People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of the Requirements for the Degree of

MASTER

In **Electronics**

Option: **Computer Engineering**

Title:

Design and Implementation of a Web-based Application to Manage Students' Grades

Presented By:

BENDERRADJI Farouk Mounsef

Supervisor:

Dr. ZITOUNI A.

Registration Number:……../2022

Abstract

The aim of this project is to create an interactive web application dedicated to the management of students' evaluation in our institute. This web application makes it possible to facilitate the work of the members of the administration and teachers by automating the management of students in promotions and groups, assigning teachers to courses, grades submission, semestrial and annual results, and the users' roles.

To improve the students' evaluation process, this project goal is to analyze the current processes and situations in students' evaluation and determine the possible areas of improvement. This work has been carried out using the modelling language, Unified Modelling Language (UML), for designing the application. As for the implementation, we have chosen to develop the application with NodeJS, MongoDB as the Database Management System (DBMS), and ReactJs for creating user interfaces.

Dedication

I dedicate this work to my beloved parents, my great sister and my great brother, to all my friends, and to all people who were there for me with help, advice and best wishes.
Thank you all for your support.

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Zitouni for his guidance through the process of realizing this project, his advice were so valuable, which made my experience more beneficial. Secondly, I would like to thank Dr. Otmani for the help he provided during the realization of the project.

Finally, my deep thanks and gratitude to all who supported me through my educational journey in the institute of electrical and electronics engineering of the University M'Hamed BOUGARA – Boumerdes.

TABLE OF CONTENTS

List of tables

List of Figures

List of abbreviations:

- SPA: Single Page Application.
- DBMS: Database Management System.
- UML: Unified Modelling Language.
- HTML: Hypertext Mark-up Language.
- CSS: Cascading Style Sheets.
- MySQL: My Structured Query Language.
- SQL: Structured Query Language.
- JSON: Javascript Object Notation.
- BSON: Binary JSON.

General Introduction:

The Internet has taken the world of computers and communications to an extraordinary stage. The invention of phones, radios, and computers opened the way for this special integration of capabilities.

Until now the computers remain the safest way to process and backup information. This feature has made it possible to computerize the data management system of companies, which is an essential part of their development today.

A web application is an application that is accessed by users over a network. Users can easily access the application from any computer connected to the Internet using a standard browser. In another term it is a software system that provides a user interface through a web browser. A web application plays a very important role in every field such as in the field of education, health, and libraries. It can give ease in every field because a web application also saves the time of the clients. Over all, a web application is very useful and convenient for clients when internet is in everyone's reach.

This project aims to design and implement an interactive, reliable, user-friendly web application to facilitate and enhance the process of students' evaluation and grading in our institute.

This report explains the process in which a carefully implemented web application can help to simplify the work of the administration members dealing with students' evaluation. In addition, it will make sure that all data and information are well maintained, as well as the precision of all generated results and averages.

Our report is organized in four main chapters:

The First chapter states the current existing solutions, defines the different objectives of this project and gives a brief overview of web applications. **The second chapter** introduces the tools and technologies used in the project. **The third chapter** concerns the design. It brings together the stages of our process of development using UML modeling language. **The fourth chapter** is devoted to the implementation of the project by introducing the different user interfaces of our application's front-end. Our report ends with a **general conclusion**.

CHAPTER ONE

An overview of project objectives and web applications.

1.1 Introduction:

In this chapter, we give a brief general presentation of our project. We discuss the main struggles in student evaluation process faced by our institute, also we speak about the objectives and motives behind establishing this work which are mainly to offer better tools to enhance the process of students grading and generating all the results in a more effective way.

1.2 Subject presentation:

Our objective is to design and implement an efficient web application to manage the evaluation process of under-graduate and graduate students in our institute for each promotion, including grades submission, averages calculation, results generation…. This work aims to solve some of the existing problems and make the evaluation process more efficient and easier for the administration members responsible for it.

1.2.1 Existing solutions:

1.2.1.1 Excel:

Excel is the traditional fashion used for the calculation of students' results based on their grades in different courses, and presenting those final results in deliberation tables, which has been for so long a convenient way for this process. However, it takes a considerable amount of time to configure it to do the desired work. And of course, it requires whoever using it to have good knowledge and experience working with it. In addition, when we think about the current days we are living and the technological development that the world achieved, it is time to introduce better, painless, and more effective tools that are specifically designed for the process of students' evaluation.

1.2.1.2 Progres application:

Progres is a web application offered by the minister of higher education to all the universities nationally, intended to automate and enhance the process of students' evaluation. It presents some good ideas to be a better tool for the administration than the traditional excel. However, after discussing with administration members in charge of

managing students' evaluation, it appears that this application has introduced more problems than excel which are as follows:

• A complicated user interface.

• The application becomes too slow when accessing some pages.

• the students result generated contains many calculation mistakes.

1.2.2 Objectives:

Our objective is to give a better solution to all the problems mentioned above, and enhance the process of evaluating students by designing and implementing a fast, scalable, and user-friendly web application that would assure data security, also the precision and correctness of all the generated results.

1.3 Web application overview:

1.3.1 definition:

A web application or "web app" is any software program that runs on a web server. Unlike the traditional desktop applications, which are lunched by the operating system, web apps must be accessed through a web browser. [1]

1.3.2 advantages:

Web apps have several advantages over desktop applications. Since web apps run inside a web browser, no complex installation is needed. Web apps also solve some of the "compatibility issues", (Windows, Mac, Linux); all that is needed is a browser. Developers do not need to distribute software updates to users when the web app is updated. By updating the application on the server, all users have access to the updated version. [1]

1.3.3 Single-Page Application:

A single-page application is an app that works inside a browser and does not require page reloading during use. SPAs are all about serving an outstanding UX(user experience) by trying to imitate a "natural" environment in the browser — no page reloads, no extra wait time. It is just one web page that you visit which then loads all other content using JavaScript — which they heavily depend on. SPA requests the markup and data independently and renders pages straight in the browser.[3]

Single-page sites help keep the user in one, comfortable web space where content is presented to the user in a simple, easy and workable fashion.

1.4 Conclusion:

Through this chapter, we specified the problems that the institute administration faced with the process of evaluating students and we proposed to help solving them using a web application. We also gave a brief overview of web applications and the advantages of using them.

CHAPTER TWO

Tools and technologies.

2.1 Introduction:

With the introduction of many popular tools and technologies, a modern web application has really come a long way over the years leading to develop a fully responsive and dynamic web app. In this chapter we define tools and technologies that we are going to use in our project.

2.2 Development tools:

2.2.1 Visual Studio Code:

Visual Studio Code is a source editor developed by Microsoft for Windows, Linux and MacOS. Visual Studio Code has support for almost every major programming language. Several are included by default, for example, JavaScript, TypeScript, CSS, and HTML but other language extensions can be found and downloaded for free from the VS Code Marketplace.

2.2.2 NodeJS:

NodeJS is an open-source development platform used for developing a server-based application. The traditional JavaScript environment has always been a client-side in a user's browser or in an application that is talking to a server. JavaScript has not been considered when it comes to the server responding to client requests, but that is exactly what node.js provides.

Node.js is not written in JavaScript. It is written in C++ but it uses the JavaScript language as an interpretive language for server-side request/response processing. NodeJS can work with both relational (such as Oracle and MYSQL Server) and non-relational databases (such as MongoDB). [4]

2.2.3 ExpressJS:

Express.js is a NodeJS web application server framework, which is specifically designed for building single-page, multi-page, and hybrid web applications. It has become the standard server framework for NodeJS. [5]

The Express.js framework makes it very easy to develop an application to handle multiple types of requests like the GET, PUT, POST and DELETE requests. And it has the ability to work with databases which are commonly required by most modern web applications.

2.2.4 MongoDB:

MongoDB is a powerful, flexible, and scalable general-purpose database management system (DBMS) that uses a document-oriented database model which supports various forms of data. A database is a collection of information that is organized so that it can be easily accessed, managed and updated. Instead of using tables and rows as in relational databases, the MongoDB architecture is made up of collections and documents. [6]

2.2.5 ReactJS:

ReactJS basically is an open-source JavaScript library which is used for building user interfaces specifically for single page applications. It’s used for handling view layer for web and mobile apps. React also allows us to create reusable UI components.

React allows developers to create large web applications which can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. [7]

2.2.6 Bootstrap:

Bootstrap is a free and open-source CSS framework directed at responsive, simplified, and mobile-first front-end web development. It contains HTML, CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. [8]

2.2.7 Web Browser:

A web browser, or simply "browser," is an application used to access and view websites. Common web browsers include Internet Google Chrome, Mozilla Firefox, and Apple Safari. The primary function of a web browser is to render HTML, the code used to design or "markup" webpages. Each time a browser loads a web page, it processes the HTML, which may include text, links, and references to images and other items, such as cascading style sheets and JavaScript functions. The browser processes these items, then renders them in the browser window. [9]

2.3 Programming languages:

2.3.1 HTML:

Stands for “Hypertext markup” language, it is the major markup language used to display Web pages on the Internet. In other words, Web pages are composed of HTML, which is used to display text, images or other resources through a Web browser.

Hypertext refers to the hyperlinks that html page may contain, and “markup language” refers to the way tags are used to define the page layout and elements within the page. [10]

2.3.2 CSS:

Stands for Cascading Style Sheets, it is a standard (or language) that describes the formatting of markup language pages. CSS enables developers to separate content and visual elements for greater page control and flexibility. A CSS file is normally attached to an HTML file by means of a link in the HTML file. [11]

2.3.3 JavaScript:

JavaScript (JS) is a scripting language, primarily used on the Web. It is used to enhance HTML pages and is commonly found embedded in HTML code. JavaScript is an interpreted language. Thus, it doesn't need to be compiled.

JavaScript renders web pages in an interactive and dynamic fashion. This allowing the pages to react to events, exhibit special effects, accept variable text, validate data, create cookies, detect a user’s browser, etc. [12]

2.3.4 TypeScript:

TypeScript is a programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and adds optional static typing to the language. It is designed for the development of large applications and transpiles to JavaScript. As it is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs.[13]

2.4 Conclusion:

Through this chapter, we gave simple and brief definitions about the tools and technologies that we used in order to implement our interactive, dynamic and responsive web application.

CHAPTER THREE

Application design.

3.1 Introduction:

Web design usually refers to the user experience aspects of web app development rather than software development. A web designer works on the appearance, layout, and, in some cases, content of a web app. We are going to define the roles of each actor that interacts with the system. In addition, we will use UML for modeling and in particular we choose the use case diagram to model these roles. We end the chapter by introducing the database and some of its models.

3.2 Design Requirements:

3.2.1 Modelling language:

A modelling language is mainly used in the field of computer science and engineering for designing models of new software, systems, devices and equipment. Unified modelling language (UML) is a popular modelling language that is used to build system and object models graphically. [14]

3.2.2 Unified Modelling Language (UML):

The Unified Modelling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artefacts of a software-intensive system. It offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components. [20]

The current UML standards call for different types of diagrams. These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams, which are as follows:

Structural UML diagrams: A type of diagram that depicts the elements of a specification that are irrespective of time. This includes class, composite structure, component, deployment, object, and package diagrams. [15]

Behavioral UML diagrams: A type of diagram that depicts behavioral features of a system or business process. This includes activity, Sequence, Use case, State, Communication, Interaction, and Timing diagrams. [15]

We are going to design and model our application using use cases diagrams.

3.3 Use Case Diagram

3.3.1 Definition:

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. Actors are people or entities operating under defined roles within the system. [16]

Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation. [17]

3.3.2 Use Cases:

Use cases are represented using ovals labeled with verbs that describes the system's functions. In our application, we define the following use cases:

a. Authentication: Login before accessing the application using a valid email and password. Moreover, authentication insures the identity of the user.

b. Register students: Allows Admin to add new students in the institute to the system with necessary information, updating students majors and mark graduate students every year.

c. Promotions: Allows Admin to create a new promotion for first year students or update an existing promotion in the beginning of every academic year.

d. Groups: Allows Admin to create, update, delete groups for every promotion.

e. Assign students: Allows Admin to add each student to a specific group.

f. Courses: Here the Admin will assign for each course a teacher, a promotion and fill the corresponding coefficient percentages for each type of evaluation.

g. Evaluations: Allows a teacher for each course he/she was assigned to, to update the grades of students depending on the type of evaluation.

h. Results and deliberation: An admin is able to visualize and print the generated results (Courses, semester and annual averages) for all students.

i. Manage users: Allows admin to add, edit, and remove users.

3.3.3 Actors:

Actors are usually individuals involved with the system defined according to their roles. The actor can be a human or other external system. Actors interacting with our application are:

I. Admin:

- Authentication.
- Register students.
- Promotions.
- Groups.
- Assign students.
- Courses.
- Evaluations.
- Results and deliberation.
- Manage users.

II. Teacher:

- Evaluations

3.3.4 Relationships:

The following summarizes the used arrows:

Table 3. 1 Summary of the used arrows.

Symbols	Description
Association ————————	**Association:** Use cases are associated with the actors that perform them. A line is used to link actors to use cases.
Include ---------->	**Include**: an include relationship shows dependency between a base use case and an included use case Every time the base use case is executed the included use case is executed as well, another way to think of it is that the base use case requires an included use case in order to be complete. When we have an included use case, we draw a dashed line with an arrow that points towards the included use case.
Extend <----------	**Extend:** it has also a base use case and an extend use case when the base use case is executed the extend use case will happen sometimes but not every time, the extended use case will only happen if certain criteria are met, another way to think of it is that you have the option to extend the behaviour of the base use case, when we have an extended use case, we draw a dashed line with an arrow that points towards the base use case.
Generalization ———▷	**Generalization** relationship is also a parent-child relationship between use cases. The child use case has the underlying business process meaning but is an enhancement of the parent use case. The child use case is connected at the base of the arrow. The tip of the arrow is connected to the parent use case.

