



# An Enrollment Dashboard to Reinforce Decision-Making for Students and Advisors

Noe Rivas<sup>1</sup> , Julià Minguillón<sup>1</sup> , and Jonathan Chacón-Pérez<sup>2</sup> 

<sup>1</sup> Universitat Oberta de Catalunya, Barcelona, Spain

{seosve, jminguillona}@uoc.edu

<sup>2</sup> Elisava, Barcelona, Spain

jchacon@elisava.net

**Abstract.** Over the past years, there has been a significant increase in the number of students attending online courses in higher education, in some cases due to the COVID-19 pandemic. Online higher education provides greater flexibility in selecting courses in the preferred order, but it also presents challenges due to the amount of different combinations during the enrollment process. In this paper, we describe a dashboard for supporting enrollment in higher education using a map as a visual metaphor, in order to provide students and their advisors with a clear overview of their situation within the degree, and help them to make better decisions. The enrollment dashboard has been evaluated through an initial series of interviews with advisors, which showed positive results in terms of usability and usefulness. However, several concerns were raised about the parameters used to create the map. These results will be used in future rounds of interviews with students and advisors.

**Keywords:** Recommendation system · enrollment · online higher education · advisors · data visualization · enrollment dashboard

## 1 Introduction

Most open universities offer flexible programs that allow students to choose from a wide range of courses according to both the semester organization and their personal interests. Although this can be seen as a convenient and flexible advantage for students, it also allows for inadequate enrollments, as students may find it challenging to sort through an overwhelming amount of information or may even find it insufficient to plan and select courses strategically [11, 12]. Some authors have explored the relationship between enrollment patterns and students' performance and dropout rates, since some combinations of courses may lead to failure [13]. Therefore, this hidden knowledge in institutional enrollment data could be utilized to support decision-making during enrollment, enabling students to be aware of their learning process in the forthcoming semester [8]. Data visualization, especially learning dashboards, have been used in educational

settings as a way to understand significant amounts of information. However, researchers often encounter difficulties in identifying the appropriate indicators to effectively apply data visualization in students' learning experiences [2, 10].

In [6], the authors described the typical uses of recommendation systems in higher education, namely e-learning, classroom activities and course selection. For instance, Karga and Satratzemi [3] designed a system that provides suggestions to enhance the quality of teaching practices and learning designs, while Zhang et al. [14] developed a recommendation system that maps students to supervisors based on quality, relevance, and connectivity criteria. A more specific study [5] compiled various examples of course recommendation systems based on principles and techniques from traditional ones. Nevertheless, although recommendation systems and data visualization are emerging fields of study within technology applied to education, there are only a few examples where both are incorporated into a learning dashboard (see Table 8 in [1]). One of the most remarkable can be found in CourseQ, an interactive course recommendation system [4], which allows courses to be explored using a visual interface, aiming to improve the transparency and user satisfaction of course recommendations. This paper presents a dashboard that enhances enrollment in higher education by providing students and advisors with an overview of their progress, facilitating informed decision-making.

## 2 Our Proposal

When students have a list of potential courses to enroll into, they often rely on various sources of information to make an informed decision. One such source is their academic advisor, who can provide guidance on which courses are most relevant to the student's academic and career goals. Additionally, students may consult other resources such as online course catalogs, syllabi, and course evaluations to gather information about course content, prerequisites, and workload [removed]. Based on this textual information, which can be very difficult to manage or even hard to find, students typically propose a set of courses to their advisor for validation. Unfortunately, some students do not make a good decision when planning their enrollment. The advisor may provide additional recommendations or suggest alternative courses based on the student's academic history, personal circumstances and goals. This process is an essential component of academic advising, as it helps students make informed decisions about their academic coursework and ensures that they are on track to meet their educational goals, avoiding inadequate course combinations.

