Recurring learning activity planning model

Ion Alexandru POPESCU

Pitesti University Center, University of Science and Technology Politehnica Bucharest, Bucharest, Romania

alexionpopescu@gmail.com

Abstract: *The paper proposes a model for recurrent planning of learning activities from an academic course or school lessons. This model can be adapted for other types of learning activities. Databases and text/pdf files were used to store the information used. The model uses two main components, one for configuring learning activities: calendar dates, time of occurrence, online connection links, necessary auxiliary materials and a second component for generating the interface of a particular learning activity: course, laboratory, seminar, lesson. Modern web programming technologies were used to implement this model.*

Keywords: learning, webpage, modelling, database, interface.

1. Introduction

Nowadays, teachers prepare their courses in detail to make their teaching work easier and to facilitate easier and more assisted learning for students. Thus, in the academic environment, the teacher and students must have the working and presentation materials at hand – in a format that is as predictable and attractive as possible. This activity is presented in (Early Education Nation, 2024) and (Homeschool Solutions, 2024). The schedule of learning activities is repetitive week after week and for this reason recurrent planning is a mechanism often used by teachers. At the beginning of the course period, the teacher prepares his/her activities and teaching materials and may have his/her own style of planning, different from those proposed by existing e-learning platforms such as MOODLE or GOOGLE CLASSROOM, aspects presented in (Hamidi et al., 2011) and (Popescu et al., 2020). Also, similar approaches for scheduling are presented in (Bradley, 2021) and (Seman et al., 2018) or testing in (Popescu et al., 2016) and (Popescu et al., 2023).

The model that we will present in this paper uses databases to store information and web pages for the learning activity interfaces. For entering data into databases, secure connection, setting activity data, generating connection links for communication and creating work interfaces we will use scripts – in this way the model can be implemented through an easy-to-manage web application.

The paper is fragmented into sections for a modular and fluent presentation, to quickly understand the ideas of the model and the implementation method. In the

https://doi.org/10.58503/icvl-v20y202535

final part of the paper, the features of the model and what activities can be integrated to extend the model and make it more efficient are presented.

2. Model presentation

The recurring learning activity planning model has two main components:

- ReConfig (Recurring configurator),
- UIGen (User Interface Generator)

ReConfig takes the course-related data from the professor and enters it into a database. Among the data taken from the professor are the day and time of the week in which the learning activity will take place, the materials (files, images, presentations, etc.) usable in all courses.

UIGen generates recurring activities for all weeks between certain time limits taken from the professor along with other restrictions. Also in this component, the materials specific to each week of the course (online connection links, files specific to certain courses, specific topics, etc.) are entered, as well as configurations of the interface specific to each week, if applicable.

Figure 1 shows the diagram of the recurring planning model that contains the two components presented previously.

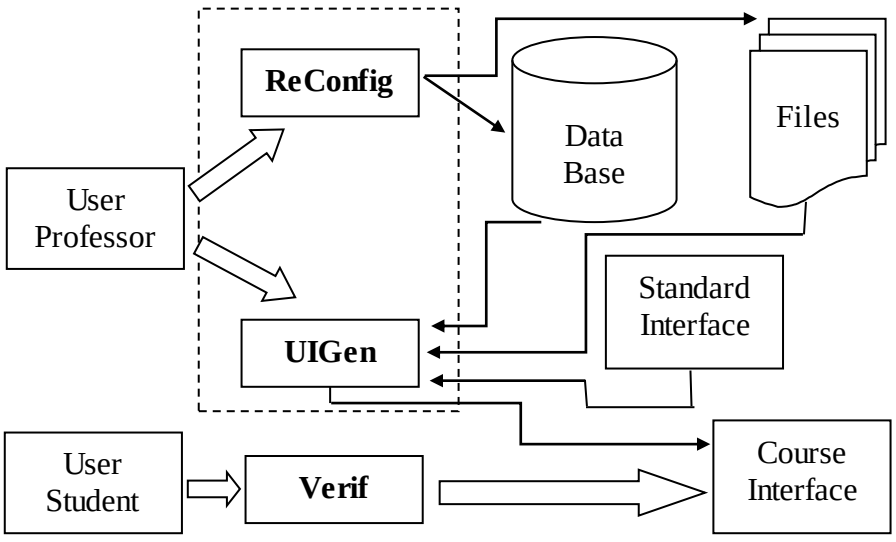

Figure 1. Recurring Learning Activity Planning Model

For access to the ReConfig and UIGen components, access is only for the professor, and for students, access is verified using the Verify component. Students only have access to the course interface which is generated recurrently by UIGen

using a Standard Interface and populated with course-specific data. The Standard Interface comprises the presentation of the scheduled activities to all users (e.g., teachers, students) and it is generated based on the database information.