3.3.5 Teacher's use case diagram design:

The figure 3.1 below shows use case diagram of the Teacher:

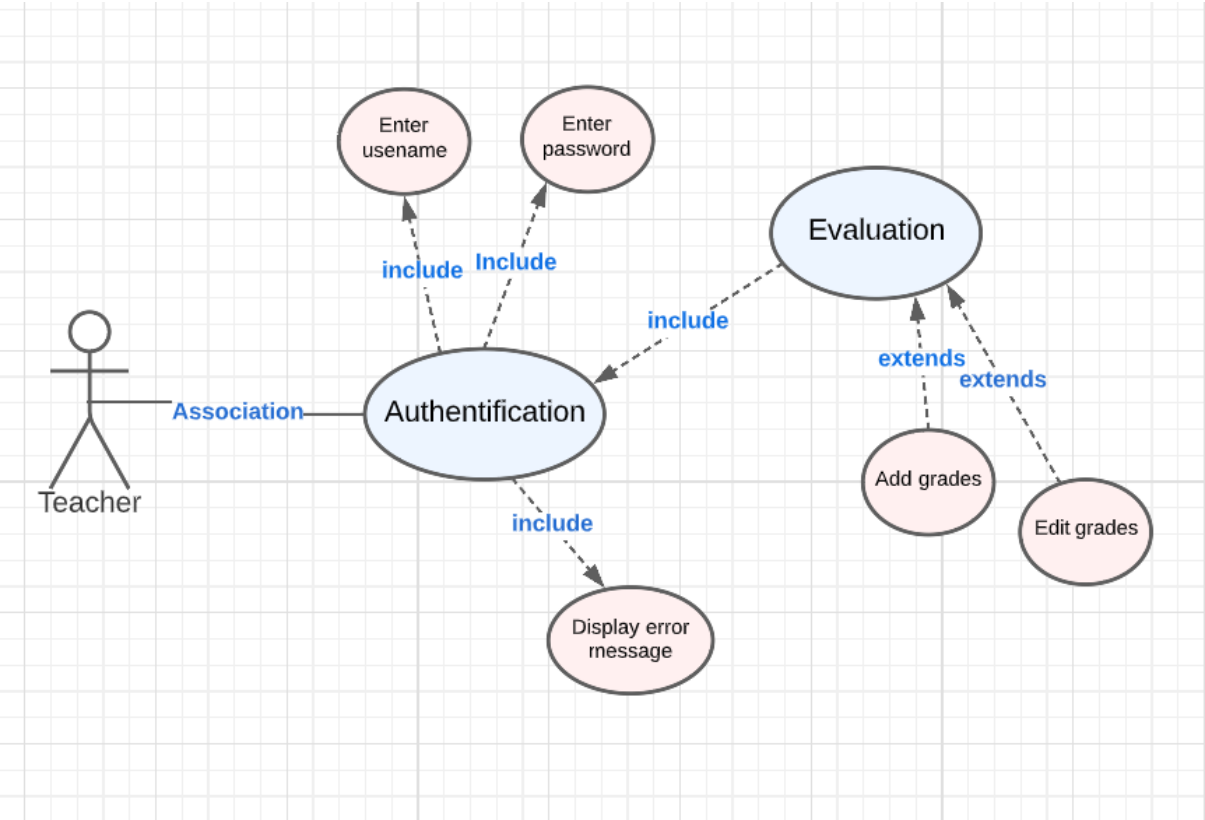


Figure 3. 1 Teacher Use Case Diagram

3.3.6 Admin's use case diagram design:

The figure 3.2 below shows use case diagram of the admin:

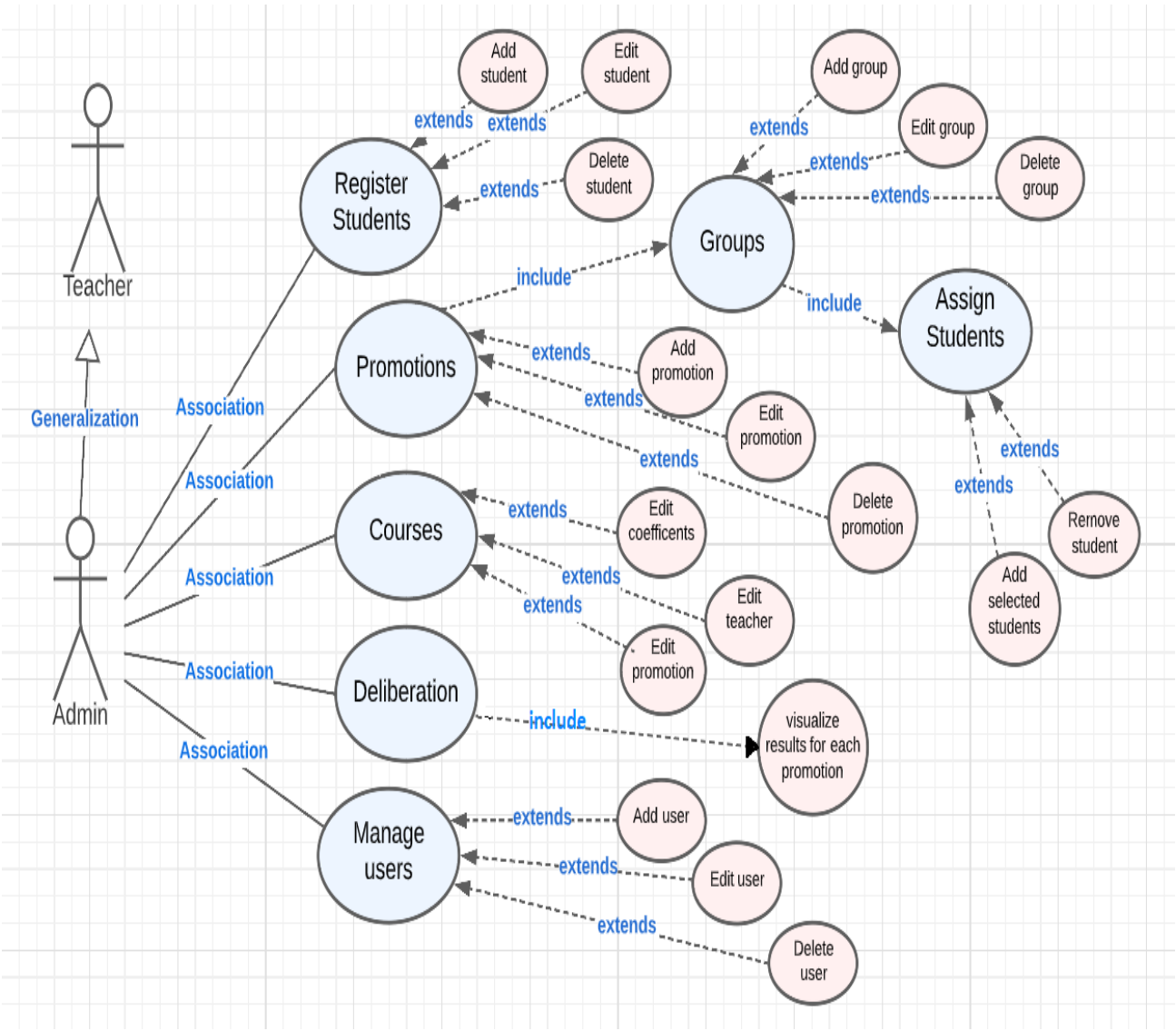


Figure 3. 2 Admin Use Case Diagram

3.3.7 Textual description of use cases:

Use case 01: Authentication

Table 3. 2 Textual description for use case "Authentication"

Use case name	Authentication
Actor	Admin, Teacher
Objective	Authenticate to have access to the application
Precondition	Browser and access to internet
Scenario	1. The user lunches the application via a browser. 2. The system asks for username and password. 3. The user enters his name and password. 4. The system checks the conformity of the information entered by sending an authentication query to the server. 5. The server verifies the query and send favourable answer. 6. The user accesses the application
Alternative	If the username or password is wrong or missed the system displays an error message.

Use case 02: Register students

Table 3. 3 Textual description for use case " Register students "

Use case name	Register students
Actor	Admin
Objective	Add, edit, or delete students from the system
Precondition	Authentication
Scenario	1. The user accesses the ‘students’ interface. 2. The user can see the list of existing students in the system. 3. The user adds, edits, or delete a student. 4. The list of students is updated directly after every submit.
Alternative	- The system displays an error message. - if changes do not reflect directly on the list of students, refresh the page.

Use case 03: Promotions

Table 3. 4 Textual description for use case " Promotions "

Use case name	Promotions
Actor	Admin
Objective	Create, edit or delete promotions.
Precondition	Authentication
Scenario	1. The user accesses the ‘promotions interface. 2. The user can visualize the list of existing promotions. 4. The user adds, edits or delete a promotion 5. The system sends a query to the server for processing. 6. The list of promotions screen is updated directly after every submit. 7. The user accesses a specific promotion interface by clicking on it in the list.
Alternative	The system displays an error message.

Use case 04: Groups

Table 3. 5 Textual description for use case " Groups

Use case name	Groups
Actor	Admin
Objective	Add, delete a group for a specific promotion
Precondition	Promotions
Scenario	1. The user accesses the interface of a single promotion. 2. The user adds a group to a promotion, or delete one 4. The list of groups can be seen under group adding area. Any changes reflect directly. 6. The user accesses a specific group interface by clicking on it in the list
Alternative	The system displays an error message.

Use case 05: Assign students

Table 3. 6 Textual description for use case " Assign students"

Use case name	Assign students
Actor	Admin
Objective	Add or remove students from a group of a promotion
Precondition	Groups
Scenario	1. The user accesses a specific group interface. 2. The user selects the students of the current promotion to add to this group. (If a student was added before, the select field assigned is disabled). 4. The user can see the list of students for this group. 5. The user can delete a student from this group.
Alternative	The system displays an error message.

Use case 06: **Courses**

Table 3. 7 Textual description for use case " Courses"

Use case name	Courses
Actor	Admin
Objective	Update courses by assigning teachers, promotions, and coefficients for each course.
Precondition	Authentication
Scenario	1. The user accesses ‘courses’ interface. 2. The user selects a major and a semester. 3. The system shows a table of courses. 4. the user update the necessary fields for each course. 5. The changes are reflected on the table for each course.
Alternative	The system displays an error message.

Use case 07: Evaluations

Table 3. 8 Textual description for use case "Evaluations"

Use case name	Evaluations
Actor	Teacher, Admin
Objective	Evaluating students
Precondition	Courses
Scenario	1. The user accesses the ‘evaluation’ interface. 2. The user selects a course, evaluation type (control, exam…), and a group. 3. The user adds a grade or absence for each student.
Alternative	The system displays an error message.

Use case 08: **Results and deliberation**

Table 3. 9 Textual description for use case " Results and deliberation"

Use case name	Results and deliberation
Actor	Admin
Objective	Obtaining generated results (averages) for each student
Precondition	Evaluations
Scenario	1. The user accesses the ‘deliberation’ interface. 2. The user selects a promotion. 3. A Table shows, containing courses, semestrial, and annual averages for each student. 4. All results are generated automatically based on the grades of each student in each enrolled course.
Alternative	The system displays an error message.

Use case 09: Manage users

Table 3. 10 Textual description for use case " Manage users "

Use case name	Manage users
Actor	Admin
Objective	Add, edit, or delete users from the system
Precondition	Authentication
Scenario	1. Admin accesses the ‘users’ interface. 2. Admin can see the list of existing students in the system. 3. Admin adds, edits, or delete a student. 4. The list of students is updated directly after every submit.
Alternative	- The system displays an error message. - if changes do not reflect directly on the list of users, refresh the page.

3.4 Introduction to Database:

3.4.1 Definition:

Database is a collection of information that exists over a long period of time. The term database refers to a collection of data that is managed by database management system (DBMS). A DBMS is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications. [21]

3.4.2 Relational Model:

A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database. Each column in a table holds a certain kind of data and a field stores the actual value of an attribute. The rows in the table represent a collection of related values of one object or entity. Each row in a table could be marked with a unique identifier called a primary key, and rows among multiple tables can be made related using foreign keys. [18]

3.4.3 Non-Relational Model:

A non-relational database is a database that does not use the tabular schema of rows and columns found in most traditional database systems. Instead, non-relational databases use a 'Document data stores' model that is optimized for the specific requirements of the type of data being stored.

A document data store manages a set of named string fields and object data values in an entity referred to as a *document*. These data stores typically store data in the form of JSON documents. Each field value could be a scalar item, such as a number, or a compound element, such as a list or a parent-child collection. The data in the fields of a document can be encoded in a variety of ways, including XML, JSON, BSON, or even stored as plain text.

The fields within documents are exposed to the DBMS, enabling an application to query and filter data by using the values in these fields. [19]

3.4.5 What is MongoDB?

MongoDB is a non-relational database developed by MongoDB, Inc. MongoDB uses a JSON-like representation format called BSON (Binary JSON) to store data. The fondamental unit to store data is called document. Related information is stored together for fast access through the MongoDB query. There is no need to declare the structure of documents to the system as documents are self-describing and the field can flexibly vary from document to another. If a new field needs to be added to a document, then the field can be created without affecting all other documents in the collection, without updating a central system, and without taking the system offline. Optionally, schema validation can be used to enforce data governance controls over each collection.

A NoSQL database like MongoDB is schema-less. Instead of storing rows in a table you create "documents" that are stored in collections.

3.4.6 Benefits of using documents in database:

• **Documents are natural**. Documents represent data in the same way that applications do. Unlike the tabular rows and columns of a relational database, data can be structured with arrays and subdocuments – in the same way applications represent data, as lists and members / instance variables respectively. This makes it much simpler and faster for developers to model how data in the application will map to data stored in the database.

• **Documents are flexible**. Each document can store data with different attributes from other documents. With JSON documents, we can add new attributes when we need to, without having to alter a centralized database schema.

• **Documents make applications fast**. With data for an entity stored in a single document, rather than spread across multiple relational tables, the database only needs to read and write to a single place. Having all the data for an object in one place also makes it easier for developers to understand and optimize query performance.

3.4.7 Description of our system's database:

Our database system contains five collections: students, promotions, courses, grades, and users.

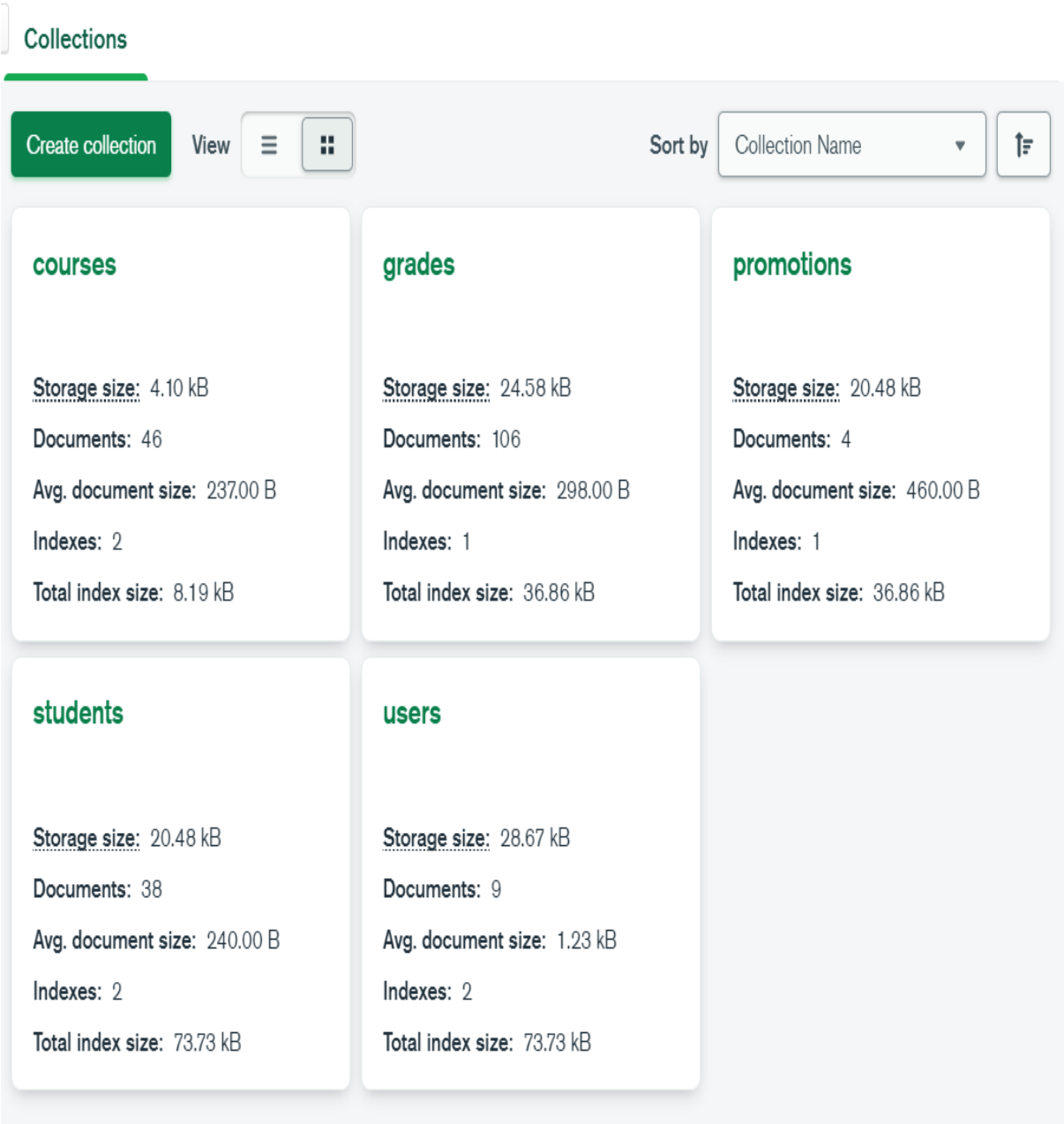


Figure 3. 3 database interface

3.4.7.1 Students collection:

Documents in that collection contain eleven attributes:

- **_id**: it is an identification number that the database inserts by default.
- **studentId**: the unique ID of the student that appear in student card.
- **firstName**: the first name of the student.
- **lastName**: the last name of the student.
- **sex**: the gendre of the student (male, female).
- **birthDate**: date of birth of the student.
- **birthPlace**: location of birth of the student.
- **degree**: the degree a student is pursuing (license, master).
- **level**: the major of the student (L1, L2...).
- **createdAt**: date of creation this document.
- **updatedAt**: date of update this document.