Our proposed design is an enrollment dashboard based on a visual metaphor, specifically a map, in which the curriculum is depicted as a set of connected regions that must be visited in a specific order, separating courses that should not be taken at the same time. In order to do so, our dashboard supports the aforementioned enrollment procedure that students employ when making academic decisions. First, a map with the student's academic record is shown. Then, the student selects those courses that should not be recommended, such as validated courses. The student can distort the map according to her preferences,



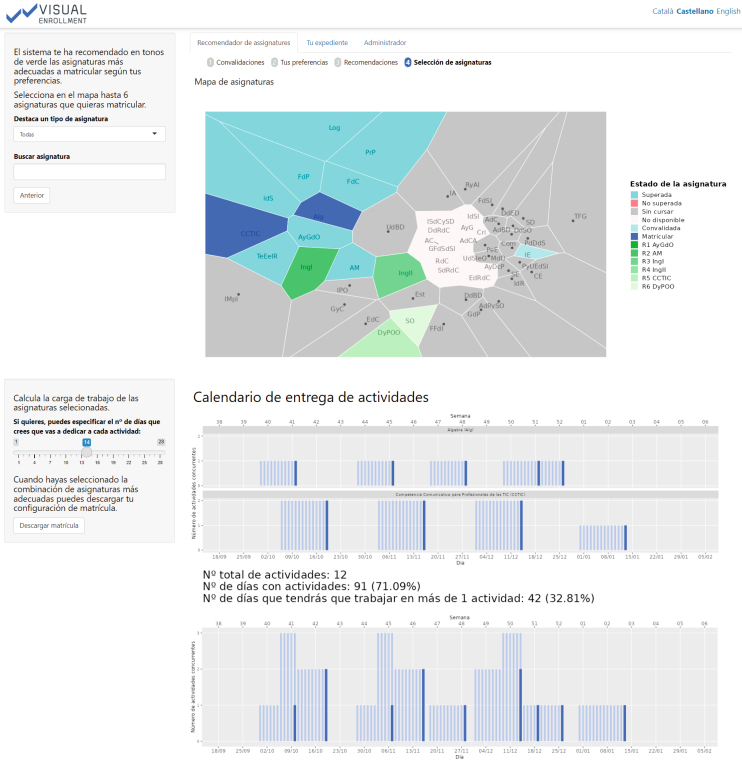
**Fig. 1.** Map showing a real student’s academic record. Completed courses are highlighted in blue, validated or discarded courses in light blue, failed courses in red, and potential courses to enroll in are displayed in gray. (Color figure online)

giving different weights to enrollment indicators [removed]. Once the map is accepted by the student, the dashboard recommends up to six courses, which can be selected or not by the student. For those courses selected by the student, additional information such as course calendar is shown. To avoid inappropriate enrollments, the map separates away those courses that should not be taken together, using the distance within the map as an abstract concept. This distance is a convex combination of different factors important for students [9], such as popularity, difficulty, prerequisites, and overlaps between course deadlines. The resulting distance matrix is then projected onto a 2D map [7], as shown in Fig. 1.

Once their preferences have been introduced by means of the sliders in Fig. 1, users (students and their advisors) can click on *Recommend courses* and the dashboard will highlight six courses in different shades of green, being the darkest the most recommended course to enroll. Notice that recommended courses are close to those courses that were already taken and passed, as if advancing in the degree was “conquering” new regions (i.e. courses) in the map, avoiding gaps. Then, users can select up to six courses (recommended or not). The selected courses are then marked in blue and additional information is provided, as shown in Fig. 2. Course assignments are displayed on the calendar, indicating the number of activities to be completed and the number of days with overlapping, which can be adjusted with an additional slider. This process can be repeated until the user considers the course selection appropriate.

