Design Project Report

Themis

Group 18:
Felicia Burlacu - s2872927
Cristian Comendant - s2751828
Adrian Josan - s2763559
Cristian Perlog - s2675994
Maxim Roşca - s2779307

Faculty of Electrical Engineering,
Mathematics and Computer Science(EEMCS)

Supervised by:
Nacir Bouali
David Huistra

April 26, 2024

Abstract

Abstract

The University of Twente has a Project-based Learning (PBL) approach when it comes to learning which means that all people involved in the process of teaching are faced with the task of grading project submissions. Currently, the process of peer-grading submissions involves the use of multiple platforms and becomes hard to manage when dealing with a large number of project teams within one course. In order to address these issues, it was proposed to design a platform that would house the entire process of grading from importing courses and assignments from Canvas, creating and editing rubrics, assigning graders, grading submissions to exporting grades and the given feedback. Additionally, it would include useful admin functionality like handling grader assignment with the option of automated assignment. This report will touch on all aspects of the design and development process.

Contents

1 Introduction

1.1 Background

The main form of instruction at the University of Twente is Project-based Learning (PBL) which focuses on students designing, developing, and constructing hands-on solutions to a problem. The educational value of PBL is that it aims to build students' creative capacity to work through difficult or ill-structured problems, commonly in small teams. Typically, PBL takes students through the following phases or steps:

- Identifying a problem
- Agreeing on or devising a solution and potential solution path to the problem (i.e., how to achieve the solution)
- Designing and developing a prototype of the solution
- Refining the solution based on feedback from experts, instructors, and/or peers.[1]

As a result, teachers and teaching assistants are tasked with the grading of a large number of submissions of projects during each module.

This report will touch on the issues of the current solution (Section 1.3) as well as on the development of our proposed solution (Section 1.4), outlining the key phases of the project's lifecycle. It starts with risk evaluation (Section 3) and follows with stakeholder definition and requirement elicitation (Section 2). Section 4 and 5 delve into the planning and prototyping phases when responsibilities are divided and the iterative testing of initial concepts began. The main body of the report contains in depth description of the design architecture of the front-end and back-end in Section **??** and Section 6. Lastly, Section 10 expands on how the platform can be enhances with more functionality and gives a brief reflection of our team over the whole design project.

1.2 Definitions

- Grading rubric - *a set of criteria and guidelines that describe how a submission should be assessed.*
- Grader - *person who assesses the submissions of students (individual or in groups) based on the grading rubric*
- Coordinator/admin - *person responsible for the coordination and organization of a module*
- Teaching assistant - *student hired by the university to aid students in the learning process and usually also tasked to help in the process of grading submissions*
- Submission - *an instance of a particular assignment; sometimes can have multiple types*
- Criteria Value - *the performance level or grade that is given to a criterion*
- Performance Level - *category or level of achievement used to assess a task or activity. Used to describe the varying degrees of quality in a criterion*
- Criterion - *a standard or principle by which a submission is assessed. Usually part of a section. Example: Quality*
- Section - *a bigger part of the rubric that contains multiple criteria, for example, Presentation*

[1]Project-Based Learning: Center for Teaching and Learning. Project-Based Learning | Center for Teaching and Learning. Boston University Website.

1.3 Current Solution

Currently the graders at the University of Twente are doing the grading of project manually by downloading the submissions from Canvas and managing them from a Google Drive. The grading rubrics are in the form of Excel sheets and are one per assignment, so there is one rubric that holds all the rubric criteria for multiple submission types. There are multiple issues with the current approach.

Issue 1: Large number of project submissions As the main approach at the University of Twente is Project - Based learning, the process of grading project submissions takes a big part of the assessment of students abilities. At the moment the graders use multiple tools during the grading process:

- Google Drive to store all the submission for grading in one place
- Mail, excel sheets or other communication platforms to divide tasks
- Excel sheets for managing each rubric
- PDF viewer
- Canvas for extracting all the submission for grading and giving the feedback and grades

It takes approximately two weeks for the coordinators to set up all the tools for the grading in the Data Science course according to our supervisor & stakeholder Nacir Bouali. This accentuates the inefficiency of the process. Additionally, having to use this many tools hinders the grading process and makes it less efficient, as graders have to constantly switch between screens (because there is no way to conveniently display the entire rubric and submission on one screen, in most cases). This is inconvenient for all parties involved: coordinators - who have to deal with the potential delay, graders - who may be get annoyed and stressed and students who may not get their grades in time.

Issue 2: Grader Assignment
At the moment grader assignment is only done manually by the module coordinators. The current issue at hand regards grading preferences and grader background. As there is no list where the graders would fill out their preferences to grade specific types of assignments and what are their educational background - to make it clear for what assignments their suitable the most - the coordinators do not take this into account when assigning them, and some discrepancies may arise, for example when the background of a teacher is very different from the content of the assignment they are supposed to assess. An additional issue is when new teachers join the course as graders and coordinators are not sure what is available for them in order to start grading submissions as well.

Issue 3: Student Feedback
An important part of the learning process is feedback, especially when dealing with applying your knowledge in practice - during project work, this is why proving students with plenty of comments and making the grading as transparent as possible is crucial. Currently graders fill in a Google form which is then exported into a Excel sheet. Based on the sheets a person is assigned to manually transfer the contents to Canvas. One can already imagine that this process is very tedious and can be made more efficient.

1.4 Proposed Solution

Based on the discussed issues, it was clear that the teaching entity at the University of Twente is in need of a platform that will be capable to accommodate peer grading in mass as current learning management systems like Canvas are not suitable for this. Additionally it should combine the benefits of Google Drive for storing the submissions of students, Excel for representation of grading criteria while also providing functionality for a smoother process of grader assignment and automated export of student feedback. In the following sections, we will describe in detail the requirements we gathered for our system (Section 2.3) as well as a detailed insights into the design (Section 7)

2 Stakeholders and Requirements Specification

2.1 Stakeholders

The process of grading student projects involves multiple parties at the university. In order to optimize the process to the fullest, it was important to identify all the groups of stakeholders and their relationship to our system.

- **Module Coordinators**
  They are responsible for the organization of the modules and their operation including assigning graders to submissions. They also overview the grading process, making sure grading deadlines are met.
- **Teachers**
  They usually create projects and assignments within a course, create grading rubrics. They are also tasked with grading submissions and give feedback for students.
- **Teaching Assistants**
  Usually they are former students of the modules for which they are working as teaching assistants, and are responsible for aiding the learning process of current students. They are also tasked to grade submissions according to the existing rubric and give feedback.
- **External Graders**
  Industry professionals or project stakeholders, responsible for assessing the degree to which a project fulfills its specified requirements.
- **Students**
  Although, they are not direct users of the platform, they will be affected by it as their submissions will be graded on the platform and will receive benefits from it like - potential reduction in grading delays and increase in amount of feedback
- **Development Team**
- **Library, IT Services & Archive (LISA)**
  They are the responsible organization for management and integration of student developed platforms like our own into the UT infrastructure.

Defining our stakeholders was a crucial step in the development process, as their active participation significantly influenced the project's direction. Nacir and David representing the teachers and coordinators, along with David who also represented LISA, played key roles. Their insights and feedback ensured that the development was aligned with the actual needs and expectations of both the academic and administrative sides of the institution.

2.2 Functional Requirements

1. **Authentication**
   (a) Role-based access control, with specific roles for teachers, coordinators, and possible teaching assistants in the future
   (b) Login based on UT credentials
2. **Course and Submission Management**
   (a) Functionality for uploading submissions
   (b) The user uploads the course by choosing it from a list of courses available to them and are not the platform
   (c) The user will be able to choose what roles they want to import from Canvas (f.e. Just teachers; teachers and TAs)

(d) Users will be able to filter the submissions by different criteria (e.g., by academic year, graded/not graded)

(e) Users will be able to customize their landing page (e.g., hide modules based on academic year)

(f) The coordinator will be able to add new graders to the platform by resyncing

3. **Grader Assignment**

   (a) Each submission will be assigned with two graders

   (b) Functionality for automatic grader assignment according to grader preferences per topic

   (c) Functionality to override a previous assignment manually

   (d) Coordinators will fill in the grading preferences for teachers to ensure assignment according to preference/expertise

   (e) The grader assignment can be changed after the grading process has started

4. **Grading Process**

   (a) Users, if given the permission, will be able to customize submission rubric with the ability to define, edit and add rubrics with descriptions on the platform

   (b) Automatic matching of submissions to the appropriate rubric based on project type

   (c) Users will not be able to see each others grades

   (d) Users will be able to see the names of the submission owners and who are the other graders

   (e) Grades can be overruled, or automatically determined (depending on the type of assignment)

   (f) The rubric and the submission will be displayed on the same screen

   (g) Each rubric point will have attached weights

5. **Statistics and Feedback**

   (a) Users will be able to see grading statistics per submission

6. **Grade Finalization and Export**

   (a) The platform will be able to export grades back to Canvas, including the rubric outcome as comments

   (b) The system will include functionality to handle discrepancies in grading

   (c) Option for coordinators to send automatic reminders for grading deadlines

   (d) Coordinators will be able to send reminders for specific submissions that need urgent attention

2.3 Non-Functional Requirements

1. **Authentication**

   (a) A role-based authentication should be provided

   (b) The identity provider should be easy to change

   (c) The login type should be easy to change

2. **Maintainability**

   (a) The app should be developed using Spring specific layer architecture to ensure easy adaptability for Spring developers

   (b) The app should be well documented

   (c) The app should be modular, with clear separation of concerns

   (d) Adding future endpoints should be intuitive and easy

(e) Securing future endpoints should be easy to do
(f) It should be easy to add or remove dependencies
(g) Best coding practices should be applied where appropriate
(h) Code should be reusable when appropriate

3. **Testing**
   (a) The app should be well tested, with 80% coverage for most classes
   (b) Testing using different types of user, environments and data should be easy to do
   (c) Automatic test run and code build on pushing to repository should be done
4. **Durability**
   (a) All the input should be validated
   (b) It should not be possible to put the database in inconsistent state without manually editing the database
5. **Deployment**
   (a) The app should be easy to deploy
   (b) The app should have all the tests ran automatically before being deployed
6. **Scalability**
   (a) The app should be vertically and horizontally scalable
7. **Reliability**
   (a) The system should be available during the grading periods
   (b) All the errors in the code should be gracefully handled
8. **Usability**
   (a) The user interface and system should be user-friendly and intuitive
9. **Integration**
   (a) The app should be integrated with Canvas
   (b) The app should provide well-defined API endpoints for integration with other systems
10. **Performance**
    (a) The app should have good performance for usual tasks
    (b) The app should be able to handle multiple users grading at the same time

2.4 User Stories (MoSCoW Prioritization)

Must

1. **Authentication**
   (a) As a user, I want to authenticate with my UT credentials so that I can securely access the grading platform
2. **Course Management**
   (a) As a coordinator, I want to import courses from Canvas
   (b) As a user I want to see all the imported modules
3. **Assignment Management**

   (a) As a coordinator I want to import assignments from Canvas
   (b) As a user, I want to see all the imported assignments

4. **Submission Management**
   (a) As a coordinator, I want to access all the submissions for a certain assignment
   (b) As a coordinator, I want to set the graders and referee for a submission
   (c) As a coordinator, I want to set the type of the submission
   (d) As a grader, I want to fill the rubric of a submission in order to grade it

5. **Types**
   (a) As a coordinator I want to see all the created types in a module
   (b) As a coordinator I want to add a type
   (c) As a coordinator I want to delete a type

6. **Rubrics**
   (a) As a coordinator I want to see all the created rubrics in a module
   (b) As a coordinator I want to add a rubric
   (c) As a coordinator I want to delete a rubric

7. **Users**
   (a) As a coordinator I want to see all the users from a module
   (b) As a coordinator I want to change a user's role

8. **Grader Assignment**
   (a) As a coordinator, I want to be able to fast assign graders based on their types
   (b) As a coordinator, I want to see an overview of who grades which submission and edit this overview

9. **Export grades**
   (a) As a coordinator, I want to be able to export the grades and feedback back to Canvas

Should

1. **Course Management**
   (a) As a user, I want to be able to see only courses from a certain year

2. **Users**
   (a) As a coordinator, I want to be able to set and change grader's types
   (b) As a coordinator, I want to be able to sync the system with Canvas so that I get the updated list of users

Could

1. **Submission Management**
   (a) As a coordinator, I want to remind the graders about an ungraded submission
2. **Rubrics**
   (a) As a coordinator, I want to copy a rubric to another module
3. **Reminders**
   (a) As a coordinator, I want to set automatic reminders on or off
   (b) As a coordinator, I want to set how far from the deadline the reminder is sent
   (c) As a coordinator, I want to edit the reminder message

Won't

1. **Submission Management**
   (a) As a grader, I do not want to see submissions that I am not grading

3 Risk Analysis

Requirement changes (Likelihood - medium, Impact - medium). In case a requirement is changed it will be extensively discussed with the stakeholders to understand its importance, urgency, and complexity. We will discuss the feasibility of the change and estimate how the change will affect our progress. After that, we will either start working on it or leave the change for the future.

Member absence or inability to work (Likelihood - medium, Impact - low). There are always personal circumstances that we can't see from the beginning and some of the members can't always do their work. To avoid this we have two persons who are responsible for a certain part, meaning that in case one of them is unavailable the other will be able to quickly understand the situation and do the job for him.

Resistance to change (Likelihood - low, Impact - low). Despite our effort to create a user friendly platform that is easy to use and understandable, we cannot deny the possibility of users' reluctance to adopt the platform. Although in most cases, the approval from top management is enough to eliminate the probability of this scenario actually happening, we tried to give the user the freedom to customize their experience (the grading process) and give them the possibility to make it similar to what it looked like before by allowing the user to download the submissions and view them from a separate PDF viewer. Therefore by the option of emulating the previous system that was in place we would avoid the potential disruption that the new platform might cause.

Security vulnerability discovered later (Likelihood - medium, Impact - high). Despite rigorous testing during development and implementation of security measures, a security vulnerability might still go undetected. In case it is discovered later, it has the potential to create high impact like exposure to malicious actors and reputation damage, but also the costs of remediation can be quite high and the damage to the users can be quite significant.

4 Planning

4.1 Project Timeline

Week 1 • Team formation • Initial Meeting • Technology stack selection	**Week 5 - 7** • Development • Progress & Feedback supervisor meeting
Week 2 • Development • Progress & Feedback supervisor meeting • Peer review meeting • Testing • Final report	**Week 8 - 9** • Development • Progress meeting
Week 3 • System design & approval • Backend project structure creation • Setup of CI/CD pipeline for backend (build + test run) • Front-end project structure creation • UX design	**Week 10-11** • Development • Progress & Feedback supervisor meeting • Peer review meeting • Testing • Final report
Week 4 • UX design • Start front-end development • Create database • Peer review meeting • Start backend development	**Week 12** • Presentation

4.2 Team Roles and Responsibilities

Responsibilities: A detailed explanation of each role is presented below.

Front-End Developer

- Build user interfaces based on UI/UX designs.
- Write clean, reusable and efficient code using Vue.js.
- Integrate APIs in the front-end components.

Name of the Team Member	Role	Working Hours	Method of communication
Felicia Burlacu	UI/UX, database developer	8 a.m - 17 p.m	Discord
Cristian Comendant	Frontend, DevOps developer	8 a.m - 17 p.m	Discord
Adrian Josan	Backend and Frontend developer	8 a.m - 17 p.m	Discord
Cristian Perlog	Frontend, Database developer	8 a.m - 17 p.m	Discord
Maxim Roşca	Backend, DevOps developer	8 a.m - 17 p.m	Discord

Table 1: Team overview

- Check and clean user inputs.
- Write e2e and unit tests.

Back-End Developer:

- Build server-side logic and infrastructure.
- Develop and secure endpoints.
- Ensure data validation
- Ensure correct implementation of REST principles
- Ensure correct data provision to the front-end
- Implement data security and authentication system.

Database Developer

- Design and implement the database architecture.
- Optimize and tune database queries for performance.
- Ensure data integrity and security.
- Collaborate with back-end developers to design efficient data models.

DevOps Developer

- Implement and maintain continuous integration/continuous deployment (CI/CD) pipelines.
- Automate deployment and infrastructure provisioning.
- Monitor and optimize system performance and reliability.
- Collaborate with development and operations teams for seamless delivery.

UI/UX Designer

- Create blueprints and prototypes for the user interface.
- Design user-friendly interfaces.
- Collaborate with front-end developers to implement the design.

4.3 Collaborative Responsibilities

Communication:

- Encourage effective communication and collaboration within the team.
- Participate in regular team meetings: a 15-minute standup or discord messages to ensure progress

Integration Testing:

- Collaborate on integration testing to ensure communication between front-end, back-end, and database components.
- Make sure each part works separately before trying integration testing, i.e write unit tests

Code Reviews:

- Do code reviews for each merge and try to understand where improvements can be done

4.4 Procedures

A detailed explanation of each procedure is presented below.

Working with Gitlab and issues. For creating and monitoring time, as well as creating and picking issues, we use the GitLab issue board. For each newly picked issue, add an estimated time to complete the task and create a new branch. The name of the branch should be the issue number followed by the issue name. After the issue is solved create a merge request to the main branch.

Solving an issue. While solving issues we will make sure to follow these simple guidelines and procedures.

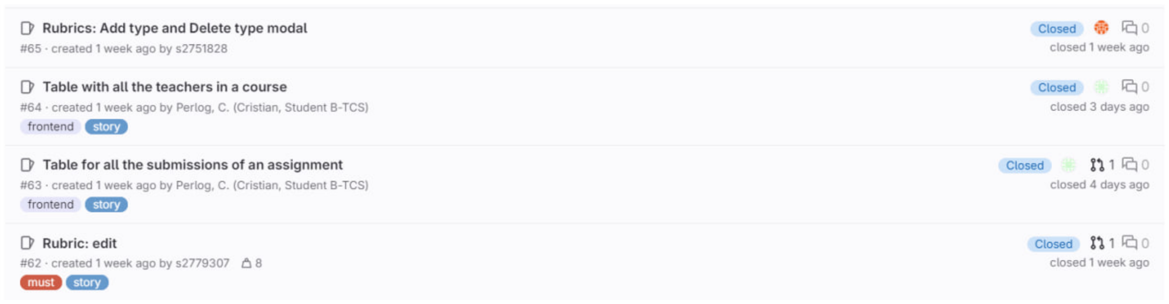
- Communication - After picking the issue from Gitlab, make sure to talk to the other members who might be involved in the issue. When working with members of other teams make sure to follow a certain standard of communication and keep it consistent (for example requests from frontend to backend).
- Documentation - Document each part of the code (Swagger for example for the backend). Make sure that the most complex parts have comments as well.
- Consistency - Keep everything consistent, make sure that it uses the same patterns as the other solutions in the project do.
- Simplicity - Keep it simple, stupid. Do not work too much on simple tasks and overthink future possibilities.
- Modularity - Make sure everything is replaceable, for example using interfaces
- Testing - Write unit tests, component tests, and integration tests for that issue. Make sure that all tests pass before you push the changes. Most classes must have a percentage of at least 80% tested. Make sure to ask for other opinions about the implemented feature.

Create an issue. A new issue is created on the board when a user requirement needs to be solved or a bug is found in the system. For each card, indicate how urgent it is to be solved using one of the tags: high, medium, or low. Provide detailed explanation with references to the UI and maybe other issues. Include data validation. Write down acceptance criteria.

Getting feedback from the clients. We get feedback from the clients during the meetings. They test the platform and provide feedback and suggestions. Also, we use Discord to message the progress and write short questions.

4.5 Project Management Tools

GitLab Issues



Rubrics: Add type and Delete type modal
#65 · created 1 week ago by s2751828
Closed 0
closed 1 week ago

Table with all the teachers in a course
#64 · created 1 week ago by Perlog, C. (Cristian, Student B-TCS)
frontend story
Closed 0
closed 3 days ago

Table for all the submissions of an assignment
#63 · created 1 week ago by Perlog, C. (Cristian, Student B-TCS)
frontend story
Closed 1 0
closed 4 days ago

Rubric: edit
#62 · created 1 week ago by s2779307 8
must story
Closed 1 0
closed 1 week ago

Figure 1: GitLab Issues

Once the requirements are established and the design process is done, a big number of tasks have to be managed and distributed among the team members in order to keep the development process organized and streamlined. For this, we chose to use GitLab issues, as it is already integrated in GitLab and allows for issue tracking, code management and review processes all from one process.

Discord Servers

Communication is one of the most important aspects of a projects, therefore establishing strong communication practices and having a reliable and comfortable communication channel is vital. For this purpose, our team decided to use Discord as a main channel for communication. There are several reasons for that: it was already familiar to us as the university uses it as a communication channel for several modules within our bachelor program; it allows to create communication channels which makes the communication organized keeping every conversation topic separate; Discord supports rich media integration which allows us to easily share images, videos, links and files directly in the chat; it supports video calls where we can arrange group sessions for collaboration. Besides the server created for our own development team, it was decided to create one more were our supervisors were also added. The purpose of the server was for having a less formal communication channel for quick feedback sessions and addressing issues that can be done outside our weekly progress & feedback meetings.

Microsoft Teams

It was decided to conduct our weekly progress & feedback meetings with the supervisors online for convenience. For this purpose, we have decided to use Microsoft Teams as a communication and collaboration platform as it provides a wide range of benefits such as: it is deeply integrated within Microsoft 365 which allows to collaborate on documents and manage your calendar directly within the platform which we though would be convenient for the supervisors; it provides a large variety of communication tools including chat and voice calls supporting recording and transcripts which was very useful as we though it was useful to record some of the meeting in order to be able to go back and revise some of the details discussed.

4.6 Methodologies

In order to deliver a product that will satisfy the client's needs while also taking part in project that enables team collaboration we have decided to use several design methodologies and extract features and principles that align with our views in order to create our own hybrid approach.

- **Agile Pillars**
  The agile methodology values customer interaction and sets a close relation between the development team and the customer from start to finish, updating them frequently and regularly - in our case during the weekly progress & feedback meetings. This allows for the development process to as transparent as possible and ensures customer satisfaction

- **Waterfall Principles**
  As this project is still part of our university schedule it was also crucial for us to keep close track of our progress and make sure we are meeting the deadline and completing the project within the given time frame of 10 weeks. This is where the waterfall principles come into place, as this is a sequential process with clear divided phases. Having distinct phases like : requirements, design, implementation, testing, deployment, maintenance, also allowed us to keep extensive documentation regarding project requirements and planning which was also useful for creating documentation documents like Project Proposal and this Design Project Report.

- **Maintainability through Attribute-driven design**
  Attribute-driven design (ADD) was crucial in the development process of our platform, given the importance of maintainability. As it is intended that once finished, the project will be transferred and maintained by a different team of developers it was important to adhere to exemplary coding practices which enables future updates and modifications and ensures that the development team can easily understand and manage the codebase. By selecting technologies and libraries that are widely used, we tried to minimize the learning curve for the new team. Additionally, we focused on a modular architecture under the ADD framework which allowed for manageable components that can be independently developed, tested and updated, therefore supporting maintainability and scalability.

5 Prototyping

During our initial meetings with the supervisors, we have focused on developing a preliminary rough mock up of our platform using Figma. This early prototyping was essential not only for making sure we understood the assignment, but also for an effective requirement elicitation. A visual mock up allowed us to visually explore and represent the functionalities included in each of the pages of our platform and construct its layout. We have also uncovered additional requirements and discussed critical areas of the project (like the rubrics-types concept) that needed more attention during development. This approach has set a solid foundation for a detailed development ensuring that the final product will meet the needs of our stakeholders more accurately.

A more detailed view of the mock up can be found in the Appendix A.1. One might observe that the pages are not role specific, but rather designed from the admin perspective. This is because the pages that would require two views (Main, Assignments, Submissions, Rubrics Pages) are very similar for the two roles, and usually the admin would have more functionality incorporated on their end, therefore it was more intuitive to first get a clear view of the platform from the admin perspective.

After the initial meetings and requirements elicitation, we have moved onto UX design which was used as a base for the final version of how the front end of our platform looks like. The final version of the design is quite different from the mock-up and that is because it was refined through a larger number of meetings and because it was started after the process of requirements elicitation was concluded. A more detailed view can be found in the Appendix A.2.

6 Global Design

6.1 Design Overview

The app needs to be accessed and used by multiple users simultaneously, the resources must be accessed remotely from any device and no installation is required. Because of this, we decided to build a web app. The app must be reusable, scalable, with good separation of concerns, and easy to work on to achieve this, we divided the web app into 3 main components: front-end, back-end, and database. The front end is responsible for the user interface and client-side logic, the backend is responsible for the business and server-side logic and the database is responsible for storing all the data. The app was designed for maintainability and a lot of decision that were made are to provide a better future development of the app.

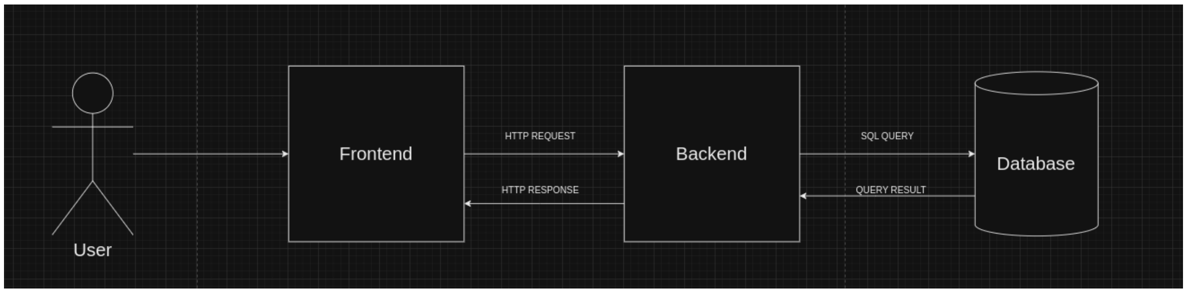


Figure 2: General Overview

6.1.1 Front End

General Description. Our front end application is structured as a Single Page Application (SPA) utilizing the Vue.js framework. The choice of the framework was primarily based on a recommendation from one of our clients/supervisors. In addition, some of us had limited experience with Vue.js from past internships. Subsequently, since we chose Vue.js, we also went for the SPA architecture, because it is the recommended way of developing Vue.js applications. SPAs ensure a fluid interface by dynamically updating content (without the need for page reloads) as users interact with the application, thus minimizing interruptions.

Compared to alternative frameworks, Vue.js offers an easier learning curve, which allowed us to quickly grasp its concepts. It uses a component-based architecture. A Vue component is a reusable unit of the UI that has its own structure, functionality and potentially state management. This is a modular approach to building the platform from separate and potentially reusable blocks.

In our application, we have used Bootstrap as the main tool for front end styling. Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. Using Bootstrap simplifies the implementation of responsive and visually appealing designs, while maintaining consistency across different viewport sizes.

To better integrate Bootstrap components with Vue.js, we utilize Bootstrap Vue Next, a library specifically designed for this purpose. This facilitates the integration of Bootstrap's UI components into Vue.js applications, enabling efficient communication between the design layer and underlying Vue components.

Furthermore, our application communicates with back end services through API endpoints, utilizing Axios as the HTTP client for making asynchronous requests. Under the hood, Axios uses AJAX calls to make asynchronous requests to endpoints. Using AJAX techniques is essential in an SPA, like Themis, since AJAX allows the application to fetch data from the server without the need for full page reloads. This means that it is possible to update parts of a component, without reloading the entire view and its children components.

General Structure. In the structural hierarchy of our Single Page Application, the main entry point is the **App.vue** component. Here, the Vue Router is mounted, serving as the navigational backbone of the application. The Vue Router coordinates the loading of distinct views based on the URL mapping, ensuring that users are directed to the appropriate views as they navigate through the application.

These views correspond to various pages within the application, such as the rubrics page, assignments page, and course page, among others. Each of these views encompasses multiple components, each responsible for encapsulating specific units of functionality or elements of the user interface. This modular approach allows for the efficient organization of the application's features and UI elements.

To execute its functionality, each component often relies on shared state information retrieved from dedicated stores. These stores serve as centralized repositories for maintaining application state, such as data related to rubrics, courses, and other pertinent information. By utilizing stores, multiple components can access and manipulate the same state, ensuring consistency across the application.

When components need to interact with the back end services to fetch or update data, they do so through dedicated stores. These stores employ the Axios library to send HTTP requests to the appropriate API endpoints. Axios serves as the intermediary for communicating with the back end, managing the required metadata for each request, such as headers, parameters, and request bodies.

Upon receiving a response from the back end, the store processes and saves the data, making it accessible to the relevant components within the application.