The model presented in the previous paragraphs can be mathematically modelled based on a set of activities that have specific characteristics and must be programmed in a defined time interval. Let there be a set of learning activities:

$$A = \{a_1, a_2, \dots, a_n\}$$

Each activity a_i is characterized by:

- predefined scheduling (S_i):

$$S_i = \{(d_{i,1}, t_{i,1}, l_{i,1}), (d_{i,2}, t_{i,2}, l_{i,2}), \dots, (d_{i,k}, t_{i,k}, l_{i,k})\}$$

where $d_{i,j}$ is the day scheduled for the activity a_i for the j^{th} repetition, $t_{i,j}$ is the starting hour and $l_{i,j}$ is the duration of the activity;

- recurrence frequency $f_i \in N$ is a number of time interval (e.g, an activity a_i is repeating after 1 week, thus $f_i = 1$);
- files and resources

$$R_i = \{r_{i,1}, r_{i,2}, \dots, r_{i,m}\}.$$

A scheduling interval T is set for the activities, T = [TS, TE], where TS is the start time and TE is the end time for the period desired to generate the activities. In this matter, each activity a_i with $(d_{i,j},t_{i,j},l_{i,j}) \in S_i$ must respect the condition:

$$TS \leq d_{i,j} \leq TE, \forall i \in \{1, \dots, n\}, \forall (d_{i,j}, t_{i,j}, l_{i,j}) \in S_i$$

For each activity a_i with $(d_{i,j},t_{i,j},l_{i,j}) \in S_i$, with k reccurences, each recurring instance is determined by the frequency f_i :

$$(d_{i,j}, t_{i,j}, l_{i,j}) = (d_{i,0} + k \times f_i, t_{i,0}, l_{i,0}), j \in \{0,1, \dots, k\}$$

To ensure that activities do not overlap on the same day, it is necessary to:

$$(t_{a,j} + l_{a,j} \leq t_{b,j'}) \vee (t_{b,j'} + l_{b,j'} \leq t_{a,j}), \forall a \neq b, \forall j, j' such\ d_{a,j} = d_{b,j'}$$

This constraint guarantees that, for any two activities scheduled on the same day, the time slots do not overlap.

This mathematical model provides a clear structure for planning recurring activities, integrating the elements of predefined schedules, frequency, resources, and scheduling period. It serves as a theoretical basis for developing an application to automatically manage schedules and resources needed in the educational environment.

The current implementation may serve as a base for developing a generator of a schedule of recurring activities where the activities have not a fixed date and

time of progress. In this matter, the model would take into account a list of activities, their duration, the frequency and the start and end time interval in weeks (e.g., Week 1, Week 2 etc.). Thus, the problem of the schedule generation transforms into an optimisational one, taking into account a minimum number of overlapping activities that form a list. The model would comprise the next elements:

- the start (TS) and end (TE) times of a time interval desired for the scheduling problem, stated as weeks (e.g., Week 1 with TS = 1, Week 14 with TE = 14);
- the list of activities:

$$A = \{a_1, a_2, \dots, a_n\}$$

- the fixed duration of an activity $l_i \in R^+$;
- the frequency of the activity $f_i \in N$, a number of time interval (e.g, an activity a_i is repeating after 1 week, thus $f_i = 1$);
- the number of instances of the activity n_i, computed as the report between the difference between TS and TE and the frequency of the activity f_i;
- the hourly starting point in time of each activity H_i;
- the day of week of each activity $D_i \in \{1, 2, \dots 7\}$, representing the weekdays.

Then, for each week S_j, $j \in \{TS, \dots, TE\}$, the total number of overlapping is computed:

$$C(S_k) = \sum_{i,j \in S_k, i \neq j} O_{i,j}, O_{i,j}$$

$$= \begin{cases} 1, if\ (D_i = D_j) \wedge (H_i < H_j + l_j) \wedge (H_j < H_i + l_i) \\ 0, otherwise \end{cases}$$

Then, using a Greedy strategy, the week with the most overlaps (maximum of $C(S_k)$) is selected. For this week, an optimisational algorithm may be used. For example, for a genetic algorithm, a chromosome is presented as the list of activities in the week k:

$$X = [(D_1, H_1, l_1), \dots, (D_n, H_n, l_n)]$$

The objective function would follow the minimisation of the overlapping of the activities represented as triplets in the week k, which would be the minimisation of the function $C(S_k)$. Then, a genetic algorithm would be applied in order to minimise the value of the function C.