### 3 Evaluation

Following the procedures described in some of the papers reviewed in [10], a prototype was initially tested by carrying out a pilot protocol for a single advisor (male, considered an expert advisor), including the following steps. First, the academic records of four different students were sent to the advisor, along



**Fig. 2.** Result of the selection of two courses (in dark blue) with their associated activity calendars. (Color figure online)

with a brief survey to find out which courses the advisor would recommend and why for each student, as well as the information taken into account when recommending courses. The second step involved conducting an interview, showing the enrollment dashboard to the advisor to gather his opinions on usefulness, effectiveness, usability, and overall understanding of the map metaphor. The advisor was asked about his opinion and whether there had been changes in his recommendation after viewing the dashboard or not.

This initial pilot study brought some interesting insights. Although the advisor had some difficulty in understanding certain aspects, such as the map visualization and the use of sliders, he found the visual recommendation (as shown in Fig. 2) very helpful. The possibility of viewing the student’s complete academic record as part of the map depicting the degree was also appreciated. Two important objectives were achieved. First, the enrollment dashboard was able to integrate a significant amount of information that was previously dispersed throughout various information sources or was hard to find. Second, the result of this evaluation allowed us to identify elements to be corrected in a following phase. After making some minor improvements to the dashboard, a

second round of interviews was conducted with eight advisors (four males, four females, all of them experienced). The protocol was changed so that the advisors themselves would use the tool instead of being presented by the interviewer. Quoting advisors' first impressions, the use of color for course status and recommendations was understood. The advisors considered course prerequisites a fundamental value, and they also valued the concept of overlapping deadlines. Adjusting the number of days until the deadline helped them to evaluate the real workload. They would like to use the dashboard in a real scenario, as it made them consider other enrollment options. Having access to the student's academic record was very appreciated. The advisors also suggested that the validated or discarded courses should be taken more into account as part of the recommendation. On the other hand, the concept of distance was not understood, and the courses were not easily found on the map. Advisors also mentioned that they would like to know the student's opinion before making a recommendation. The most interesting result was that in most cases (24 out of 32 times), advisors changed their opinion when viewing the map. However, in more than half of the cases (18 out of 32 times), they found the recommendation unsuitable because it did not give enough weight to course prerequisites. On the other hand, in most cases (26 out of 32 times), the deadline calendar was crucial to decide about the recommendation according to the enrollment total workload.

## 4 Conclusions

It must be stated that the proposed dashboard replicates a particular enrollment system. Therefore, it would not be suitable for other scenarios, with different curricula, enrollment procedures, or data collection methods. However, it should be noted that dashboards must often be adapted to the context for which they were created [10]. The initial pilot study provided valuable insights and identified areas for improvement, as advisors appreciated visual recommendations and suggested improvements regarding course prerequisites and recommendations. In general, the dashboard was well received as a tool to reinforce the current enrollment process. However, some aspects related to its usability and usefulness need to be better explained, such as the concept of distance embedded in the sliders and course prerequisites. For instance, sliders could be eliminated or replaced by values obtained from simulations with real data and the recommendation system could be also adjusted to give more weight to course prerequisites. Finally, a further iteration with students and their advisors will be carried out in order to evaluate the dashboard in a real enrollment scenario.


## References

1. Bodily, R., Verbert, K.: Review of research on student-facing learning analytics dashboards and educational recommender systems. *IEEE Trans. Learn. Technol.* **10**(4), 405–418 (2017). Conference Name: IEEE Transactions on Learning Technologies <https://doi.org/10.1109/TLT.2017.2740172>

