Personal Learning Records

Design Report

Vladislav Avramov - s2292165

Plamen Bozov - s2319217

Venelina Pocheva - s2298805

Bartosz Prządka - s2096757

Aleksandra Siderova - s2249987

Abstract

In this report, the overall design process for the Personal Learning Records system, supervised by Wallace Corbo Ugulino, is explained in detail. In particular, the topics discussed include the scope of the project and its stakeholders, the system requirements, the design and implementation process, as well as risk assessment and limitations.

List of Acronyms

PLR	Personal Learning Records
MH	Must-have requirement
NTH	Nice-to-have requirement
GDPR	General Data Protection Regulation
UT	University of Twente
UML	Unified Modeling Language
BIT	Business Information Technology
ICT	Information & Communication Technology
LISA	ICT, Library & Archiving Services
LMS	Learning Management System

1. Introduction

In order to support the study progress of students at the University of Twente (UT), a system for self-assessment called the Personal Learning Records (PLR) was developed under the guidance of our client and supervisor, dr. Wallace Corbo Ugulino, and is currently being utilized by the Business Information Technology (BIT) study at the university, with the future possibility of being used in other technical and mathematical courses in universities all around the world. The system is used by students to track their own progress, as well as by their mentors and teachers in order to provide supervision and feedback with the aim of guiding students through the learning process.

However, the system in its current state is difficult to use in multiple aspects - it has excessively long loading times, it contains many bugs, it lacks certain crucial features and it is built using PHP, which makes it difficult to maintain by our client. As such, the task of this project is to recreate the system from scratch with the above mentioned issues resolved and new features implemented.

2. Project Scope & Stakeholders

2.1. Project Scope

The base implementation of the project involves the recreation of the features of the already existing PLR system. To begin with, the system must include a secure login and four separate views for each role - student, mentor, teacher and administrator.

Students have the task of filling in their perceived experience and proficiency level for a number of topics that are part of the course they are currently taking. They can select between three levels in ascending order of proficiency - Entry, Intermediate and Target. Additionally, students can view mentor- or teacher-submitted related materials for a topic if they need help understanding it.

Mentors are responsible for specific students or groups of students in a course. As such, mentors are able to track the progress of students and give feedback on their progress. They can also add and remove students from their assigned groups and upload the above-mentioned related materials.

Teachers and administrators are able to carry out many of the same tasks. On top of what mentors are capable of, teachers and administrators can additionally create users and groups, as well as modify members of all groups of a course. The main difference between the two is that access for teachers is restricted to the courses that they have been assigned to, while the administrators have access to all users, courses, topics and materials.

The functionalities described above are already part of the current PLR system and in this project they have been recreated in Java rather than PHP. On top of those processes, this project has also added additional features, among which are notifications received by mentors and teachers when a student completes a self-assessment, notifications for students when they receive feedback, as well as the possibility of using Canvas-generated CSV files to create students for a course.

2.2. Stakeholder Analysis

The primary stakeholders, those for whom the system has the greatest impact, are the direct users of the system - generally speaking, those are students, mentors and teachers belonging to courses that utilize the system. For the time being, those are users belonging to the first two modules of the BIT study at the UT, however, it is worth keeping in mind that our client intends for the system to be used for other technical and mathematical courses at the UT, as well as at universities all over the world in the future.

Stakeholders with a more indirect connection to the project, include some of the remaining staff at the university (for the time being only the UT), such as the module coordinators, programme directors, the university ICT staff (such as LISA at the UT), etc. However, due to their indirect involvement in the project, they can be considered to be of lower priority than the stakeholders outlined in the previous paragraph.

The prioritization of stakeholders for this project has been performed using the Power/Interest grid [7] and can be seen in Figure 1. In the grid, stakeholders are placed in different quadrants based on their influence over the project, as well as their interest in the final result, with the corresponding recommended management and communication approach.

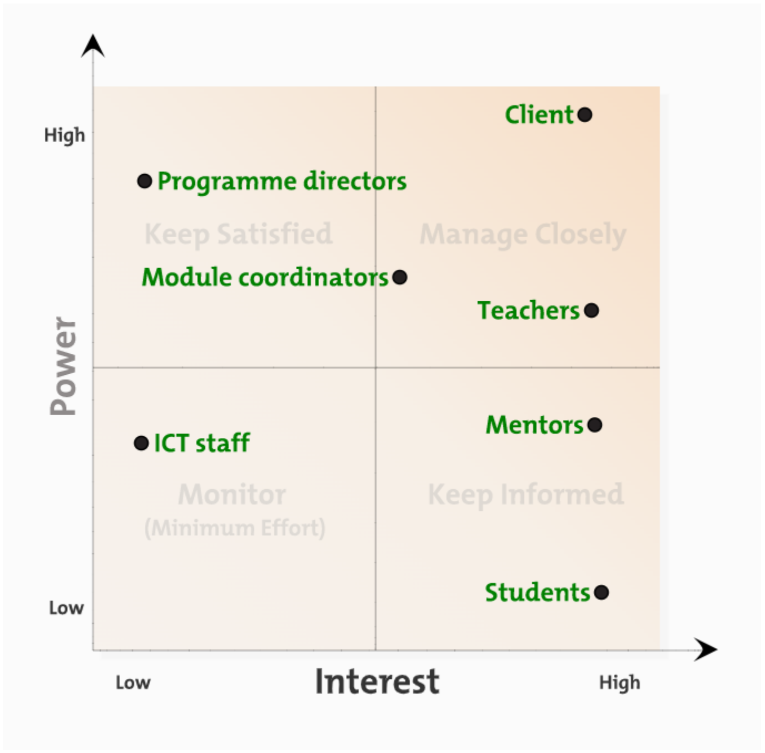


Figure 1. Stakeholder prioritization

3. Requirements

At the start of our designing process, the following requirements were gathered with the approval of our client, which have been divided into functional and non-functional, with the various business and prioritization levels of each requirement.

The prioritization is done using the MoSCoW priority scale [1], of which it was deemed that only the Must Have (MH) and Should Have/Nice To Have (NTH) scales were necessary for this project. Requirements marked as MH are necessary for the system to function as intended, while NTH requirements are only included in the final product if there is still time to implement them before the final stages of the project.

The business levels of the requirements are chosen based on the goal-design scale, which is divided in goal, domain, product and design level [2]. Goal-level requirements describe the purpose of the overall system, domain-level requirements describe the support tasks for each role, product-level requirements describe the functionality of the product and design-level requirements describe the way in which they are implemented in terms of UI.

The list of requirements related to the development of the PLR system are described in the following sections. Some changes were made throughout the course of the project, in particular to the NTH requirements as per the client's wishes - those are noted both in the list of requirements and the dedicated section of the report following that.

3.1. Functional Requirements

3.1.1. Goal-level Requirements

(i) The system aims to help students assess their own learning progress. (MH)
(ii) The system aims to help students find materials related to their studies. (MH)
(iii) The system aims to allow mentors and teachers to observe each student's learning progress. (MH)
(iv) The system aims to notify mentors, students and teachers about relevant events. (MH)

3.1.2. Domain-level Requirements

(i) The system should support students in the task of filling in their self-assessment. (MH)
(ii) The system should support students in viewing related course material. (MH)
(iii) The system should support students and mentors in keeping them informed about deadlines and submissions. (NTH)
(iv) The system should support teachers and mentors in monitoring their students' progress using statistical reports. (NTH)
(v) The system should support users in the task of sharing study materials with each other. (MH)

3.1.3. Product-level Requirements

(i) The system should consist of four separate views for students, mentors, teachers and administrators. (MH)
(ii) The system should send notifications to mentors when a mentee has completed a self-assessment. (MH)

(iii) The system should send notifications to mentors when a mentee has submitted an assignment. (NTH)
(iv) The system should send notifications to mentors and students when a deadline is approaching. (NTH)
(v) The administrator and teacher should be able to create courses. (MH)
(vi) The teacher should be able to manipulate the users within the assigned courses. (MH)
(vii) The teacher and administrator should be able to create a single user account. (MH)
(viii) The teacher and administrator should be able to create a user account by uploading the CSV file. (NTH)
(ix) The system should be integrated with the Canvas LMS platform in terms of manipulating data in both directions. (NTH) - client eventually chose to leave this requirement out.
(x) The administrator, teacher and mentor should be able to see statistics for a specific assignment with the view of a particular group. (NTH)
(xi) The teacher and administrator should be able to create student groups. (MH)
(xii) The mentors, teachers and administrators should be able to add students to groups. (MH)
(xiii) The mentors, teachers and administrators should be able to remove students from groups. (MH)
(xiv) The user should have the option to be assigned multiple roles. (MH)
(xv) The teacher should be able to create points for self-assessment. (NTH)
(xvi) The students should be able to do a self-assessment easily. (MH)
(xvii) The mentors should be able to assess their mentees easily. (MH)
(xviii) Students should be able to view the available related materials. (MH)
(xix) Mentors and teachers should be able to add new related materials. (MH)
(xx) Mentors and teachers should be able to upload files into the system and share them with students. (NTH)
(xxi) Students should be able to request to be added to a course. (MH)

(xxii) The administrator should be able to manage all users. (MH)
(xxiii) The system should allow mentors, teachers and administrators to leave internal feedback not visible to the students. (NTH) - this requirement was added later at the request of the client.
(xxiv) Students should be able to give likes to the various related materials. (NTH) - this requirement was added later at the request of the client.
(xxv) Students should be able to request a checkpoint meeting using the system. (NTH) - this requirement was added later at the request of the client.

3.1.4. Design-level Requirements

(i) The system should be developed using Java or Python. (MH)
(ii) The system should utilize up-to-date Javascript frameworks. (MH)
(iii) The system should be integrated with Canvas LMS. (NTH) - client eventually chose to leave this requirement out.
(iv) The system should allow users to log in using their UT accounts. (NTH)
(v) The system should allow users to log in using the single sign-on (SSO) method. (NTH)
(vi) The system should be mobile-friendly. (NTH)
(vii) The system should give users information about the statistics. (NTH) - this requirement was added later at the request of the client.

3.2. Non-functional Requirements

(i) The system should be developed in compliance with the GDPR. (MH)
(ii) The system should follow the ethical norms from the University of Twente. (MH)
(iii) The system should be secure for usage. (MH)
(iv) The system should be well-documented. (MH)
(v) The system should be easy to maintain. (MH)
(vi) The system should be tested thoroughly. (MH)
(vii) The system should be implemented in English. (MH)

3.3. Requirements Changes

Throughout the course of the project, the requirements changed occasionally, in particular the nice-to-have requirements, while the core functionality remained more or less unchanged. This could be accommodated relatively easily due to the team's use of the Agile methodology, described in more detail in Section 5.

In particular, the most important NTH requirement at the beginning of the project was considered the Canvas integration. However, despite the team's best efforts, it eventually turned out to be impossible to implement for the given timeframe, the reason for which is explained later in the report. As such, the client decided to substitute that requirement for a few others, also considered to be NTH.

In particular, the new requirements included giving the users information about the presented statistics, allowing mentors, teachers and administrators to give internal feedback about students without it being visible to said students, and giving students the opportunity to arrange a checkpoint meeting through the application, as well as to like certain materials they find particularly helpful.

4. Tools

In this section, the tools involved in the development of the system are described, from planning to implementation.

4.1. Planning

4.1.1. Jira

In order to oversee our task division and to ensure that the team stays on track throughout the course of the project, the issue-tracking tool Jira [3] was made use of to create and assign tasks. The reason such a tool was selected to be used is due to the fact that the project management tool chosen for this project is the Agile approach [5], described in more detail in Section 5, and, additionally, to easily check each team member's progress throughout each sprint.

4.1.2. Miro

To create a feasible 10-week implementation plan for the project, the digital whiteboard tool Miro [4] was used. With it, the planning as seen in Figure 2 was obtained at the beginning of our project.

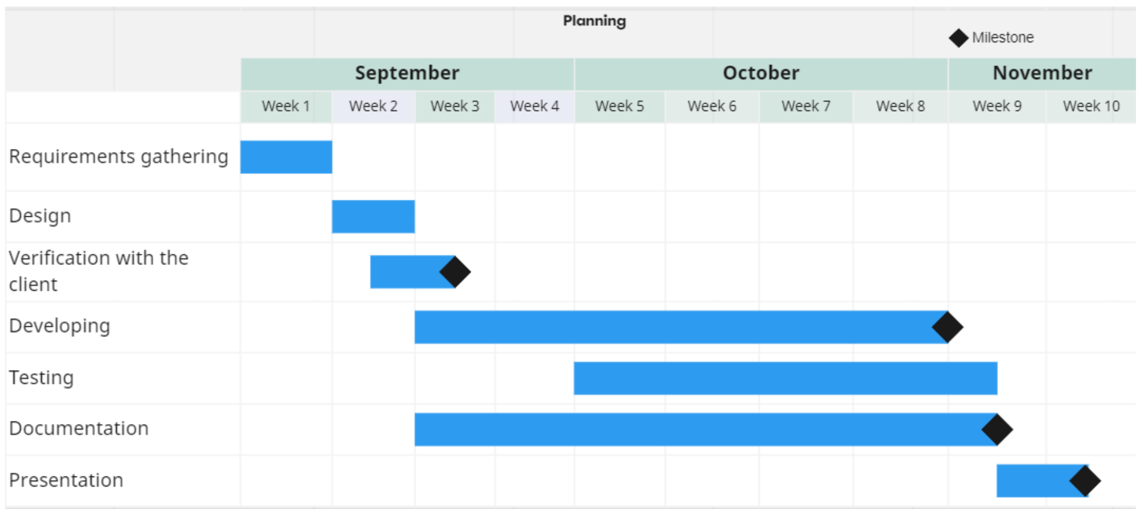


Figure 2. Planning timeline for the implementation of the project.

The adherence to the planning is discussed in Section 5.

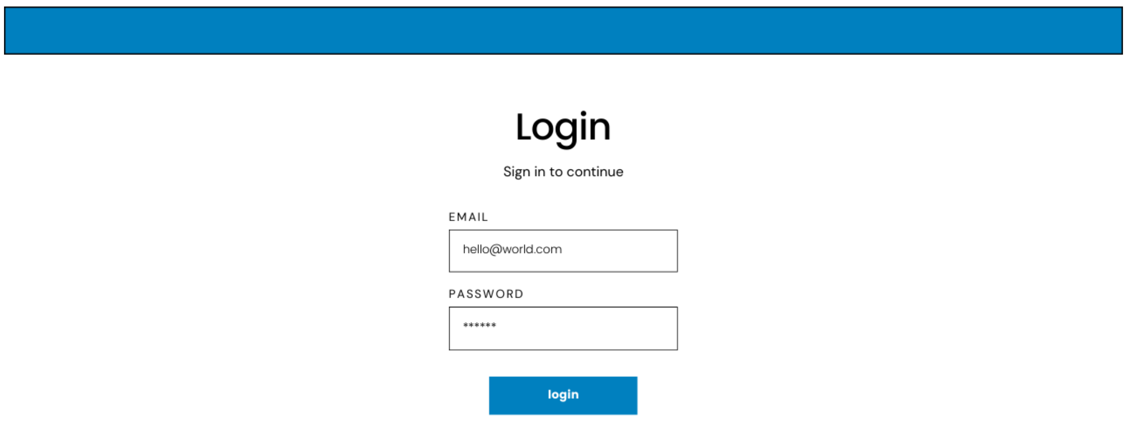
4.2. Design

4.2.1. Visual Paradigm

To assist our design phase, multiple UML diagrams were created describing the system (more details about them in Section 5). To accomplish this, the tool used is Visual Paradigm [6], which allows the team to create activity diagrams, class diagrams, etc.

4.2.2. Canva

Before the implementation begins, mockups are created by the team with the help of the tool Canva [8] in order to represent an approximate idea of what the website should look like and behave, and these models were closely adhered to in the creation of the system. The mockups can be seen in Figures 3 - 10.


Login

Sign in to continue

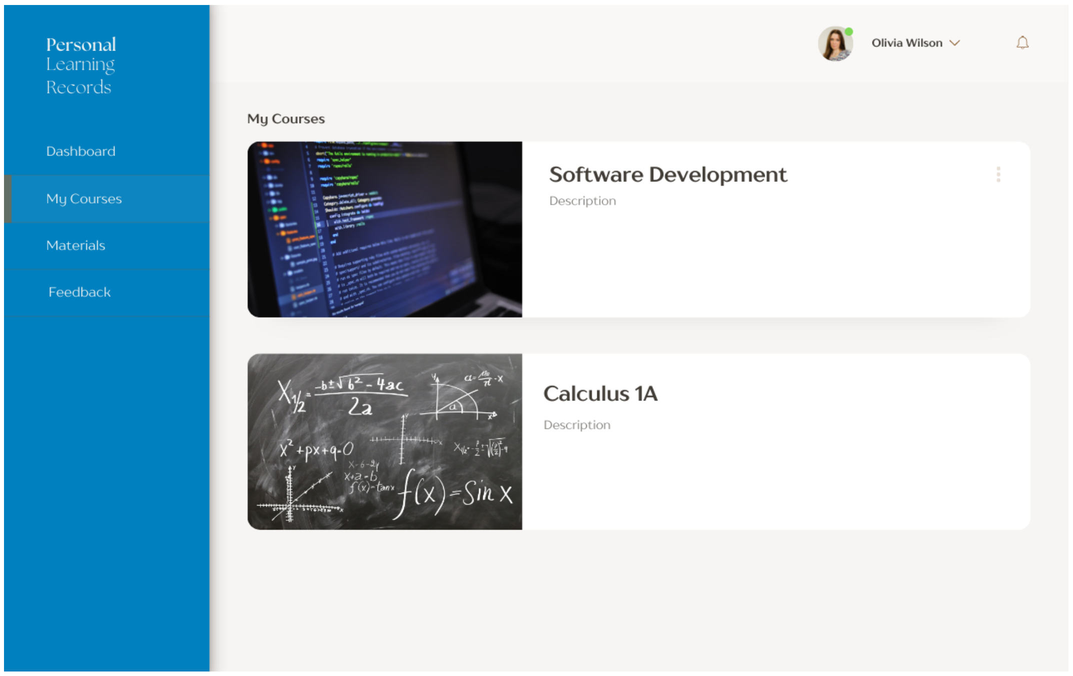
EMAIL

hello@world.com

PASSWORD

******

login



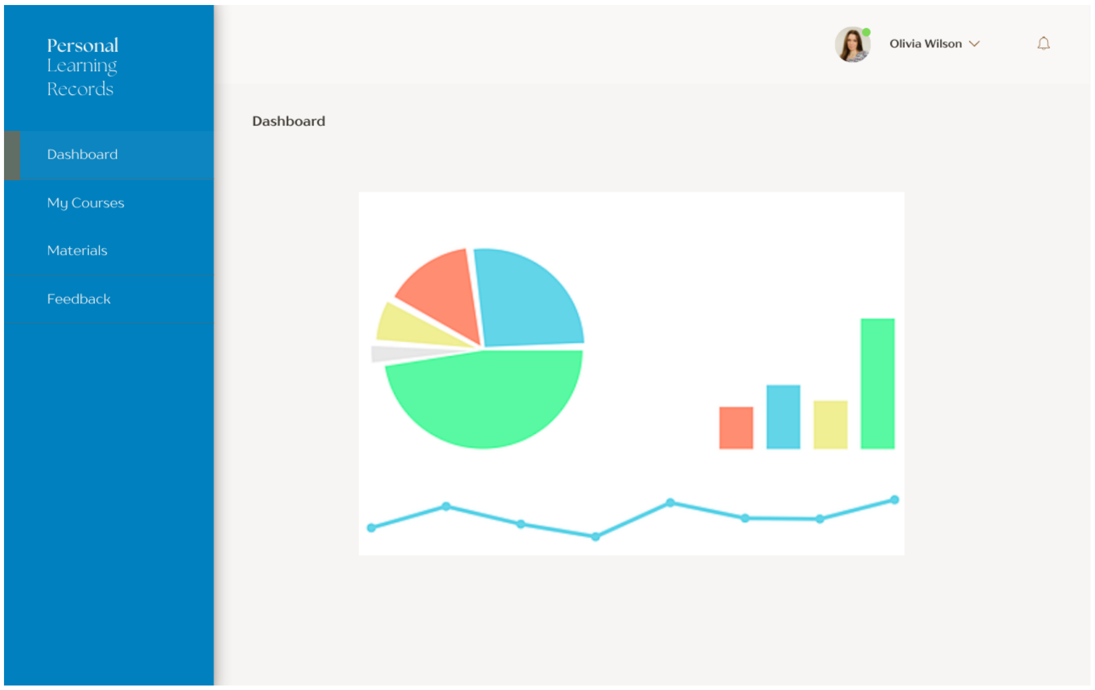
My Courses

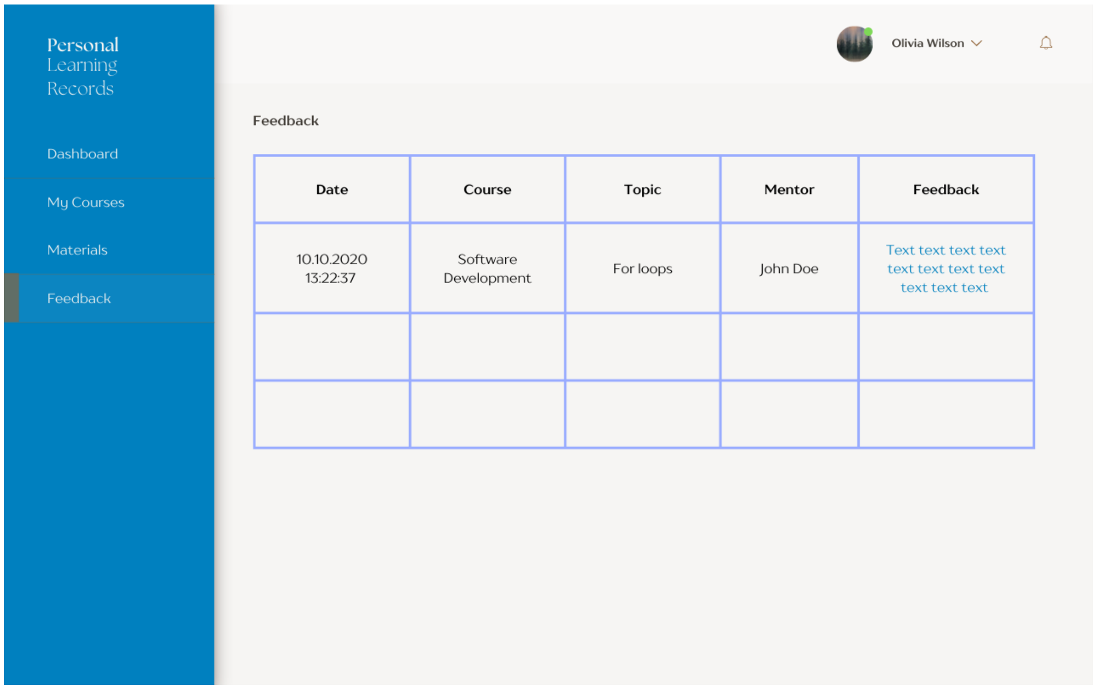
Software Development

Description

Calculus 1A

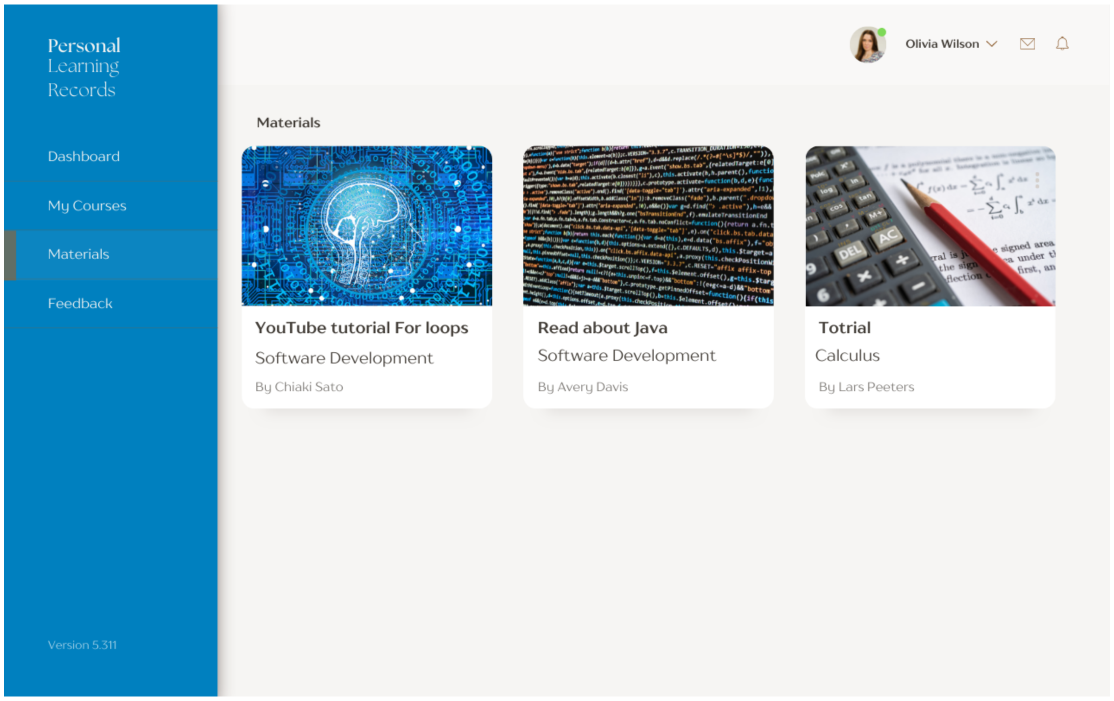
Description





Feedback

Date	Course	Topic	Mentor	Feedback
10.10.2020 13:22:37	Software Development	For loops	John Doe	Text text text text text text text text text text text