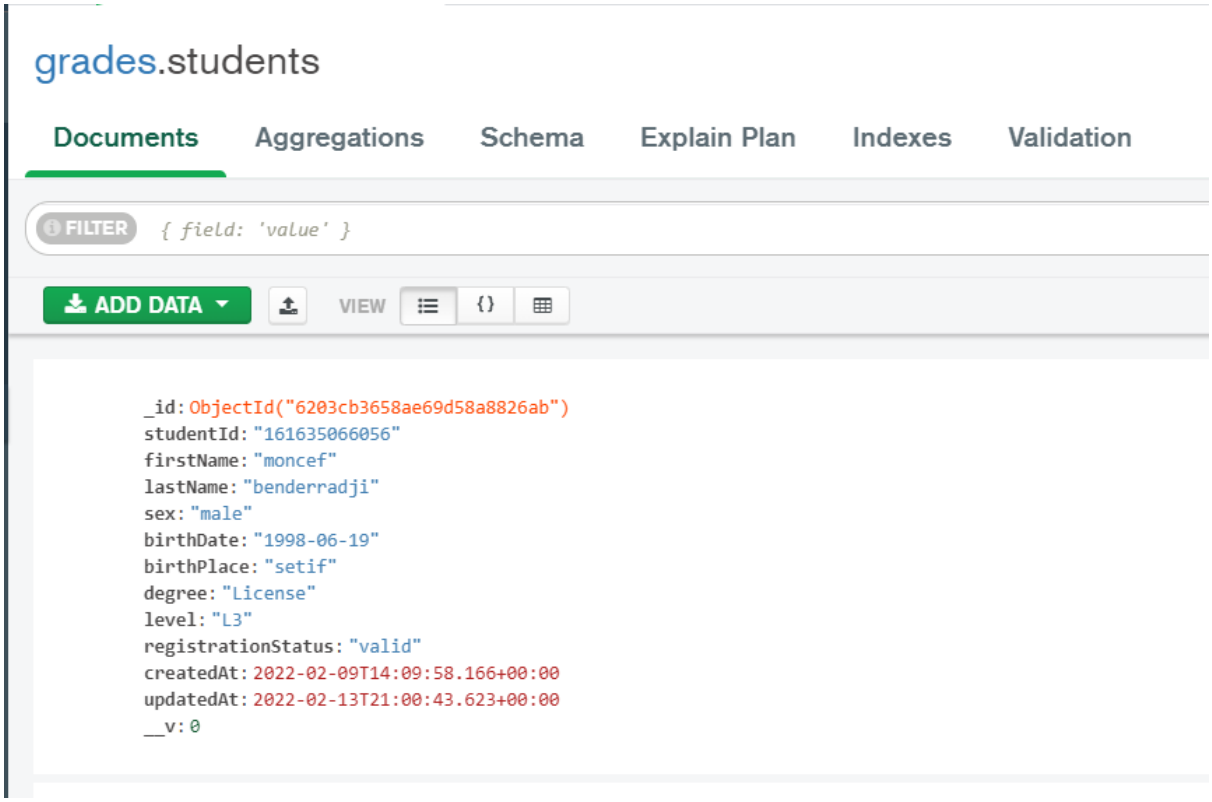


Figure 3. 4 students collection

3.4.7.1 Users collection:

Documents in that collection contain seven attributes:

• **_id**: it is an identification number that the database inserts by default.

• **firstname**: the first name of the user.

• **lastname**: the last name of the user.

• **username**: the username used by the user to login.

• **salt**: hashing the password.

• **hash**: hashing the password.

• **role**: the role of the user (Teacher, Admin).

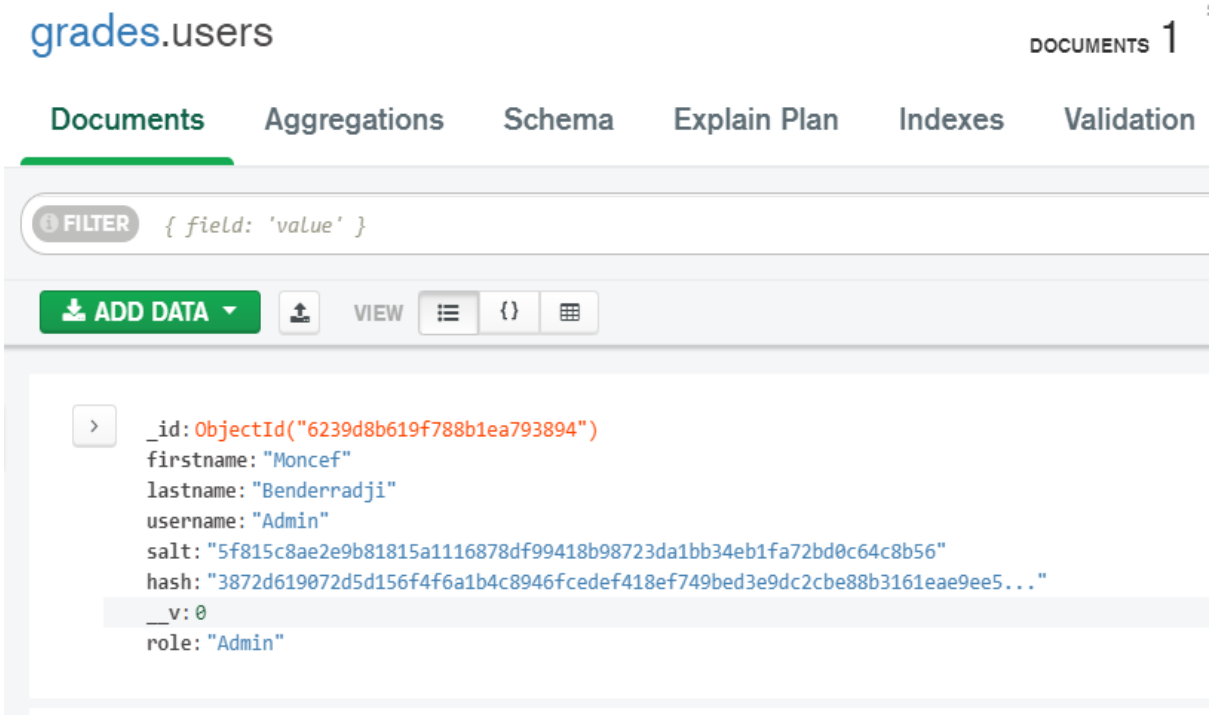


Figure 3. 5 users collection

3.4.7.1 Promotions collection:

Documents in that collection contain eight attributes:

• **_id**: it is an identification number that the database inserts by default.

• **year**: the academic year of the promotion.

• **degree**: the degree that students of the promotion pursue (license, master).

• **major**: the major of students of the promotion.

• **numberOfGroups**: the number of groups of the promotion.

- **groups**: an array of group sub-documents, each sub-document contains:
    - **_id**: it is an identification number that the database inserts by default.
    - **groupNumber**: the number of the group.
    - **students**: an array of id references to students' collection.
    - **createdAt**: date of creation of that sub-document.
    - **updatedAt**: date of update of that sub-document.
- **createdAt**: date of creation of that document.
- **updatedAt**: date of update of that document.

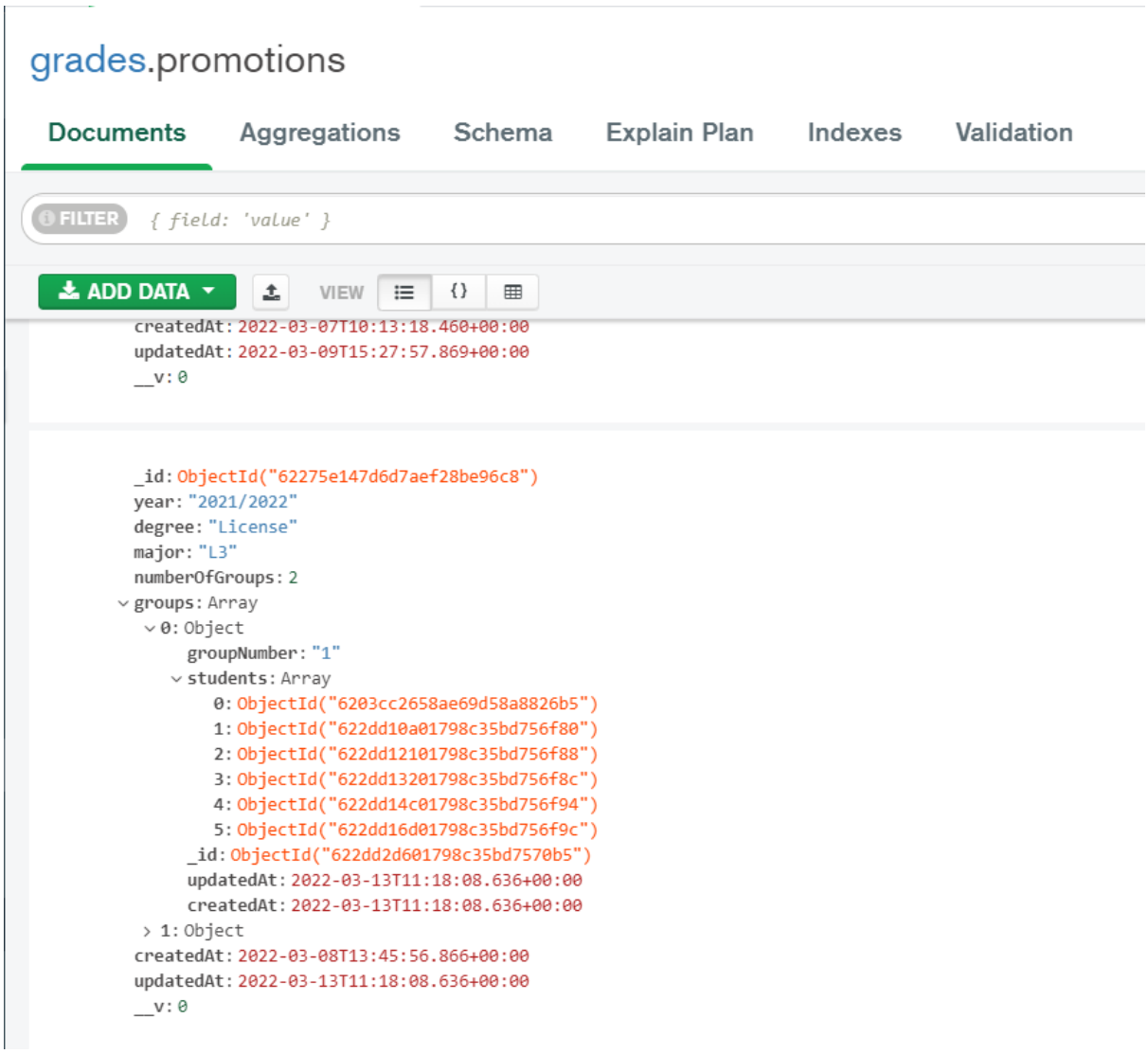


Figure 3. 6 promotions collection

3.4.7.1 courses collection :

Documents in that collection contain twelve attributes:

• **_id**: it is an identification number that the database inserts by default.

• **name**: the name of the course.

• **teacher**: an id reference to users' collection, it references only documents with role attribute 'Teacher'.

• **type**: the type of the course (lecture, lab).

• **promotion**: an id reference to promotions collection.

• **major**: the major of students enrolled in this course.

• **semester**: the semester in which this course is taught.

• **coef**: the coefficient of this course.

• **createdAt**: date of creation of that course.

• **updatedAt**: date of update of that course.

• **examCoef**: the coefficient of the course's exam.

• **controlCoef**: the coefficient of the course's control.

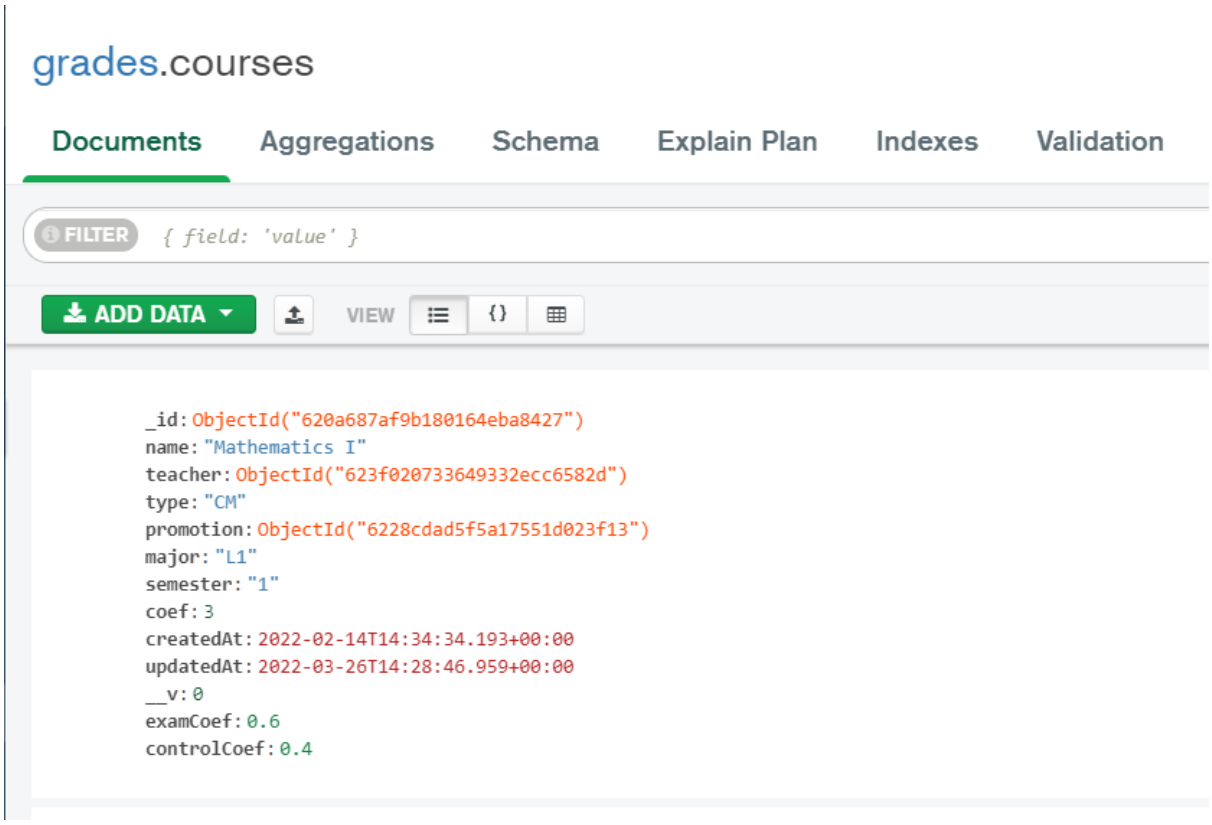


Figure 3. 7 courses collection

3.4.7.1 Grades collection :

Documents in that collection contain four attributes:

- **_id**: it is an identification number that the database inserts by default.
- **student**: an id reference to students' collection.
- **course**: an id reference to courses collection.
- **evaluations**: an array of evaluation sub-documents, each sub-document contains.
    - **_id**: it is an identification number that the database inserts by default.
    - **type**: the type of the evaluation (control, exam…).
    - **absent**: indicates the presence or absence of the student in this evaluation.
    - **value**: the grade of the student in this evaluation.
    - **createdAt**: date of creation of that evaluation.
    - **updatedAt**: date of update of that evaluation.
- **createdAt**: date of creation of that document.
- **updatedAt**: date of update of that document.

