IBU | International Burch University

IT 309 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

Grading Management System

Prepared by:
Tarik Maljanović
Melisa Geca
Kerim Šabić

Proposed to:
Nermina Durmić, Assist. Prof. Dr.
Naida Fatic, Teaching Assistant

March, 2024

TABLE OF CONTENTS

1. Introduction

1.1. About the Project

The project we are working on is a web application for grading management. The target audience are educational institutions, their staff members and students. It makes tracking, managing and maintaining grades across different courses easy. By assigning professors and students to their designated courses, each student has an organized environment for tracking their progress of a course, professors maintain the grading system for each of their courses, and administrators manage the users of the system.

1.2. High-level Plan

For our project we used the Scrum Agile methodology of high-level planning. Thus, the following section features a product roadmap and a release plan.

Roadmap

-	Q1	Q2
Timeframe	First quarter of May	First quarter of June
Release Name	MVP (Minimum Viable Product)	Final Version
Goal	Goal is to make the most critical features available on the first release.	Goal is to expand the application with additional features for every user type.
Features	Assigning students and professors to courses, management of courses, assignments and grades, availability and maintainability of user information.	Increase security measures by giving the user the ability to change their password. Included additional information about the system and the institution that uses it, along with a profile for each user, along with additional details about the courses and their content.
Metrics	Use by at least 3 schools/ universities in Sarajevo	Heighten the security of the system, offer additional contact and information about the system and the institution for increased personalization.

Release Plan

The release plan describes how the work is organized across sprints.

Sprint 1

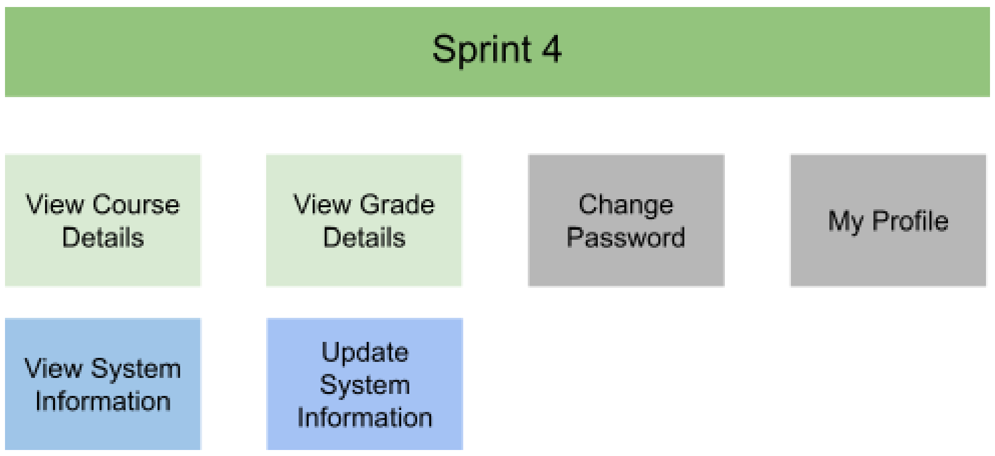
- Login Logout
- Create User
- Delete User
- View User
- View Users

Sprint 2

- Create Course
- Delete Course
- Update Course
- View Course
- View Courses
- Assign Student to Courses
- Unassign Student to Courses
- Assign Professor to Courses
- Unassign Professor to Courses

Sprint 3

- Create Assignment
- Delete Assignment
- Update Assignment
- Grade Student
- View Grade



1.3. Project Requirements

The following section features the product backlog of our project. It is a prioritized and detailed list of features.

1. **Login and Logout**

   As a user, I want to be able to log into the system, in order to access the features and data necessary for accomplishing desired tasks. And I would like to leave the system when done by logging out.

   Acceptance Criteria:
   - The user enters the email address in the “Email” input field
   - The user enters the password in the “Password” field
   - Both fields are required for the login
   - The email needs to be valid
   - The password needs to be at least 8 characters long
   - If both credentials are correct, the user is logged in
   - If the credentials are not correct, the user is met with an error message

2. **Create Professor/Student**

   As an administrator, I want to be able to create accounts for each student and professor, in order to grant access to people who are part of the institution.

   Acceptance Criteria:
   - The administrator clicks on the “Add User” button
   - The administrator is met with a form appropriate to the type of user they are creating
   - The type of user is chosen in the dropdown menu, offering the “Professor” and “Student” role
   - Every field needs to be filled out
   - The Email needs to be in the correct format
   - The Password needs to be at least 8 characters long
   - When the administrator clicks on the “Create User” button, a new user is created if every field is filled out and all the input is valid

- When the administrator clicks on the “Create User” button, and if an input field is left empty or some input is invalid, an error message appears
- Upon successful creation, a success message appears

3. **Delete Professor/Student**

   As an administrator, I want to be able to delete user accounts for each student and professor, in order to remove users which are not part of the institution.

   Acceptance Criteria:
   - The administrator selects a user and clicks on the “Delete User” button
   - A prompt appears asking the user to either confirm or cancel the action
   - If the administrator clicks “Confirm Deletion”, the user is deleted from the system
   - If the administrator clicks the “Cancel” button, the prompt closes
   - Upon successful deletion, a success message appears

4. **Update Professor/Student**

   As an administrator, I want to be able to update information about each user, in case their information needs to be changed.