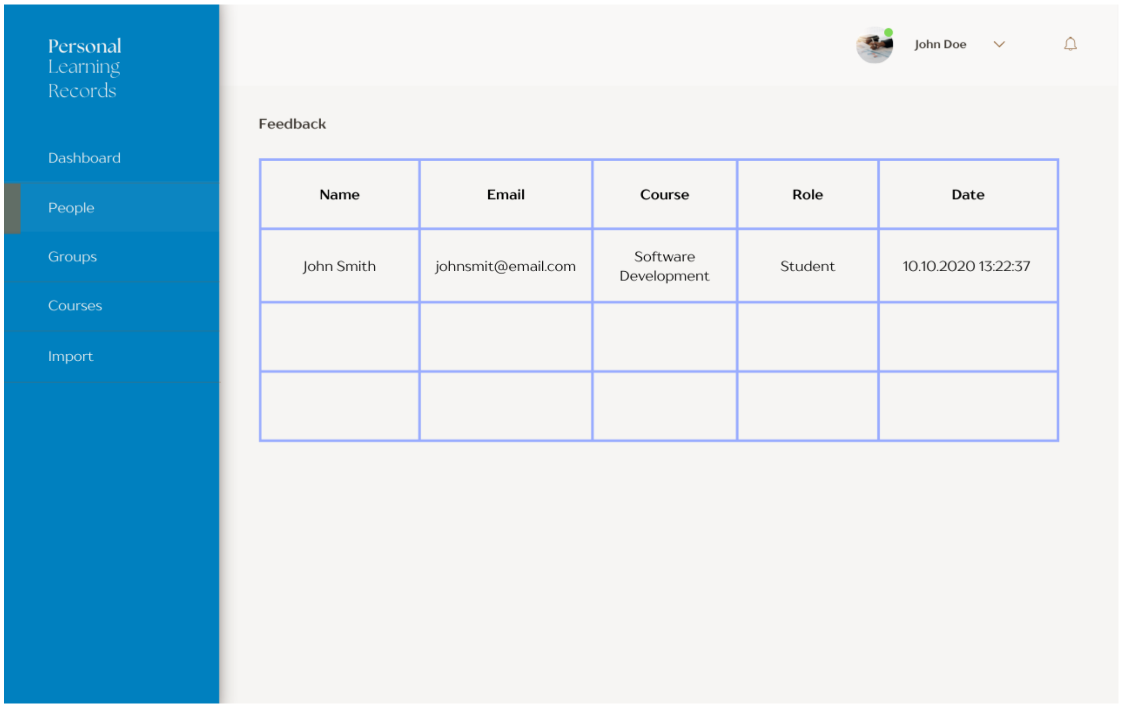


Materials

YouTube tutorial For loops
Software Development
By Chiaki Sato

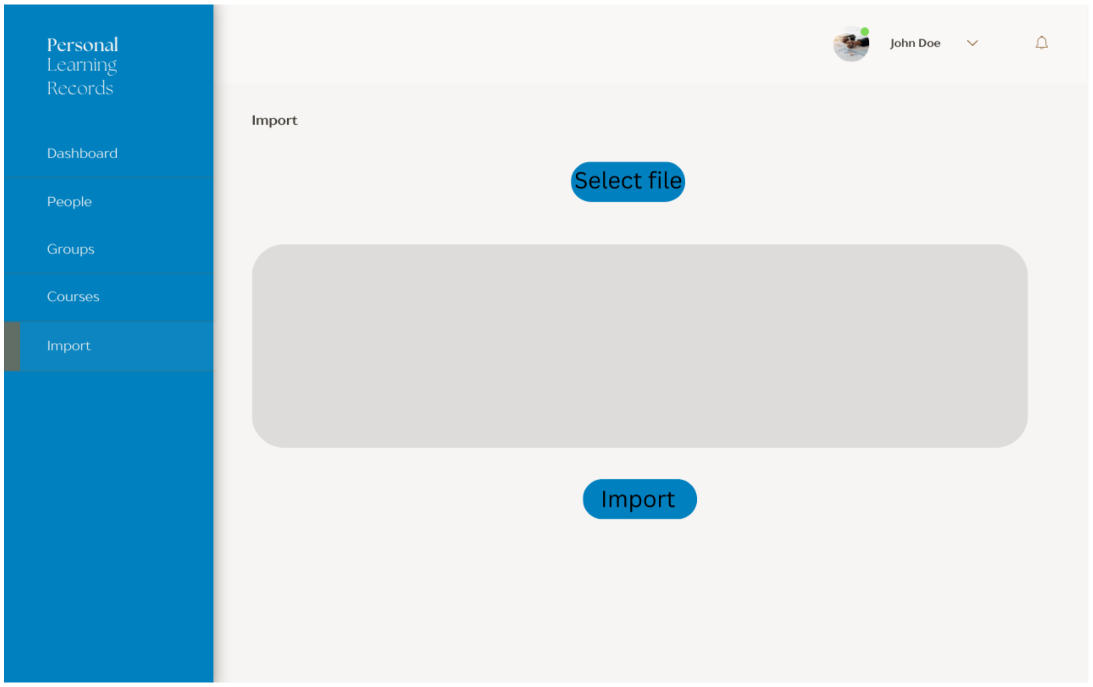
Read about Java
Software Development
By Avery Davis

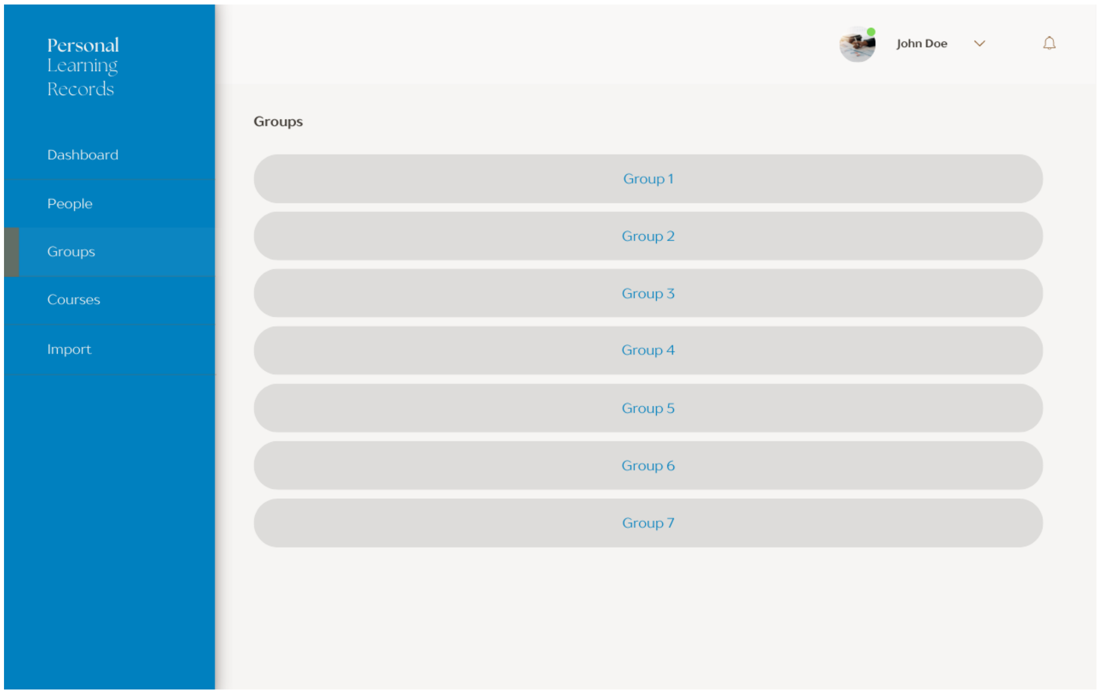
Totrial
Calculus
By Lars Peeters



Feedback

Name	Email	Course	Role	Date
John Smith	johnsmit@email.com	Software Development	Student	10.10.2020 13:22:37





Figures 3 - 10. Mockup pages for the project

4.3. Implementation

4.3.1. React & MaterialUI

For the frontend development, the React and MaterialUI libraries were used in order to save time when creating the pages of the website, as well as to ensure a layout and appearance that is comfortable to use and pleasing to the user's eye.

4.3.2. GraphQL

The data query language GraphQL was used in order to create a connection to the database between the backend and the frontend. Despite not being a RESTful service, it was chosen with the approval of the client primarily due to its fast learning curve, servicing the goal of maintainability of the system, as desired by the client, as well as the team's familiarity with its usage. To facilitate the use of GraphQL, the Apollo Server Javascript library was used.

4.3.3. GitLab

In this project, managing the versioning was done through the use of a GitLab repository. Each feature in the requirements was implemented on a separate branch in order to avoid conflicts.

4.3.4. Spring & Hibernate

For the backend, the team chose to work with the Spring framework to facilitate the development of features in the backend. Additionally, the Hibernate framework was used to support interaction with the database.

5. Planning & Approach

In this section, the planning created by the team at the beginning of the project is described, along with the approach decided on to tackle the development.

5.1. Design Phase

The design phase consisted of the following subphases, as seen in the planning in Figure 2 - gathering requirements for the project, creating a rough outline for the proposed system and, finally, obtaining the client's approval in order to proceed with the development. The planning in the figure was followed, with some additional overlap between the requirements-gathering phase and the design phase while waiting for the client's confirmation. Additionally, the development environment was set up to save time during the implementation phase. The subphases are described in more detail in the following sections.

5.1.1. Requirements Gathering

The first step taken towards the design phase of our project was to contact our client and gather his requirements for the system, as well as to clarify his preferences for the implementation. For this purpose, multiple meetings with the client were conducted at the start of the project. During these meetings it was determined that the client expects as a result a base recreation of the existing system in Java or Python, preferably by using RESTful web services.

As an additional functionality, the client wants the system to include a notification functionality to alert teachers and mentors when a student has completed a self-assessment or submitted an assignment, and, additionally, students and mentors should receive notifications when a deadline is approaching. The more detailed results of this requirements-gathering process can be found in Section 3.

5.1.2. UML Diagrams

In order to model and visualize the functionalities required by the client, the team created multiple UML diagrams, most vital of which for this project is the class diagram, representing the relationships between the classes in the project. It can be seen in Figure 11 below.

Additionally, the team created activity diagrams to model the processes to be undertaken by the users of the system. An example can be seen in Figure 12 and Figure 13 below. The activity diagram shown on Figure 12 depicts the process of a student filling in a self-assessment, while the activity diagram in Figure 13 shows the steps for logging in for any role in the system.

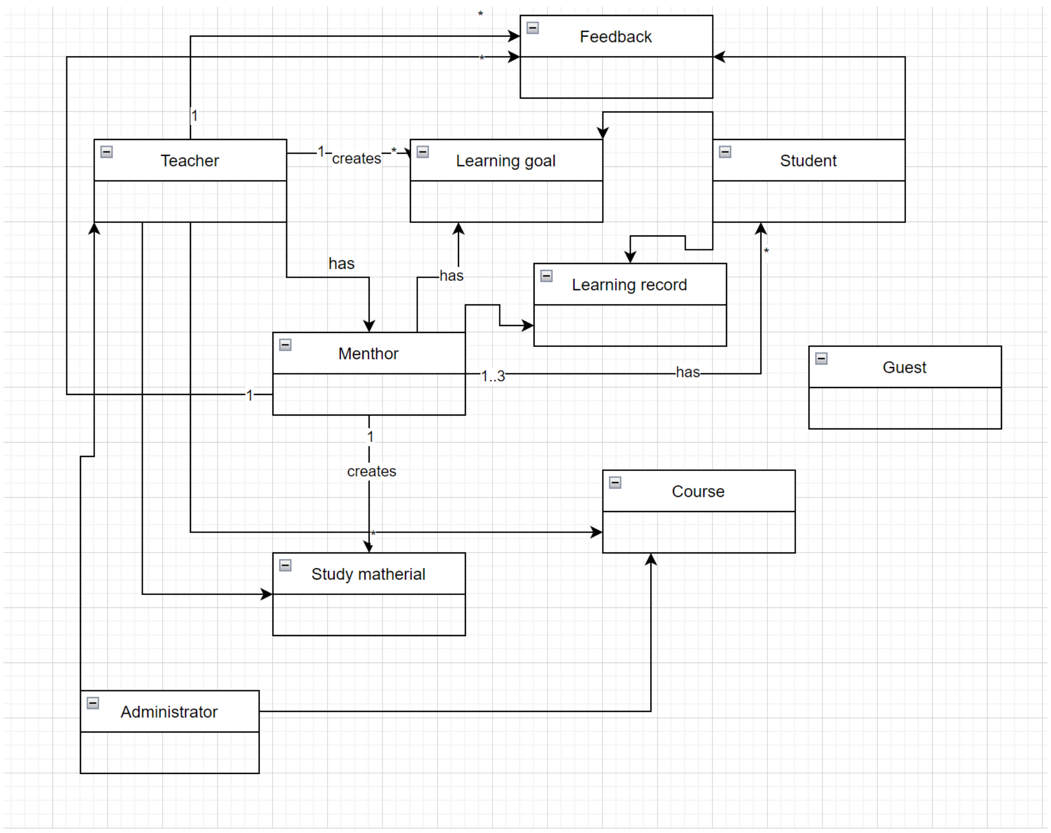


Figure 11. Class diagram

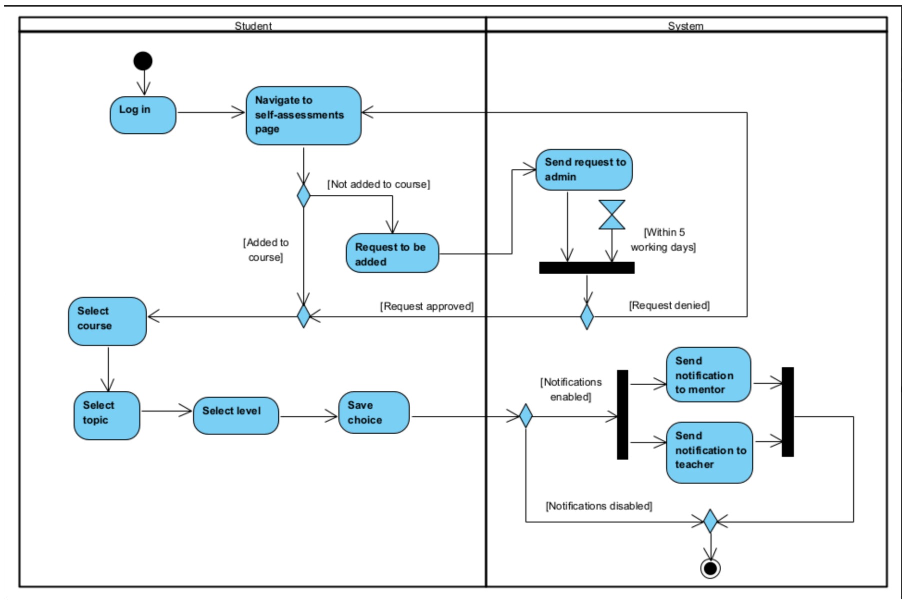


Figure 12. Activity diagram for self-assessment

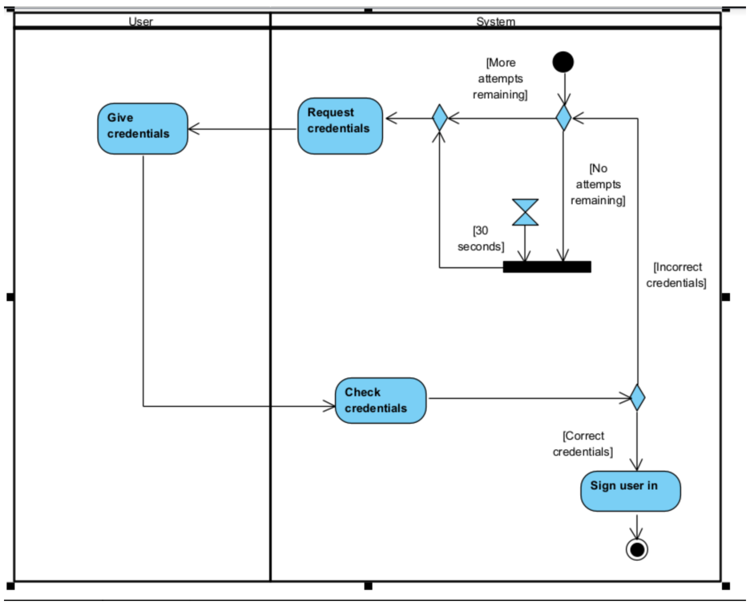


Figure 13. Activity diagram for login

5.1.3. Agile Methodology

The next step of the design process was to select a project management method. The primary options were between the traditional Waterfall project management and the Agile project management.

Taking into consideration the fact that the project's specifications and details of implementation were likely to be subject to change throughout the development process, it was judged that Agile would be the most suitable methodology to use, due to its relative flexibility compared to the Waterfall methodology, which does not easily allow for adjustments once the planning has been set and development is in motion.

Additionally, the Waterfall methodology is fairly restrictive on the frequency with which the development team can interact with the client, compared to the Agile methodology. Considering that the client's frequent input and feedback on the progress is vital for the success of the final product, this was another contributing factor to the decision to use Agile methodology over Waterfall.

Agile development was primarily implemented through the use of standup meetings each time the team meets to work on the project. As such, standup meetings occurred at least once a week, ensuring a smooth planning process, no overlap between various tasks, as well as each team member receiving help whenever needed. Additionally, the client was frequently asked for his input on certain aspects and decisions about the system, ensuring that the system meets his requirements and flexible criteria.

5.1.4. Database Design

We decided to stick to a simple database design that suits only the needs to store the required data, due to the choice of frameworks we made. Because of that, we stick only to the strictly necessary tables, such as course, topic etc. We created separate wrapper tables to support the many-to-many relationships present in the project.

We make use of the Spring framework and Hibernate (detailed in Section 4), which allows us to create JPA queries and handle all the data in our middleware by using backend object classes instead of working directly with the data from the database. In that case our conditions were only related to the specific objects, which improved the performance considerably and simplified our database design as much as possible. In the appendix, a full schema of the final database design can be found.

5.2. Development & Testing Phase

For the development we decided to make use of a couple of frameworks common for developing web applications that are intended to support the handling of large amounts of data, as well producing high performance without a high demand of resources.

5.2.1 Front end

For frontend development we decided to use React and TypeScript as they are really commonly used JavaScript frameworks that support front end development. For the design of the web application we made use of MaterialUi, which provides an easy way to access elements and helps to maintain the reusability and the consistency of the components for each page. These are described in more detail in the Tools section.

5.2.2 Back-end

The back-end of the application is handled by Spring Boot, that also uses Hibernate so as to be able to select data from the database by using JPA query and from there to modify the data to make it suitable for representation in the front-end. All the components of the application are represented as objects that allow easy modification of the data in the database, adding, deleting and querying.

5.2.3 Connection Front-end and Back-end

For an easy way to ensure communication between the front end of our application and the back end we use GraphQl (also mentioned Tools section), that ensures easily selecting the necessary data from the backend and directly represent it as an object in the React application as well as taking information from user input and easily send it to the back-end and respectively to the database.

5.2.4 Information flow

The data is being input by a user, after which it is being modified to suit a specific input object for the specific functionality. By using GraphQl mutation, that object is being passed to a listener mutation method in the backend from where its data is being extracted and used for the necessary purposes. On the way back, data is being selected by using a JPA query with predefined conditions. From that place, that information is being modified to a state suitable for showing to the user. By using GraphQl query mapping, as well as fragments and types the data is sent to the React application where it is being modified to enter a certain visual component and be shown to the user of the application

5.2.5 Log in

The system ensures accessibility to the system by requiring log in from the end users. In order to be secure for a standard entry, session tokens , Spring security and encryption are used to ensure that in case of data breach there will be no sensitive information leaked. There is also Microsoft log in, where an external API is used to verify a certain user, to enter that profile in the system and ensure authorization and proper usage.

6. Functionality

In this section, we describe the functionalities that the system is capable of performing. The subsections are divided based on the page of the website that the functionality can be found on, including screenshots of the respective page.