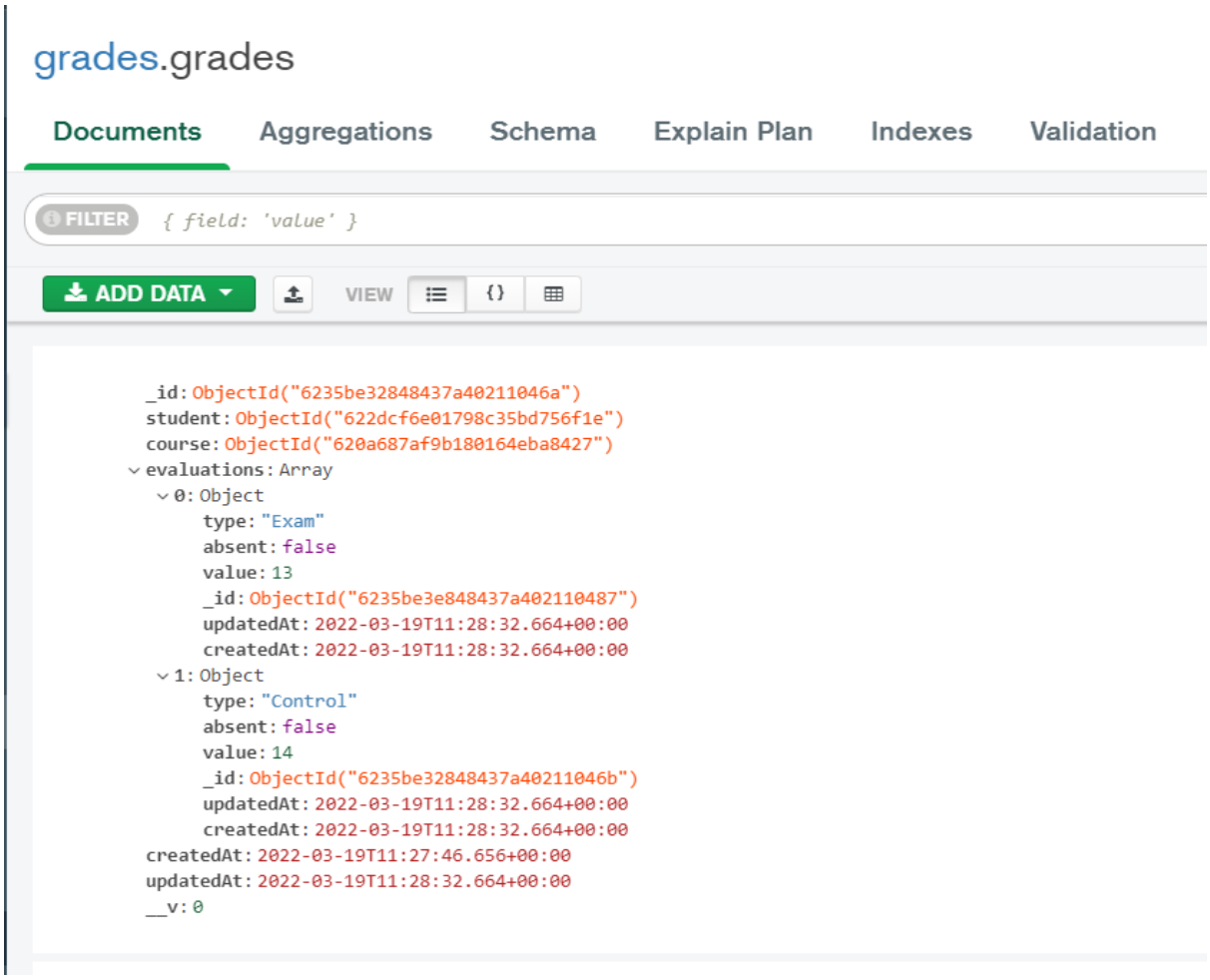


Figure 3. 8 users collection

3.3.3 Relationships between documents:

Relationships in MongoDB represent how various documents are logically related to each other. Relationships can be modeled via Embedded and Referenced approaches. Such relationships can be either 1:1, 1:N, N:1 or N:M. [20]

3.3.3.1 Embedded Relationships

Embedded documents capture relationships between data by storing related data in a single document structure. MongoDB documents make it possible to embed document structures in a field or array within a document. These allow applications to retrieve and manipulate related data in a single database operation.

Figure 3. 9 Embedded data relationship

3.3.3.2 Referenced Relationships

References store the relationships between data by including links or references from one document to another. Applications can resolve these references to access the related data.

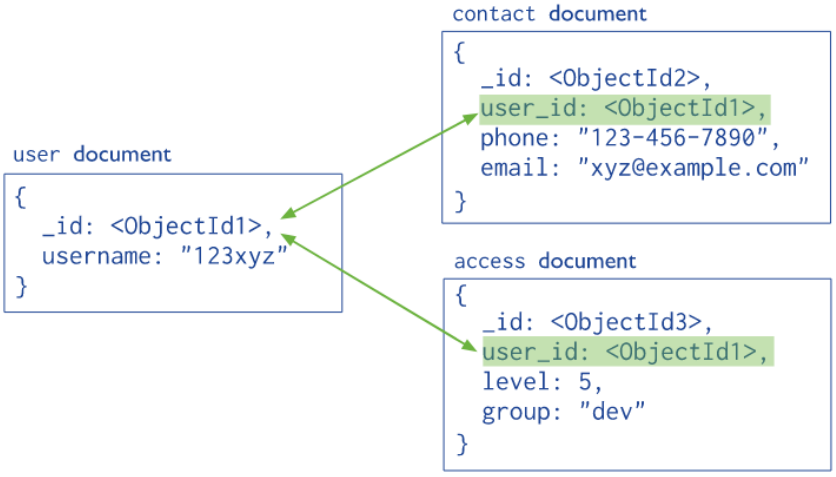


Figure 3. 10 References relationship

The following table summarizes the used arrows:

Table 3. 11 used arrows in relationships diagram

Symbol	Description
	One to many: one document in a collection can be associated with one or more documents in another collection
	One to one: one document in a collection is associated with one and only one document in another collection

Relationships diagram of our system:

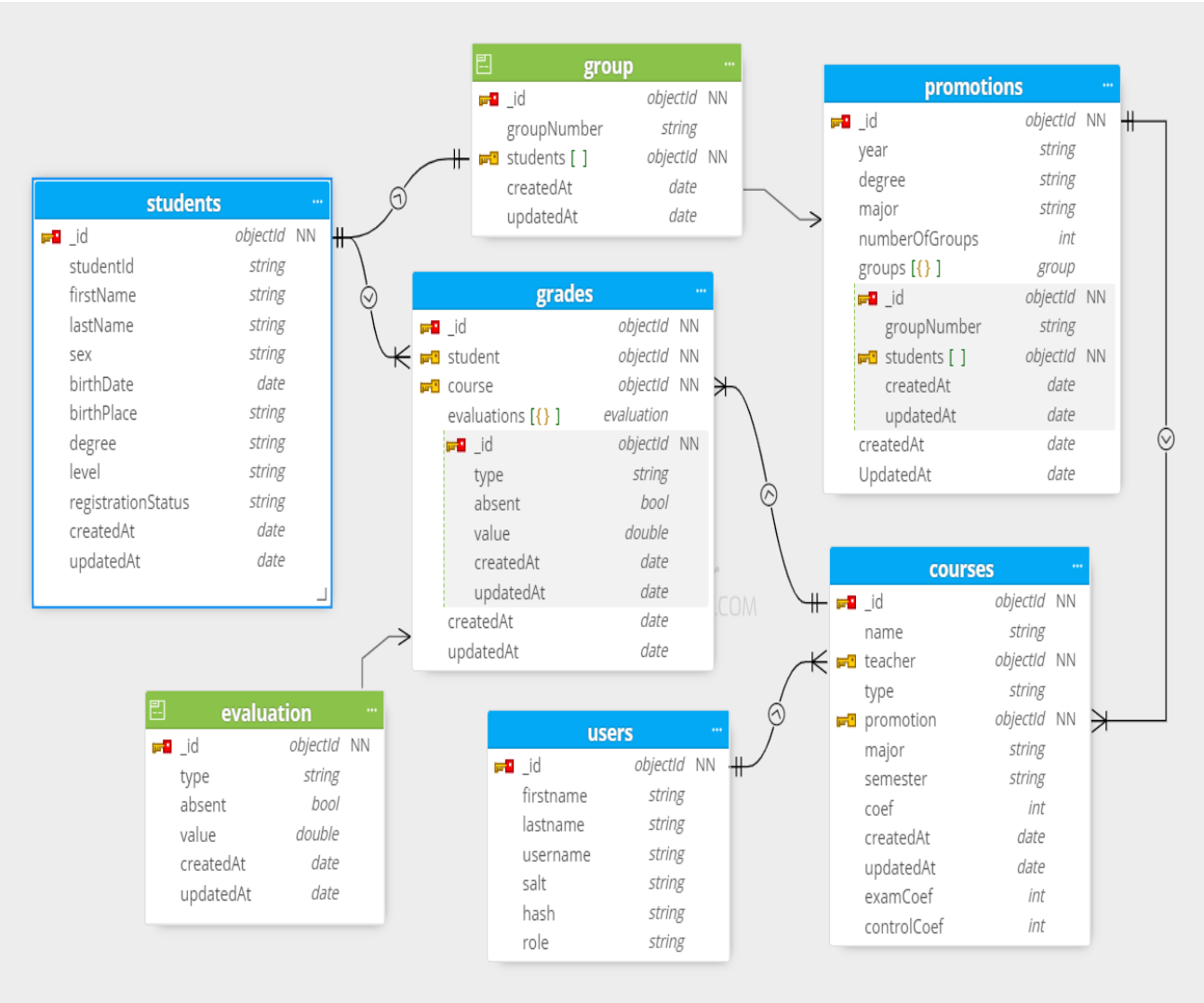


Figure 3. 11 Relationships between documents.

3.5 Conclusion:

In this chapter, we have used features of UML to model all functionalities involved in our system through use case diagram. Moreover, we modelled our application by providing answers to our modelling and design, based on the analysis of needs of our application. Finally, we introduced our database which uses the non-relational model, and we gave some benefits of using this model.

CHAPTER FOUR

Implementation of application.

4.1 Introduction:

After designing the application and specifying the roles of each user, this last chapter is concerned with the implementation. We give a presentation of the application and its functionalities accompanied with some user interfaces.

4.2 Interfacing with the application:

After implementing the application with the tools specified in the previous chapters, in this paragraph, we present the different interfaces, the user will be dealing with.
It is to be noted that all the filled data that are saved in the database was generated by a faker library that give as a random string.

4.2.1 Authentication interface:

As all secured applications, this following interface is the first one that will be displayed when the user wants to access the application.
This application came with a single admin account. This admin is responsible of registering other admins or teachers to the system. The registered admins become also eligible of signing up other admins and teachers, because they would have access to the users' interface where they can achieve that. Any registered users will receive by email their usernames and passwords, so they could access the application.
When the user enters username and password, the system sends a query to the server for checking. If this information exists in the database he can access, otherwise an error message will be displayed. The figure 4.1 below shows the authentication interface.

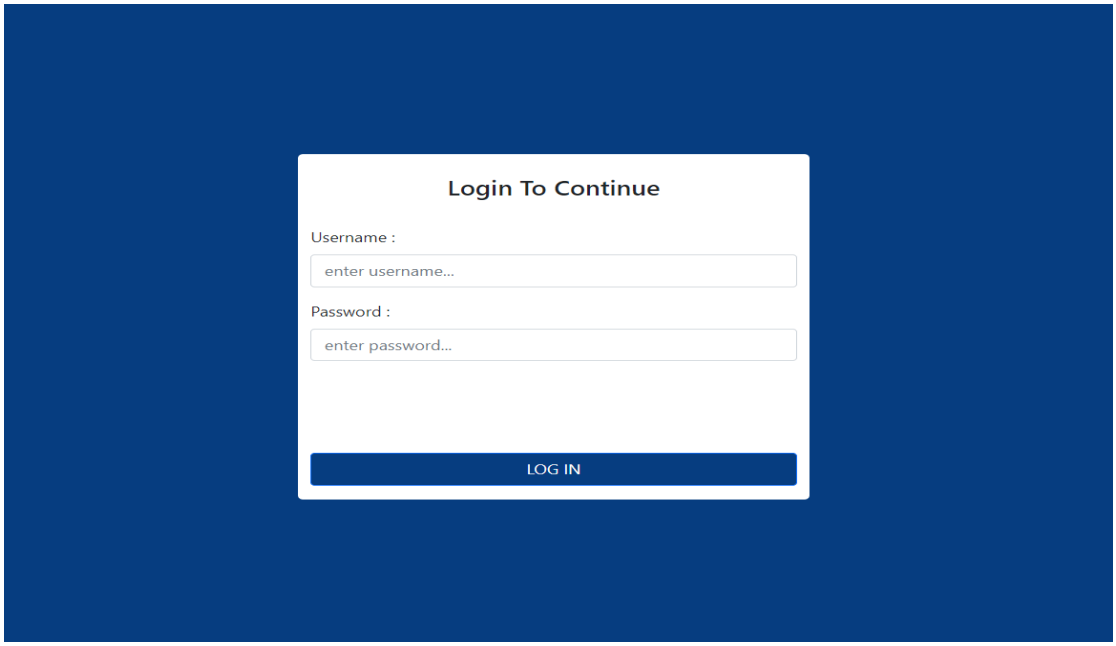


Figure 4. 1 Authentication Interface

4.2.2 Homepage interfaces for each user:

All users have two parts in their homepage interface, a main area and a sidebar. The main area is shared between all users. It contains a simple dashboard with some statistics about students and access to different promotions interfaces for admins. The sidebar differs between admins and teachers:

- For the teacher, it contains access to only Home page and Evaluation.
- For the admin, it contains access to all the available interfaces.