2. Duval, E.: Attention please! learning analytics for visualization and recommendation. In: Proceedings of the 1st International Conference on Learning Analytics and Knowledge, LAK 2011, pp. 9–17. Association for Computing Machinery, New York (2011). <https://doi.org/10.1145/2090116.2090118>
3. Karga, S., Satratzemi, M.: Using explanations for recommender systems in learning design settings to enhance teachers' acceptance and perceived experience. *Educ. Inf. Technol.* **24**(5), 2953–2974 (2019). <https://doi.org/10.1007/s10639-019-09909-z>
4. Ma, B., Lu, M., Taniguchi, Y., Konomi, S.: CourseQ: the impact of visual and interactive course recommendation in university environments. *Res. Pract. Technol. Enhanc. Learn.* **16**(1), 18 (2021). <https://doi.org/10.1186/s41039-021-00167-7>
5. Maphosa, M., Doorsamy, W., Paul, B.: A review of recommender systems for choosing elective courses. *Int. J. Adv. Comput. Sci. Appl.* **11**(9) (2020). <https://doi.org/10.14569/IJACSA.2020.0110933>
6. Maphosa, V., Maphosa, M.: Fifteen years of recommender systems research in higher education: current trends and future direction. *Appl. Artif. Intell.* **37**(1), 2175106 (2023). <https://doi.org/10.1080/08839514.2023.2175106>
7. Minguillón, J., Rivas, N., Chacón, J.: Supporting enrollment in higher education through a visual recommendation system. In: Artificial Intelligence Research and Development: Proceedings of the 23rd International Conference of the Catalan Association for Artificial Intelligence, vol. 339, p. 177. IOS Press (2021). <https://doi.org/10.3233/FAIA210131>
8. Othman, M.H., Mohamad, N., Barom, M.N.: Students' decision making in class selection and enrolment. *Int. J. Educ. Manage.* **33**(4), 587–603 (2019). Publisher: Emerald Publishing Limited <https://doi.org/10.1108/IJEM-06-2017-0143>
9. Rivas, N., Minguillón, J., Chacón, J.: Enrolling habits in higher education. what sources of information do students have and what are missing? In: INTED2021 Proceedings, pp. 4980–4988. 15th Int. Technology, Education and Development Conf, IATED (8–9 March, 2021 2021). <https://doi.org/10.21125/inted.2021.1025>
10. Schwendimann, B.A., Rodriguez-Triana, M.J., Vozniuk, A., Prieto, L.P., Boroujeni, M.S., Holzer, A., Gillet, D., Dillenbourg, P.: Perceiving learning at a glance: a systematic literature review of learning dashboard research. *IEEE Trans. Learn. Technol.* **10**(1), 30–41 (2016). <https://doi.org/10.1109/TLT.2016.2599522>
11. Scott, M., Savage, D.A.: Lemons in the university: asymmetric information, academic shopping and subject selection. *High. Educ. Res. Dev.* **41**(4), 1247–1261 (2022). <https://doi.org/10.1080/07294360.2021.1887094>
12. Stevens, M., Harrison, M., Thompson, M.E., Lifschitz, A., Chaturapruek, S.: Choices, identities, paths: Understanding college students' academic decisions (2018). <https://doi.org/10.2139/ssrn.3162429>
13. Wladis, C., Wladis, K., Hachey, A.C.: The role of enrollment choice in online education: Course selection rationale and course difficulty as factors affecting retention. *Online Learn.* **18**(3) (2014). <https://eric.ed.gov/?id=EJ1043163>, publisher: Online Learning Consortium, Inc ERIC Number: EJ1043163
14. Zhang, M., Ma, J., Liu, Z., Sun, J., Silva, T.: A research analytics framework-supported recommendation approach for supervisor selection. *Br. J. Edu. Technol.* **47**(2), 403–420 (2016). <https://doi.org/10.1111/bjet.12244>



# A Moodle Plugin for Rich xAPI Data Logging

Daniela Rotelli<sup>1</sup> , Yves Noël<sup>2</sup>, Sébastien Lallé<sup>2</sup>, Vanda Luengo<sup>2</sup>,  
and David Pesce<sup>3</sup>

<sup>1</sup> Department of Computer Science, University of Pisa, Pisa, Italy  
[daniela.rotelli@phd.unipi.it](mailto:daniela.rotelli@phd.unipi.it)