6.1. Login

The first aspect of the project that any user, regardless of their role and access, will encounter when they use the system is the login page, which can be seen in Figure 14.

Users from outside the university can only use the standard login, which requests an email address and a password, while other users have the option to use either the standard login or the Microsoft login, which asks them to login through their Microsoft account. Once a user logs in, they are automatically added to the system.

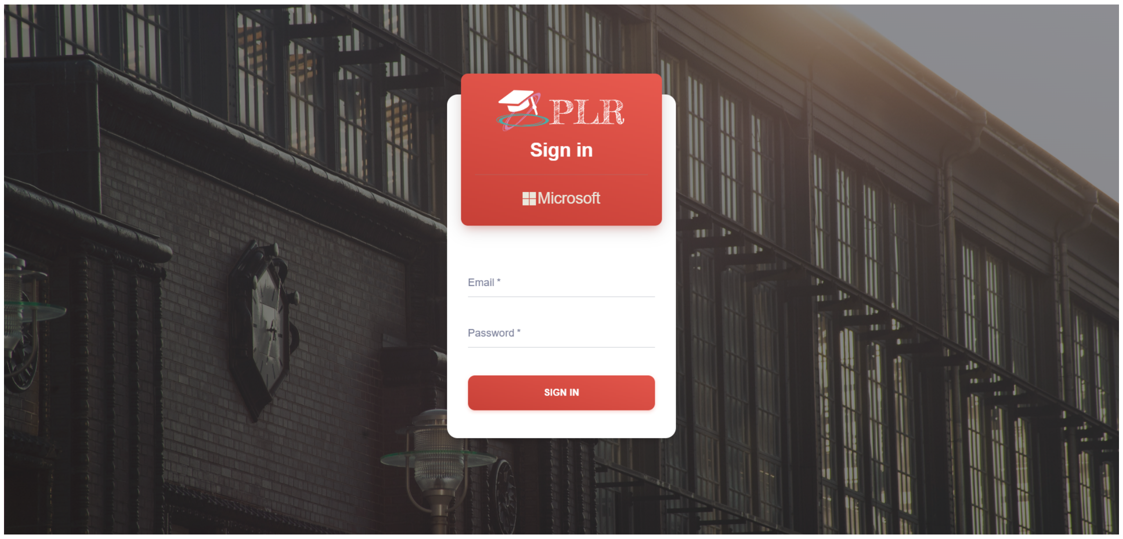


Figure 14. Login page

6.2. Navigation Bar

The navigation bar, as seen in its closed and opened version in Figure 15 and Figure 16, respectively, is an element present on every page, with the exception of the login page. While still closed, it shows the PLR logo, as well as a notification icon, which, when opened, shows a list of all unread messages that the user has received. If applicable, the navigation bar also gives the option to change user roles, which changes the view to the selected one.

The opened version also depicts the logo, but it also shows the user's account information and gives the option to log out. Additionally, the settings tab, when opened, gives the user the possibility to change their password. Doing so causes an email to be sent to the user's inbox with instructions about the password change. The rest of the navigation bar handles the movement between various pages.

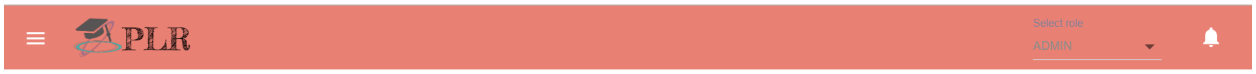


Figure 15. General navigation bar

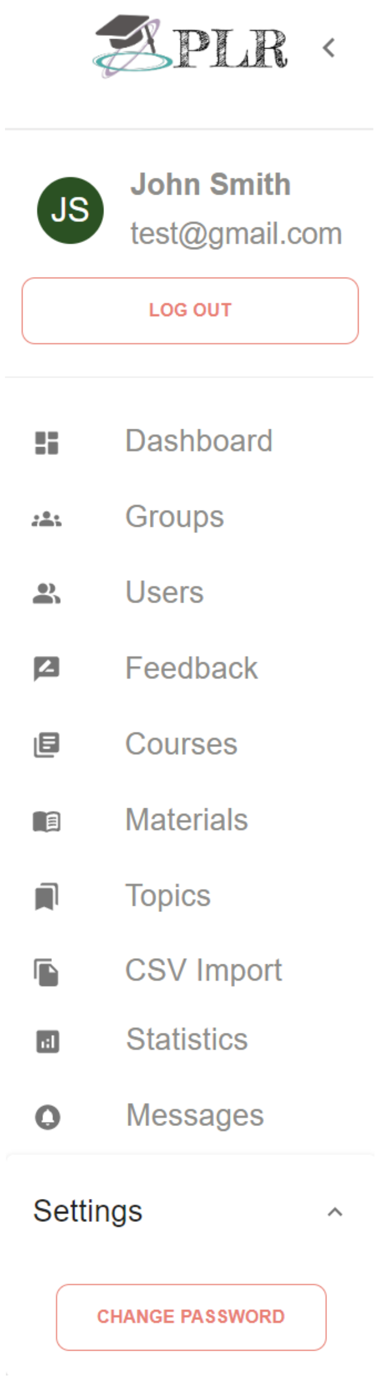


Figure 16. Opened navigation bar for the administrator

6.3. Landing Page

Once the user has logged in, they are redirected to the landing page, whose layout is dependent on the role of the user. The landing pages for the student, mentor, teacher and administrator can be seen in Figures 17, 18, 19 and 20, respectively. Selecting the Dashboard option from the navigation bar brings the user to this page as well.

The student's landing page is fairly simple - it showcases all courses in which the student is enrolled. Clicking on any of the courses listed on the page takes the student to the corresponding feedback page of the selected course. The feedback process is described in a later subsection.

The landing page for mentors, teachers and administrators is fairly similar - it depicts statistics regarding the overall progress of the students that the user is responsible for, regardless of their role. The statistics the team has implemented include Mastery over time, which shows the progression of the students' self-assessments, and Current mastery, which shows the current state of the students' self-assessments. The users can hover over the Information icon next to each statistic to see more detailed information about what the statistic measures.

Additionally, filters are provided to sort out any information that the user deems to be currently unnecessary, such as showing only the self-assessments by toggling the corresponding button, or filtering by various courses that the user has access to. Any new and unread messages can be seen in the Messages box next to the statistics.

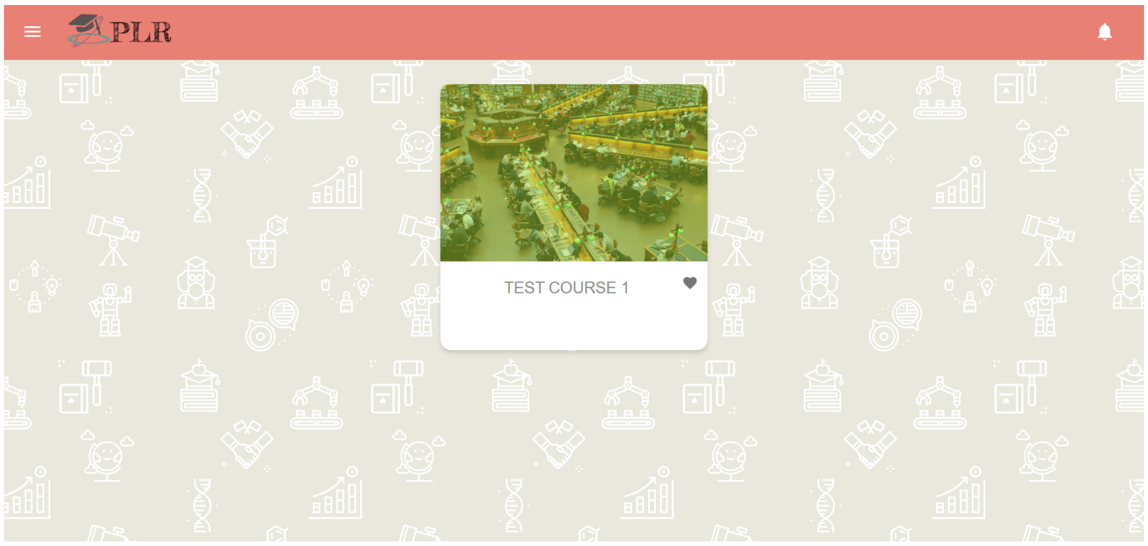


Figure 17. Landing page - student

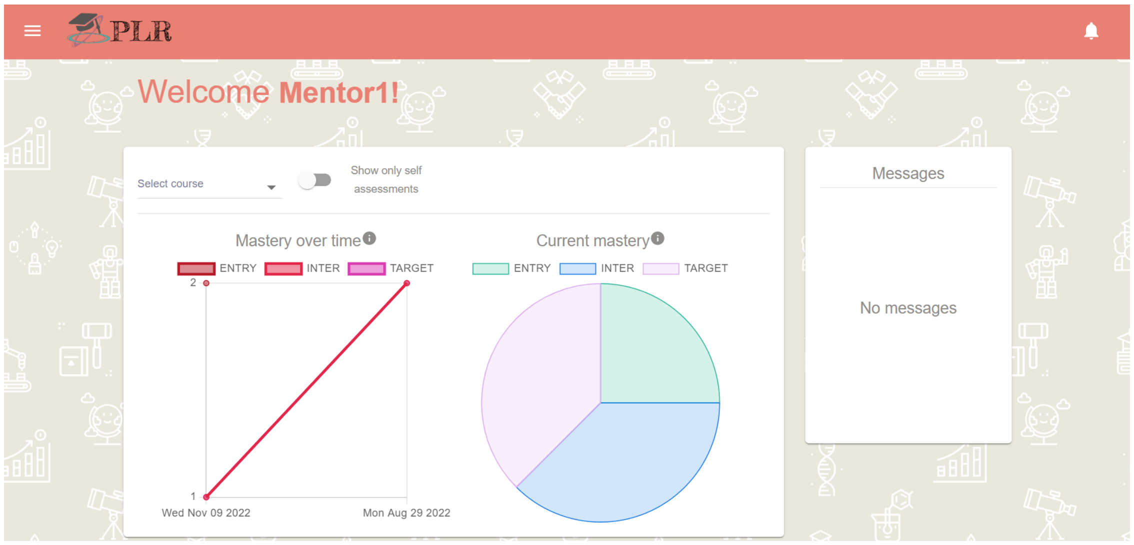


Figure 18. Landing page - mentor

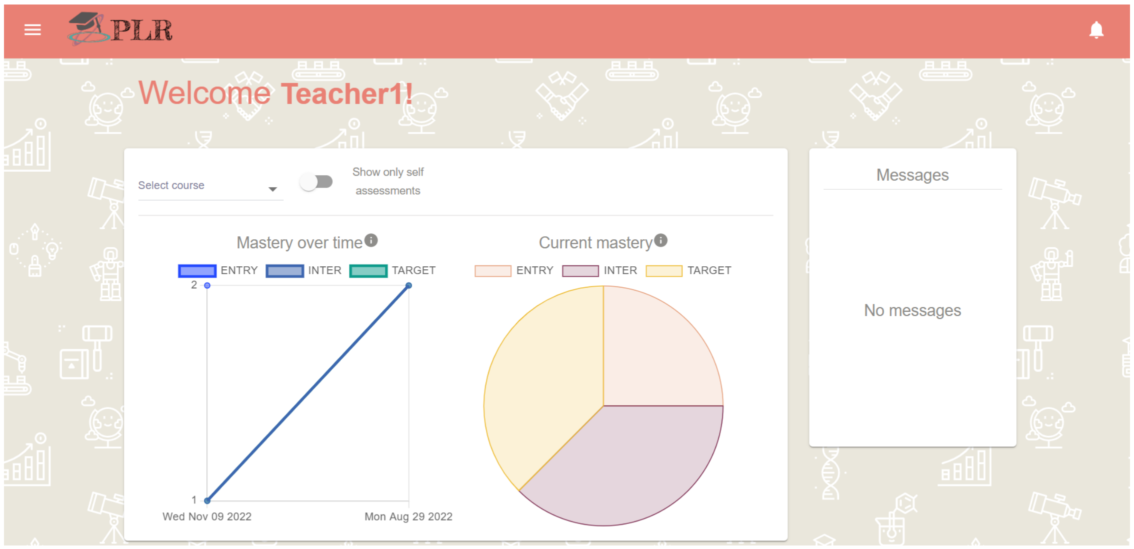


Figure 19. Landing page - teacher

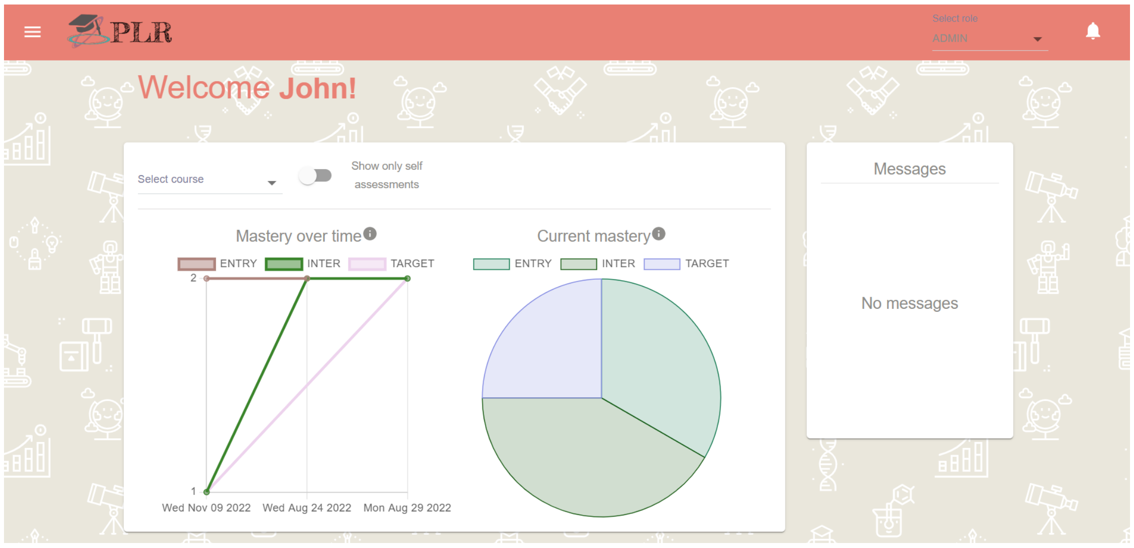


Figure 20. Landing page - administrator

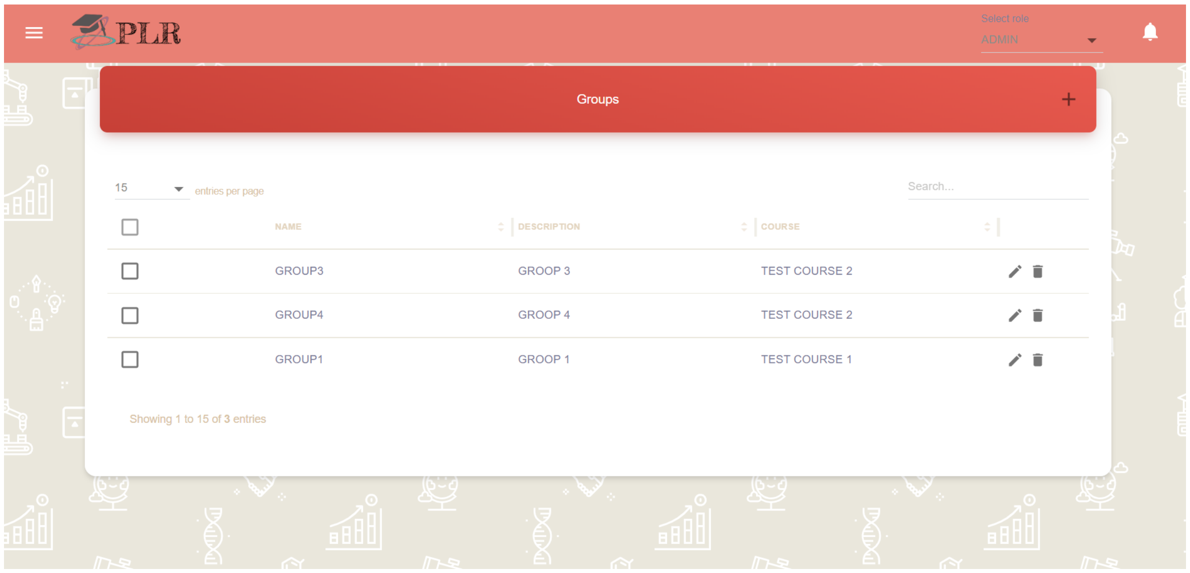
6.4. Groups

In this section, the functionalities discussed are the overview of the groups, the editing of a group, as well as the creation of a new group. These pages can be seen in Figures 21, 22 and 23, respectively.

The groups page shows an overview of each group that the user is attached to. Depending on the user's role and permissions, they can also be edited and deleted. Creating a new group can be done by clicking on the + button in the table.

The page for editing a group gives the user the ability to edit the name and description of a course, while also giving them the option to change the users associated with the specific course, in particular by adding or removing users from a group. It is worth pointing out that each instance in the website for editing a description includes a rich text editor.

Finally, the page for creating a group allows users to create groups based on the input they give for a name and description, as well as the course that the group will be a part of.



PLR

Select role: ADMIN

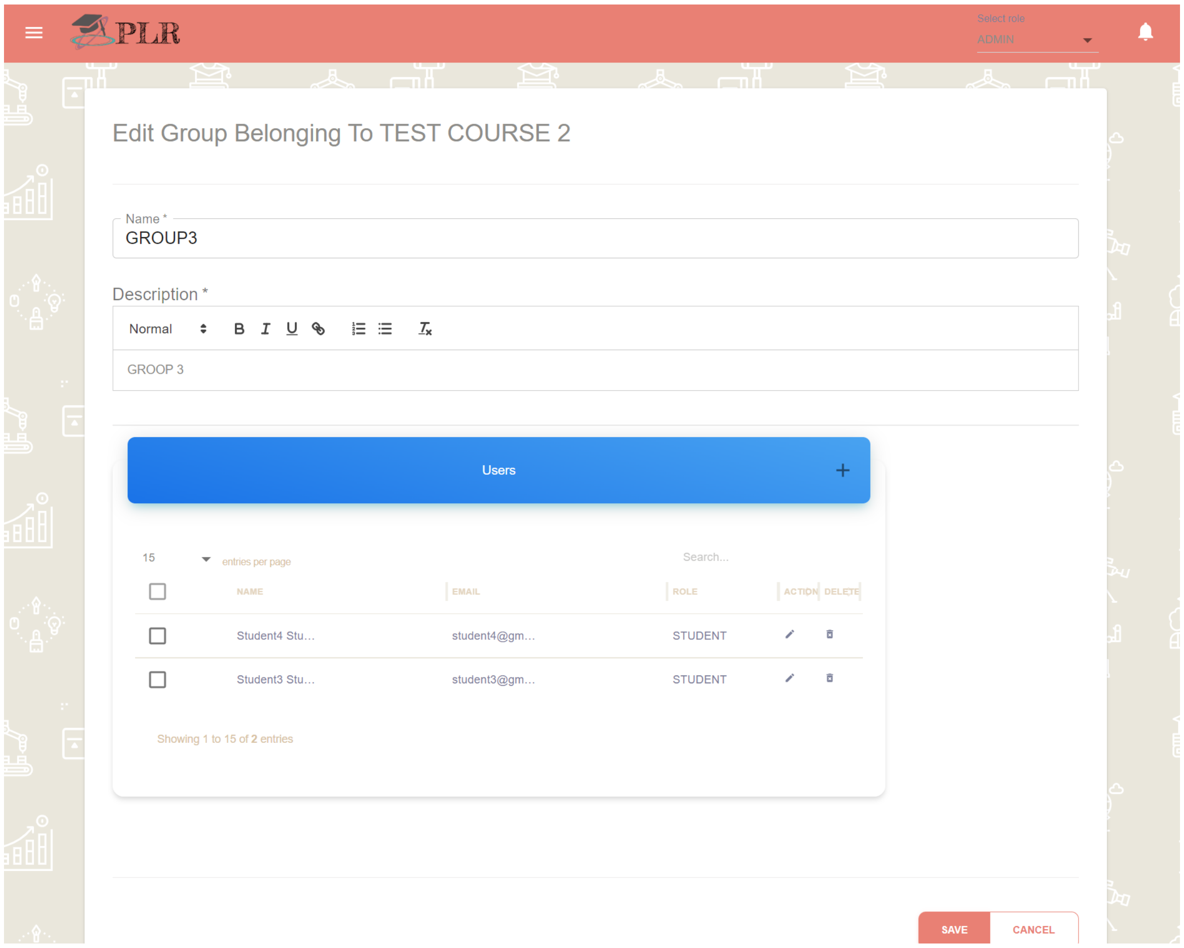
Groups

15 entries per page | Search...

	NAME	DESCRIPTION	COURSE
☐	GROUP3	GROOP 3	TEST COURSE 2
☐	GROUP4	GROOP 4	TEST COURSE 2
☐	GROUP1	GROOP 1	TEST COURSE 1

Showing 1 to 15 of 3 entries

Figure 21. Groups overview page



PLR

Select role: ADMIN

Edit Group Belonging To TEST COURSE 2

Name *: GROUP3

Description *

Normal

GROOP 3

Users

15 entries per page | Search...