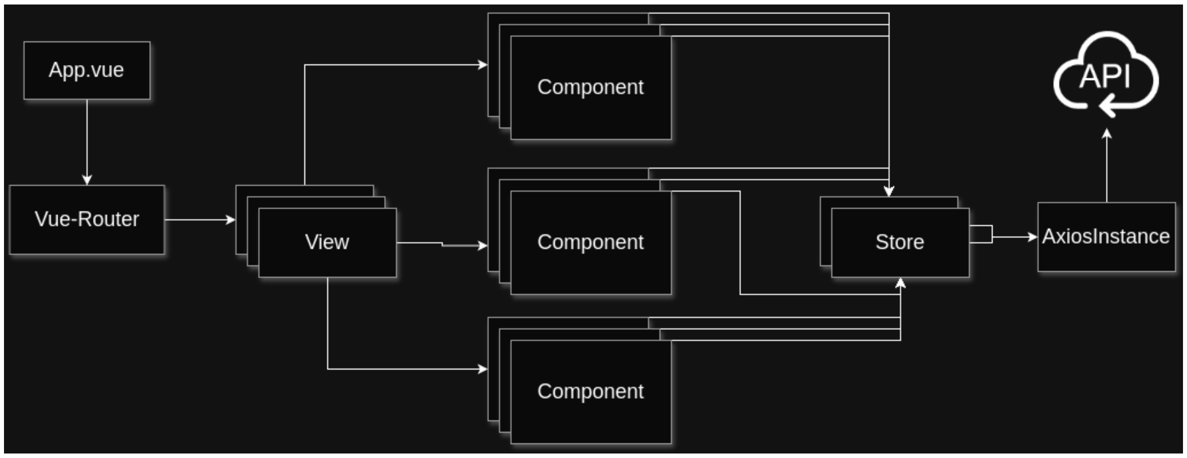


Figure 3: Front End SPA Architecture

6.1.2 Back End

General description. Our backend is based on the REST architectural style [1]. The main benefits of it are scalability, independence, and modularity. The back end can be used independently of the front end and integrated with other apps. Respecting the Fielding's constraints [1] allows to scale the REST service and, for example, have two instances running at the same time with a load balancer [2] in front to handle more requests at the same time and facilitate horizontal scaling.

The two big components of the backend are the backend itself, which contains the business and server-side logic and the database 2.

The client side must be separated from the business logic, one of the main reasons is that the client's actions must be limited and checked before editing any kind of data. For the backend - the chosen programming language is Java 17, which is high-level, class-based, and object-oriented, being a good fit for our project. Creating a backend server is no easy task and a framework is required for that, our choice being Spring. Spring is a modular and lightweight web development framework that offers integration with a lot of popular frameworks and technologies, offers a bunch of tools, and drastically reduces the boilerplate code, allowing us to focus mainly on business logic.

All the data about the user and platform needs to be stored, that's why a database is needed. For the database - SQL database was chosen, as our system has structured data and based on our experience is a better choice than NO-SQL databases. The specific database that we chose is PostgreSQL, mainly because our team already had experience with it.

General structure. Following Spring best practices and coding principles, to make the development process as easy and clear as possible the backend was further divided into multiple layers 4. This is necessary to better follow SOLID principles, mainly the separation of concerns. The main components are controllers, validators, mappers, services, and repositories.
Controllers - are the entry point of the application, they receive the HTTP request and decide which business logic will be executed and if it will be executed at all.
Services - responsible for the business logic and are called by the controllers. They can also communicate with other services.
Validators - decide if the request is a valid one and won't break anything or have unwanted consequences. They are used in the controller and services.
Mappers - they are responsible for transforming one Java object into another. This is required to give the user only the required information
Repositories - responsible for communication with the database.

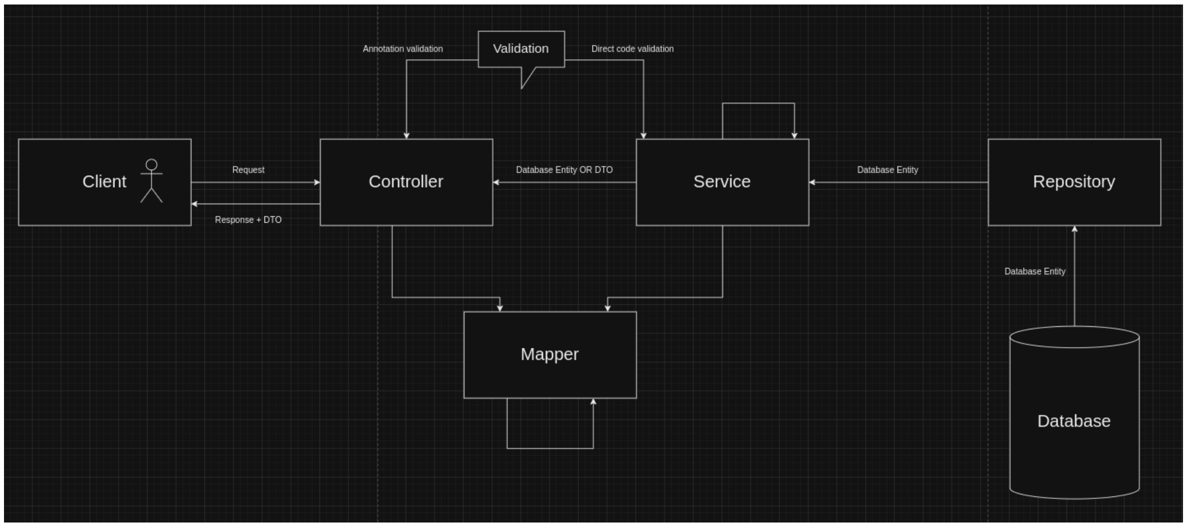


Figure 4: Backend Structure

Backed-database communication To retrieve data and provide it to the client the backend needs a means of communication with the database. SQL is the main language used for that, but most of the queries are standard, and writing them manually would take some time and will result in more code. To solve this problem Java Persistence API is used, one of its implementations being Hibernate, which is used by default by Spring. Hibernate allows us to use annotations to map a Java object to a database entity. Using Spring JPARepository we can define the SQL Queries easily, it offers out-of-the-box all the CRUD operations, but we can define our own by using special keywords when defining methods, or by defining our queries. It is important to track changes done in the database, to achieve that a tool named flyway is used. It allows us to specify these changes with the so-called migrations, which are files containing database structure changes.

Security Security is one of the most important aspects of our app. The users must be able to log in using the university credentials. The university uses SAML protocol for that and that's what we used as well. Because the university identity provider can't be used, we simulated our own using Auth0, a platform that allows you to create these types of servers. For the development environment, Basic Authentication is used with the same password for every user, such that tests can be done using different users. Users have two possible roles Grader, who has access to only grading the submissions assigned to him, and Coordinator, who has access to everything the platform has to offer, both roles are per course.

External systems There are 3 main systems with which the backend communicates, excluding the database front-end, canvas, and the identity provider.

Communication with the front end is done through HTTP requests to the endpoints specified in the backend. Most of the endpoints are done to match the requirements specified by the front end.

To get and add information to Canvas the backend does the same HTTP requests. It does get requests to retrieve information, such as course information or user information, and post requests to add information, for example, grades and comments to a submission.

The third external system with which the backend communicates is the identity provider, which is required for authentication. The information about it is specified using metadata and Spring Security is used to do all the requests automatically.

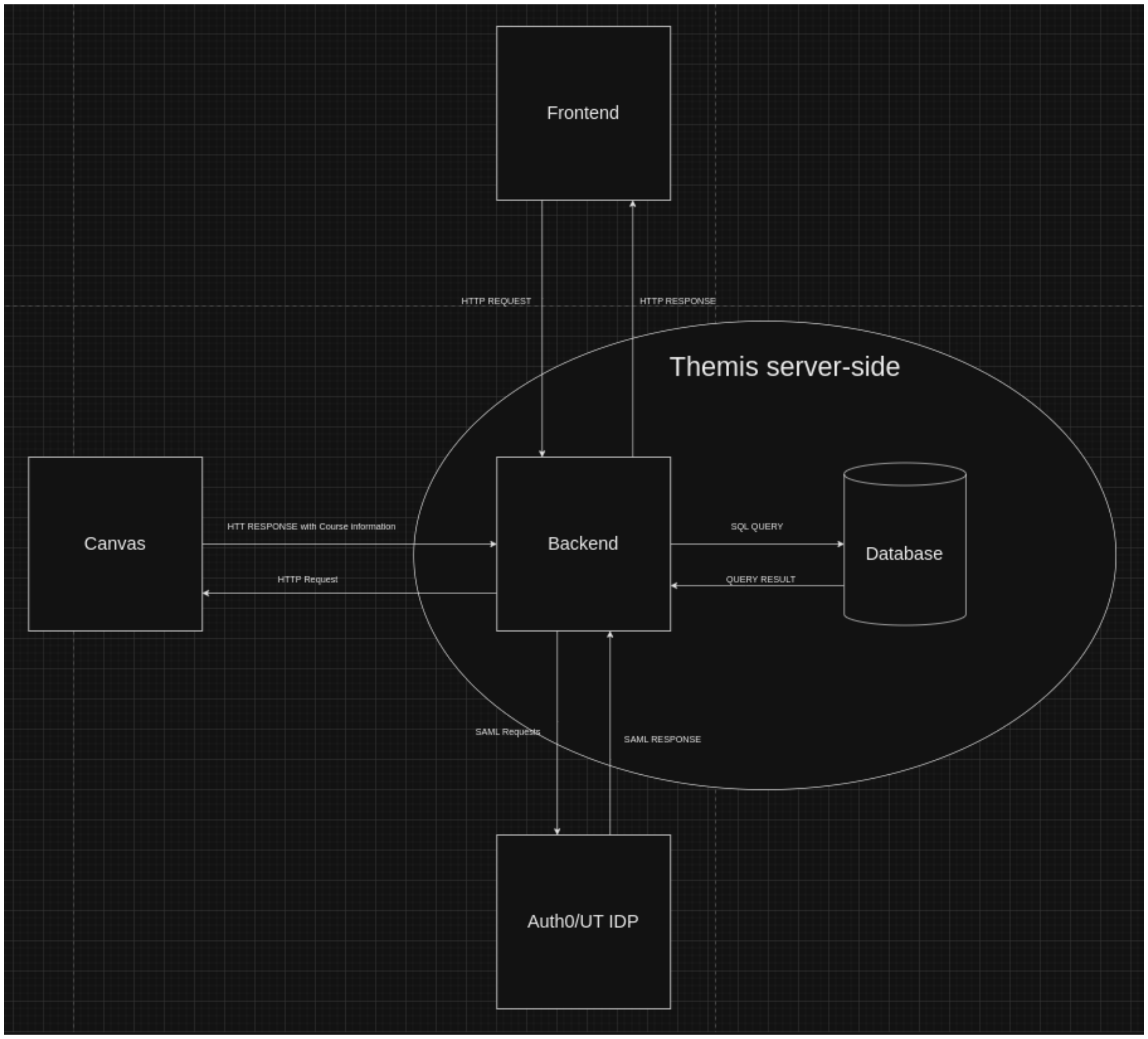


Figure 5: External systems

Shipping our app Setting up the right environment, with all the right dependencies most of the time is not the easiest task. To overcome this problem we use multiple tools. Gradle is used to manage Java dependencies. To set up the actual environment in which the backend and the database would run Docker is used. It allows us to set up the whole environment and install all the required software by just running a command. The app version that is present on the main branch of our GitLab repository must be always stable, to ensure that we have a CI/CD pipeline set up. The pipeline currently builds the app and runs the tests. We plan to add a step that would build the docker image and push it to the docker repository.

Environments To facilitate an easier developing experience we use multiple environments, mainly production, development, and testing. The main difference between them is the use of Basic Authentication instead of SAML in the development and testing environment. Also, the testing environment will run the tests. In the development environment, all users from the database can be impersonated by using the same password for all of them.

7 Lower-level Design

In the following section, a more detailed description of the front-end, back-end, and overall design will be given. It will include interesting issues we had and how we solved them. Also, it will include all the non-standard decisions that were made.

7.1 Back-end

We solved multiple issues and have made multiple decisions to improve the design of the system. Below you'll find a list of them with detailed descriptions.

Java boilerplate code Some classes, such as data transfer objects used to send the data to the client always require getters, setters, and a constructor without arguments. Also, other classes require constructors that would set the dependencies in the class. As a result, the classes become crowded with unnecessary code. To solve this problem we used a library called Lombok. Lombok uses annotation to specify what you need, for example, getters and setters for all the fields, a constructor with no arguments, or a constructor for all the final dependencies.

Code reusability Spring makes heavy use of annotations, especially for validation. Such annotations as @NotNull are always used on top of data transfer objects to make sure the user sends fields that are not null. There are cases in which such annotations are not defined by default and you have two options, to either do the validation in the code that uses that data transfer object or create your annotation. We went with the second approach, mainly because it helps with code reusability, and by just putting the annotation on top of the required field you can pass it to any function. Such example of annotation is @Exists, which would check if the ID provided by the user exists in a provided database table.

For loops In old Java code you'll always see for loops, which at times become quite big and complicated. In our code, you won't see a single for loop, this is because we use Java Stream API. It allows us to use functional programming elements, such as filter, map, and flatMap to work with lists. A 10 lines of code for loop becomes a single line of code with streams.

Returning error codes Returning the right response to the user, especially when something goes wrong might not be the easiest task. Fortunately, Spring has a clever way to deal with it. When something goes wrong, we throw custom-made exceptions. These exceptions are then intercepted by the so-called Controller Advice, which is responsible for handling the errors and returning the appropriate error code and message based on each exception. We created special data transfer objects for errors, in case there are multiple ones, such as the errors from validating an object.

Canvas communication Our project requires communication with Canvas. There is an already-defined library for that, but unfortunately, some of the functionality is missing from it. Some of the endpoints that we needed to call were not present and we needed to do custom API calls with the required parameters and deserialize the response. Another problem that we faced is that the classes from the library that represent the Canvas API responses do not contain all the fields. To solve this we created a new class that inherited the library object and added the required fields.

Security Setting up multiple environments for development is trickier than it seems. Once the SAML login was set, most of the classes where the current user was present required a SAML-specific class. To solve this we created the PrincipalService class, the class responsible for the current user, and in there we created a get method that would convert the development environment user to the SAML-specific user.
Another issue that we faced with security is the fact that we simulated the UT Login server (i.e. the identity provider). The simulated server returns different attributes about the user than the UT IDP would provide, to solve this we created an AttributeExtractor interface that would return the needed attribute for the required information, such as username, employee/student number, and email.
The last issue that we faced was the fact that the role that an account has is per-course, meaning that

a user has multiple roles. Spring has a concept called Authority, which would be the authority of a user. We create our custom authorities, consisting of course ID + Role, and use them appropriately. We created an annotation called @Authorised that would tell if a user is allowed to access a resource or not.

Dependency Injection Many classes do not need to be instantiated multiple times, mainly because these classes do not have any state, so you can't break them. Also, if you need to change a dependency, you always need to go to each class that uses that dependency, or every place where the dependency is instantiated and change that. These two are only a few of the problems that dependency injection solves. The main idea behind it is the fact that we delegate the creation and management of the classes and objects to another entity. By default, it creates a single instance of the classes that are provided and injects it wherever it is required. This results in nearly no 'new' keyword present in our code, lose-coupling, and a lot of flexibility.

Returning only the required information It is important that no information that shouldn't be shown to a certain user or on a certain page is returned. To achieve this data transfer objects are used. An object is created that contains only the fields required for the user to see, while the database object might contain other information, that, for example, should be only available for the Coordinator. To make the conversion of the database object to the data transfer object we use mappers. Mappers are classes that have functions in them that take one object and return another one, based on the input. For that, we used a library called MapStruct.

Future changes Changes in code are inevitable and we made sure that every part of the app that has a slightly higher chance of being replaced or changed won't be a pain to work with. You can already see, from the general design, and the points mentioned above that we try to ensure that for all the classes, by good separation of concerns, good code reusability, as little boilerplate code as possible, and using interfaces where appropriate. There are classes, such as AccountService or PrincipalService that have a bigger chance of being changed, for example, when users are stored in another place or a different login type is used. For those we have defined interfaces and they are used everywhere instead of concrete classes.

7.2 Front-end

Now we will delve into the specifics of our design approach, by outlining the fundamental principles and considerations guiding our front-end development process.

Composition API There are 2 approcahes in Vue for building components: either using Options API or Composition API. While the first one is considered the traditional and easier way of building components, for our application, we chose the Composition API because it offers some advantages over the Options API. The main one is that we anticipated having some complex components, which would add the challenging task of maintaining and managing a large number of options. Now, we will discuss more about the Composition API and its advantages. It is a set of APIs that allows creating Vue components using imported functions instead of declaring options. One of its main advantages is the reusability of components. It allows strucuturing the code in a more functional manner (isolated by logic). Next, Composition API covers reactivity. It allows declaring a reactive state using the ref() function. It's main purpose and advantage (over plain variables) is that whenever there is a change to the ref's value, Vue automatically detects this and the DOM(data represantation of the objects) is updated accordingly. Also, refs can hold any value type. It also uses lifecycle hooks such as onMounted(), which waits for the component to be mounted to the DOM. It was mainly used by us when fetching data from APIs or other asynchronous operations.

Stores - Pinia In our application, there are multiple instances where the state of various elements is shared among multiple components. Therefore, using stores which provide a centralized location for managing application state was the way to go. Also, the stores use Vue's reactivity system, meaning that changes to a state in a store, would automatically update components depending on that state. Moreover, an advantage of using stores is that the alternative approach would require the use of props and emit events to communicate changes. When dealing with nested components, it would have the negative aspect called "props drilling", which means that the props would require to be passed down multiple layers to reach the destination component, leading to a verbose code, harder to be maintaned and/or refactored.

Our final choice of the store library was to use Pinia. It is at the moment of writing the report the official state management library for Vue. Also, since we use Composition API - it made sense to combine it with Pinia, since it is optimized especially for this set of APIs. In addition, Pinia has good integration with the Vue.js devtools, which is an official extension available in most modern browser (e.g. Google Chrome). This feature makes it easier to observe the state of the stores during debugging.

Vue-Router Vue Router is the official client-side routing solution for Vue. The URL updates accordingly, but the page doesn't need to be reloaded from the server. In our application, there are a lot of cases where we need to map routes with the given pattern for the same component (such as the assignment component which should be rendered for all the assignments with different ids). For these cases, we dynamically match the route by the means of a param (an example: "/course/:courseid/rubrics", where :courseid is the param). This means the same component instance is being reused when navigating through different courses, linking it to the example provided. We also use named routes, so that we can then use the name instead of the path when passing the ":to" prop to <router-link> (example: <router-link :to=" name: 'rubric', ...).

BootstrapVueNext Our design was made in BootstrapStudio and it already had quite a bit of functionality in it. Therefore, we wanted to make a smooth transition of the design to the front-end. This, however, wasn't possible and we were quite limited in solutions when choosing a framework which combines bootstrap and vue. The main problem was that we used the latest Bootstrap version (v5) in the design process, and also wanted to use the newer Vue.js version (v3). This meant that we couldn't use the more popular and established framework BootstrapVue (made for Bootstrap v4 and Vue.js v2) due to incompatibility of the versions. Then, we found that there was a framework called BootstrapVueNext which would suit our requirements. We were a bit hesitant in choosing it, due to it being in alpha version, but made the decision to give it a try, since it would be easier making use of already built-in components, rather than creating our own. It is much more efficient and suitable given the total amount of time dedicated to this project. It being alpha version meant we encountered some issues with that. Most major ones are lack of documentation, incomplete and missing features of some components (such as for tables, modals). Also, some bugs such as recursive calls were encountered during development. We have noticed during the development process, that changes in the code can sometimes trigger recursive calls, especially when using certain components like the modal from BootstrapVueNext. This issue doesn't seem to occur in production, but during development, it might require a page refresh to resolve and ensure smooth functionality.

Use of other libraries To streamline our development workflow, we incorporated additional libraries like lodash and gsap. Gsap, a JavaScript animation tool, helped us animate the transition of course cards when they're added to the DOM. Lodash, on the other hand, provided handy functions such as isEqual(), which allowed us to compare arrays deeply, ensuring they have the same properties and values.

7.3 Class diagram

There are a few interesting and non-standard decisions that were made while creating the class diagram and this section is dedicated to them. Below you'll find a description of the diagram as well as the aforementioned decisions.

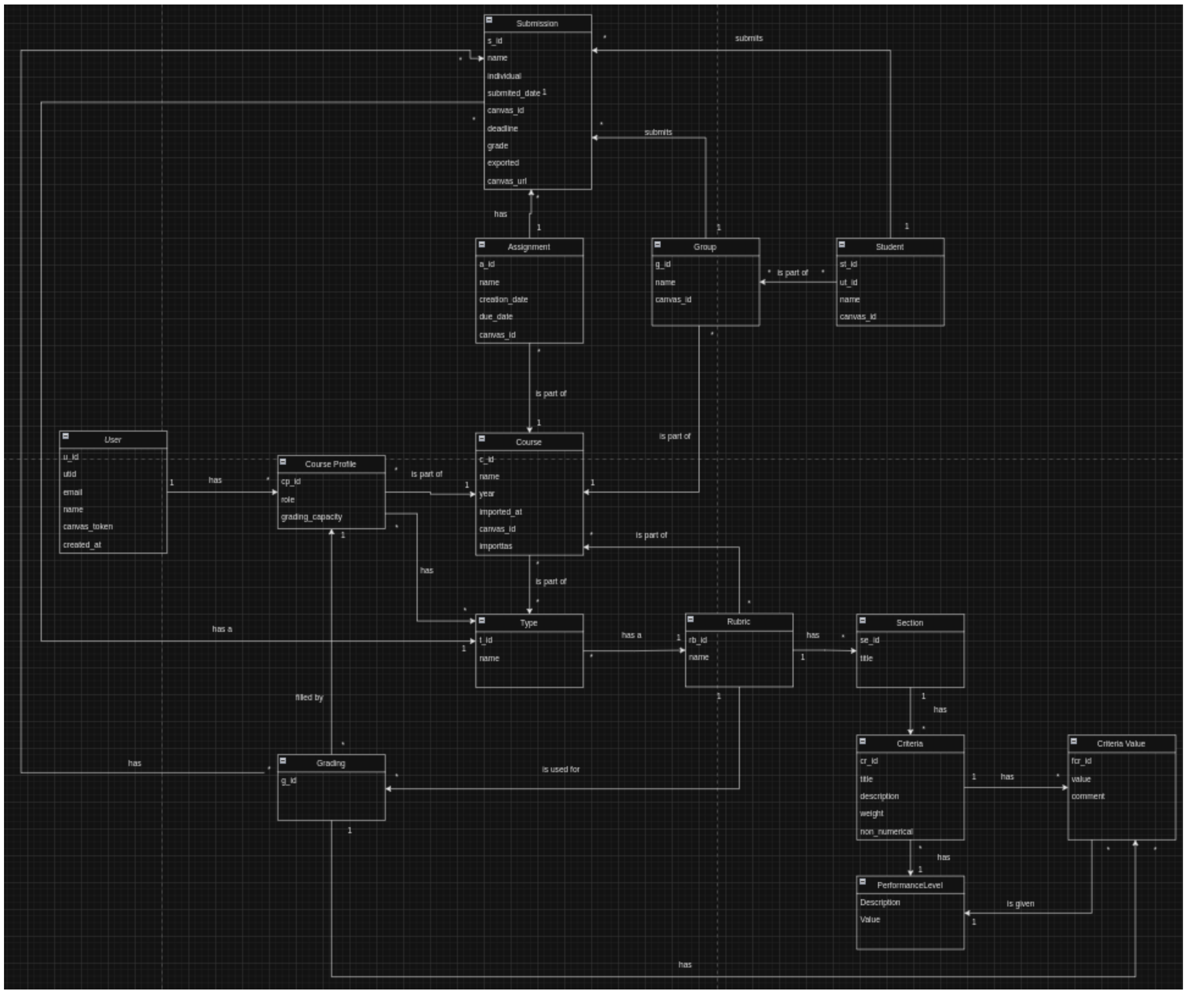


Figure 6: Class Diagram

User Each user, when logging in has a general account created, that would contain general information such as name, student/employee number, or email. For each course an account has a course profile, resulting in an account having multiple profiles. This is necessary because the preferences (grading capacity and preferred types of submissions) and roles are different for each course. 7

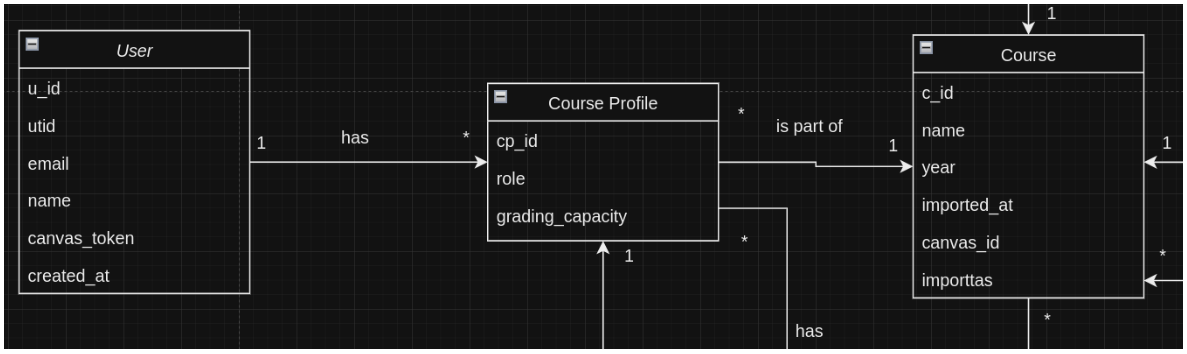


Figure 7: Account-Profile-Course

Course The course structure is quite simple, the course has assignments, assignments have submissions, submissions have a student group and student groups have multiple students. This is the same structure as in Canvas. 8

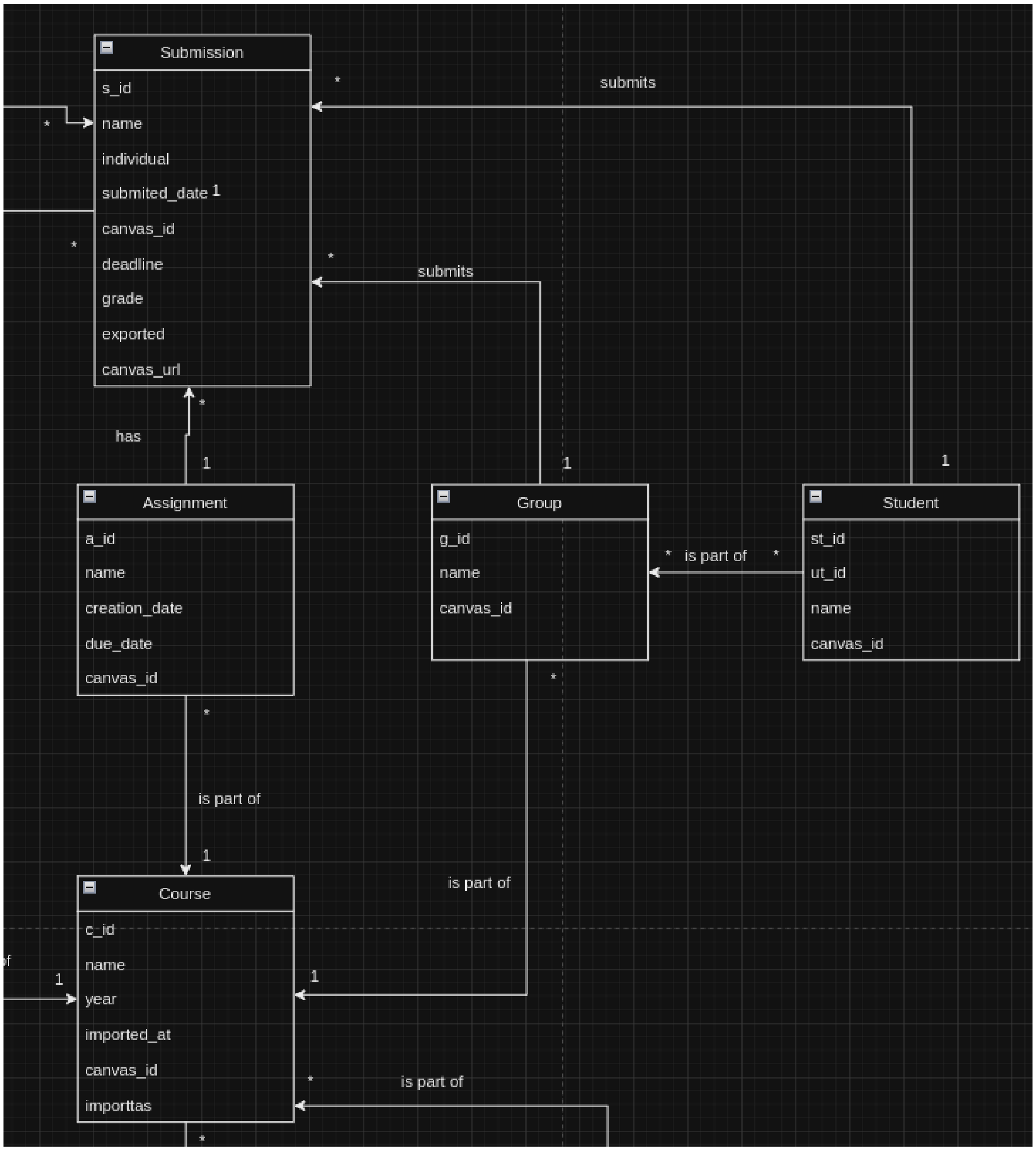


Figure 8: Course

Rubric The structure of a rubric is fairly simple. Each rubric has multiple sections, for example, "Presentation" or "Code". Each section has several criteria, for example, "Presentation quality" or "Code reusability". Criteria can be of two types, numerical and non-numerical. The main difference is that non-numerical ones have, so-called performance levels, which are suggestions for the graders of what grade should be given, for example, "Good" - 8, "Bad" - 4. These suggestions are not enforced and custom grades and comments can be given, such as in numerical criteria. 9

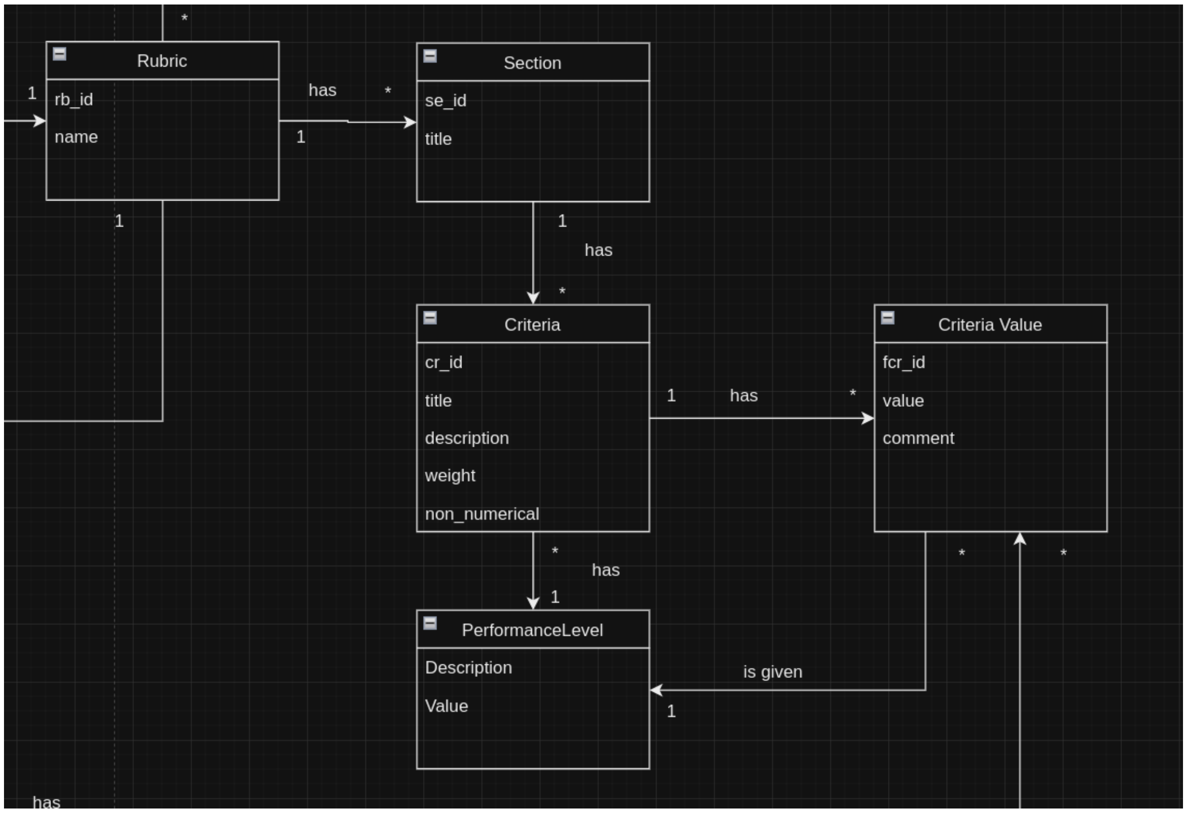


Figure 9: Rubric

Rubric and Type The most confusing part of the design is the types and rubrics part. Each submission has a specific rubric that needs to be assigned, but some submissions have the same rubric, while their type (topic) is different. This creates a problem when it comes to graders specifying which submissions they want to grade because two graders can be qualified to grade two submissions that have the same rubric, but different types. This situation would create the necessity of a submission having a type and a rubric. This is solved by the submission being assigned only the type, and to the type is assigned a rubric.