<sup>2</sup> LIP6, CNRS, Sorbonne University, Paris, France

<sup>3</sup> Exputo Inc., Honeoye Falls, NY, USA

**Abstract.** The eLearning specification xAPI, which employs a shared format for receiving and transmitting data, is used to collect data about the diverse range of experiences within online learning activities, thereby enabling the exchange of knowledge between multiple systems. This paper presents *Logstore xAPI*, a plugin that emits Moodle events as xAPI statements, allowing a modular, interoperable, performing and secure way of logging user interactions and learning experiences, and send them to a Learning Record Store (LRS) to be stored and further analysed. We describe all phases of mapping Moodle events to statements and storing them in the LRS, along with the issues we encountered and our solutions.

**Keywords:** xAPI · Moodle · Learning Analytics · Educational log data

## 1 Introduction and Related Work

The processing and analysis of educational data have fostered a significant expansion of knowledge discovery approaches and created new opportunities for data-driven support and evaluation of learning practices. To facilitate the dissemination of relevant educational data and their interoperability across learning platforms, standardised data formats have emerged, with xAPI being the most recent and versatile standard [5]. However, the existing LMSs typically use ad-hoc data format that makes it harder to implement, replicate and generalise educational data analysis. Indeed, the migration from ad-hoc data format to the xAPI format remains a time-consuming and challenging task, especially in complex systems such as LMSs where numerous actions and activities are possible.

Moodle is one of the most popular LMS, with over 350 millions unique users as of 2023 (*stats.moodle.org*). Despite the steep learning curve when it comes to the Moodle database [6], Moodle logs have been extensively used in Learning Analytics research. Numerous studies have been conducted on Moodle logs to understand how learners organise their learning time [4], as well as to model their learning [2], academic performances [1], and engagement [8], among other traits. However, to the best of our knowledge, these works did not rely on a

standardised data format, and rather directly queried or exported logs from the Moodle's tables, which is a threat to generalisability and privacy. In this paper, we make a further step toward standardisation by proposing *Logstore xAPI*, a plugin to convert Moodle logs into xAPI statements ([shorturl.at/zTV23](http://shorturl.at/zTV23)).

Logstore xAPI processes and stores the xAPI statements derived from Moodle logs into a Learning Record Store (LRS), where data from other platforms can be integrated as well in the same format, thus maximising interoperability. Logstore xAPI also augments the statements with information about the user roles and course content, which is key for contextualising the data when they were generated. Logstore xAPI provides options to anonymise the logs, and can be used both live and on historical data. While there has been another attempt at xAPI logging for Moodle, TRAX Logs plugin is currently still in the alpha version and the developers advise against using it in production [3]. Moreover, they only supports a few statements, whereas we handle all of the 216 possible student interactions. Furthermore only our plugin can handle historical data, which is key for learning analytics. As a test-bed for testing Logstore xAPI, we have deployed it in a major French university with over 25,000 students, and we report on the plugin's performances and reliability.

## 2 Moodle Logs

Moodle stores user activity in a relational database ([moodleschema.zoola.io](http://moodleschema.zoola.io)) When a user access a course, submit an assignment, take a quiz, or read/write posts in a forum, a log of that activity is recorded in the *mdl.logstore\_standard\_log* table (*logstore* in the following) of Moodle database. Recorded log data can be extracted in two ways, albeit both are specific to Moodle and suffer from important drawbacks.

**Database.** Stored log data can be extracted by directly querying the *logstore* table that is supplied with 21 fields, among which: *eventname*, *component*, *action*, *target*, *objecttable*, *objectid*, *contextid*, *contextinstanceid*, *userid*, *courseid*, *relateduserid*, *timecreated*. Originally, Moodle logs were not intended for data mining, but rather as a necessary part of maintaining the LMS. Therefore, despite being exhaustive, they are unintelligible to humans and lack context. To make them useful for analysis, it is necessary to perform multiple joins with other database tables, which requires in-depth knowledge of the Moodle system.