The implementation of this model optimizes activity scheduling by reducing overlaps and accelerating computation, as the genetic algorithm only acts on the most problematic week. Greedy selection prioritizes conflicts, and crossover and

mutation allow for rapid adjustments. The model is scalable, flexible, and easy to extend with additional constraints, such as room or teacher allocation.

Overall, extending the initial model with a genetic algorithm allows for optimizing the distribution of scheduled activities, reducing overlaps and maximizing planning efficiency. The algorithm can adjust the allocation of resources (rooms, teachers) and balance the timetable load, avoiding inefficient scheduling.

3. Implementation

The recurrent learning activity planning model presented in the previous section was implemented using the following web technologies:

- Laravel 11 (PHP) - was used to develop the backend component, providing a scalable and robust base to manage the logic part of the application, the interaction with the database, as well as ensuring secure communication between the server and the client;
- Vue.js 3 - was used to develop the frontend component, providing a flexible and reactive framework to create dynamic and user-friendly interfaces. Its component-based structure ensures modularity of the user interface but also easier maintainability;
- Mysql 8 - the relational database that was used to store the application data. Due to its advanced functionalities, such as JSON support, but also the improved indexes mechanisms, it allowed the creation of efficient queries for scheduling activities.

Laravel 11 was mainly used for:

- providing API endpoints for the frontend
- interacting with the database to store and retrieve information efficiently
- managing activities by implementing their storage logic

Vue.js was mainly used for:

- designing an interactive interface for users, where they can create, edit, and view recurring learning activities
- implementing real-time validations of activities
- using Vuex for state management to ensure a seamless experience

Figure 2 shows the interface for the activities calendar. As we can see in the calendar, on a given day, the group of students who are going to have the lesson at the mentioned start time is marked. We have several colors present, these represent:

- green for a lesson that will take place online
- gray for a lesson that will take place physically (offline)
- red-gray represents a lesson that has been canceled (for example, for the image, 3 lessons were scheduled for that group but the one on January 3rd could not be held, so it is marked as canceled).

In Figure 2 we have the weekly version of the calendar where we can see in more detail how long each lesson lasts (the block associated with the lesson stretching over the duration range).

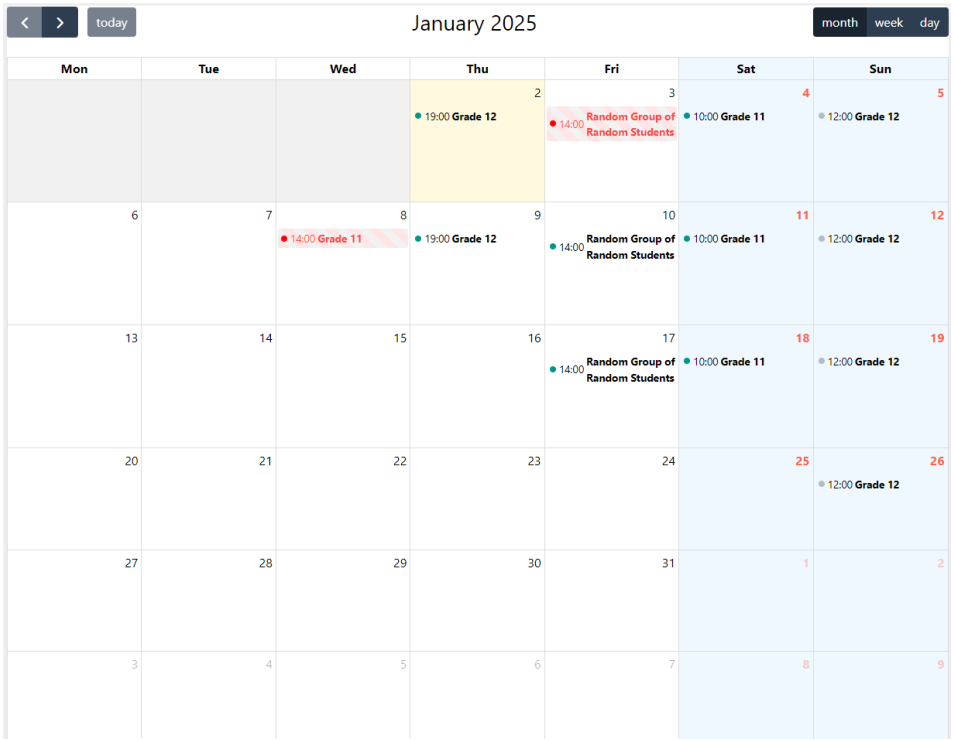

Figure 2. Recurring Learning Activities Calendar (monthly view)

To achieve these things in the interface, the Fullcalendar component from Vue.js was used.