	NAME	EMAIL	ROLE	ACTION	DELETE
☐	Student4 Stu...	student4@gm...	STUDENT		
☐	Student3 Stu...	student3@gm...	STUDENT		

Showing 1 to 15 of 2 entries

SAVE | CANCEL

Figure 22. Editing a group

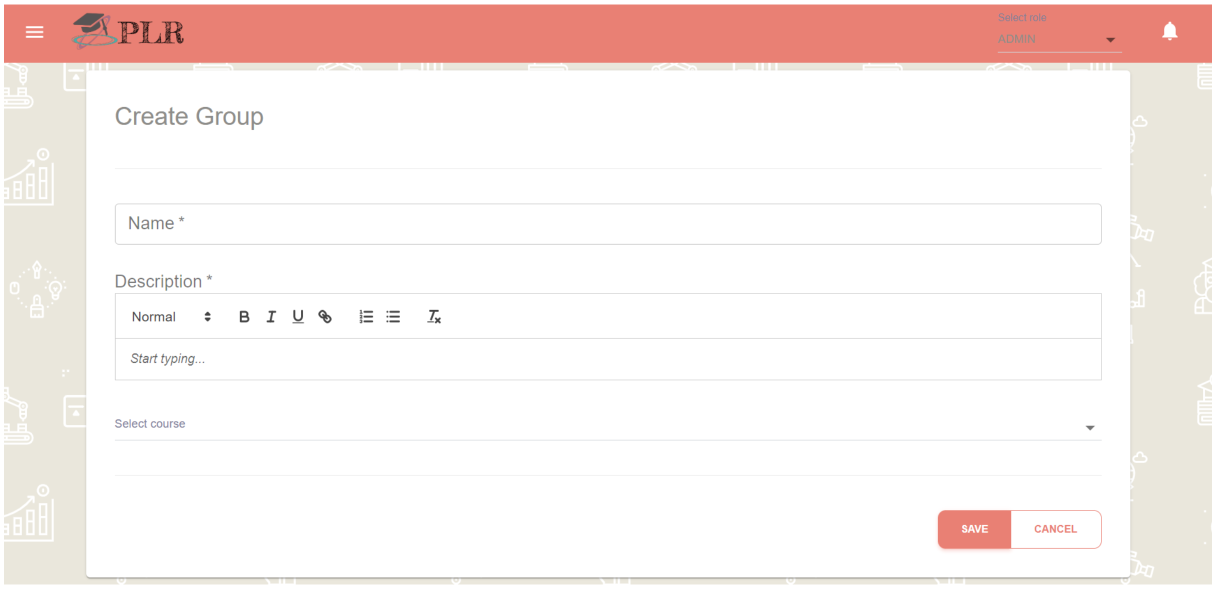
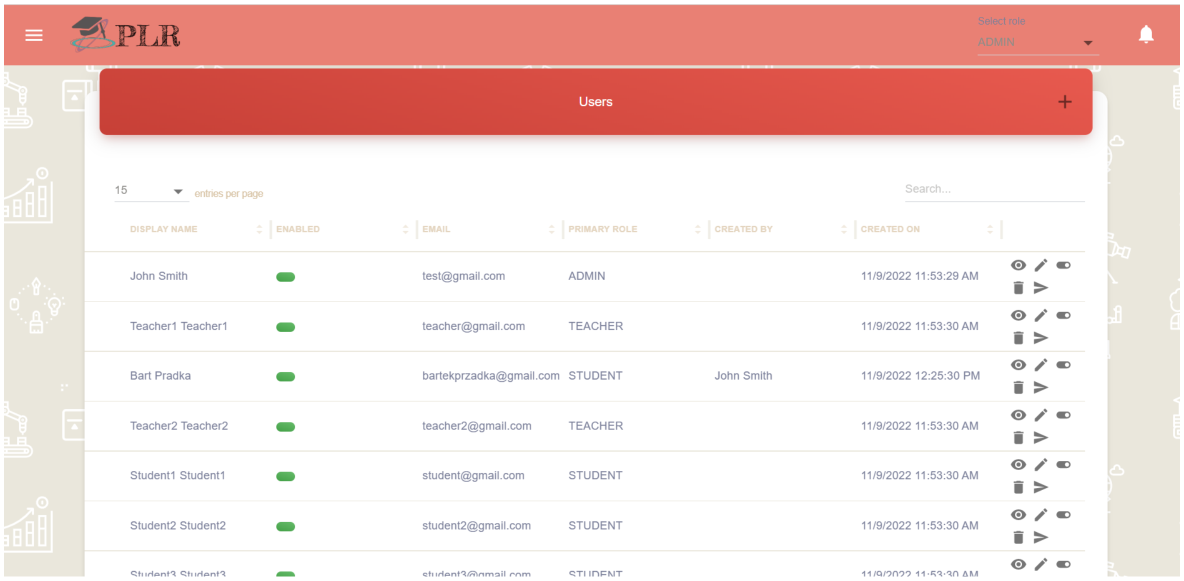


Figure 23. Creating a group

6.5. Users

In this section, the items related to the users are shown, as accessed from the Users button in the navigation bar. The overview, as seen in Figure 24, shows a list of all the users that the user has access to. Depending on the user's permissions, the users can be viewed, edited and deleted, while also giving the possibility to enable and disable a user, as well as to change said user's password (this functionality works in the same way as changing your own password, described in more detail in an earlier subsection). Additionally, creating a new user is possible by clicking on the + button in the table.

The page for editing the details of a specific user can be seen in Figure 25. The attributes that can be edited are first, last and display name, email and primary role. The page for creating a user is shown in Figure 26, where the creator has to fill in those attributes to add a new user to the system.

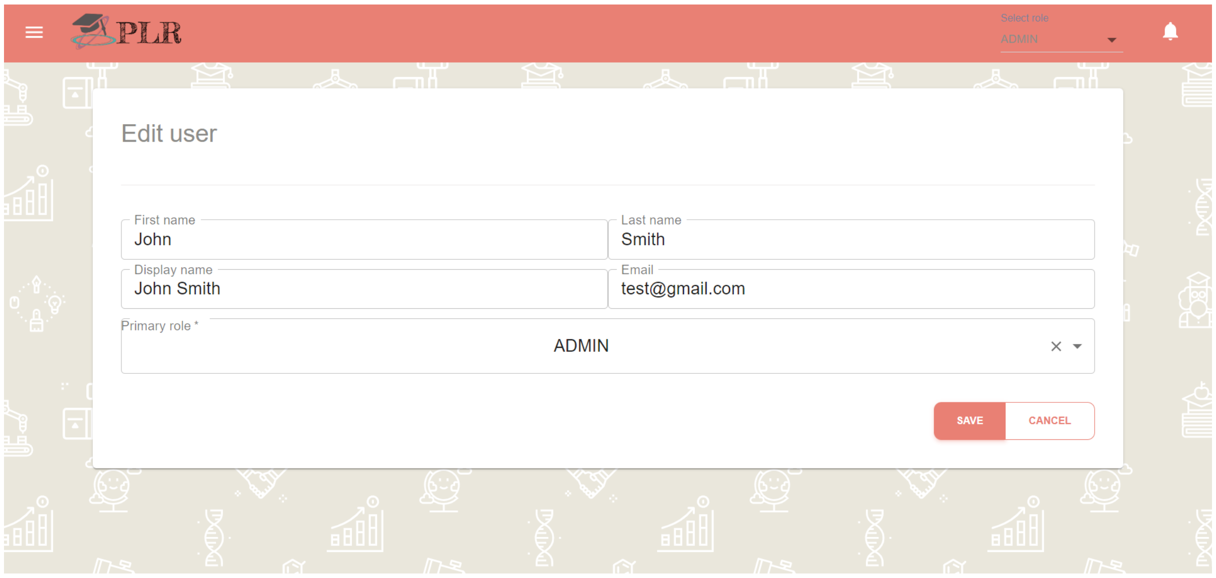


15 entries per page

Search...

DISPLAY NAME	ENABLED	EMAIL	PRIMARY ROLE	CREATED BY	CREATED ON
John Smith	🟩	test@gmail.com	ADMIN		11/9/2022 11:53:29 AM
Teacher1 Teacher1	🟩	teacher@gmail.com	TEACHER		11/9/2022 11:53:30 AM
Bart Pradka	🟩	bartekprzadka@gmail.com	STUDENT	John Smith	11/9/2022 12:25:30 PM
Teacher2 Teacher2	🟩	teacher2@gmail.com	TEACHER		11/9/2022 11:53:30 AM
Student1 Student1	🟩	student@gmail.com	STUDENT		11/9/2022 11:53:30 AM
Student2 Student2	🟩	student2@gmail.com	STUDENT		11/9/2022 11:53:30 AM
Student3 Student3	🟩	student3@gmail.com	STUDENT		11/9/2022 11:53:30 AM

Figure 24. Users overview page



Edit user

First name: John

Last name: Smith

Display name: John Smith

Email: test@gmail.com

Primary role *: ADMIN

Figure 25. Editing a user

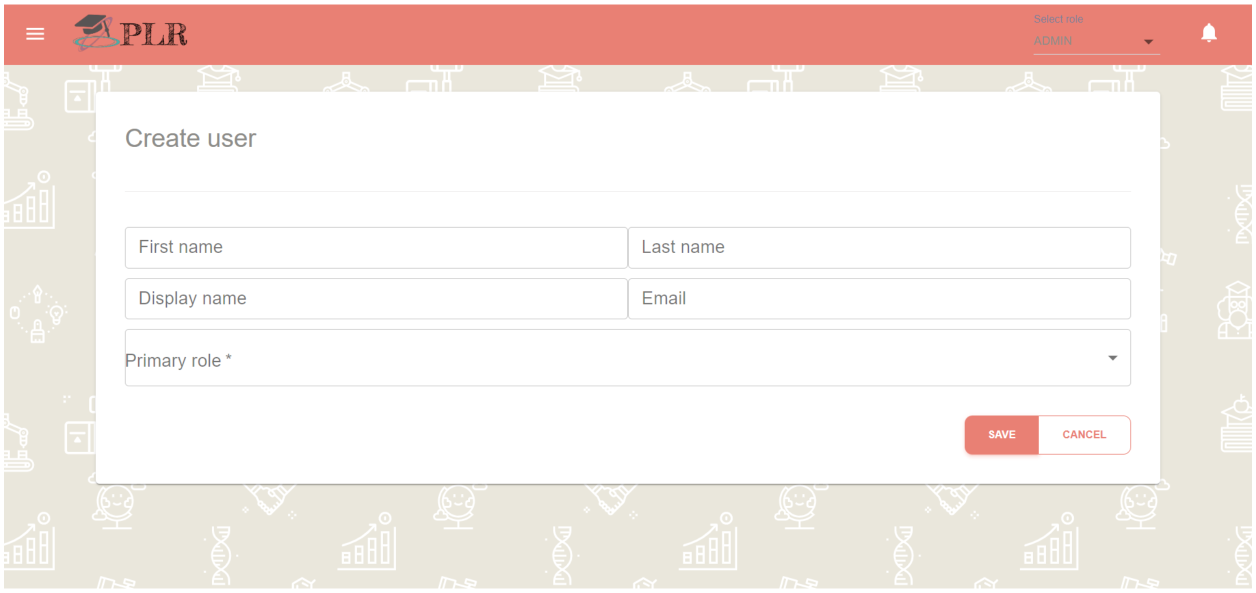
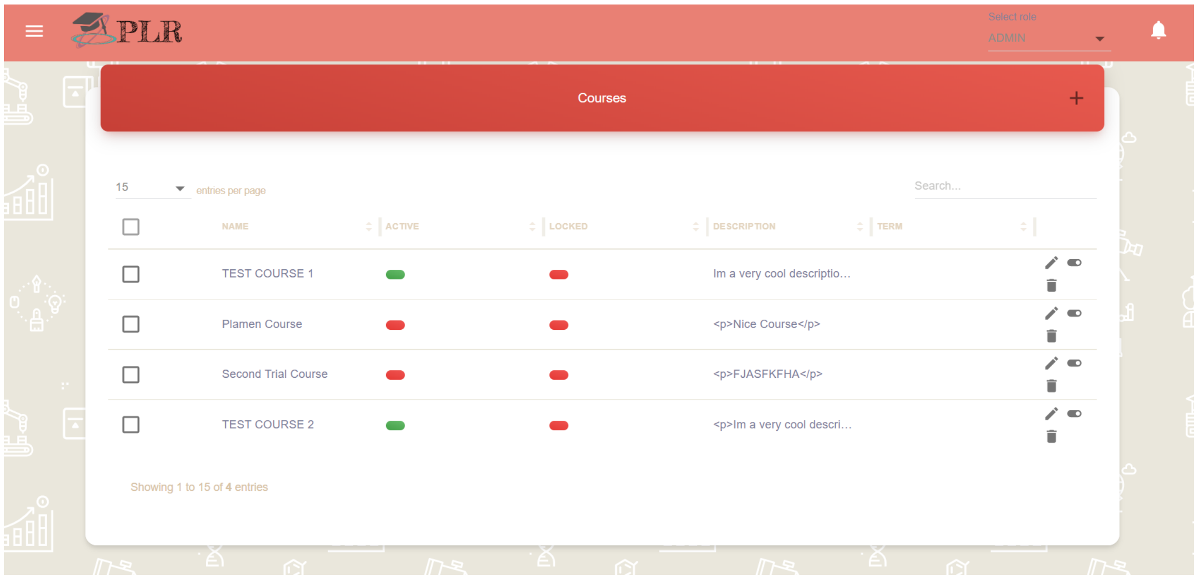


Figure 26. Creating a user

6.6. Courses

The courses section of the application is reachable from the Courses tab in the navigation bar, which leads to an overview like the one depicted in Figure 27, which allows the users with the correct access to edit, delete and enable/disable certain courses. The table also shows the status of each course.

The editing page can be seen in Figures 28, 29, 30 and 31. The details that can be edited per course are the name, description and start/end dates. Scrolling down, the user can see a table with three tabs, each of which represents the users, topics/mastery levels and groups associated with that particular course, all of which can be edited, provided that the user has such permissions. Creating a course, on the other hand, requires that the user fill in a valid name, course code, description and start/end dates, as can be seen in Figure 32.



Select role
ADMIN

Courses

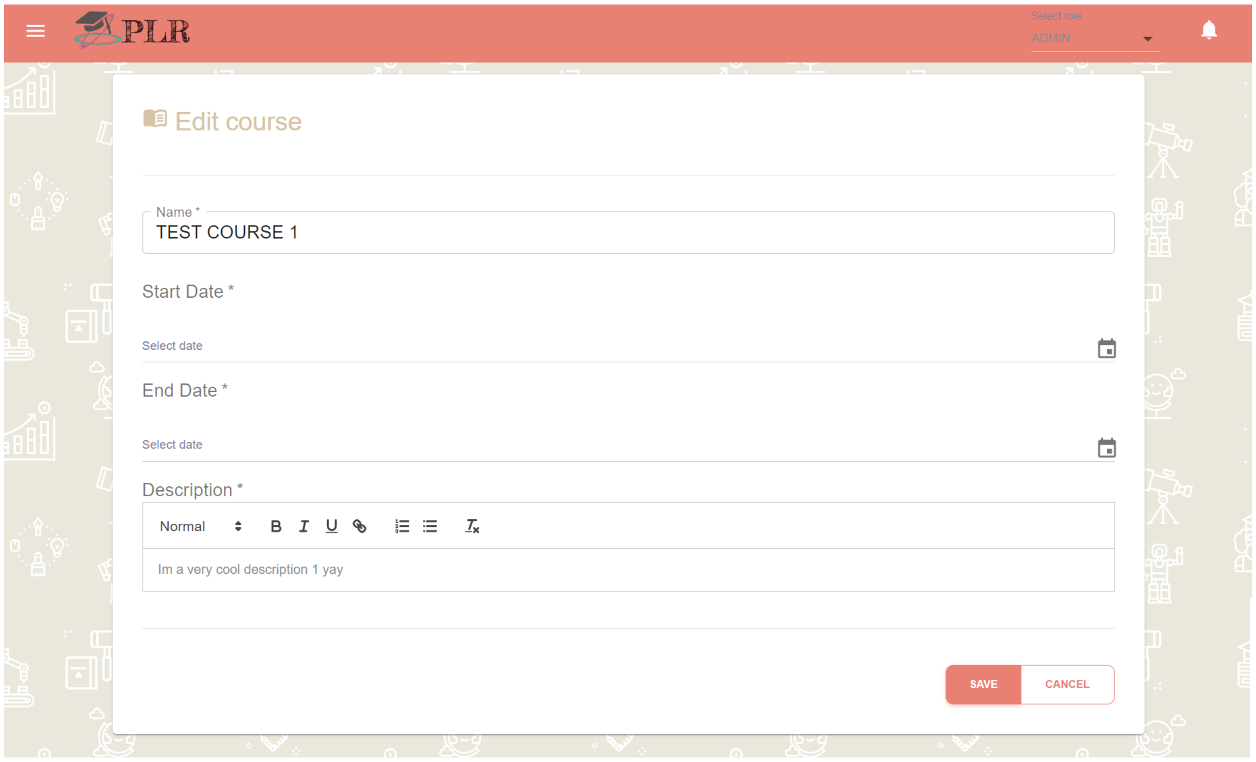
15 entries per page

Search...

	NAME	ACTIVE	LOCKED	DESCRIPTION	TERM
☐	TEST COURSE 1	🟩	🟥	Im a very cool descriptio...	
☐	Plamen Course	🟥	🟥	<p>Nice Course</p>	
☐	Second Trial Course	🟥	🟥	<p>FJASFKFHA</p>	
☐	TEST COURSE 2	🟩	🟥	<p>Im a very cool descri...	

Showing 1 to 15 of 4 entries

Figure 27. Courses overview page



Select role
ADMIN

Edit course

Name *
TEST COURSE 1

Start Date *
Select date

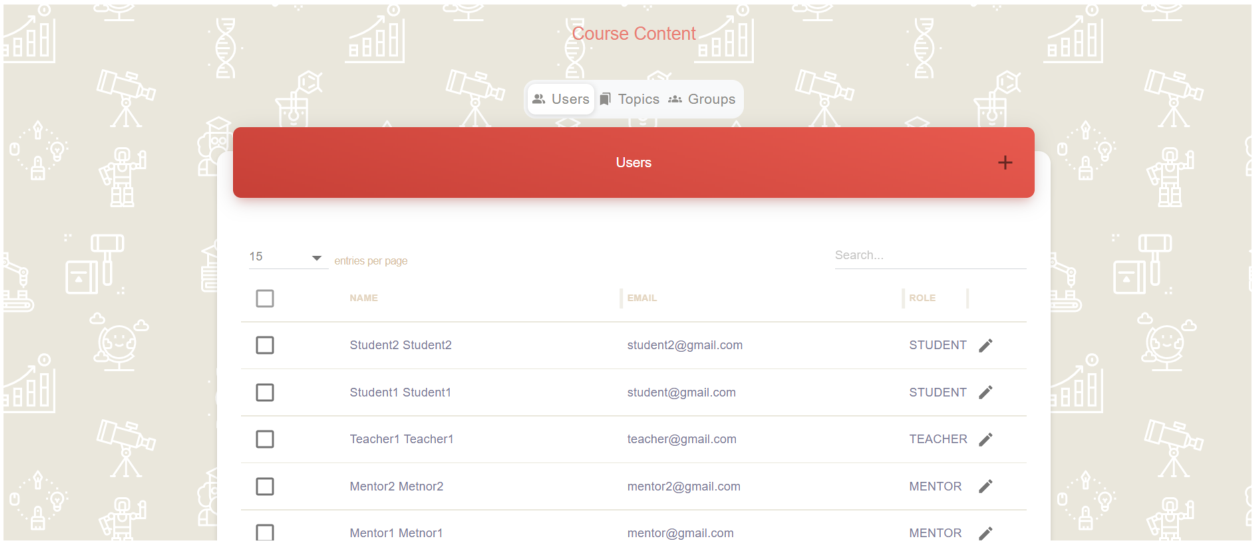
End Date *
Select date

Description *
Normal

Im a very cool description 1 yay

SAVE CANCEL

Figure 28. Editing course details


Course Content

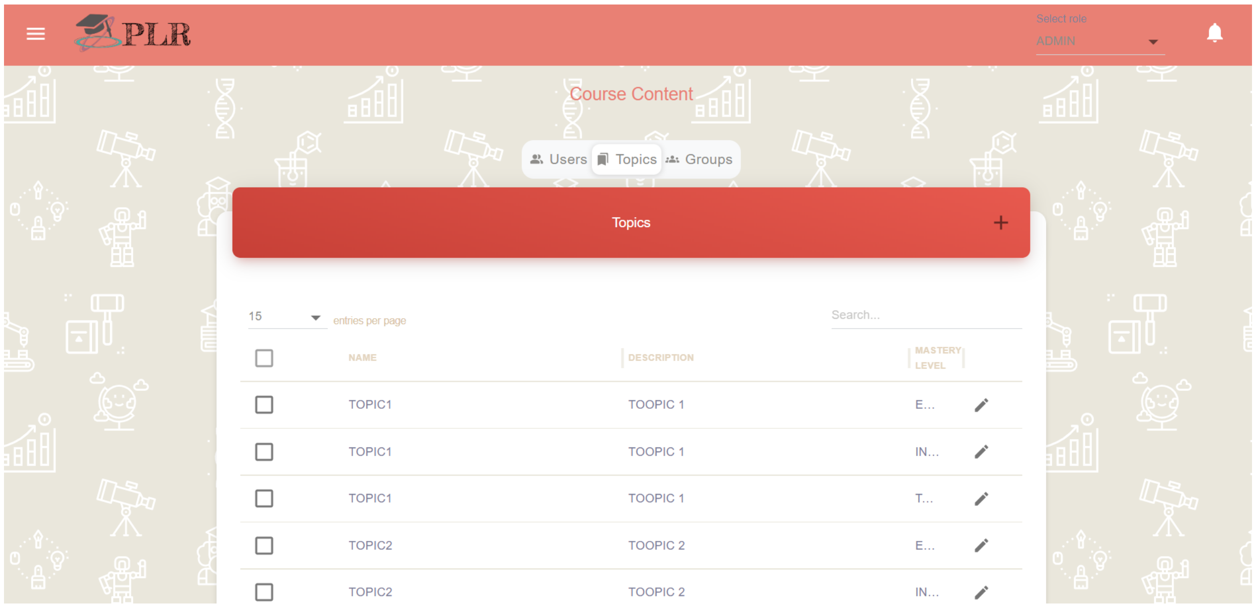
Users Topics Groups

Users +

15 entries per page — Search...

	NAME	EMAIL	ROLE
☐	Student2 Student2	student2@gmail.com	STUDENT
☐	Student1 Student1	student@gmail.com	STUDENT
☐	Teacher1 Teacher1	teacher@gmail.com	TEACHER
☐	Mentor2 Metnor2	mentor2@gmail.com	MENTOR
☐	Mentor1 Metnor1	mentor@gmail.com	MENTOR

Figure 29. Editing course tables - users table


PLR — Select role ADMIN

Course Content

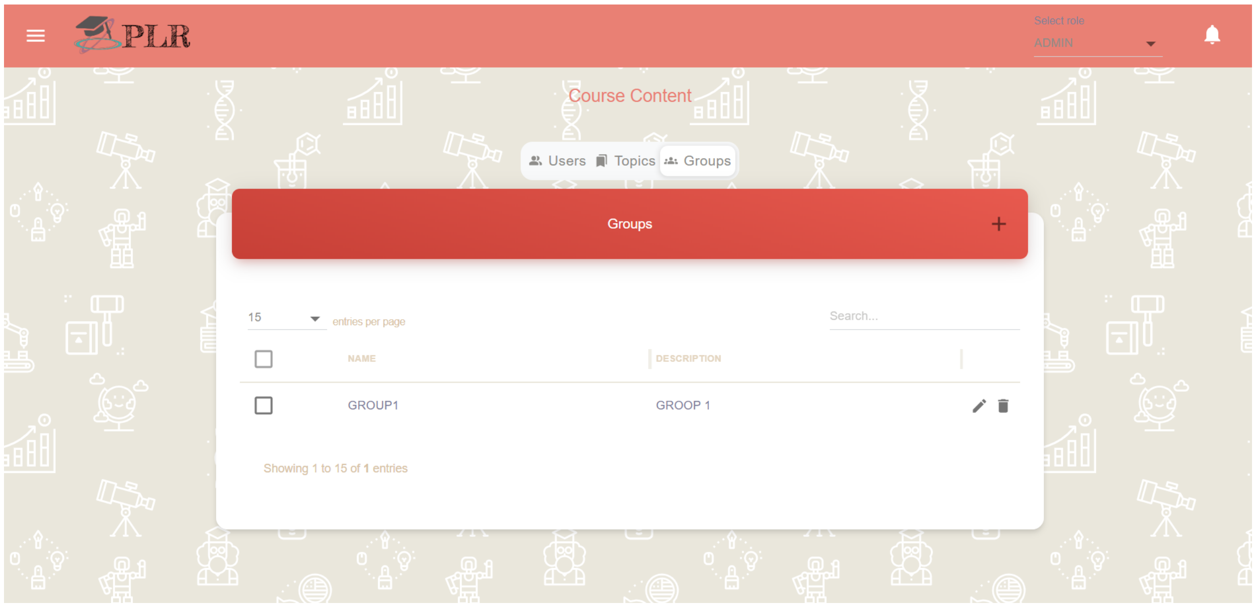
Users Topics Groups

Topics +

15 entries per page — Search...