   Acceptance Criteria:
   - The administrator selects a user and clicks on the “Edit User” button
   - A form appears with all the information which can be changed
   - When the administrator clicks the “Save Changes” button, and if every field is valid, the user’s data is updated
   - If any field is left empty or the input is invalid, an error message is shown
   - A success message is displayed upon a successful update

5. **View List of Professors/Students**

   As an administrator/professor, I want to be able to see a list of students and professors of a course, in order to gain insight about which user is assigned to which course.

   Acceptance Criteria:
   - Each course should have a “Participants” tab where a list of all Students and Professors of the course can be found
   - Each entry in the list should display the following information: ID, First Name, Last Name, Role
   - Each entry is clickable and will display details about the selected user

6. **View Professor/Student**

   As an administrator/professor, I want to be able to see all the details of a student or professor of a course, in order to gain information about them.

   Acceptance Criteria:
   - When selecting a user from a list of Course Participants, their details should be displayed
   - The window should display the following information: ID, First Name, Last Name, Email, Role
   - The window should also feature the “Delete User” and “Edit User” buttons

7. **Create Course**

   As an administrator, I want to be able to create a new course, in order to contain all the information and grading related to the course in one place.

Acceptance Criteria:

- When the administrator clicks the “Create Course” button, form should appear asking the administrator to enter the required information
- The information required for creating a course is: Course Title, Description, Number of ECTS points, Course Code
- If every input field is filled out and if all the input is valid, the course is created
- If an input field is left empty or if any input is invalid, an error message is displayed
- A success message should be displayed upon successful creation

8. **Update Course**

   As an administrator, I want to be able to update the course data, in order to change the information when needed.

   Acceptance Criteria:

   - Inside the course page, the administrator clicks the “Edit Course” button to see the form for editing course information
   - Each field contains information about the course and is editable
   - The administrator clicks the “Save Changes” button to save the changes
   - If all input fields have been filled out and if there is no invalid input, the course is updated and a success message is displayed
   - If there is an empty field or if any input is invalid, and error message appears

9. **Delete Course**

   As an administrator, I want to be able to delete the course from the system, in order to remove courses that are no longer available.

   Acceptance Criteria:

   - The administrator clicks on the “Delete Course” button and is met with a prompt which asks the administrator to either confirm or cancel the action
   - If the administrator clicks the “Confirm Deletion” button, the course is deleted and a successful message is displayed
   - If the administrator clicks the “Cancel” button, the prompt is closed

10. **View List of Courses**

    As a user, I want to access a comprehensive list of all available courses, so that I can have an overview of the courses offered within the institution.

    Acceptance Criteria:

    - When accessing the "Courses" section of the platform, a list of all courses is displayed.
    - Each course entry in the list includes the following information: Course Title, Number of ECTS points, Course Code.
    - The list is easily navigable and scrollable.
    - Each course entry is clickable, allowing for detailed exploration of individual courses.
    - Clicking on a course entry provides further information about the selected course, such as enrolled students, assigned professors.

11. **View Course Details**

    As a user, I want to access detailed information about a specific course when selecting it from the list of courses, in order to see all details and participants.

Acceptance Criteria:

- When you click on a course from a list of courses, a popup window displays all of the course details.
- The information displayed is the following: Course Code, Course Title, Description, Number of ECTS Points.
- It also features a list of all students and professors assigned to the course

12. Assign Professor to Course

As an administrator, I want to assign a professor to a specific course,
to make the course available to the professor on the system

Acceptance Criteria:

- Within the course management interface, there should be an option to assign a professor to the course.
- When selecting the option to assign a professor, a list of available professors within the institution is presented.
- The list of available professors includes their IDs, First Names, Last Names, and possibly other relevant information for identification.
- The administrator can search and filter the list of professors by various criteria, such as name or department, to facilitate finding the appropriate instructor.
- After selecting a professor from the list, they are assigned to the course, and their details (ID, First Name, Last Name, Role) are displayed alongside the course information.
- Once assigned, the professor gains access to the course management features associated with their role in the course.
- Upon successful assignment, a confirmation message appears, acknowledging the professor's assignment to the course.

13. Unassign Professor from Course

As an administrator, I want to remove a professor from a specific course, so that the course can be reassigned to another professor.

Acceptance Criteria:

- Within the course management interface, there should be a designated option for unassigning a professor from a course.
- Upon selecting this option, the administrator is prompted to confirm the unassignment action.
- If confirmed, the professor is removed from the course, and their association with the course is terminated.
- After unassigning the professor, the course remains accessible for further management and potential re-assignment.
- If desired, the administrator can proceed to assign another professor to the course or leave it without an assigned instructor.
- A confirmation message is displayed upon successful unassignment, confirming the removal of the professor from the course.

14. Assign Student to Course

As an administrator/professor, I want to enroll a student in a specific course,
in order to enable the student's access and participation in course activities.

Acceptance Criteria:

- Within the course management interface, there should be an option to assign students to the course.
- Both administrators and professors have access to this functionality.
- Upon selecting the option to assign a student, a list of available students within the institution is presented.
- The list includes pertinent details for each student, such as their IDs, First Names, and Last Names.
- Administrators and professors can efficiently search and filter the list of students by different criteria (e.g., name, department) to facilitate the selection process.
- Once a student is chosen from the list, they are officially enrolled in the course, and their details are associated with the course information.
- The assigned student gains access to course details and related functionalities, allowing them to participate fully in the course.
- A confirmation message is displayed upon successful enrollment, confirming the student's inclusion in the course.