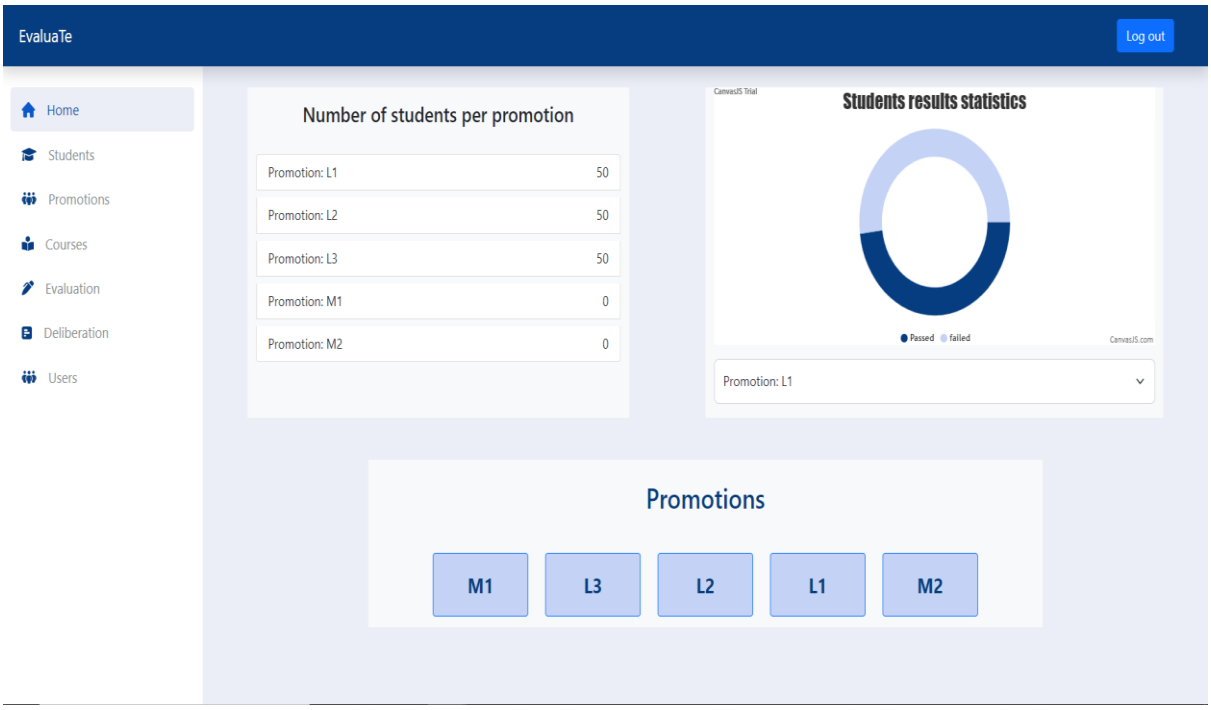


Figure 4. 2 Admin's homepage interface

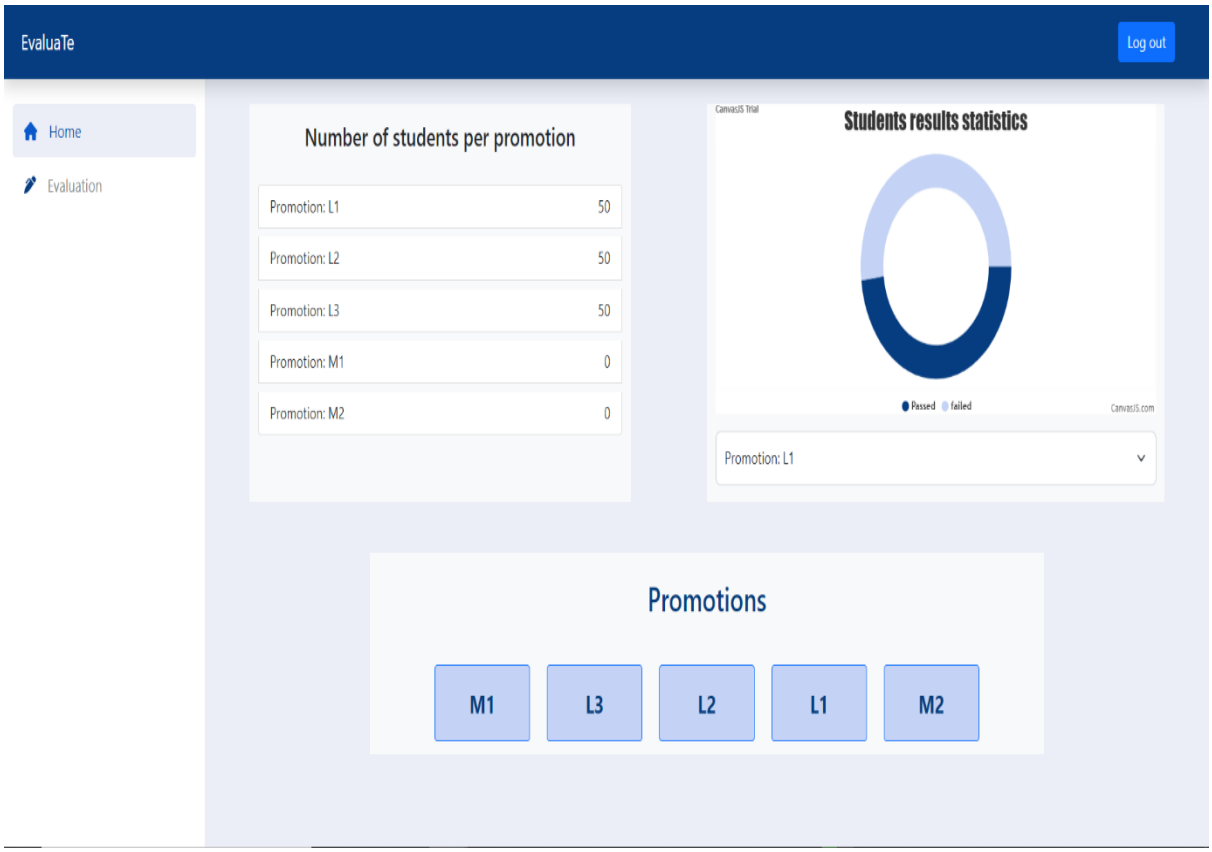


Figure 4. 3 Teacher's homepage interface

4.2.3 Registering users interface:

The admin account is the one eligible for registering users. When admin accesses this interface, a list of already registered users with their information would appear. On top of it, a button exists for adding a new user to the system, also in the right side exists a search bar to find a specific user based on a field of information related.

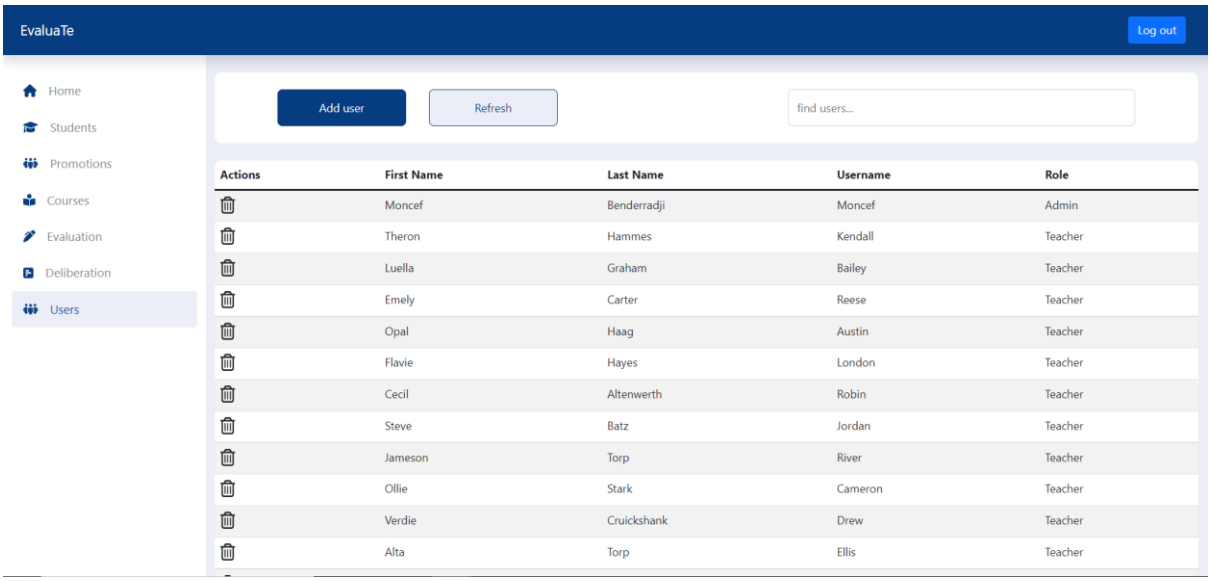


Figure 4. 4 Users interface

When add user button is clicked, the interface for adding a new user appears as follows in figure 4.5:

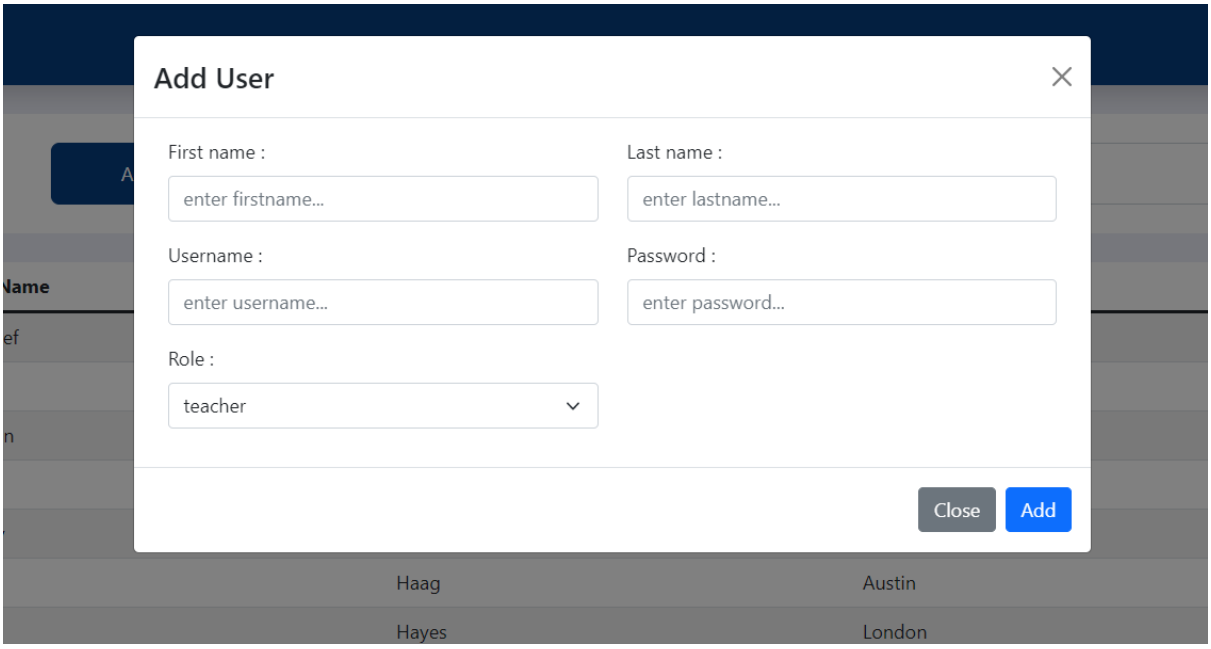


Figure 4. 5 Add user interface

When ‘Add’ is clicked, a new user would be added to the system and the list of users is updated.

4.2.4 Students interface:

When the admin accesses students' interface, a list of existed students in the system appears with the necessary information for each student, the admin have a search bar that helps in finding a specific student or a bunch of students with a shared information.

The admin have a button when clicked leads to the interface for adding a new student.

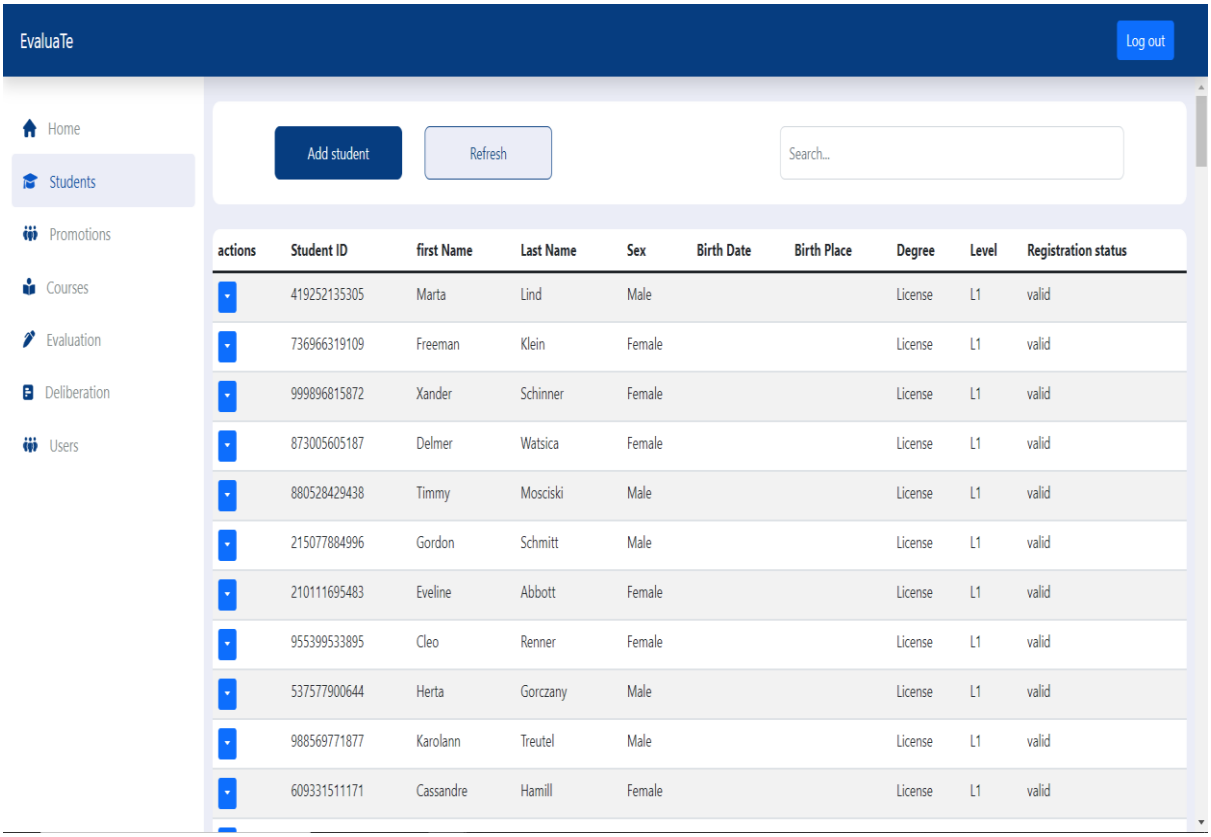


Figure 4. 6 Students interface

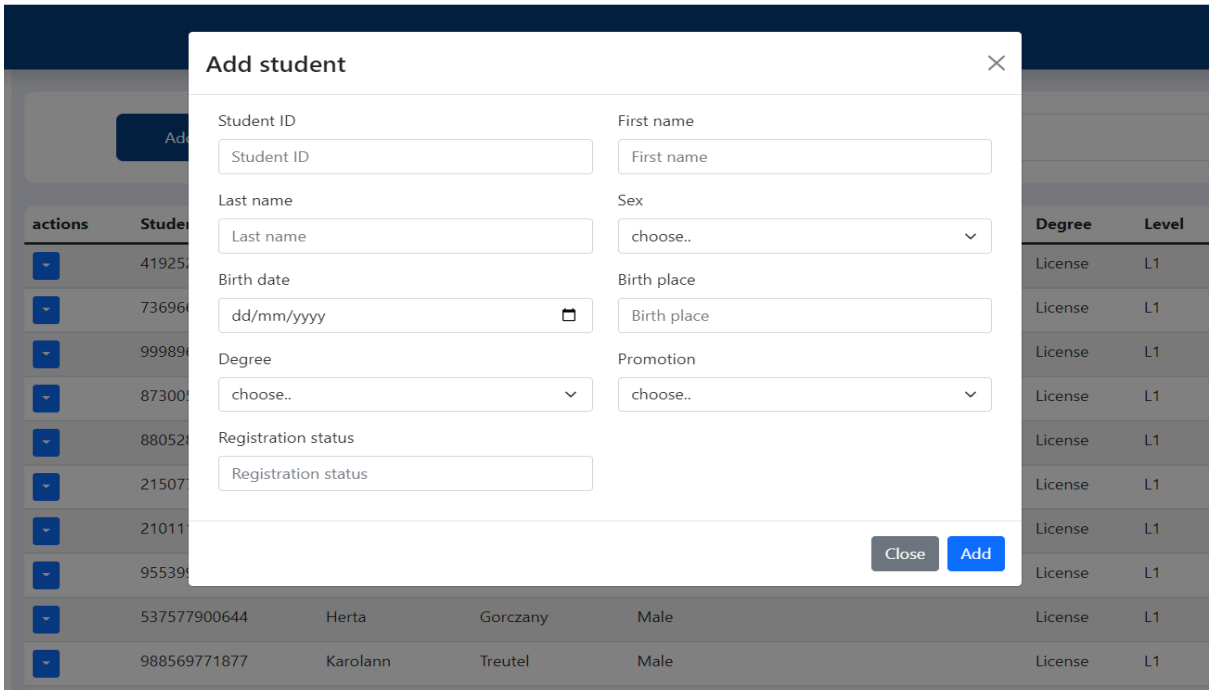


Figure 4. 7 Add student interface

In the list of students there exists for each student an actions field used to either edit informations of a specific student or delete this student from the system. The admin has two buttons: an edit button for updating, and a delete button. When either of them is clicked, the corresponding interface appears (edit student interface and delete student interface).