**Log Generation Interface.** Logs stored in the *logstore* table can also be extracted via the *log generation interface* [7] and converted into a table with 9 features: *Time*, *User full name*, *Affected user*, *Event context*, *Component*, *Event name*, *Description*, *Origin*, *IP address*. Despite being human-readable, this table lacks information about the course and the role assigned to the user in a specific context [7]. In addition, as the number of logs increases, PHP's memory size limit must be enlarged so that the table can be downloaded for analysis.

Importantly, the *logstore* table can be periodically purged. This occurs more frequently on larger Moodle instances and, as a result, direct *logstore* querying could not provide a complete data sample. Moreover, both approaches require



admin privileges in Moodle, which is not always possible nor advisable in educational institutions, due to safety and privacy issues. The Logstore xAPI plugin we propose in this work alleviates the aforementioned issues, as it produces rich and comprehensive real-time data in an interoperable format, without needing admin rights, as described next.

### 3 The Logstore xAPI Plugin

Logstore xAPI retrieves logs from the *logstore* table, stores them in the *mdl\_logstore\_xapi\_log* table (*logstore\_xapi* in the following) to produce xAPI statements, and emits them to a Learning Record Store (LRS). The LRS (*shorturl.at/bpFX1*) is the central repository of any xAPI ecosystem, receiving, storing, and providing data on learning experiences, achievements, and performance from a range of systems. To avoid blocking page responses, the plugin operates in the background by default (this can be changed in the settings) via Cron tasks. While this makes the process less real-time, it prevents Moodle’s performance from fluctuating based on the performance of the LRS. The endpoint, key/username, and password to the LRS must be specified in the configuration page after installation. The plugin is currently compatible with Moodle versions 3.9 to 4.1.

#### 3.1 Plugin Architecture

The plugin is made up of three parts, an *Expander*, a *Translator*, and an *Emitter* (Fig. 1). Every log entry goes through each of these parts before finally reaching the LRS, as follows.

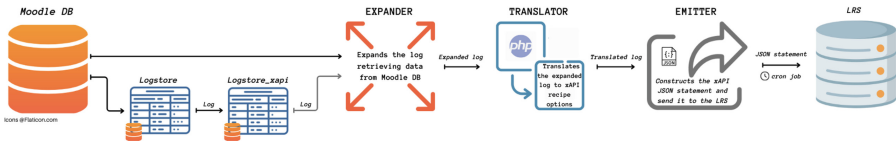


Fig. 1. The plugin architecture.

**Expander.** The Expander augments the log entry with data from Moodle database and passes them to the Translator. This step is key to retrieve the context related to the log entry. Information about the context is retrieved from both the *logstore* table and all the tables referenced in the event. For example, in Table 1, to retrieve the *actor* whose *userid* is 24, the plugin queries the *mdl\_user* table to retrieve all associated information. The *mdl\_forum\_discussions* table and the *mdl\_course\_modules* table (*contextinstanceid*=62) contain the entirety of the information relevant to the discussion *object* (*objectid* = 4). The *context* of the discussion is the course and the forum in which the discussion is created. Course information is retrieved from the *mdl\_course* table (*courseid* 38), whereas forum information is retrieved from the *mdl\_forum\_discussions* table, where *objectid*=4 corresponds to *forumid*=3, whose information is retrieved from *mdl\_forum* table.

**Table 1.** Entry example in the *logstore* table.

eventname	target	objecttable	objectid	contextinstanceid	userid	courseid
mod_forum\ discussion_created	discussion	forum_discussions	4	62	24	38

**Translator.** The Translator converts the expanded logs to xAPI using a set of xAPI “recipes”, i.e., standard ways of expressing a particular event. A statement must include at least three properties: an *actor*, a *verb*, and an *object*, along with additional optional properties such as the *id*, *context*, and *timestamp* ([shorturl.at/mzMU9](http://shorturl.at/mzMU9)). It is worth noting that xAPI provides a predefined list of verbs and objects to enforce a shared terminology across learning platforms ([shorturl.at/iluT0](http://shorturl.at/iluT0)). Our plugin currently implements xAPI recipes for 216 events, thus accounting for all *participating* events (i.e., related to student learning experience) performed in Moodle standard and some external plugins such as BigBlueButton, Questionnaire, and Scheduler ([moodle.org/plugins](http://moodle.org/plugins)).