15. Unassign Student from Course

As an administrator/professor,I want to remove a student from a specific course, in order to manage student enrollments and course participation effectively.

Acceptance Criteria:

- Within the course management interface, there should be an option to unassign a student from the course.
- Upon selecting the option to unassign a student, the system prompts for confirmation of the action.
- If confirmed, the student is removed from the course, and their association with the course is terminated.
- After unassigning the student, the course remains accessible for further management and potential re-enrollment.
- If desired, administrators or professors can re-enroll the student in the course or leave the seat vacant.
- A confirmation message is displayed upon successful unassignment, confirming the removal of the student from the course.

16. Create Assignment

As a professor, I want to create assignments for students within a course, in order to assess their understanding of the course material and track their progress.

Acceptance Criteria:

- When accessing the "Courses" section of the platform, there should be an option for professors to create assignments.
- Professors have exclusive access to this functionality.
- When creating an assignment, professors should be able to provide a title, description, due date, and any additional instructions or attachments.
- The assignment creation form should validate input fields, ensuring that essential information is provided and the due date is set appropriately.
- Professors can preview the assignment before finalizing it, ensuring accuracy and completeness.
- A confirmation message is displayed upon successful creation of the assignment, informing the professor that the task has been completed.

17. **Update Assignment**

As a professor, I want to modify existing assignments within a course, in order to accommodate changes in course requirements or provide additional instructions.

Acceptance Criteria:

- Within the course management interface, professors should have the option to update assignments they have created.
- Professors have exclusive access to this functionality.
- When updating an assignment, professors should be able to edit the title, description, due date, and any other relevant details.
- The assignment update form should validate input fields, ensuring that essential information is provided and the due date is adjusted appropriately if necessary.
- Professors can preview the updated assignment before finalizing the changes, ensuring accuracy and completeness.
- A confirmation message is displayed upon successful update of the assignment, informing the professor that the changes have been saved.

18. **Delete Assignment**

As a professor, I want to delete existing assignments within a course in order to remove outdated assignments

Acceptance Criteria:

- Within the course management interface, professors should have the option to delete assignments they have created.
- Professors have exclusive access to this functionality.
- When deleting an assignment, professors should be able to be shown a window with the confirmation of his action.
- Professors should be able to select the option to delete or to cancel the deletion from the pop-up window.
- Upon confirmation the assignment should be permanently deleted from the system.
- Any associated grades related to the assignment should be removed.

19. **Grade Student**

As a professor, I want to be able to enter a grade for a student for a specific assignment in order to mark their knowledge for that particular assignment.

Acceptance criteria:

- Professor click on the “Courses” in the navigation menu, list of all courses is displayed
- Professor click on the wanted course, where details about the course are displayed
- Clicking on the create create grade, form is displayed where the necessary information about the assignment is entered
- In the same window professors can see the list of student in that course
- Professor elects a student that needs to be graded and enters the grade
- Once done professor click the save button and all the changes are saved to the system

20. **View Grades**

As a student, I want to be able to see my grades, in order to access the final results of my assignments.

Acceptance Criteria:

- Student clicks on a “Courses” in the navigation, list of all courses is displayed
- Each course entry is clickable, allowing for detailed exploration of individual courses.
- Clicking on a course entry provides further information about the selected course
- The detailed view of a course should include a list of assignments for that course along with the grades received for each assignment.
- The assignments should be clickable, enabling the student to view additional details about each assignment.

21. **View Grade Details**

As a student, I want to be able to access the single grade from the assignment, in order to see detailed information about the assignment and the grade.

Acceptance Criteria:

- Upon clicking on the particular course from the list of courses, student click on the wanted assignment
- The window should display with the detailed information about the selected assignment and the grade
- Window consists of the title of the assignment, description of the grade, date and the grade itself with a close button.
- Accessing the grade and assignment details is restricted to authenticated student with permission to see their grades

22. **Change password**

As a user, I want to be able to change my password, in order to access the system with a new password.

Acceptance criteria:

- The user enters the system and selects the “My Profile” icon in the navigation.
- When the user click on the “Change password” button a form should appear
- The form consists of 3 fields: Old password, New Password, Repeat new password
- All fields need to be filled out
- The new password needs to be at least 8 characters long
- If all credentials are correct, the user is clicks Change password
- Confirmation window appears with option to reset the password or to cancel the reset
- Upon confirmation the new password should be replaced with the new password
- User will be logged out and can log in with the new password.

23. **My Profile**

As a user, I want to be able to access my profile page within the system, in order to see my personal information.

Acceptance criteria:

- The user enters the system and selects the “My Profile” icon in the navigation.
- The profile page should display the user's personal information, including their first name, last name, email address, role, and any other relevant details.
- Only administrator should be able to edit their personal details
- All users should be able to change their password

24. **View System Information**

As a user, I want to be able to see the information about the system, in order to get more information about the system.

Acceptance criteria:

- User click on the “About” in the navigation menu where new page is displayed with the correct information
- User is able to see necessary information about the system such as: name of the system, contact information such as phone number, email address

25. **Update System Information**

As an administrator, I want to be able to edit the information in the View System section of the website, in order to keep the contact information up to date.