Grading The grading table is the connection table between graders and submissions. Each submission can have multiple gradings, the same goes for the graders, a grader can have multiple gradings. The way to map the grades and comments given by the graders to the rubric is through criteria values. A criteria value is linked to a criterion and has the following fields: grade, performance level, and comment. Each grading has multiple criteria values that match the criteria from the submission rubric.

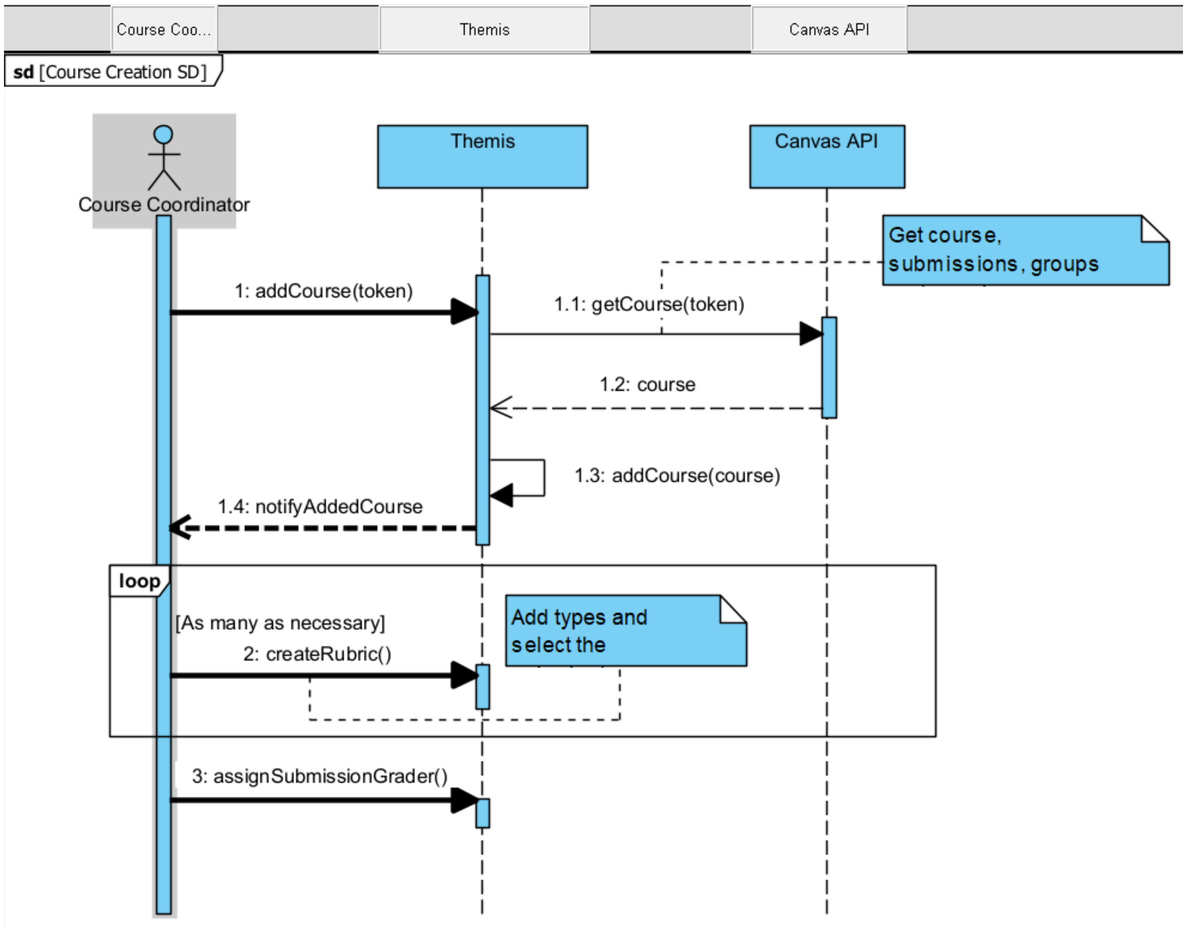


Figure 10: Course Creation

8 Testing

In order to make sure that our created product is of good quality and future changes in code won't break anything in the app thorough testing was done. There are multiple types of testing that were done: manual testing, unit testing, integration testing and acceptance testing.

8.1 Test plan

The main goal is to have enough tests that cover most of the classes from our project and the ones that are not covered using unit tests are tested through manual and usability testing. Some of the components, that communicate to external systems won't be tested, as it is expected that the other systems will work properly and because they are harder to test. Integration testing will be provided for every endpoint present, except the ones that contain communication with other systems. Usability and manual testing focus on both front-end and back-end integration.

Usability testing For usability testing we used our friends to show them the system, and, with the help of the manual they should be able to complete the flow of importing a canvas course to exporting the grades. This testing provided insight on how we should better write the user manual and what improvements could be done in terms of user experience. We also did usability testing with our clients, during the meetings and using Discord to showcase some of the functionality. We got feedback that was always incorporated into our app.

Manual testing In order to make sure that, for example, the backend meets the requirements for the frontend manual testing was done. This includes calling endpoints from the backend and testing if all the outputs are received and if all the values are the required ones. Also, some of the edge cases that, for example, the frontend can provide in terms of input data might not be always taken into consideration when testing the backend, but might appear during manual testing.

Functionality to be tested:

- View courses
- Import course
- Filter course by year
- View assignments
- Sync with canvas
- Import assignments
- View submissions
- View particular submission
- Add submission type
- Add submission grader
- Remove submission grader
- Change submission type
- Change submission deadline
- Grade submission using performance levels
- Grade submission using numerical values
- Finish grading

- Create rubric (add section, add criteria, delete section, delete criteria, add performance level)
- Add type
- Delete type
- Delete rubric
- Copy rubric
- Fast grader assignment
- Change account canvas token
- Change course profile grading preferences
- Change course profile grading capacity
- Change course profile role
- Export gradings
- Login
- Logout

8.2 Backend Test results

The main components of the backend, that were mentioned in the global design 6 were all carefully tested. For most of the components the required line coverage for our project was 80%, which was achieved for most of the classes. For each of them unit tests have been done to test positive and negative flows. In order to ensure that all these components work well together integration tests were done for each endpoint, except the ones that communicate with Canvas. In total, we have over 150 tests and covered more than 70% of all the lines from the project.

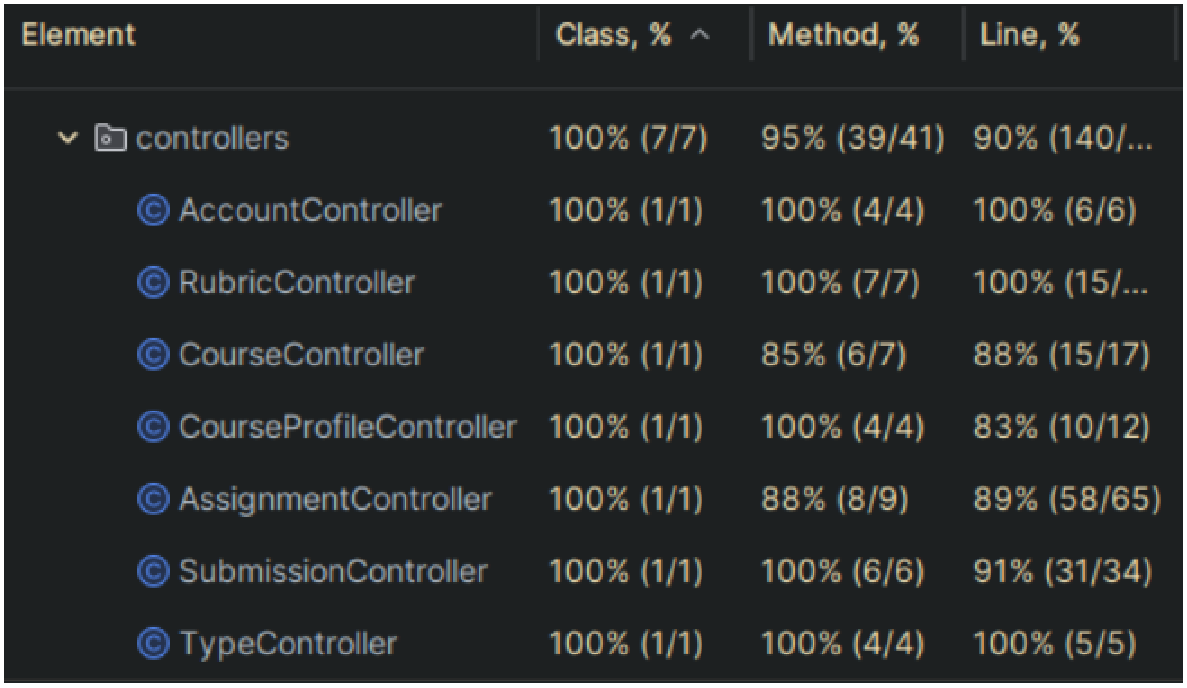
Mocking Not all the dependencies of a class can be instantiated and for a unit test, usually instantiating other dependencies is unnecessary. These dependencies can be mocked, i.e. default implementation can be given to all the methods and for required methods their behavior can be just simulated. To achieve this we used a library called Mockito.

Controller testing As can be seen in figure 11 we achieved our goal of testing at least 80% of the lines from the controllers. Tests for controllers include such cases as: calling the appropriate service, allowing and rejecting access to a resource based on the user role and input validation.

Mapper testing As can be seen in figure 12 we achieved our goal of testing at least 80% of the lines from the mappers. The only exception is the CanvasMapper, it wasn't tested due to the fact that mocking Canvas entities is more difficult than normal classes. Main two tests that were done for most of the mappers is if the mapper properly maps the provided object and that it doesn't crash when null values are provided, as null pointers inside the mappers created some problems while development.

Service testing As can be seen in figure 14 we achieved our goal of testing at least 80% of the lines from the services. The only exception is the CanvasService, it wasn't tested due to the fact that mocking Canvas entities is more difficult than normal classes. The tests for the service are all non-standard and usually include mocking the behaviour from the repository and testing the behaviour of the service based on the provided parameters and results returned by other services or repositories.

Repository testing Our app contains multiple custom-made queries that need to be tested. These queries require persistence storage. In such cases, an external database can't be used as we can't be sure of its status and the data that it already contains. In order to solve this problem we used an in-memory database called H2. It allowed us to create a database only for tests, which is removed afterward.



Element	Class, %	Method, %	Line, %
controllers	100% (7/7)	95% (39/41)	90% (140/...
AccountController	100% (1/1)	100% (4/4)	100% (6/6)
RubricController	100% (1/1)	100% (7/7)	100% (15/...
CourseController	100% (1/1)	85% (6/7)	88% (15/17)
CourseProfileController	100% (1/1)	100% (4/4)	83% (10/12)
AssignmentController	100% (1/1)	88% (8/9)	89% (58/65)
SubmissionController	100% (1/1)	100% (6/6)	91% (31/34)
TypeController	100% (1/1)	100% (4/4)	100% (5/5)

Figure 11: Controller test coverage

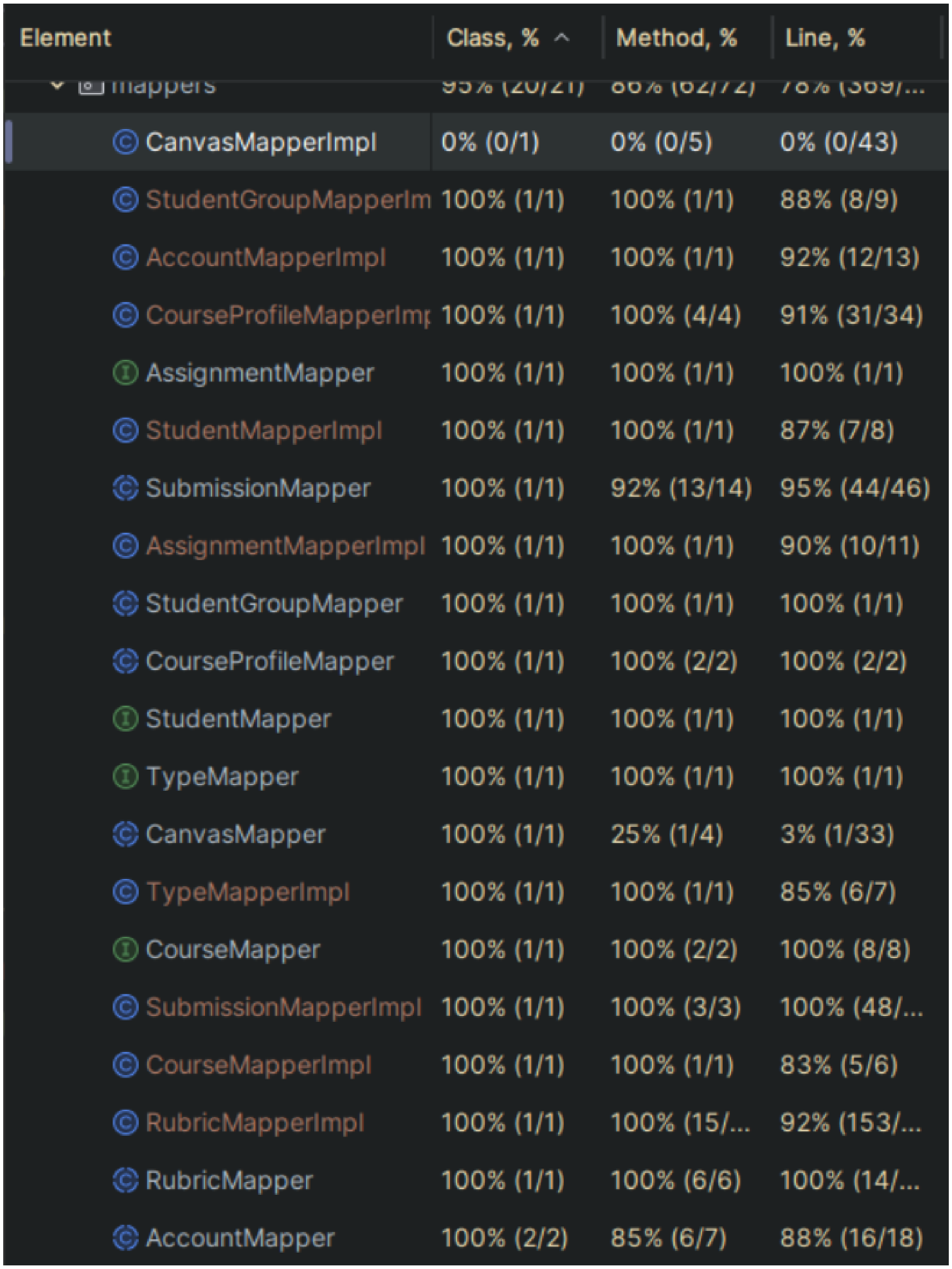
Integration testing In order to make sure that all the components function properly together we've done integration testing. We made calls to each endpoint of the app and tested if the required result is returned. The only endpoints that are not tested are the ones that require Canvas. For persistance storage the same H2 database is used.

Manual testing A big part of the bugs that were actually detected by doing manual testing, mostly during the frotnend development. This is because, the test environment is usually a bit different than the actual use-case. Manual testing also allowed to us to test some assumptions about how components interact and how the system works in general. Because all the classes related to Canvas can't be proprely tested manual testing was done for them and we were successful in detecting some interesting bugs, for example: individual submissions were not supported by the backend (now they are) or some fields, such as deadline of a submission can be null.

8.3 Frontend Test results

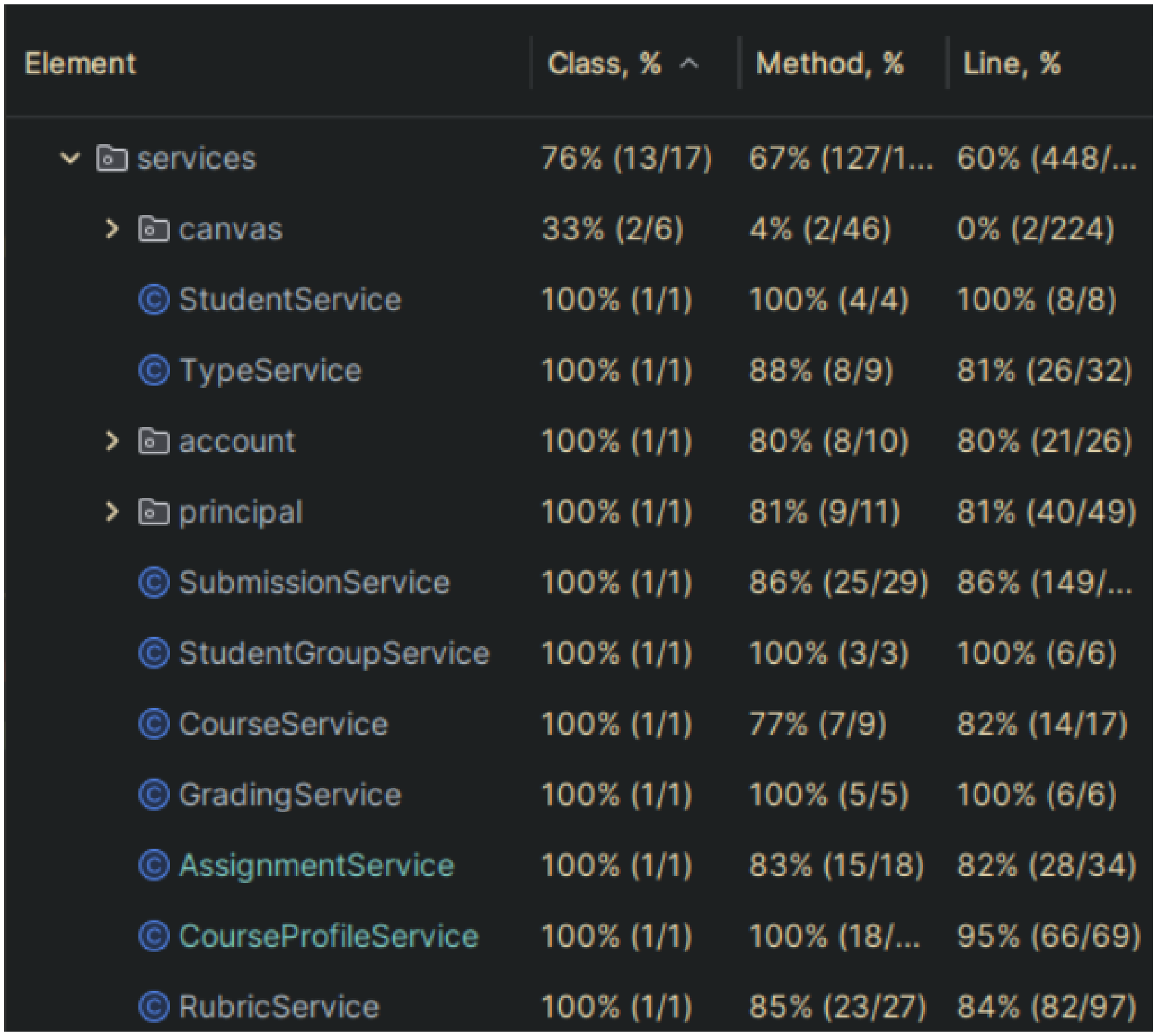
In our frontend testing approach, we used Vitest for unit and snapshot testing and Cypress for end-to-end (e2e) testing. The primary objective was to test key components and main pages of the application, focusing on areas prone to bugs. We followed a strategy that simulated typical user behaviors, specifically those of a coordinator and a grader, to ensure that our tests reflected real-world usage scenarios. For unit testing with Vitest, we wrote tests to verify the functionality and behavior of individual components, ensuring they functioned correctly in isolation. Vitest provided an API for writing these tests, allowing us to focus on testing behavior rather than implementation details. For e2e testing with Cypress, we created automated tests that simulated user interactions across multiple components and pages, verifying the end-to-end flow of the application. This approach helped us identify issues in the development process, leading to a more robust and reliable frontend.

Import Course testing When a user first time opens the website it starts his journey by importing a course. This is a crucial component to be tested because the website is useful only if there are available courses that could be imported from Canvas. In total we covered 80.5% of all the lines, and wrote 21 tests. Tests for importing a course include such cases as: rendering the modal when the button Import Course is clicked, allowing users to navigate through modal's components if their input



Element	Class, %	Method, %	Line, %
mappers	95% (20/21)	86% (62/72)	78% (369/...
CanvasMapperImpl	0% (0/1)	0% (0/5)	0% (0/43)
StudentGroupMapperIm	100% (1/1)	100% (1/1)	88% (8/9)
AccountMapperImpl	100% (1/1)	100% (1/1)	92% (12/13)
CourseProfileMapperImp	100% (1/1)	100% (4/4)	91% (31/34)
AssignmentMapper	100% (1/1)	100% (1/1)	100% (1/1)
StudentMapperImpl	100% (1/1)	100% (1/1)	87% (7/8)
SubmissionMapper	100% (1/1)	92% (13/14)	95% (44/46)
AssignmentMapperImpl	100% (1/1)	100% (1/1)	90% (10/11)
StudentGroupMapper	100% (1/1)	100% (1/1)	100% (1/1)
CourseProfileMapper	100% (1/1)	100% (2/2)	100% (2/2)
StudentMapper	100% (1/1)	100% (1/1)	100% (1/1)
TypeMapper	100% (1/1)	100% (1/1)	100% (1/1)
CanvasMapper	100% (1/1)	25% (1/4)	3% (1/33)
TypeMapperImpl	100% (1/1)	100% (1/1)	85% (6/7)
CourseMapper	100% (1/1)	100% (2/2)	100% (8/8)
SubmissionMapperImpl	100% (1/1)	100% (3/3)	100% (48/...
CourseMapperImpl	100% (1/1)	100% (1/1)	83% (5/6)
RubricMapperImpl	100% (1/1)	100% (15/...	92% (153/...
RubricMapper	100% (1/1)	100% (6/6)	100% (14/...
AccountMapper	100% (2/2)	85% (6/7)	88% (16/18)

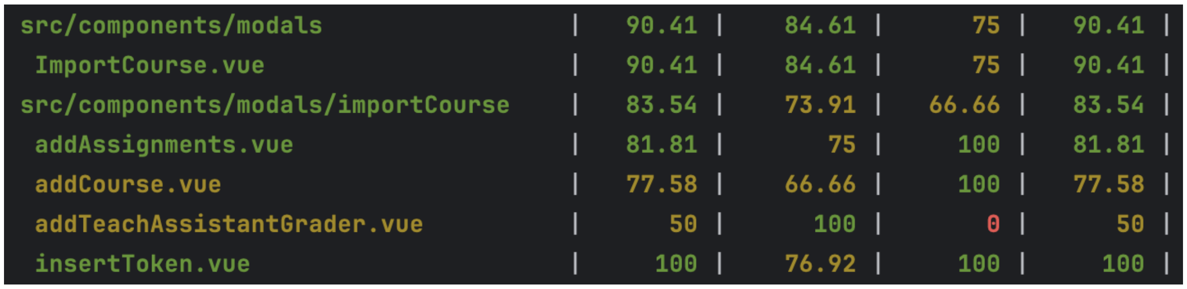
Figure 12: Mappers test coverage



Element	Class, %	Method, %	Line, %
services	76% (13/17)	67% (127/1...	60% (448/...
canvas	33% (2/6)	4% (2/46)	0% (2/224)
StudentService	100% (1/1)	100% (4/4)	100% (8/8)
TypeService	100% (1/1)	88% (8/9)	81% (26/32)
account	100% (1/1)	80% (8/10)	80% (21/26)
principal	100% (1/1)	81% (9/11)	81% (40/49)
SubmissionService	100% (1/1)	86% (25/29)	86% (149/...
StudentGroupService	100% (1/1)	100% (3/3)	100% (6/6)
CourseService	100% (1/1)	77% (7/9)	82% (14/17)
GradingService	100% (1/1)	100% (5/5)	100% (6/6)
AssignmentService	100% (1/1)	83% (15/18)	82% (28/34)
CourseProfileService	100% (1/1)	100% (18/...	95% (66/69)
RubricService	100% (1/1)	85% (23/27)	84% (82/97)

Figure 13: Services test coverage

is valid, showing relevant messages based on the user input, update the courses page after submit button was pressed.



```
src/components/modals                   |   90.41 |   84.61 |      75 |   90.41 |
 ImportCourse.vue                       |   90.41 |   84.61 |      75 |   90.41 |
src/components/modals/importCourse      |   83.54 |   73.91 |   66.66 |   83.54 |
 addAssignments.vue                     |   81.81 |      75 |     100 |   81.81 |
 addCourse.vue                          |   77.58 |   66.66 |     100 |   77.58 |
 addTeachAssistantGrader.vue            |      50 |     100 |       0 |      50 |
 insertToken.vue                        |     100 |   76.92 |     100 |     100 |
```

Figure 14: Import course component coverage

Snapshot testing Our testing approach also focused on validating the visual consistency and responsiveness of the user interface. We used snapshot testing to capture the appearance of components and pages, ensuring that they rendered correctly.

E2E testing In our end-to-end (e2e) testing, we aimed to test the functionality of key features within the rubric system. Specifically, we focused on two critical aspects: adding and deleting individual sections of a rubric, as well as adding and removing entire rubrics. These tests ensured that users could use rubric structures, without worrying of seeing unexpected error messages. Additionally, we extended our e2e testing to cover other essential pages, including the Assignments, Graders, and Profile pages. This approach allowed us to validate the overall functionality and user experience across multiple areas of the application, ensuring its reliability in real-world scenarios.

Manual testing In addition to automated testing, manual testing played an important role in ensuring the quality and usability of our frontend application. Manual testing allowed us to interact with the application as end-users, and uncovered issues and edge cases that may not have been captured by automated tests. This involved a diverse set of test scenarios, covering various user journeys and use cases to validate the application's functionality. Additionally, manual testing provided an opportunity to evaluate the application's visual aesthetics, ensuring that the user interface elements were well-designed and aligned with the overall design.

9 Product

All the code for the product can be found at this link (you need access to it): here. Also, the app will be running at: https://themis.apps.utwente.nl. It might not be online when reading this or a different version might be present with some functionality disabled (for security and privacy reasons). Please contact one of the supervisors in order to get credentials to test the app (if they are required). Documentation can be found in each specific folder of the components (i.e. frontend and backend).

User Manual

This manual is centered around the pages of the application and the functionalities available on that page.

9.1 Login

The first page you are greeted with is the login page (Figure 9.15) which will redirect you for logging in with your UT account.