**Emitter.** The Emitter constructs the translated event into an xAPI JSON statement and emits it to the LRS. The event is removed from the *logstore\_xapi* table once it has been sent. If something goes wrong and the translated event is not sent to the LRS (e.g., because of connection issues), it is moved to the *mdl\_logstore\_xapi\_failed\_log* table (*xapi\_failed* in the following).

### 3.2 xAPI Implementation

To transform logs in xAPI statements, information is retrieved from both the *logstore* table and all the tables referenced in the event. For example, in Table 1, to retrieve the *actor* whose *userid* is 24, we query the *mdl\_user* table to retrieve all associated information. The *mdl\_forum\_discussions* table and the *mdl\_course\_modules* table (*contextinstanceid*=62) contain the entirety of the information relevant to the discussion *object* (*objectid*=4). The *context* of the discussion is the course and the forum in which the discussion is created. Course information is retrieved from the *mdl\_course* table (*courseid* 38), whereas forum information is retrieved from the *mdl\_forum\_discussions* table, where *objectid*=4 corresponds to *forumid* = 3, whose information is retrieved from *mdl\_forum* table. The transcription of the statement is accessible on GitHub ([shorturl.at/eoRY1](http://shorturl.at/eoRY1)).

### 3.3 Historical Events

The plugin is not natively designed to handle historical events that occurred prior to its installation into Moodle. These events can however be copied from the *logstore* table to the *logstore\_xapi* table, e.g., via a SQL query. The plugin will then process them whenever the Cron script is executed (Sect. 3). The main challenge with historical events is that part of the information might be missing in the database due to a deleted course, activity, module, or user. Except for the user, Moodle does not keep track of deleted entries. During the development

of the plugin, this caused a number of failures, resulting in the transfer of the events in the *xapi\_failed* table. We thus handled these issues as follows.

*Deleted Users.* When users are deleted in Moodle, they are not erased from the database; instead, the ‘deleted’ field in the *mdl\_user* table changes from 0 to 1. To account for the user deletion, we use the value ‘deleted’ as their fullname.

*Deleted Courses, Modules and Related Activities.* Courses and modules can be deleted from Moodle, along with all related data (e.g., grades, groups). However, logs of actions performed by users on deleted courses and modules are still available in the *logstore* table. This means that attempting to retrieve data from the database about that deleted course, module or their related activities to enrich the statements will fail. To avoid losing these events, we add the ‘deleted’ value to the description of these items in the xAPI statements. We do the same for the deletion of a group, a grade, a forum post reply, or a message.

We wish to emphasise that with the log generation interface (Sect. 2), it is not possible to identify deleted users, but actions performed on deleted modules or courses can be identified looking at the “Event context”, which is set to *Other*. These two elements of information are inaccessible in database-extracted logs. Our plugin thus allows retrieving more than the simple interaction.

### 3.4 Privacy

The data sent to the LRS include information about the users, their profile, and details about completed courses. Therefore, if Moodle and the LRS are not within a private network, a secure connection between them is always advised. To answer the challenge of protecting students’ privacy and enhancing reproducibility by permitting data sharing, we introduced the option to pseudonymise user data (i.e., userid, username, email, and user fullname). To enable this option, the user must select to hash data and specify a secret key in the plugin configuration so that data will be hashed with the secret key before being sent to the LRS.