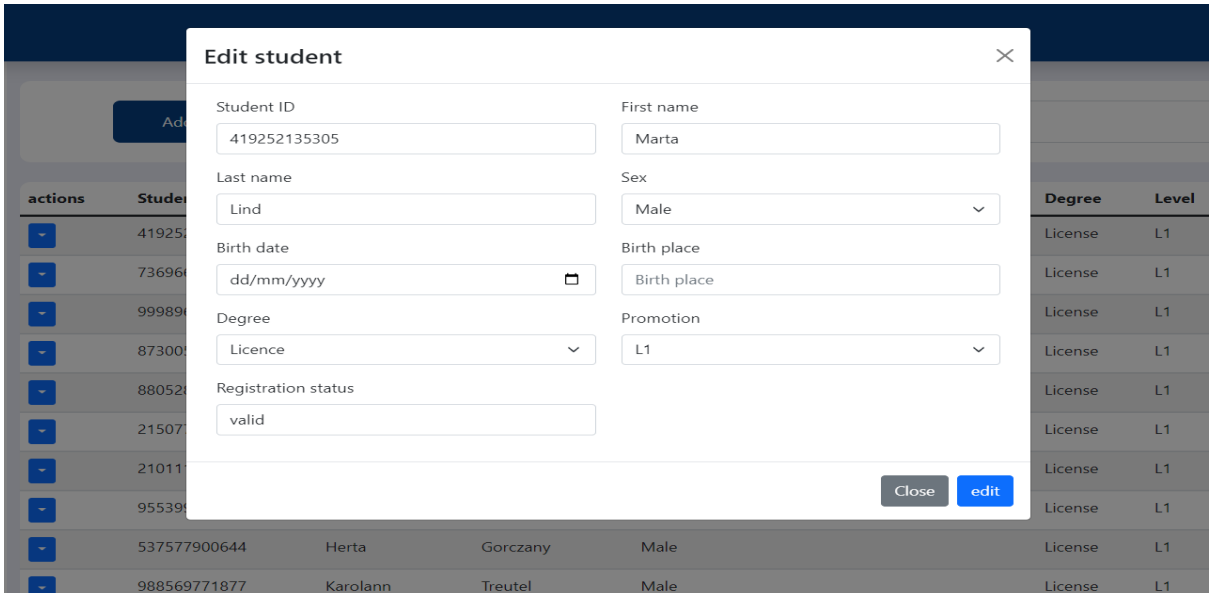


Figure 4. 8 Edit student interface

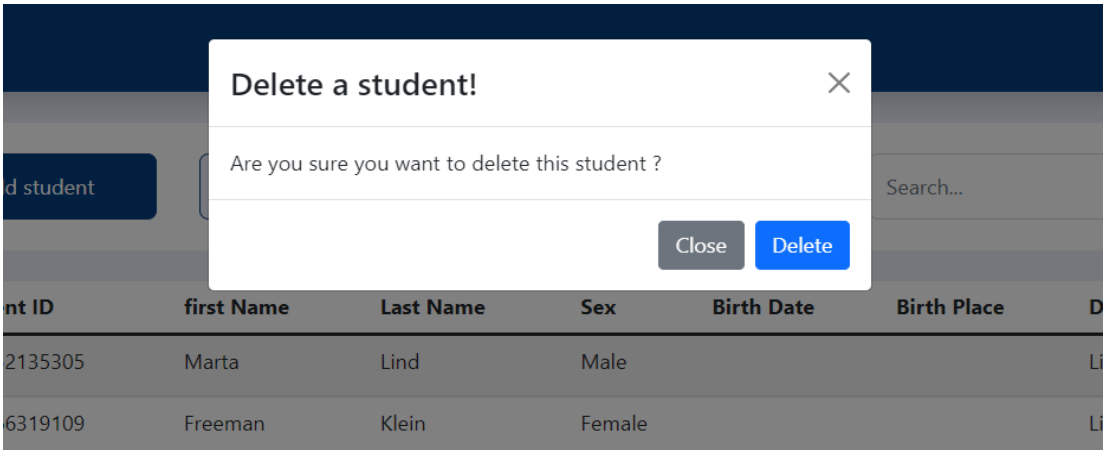


Figure 4. 9 Delete student interface

4.2.5 Promotions interface:

When the admin accesses the promotions interface, a list of existing promotions appears. Admin have a button on top of it when clicked, it leads to the interface for adding a new promotion by filling the required fields and submitting.

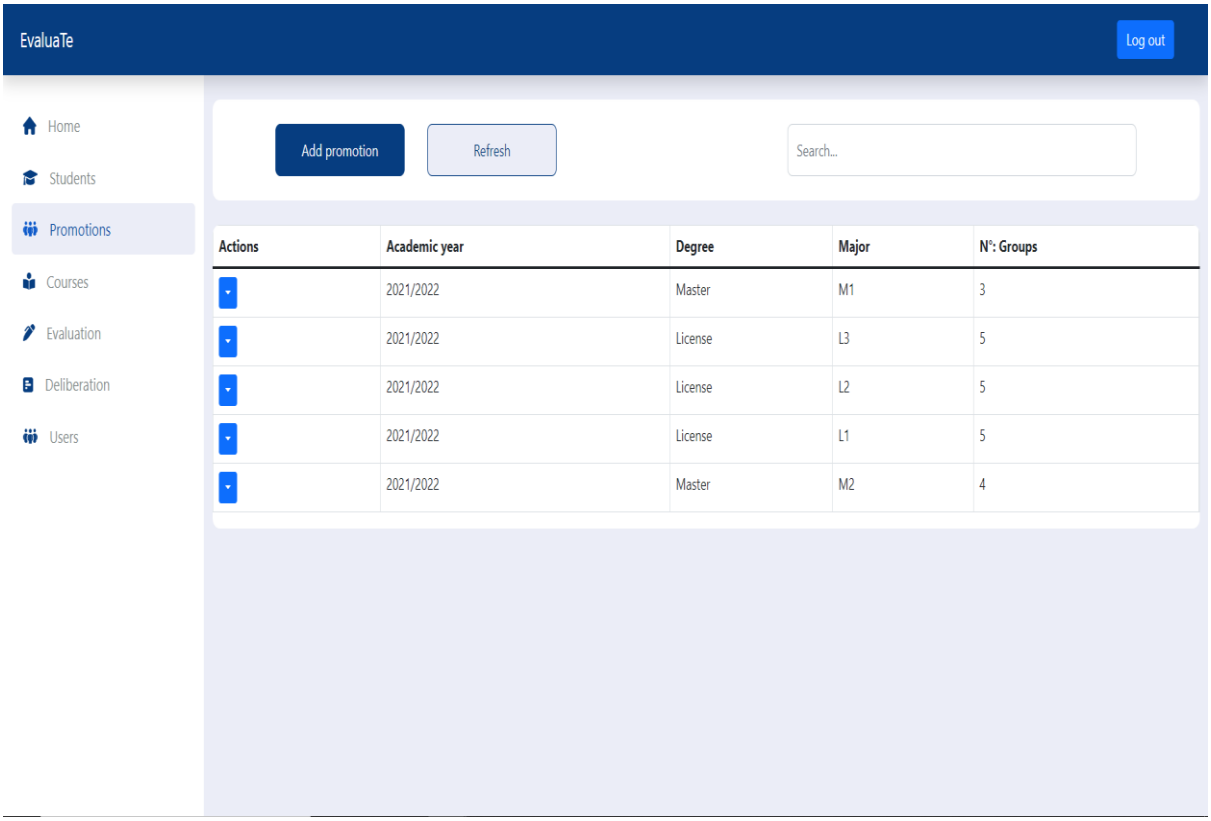


Figure 4. 10 Promotions interface

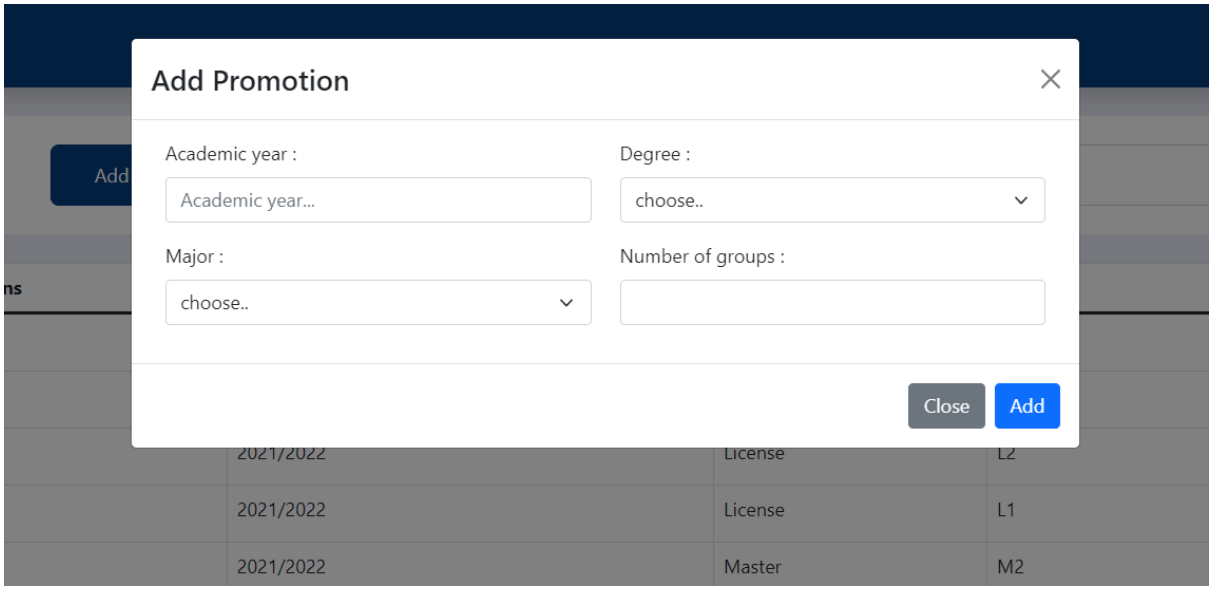


Figure 4. 11 Add promotion interface

The list of promotion contains for each promotion an actions field. When clicked a dropdown menu appears with three buttons: an edit button to access edit promotion interface, a delete button to access the delete promotion interface, and a final button when clicked, it leads to the groups interface for the specified promotion.

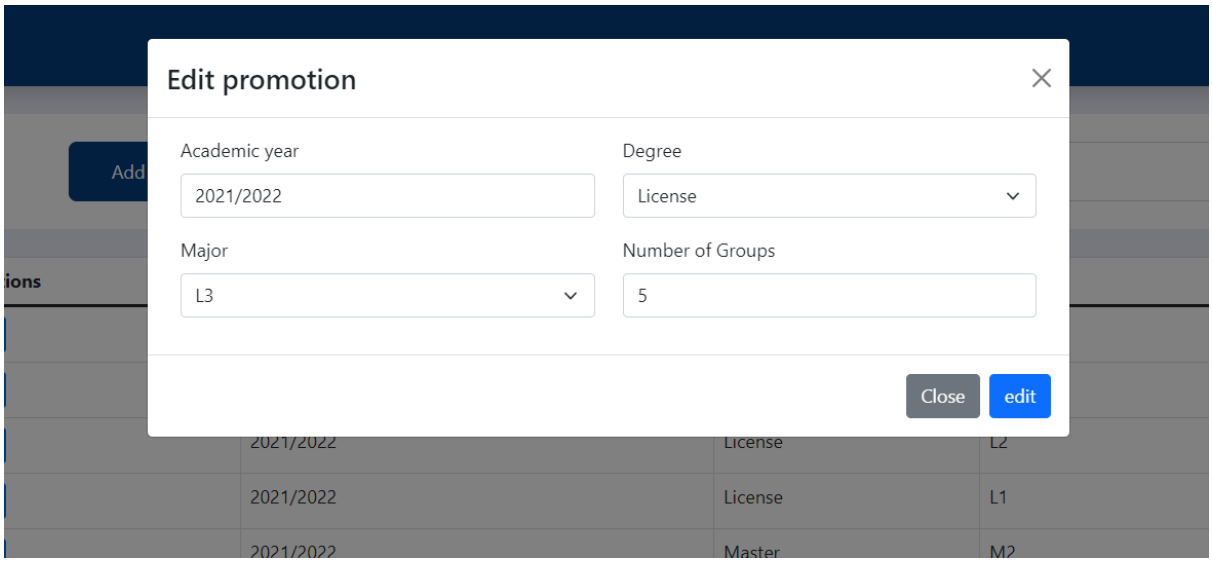


Figure 4. 12 Edit promotion interface

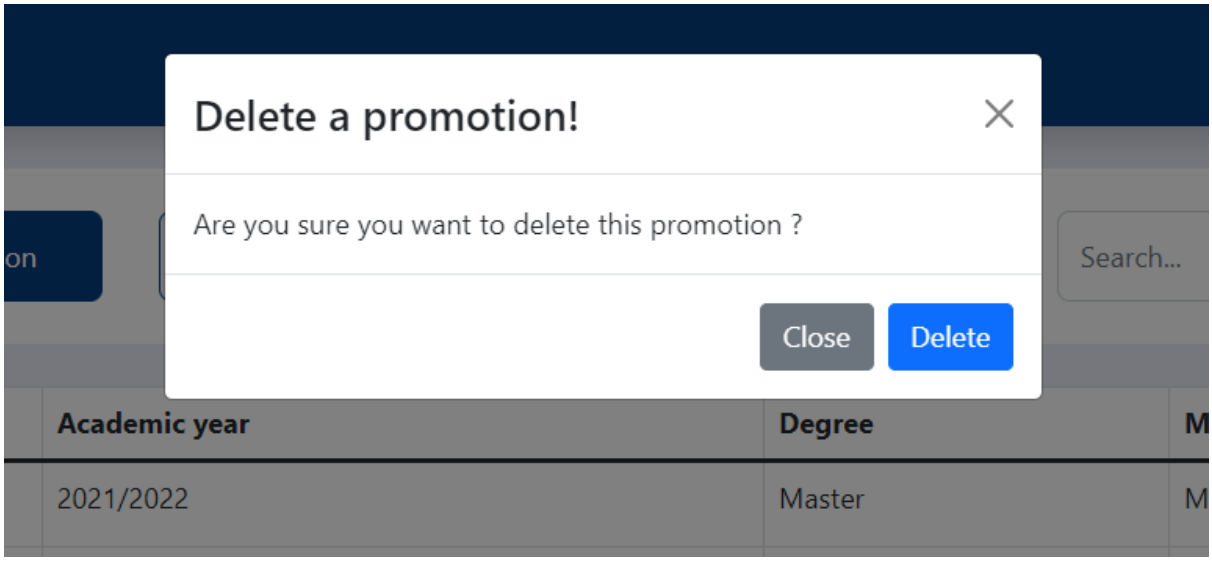


Figure 4. 13 Delete promotion interface

4.2.6 Groups interface:

This interface is accessed by admin from the promotions page, each promotion has its specific groups interface. When accessed the first time, it shows two areas: One for adding a new group based on the number of groups previously specified for the current promotion, and the other is for showing the current existing groups. Each time a group is added, it is reflected in the second area.