	NAME	DESCRIPTION	MASTERY LEVEL
☐	TOPIC1	TOOPIC 1	E...
☐	TOPIC1	TOOPIC 1	IN...
☐	TOPIC1	TOOPIC 1	T...
☐	TOPIC2	TOOPIC 2	E...
☐	TOPIC2	TOOPIC 2	IN...

Figure 30. Editing course tables - topics table


PLR

Select role

ADMIN

Course Content

Users Topics Groups

Groups

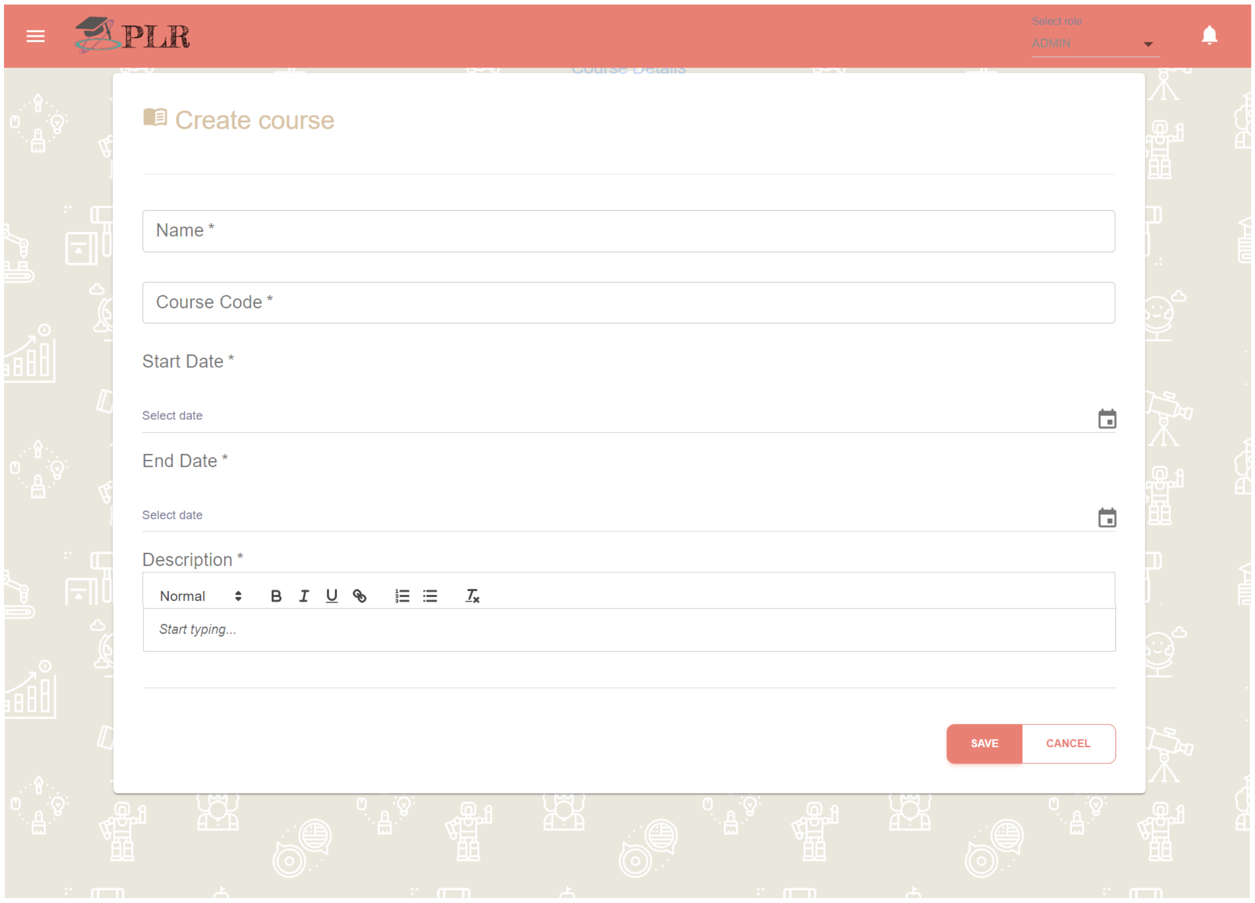
15 entries per page

Search...

	NAME	DESCRIPTION	
☐	GROUP1	GROOP 1	

Showing 1 to 15 of 1 entries

Figure 31. Editing course tables - groups table


PLR

Select role

ADMIN

Create course

Name *

Course Code *

Start Date *

Select date

End Date *

Select date

Description *

Normal B I U

Start typing...

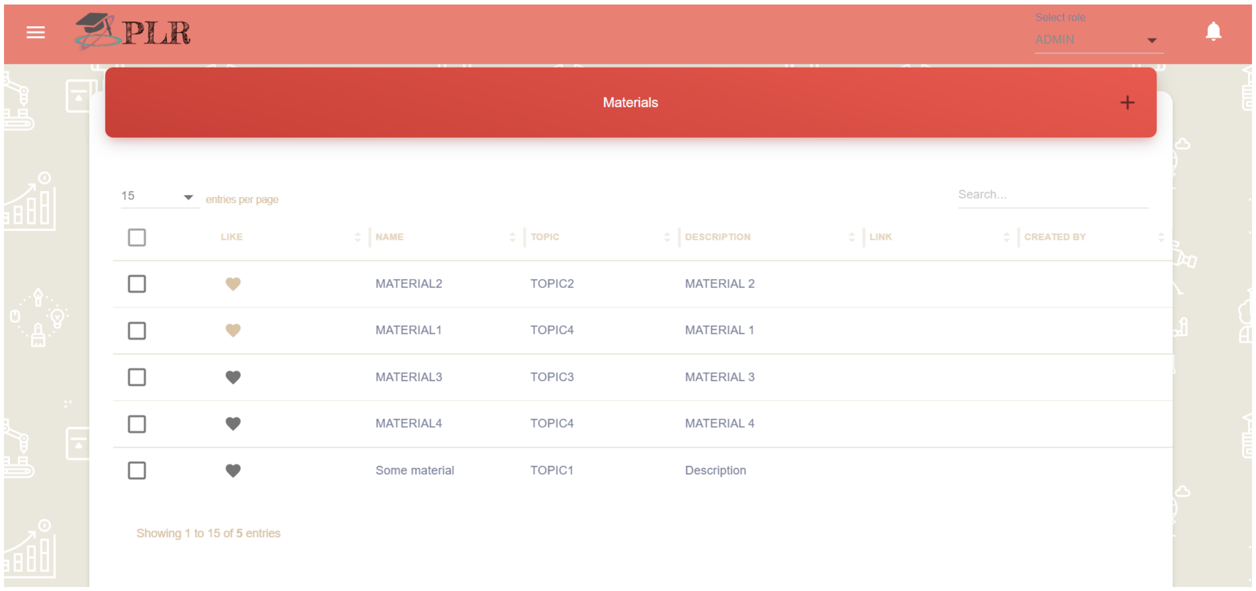
SAVE CANCEL

Figure 32. Creating a course

6.7. Materials

The materials page shows all materials for each course, intended to support students in their learning process. It can be accessed through the Materials button in the navigation bar. The materials overview can be seen in Figure 33. In this overview, users can save their favorite materials by clicking the like button. Additionally, materials can be viewed (as seen in Figure 34), edited and deleted.

Editing a material can be seen in Figure 35. The user must enter a name and a description, as well as attach it to a specific topic that the material is intended to support. Optionally, a link and text can be added, depending on the material. Creating a material happens in effectively the same manner.



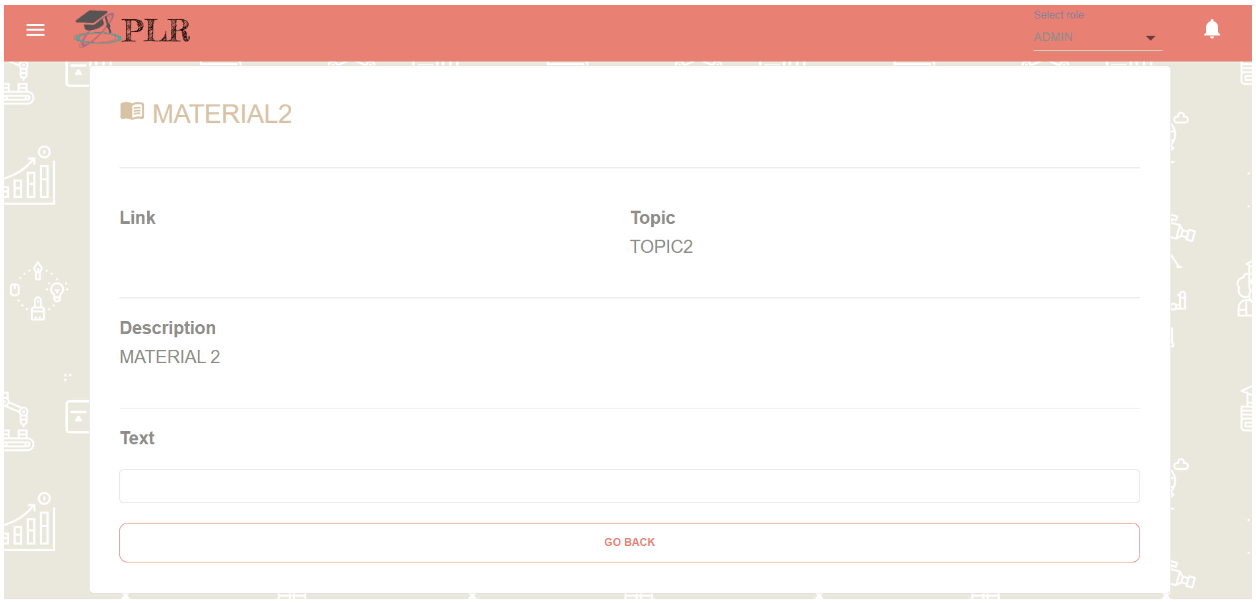
15 entries per page

Search...

	LIKE	NAME	TOPIC	DESCRIPTION	LINK	CREATED BY
☐	♥	MATERIAL2	TOPIC2	MATERIAL 2		
☐	♥	MATERIAL1	TOPIC4	MATERIAL 1		
☐	♥	MATERIAL3	TOPIC3	MATERIAL 3		
☐	♥	MATERIAL4	TOPIC4	MATERIAL 4		
☐	♥	Some material	TOPIC1	Description		

Showing 1 to 15 of 5 entries

Figure 33. Materials overview page



MATERIAL2

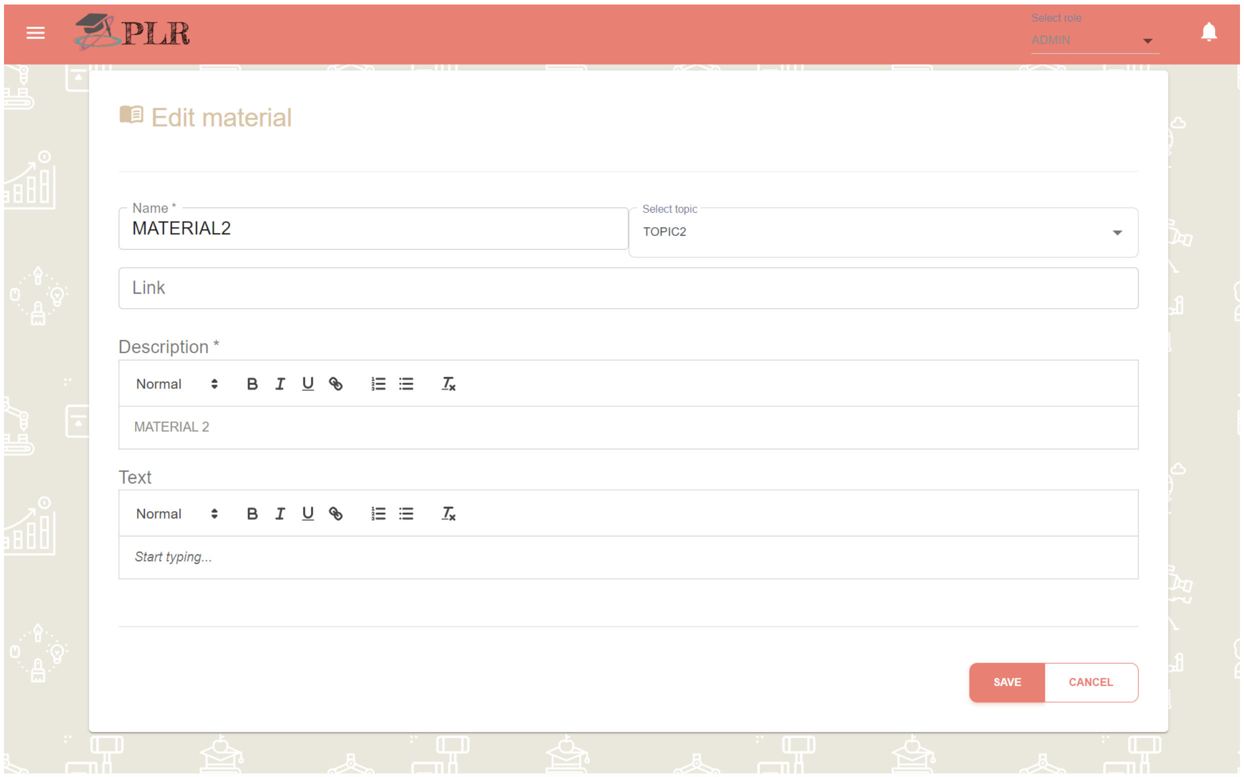
Link

Topic
TOPIC2

Description
MATERIAL 2

Text

Figure 34. Viewing a material



Edit material

Name * MATERIAL2

Select topic TOPIC2

Link

Description *
MATERIAL 2

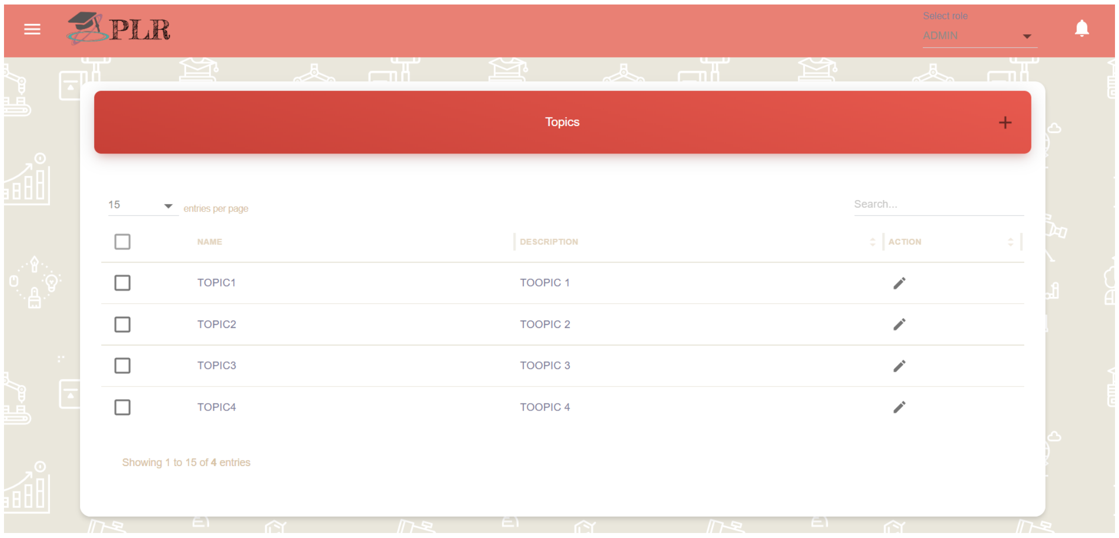
Text
Start typing...

Figure 35. Editing a material

6.8. Topics

The pages discussed in this section can be reached by selecting the Topics option in the navigation bar. The first page, shown on Figure 36, presents to the user an overview of all topics accessible to said user.

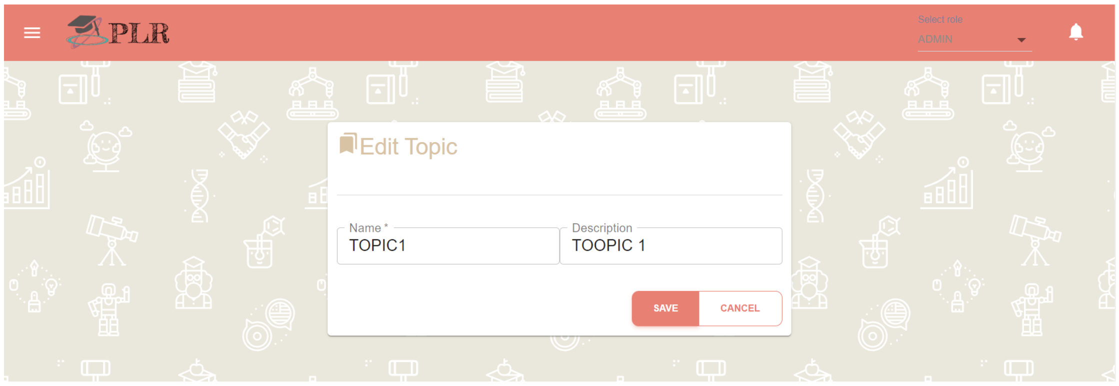
Each of these topics can be edited, which redirects the user to the page shown on Figures 37, 38 and 39. First, the user can edit the details of the selected topic, namely its name and description. Then, in the tables below those editing fields, the user can detach materials from the topic and edit its corresponding courses. Editing a course leads the user to the page shown in Figure 40, where the user can edit the mastery levels associated with the selected topic in the selected course.



	NAME	DESCRIPTION	ACTION
☐	TOPIC1	TOOPIC 1	✎
☐	TOPIC2	TOOPIC 2	✎
☐	TOPIC3	TOOPIC 3	✎
☐	TOPIC4	TOOPIC 4	✎

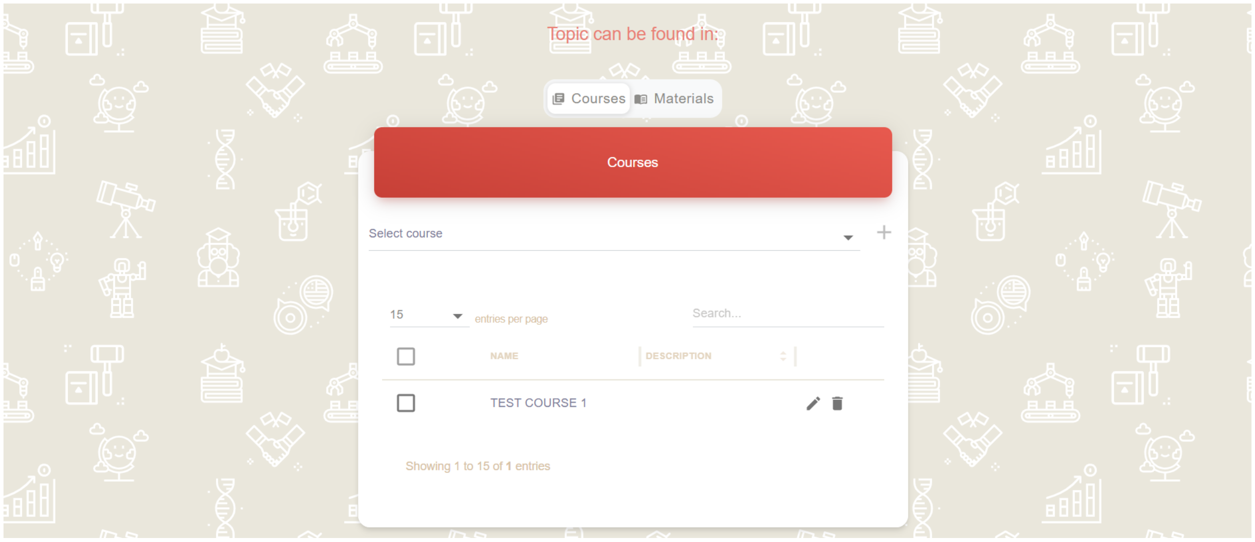
Showing 1 to 15 of 4 entries

Figure 36. Topics overview page



Name *: TOPIC1
Description: TOOPIC 1

Figure 37. Editing topic details


Topic can be found in:

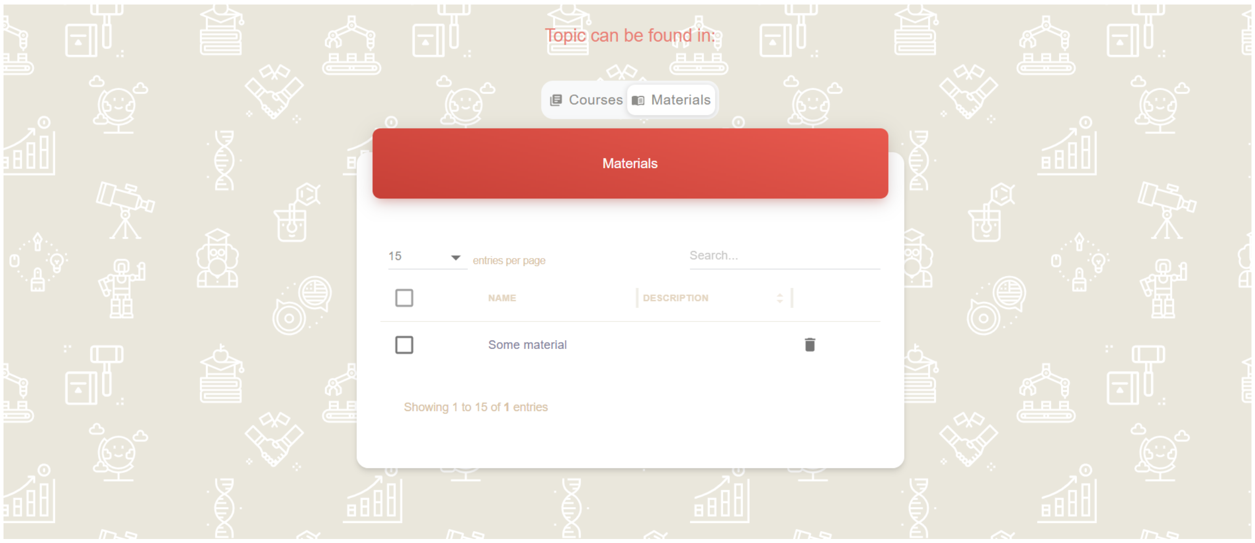
	NAME	DESCRIPTION
☐	TEST COURSE 1	

15 entries per page

Search...