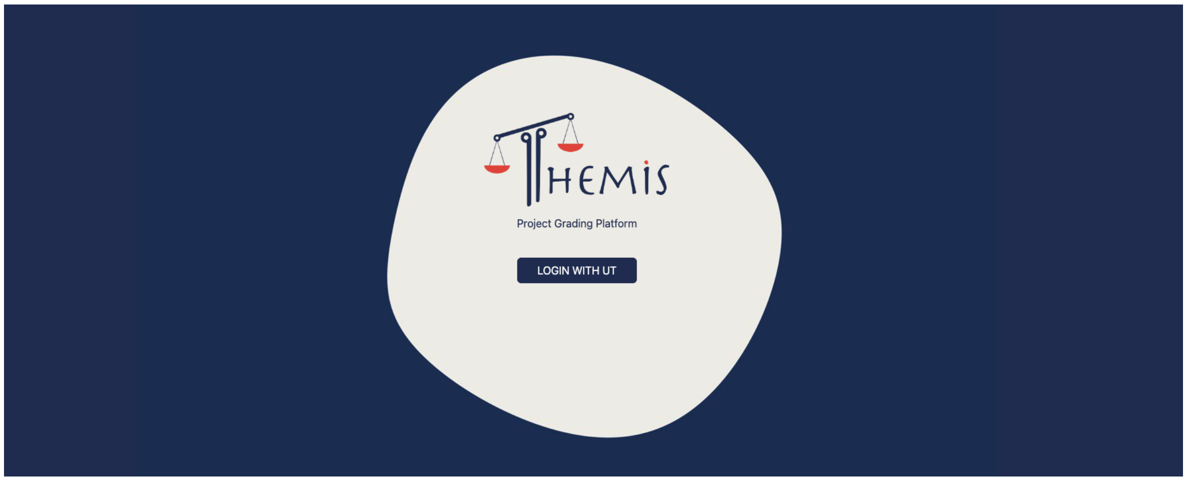


Figure 9.15: Log In Page

To sign out, you would click on their profile icon on the left sidebar, and choose the Sign Out option from the menu. (Figure 9.16)

9.2 Home Page

After a successful login, the user will be redirected onto the main page. You will be presented with an overview of all of the courses that have been imported on the platform (Figure 9.16). On the top bar there will have the option to filter the courses based on the academic year, and also import a course from Canvas.

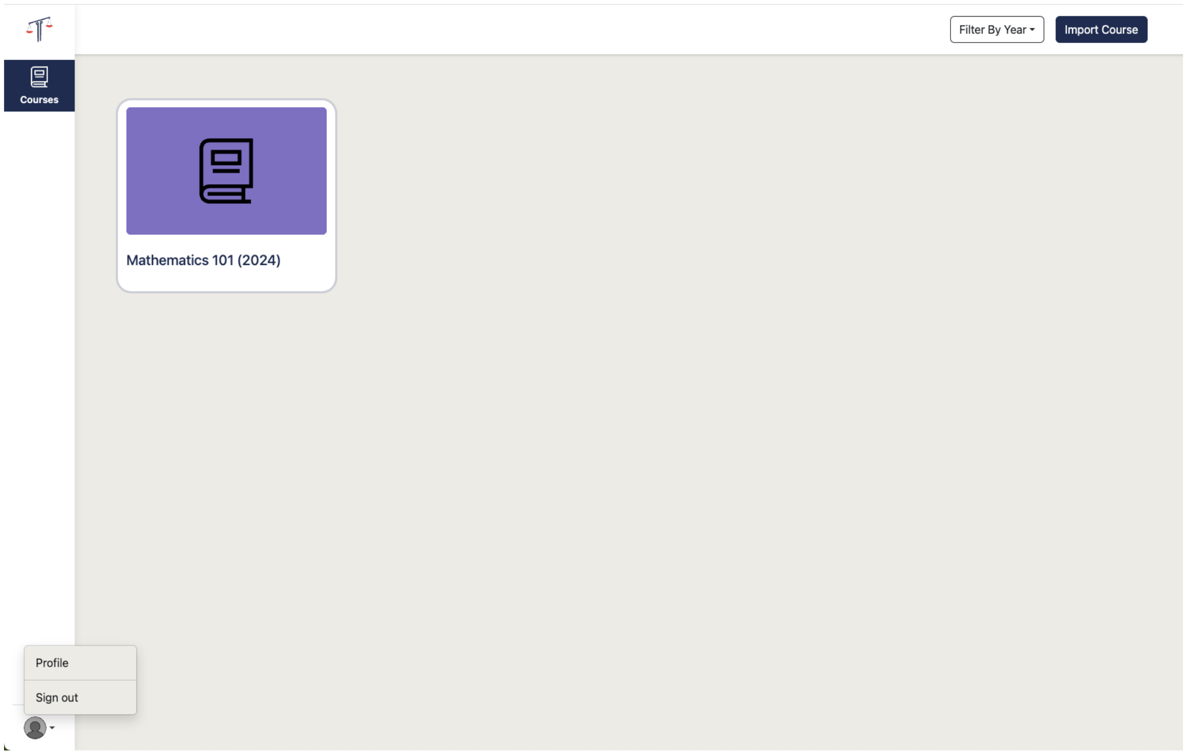


Figure 9.16: Home

Import a course. There are 4 simple steps to import a course:

1. Introduce a valid Canvas token (Figure 9.17)

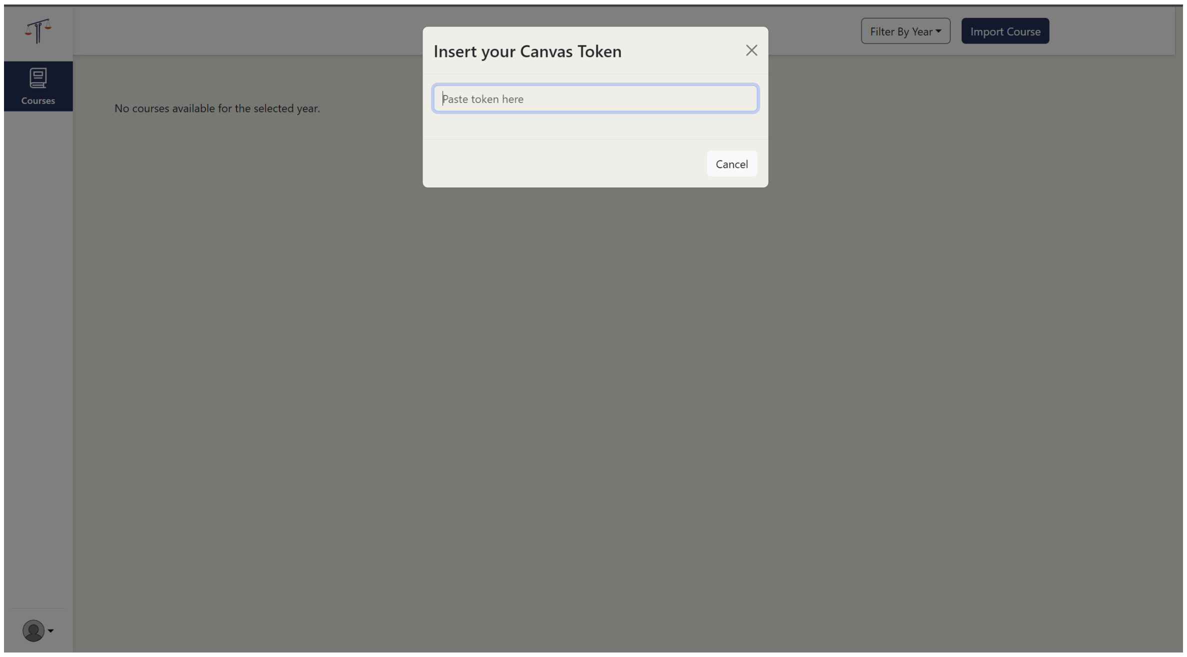
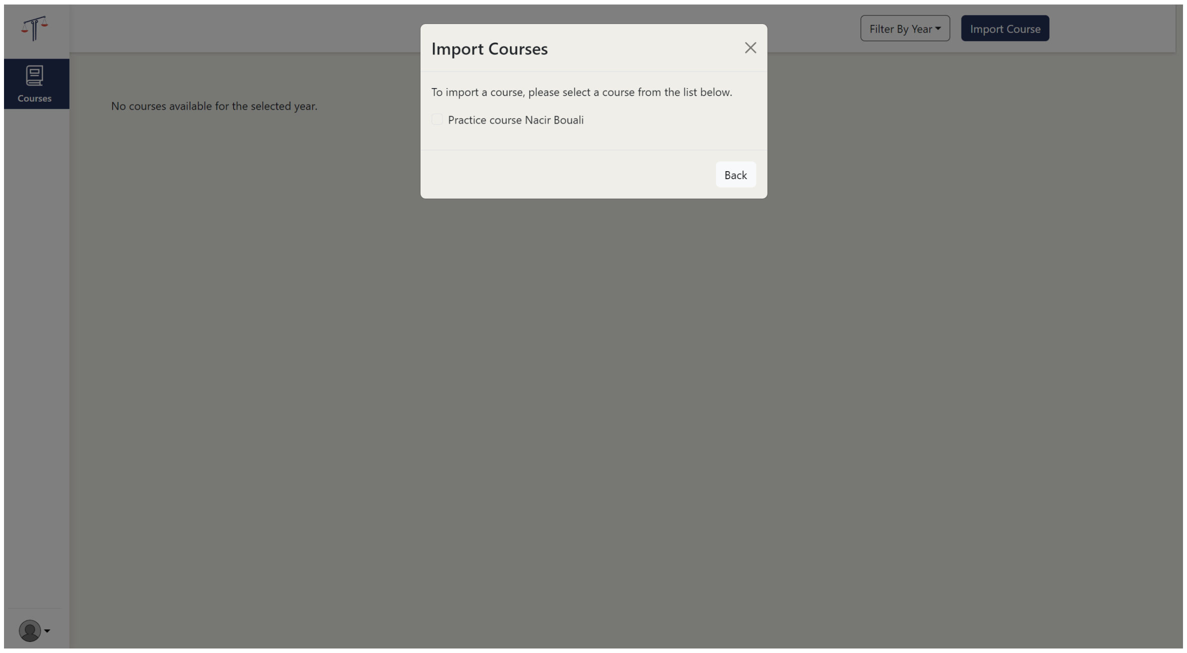


Figure 9.17: Canvas Token

2. Choose the course that you want to import (Figure 9.18)



No courses available for the selected year.

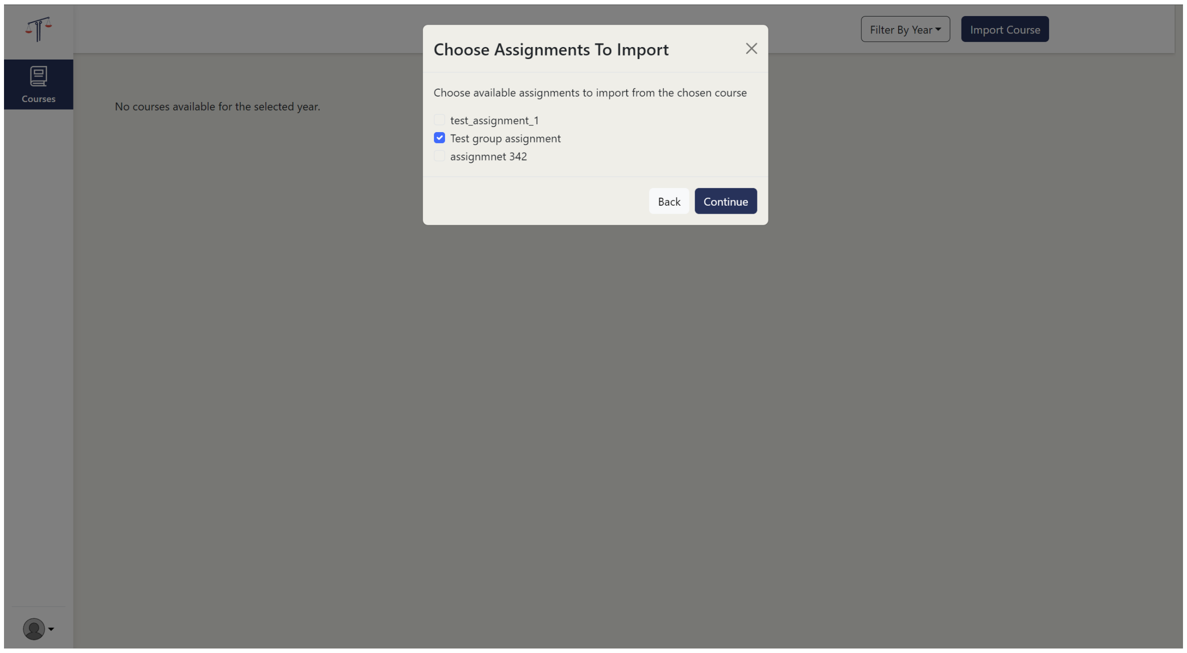
Import Courses

To import a course, please select a course from the list below.

- [ ] Practice course Nacir Bouali

Figure 9.18: Choose course for import

3. Choose the assignment(s) that you want to import (Figure 9.19)



No courses available for the selected year.

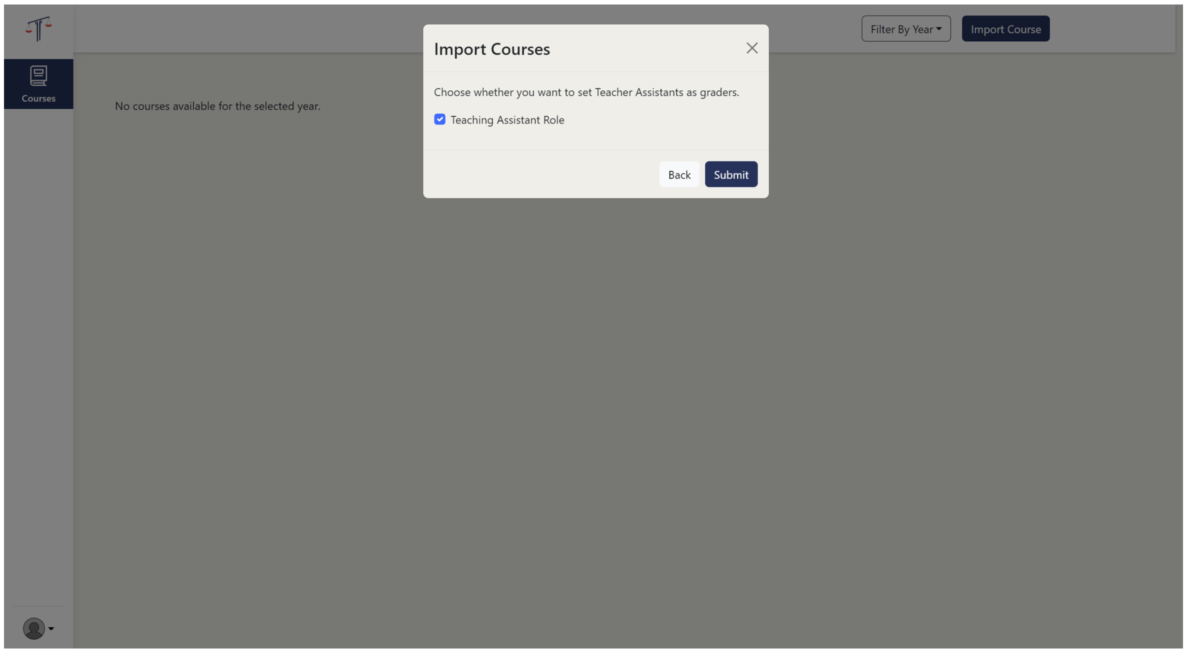
Choose Assignments To Import

Choose available assignments to import from the chosen course

- [ ] test_assignment_1
- [x] Test group assignment
- [ ] assignmnet 342

Figure 9.19: Choose assignments for import

4. Choose whether or not you want to import teaching assistants as graders. Finally, submit the request. (Figure 9.20)



No courses available for the selected year.

Import Courses

Choose whether you want to set Teacher Assistants as graders.

☑ Teaching Assistant Role

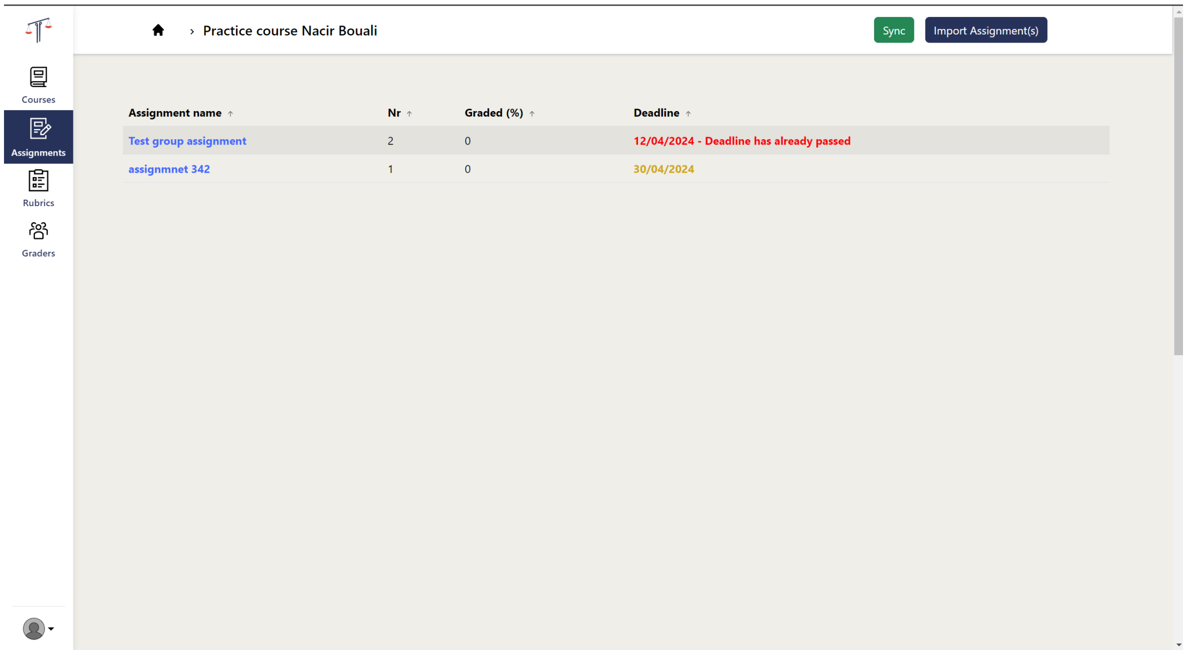
Figure 9.20: Choose teaching assistants as graders

9.3 Course Page

The course page contains the list of all assignments associated with that specific course, as well as information about each of them such as number of submissions for each assignment, the deadline for grading and statistics about grading progress. On the left side bar there are navigation buttons to pages associated to the course. To navigate to a specific assignment click on his name.

9.3.1 Coordinator View

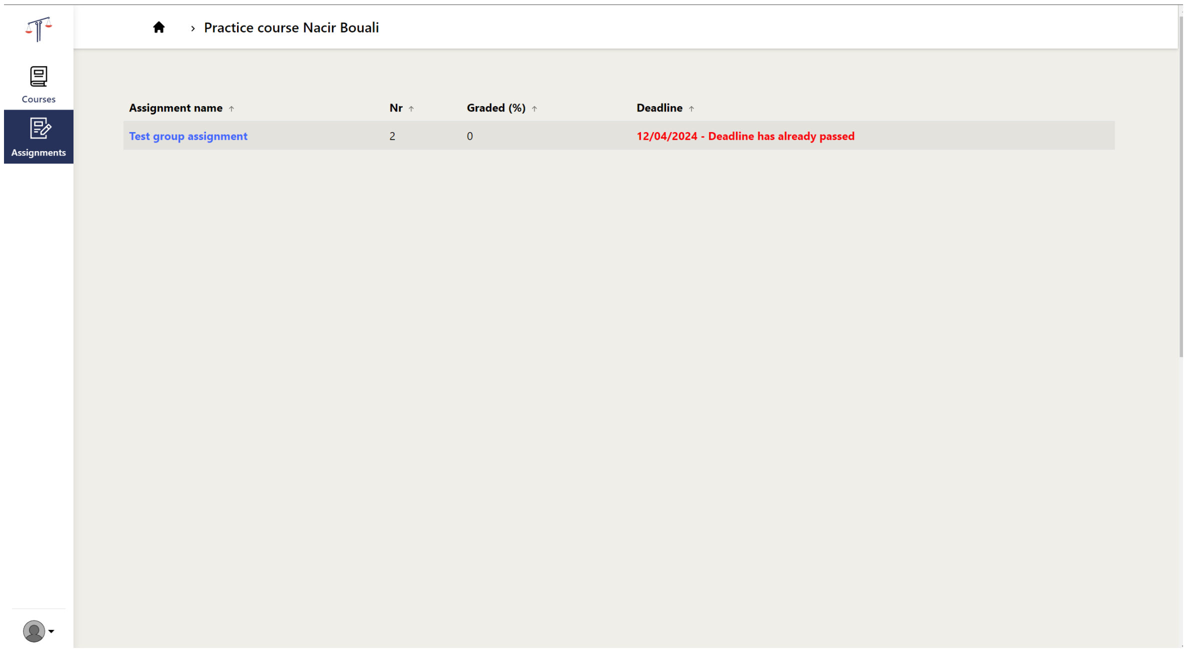
If you are logged in as a coordinator (Figure 9.21) you can import assignments from canvas and synchronised all the data with Canvas. This button is useful to update the submissions, or if the role of a user is changed from Canvas. Coordinator has access to the rubrics page. To navigate there, a 'Rubrics' button is shown on the left side bar. 'Graders' button is available only for this role, and can be used to access the graders page.



Assignment name	Nr	Graded (%)	Deadline
Test group assignment	2	0	12/04/2024 - Deadline has already passed
assignmnet 342	1	0	30/04/2024

Figure 9.21: Coordinator course main page

9.3.2 Grader View



Assignment name	Nr	Graded (%)	Deadline
Test group assignment	2	0	12/04/2024 - Deadline has already passed

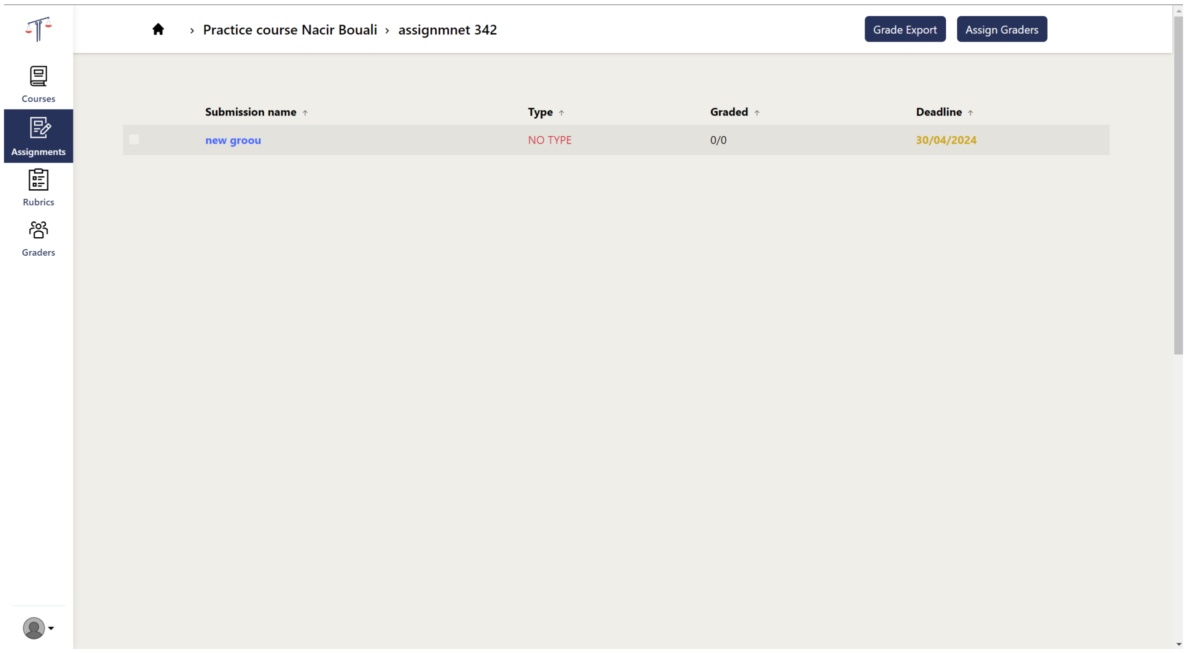
Figure 9.22: Grader course main page

9.4 Submissions Page

This is the page that holds all the submissions for a specific assignment and relevant information about each of them such as: the names of the students that submitted the work, type of submission, the current assigned grade, whether it was graded and by how many graders and what is the grading deadline. The deadlines are color coordinated. Each submission can have a unique deadline (that can be the case for exchange students that may require their grades sooner than regular UT students).

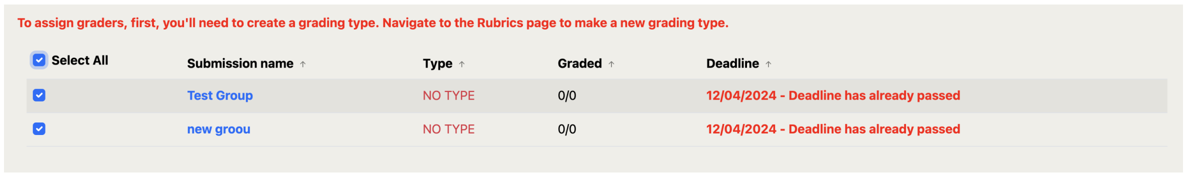
9.4.1 Coordinator View

If you are logged in as a coordinator (Figure 9.23), there are multiple actions that can be done. The most important one is to set a type. However , initially there are no types (Fig 9.24). Therefore the user must go to 'Rubrics' page to create a rubric and a type. After creating the type and selecting a submission a dropdown list with all the available types will appear above the submissions table. (Figure 9.25) After selecting a type, the submission(s) will be updated with the new type. (Figure 9.26)



Submission name	Type	Graded	Deadline
new groou	NO TYPE	0/0	30/04/2024

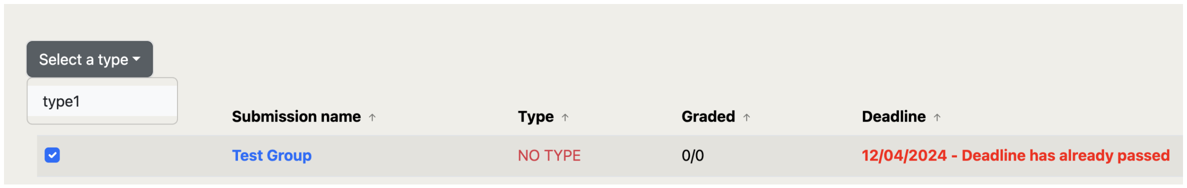
Figure 9.23: Coordinator Submissions Page



To assign graders, first, you'll need to create a grading type. Navigate to the Rubrics page to make a new grading type.