Figure 3 shows the interface for the form for adding an activity. The fields in the form represent:

- group - the group for which the activity is planned
- day of the week - the day of the week it will take place (1 for Monday, 2 for Tuesday, etc.)
- start and end time - the interval in which it will take place

- start date - the date it will start
- occurrences - the number of weeks in which it will take place (for example 4 to have each week for a month), and the end date (end date) will be automatically filled in based on the chosen value
- online lesson - a simple checkbox to determine whether the activity is online or not

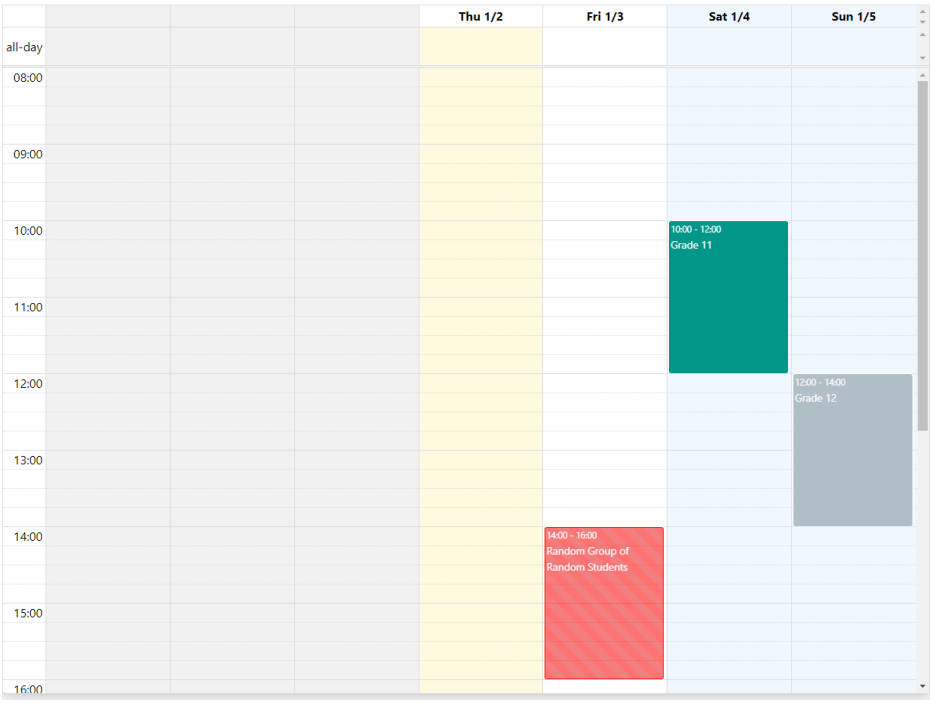

Figure 3. Recurring Learning Activities Calendar (weekly view)

Once completed, the form in figure 4 is first checked by the interface (the end time should not be before the start time as well as other validations), then this data is sent to the backend via a request. Once they arrive here, they are checked again much more strictly (there are validations to avoid overlapping activities, for example), and then if everything is considered in order, they are stored in the database.