Showing 1 to 15 of 1 entries

Figure 38. Editing courses related to a topic


Topic can be found in:

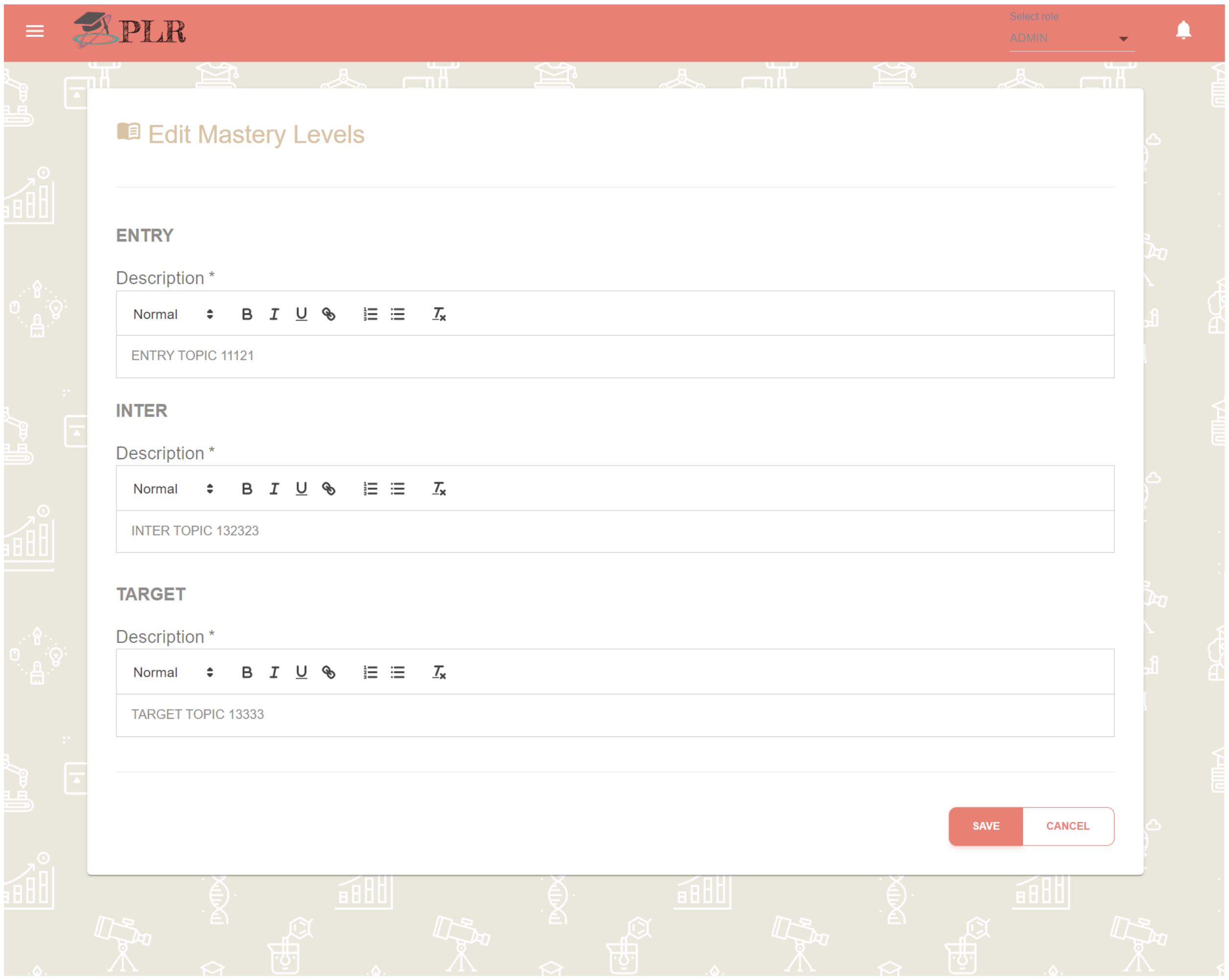
	NAME	DESCRIPTION
☐	Some material	

15 entries per page

Search...

Showing 1 to 15 of 1 entries

Figure 39. Editing materials related to a topic



Edit Mastery Levels

ENTRY

Description *

ENTRY TOPIC 11121

INTER

Description *

INTER TOPIC 132323

TARGET

Description *

TARGET TOPIC 13333

Figure 40. Editing mastery levels of a topic

6.9. CSV Import

Another feature of the system is the CSV import, shown on Figure 41, which allows the user to upload a file with a .csv extension from their device, which imports the data in that file to the selected target course and target group. The system is then populated with the data in the file.

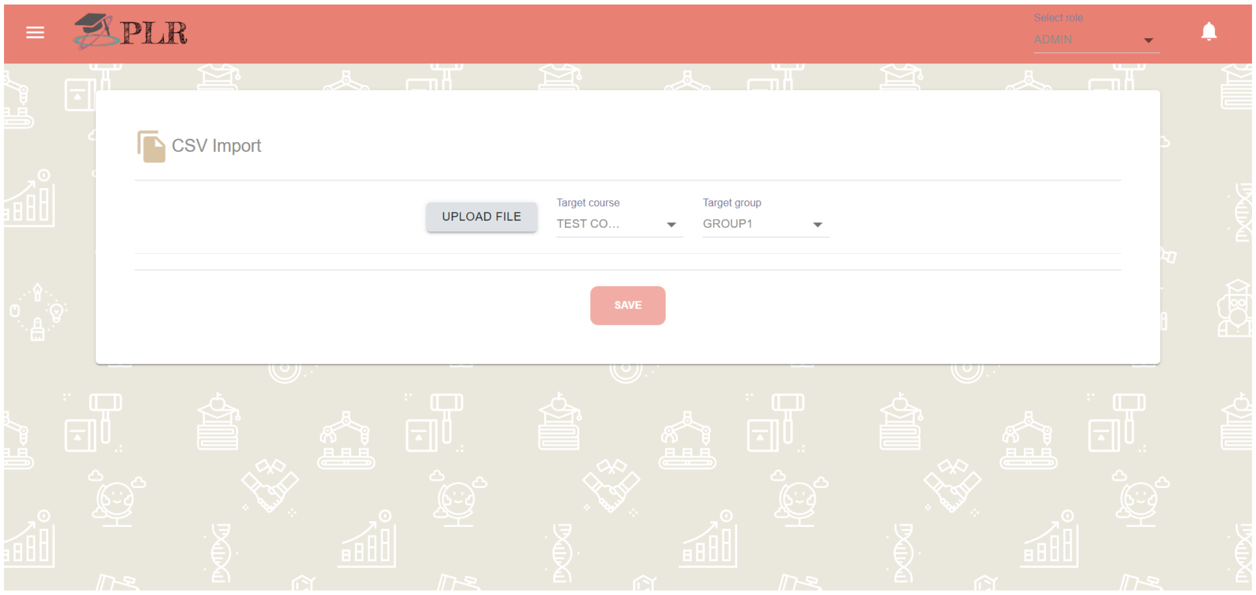


Figure 41. CSV import page

6.10. Statistics

The page containing the statistics is reachable from the Statistics tab in the navigation bar, as seen in Figures 42 and 43. It contains information similar to the one shown in the landing page for mentors, teachers and administrators. In particular, the first part of the page shows the same graphs for Mastery over time and Current mastery, with the distinction of giving the option for more filters - filtering by course, group and users.

The second part of the page contains a history of the feedback given to students - by the students to themselves, by their mentors or by teachers and administrators. It shows the submission date and time, as well as an indication of whether the feedback given was a self-assessment or not. Additionally, more details about the feedback can be seen by clicking on the view icon next to the feedback entry in the table, a snippet of which can be seen in Figure 44.

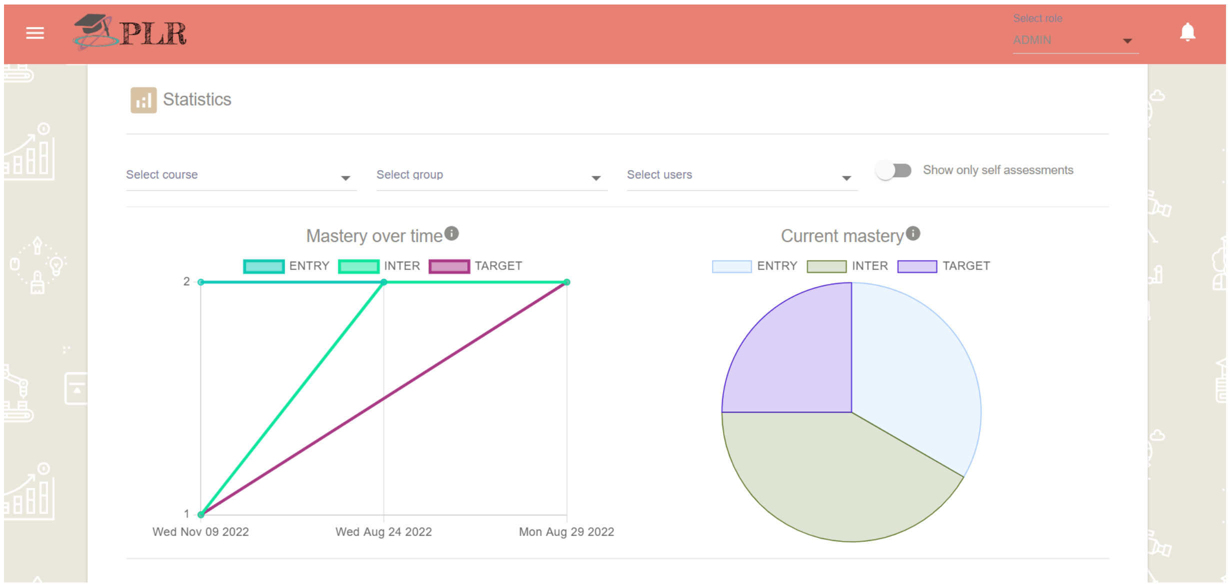
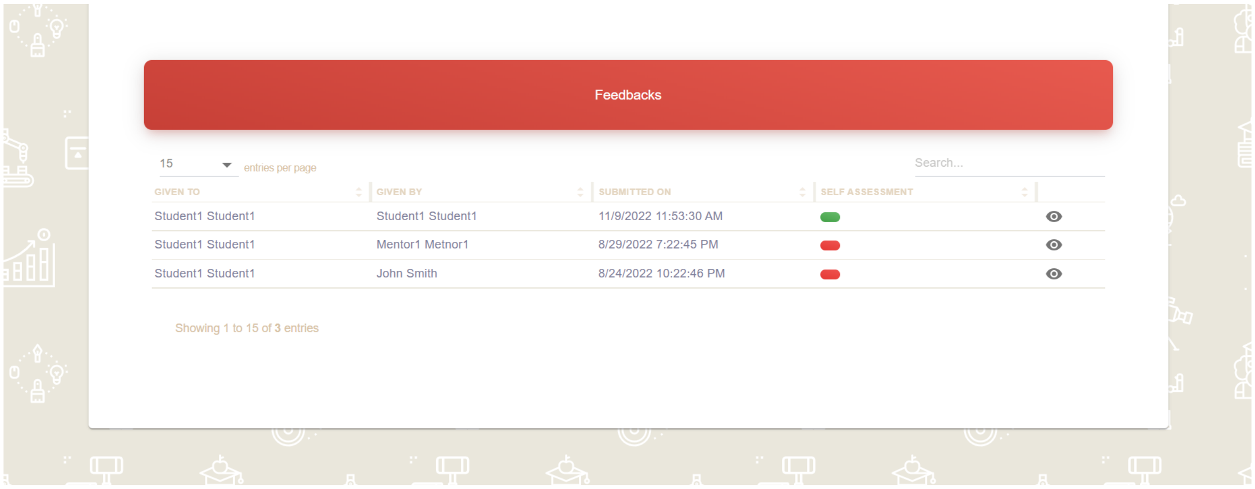


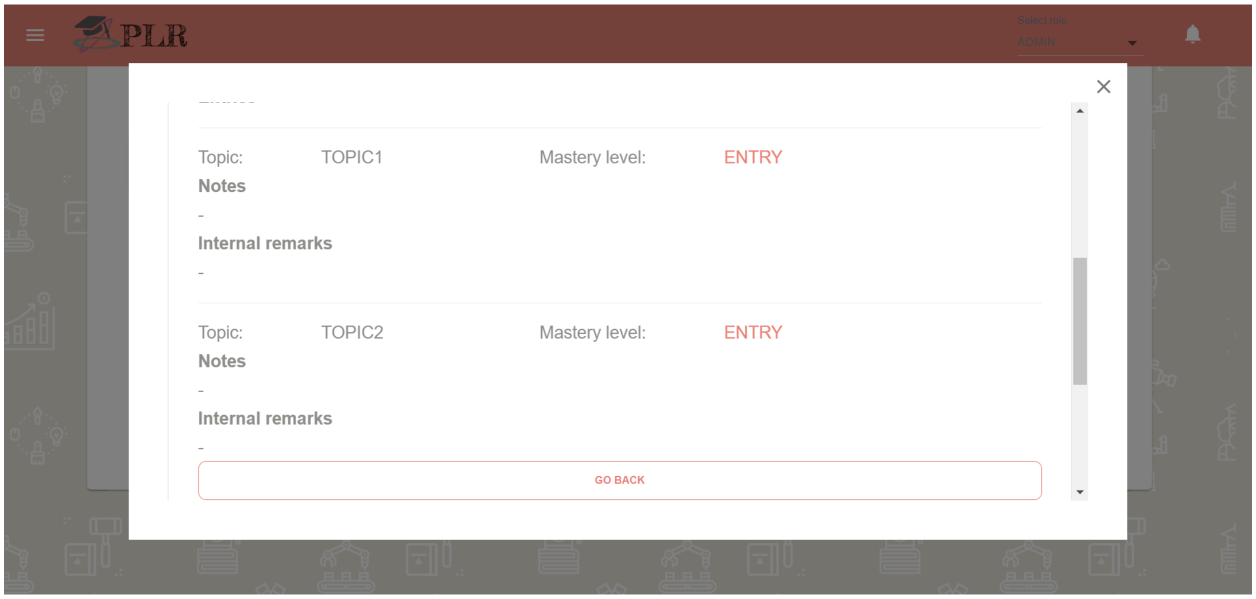
Figure 42. Statistics page - graphs section



GIVEN TO	GIVEN BY	SUBMITTED ON	SELF ASSESSMENT	
Student1 Student1	Student1 Student1	11/9/2022 11:53:30 AM	🟩	👁
Student1 Student1	Mentor1 Metnor1	8/29/2022 7:22:45 PM	🟥	👁
Student1 Student1	John Smith	8/24/2022 10:22:46 PM	🟥	👁

Showing 1 to 15 of 3 entries

Figure 43. Statistics page - feedback section



Topic: TOPIC1 Mastery level: ENTRY
Notes
-
Internal remarks
-

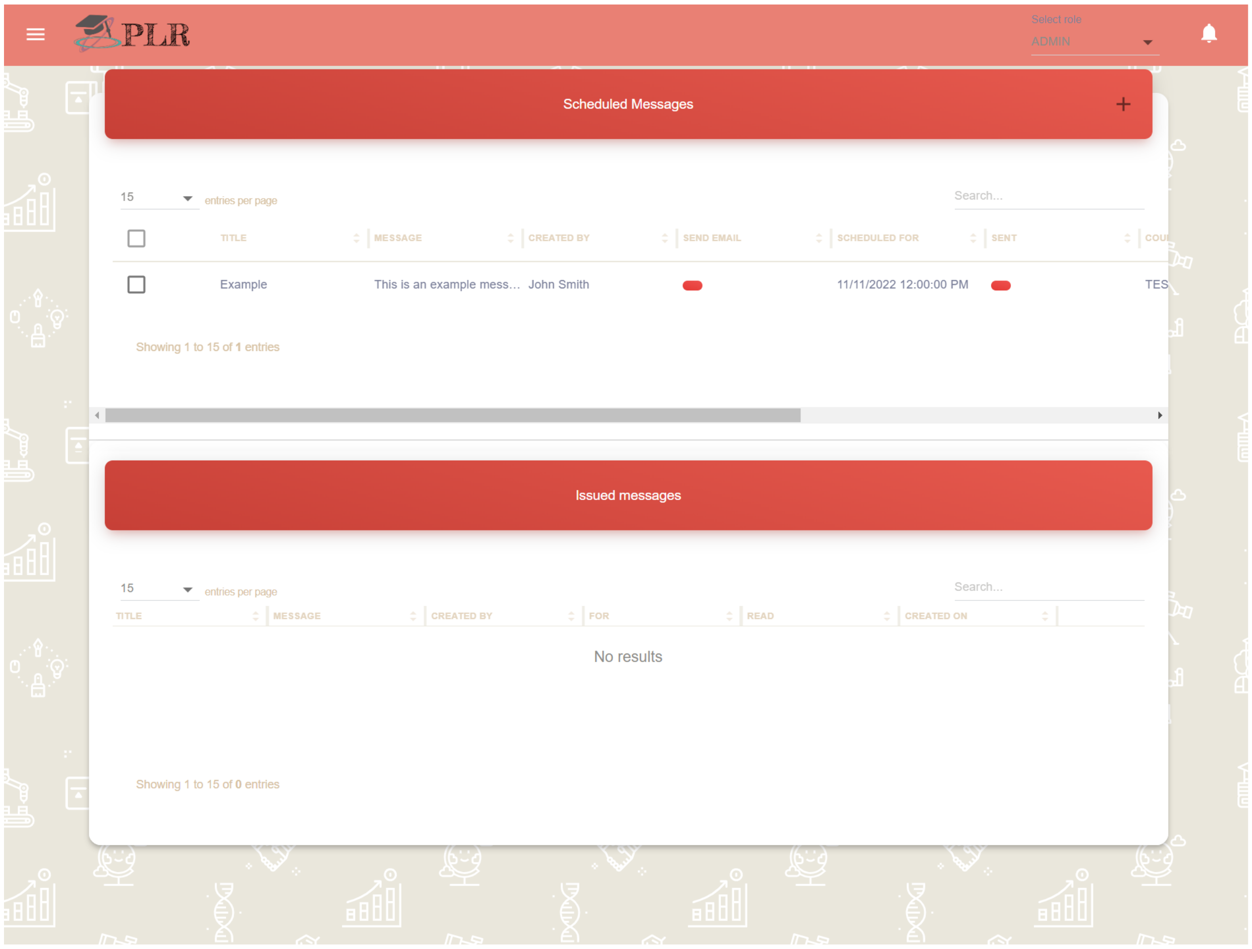
Topic: TOPIC2 Mastery level: ENTRY
Notes
-
Internal remarks
-

Figure 44. Part of the feedback view

6.11. Messages

Users with the right access have the option to create messages of two types - scheduled and instant. A message of either type can be created by clicking on the + icon of the Scheduled Messages table (as seen in Figure 45), which leads to the page shown in Figure 46.

On that page, users can enter the details of the message they want to create - its title, the course it should be sent to and its contents, as well as, optionally, the role and/or of the users who should receive the message, and, if the message should be scheduled, the date and time at which it should be sent to its intended recipients. Additionally, users can indicate whether the message should be sent exclusively through the website, or if an email should be sent to the recipients as well. An example of a scheduled message that has yet to be sent can be seen in the Scheduled Messages table in Figure 45.


Scheduled Messages

15 entries per page

	TITLE	MESSAGE	CREATED BY	SEND EMAIL	SCHEDULED FOR	SENT	COU
☐	Example	This is an example mess...	John Smith	🟥	11/11/2022 12:00:00 PM	🟥	TES

Showing 1 to 15 of 1 entries

Issued messages

15 entries per page

TITLE	MESSAGE	CREATED BY	FOR	READ	CREATED ON

No results

Showing 1 to 15 of 0 entries

Figure 45. Messages overview page

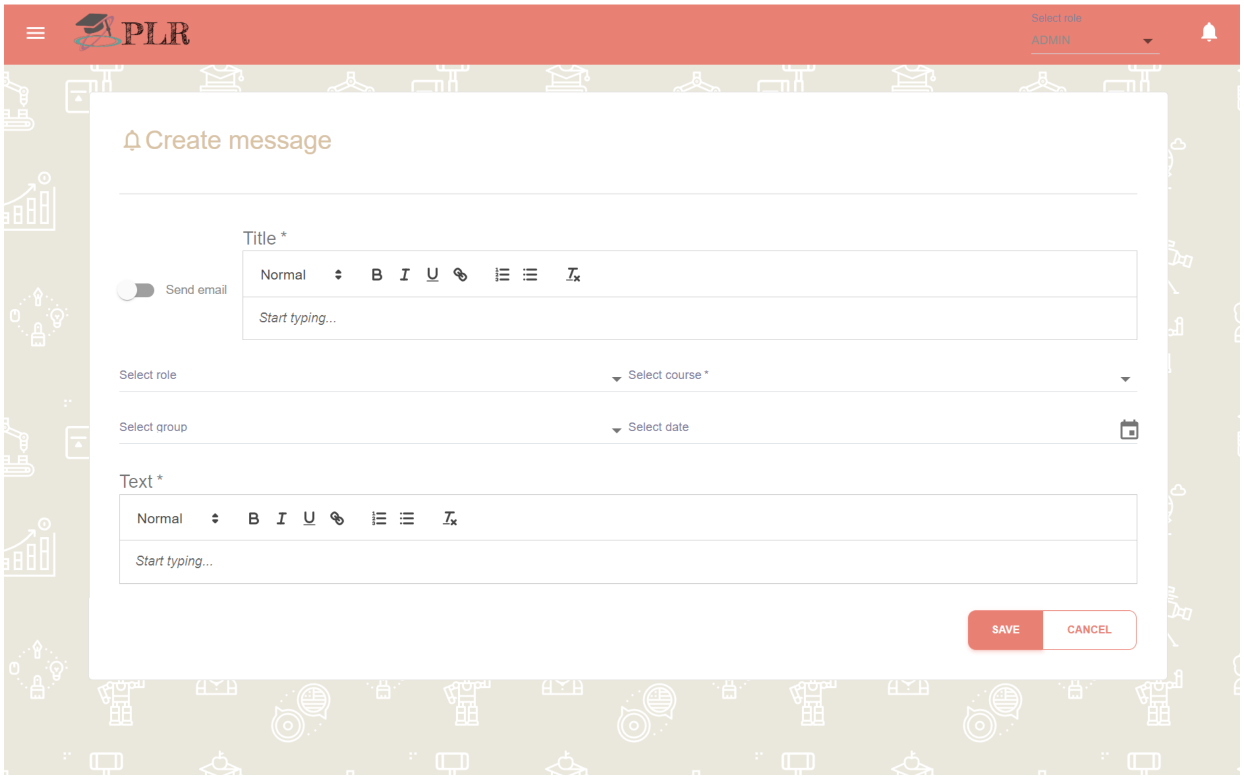


Figure 46. Creating a new message

6.12. Feedback

In this section, the feedback process is described from the perspective of both a student giving themselves feedback, as well as mentors/teacher/administrators. Regardless of the perspective, the feedback process can be started by clicking on the Feedback button in the navigation bar. Alternatively for students, feedback can be started directly from the landing page for their convenience, since for students that is the primary purpose of using the system.