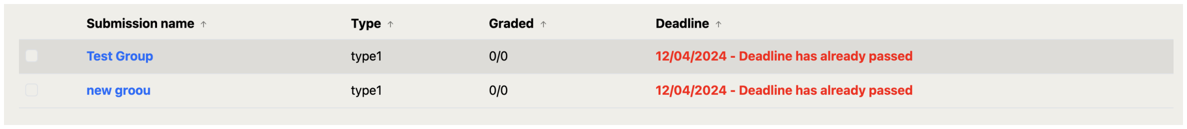
Select All	Submission name	Type	Graded	Deadline
✓	Test Group	NO TYPE	0/0	12/04/2024 - Deadline has already passed
✓	new groou	NO TYPE	0/0	12/04/2024 - Deadline has already passed

Figure 9.24: Submissions no types



	Submission name	Type	Graded	Deadline
✓	Test Group	NO TYPE	0/0	12/04/2024 - Deadline has already passed

Figure 9.25: Submissions set types

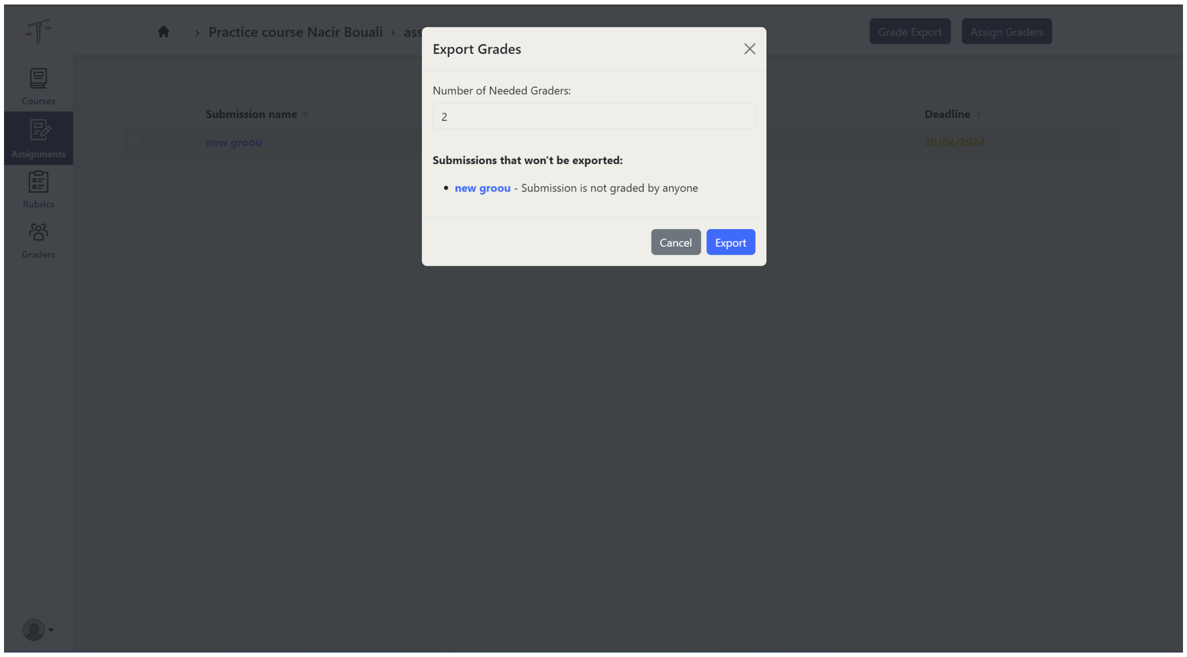


	Submission name	Type	Graded	Deadline
	Test Group	type1	0/0	12/04/2024 - Deadline has already passed
	new groou	type1	0/0	12/04/2024 - Deadline has already passed

Figure 9.26: Submissions with type

Coordinator can perform some actions regarding the assignment as a whole :

- exporting grades (Fig 9.27) which gives an overview of the number of grades to be exported and number of warnings : grades for which the difference in grades of the two graders was greater than 1
- assigning graders (discussed in the next section 9.5)



Export Grades

Number of Needed Graders:

2

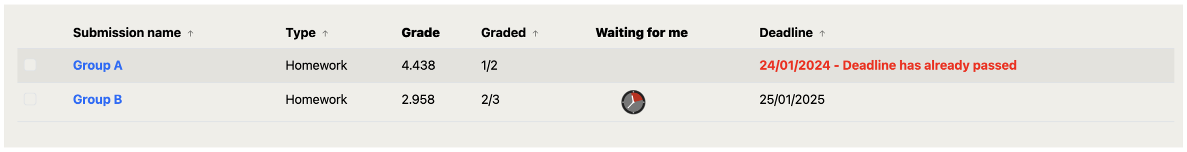
Submissions that won't be exported:

- **new groou** - Submission is not graded by anyone

Figure 9.27: Grade Export

9.4.2 Grader View

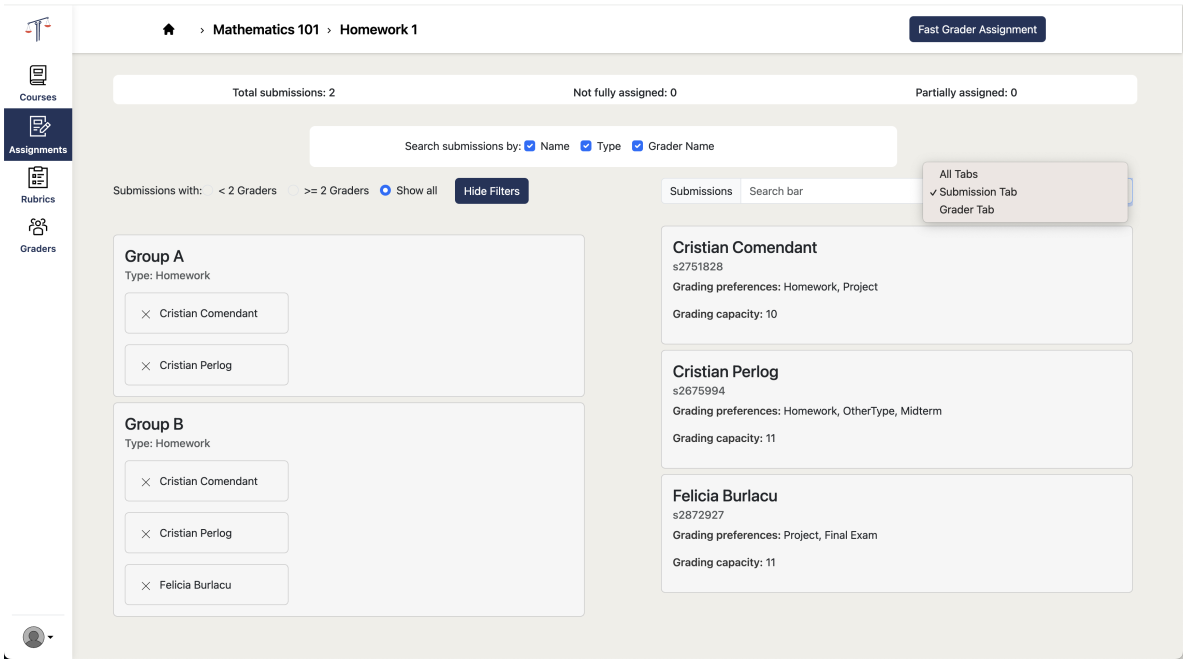
If you are logged in as a grader (Figure 9.28), besides the displayed information, you have the option to sort based on submission titles. Additional to that a 'waiting for me' notification will pop up if other graders already graded the submission. The purpose of this icon is to trigger a small alarm in order to grade the submission as soon as possible.



Submission name	Type	Grade	Graded	Waiting for me	Deadline
Group A	Homework	4.438	1/2		24/01/2024 - Deadline has already passed
Group B	Homework	2.958	2/3	●	25/01/2025

Figure 9.28: 'Waiting for me' notification

9.5 Assign Graders



Total submissions: 2
Not fully assigned: 0
Partially assigned: 0

Search submissions by: ✓ Name ✓ Type ✓ Grader Name

Submissions with: < 2 Graders >= 2 Graders [● Show all]

Group A
Type: Homework
× Cristian Comendant
× Cristian Perlog

Group B
Type: Homework
× Cristian Comendant
× Cristian Perlog
× Felicia Burlacu

Cristian Comendant
s2751828
Grading preferences: Homework, Project
Grading capacity: 10

Cristian Perlog
s2675994
Grading preferences: Homework, OtherType, Midterm
Grading capacity: 11

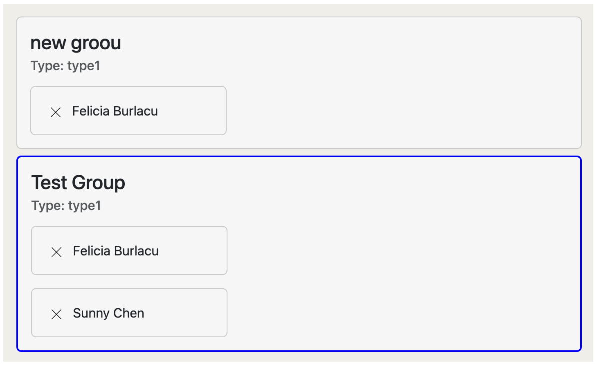
Felicia Burlacu
s2872927
Grading preferences: Project, Final Exam
Grading capacity: 11

Figure 9.29: Coordinator 'assign graders' page

The 'assign graders' page displays all submissions that need to be assigned for grading along with some information about them at the top (number of total submissions, number of submissions that are not fully assigned - do not have 2 graders assigned - at the moment, number of submissions that are partially assigned - that have one grader - at the moment). This can be done in a just a few steps:

- select one or multiple submissions for grading by clicking on them
- select the grader from the list on the right by clicking on the according card. This will assign the grader to all the selected submissions.

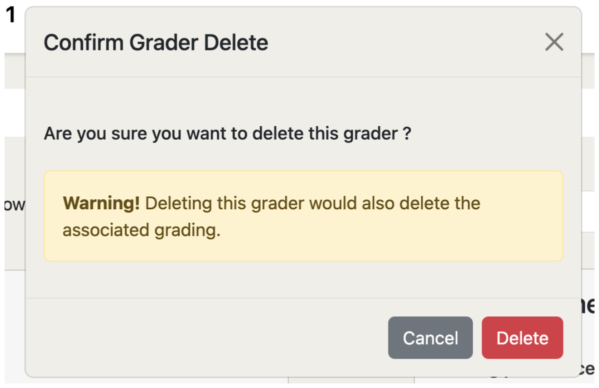
In case you want to delete an assigned grader, he can be removed by clicking on the 'x' next to his name. (Fig 9.30). It is worth mentioning that if a grader already started to grade a submission and suddenly the coordinator wants to delete him from that submission a warning message will pop up. (Fig 9.31)



new groou
Type: type1
× Felicia Burlacu

Test Group
Type: type1
× Felicia Burlacu
× Sunny Chen

Figure 9.30: Submission Card with assigned graders


Confirm Grader Delete

Are you sure you want to delete this grader ?

Warning! Deleting this grader would also delete the associated grading.

Figure 9.31: Delete grader Warning

Additionally, there is an option in the top bar, called ’Fast Grader Assignment’ to automatically assign graders for all submissions. This depends on the grader’s capacity and grader’s preferences type.

The page contains 3 search bars. The main search bar is used to search over both submissions and graders. The submission search bar could be used to search specifically on the submission tab. Grader search bar is used on the grader tab. For more flexibility unique filters are available for submission and grader search bars.

It might happen that a specific submission to not have a type. In this scenario, the user would not be allowed to set graders, that funtion would be disabled unless a type is set. For additional comfort the type could also be set up on this page by selecting all submissions that do not have a type. (Fig 9.32)

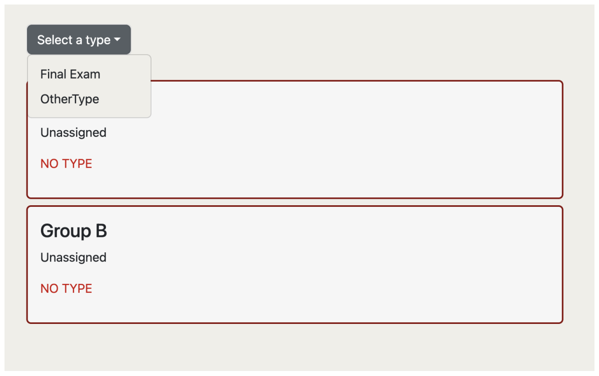


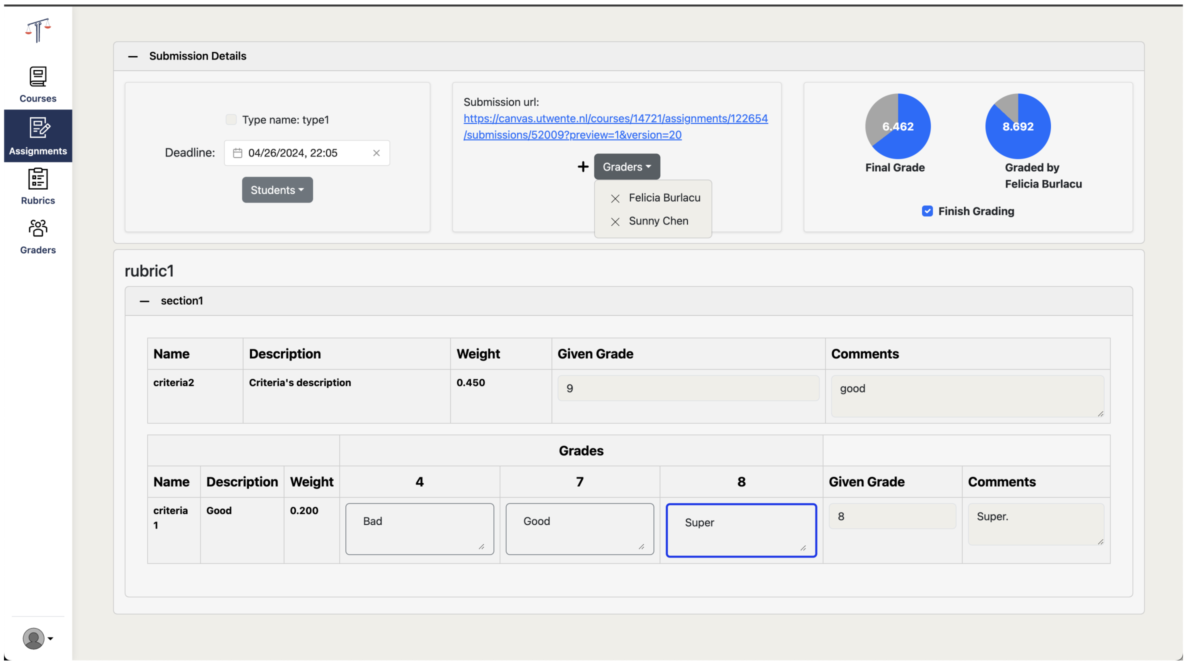
Figure 9.32: ’Selected submissions with ’NO TYPE”

9.6 Grade Submission

This is the page where teachers grade their assigned submissions. On top of the page are displayed the submission details. Below, an interactive and user-friendly rubric is displayed. Each section is extensible, so in case there are many of them and you want to keep your focus only on a specific one, you can minimize them by clicking on its name.

9.6.1 Coordinator View

Coordinator have more options than a grader. (Fig 9.33) He can set deadlines, add graders, see the final grade and the current grade given by a specific grader, and he can modify the grade of any submission by changing the rubric values. On the submission details tab, coordinator can set or change an existing type. However, if the type is changed, a warning message will pop up, notifying that by changing the type all the associated gradings would be deleted. (Fig 9.34)



Submission Details

Type name: type1

Deadline: 04/26/2024, 22:05

Students

Submission url: https://canvas.utwente.nl/courses/14721/assignments/122654/submissions/52009?preview=1&version=20

Graders
- Felicia Burlacu
- Sunny Chen

6.462 Final Grade

8.692 Graded by Felicia Burlacu

Finish Grading

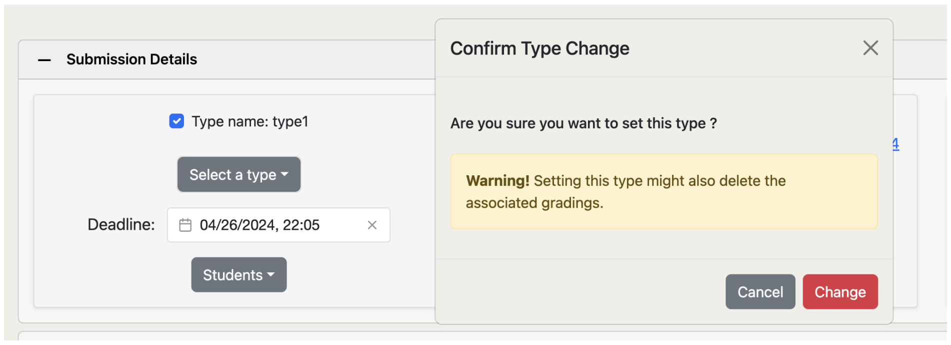
rubric1

section1

Name	Description	Weight	Given Grade	Comments
criteria2	Criteria's description	0.450	9	good

			Grades				
Name	Description	Weight	4	7	8	Given Grade	Comments
criteria 1	Good	0.200	Bad	Good	Super	8	Super.

Figure 9.33: 'Coordinator Grade Submission Page'



Submission Details

Type name: type1

Select a type

Deadline: 04/26/2024, 22:05

Students

Confirm Type Change

Are you sure you want to set this type ?

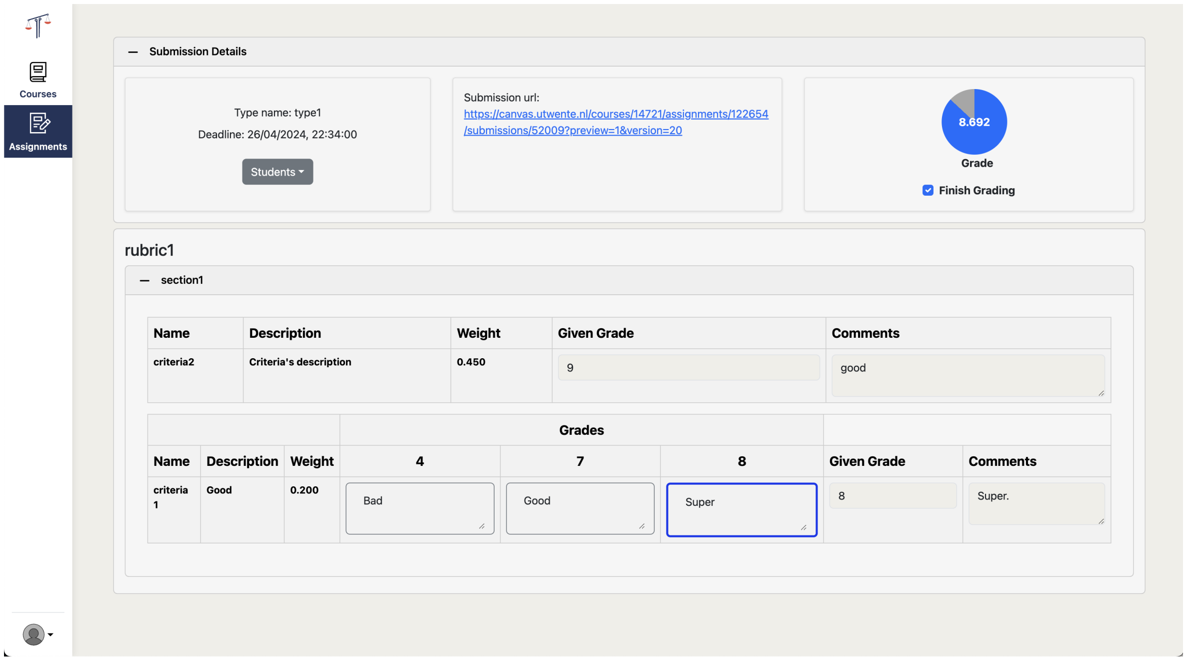
Warning! Setting this type might also delete the associated gradings.

Cancel Change

Figure 9.34: 'Change Type Warning'

9.6.2 Grader View

When grading the submission there are 2 types of criteria. Numerical and non-numerical. Non-numerical might contain performance levels. To grade a non-numerical criteria, press on the performance level card and its grade value and comment will be automatically filled. Comments boxes will change their text based on the performance level card. However, it will remain fixed if a custom comment is added. When the submission was graded the grader must click on the finish grading button. After that, the user can hover over the 'finish grading' checkbox to see the last date when the submission was graded. (Fig 9.36)



Type name: type1
Deadline: 26/04/2024, 22:34:00

Submission url: https://canvas.utwente.nl/courses/14721/assignments/122654/submissions/52009?preview=1&version=20

Name	Description	Weight	Given Grade	Comments
criteria2	Criteria's description	0.450	9	good

			Grades				
Name	Description	Weight	4	7	8	Given Grade	Comments
criteria 1	Good	0.200	Bad	Good	Super	8	Super.

Figure 9.35: 'Grader Grade Submission Page'

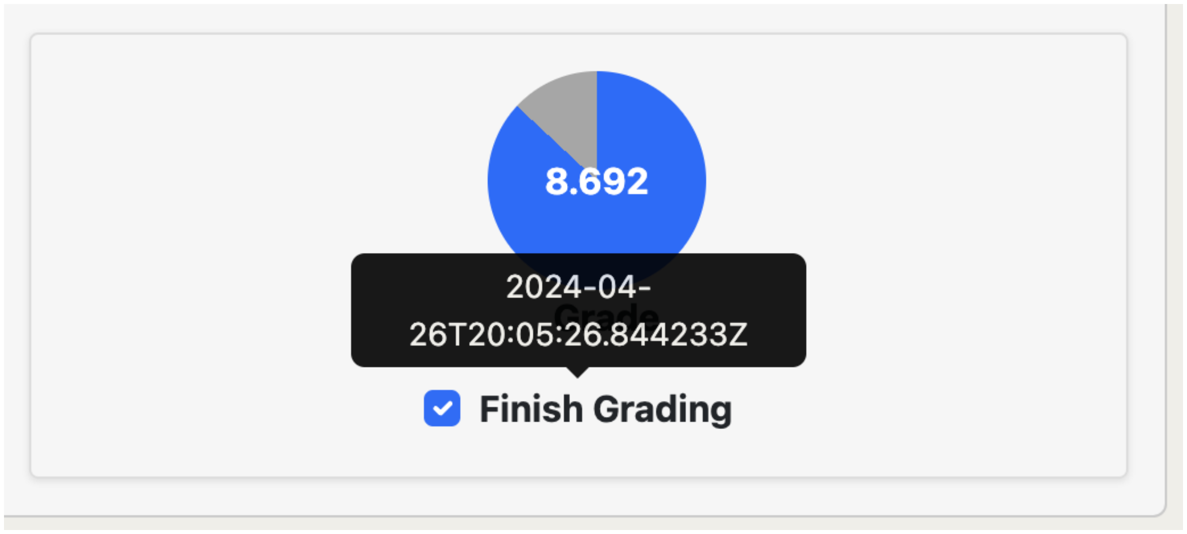
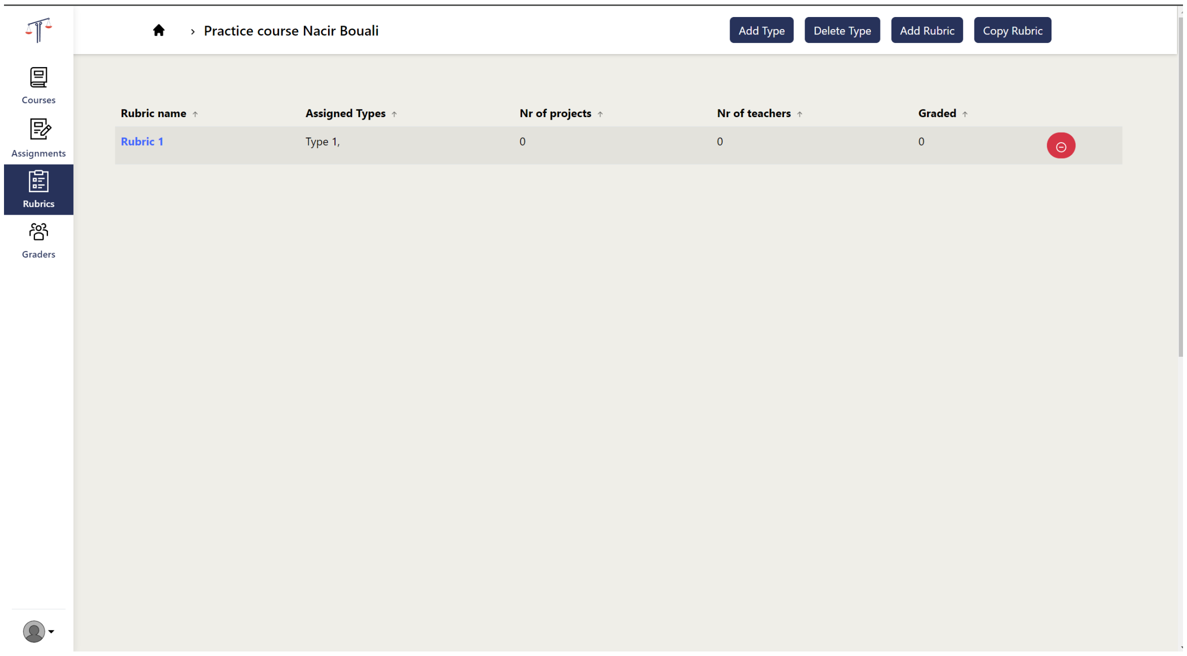


Figure 9.36: 'Finish grading date'

9.7 Rubrics

The rubrics page holds all the information about the created rubrics for a specific course, the assigned types for each of them, the number of projects that are assigned to this rubric, the number of teachers (graders) that are assigned to those projects, and the grading progress.

You have access to rubrics only if you are signed in as a coordinator. On the rubrics page, you have the option to delete and alter rubrics and their associated types (Fig 9.37).



Rubric name	Assigned Types	Nr of projects	Nr of teachers	Graded
Rubric 1	Type 1,	0	0	0

Figure 9.37: Rubrics Page

9.7.1 Add Rubric Type

To add a type, click on the "Add Type" button, write the type name, and select the target rubric. Click "Submit"

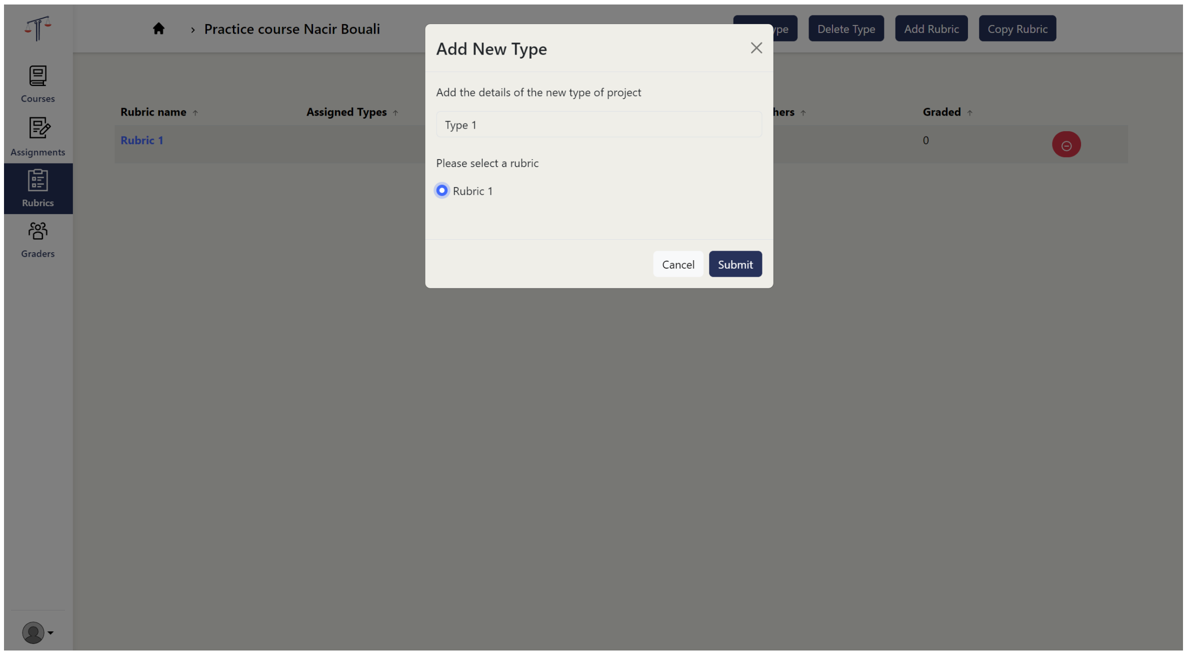


Figure 9.38: Add type

9.7.2 Delete Rubric Type

To delete a type, click on the "Delete Type" button. Then, choose the respective type from the drop-down list under the desired rubric. Click "Confirm".

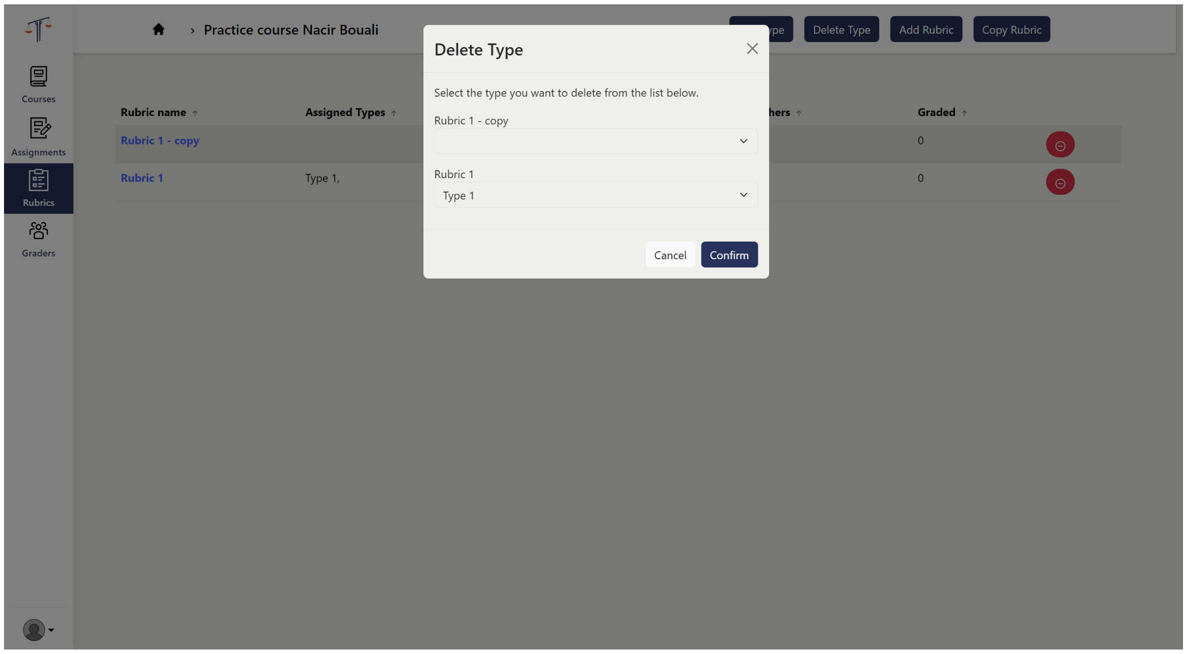


Figure 9.39: Delete type

9.7.3 Create New Rubric

To create a new rubric click on "Add Rubric" 9.40. Then type the rubric name in the input field and click "Create". The rubric should appear in the list of rubrics.
To view the sections and criteria for a specific rubric, click on the name of the desired rubric from the

rubrics list.