## 4 Plugin Application

To assess the plugin, we deployed it in the Moodle installation of Sorbonne University, and we report on the plugin’s reliability and efficiency. To demonstrate quantifiable elements of what we performed and to evaluate the management of deleted elements, we used historical data (Sect. 3.3).

The *logstore* table contained 89,865,813 entries of the activities performed by 25,386 users over the course of a year. To copy all entries to the *logstore\_xapi* table required approximately two days (500,000 every 15 min). To process data and send them to the LRS, the plugin required approximately 6 d (5,000 every 30 s). We did not get any failed logs, which shows in particular that our solutions to handle missing and deleted events (cf. Section 3.3) was suitable. Since the logs are processed every minute via Cron, it can be assumed that even if there was an overload of approximately 500,000 logs in real-time during the day, these would be processed within an hour overnight.

## 5 Conclusion and Future Work

In this paper, we proposed *Logstore xAPI*, a plugin that converts Moodle logs into xAPI statements to be sent to an LRS, where data from other platforms can be integrated as well in the same format, thus maximising interoperability.

Currently, we have focused on all events relating to student actions (*participating* level - Sect. 3.1), which was a prerequisite for analysing student behaviour. As future work, we intend to include all *teaching* level activities and events that Moodle categorises as *other* level activity, as well as all the statements we were unable to transcribe because either the *verb* or the *object* was absent in the Registry. In an effort to make educational data more accessible and to comply with GDPR, we would like to give the users the option to retrieve only the data that they specifically request. Thus, we intend to include an option to specify which types of data to extract.

In the long run, we aim to support researchers and data scientists in analysing Moodle logs, as well as integrating them with data from other educational systems typically used in conjunction with LMSs, e.g., video platforms, serious games. This is thanks to the standardised, interoperable, and well-documented xAPI data format outputted by Logstore xAPI.

**Acknowledgements..** We thank Capsule (*capsule.sorbonne-universite.fr*), the pedagogical innovation center of Sorbonne University, for the data collection. This work has been partially supported by “SoBigData++: European Integrated Infrastructure for Social Mining and Big Data Analytics” and by PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - “FAIR - Future Artificial Intelligence Research” - Spoke 1 “Human-centered AI”, funded by the European Commission under the NextGeneration EU programme.

## References

1. Ademi, N., Loshkovska, S., Kalajdziski, S.: Prediction of student success through analysis of moodle logs: case study. In: Gievska, S., Madjarov, G. (eds.) ICT Innovations 2019. CCIS, vol. 1110, pp. 27–40. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-33110-8\\_3](https://doi.org/10.1007/978-3-030-33110-8_3)
2. Ikawati, Y., Al Rasyid, M.U.H., Winarno, I.: Student behavior analysis to predict learning styles based felder silverman model using ensemble tree method. Int. J. Eng. Technol. (2021)
3. Judel, S., Schnell, E., Schroeder, U.: Performantes xapi logging in moodle. 20. Fachtagung Bildungstechnologien (DELFI) (2022)
4. Maslennikova, A., Rotelli, D., Monreale, A.: Visual analytics for session-based time-windows identification in virtual learning environments. In: 26th IV. IEEE (2022)
5. Nouira, A., Cheniti-Belcadhi, L., Braham, R.: An enhanced xapi data model supporting assessment analytics. Procedia Computer Science (2018)
6. Romero, C., Romero, J.R., Ventura, S.: A survey on pre-processing educational data. In: Peña-Ayala, A. (ed.) Educational Data Mining. SCI, vol. 524, pp. 29–64. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-02738-8\\_2](https://doi.org/10.1007/978-3-319-02738-8_2)

7. Rotelli, D., Monreale, A.: Time-on-task estimation by data-driven outlier detection based on learning activities. In: LAK (2022)
8. Rotelli, D., Monreale, A., Guidotti, R.: Uncovering student temporal learning patterns. In: EC-TEL 2022. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-16290-9\\_25](https://doi.org/10.1007/978-3-031-16290-9_25)