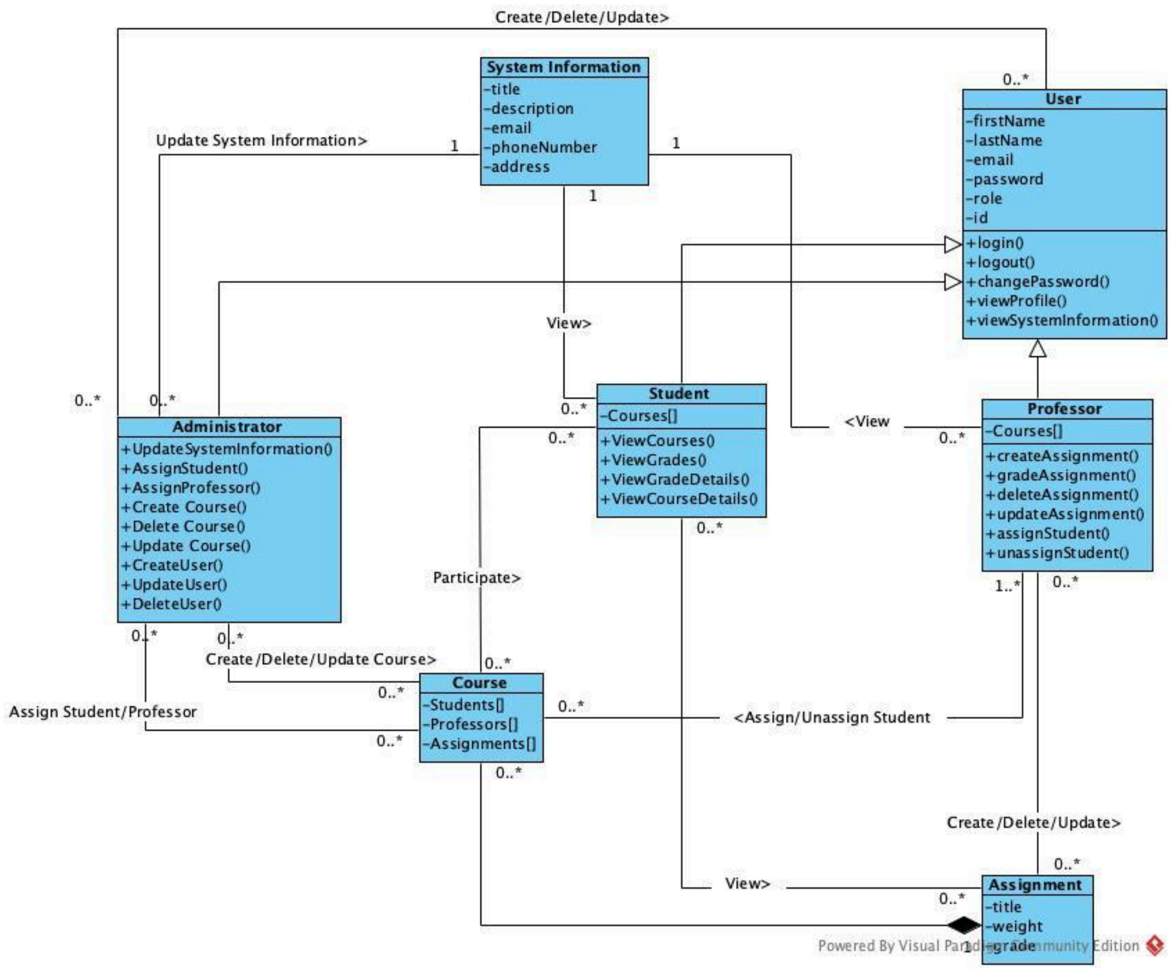
Acceptance Criteria:

- Administrator clicks on the “About” in the navigation menu where new page is displayed with the correct information about the system
- Administrator sees the system and contact information
- Clicking on the edit button, all fields become editable
- Once wanted fields are edited administrator clicks on the save button where the appropriate confirmation message is displayed
- Once the confirmation is approved the edited information is saved to the system