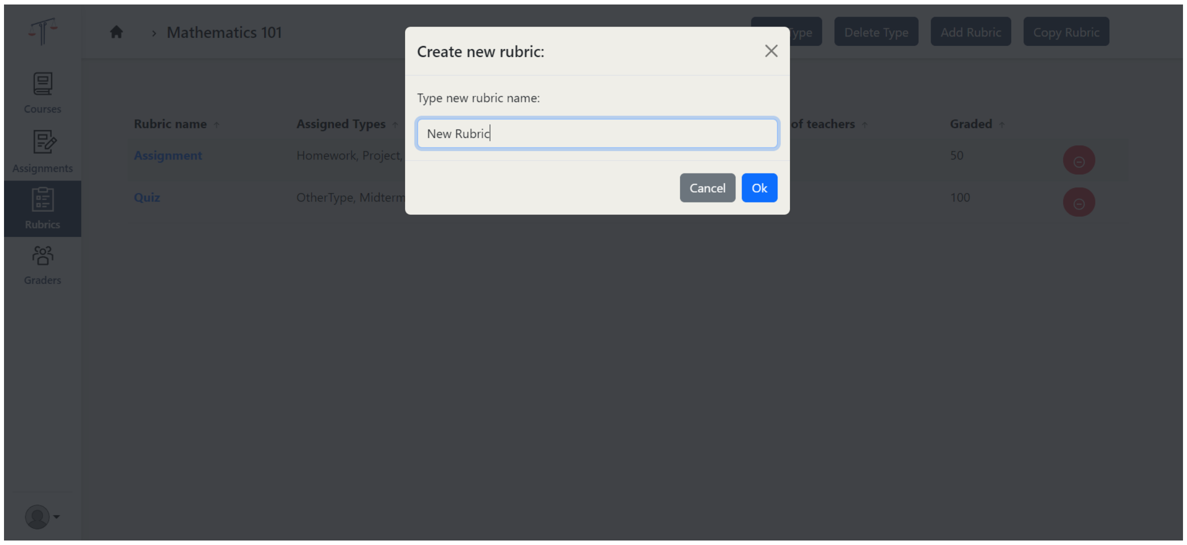


Figure 9.40: Create Rubric

9.7.4 Copy Rubric

You can copy an existing rubric to the current course or a different one 9.41. This feature could be useful when importing existing rubrics from a previous version to a newer version of the same course. To copy a rubric, click on the "Copy Rubric" button. Then select the rubric that is to be copied from the first drop-down list. Next, select the target course from the second drop-down list. Finally, click "Ok".

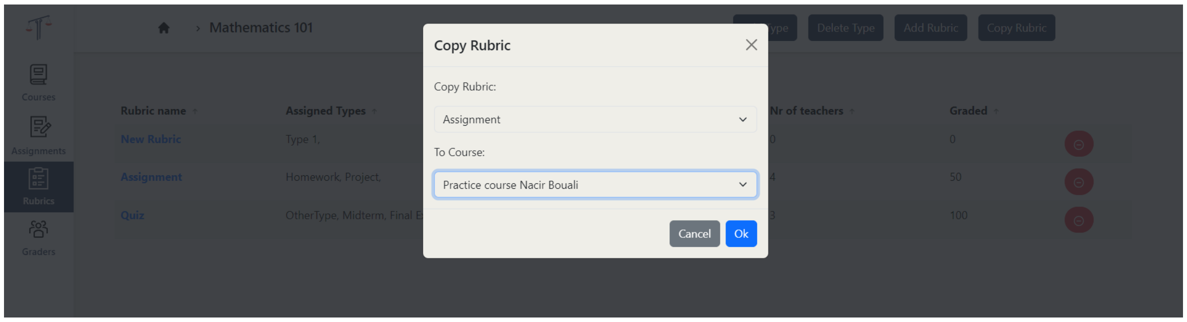


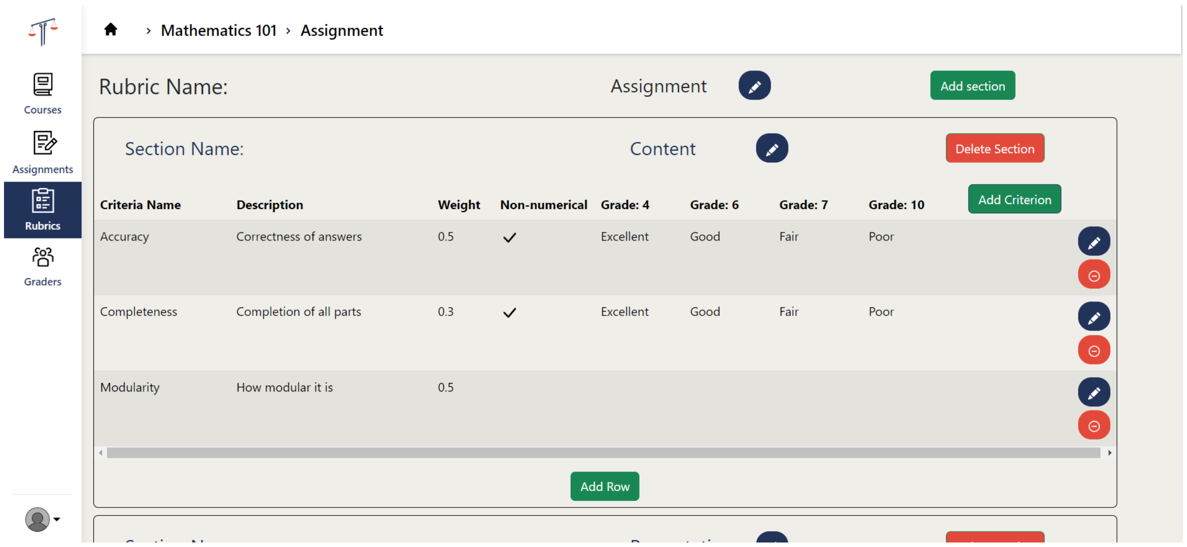
Figure 9.41: Copy Rubric

9.8 Rubric

The Rubric page contains a list of sections, and each section contains a list of criteria 9.42. A criterion can either be Numerical or Non-numerical. The difference is that Non-numerical criteria can only accept a value from the list of performance values. On the other hand, Numerical criteria can accept any grade. Here the performance levels serve as guidance.

9.8.1 Add Section

When you create a new rubric, it has no sections. Therefore, you need to add one by clicking "Add section" 9.43. Type the name and then click "Add".



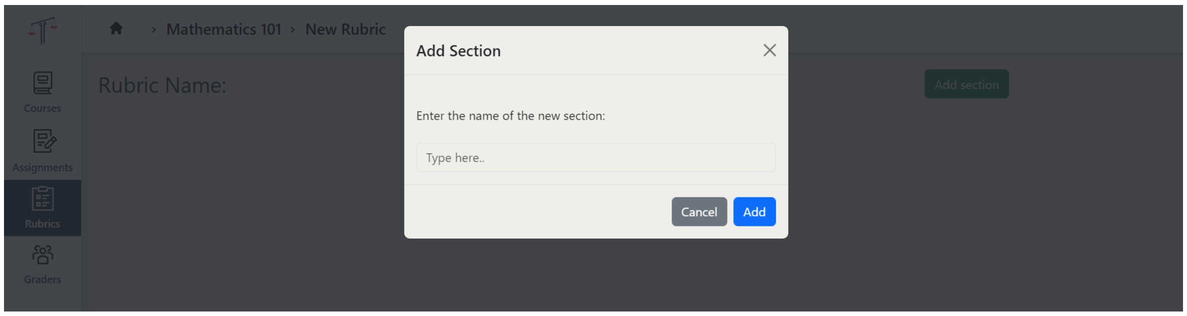
Mathematics 101 › Assignment

Rubric Name: Assignment

Section Name: Content

Criteria Name	Description	Weight	Non-numerical	Grade: 4	Grade: 6	Grade: 7	Grade: 10
Accuracy	Correctness of answers	0.5	✓	Excellent	Good	Fair	Poor
Completeness	Completion of all parts	0.3	✓	Excellent	Good	Fair	Poor
Modularity	How modular it is	0.5					

Figure 9.42: Rubric Page



Mathematics 101 › New Rubric

Rubric Name:

Add Section

Enter the name of the new section:

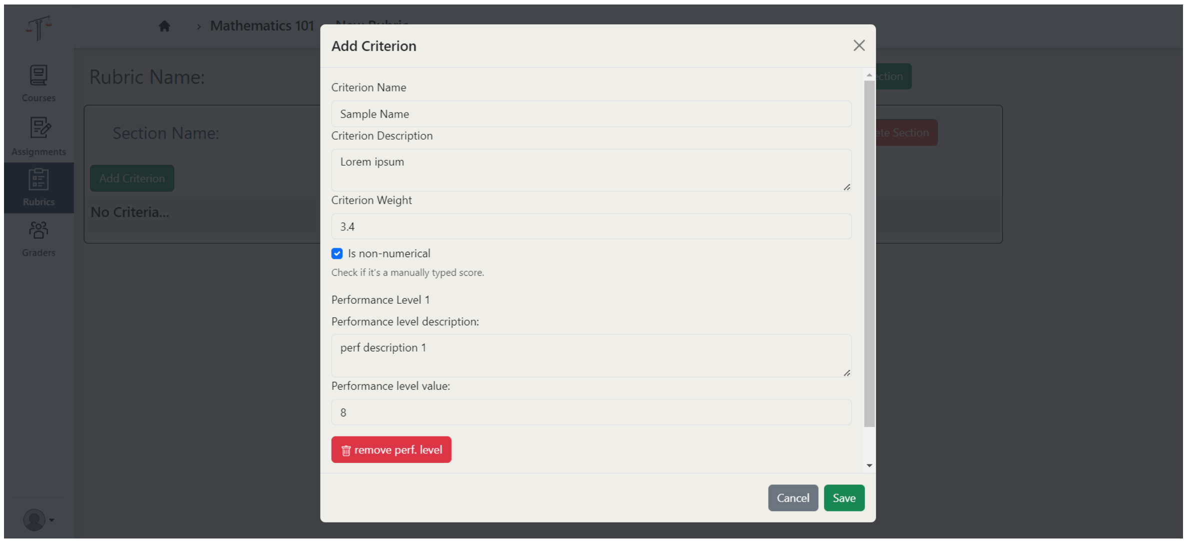
Type here..

Figure 9.43: Add Section

9.8.2 Add Criterion

To add a criterion, click on the "Add criterion" inside a section 9.44. Inside the criterion form, you:

- have to write a name for the criterion
- have the option to enter a description
- have to write the criterion weight
- have to select if the criterion is Non-numerical (performance level-based grade)
- have to add performance levels (performance levels are optional for Numerical criteria):
  - You can add a description to the performance level
  - You have to add the value for the performance level



Criterion Name: Sample Name
Criterion Description: Lorem ipsum
Criterion Weight: 3.4
☑ Is non-numerical
Check if it's a manually typed score.
Performance Level 1
Performance level description: perf description 1
Performance level value: 8

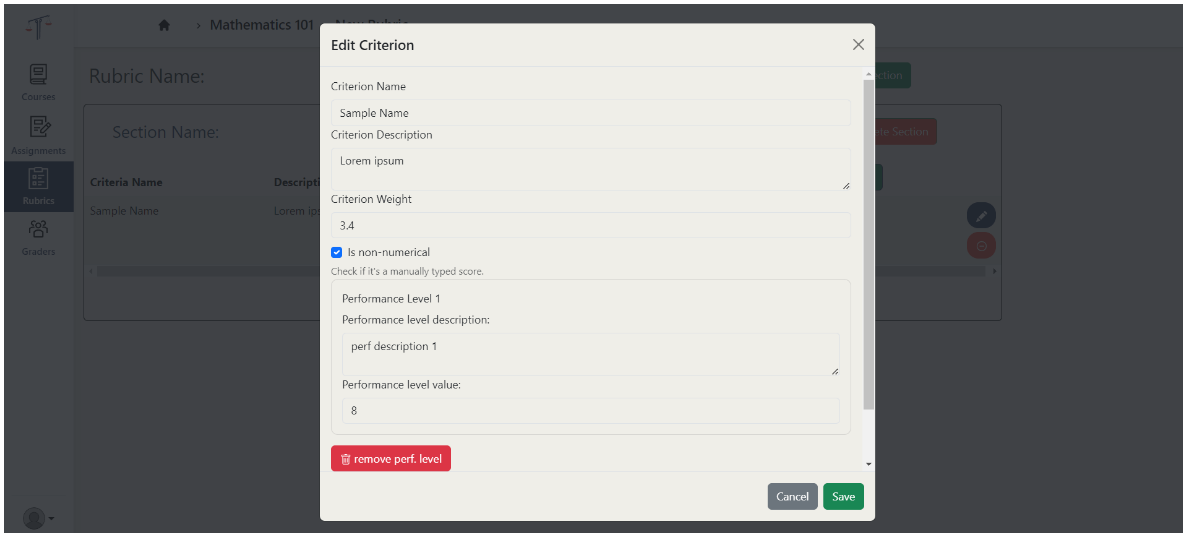
Figure 9.44: Add Criterion

9.8.3 Edit Criterion

You can also edit an already created criterion by clicking on the edit icon button 9.45. Then, the criterion form will be filled with existing data. The validation rules for creating criteria also apply here. Click "Save" to save changes.

9.9 Graders Page

If you are logged in as an coordinator (Figure 9.46) you can see all the people that were imported for a course along with their university ID, assigned role, and total number of assigned gradings. Coordinator can set the graders' grading preferences, and their grading capacity. As a coordinator (Figure 9.47), you can also access the information about each person individually as well, by clicking on the teacher name.



Edit Criterion

Criterion Name: Sample Name

Criterion Description: Lorem ipsum

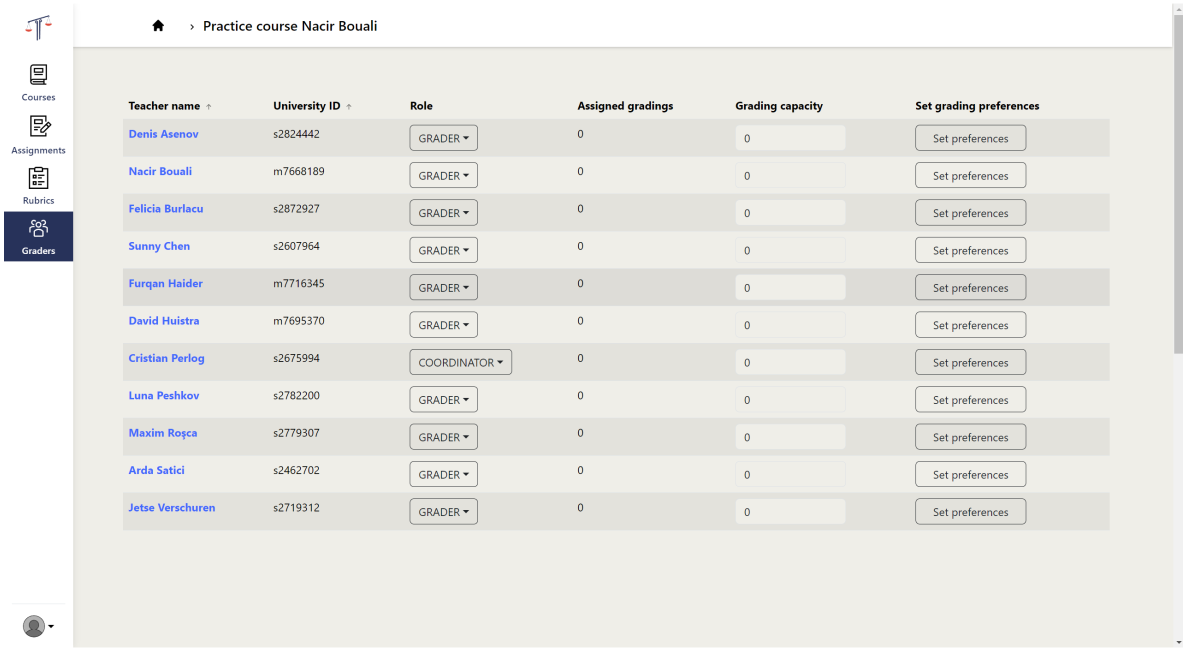
Criterion Weight: 3.4

[✓] Is non-numerical
Check if it's a manually typed score.

Performance Level 1
Performance level description: perf description 1
Performance level value: 8

Criteria Name	Description
Sample Name	Lorem ipsum

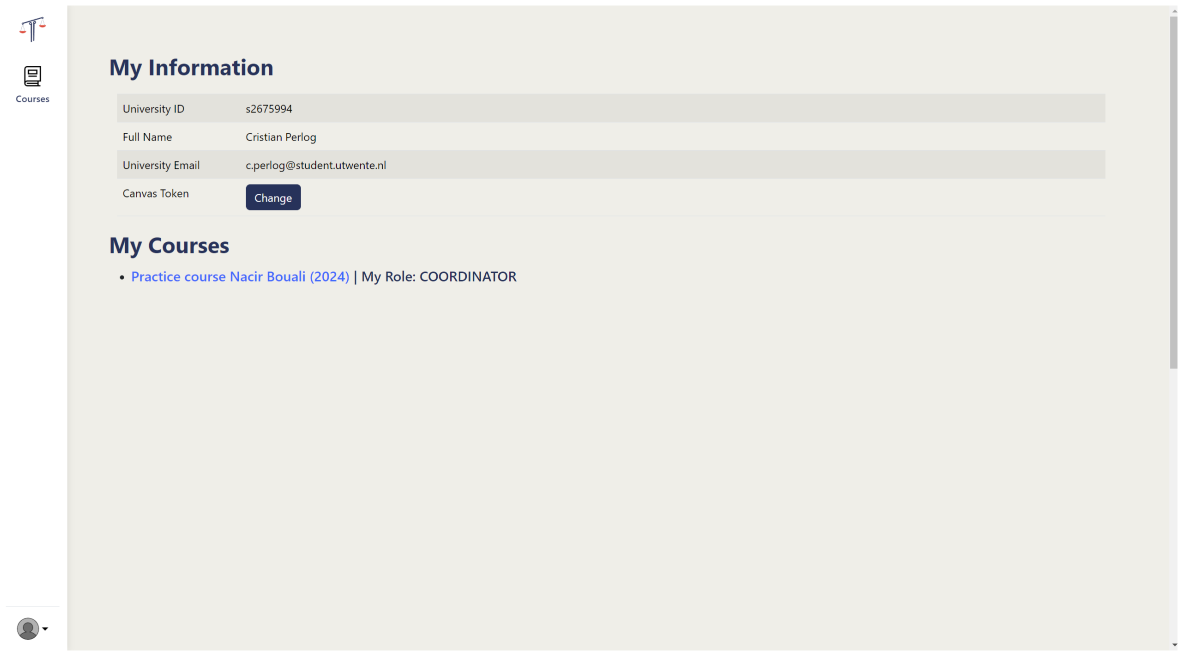
Figure 9.45: Edit Criterion



Teacher name	University ID	Role	Assigned gradings	Grading capacity	Set grading preferences
Denis Asenov	s2824442	GRADER	0	0	Set preferences
Nacir Bouali	m7668189	GRADER	0	0	Set preferences
Felicia Burlacu	s2872927	GRADER	0	0	Set preferences
Sunny Chen	s2607964	GRADER	0	0	Set preferences
Furqan Haider	m7716345	GRADER	0	0	Set preferences
David Huistra	m7695370	GRADER	0	0	Set preferences
Cristian Perlog	s2675994	COORDINATOR	0	0	Set preferences
Luna Peshkov	s2782200	GRADER	0	0	Set preferences
Maxim Roşca	s2779307	GRADER	0	0	Set preferences
Arda Satici	s2462702	GRADER	0	0	Set preferences
Jetse Verschuren	s2719312	GRADER	0	0	Set preferences

Figure 9.46: Coordinator Graders Page

9.10 Person Page



My Information

University ID	s2675994
Full Name	Cristian Perlog
University Email	c.perlog@student.utwente.nl
Canvas Token	Change

My Courses

- Practice course Nacir Bouali (2024) | My Role: COORDINATOR

Figure 9.47: Coordinator Person Page

On this page each user can see his university id, full name, university email and can add or change a canvas token. Below of the personal information all courses to which the user is enrolled are displayed.

10 Reflection

10.1 Future work

The concept of a grading platform can include a vast majority of functionalities. While we focused on the basics during our project, there's room for a lot of useful additional features for graders and coordinators. Such examples include:

- **Rubric history**
  Possibility to see the history of changes for a specific rubric in order to increase accountability.
- **More in-depth grading statistics**
  A dashboard display with extensive statistics about the grading of a current assignment could contain the nr of students that have received a positive grade, the average grade at the moment, etc.
- **To-do lists for graders**
  A potential useful feature for graders is to have a to do list where tasks could be organized in order of urgency, so that after logging in a grader can easily prioritize the work that needs to be done.
- **In-house notifications for rubric and deadline changes**
- **Different fast assignment options**
  Coordinator could choose between a random fast assignment of graders based on availability only or a smart algorithm that takes into consideration the topic preferences of each grader.

10.2 Final Remarks

Overall, we had a good development process and managed to create a quality product. We are thankful to our supervisors, David, and Nacir, especially for the support and help provided during the development process. Special thanks to David for helping with some parts of the code, advice on how to implement certain things, fast response to messages, even on weekends, and the amount of feedback given. We worked well as a team, with no conflicts and differences in opinions. If we do the same project again we will start the development a bit earlier, so that we have a week at the end for the report and presentation, and software without some minor bugs and of the best quality.

Backend The backend turned out to be quite qualitative, with good patterns and quality code. We are proud of some of the more non-standard decisions made, such as code reuse in validation or custom annotations for authorization. There are improvements that can be made to the systems and we are aware of. The main reason why these improvements are not implemented is because of lack of time. Sometimes the development was rushed, thus resulting in the fastest-to-implement solution being done. Also, this resulted in the fact that some of the components were tested mostly at the end and bugs were manually detected while doing the front-end development. There are endpoints that might be broken down into multiple smaller endpoints, this is because the endpoint is hard to use, especially for the frontend. Nevertheless, it turned out to be fully functional, easy to maintain, and facilitates the implementation of the aforementioned improvements and we are proud of that.

Frontened The frontend part of the project turned out to be a bit of a challenge, but we think we managed to finish it in a good state. After deciding on choosing the Vue JavaScript framework, we encountered some issues regarding the transition of the design made in BootstrapStudio to the frontend in Vue.js. It was mainly due to the incompatibility between the bootstrap version used in the design (v5, the newest one) and the libraries which provide useful components (such as BootstrapVue). Hence, we ended up using a library with components still in alpha version, BootstrapVueNext, which had its drawbacks. We still think it was the right decision, since it saved quite a bit of time and made the development process easier. For the future, it would be wiser to decide on all the frameworks that we would use before starting any implementation, to avoid and mitigate these kind of challenges. Another point of improvement would be trying to make components less complex, since there a few of them which are maybe more complex than necessary. Overall, it turned out to be a solid implementation, which can become of even higher quality and standards by investing more time and improving the current version.

Stories over-estimation We were quite optimistic when estimating the amount of work needed for each user story. We overestimated at least 1/3 of our stories, especially the ones related to Canvas. This is due to a lack of experience. Also, some stories took more than we expected and some a bit less because of the time when they were implemented. The stories from the beginning took way more time than the ones at the end. Also, some of the possible impediments were not taken into consideration, such as the incompatibility of some libraries.

Gained experience and lessons Some of the technologies used in the project were not fully familiar to us, meaning that another challenge that we had to overcome was understanding and properly using these technologies and frameworks. For the frameworks that we were familiar with already, we still needed to figure out how to do specific tasks or how to implement a certain thing in the best way. We got a lot of feedback from our mentors regarding how to better implement certain stuff and we are thankful for that. There are a lot of things that could've been done in different ways, but if the other way is better or not is highly debatable. At the same time, there are things that, if done again, would certainly be done in another way, or at least another approach would be used. Some of the development process was hindered by some weird errors, some of them taking a day or two to solve. Nevertheless, we now are more familiar with the frameworks that we used, we have a better understanding of how they work and how to use them properly. Also, some of the more annoying issues and errors that occur during the development are already known to us, meaning that in the future, the development process will be more efficient.

A Appendix Mock Up

A.1 Mock Up

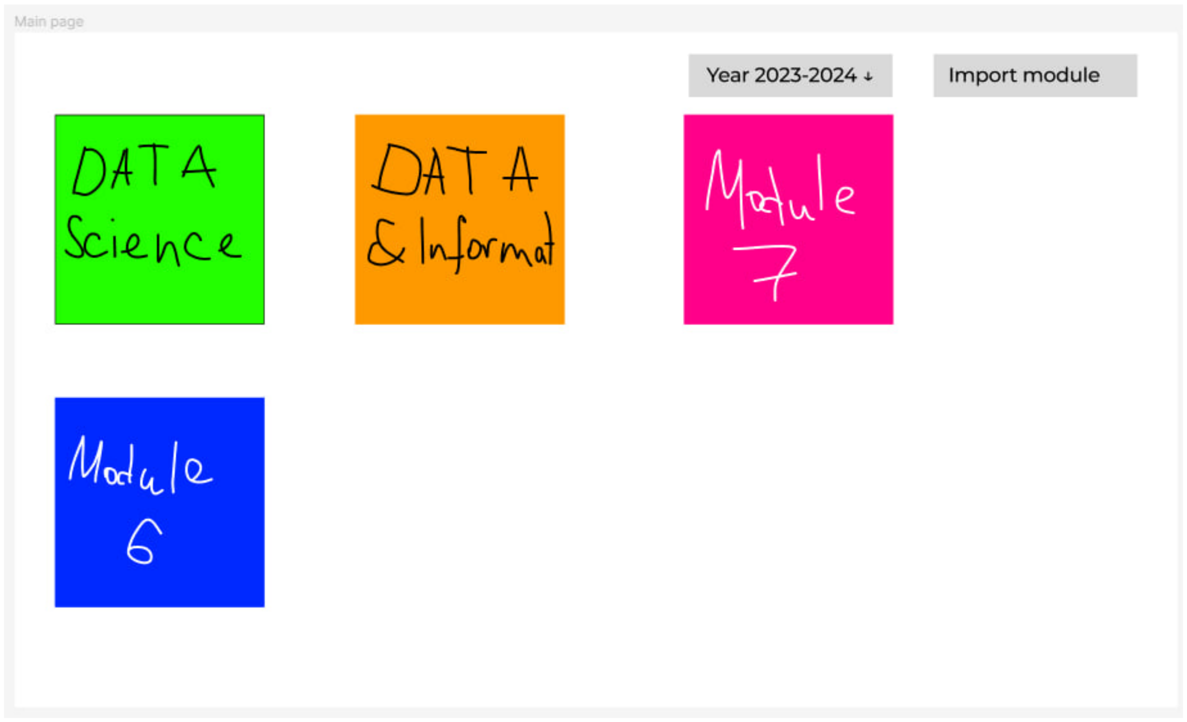
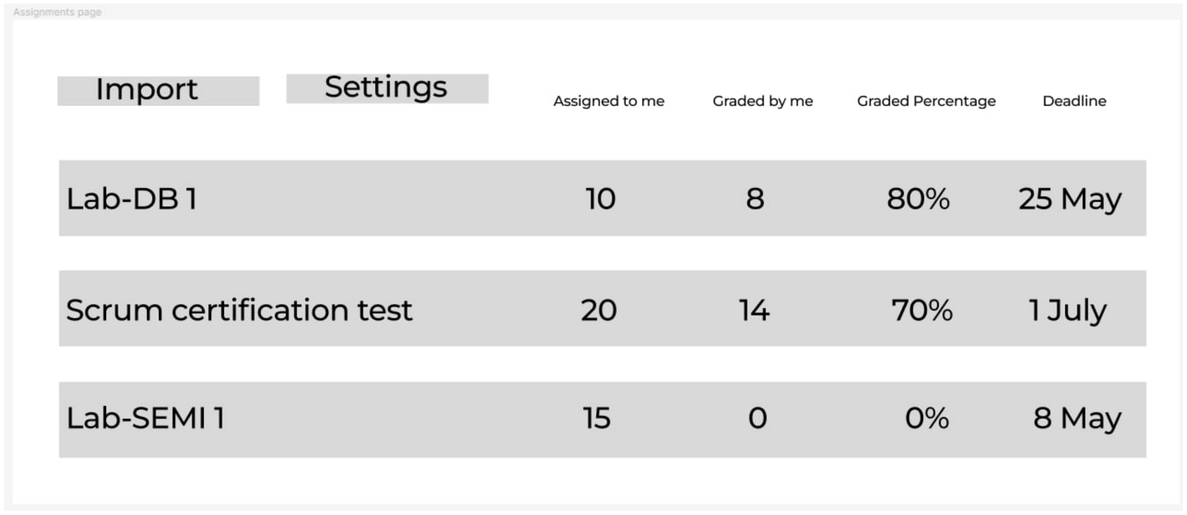
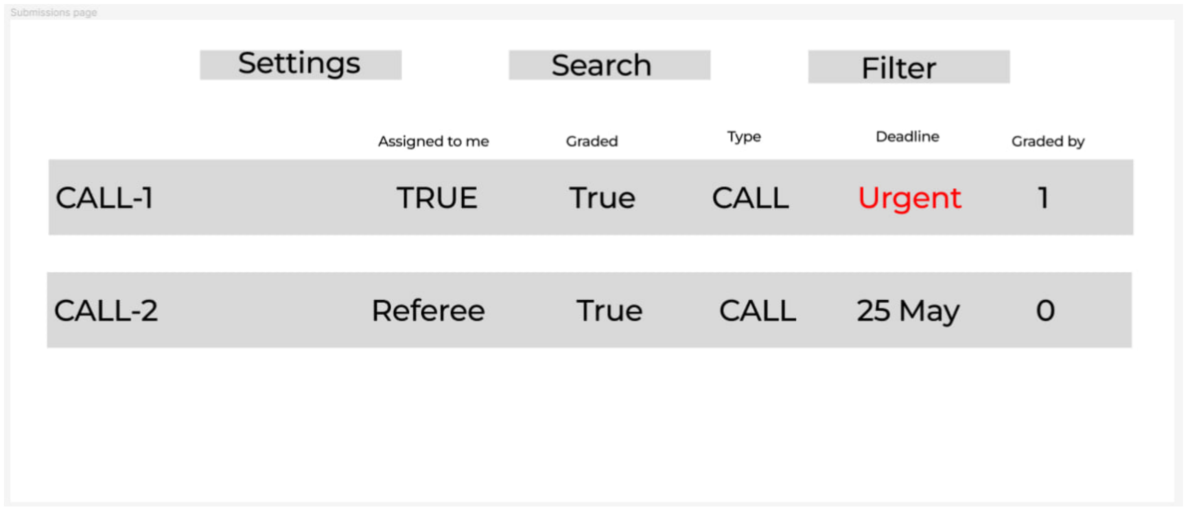