6.12.1. Student Perspective

The process for a student begins from the page shown in Figure 47. There, the student can select which course of those they are enrolled in they would like to self-assess for. Additionally, students have the option to mark a course as a favorite for faster access to it.

After selecting a course, the students can begin the self-assessment. They can give themselves general remarks for the course, which can be seen in Figure 48, and furthermore, they can select their level of proficiency at a specific topic and give remarks for that particular topic, which can be seen in Figure 49.

Additionally, students can see a history of their feedback by clicking on the History button to the left of the general self-assessment. Students can also request feedback from their mentors, which prompts a notification on the mentors' side. By clicking on the Save button, the self-assessment process is complete for the student.

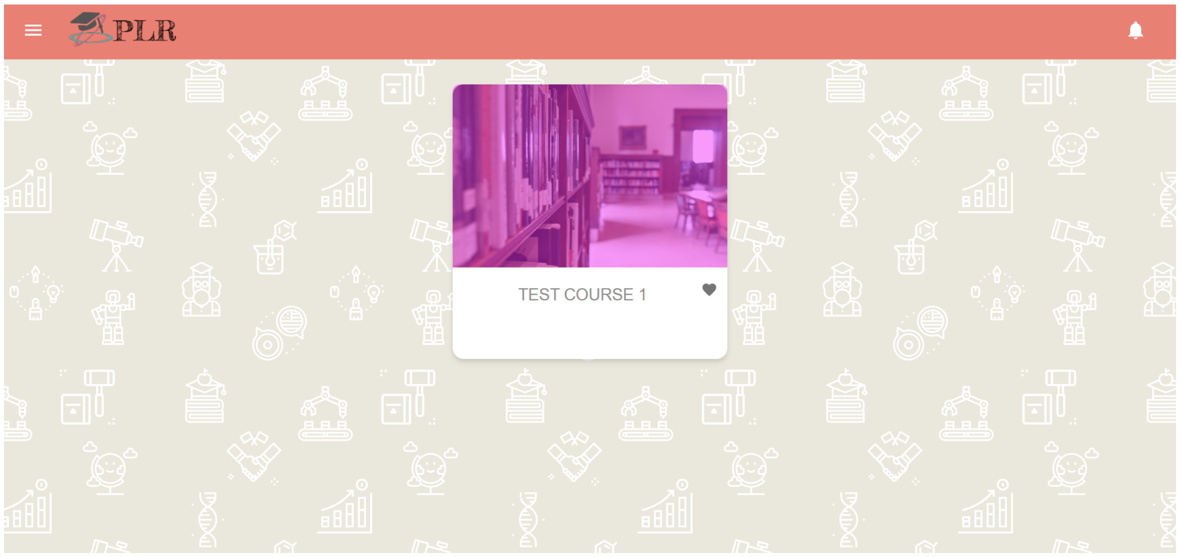
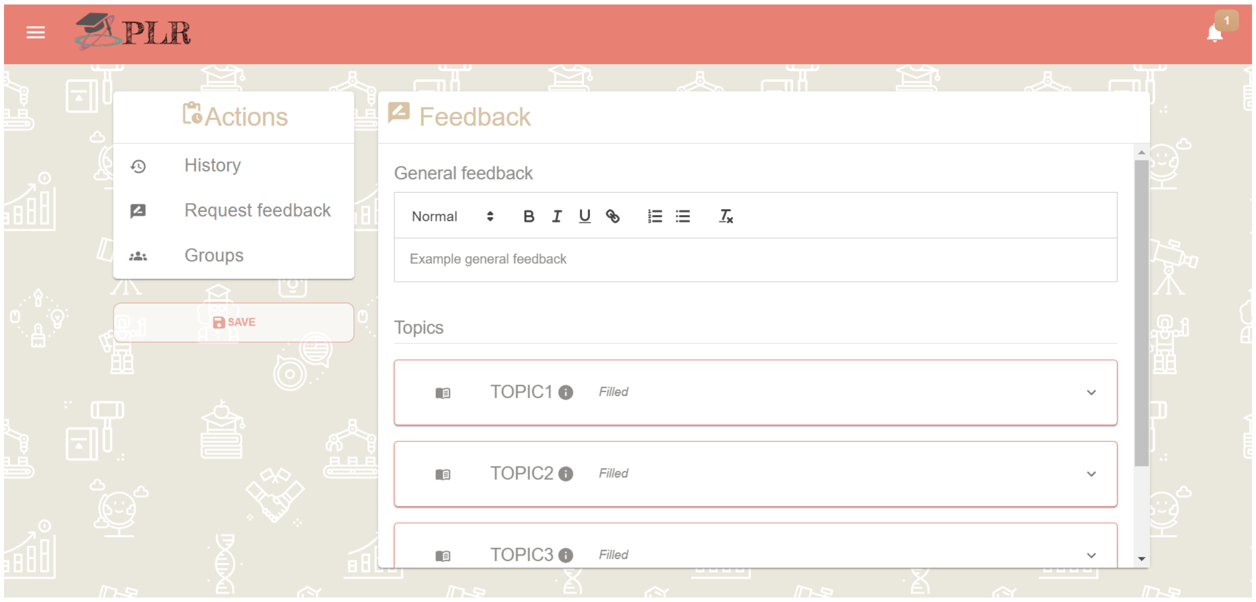
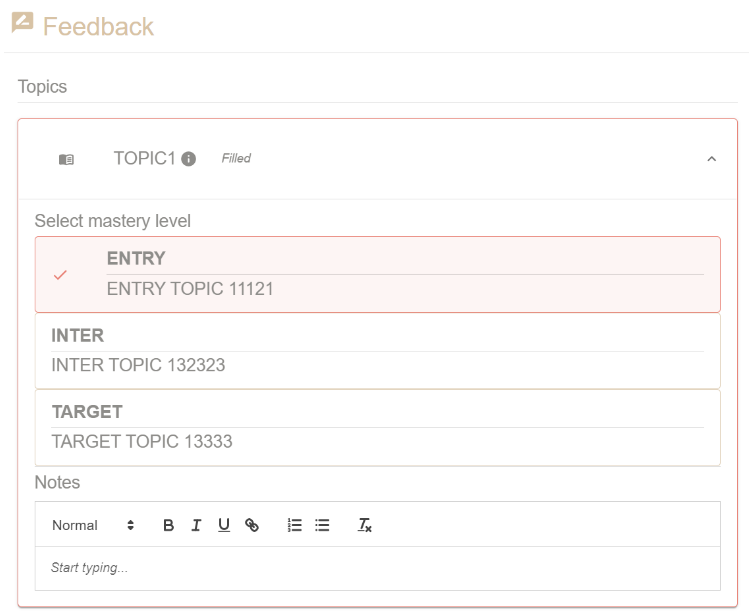


Figure 47. Feedback process - selecting a course



Example general feedback

Figure 48. Submitting self-assessment - general feedback and overview


Feedback

Topics

TOPIC1 Filled

Select mastery level

✓ ENTRY
ENTRY TOPIC 11121

INTER
INTER TOPIC 132323

TARGET
TARGET TOPIC 13333

Notes

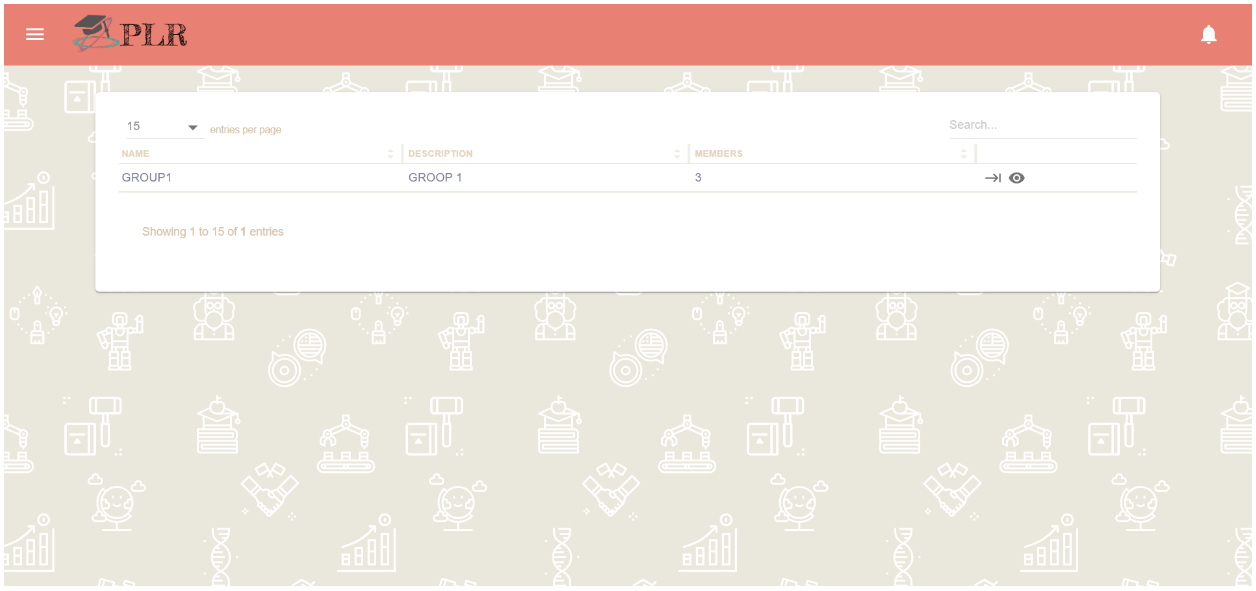
Figure 49. Submitting self-assessment - for a specific topic.

6.12.2. Mentor/Teacher/Administrator Perspective

The process of giving feedback to a student is similar for the mentors, teacher and administrators, which is why it will be discussed in the same section. The process begins in a similar way as it does for the students - a page with all the associated courses is shown, much like the one in Figure 47. The user then selects the course they wish to give or view feedback for, which brings them to a page where they can see the groups for that particular course, as shown in Figure 50.

When the user wants to proceed with a specific group, they can either view the group's details by clicking the View icon (Figure 51), or they can proceed by clicking the button next to it, which takes them to the page shown in Figure 52. On that page, the user can select which user they would like to give feedback to.

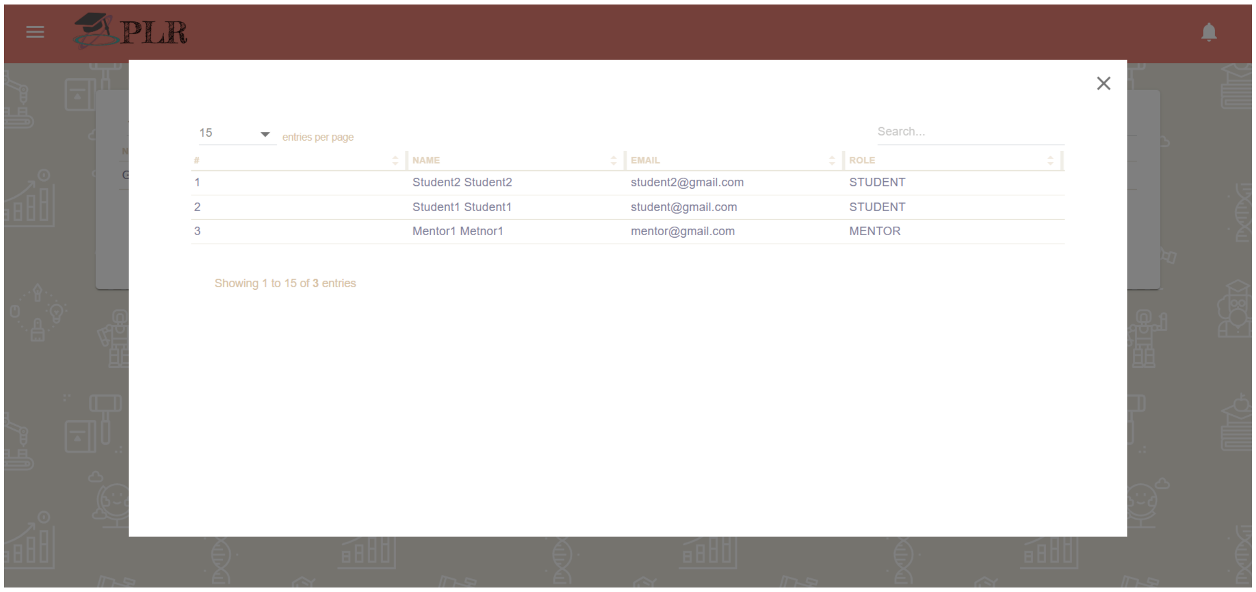
Finally, the feedback can be given following the depiction in Figure 53. The primary difference between the process of a student filling in a self-assessment and a mentor/teacher/administrator giving feedback is in the internal remarks sections - the user can give feedback towards the student, but that feedback is only visible to other mentors/teachers/administrators. The user is informed of this by hovering over the Information icon. Other than that, the process is the same, only the perspective is different.



NAME	DESCRIPTION	MEMBERS	
GROUP1	GROOP 1	3	

Showing 1 to 15 of 1 entries

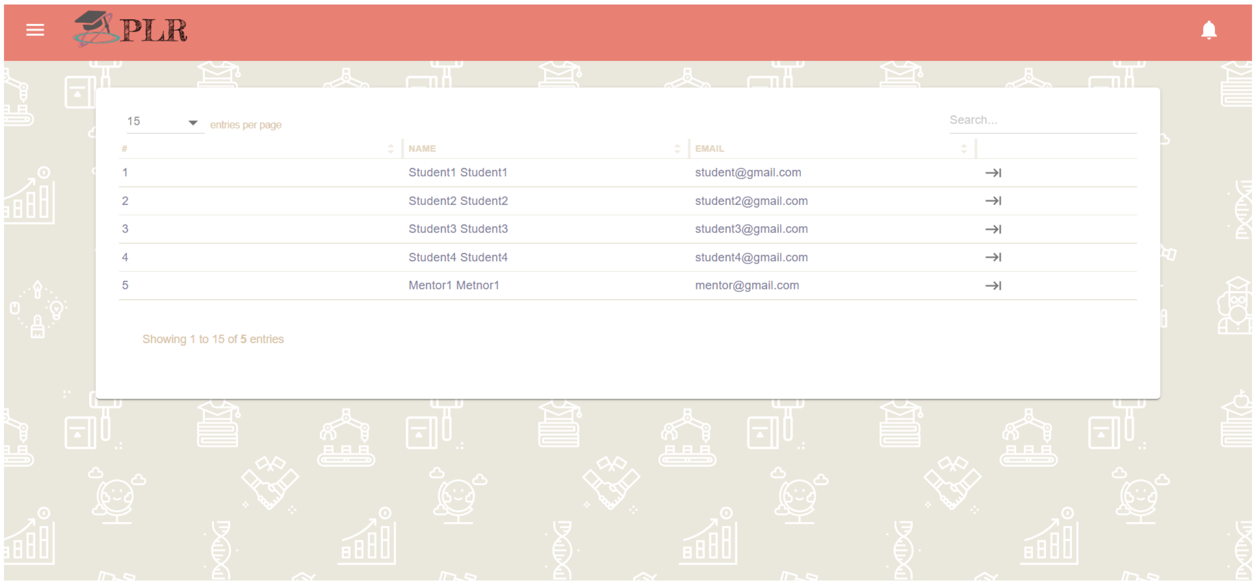
Figure 50. Groups overview for a course



#	NAME	EMAIL	ROLE
1	Student2 Student2	student2@gmail.com	STUDENT
2	Student1 Student1	student@gmail.com	STUDENT
3	Mentor1 Metnor1	mentor@gmail.com	MENTOR

Showing 1 to 15 of 3 entries

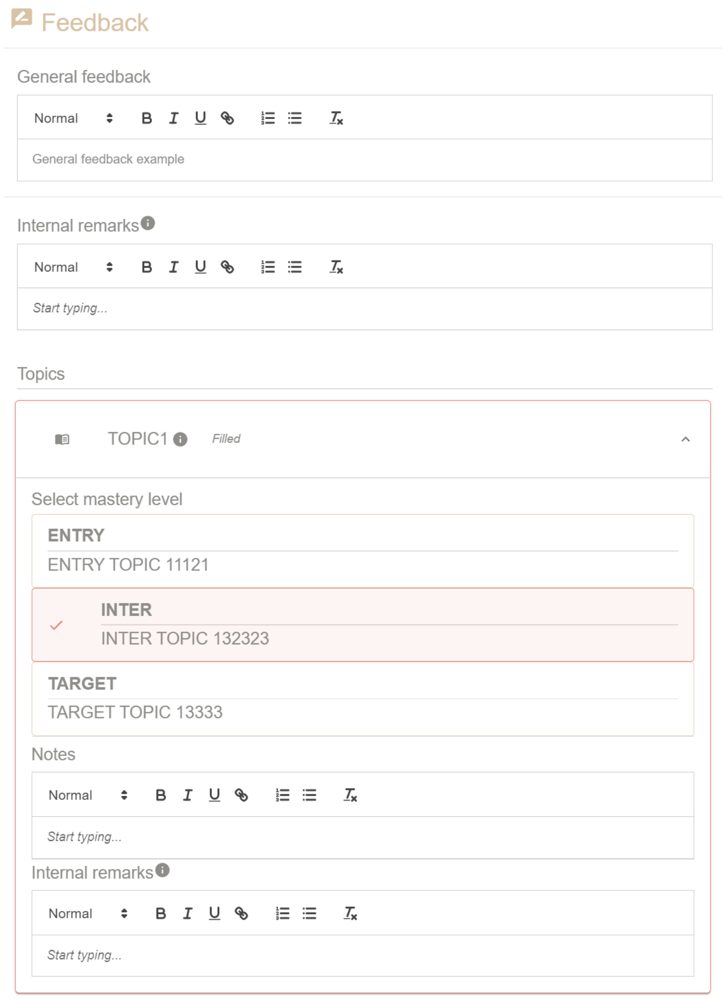
Figure 51. Viewing a group



#	NAME	EMAIL	
1	Student1 Student1	student@gmail.com	→|
2	Student2 Student2	student2@gmail.com	→|
3	Student3 Student3	student3@gmail.com	→|
4	Student4 Student4	student4@gmail.com	→|
5	Mentor1 Metnor1	mentor@gmail.com	→|

Showing 1 to 15 of 5 entries

Figure 52. Selecting a user to give feedback to



General feedback

General feedback example

Internal remarks

Start typing...

TOPIC1 *Filled*

ENTRY
ENTRY TOPIC 11121

✓ INTER
INTER TOPIC 132323

TARGET
TARGET TOPIC 13333

Notes

Start typing...

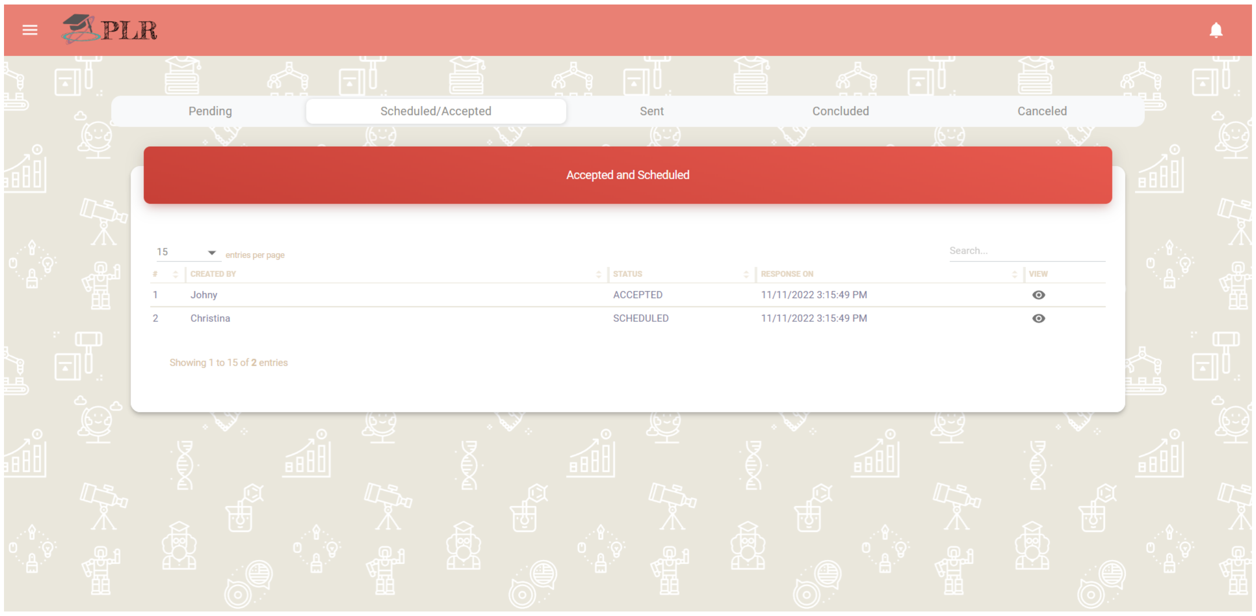
Internal remarks

Start typing...

Figure 53. Giving feedback to a user

6.13. Requests

Students are able to submit requests for feedback to their mentor(s) through the feedback view or alternatively through the screen at Figure 54. Alternatively they can also request a checkpoint meeting to be scheduled. The aforementioned screen hosts all the various events connected to requests that might take place, such as currently accepted feedback or scheduled meetings. Also, it has the previously conducted meetings, so mentors can easily check the previous checkpoint meetings conducted. Furthermore, it's easy for mentors to ping students too, if they have to self-assess, or respond to a checkpoint meeting that has to be scheduled.



Accepted and Scheduled

15 entries per page

Search...

#	CREATED BY	STATUS	RESPONSE ON	VIEW
1	Johny	ACCEPTED	11/11/2022 3:15:49 PM	👁
2	Christina	SCHEDULED	11/11/2022 3:15:49 PM	👁

Showing 1 to 15 of **2** entries

Figure 54. Viewing and responding to requests

6.14. Mobile Version

A requirement by the client that was deemed nice-to-have is mobile friendliness, which the team was able to implement. As such, the application can be run on various types of devices without the user encountering any issues.

7. Testing

7.1. Introduction

Since the beginning of the project testing was an integral part of the factors that we had to consider. On the one hand, we have to ensure a proper and error free experience of the user. On the other hand, we also have a small window to develop the system and using recurring testing throughout the process helps immensely with time management and keeping everything up and running at all times. As explored in the previous sections, our system packs quite a lot of features and with those features come a lot of complex systems behind the curtain. Those systems require different types of testing, fundamentally speaking. And we had to come up with a suite of methods to thoroughly test all aspects of the system.