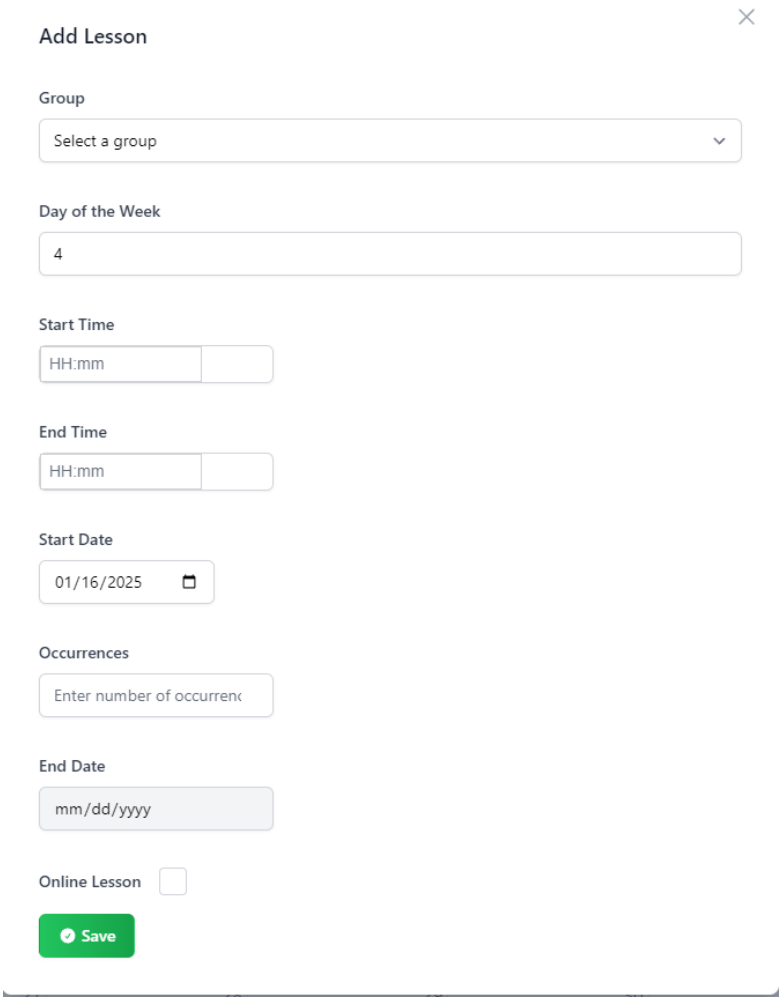

Figure 4. Recurring Learning Activities Calendar (activity add form)

4. Conclusions and future work

The Recurring Learning Activity Planning model contains the mechanisms necessary to plan a series of activities using various information resources in a simple and intuitive presentation. This model can be integrated into e-learning tools and platforms at the institutional level, allowing the use of specific configurations. In the coming period, we want to use the model in a complex teaching - learning - assessment application, such as those presented the mentioned research papers, and to develop it to show its efficiency. At the same time, recurring tasks and assignments for students can be planned or testing methods can be used, like in those specified in the mentioned similar works. Also, the implementation of an additional feature of an alternative scheduling configuration using genetic

algorithm may lead to improvements. Extending the original model with a genetic algorithm allows for the optimization of activity scheduling, modifying hours and days to minimize overlaps and create a more balanced schedule. By applying selection, mutation, and crossover, the algorithm can find more efficient distributions of activities, respecting the imposed constraints and maximizing the use of available time.

REFERENCES

Bradley, V. M. (2021) Learning management system (lms) use with online instruction. *International Journal of Technology in Education*. 4 (1), 68–92.

Early Education Nation (2024) *How to Plan a Lesson or Activity - Activity Planner*, https://support.lillio.com/s/article/director-Creating-a-lesson-on-the-Activity-Planner [Accessed 2th January 2025].

Hamidi, F., Meshkat, M., Rezaee, M., Jafari, M. (2011) Information Technology in Education. *Procedia Computer Science*. 3, 369-373. doi: 10.1016/j.procs.2010.12.062

Homeschool Solutions (2024) *Creating a Lesson Plan with Repeating Pattern*, http://help.homeschooltracker.com/LP_Intervals.aspx, [Accessed 2th January 2025]

Popescu, D. A., Bold, N., Nijloveanu, D. (2016) A Method Based on Genetic Algorithms for Generating Assessment Tests Used for Learning. *Journal Polibits*. 54, 53-60. doi: 10.17562/PB-54-7

Popescu, D. A., Cristea, D. M., Bold, N. (2023) *A Transfer Learning Approach Interaction in an Academic Consortium*, *22nd ICWL 2023: Sydney, NSW, Australia*. pp. 204-219.

Popescu, D. A., Tița, V. & Bold, N. (2020) Model of online course activities management. *9-th International Workshop on Soft Computing Applications (SOFA)*. pp. 153-158.

Seman, L. O., Hausmann, R., Bezerra, E. A. (2018) On the students' perceptions of the knowledge formation when submitted to a project-based learning environment using web applications. *Computers Education*. 117, 16–30. doi: 10.1016/j.compedu.2017.10.0