Figure 48: Main page



	Assigned to me	Graded by me	Graded Percentage	Deadline
Lab-DB 1	10	8	80%	25 May
Scrum certification test	20	14	70%	1 July
Lab-SEMI 1	15	0	0%	8 May

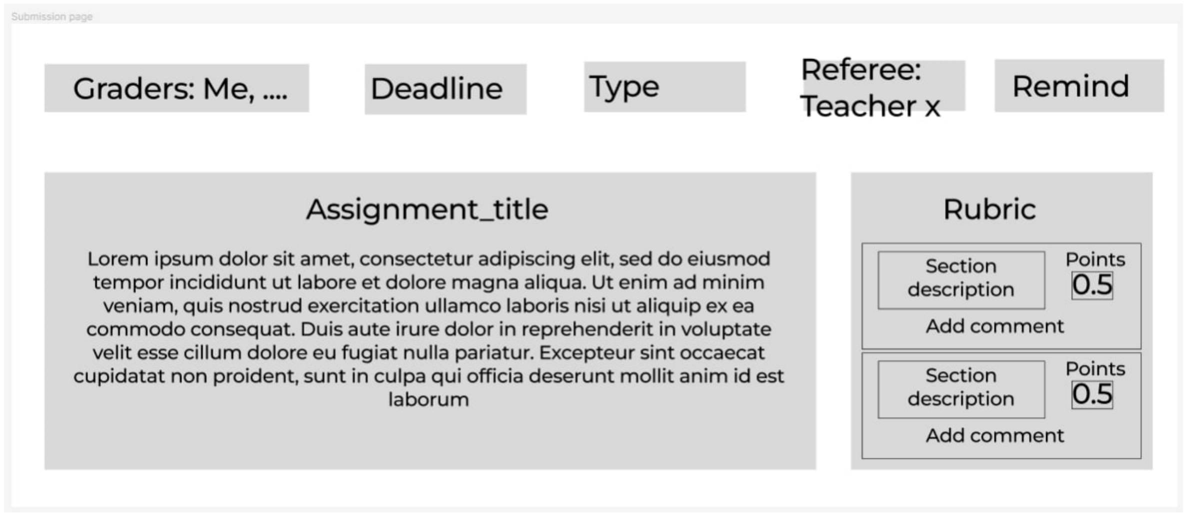
Figure 49: Caption


Submissions page

Settings | Search | Filter

	Assigned to me	Graded	Type	Deadline	Graded by
CALL-1	TRUE	True	CALL	Urgent	1
CALL-2	Referee	True	CALL	25 May	0

Figure 50: Submissions Page


Submission page

Graders: Me, ... | Deadline | Type | Referee: Teacher x | Remind

Assignment_title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Rubric

Section description | Points 0.5 | Add comment

Section description | Points 0.5 | Add comment

Figure 51: Submission Page

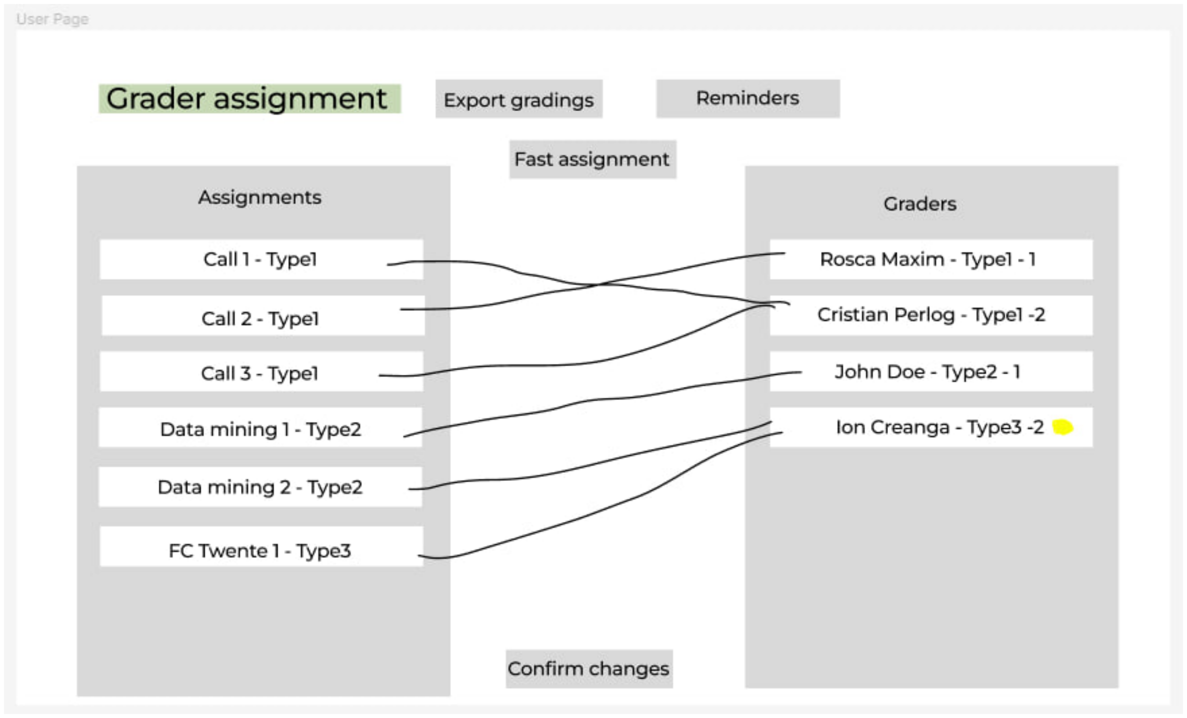
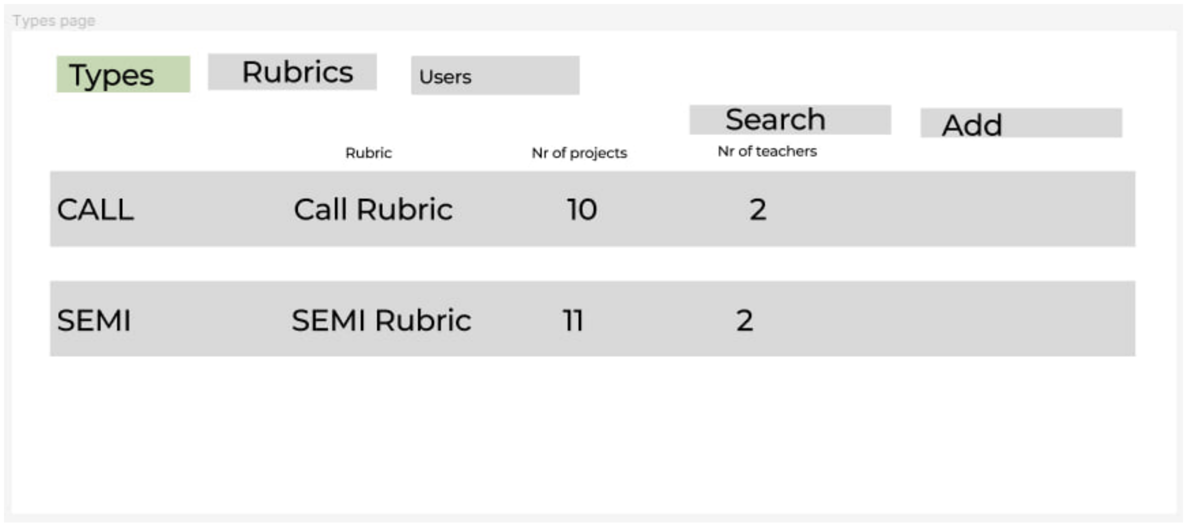


Figure 52: Grader Assignment Page



	Rubric	Nr of projects	Nr of teachers
CALL	Call Rubric	10	2
SEMI	SEMI Rubric	11	2

Figure 53: Types Page

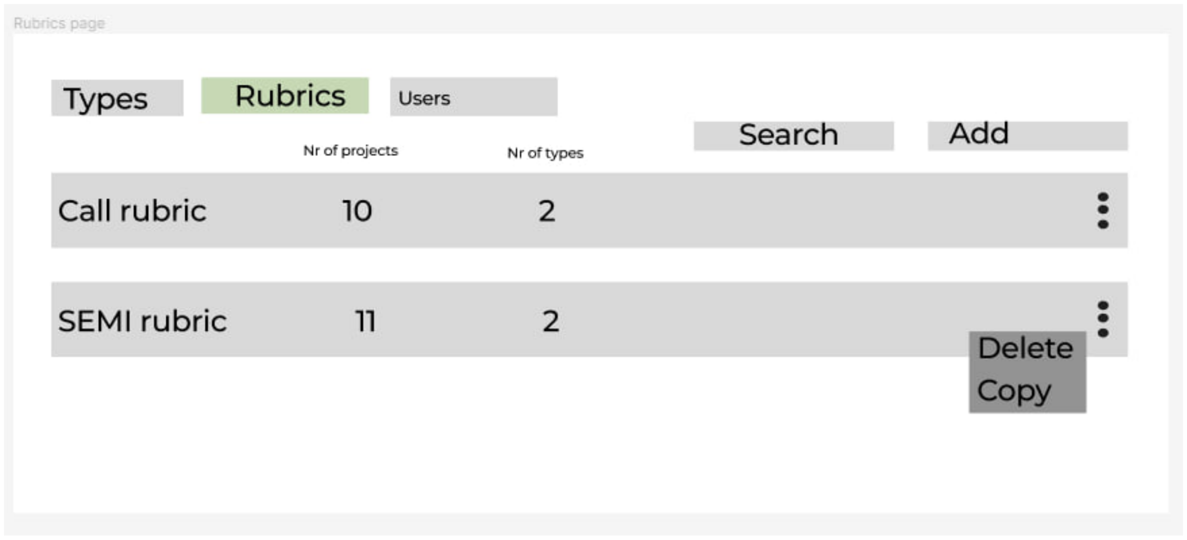


Figure 54: Rubrics Page

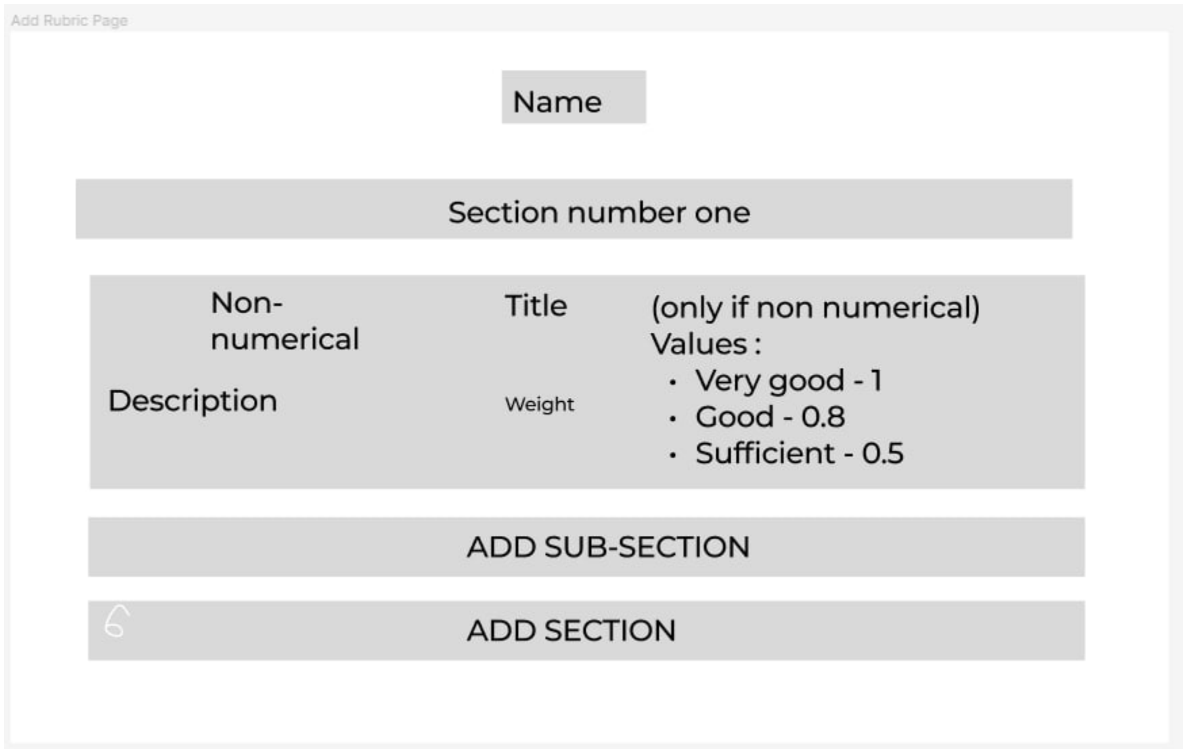
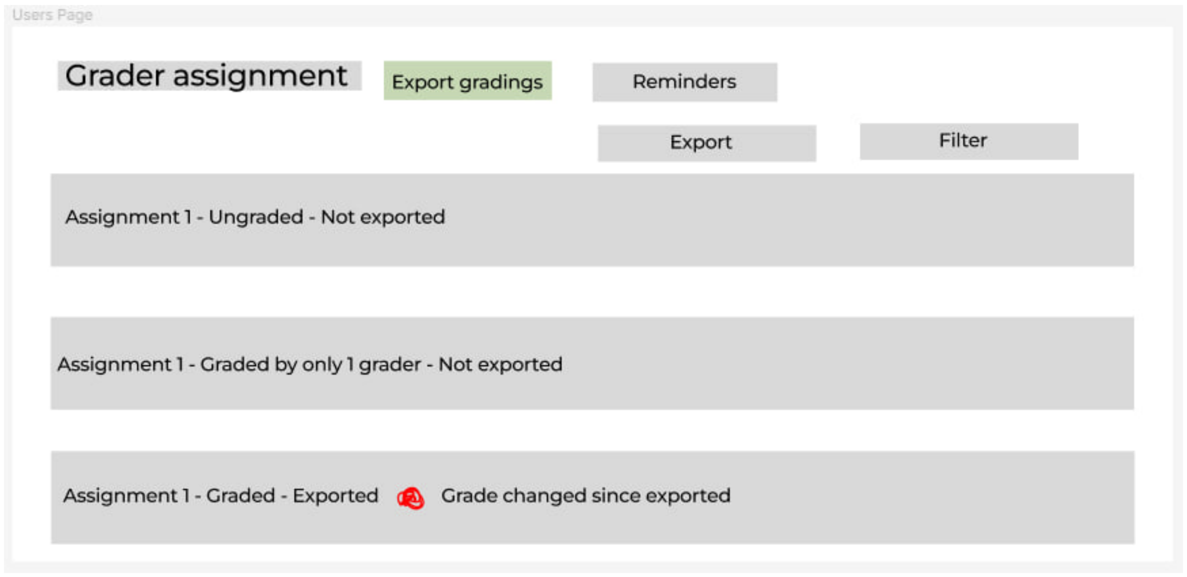


Figure 55: New Rubric Page

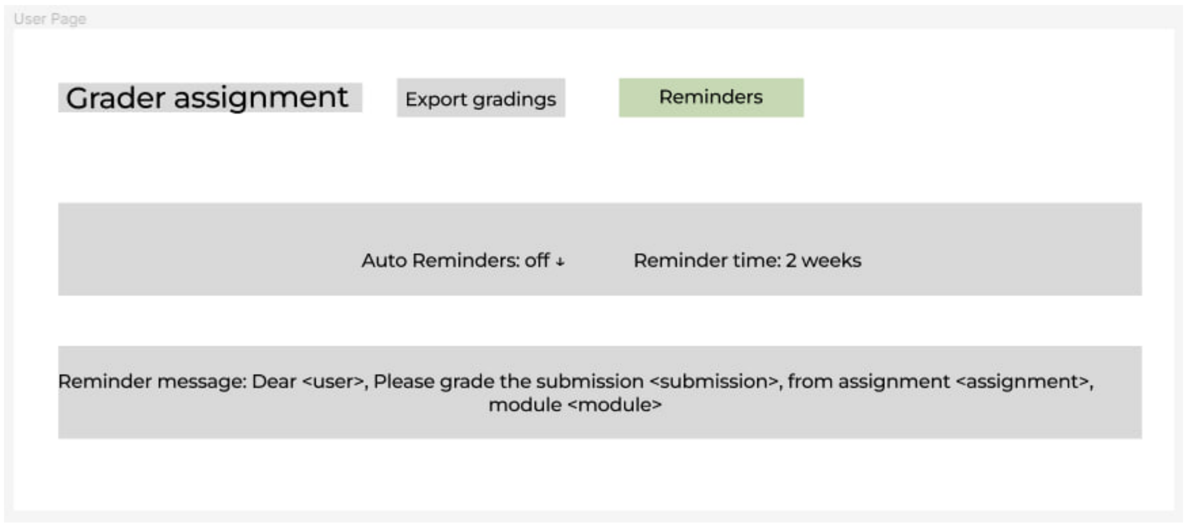


Assignment 1 - Ungraded - Not exported

Assignment 1 - Graded by only 1 grader - Not exported

Assignment 1 - Graded - Exported Grade changed since exported

Figure 56: Export Grades Page



Auto Reminders: off ↓ Reminder time: 2 weeks

Reminder message: Dear <user>, Please grade the submission <submission>, from assignment <assignment>, module <module>

Figure 57: Reminders Page

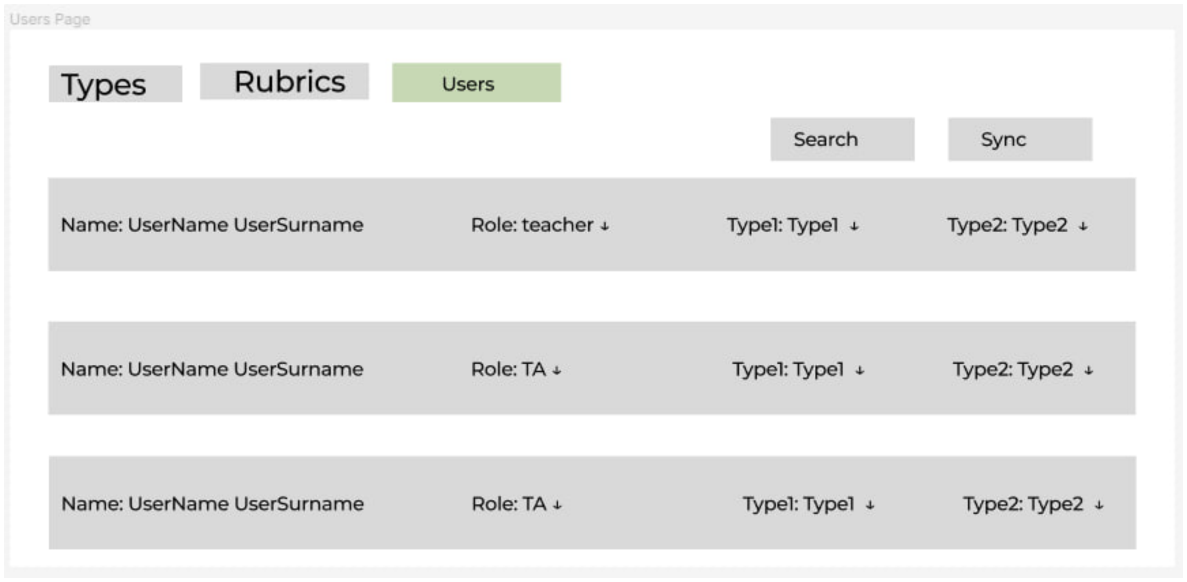


Figure 58: Users Page

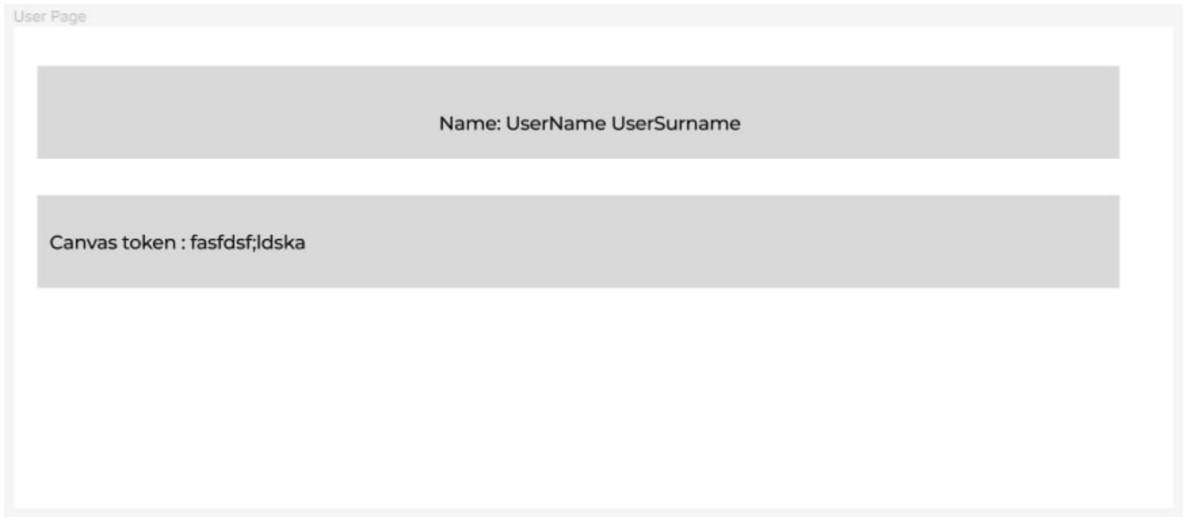


Figure 59: User Page

A.2 UX Design

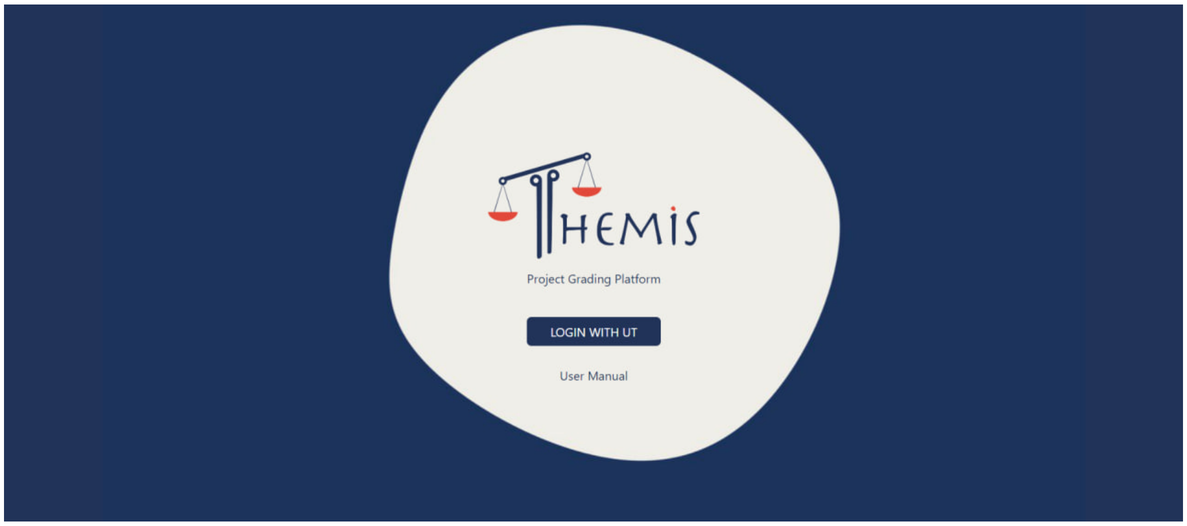


Figure 60: Login Page

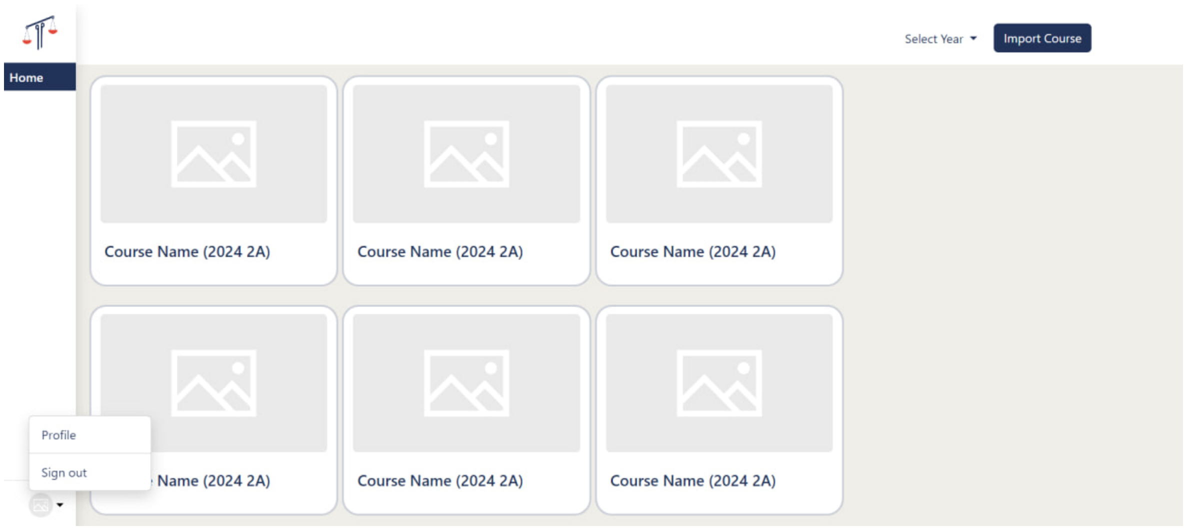
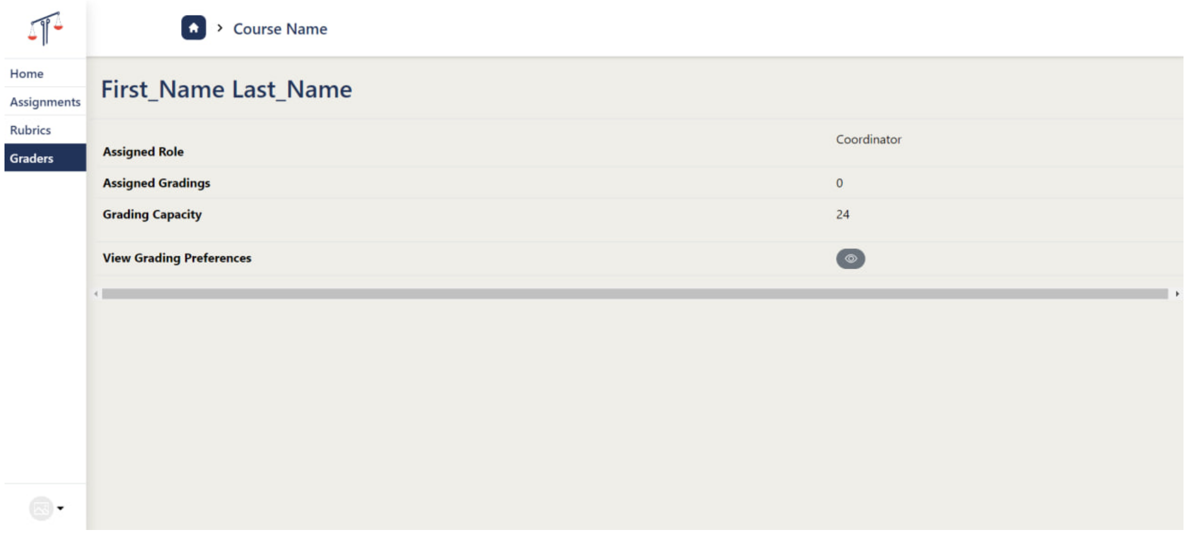


Figure 61: Logout Page



First_Name Last_Name

Assigned Role	Coordinator
Assigned Gradings	0
Grading Capacity	24
View Grading Preferences	