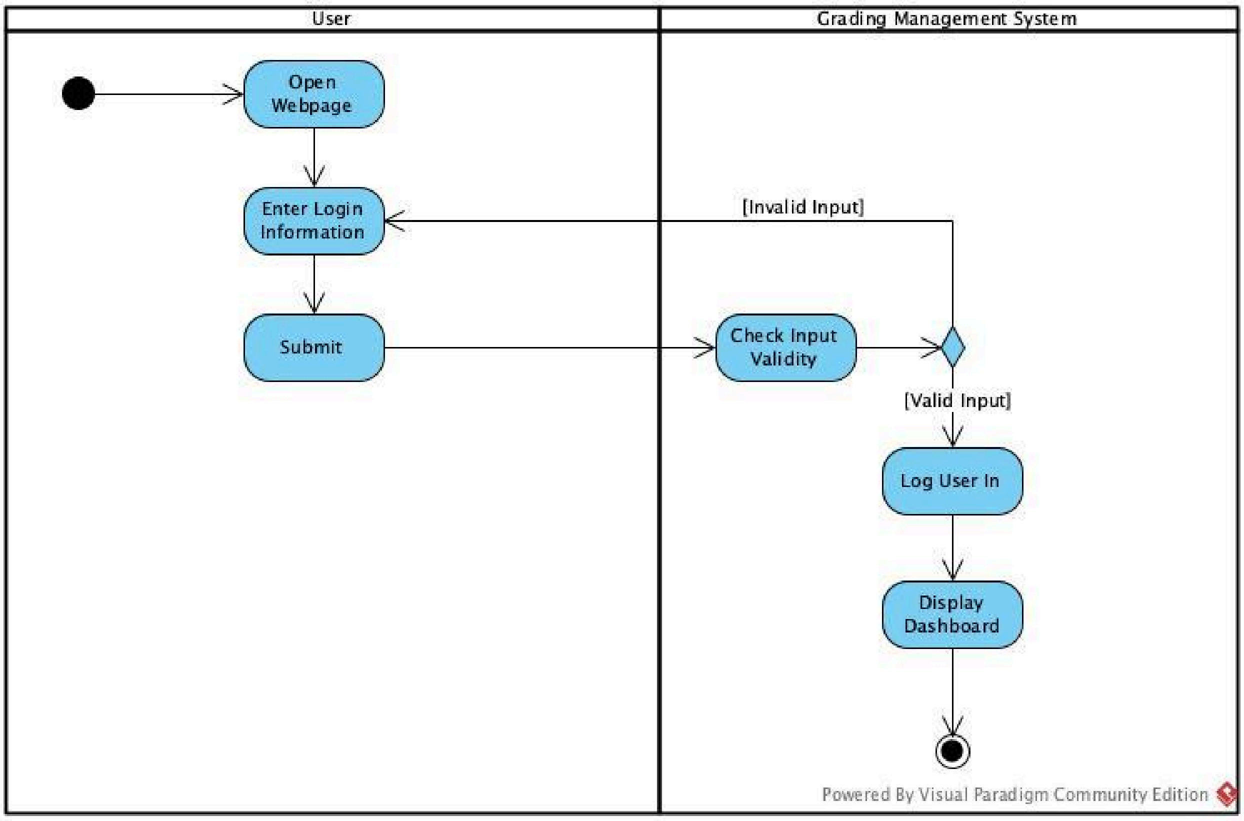
1.4. UML diagrams

The following section describes the three most important features of the project, which are described using Sequence and Activity Diagrams, along with the structure of the product being described using a Class diagram.

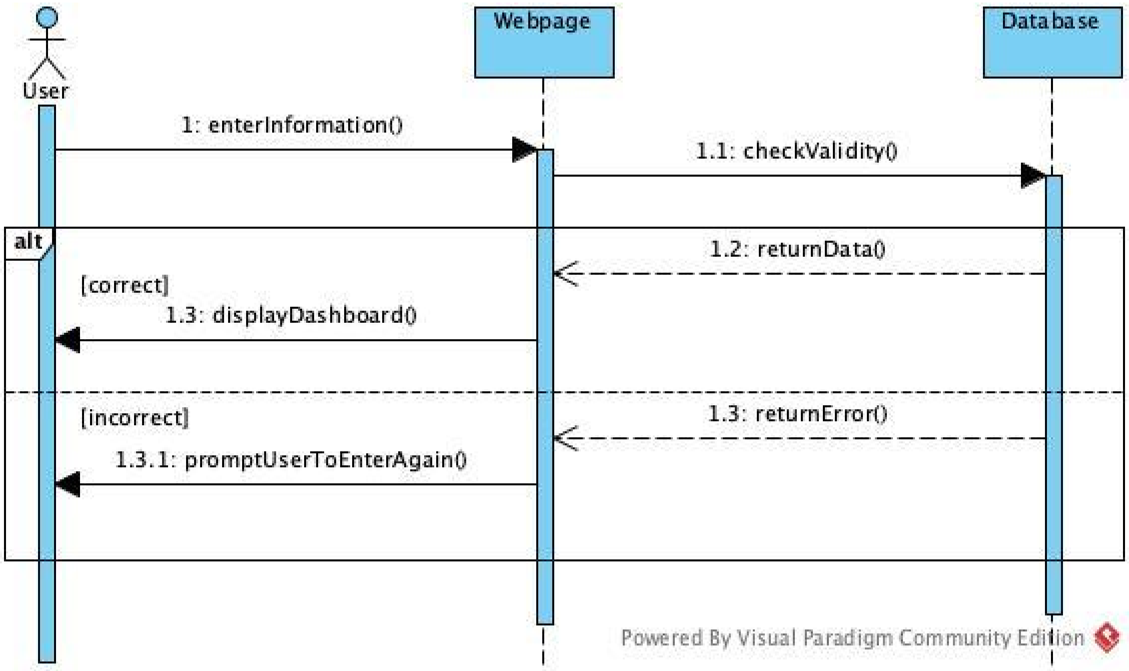
Class Diagram:



Login

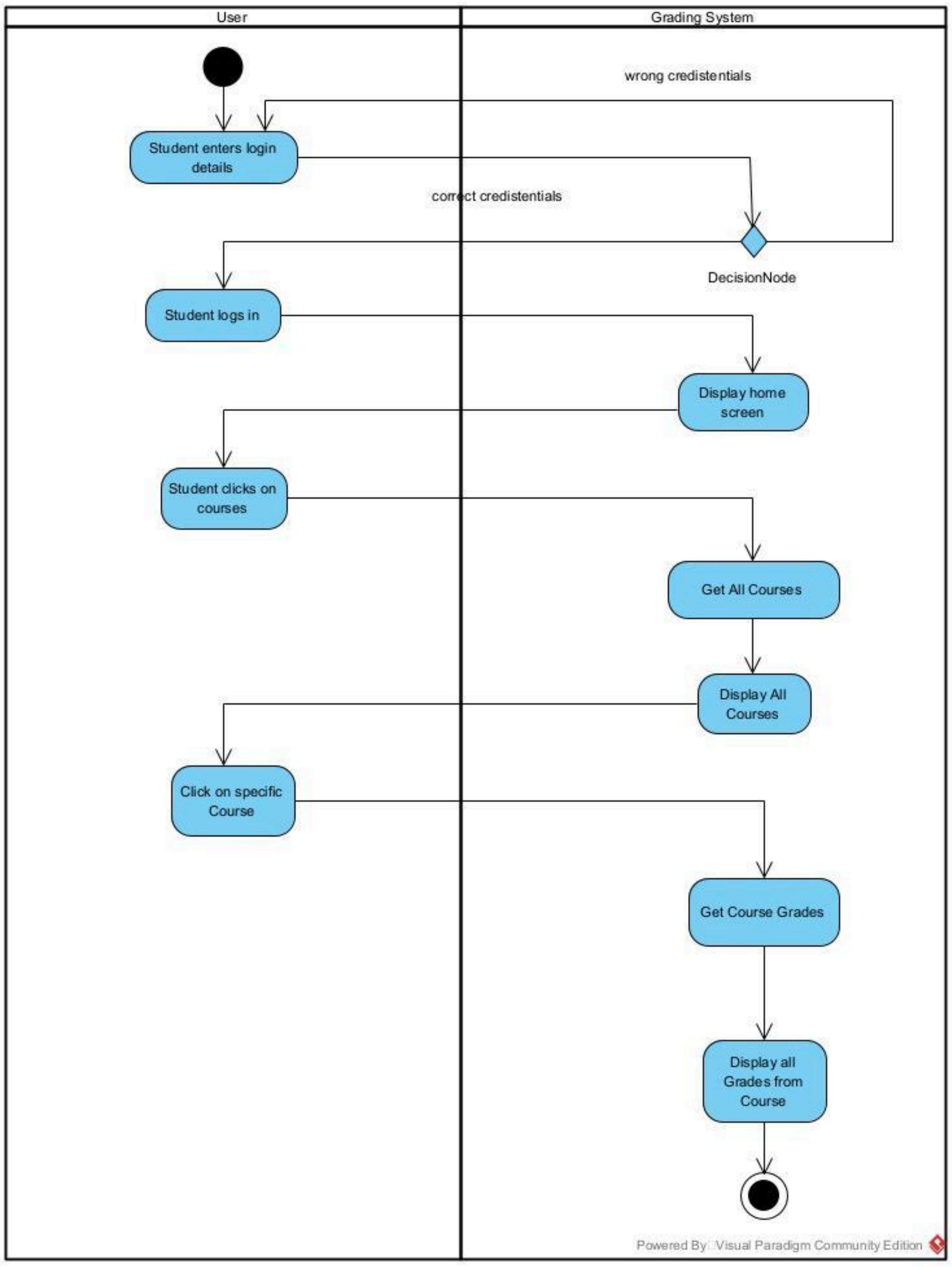


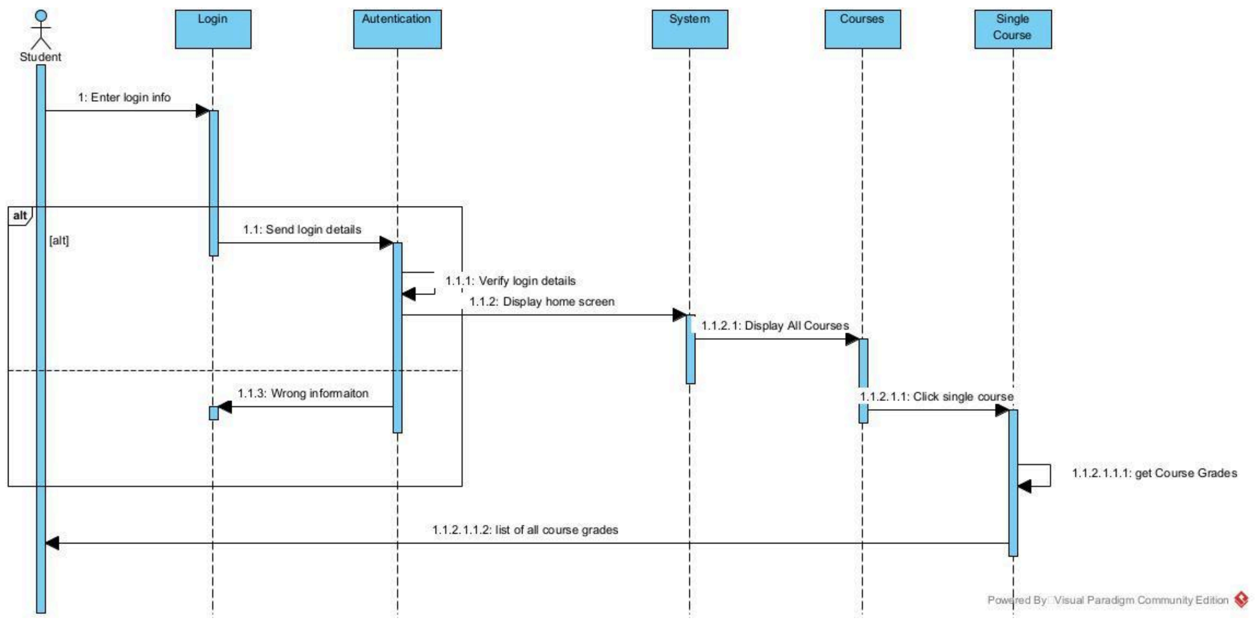
User
Grading Management System
Open Webpage
Enter Login Information
Submit
Check Input Validity
[Invalid Input]
[Valid Input]
Log User In
Display Dashboard
Powered By Visual Paradigm Community Edition