In short our system is a full-stack application, meaning we have a backend (database) and frontend (user interface). These are fundamentally different to develop and thus test, using vastly different languages and frameworks. Our decision was to dedicate time after developing a feature to write and prepare new tests for both and rerun all to ensure everything is running smoothly at all times.

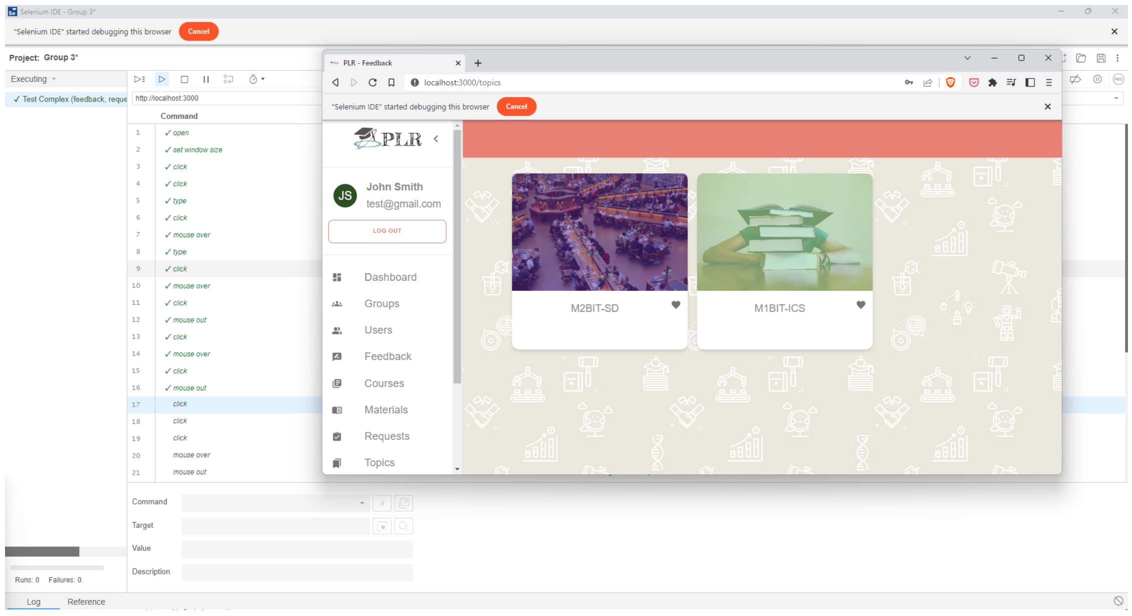
7.2. Testing the Backend

For the backend, as discussed, we used Spring Boot, and prepared our data structures with Java. As this is a standard procedure, we used a standard method to test them - regular JUnit tests. However, upon connecting the frontend with the backend via GraphQL and expanding our practices and highly modernizing it, we decided to use Mockito in order to properly simulate data modification and requests via the backend, so we can ensure the proper behavior. Mockito allowed us to properly test all dependencies and also repositories that our modern backend allows, and keep them in check at all times. Furthermore, its use was extended to ensure that the data collected is in a proper condition for the statistical analysis that could take place at any point.

7.3. Testing the Frontend

The frontend is arguably as important for the user as are the statistics and logging of the backend for the teacher/administrator. In order to ensure that the frontend was operational every step of the way we had to come up with a reproducible behavior for testing it. We used an IDE called Selenium that allows programmers to test the frontend as a real user would - clicking buttons, filling in text and any other possible functionality. In this way we can ensure that the use of the system can be kept intact as it will always try to perform exactly the same actions. As an example - we can set up a test for a user that logs in, goes and fills in a feedback for himself, submits it and logs out. That will be run always the same way so we can ensure that we don’t break the frontend by changing an element of an additional feature. We came up with dozens of tests going over every part of the system. Those tests were recorded once, and after that could be easily automated for testing with Selenium

As an example , that’s how that system works in practice - it runs your system and traces your previously recorded steps and replicates your testing in a window.


Selenium IDE - Group 3*

"Selenium IDE" started debugging this browser

Project: Group 3*

✓ Test Complex (feedback, reque

http://localhost:3000

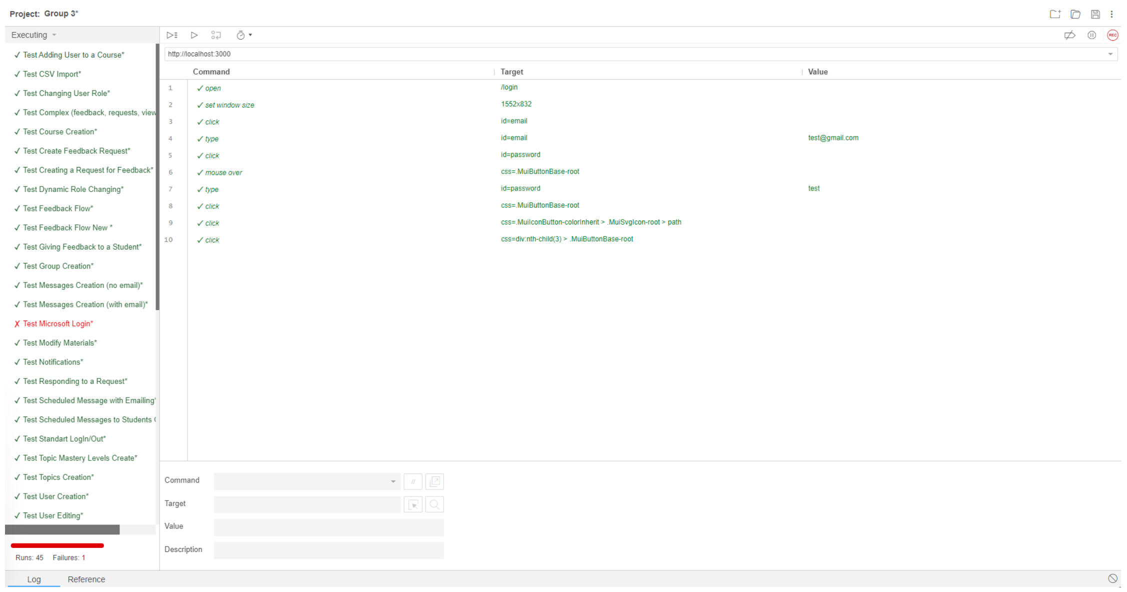
```
Command
1   ✓ open
2   ✓ set window size
3   ✓ click
4   ✓ click
5   ✓ type
6   ✓ click
7   ✓ mouse over
8   ✓ type
9   ✓ click
10  ✓ mouse over
11  ✓ click
12  ✓ mouse out
13  ✓ click
14  ✓ mouse over
15  ✓ click
16  ✓ mouse out
17    click
18    click
19    click
20    mouse over
21    mouse out
```

Runs: 0 Failures: 0

localhost:3000/topics

Figure . Frontend testing

Also we can show off how these steps look. Selenium operates on a per-step basis that allows us to easily identify issues in tests. As an example, we can look at the next screenshot. We recently made some changes to some elements used in different places in the system as well as our login screen. Out of 46 tests, 45 passed. However, we discovered that we had a bug with the Microsoft Login specifically from what was recently developed. Having that immediate feedback allowed us to find the issue and eliminate it on arrival, instead of debugging for hours later. We were able to see the specific element the test failed on and instantly narrow down the possible issues.



Project: Group 3*

Executing

- ✓ Test Adding User to a Course*
- ✓ Test CSV Import*
- ✓ Test Changing User Role*
- ✓ Test Complex (feedback, requests, view
- ✓ Test Course Creation*
- ✓ Test Create Feedback Request*
- ✓ Test Creating a Request for Feedback*
- ✓ Test Dynamic Role Changing*
- ✓ Test Feedback Flow*
- ✓ Test Feedback Flow New *
- ✓ Test Giving Feedback to a Student*
- ✓ Test Group Creation*
- ✓ Test Messages Creation (no email)*
- ✓ Test Messages Creation (with email)*
- ✗ Test Microsoft Login*
- ✓ Test Modify Materials*
- ✓ Test Notifications*
- ✓ Test Responding to a Request*
- ✓ Test Scheduled Message with Emailing*
- ✓ Test Scheduled Messages to Students (
- ✓ Test Standart LogIn/Out*
- ✓ Test Topic Mastery Levels Create*
- ✓ Test Topics Creation*
- ✓ Test User Creation*
- ✓ Test User Editing*

Runs: 45 Failures: 1

http://localhost:3000

	Command	Target	Value
1	✓ open	/login	
2	✓ set window size	1552x832	
3	✓ click	id=email	
4	✓ type	id=email	test@gmail.com
5	✓ click	id=password	
6	✓ mouse over	css=.MuiButtonBase-root	
7	✓ type	id=password	test
8	✓ click	css=.MuiButtonBase-root	
9	✓ click	css=.MuiIconButton-colorInherit > .MuiSvgIcon-root > path	
10	✓ click	css=div:nth-child(3) > .MuiButtonBase-root	

Figure . Frontend testing

7.4. Outcomes

Essentially, performing automated testing after developing a feature allowed us to keep everything already developed in check. Adding new features was seamless, and those were immediately debugged so are safely deployed to our main stable branch. Having multiple types of tests also allowed more granular detail in testing and gave complementary data so we can identify a bug in the matter of minutes. And our system overall, allowed us to have a production-ready system that is stable in just 8 weeks. That also emphasized the importance of testing from the very beginning and the fact that it should be done regularly to ensure a much smoother and faster development process, while also ensuring a more stable and enjoyable to use system.

8. Risk Assessment

In this section, the potential risks that the team can encounter during the process of development, as well as possible mitigation strategies, are outlined.

8.1. Delayed Development

The first risk discussed is also the one that the team deems most likely to happen, namely the possibility of the team running out of time before the project can be completed. The reason for this level of prioritization is that there are many possible causes for the deadline to be reached before finalizing the project. However, the most likely cause can be overestimation of the requirements, either in terms of difficulty or quantity, and either at the start of the project or as new requirements arise throughout the course of the development.

The team's mitigation strategy for this problem primarily involves being realistic in terms of estimating the difficulty of developing specific features - assigning weights to each task, thoroughly discussing their feasibility among the team, as well as assigning tasks according to each team member's strengths.

Additionally, the team's mitigation strategy against taking on more tasks than can be managed involves clear, frequent and honest communication with the client - the client is free to express his desired requirements throughout the course of the project, and team will do their best to accommodate and implement them, but if a requirement is unrealistic for the given timeframe, the team will let the client know as soon as possible, with the intent of finding a compromise that is more feasible, but will keep the client satisfied with the end result.

8.2. External Actors

The second risk that the team estimates to be likely to occur involves interruptions to the development process by external actors, either through lack of cooperation or communication, or due to time constraints. In particular, the team is concerned about the involvement of the LISA and Canvas LMS Integration teams of the UT for the Microsoft login using university credentials, as well as the Canvas LMS integration aspect of the system. The team's concern stems from the warnings given by teachers that obtaining the necessary credentials can be a slow process, as well as the fact that it is likely that the LISA and Canvas LMS integration teams will be quite busy throughout the duration of the course.

The team's mitigation strategy primarily involves getting in touch with the LISA and Canvas LMS integration teams as quickly as possible at the start of the project, as well as informing the client of the possibility that the implementation of these features can potentially be hindered due to miscommunication or time constraints that are outside the development team's control.

8.3. Security Risks

A risk that is present for many systems such as this one is involving security. This is a broad topic that can include many different types of security risks, such as data breach, incorrect role access, injection attacks, etc. Although security risks are always present and cannot be fully eliminated, steps can be taken to mitigate their impact and their likelihood to occur. In particular, the team has ensured that all aspects of the system are implemented according to security standards with the help of secure frameworks, as well as developing each page with various roles and their corresponding access in mind. Additionally, thorough testing has been performed to check that each role can only access pages and functionalities of the website that they should be able to view and interact with. As for privacy, the system will comply with the GDPR requirements for protecting the privacy and data of the system's users, for example by only collecting data that is needed for the functioning of the website, processing it in a secure manner and not keeping it for longer than necessary.

8.4. Personal Circumstances & External Factors

This section describes a particular type of risk that the team deems likely to occur, but generally speaking not likely to affect the project too much. In particular, these risks include team members falling sick, especially taking into account that the Coronavirus pandemic is still ongoing and new restrictions could be enforced at any point in the development of the project, as well as team members dropping out of the project and more external factors such as the ongoing war in Ukraine.

Sickness is likely to happen, but such an issue is relatively easy to resolve, provided that the team sticks to the planning. If a team member has an illness, the rest of the team should realistically be able to cover that person's tasks, possibly by splitting them across the team to ensure that no one team member becomes overburdened and thus has to neglect their own tasks, creating more problems in the long run. A team member dropping out, however, is deemed unlikely, especially considering that the team members are all very close to graduating. However, if such an event occurs, the team can discuss this with the client/supervisor, as well as the responsible contact persons for the module.

External events such as new Coronavirus restrictions and the war in Ukraine are judged by the team to be unlikely to affect the development process - the team has the option to work online and has plenty of experience doing so, while the war is too distantly connected to the development to be a viable risk, according to the team's judgment.

9. Limitations & Future Work

This section covers the potential future developments for the system. The current version of the system is defined and limited to the aforementioned scope, requirements and functionalities. Due to time limitations, not all features could be implemented. Before the system is put to use some enhancements could be done.

9.1. Canvas LMS Integration

The most important future implementation should be the Personal Learning Records' integration with Canvas LMS. As it is in the current system, the team was unable to obtain the necessary token from the LISA and Canvas LMS integration teams on time, meaning that the implementation of this feature will be left for future development.

9.2. Features Improvement

Future application development will focus on enhancing existing functionality, such as expanding the course page to allow users to copy courses from prior years rather than having to manually enter all the information for a new course each time.

Another aspect that can be enhanced is the import functionality, in particular by making it possible to import from different types of files like .txt of .pdf - at the time of writing this report, importing data is only possible through uploading a .csv file. Additionally, the ability to import topics should also be available on the topics page.

9.3. Additional Features

The addition of new features, whether or not they were specified in the nice-to-have sections, is the section that will require the most future work. One such function is including a calendar. This function would be a new tab on the website that would let students view their mentors' checkpoint meeting availability. It will be much simpler to schedule checkpoint meetings this way. The calendar should list all potential days, timeslots, and meeting types (online/on-campus). Another potential feature is to integrate Microsoft Teams or Zoom with the calendar for further convenience for the users of the system.

10. Conclusion

Throughout the development of the Personal Learning Records project, a new and improved version of the system was created, implemented with the specifications that the client requested at the beginning of the project. The creation of this system and its subsequent integration into the UT's courses will allow students to realistically assess their progress throughout the module, while simultaneously giving their mentors and teachers a platform to monitor their progress and assist students whenever the need arises.

11. Acknowledgements

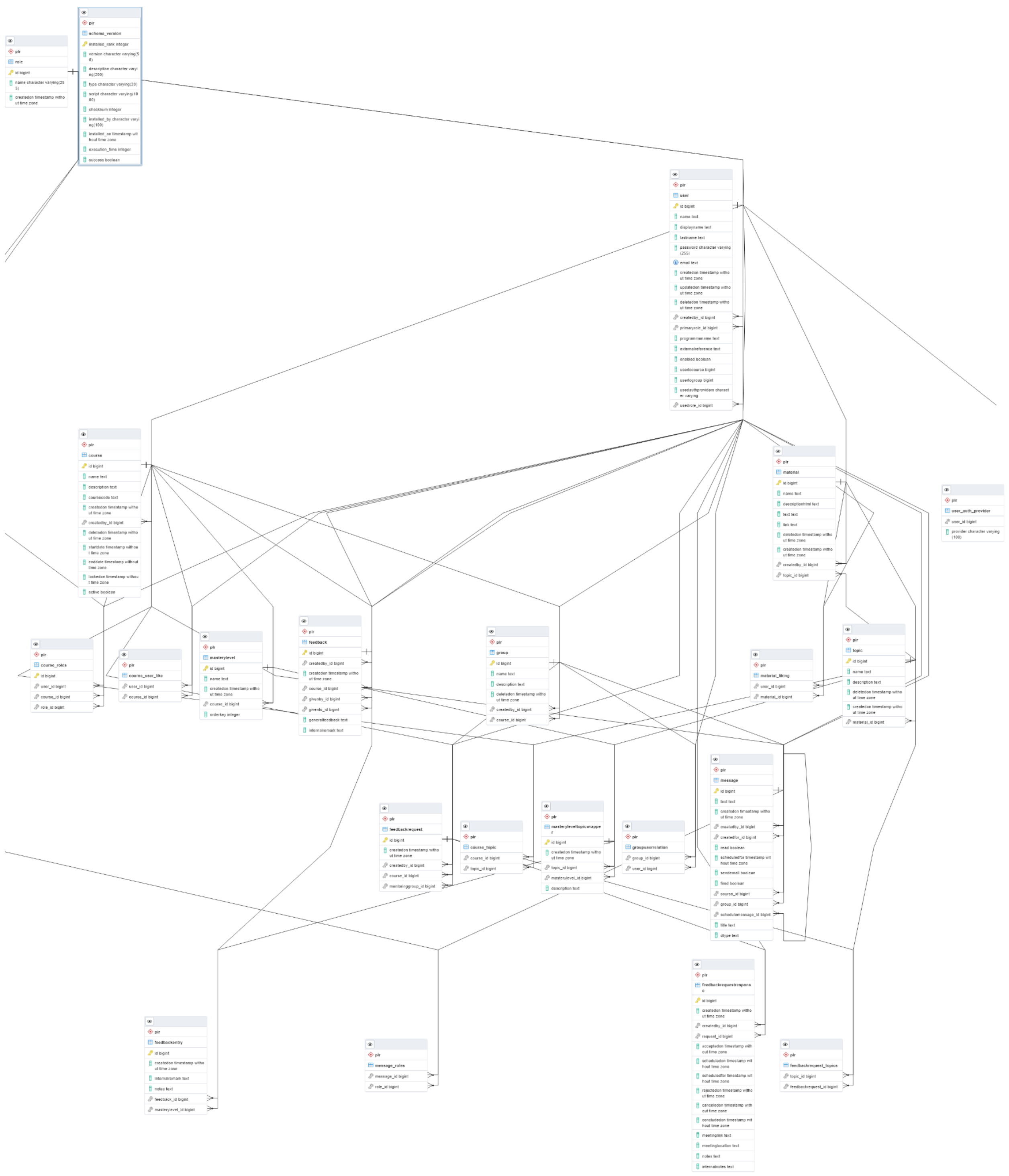
The team would like to thank our supervisor and client, Wallace Corbo Ugulino, for giving us the opportunity to work on this project and guiding us throughout it.

12. References

1. Business, A. (2022, October 18). Chapter 10: MoSCoW Prioritisation. Retrieved from https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html
2. Lauesen, S. (2002). Software Requirements - Styles and Techniques. ResearchGate. Retrieved from https://www.researchgate.net/publication/243785609_Software_Requirements-Styles_and_Techniques
3. Atlassian. (2022, October 18). Jira | Issue & Project Tracking Software | Atlassian. Retrieved from https://www.atlassian.com/software/jira
4. The Visual Collaboration Platform for Every Team | Miro. (2022, October 18). Retrieved from https://miro.com
5. Manifesto for Agile Software Development. (2020, June 09). Retrieved from http://agilemanifesto.org
6. Ideal Modeling & Diagramming Tool for Agile Team Collaboration. (2022, October 18). Retrieved from https://www.visual-paradigm.com
7. Mendelow, A. L. (1981). Environmental Scanning-The Impact of the Stakeholder Concept. Retrieved from https://www.semanticscholar.org/paper/Environmental-Scanning-The-Impact-of-the-Concept-Mendelow/7b361652157989c77ed442dd387ec9b1a9b99632?p2df
8. Free Design Tool: Presentations, Video, Social Media | Canva. (2022, November 10). Retrieved from https://www.canva.com

Appendix

Database Design


plr
role
id bigint
name character varying(255)
createdon timestamp without time zone
plr
schema_version
installed_rank integer
version character varying(50)
description character varying(200)
type character varying(20)
script character varying(1000)
checksum integer
installed_by character varying(100)
installed_on timestamp without time zone
execution_time integer
success boolean
plr
user
id bigint
name text
displayname text
lastname text
password character varying (255)
email text
createdon timestamp without time zone
updatedon timestamp without time zone
deletedon timestamp without time zone
createdby_id bigint
primaryrole_id bigint
programmename text
externalreference text
enabled boolean
usertocourse bigint
usertogroup bigint
usedauthproviders character varying
usedrole_id bigint
plr
course
id bigint
name text
description text
coursecode text
createdon timestamp without time zone
createdby_id bigint
deletedon timestamp without time zone
startdate timestamp without time zone
enddate timestamp without time zone
lockedon timestamp without time zone
active boolean
plr
material
id bigint
name text
descriptionhtml text
text text
link text
deletedon timestamp without time zone
createdon timestamp without time zone
createdby_id bigint
topic_id bigint
plr
user_auth_provider
user_id bigint
provider character varying (100)
plr
course_roles
id bigint
user_id bigint
course_id bigint
role_id bigint
plr
course_user_like
user_id bigint
course_id bigint
plr
masterylevel
id bigint
name text
createdon timestamp without time zone
course_id bigint
orderkey integer
plr
feedback
id bigint
createdby_id bigint
createdon timestamp without time zone
course_id bigint
givenby_id bigint
givento_id bigint
generalfeedback text
internalremark text
plr
group
id bigint
name text
description text
deletedon timestamp without time zone
createdby_id bigint
course_id bigint
plr
material_liking
user_id bigint
material_id bigint
plr
topic
id bigint
name text
description text
deletedon timestamp without time zone
createdon timestamp without time zone
material_id bigint
plr
message
id bigint
text text
createdon timestamp without time zone
createdby_id bigint
createdfor_id bigint
read boolean
scheduledfor timestamp without time zone
sendemail boolean
fired boolean
course_id bigint
group_id bigint
schedulemessage_id bigint
title text
dtype text
plr
feedbackrequest
id bigint
createdon timestamp without time zone
createdby_id bigint
course_id bigint
mentoringgroup_id bigint
plr
course_topic
course_id bigint
topic_id bigint
plr
masteryleveltopicwrapper
id bigint
createdon timestamp without time zone
topic_id bigint
masterylevel_id bigint
description text
plr
groupuserrelation
group_id bigint
user_id bigint
plr
feedbackrequestresponse
id bigint
createdon timestamp without time zone
createdby_id bigint
request_id bigint
acceptedon timestamp without time zone
scheduledon timestamp without time zone
scheduledfor timestamp without time zone
rejectedon timestamp without time zone
canceledon timestamp without time zone
concludedon timestamp without time zone
meetinglink text
meetinglocation text
notes text
internalnotes text
plr
feedbackentry
id bigint
createdon timestamp without time zone
internalremark text
notes text
feedback_id bigint
masterylevel_id bigint
plr
message_roles
message_id bigint
role_id bigint
plr
feedbackrequest_topics
topic_id bigint
feedbackrequest_id bigint