Figure 62: Profile Page

A.2.1 Admin View

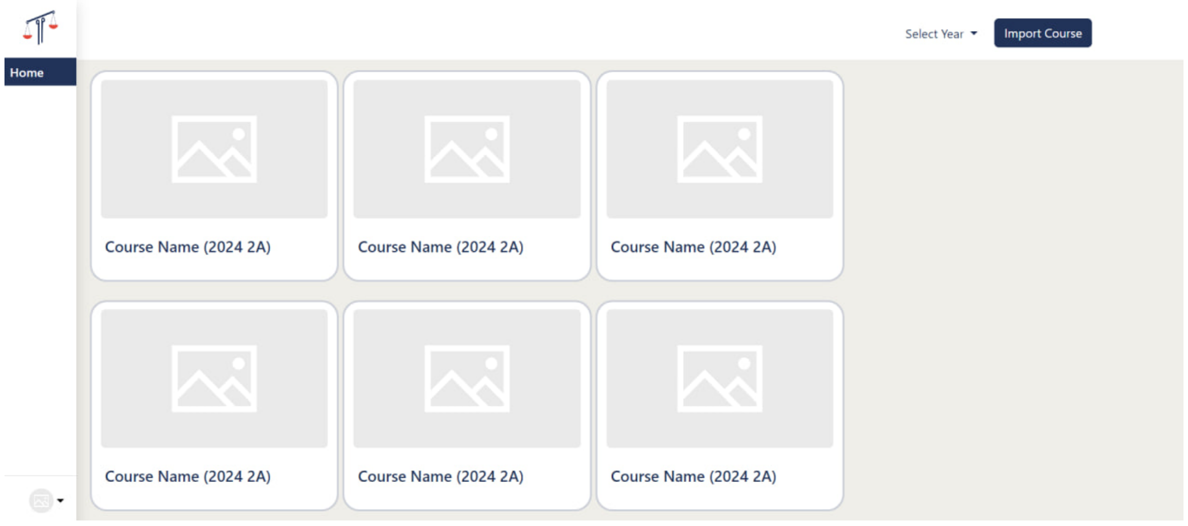
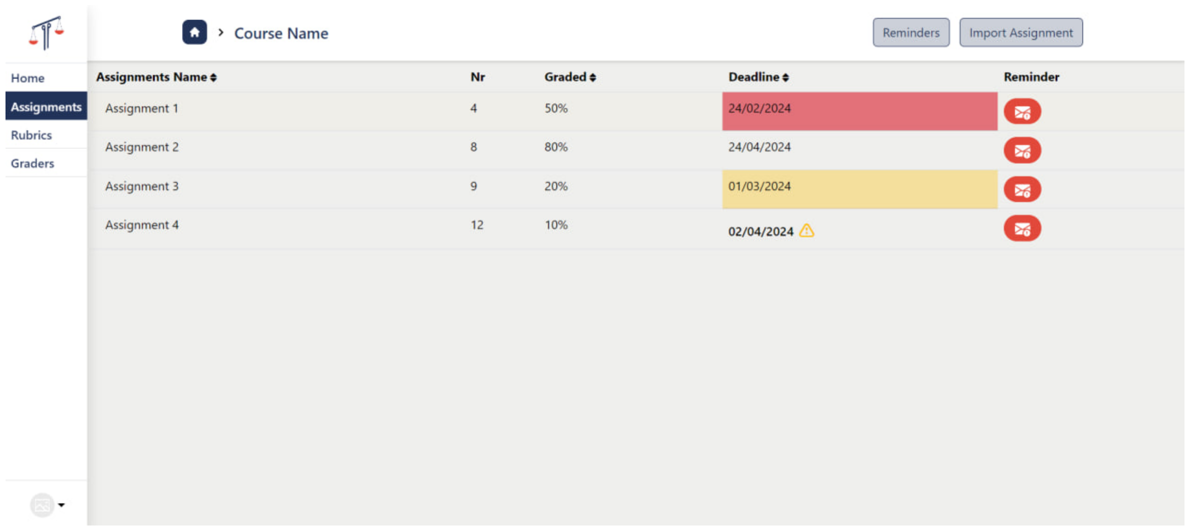
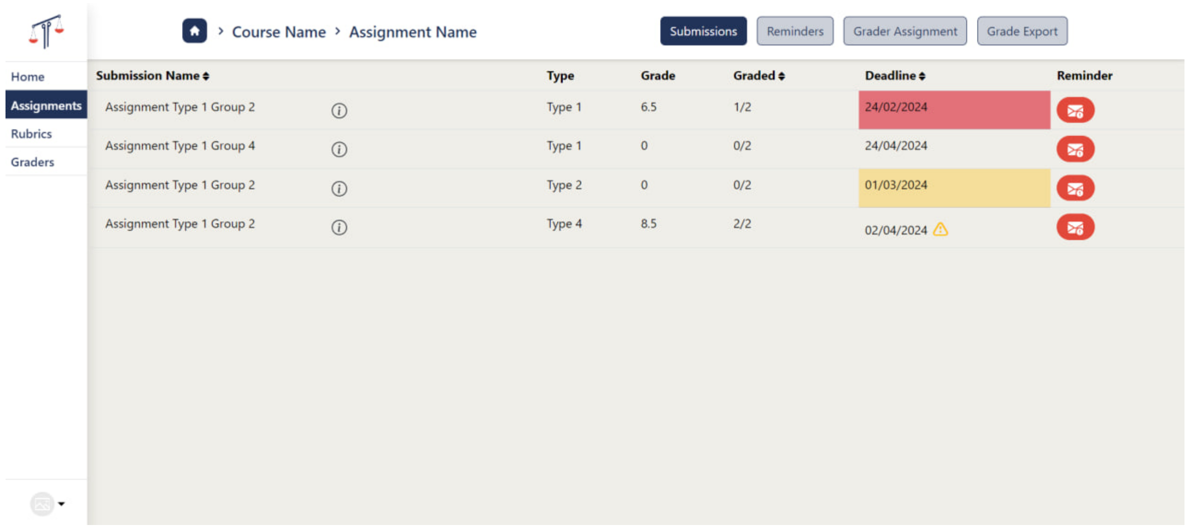


Figure 63: Admin Main Page



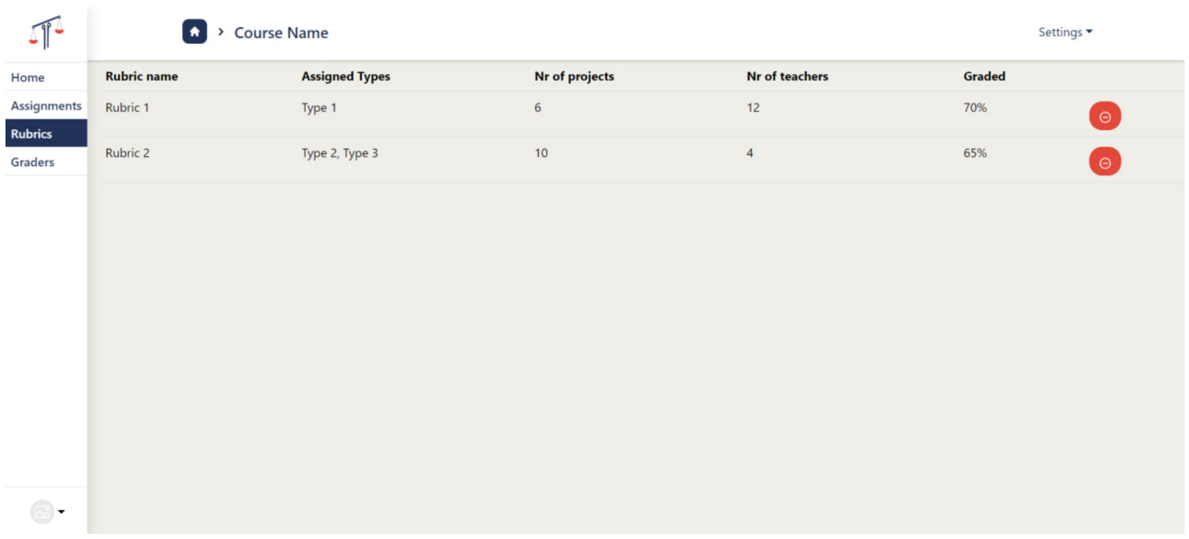
Assignments Name	Nr	Graded	Deadline	Reminder
Assignment 1	4	50%	24/02/2024	
Assignment 2	8	80%	24/04/2024	
Assignment 3	9	20%	01/03/2024	
Assignment 4	12	10%	02/04/2024 ⚠	

Figure 64: Admin Course Main Page



Submission Name		Type	Grade	Graded	Deadline	Reminder
Assignment Type 1 Group 2	ⓘ	Type 1	6.5	1/2	24/02/2024	
Assignment Type 1 Group 4	ⓘ	Type 1	0	0/2	24/04/2024	
Assignment Type 1 Group 2	ⓘ	Type 2	0	0/2	01/03/2024	
Assignment Type 1 Group 2	ⓘ	Type 4	8.5	2/2	02/04/2024 ⚠	

Figure 65: Admin Submissions Page



Rubric name	Assigned Types	Nr of projects	Nr of teachers	Graded
Rubric 1	Type 1	6	12	70%
Rubric 2	Type 2, Type 3	10	4	65%

Figure 66: Admin Rubrics Page

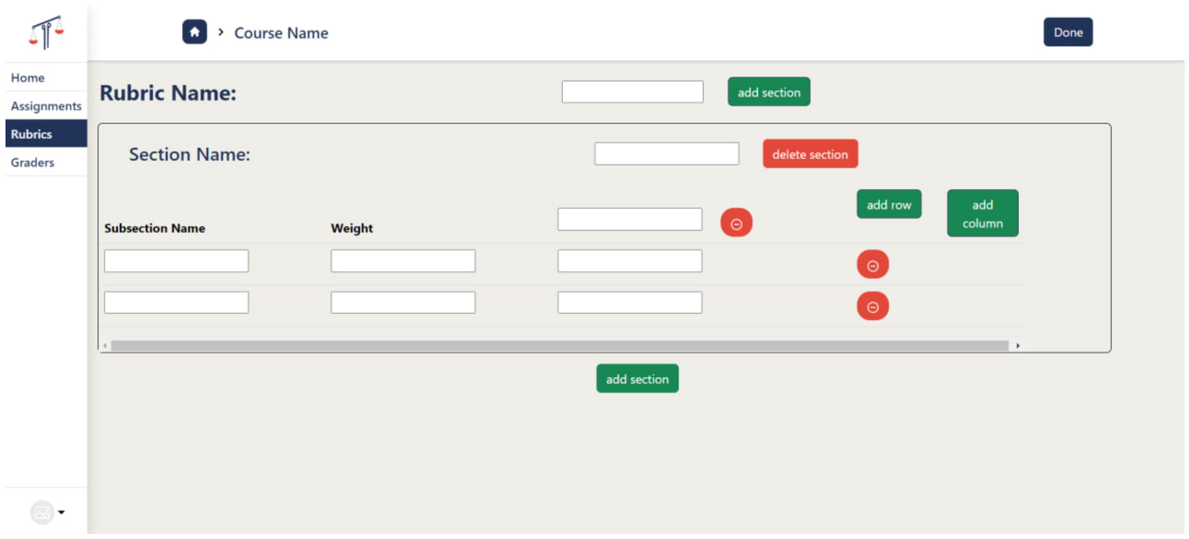
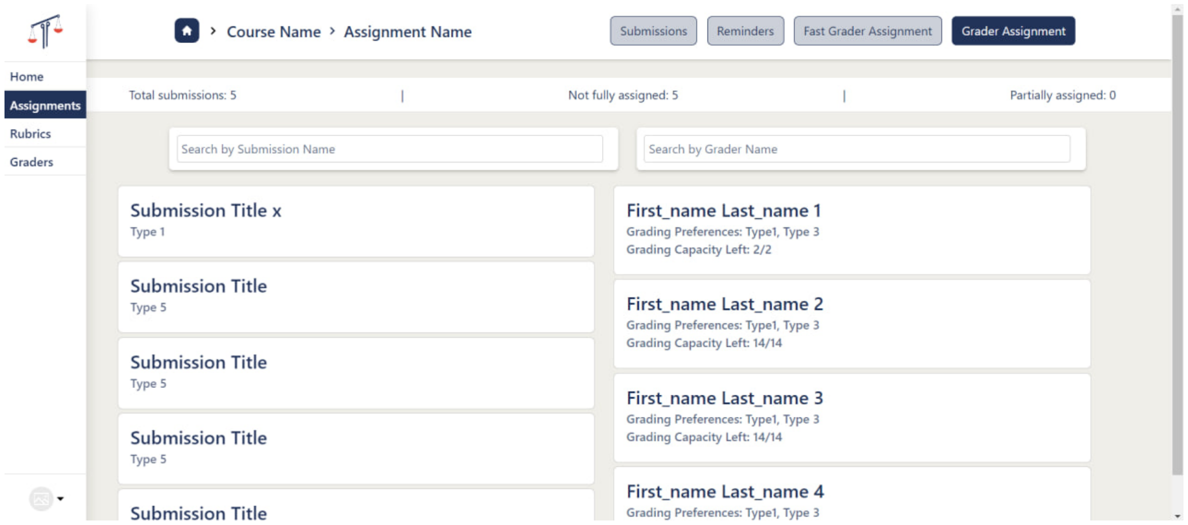


Figure 67: Admin New Rubric Page



Total submissions: 5 | Not fully assigned: 5 | Partially assigned: 0

Search by Submission Name

Search by Grader Name

Submission Title x
Type 1

Submission Title
Type 5

Submission Title
Type 5

Submission Title
Type 5

Submission Title

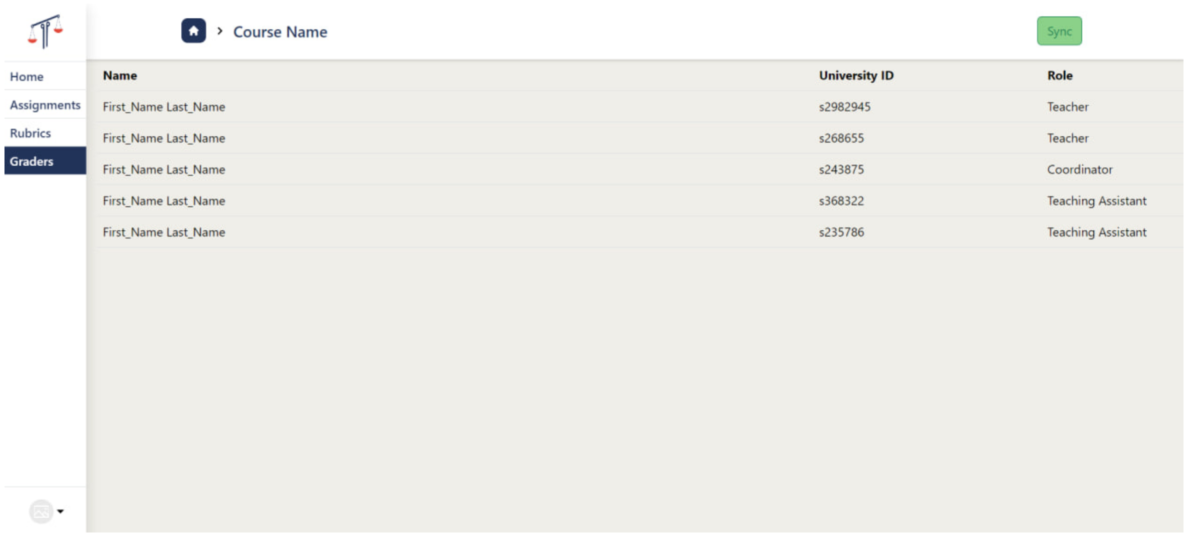
First_name Last_name 1
Grading Preferences: Type1, Type 3
Grading Capacity Left: 2/2

First_name Last_name 2
Grading Preferences: Type1, Type 3
Grading Capacity Left: 14/14

First_name Last_name 3
Grading Preferences: Type1, Type 3
Grading Capacity Left: 14/14

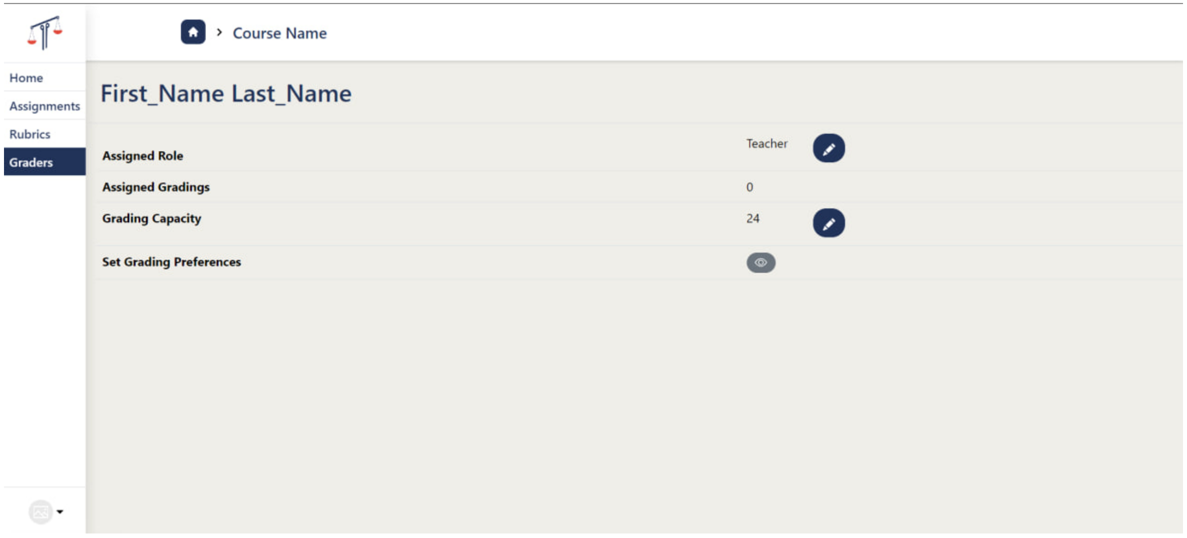
First_name Last_name 4
Grading Preferences: Type1, Type 3

Figure 68: Admin Grader Assignment Page



Name	University ID	Role
First_Name Last_Name	s2982945	Teacher
First_Name Last_Name	s268655	Teacher
First_Name Last_Name	s243875	Coordinator
First_Name Last_Name	s368322	Teaching Assistant
First_Name Last_Name	s235786	Teaching Assistant

Figure 69: Admin Graders Page


Course Name

First_Name Last_Name

Assigned Role	Teacher	
Assigned Gradings	0	
Grading Capacity	24	
Set Grading Preferences		

Figure 70: Admin Grader Page

A.2.2 Grader View

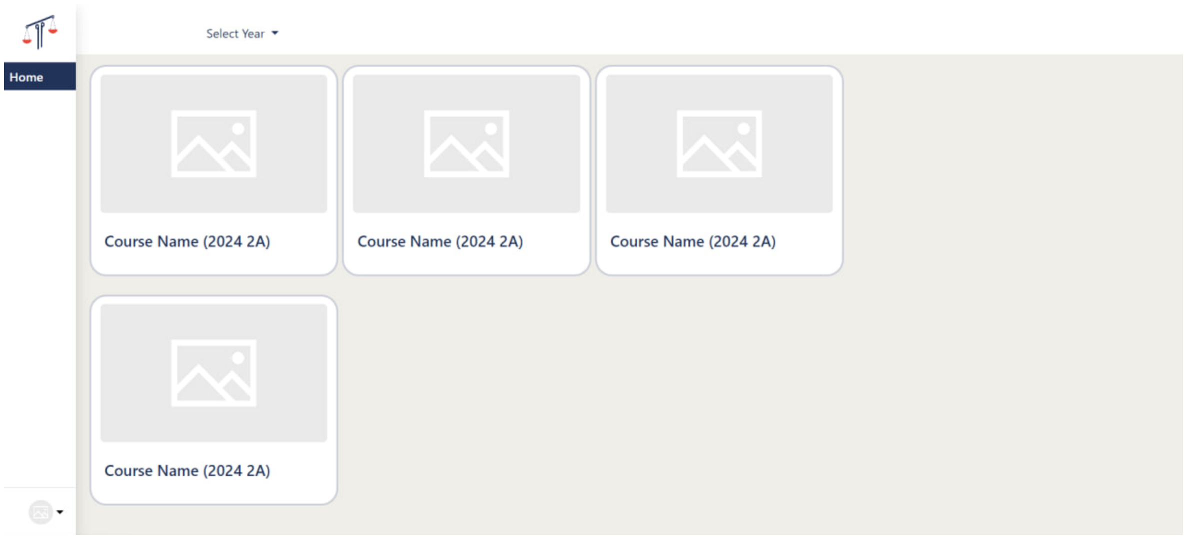
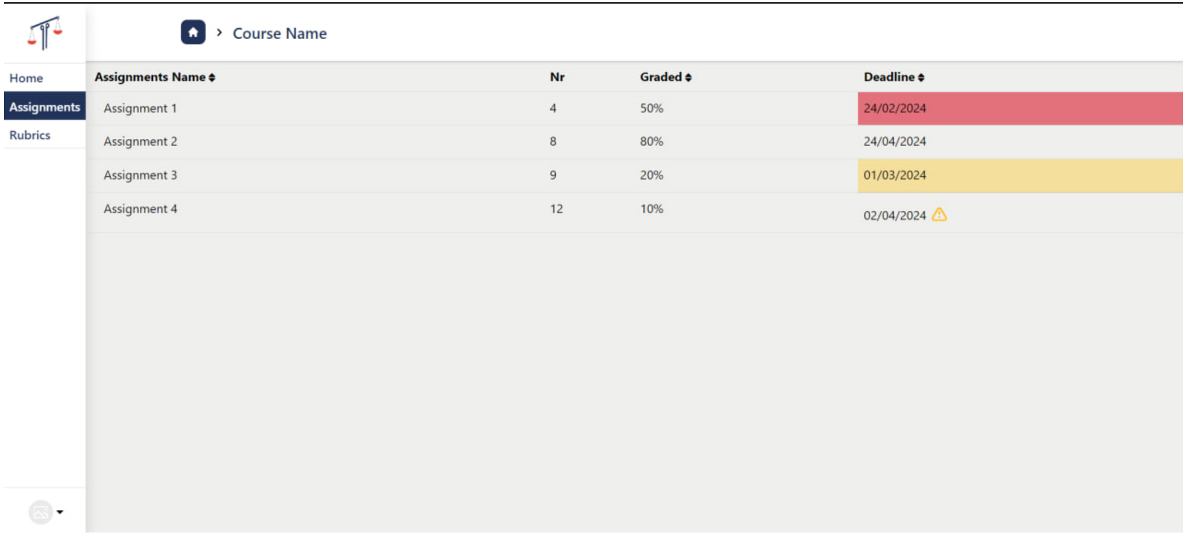


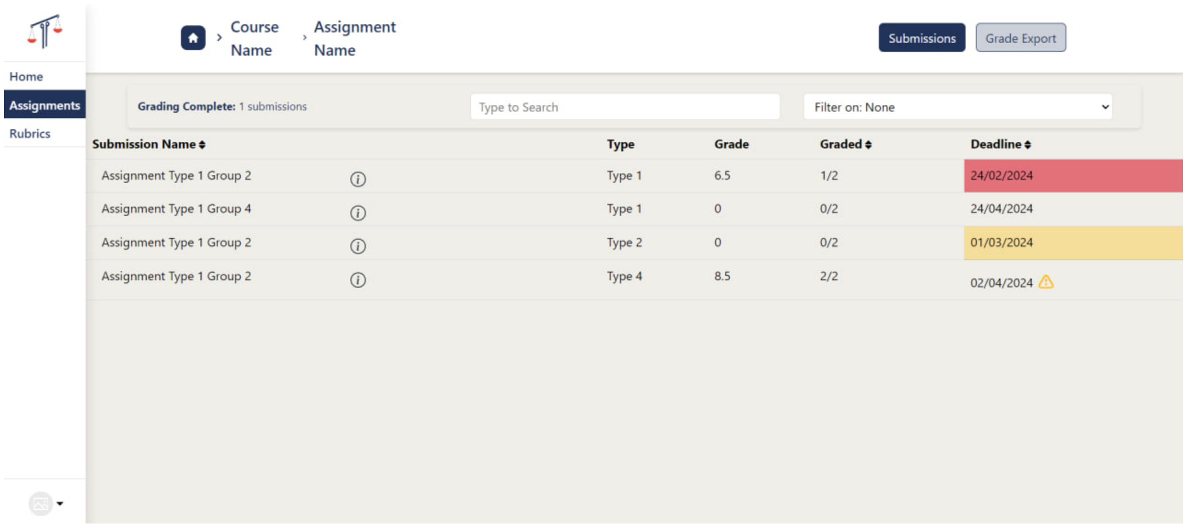
Figure 71: Grader Main Page



Course Name

Assignments Name	Nr	Graded	Deadline
Assignment 1	4	50%	24/02/2024
Assignment 2	8	80%	24/04/2024
Assignment 3	9	20%	01/03/2024
Assignment 4	12	10%	02/04/2024 ⚠

Figure 72: Grader Course Main Page

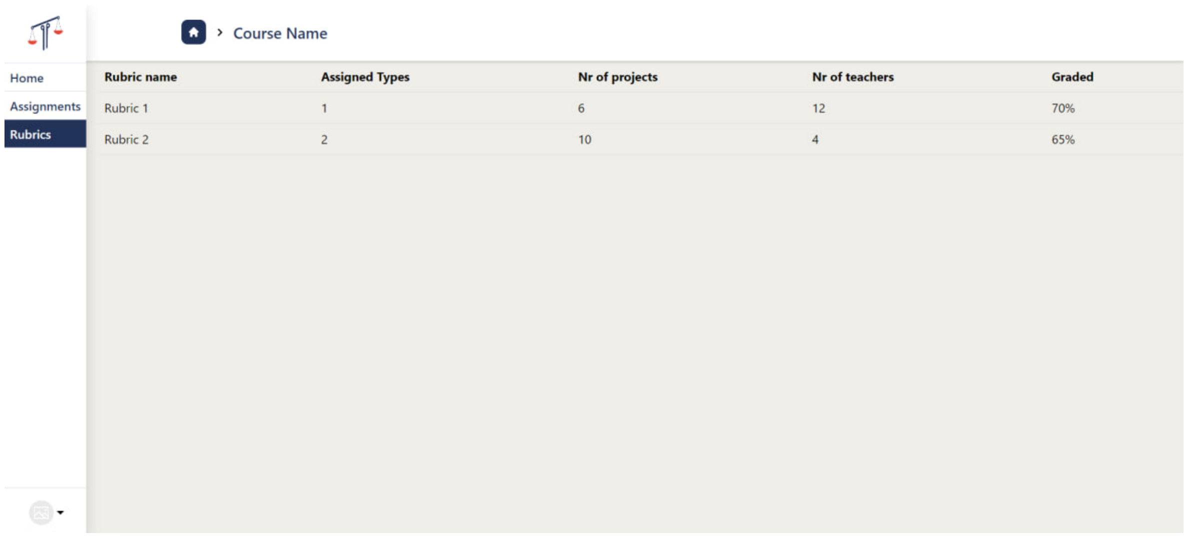


Grading Complete: 1 submissions

Filter on: None

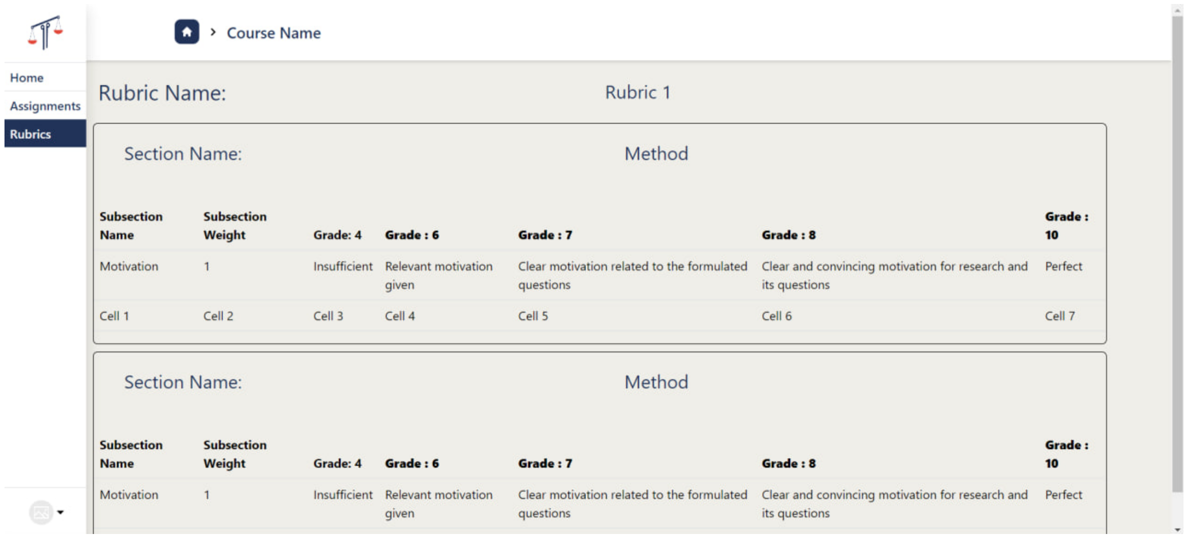
Submission Name	Type	Grade	Graded	Deadline
Assignment Type 1 Group 2	Type 1	6.5	1/2	24/02/2024
Assignment Type 1 Group 4	Type 1	0	0/2	24/04/2024
Assignment Type 1 Group 2	Type 2	0	0/2	01/03/2024
Assignment Type 1 Group 2	Type 4	8.5	2/2	02/04/2024 ⚠

Figure 73: Grader Submissions Page



Rubric name	Assigned Types	Nr of projects	Nr of teachers	Graded
Rubric 1	1	6	12	70%
Rubric 2	2	10	4	65%

Figure 74: Grader Rubrics Page



Rubric Name: Rubric 1

Section Name: Method

Subsection Name	Subsection Weight	Grade: 4	Grade : 6	Grade : 7	Grade : 8	Grade : 10
Motivation	1	Insufficient	Relevant motivation given	Clear motivation related to the formulated questions	Clear and convincing motivation for research and its questions	Perfect
Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	Cell 6	Cell 7

Section Name: Method

Subsection Name	Subsection Weight	Grade: 4	Grade : 6	Grade : 7	Grade : 8	Grade : 10
Motivation	1	Insufficient	Relevant motivation given	Clear motivation related to the formulated questions	Clear and convincing motivation for research and its questions	Perfect

Figure 75: Grader Rubric Page

References

[1] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002, ISSN: 1533-5399. DOI: 10.1145/514183.514185. [Online]. Available: https://doi.org/10.1145/514183.514185.

[2] A. Khiyaita, H. E. Bakkali, M. Zbakh, and D. E. Kettani, "Load balancing cloud computing: State of art," in *2012 National Days of Network Security and Systems*, 2012, pp. 106–109. DOI: 10.1109/JNS2.2012.6249253.