User
Webpage
Database
1: enterInformation()
1.1: checkValidity()
alt
1.2: returnData()
[correct]
1.3: displayDashboard()
[incorrect]
1.3: returnError()
1.3.1: promptUserToEnterAgain()
Powered By Visual Paradigm Community Edition

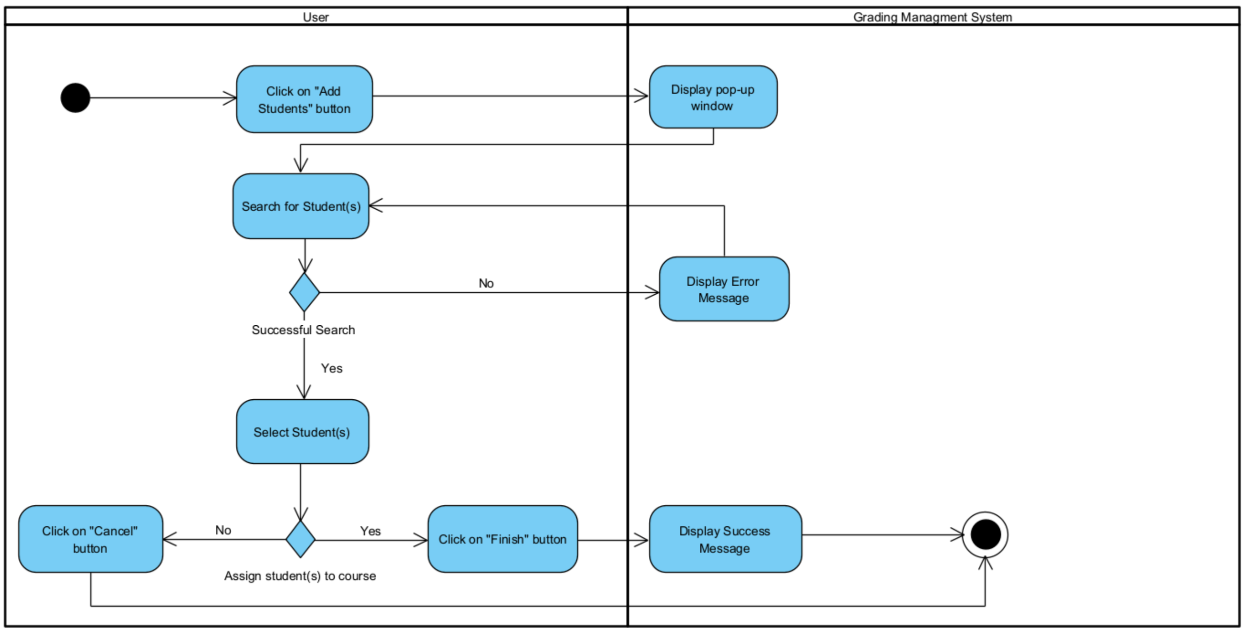
View List of Course Grades

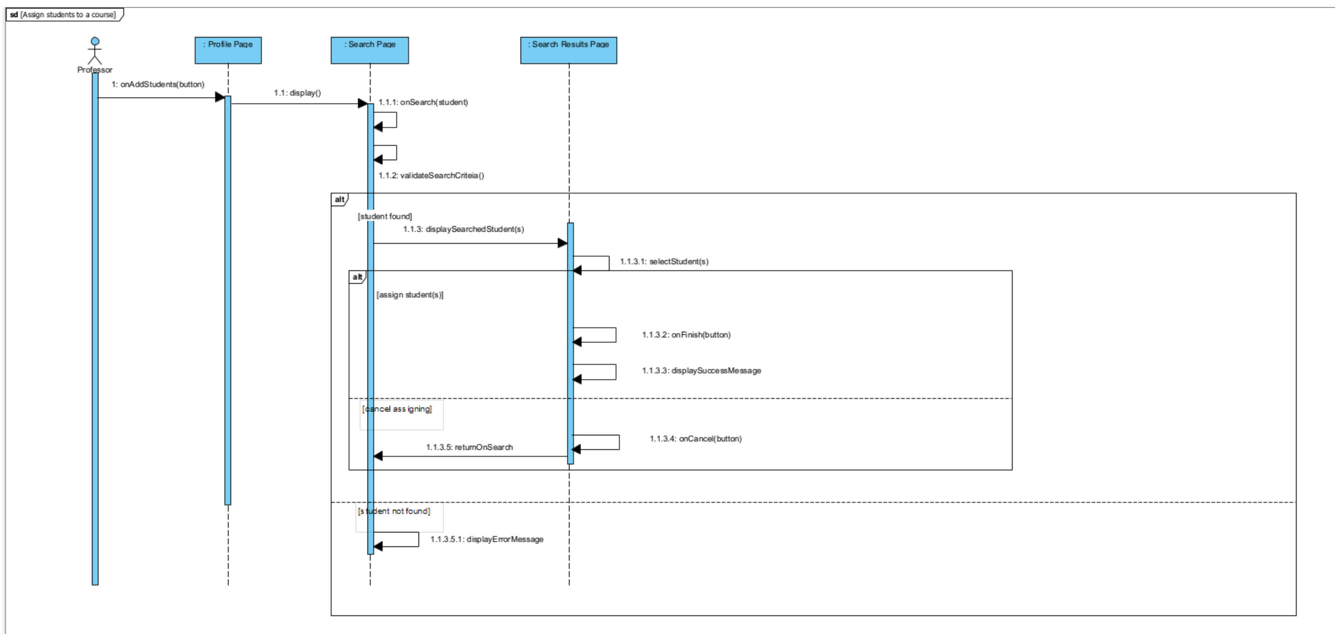




Student
Login
Autentication
System
Courses
Single Course
1: Enter login info
alt
[alt]
1.1: Send login details
1.1.1: Verify login details
1.1.2: Display home screen
1.1.2.1: Display All Courses
1.1.3: Wrong informaiton
1.1.2.1.1: Click single course
1.1.2.1.1.1: get Course Grades
1.1.2.1.1.2: list of all course grades
Powered By: Visual Paradigm Community Edition

Assign student to course




sd [Assign students to a course]
Professor
: Profile Page
: Search Page
: Search Results Page
1: onAddStudents(button)
1.1: display()
1.1.1: onSearch(student)
1.1.2: validateSearchCriteia()
alt
[student found]
1.1.3: displaySearchedStudent(s)
1.1.3.1: selectStudent(s)
alt
[assign student(s)]
1.1.3.2: onFinish(button)
1.1.3.3: displaySuccessMessage
[cancel ass igning]
1.1.3.4: onCancel(button)
1.1.3.5: returnOnSearch
[student not found]
1.1.3.5.1: displayErrorMessage

2. Project Structure

2.1. Technologies

2.1.2. Backend: PHP with Flight

For our backend, we chose PHP combined with the Flight micro-framework. PHP handled all the server-side operations, like processing data and managing how the application works behind the scenes. Flight made it easier to organize our code and handle different requests from users.

Our backend and MySQL database worked closely together. MySQL is a database management system that stores and organizes our data securely. It handles everything from saving user information to running complex searches and managing transactions. This setup ensured our application could handle a large amount of data efficiently and reliably.