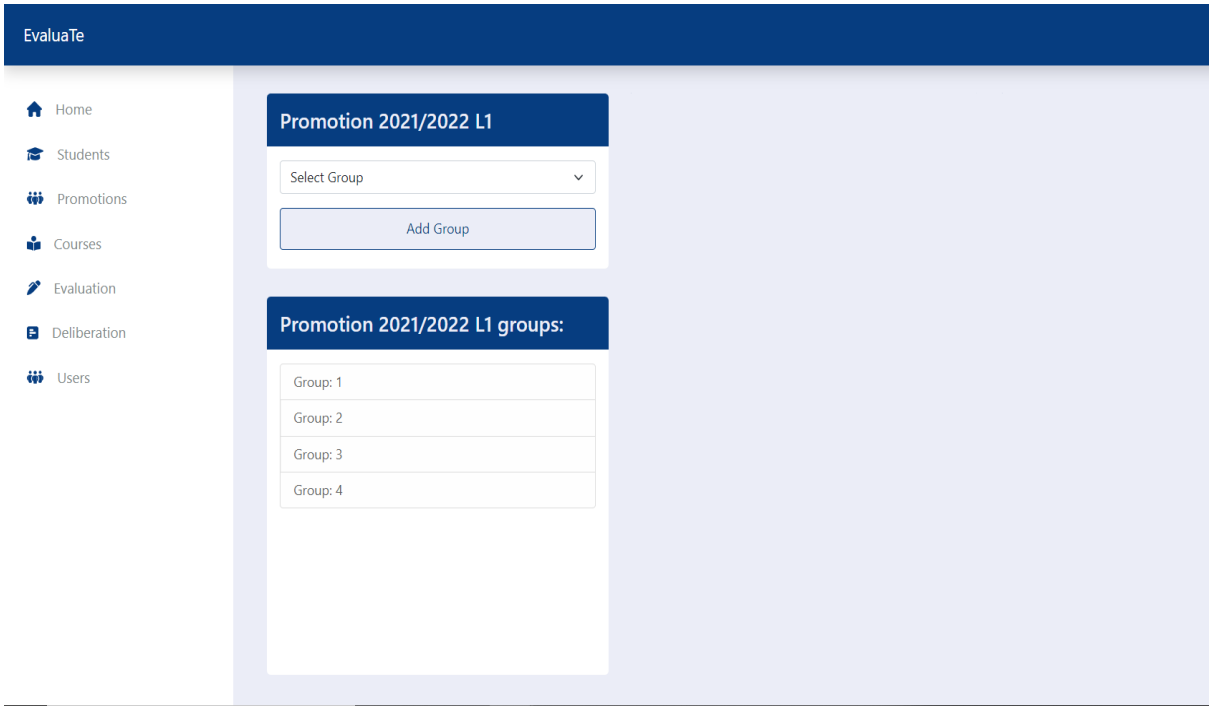


Figure 4. 14 Promotion groups interface

When a specific group is clicked, another two areas would appear on the right side. The first one would show the list of students for the current promotion with a select field attached to each student. The admin selects the students to add to the specified group and clicks update. The list of added students to this group appears in the second area with a delete icon attached to each student. The added students select fields will appear disabled to ensure that a specific student would exist in a single group.

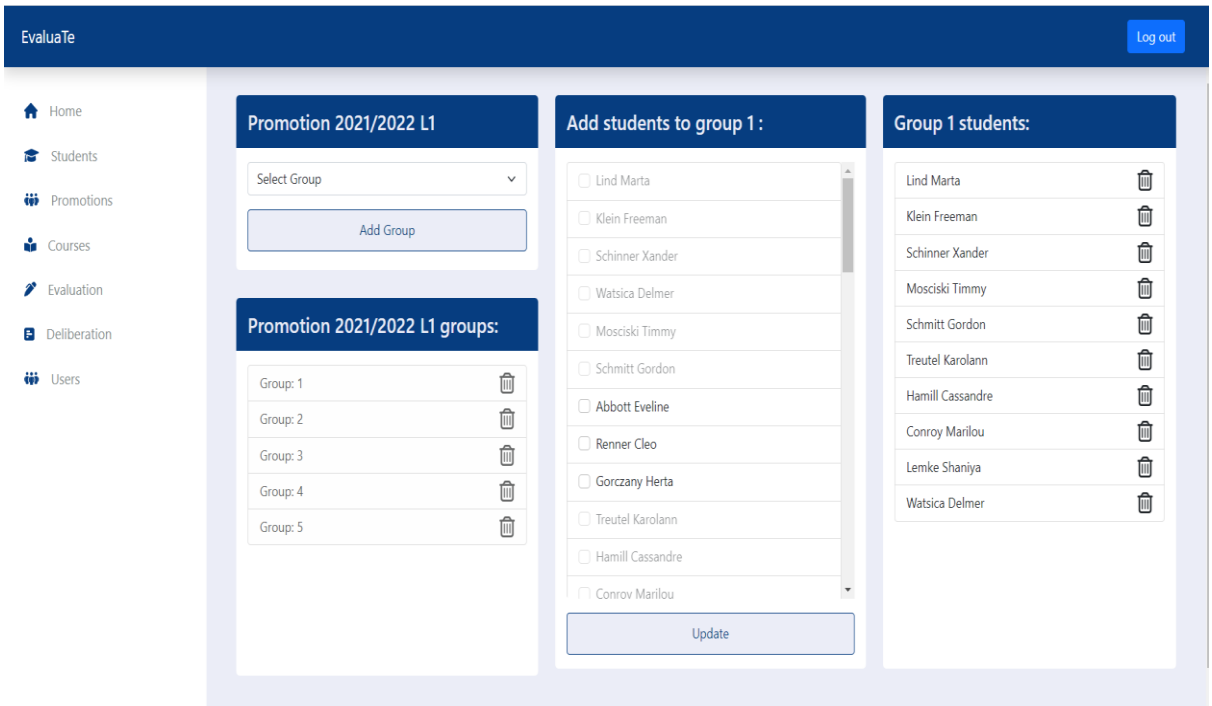


Figure 4. 15 Adding students to groups

4.2.7 Courses interface:

When admin accesses this interface, it shows, first of all, an area containing two select fields (Major and Semester), and a button. When both fields get specified and the apply button is clicked, a list of courses per semester table appears appropriately.

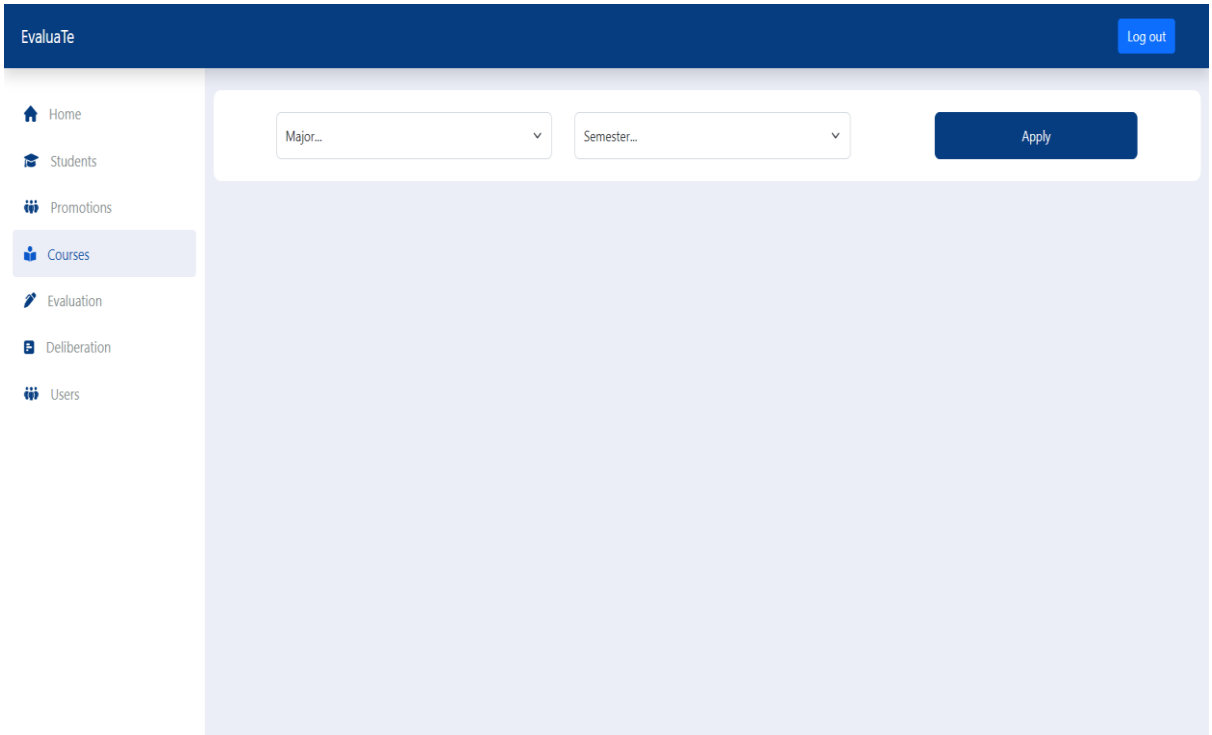


Figure 4. 16 Courses interface

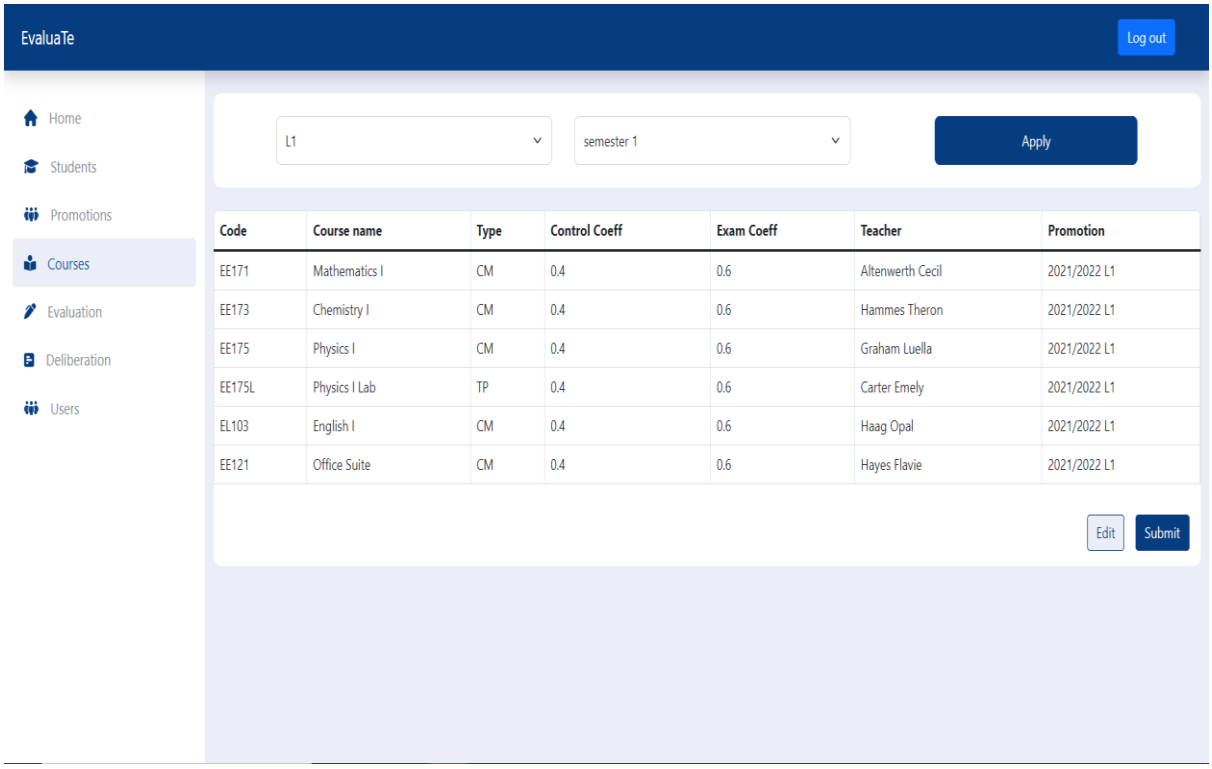


Figure 4. 17 List of courses per semester

When the list is shown, the user can edit the following fields for each course:

- Control and Exam coefficients: which are used later to calculate course average for each student enrolled.
- Teacher: a select field contains the list of teacher users registered in the system used to assign a teacher to each course. Teachers would be responsible for the evaluations of only courses assigned to them.
- Promotion: each course is assigned a promotion, meaning enrolling each promotion in the courses that are supposed to be taken for the current academic year.

The above fields are all required for other sections to work properly. After updating, the admin clicks on submit to save the updates to our system.

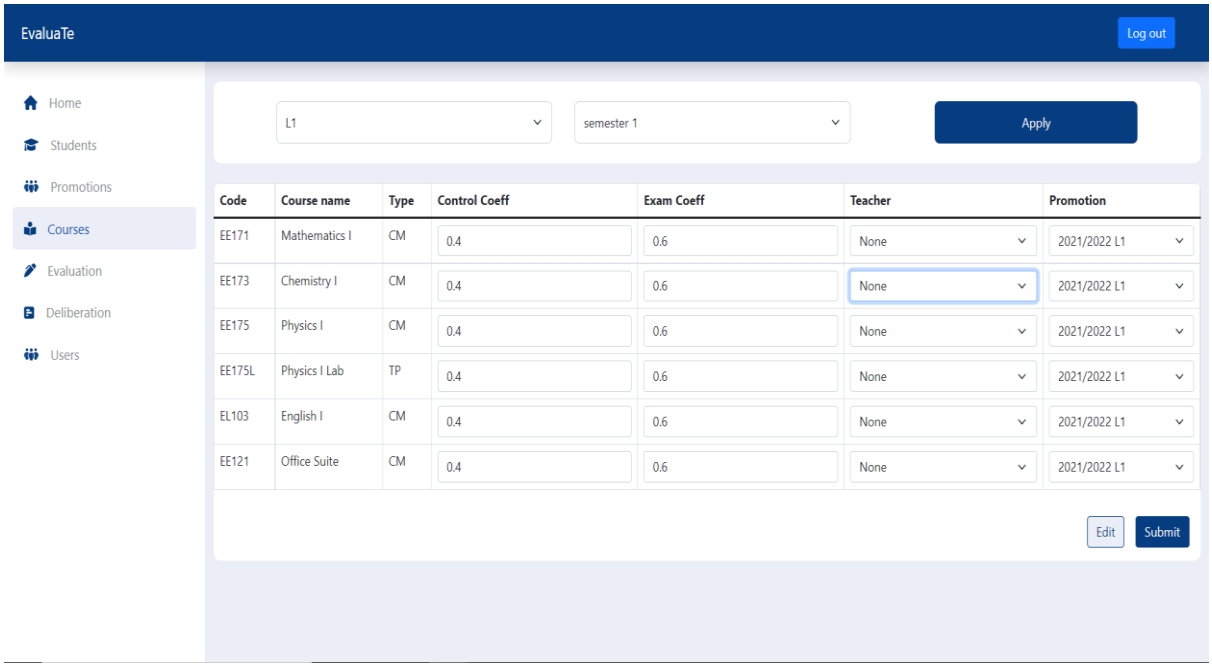


Figure 4. 18 Updating courses related informations

4.2.8 Evaluations interface:

Working with This interface is mainly the responsibility of teacher users. When accessed, an area section is shown with three select fields (course, evaluation type, and group).

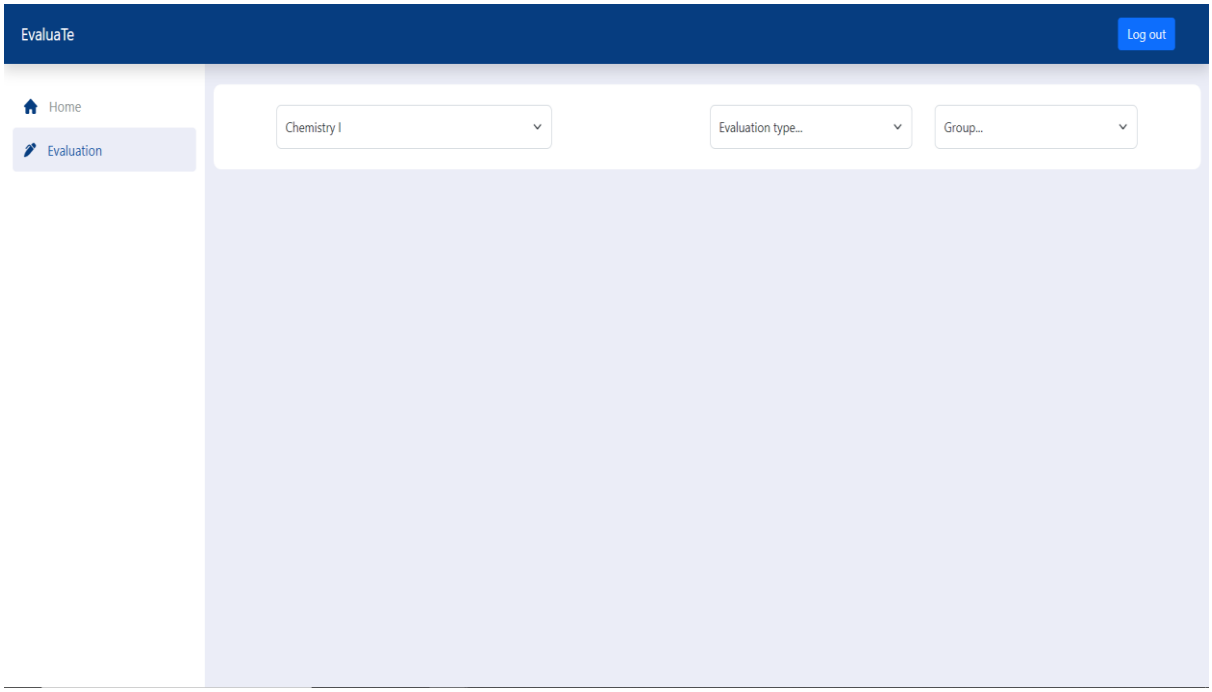


Figure 4. 19 Evaluations interface

After selecting all fields, a list table is shown which would help the teacher to grade students of the specified group enrolled in the specified course for the specified type of evaluation (control, exam…).

