024. doi:10.29007/dsxd.

[22] Mitchell Jhon Vásquez Bermúdez, Miguel Giovanny Molina Villacís, Karina Paola Real Avilés, and Manuel Valverde Minchalo. Design of a dashboard module in the moodle learning management system for activity tracking. *Ecuadorian Science Journal*, 8:14–21, 1 2025. doi:10.46480/esj.8.2.203.

[23] Sri Dewi Sugiyanti, Riya Widayanti, M Bahrul Ulum, Gerry Firmansyah, and Anik Hanifa Azizah. Design dashboard monitoring teacher performance assessment at cinta kasih tzu chi high school. *IAIC Transactions on Sustainable Digital Innovation (ITSDI)*, 4:46–56, 10 2022. doi:10.34306/itsdi.v4i1.569.

[24] Daniel Pérez-Berenguer, Mathieu Kessler, and Jesús García-Molina. A customizable and incremental processing approach for learning analytics. *IEEE Access*, 8:36350–36362, 2020. doi:10.1109/ACCESS.2020.2975384.

[25] Liuyue Jiang, Nguyen Khoi Tran, and Muhammad Ali Babar. Mod2dash: A framework for model-driven dashboards generation. *Proceedings of the ACM on Human-Computer Interaction*, 6:1–28, 6 2022. doi:10.1145/3534526.

[26] Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux, and Yves Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51:456–479, 2 2007. doi:10.1016/j.comnet.2006.08.008.

[27] Alain Abran, Adel Khelifi, Witold Suryn, and Ahmed Seffah. Usability meanings and interpretations in iso standards. *Software Quality Journal*, 11:325–338, 11 2003. doi:10.1023/A:1025869312943.

Amirhossein Badiei is a skilled software developer specializing in frontend development and Unity game development. He received his M.Sc. degree in Computer Engineering from the University of Isfahan, Isfahan, Iran, in 2024, and his B.Sc. degree in Computer Engineering from the Shiraz University of Technology, Shiraz, Iran, in 2020.

Mansooreh Ezhei is an Assistant Professor in the Software Engineering Department at the University of Isfahan. She holds a Bachelor's, Master's, and Doctorate in Computer Engineering with a focus on Software Engineering from the University of Isfahan, and she completed a postdoctoral fellowship at Isfahan University of Medical Sciences. Her research interests include text mining, social network analysis, medical data analysis, and deep learning.

Mohammadreza Sharbaf is an Assistant Professor in Computer Engineering at the University of Isfahan (UI). He is interested in Model-Driven Software Engineering, Collaborative Modeling, Low-Code Development Platforms, Software Development Methodologies, Design Patterns, and Semantic Web (Semantic Reasoning). His current research is focused on software testing, inconsistency management, and multi-view modeling. Mohammadreza received his B.Sc. from the Isfahan University of Technology, Isfahan, Iran, in 2013, and his M.Sc. and Ph.D from the UI, Isfahan, Iran, in 2016 and 2022, both in Software Engineering. Now, he is the director of the Model-Driven Software Engineering Research Group (MDSERG) at UI.