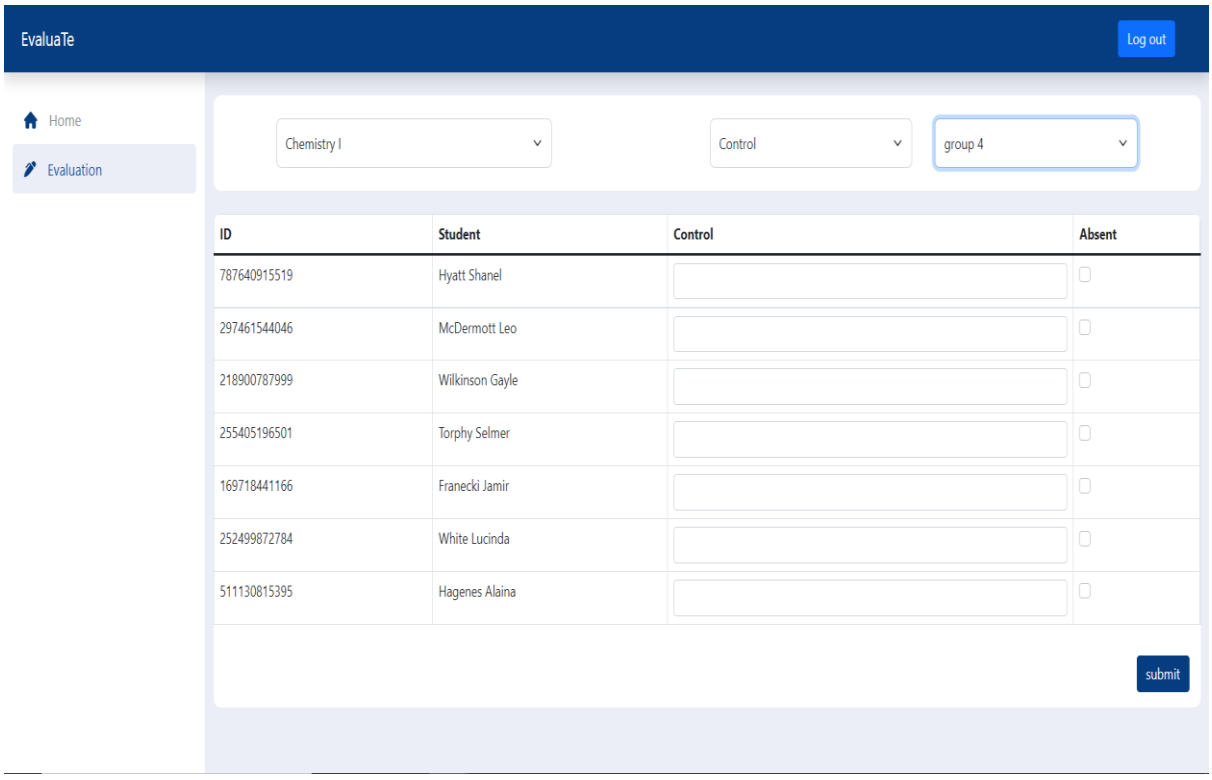


Figure 4. 20 Students grading process

After filling the fields for each student, the user clicks submit to update the database with students' grades for each course.

4.2.9 Deliberation interface:

This interface is for showing the final results and averages for all students. When admin accesses this page, he would select a promotion, a group, and clicks apply.

Then, the results table for the specified group students appears with each course average, each semester average, annual average, and a field for final decision if the student passes the year successfully or not.

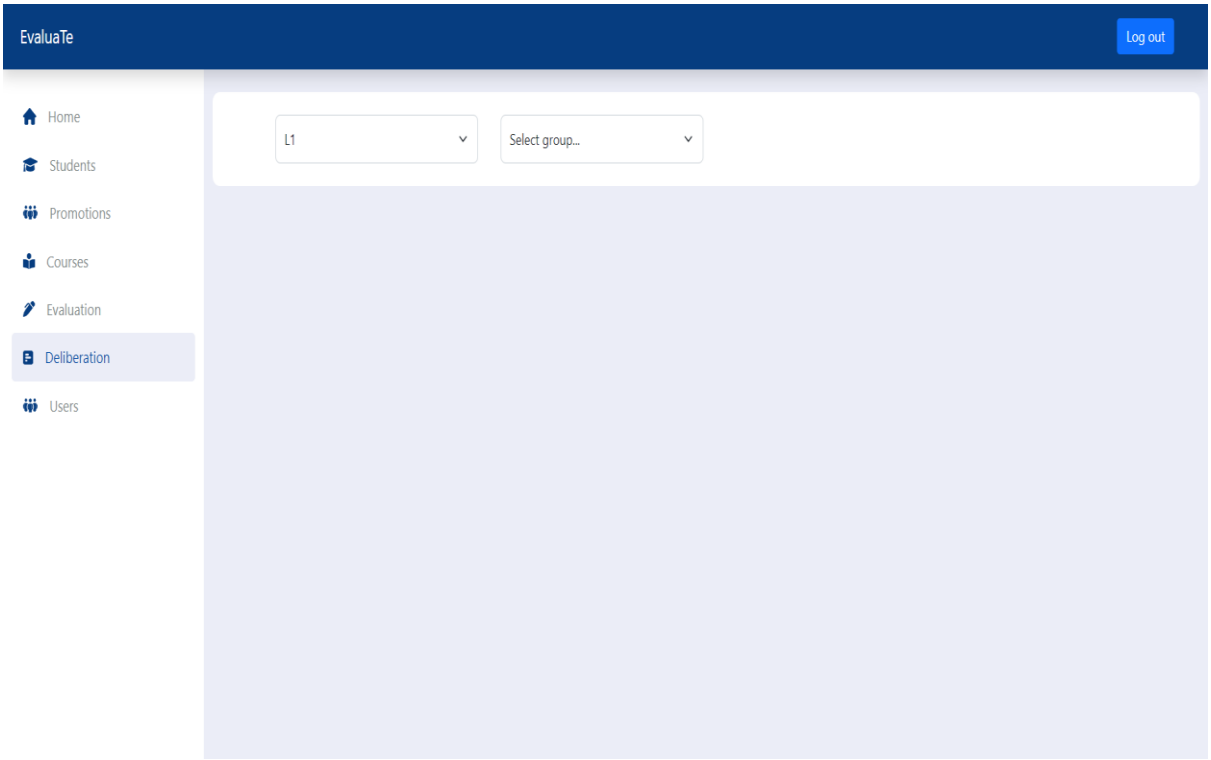
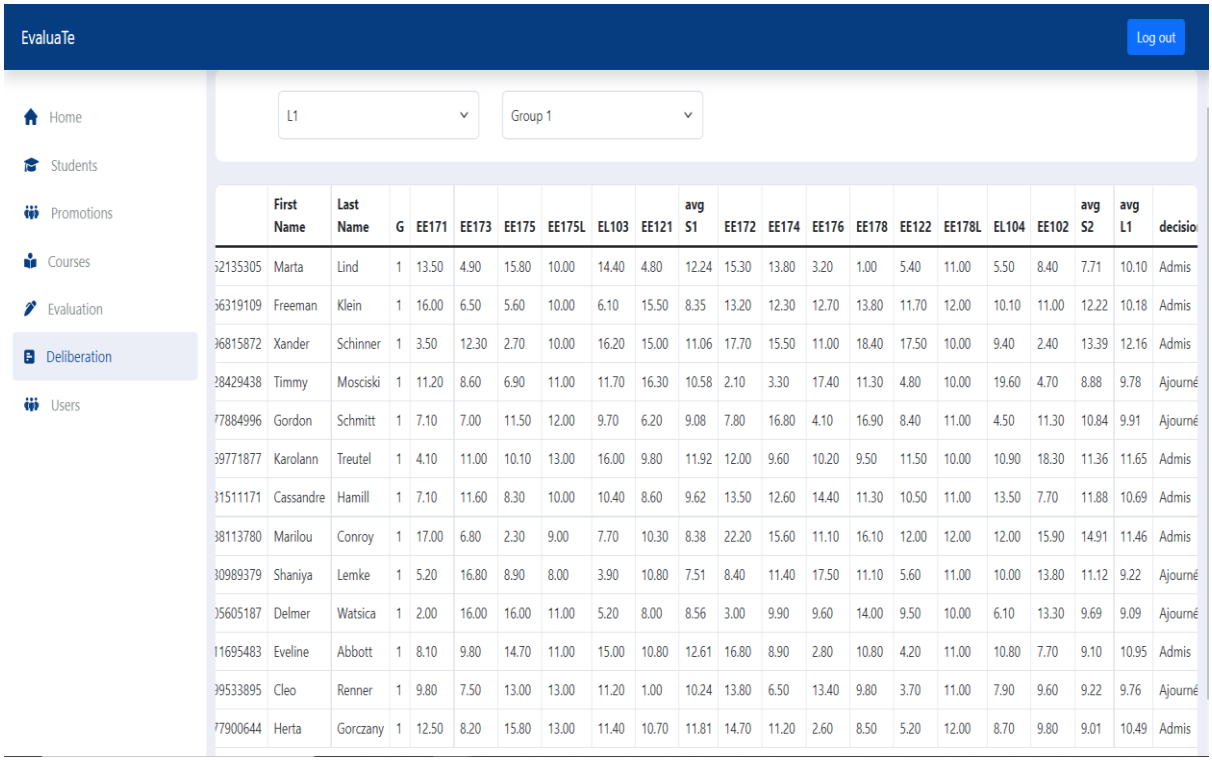


Figure 4. 21 Deliberation interface



	First Name	Last Name	G	EE171	EE173	EE175	EE175L	EL103	EE121	avg S1	EE172	EE174	EE176	EE178	EE122	EE178L	EL104	EE102	avg S2	avg L1	decisio
52135305	Marta	Lind	1	13.50	4.90	15.80	10.00	14.40	4.80	12.24	15.30	13.80	3.20	1.00	5.40	11.00	5.50	8.40	7.71	10.10	Admis
56319109	Freeman	Klein	1	16.00	6.50	5.60	10.00	6.10	15.50	8.35	13.20	12.30	12.70	13.80	11.70	12.00	10.10	11.00	12.22	10.18	Admis
96815872	Xander	Schinner	1	3.50	12.30	2.70	10.00	16.20	15.00	11.06	17.70	15.50	11.00	18.40	17.50	10.00	9.40	2.40	13.39	12.16	Admis
28429438	Timmy	Mosciski	1	11.20	8.60	6.90	11.00	11.70	16.30	10.58	2.10	3.30	17.40	11.30	4.80	10.00	19.60	4.70	8.88	9.78	Ajourné
77884996	Gordon	Schmitt	1	7.10	7.00	11.50	12.00	9.70	6.20	9.08	7.80	16.80	4.10	16.90	8.40	11.00	4.50	11.30	10.84	9.91	Ajourné
59771877	Karolann	Treutel	1	4.10	11.00	10.10	13.00	16.00	9.80	11.92	12.00	9.60	10.20	9.50	11.50	10.00	10.90	18.30	11.36	11.65	Admis
31511171	Cassandre	Hamill	1	7.10	11.60	8.30	10.00	10.40	8.60	9.62	13.50	12.60	14.40	11.30	10.50	11.00	13.50	7.70	11.88	10.69	Admis
38113780	Marilou	Conroy	1	17.00	6.80	2.30	9.00	7.70	10.30	8.38	22.20	15.60	11.10	16.10	12.00	12.00	12.00	15.90	14.91	11.46	Admis
30989379	Shaniya	Lemke	1	5.20	16.80	8.90	8.00	3.90	10.80	7.51	8.40	11.40	17.50	11.10	5.60	11.00	10.00	13.80	11.12	9.22	Ajourné
05605187	Delmer	Watsica	1	2.00	16.00	16.00	11.00	5.20	8.00	8.56	3.00	9.90	9.60	14.00	9.50	10.00	6.10	13.30	9.69	9.09	Ajourné
11695483	Eveline	Abbott	1	8.10	9.80	14.70	11.00	15.00	10.80	12.61	16.80	8.90	2.80	10.80	4.20	11.00	10.80	7.70	9.10	10.95	Admis
99533895	Cleo	Renner	1	9.80	7.50	13.00	13.00	11.20	1.00	10.24	13.80	6.50	13.40	9.80	3.70	11.00	7.90	9.60	9.22	9.76	Ajourné
77900644	Herta	Gorczany	1	12.50	8.20	15.80	13.00	11.40	10.70	11.81	14.70	11.20	2.60	8.50	5.20	12.00	8.70	9.80	9.01	10.49	Admis

Figure 4. 22 Showing the results of students

4.3 Conclusion:

In this chapter, we have given a brief explanation about the various user interfaces of our application's front-end, an how different users may interact with the system, access the necessary information, and submit the required data in a simplified and convenient way so that different tasks and scenarios will be accomplished effectively.

General Conclusion

In this report, we have presented the different steps to design and develop a web application intended for managing the process of students' evaluation in our institute. In order to realize this project, we have gathered and analyzed different problems encountered by the administration and areas that need improvements.

The major objective of our project is to facilitate the work of the administration members including the security of data and the rapid access to it, and the precision of the generated results. We have implemented a web application, so it can be accessed from anywhere at any time using internet.

This project, which falls in the field of the design and implementation of an information system, was very interesting and allowed me to become familiar with new concepts, and to improve my knowledge and skills in the field of web development.

Future Works

This web application that we have built may be enhanced more in the future by including other functionalities and improving some areas. The generation of other types of documents like students' final transcripts is one example. Providing access to students where they can see their results and different announcement would be great to add. Also, it is important to further improve the security and performance of this application.

Webography

[1] https://techterms.com/definition/web_application. (20/07/2022)

[3] https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58. (20/07/2022)

[4] https://www.techopedia.com/definition/27927/nodejs. (20/07/2022)

[5] https://www.guru99.com/node-js-express.html. (20/07/2022)

[6] https://searchdatamanagement.techtarget.com/definition/MongoDB (20/07/2022)

[7] https://www.c-sharpcorner.com/article/what-and-why-reactjs/. (20/07/2022)

[8] https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework). (20/07/2022)

[9] https://techterms.com/definition/web_browser. (20/07/2022)

[10] https://www.techopedia.com/definition/1892/hypertext-markup-language-html. (20/07/2022)

[11] https://www.techopedia.com/definition/26268/cascading-style-sheet-css. (20/07/2022)

[12] https://www.techopedia.com/definition/3929/javascript-js. (20/07/2022)

[13] https://en.wikipedia.org/wiki/TypeScript. (20/07/2022)

[14] https://www.techopedia.com/definition/20810/modeling-language. (20/07/2022)

[15] http://www.agilemodeling.com/essays/umlDiagrams.htm. (20/07/2022)

[16] https://www.smartdraw.com/use-case-diagram/ (20/07/2022)

[17] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (20/07/2022)

[18] https://aws.amazon.com/relational-database/. (20/07/2022).

[19] https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data. (20/07/2022)

[20] https://www.tutorialspoint.com/mongodb/mongodb_relationships.htm (20/07/2022)

Bibliography

[20] G. Booch, J. Rumbaugh, I. jacobson, The Unified Modelling Language Reference Manual, 2nd ed. 2005

[21] Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database Systems, 6th Ed. (2010).