2.1.3. Frontend: React.js with Vite

On the frontend, we used React.js, a powerful JavaScript library for building user interfaces. It allowed us to create interactive and dynamic components that users interact with, like buttons, forms, and lists. Vite, our frontend tool, helped us develop faster by quickly updating changes we made to our code.

2.1.4. Communication and Integration

We integrated the backend and frontend to ensure seamless interaction within our application. Here’s how it works: when a user engages with our application—whether by submitting a form or loading a page—the frontend initiates requests to the backend. These requests contain specific instructions or data needs.

On the backend, powered by PHP, these requests are received and processed. PHP handles tasks such as retrieving or updating data in our MySQL database, executing business logic, and preparing responses.

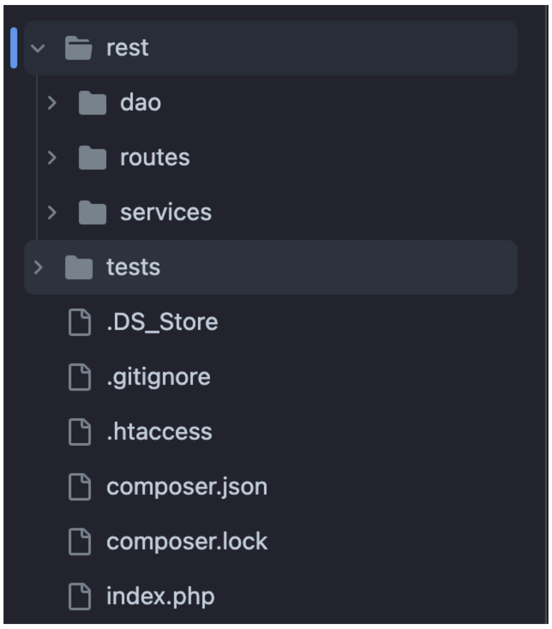
Once the backend processes the request and gathers the required data, it sends back a response. This response typically contains the requested data formatted in a way that the frontend understands.

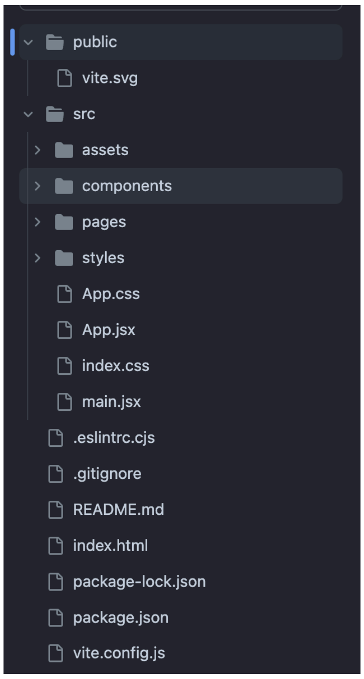
React.js, on the frontend, plays a crucial role in this process. It listens for responses from the backend and dynamically updates the user interface based on the received data. This interaction ensures that users experience a fluid and responsive interface, where changes happen quickly and seamlessly without requiring full page reloads.

Overall, our technology stack—PHP with Flight for the backend, React.js with Vite for the frontend, and MySQL for the database—worked together seamlessly. This combination allowed us to build a reliable, fast, and user-friendly web application that meets the needs of our users effectively.

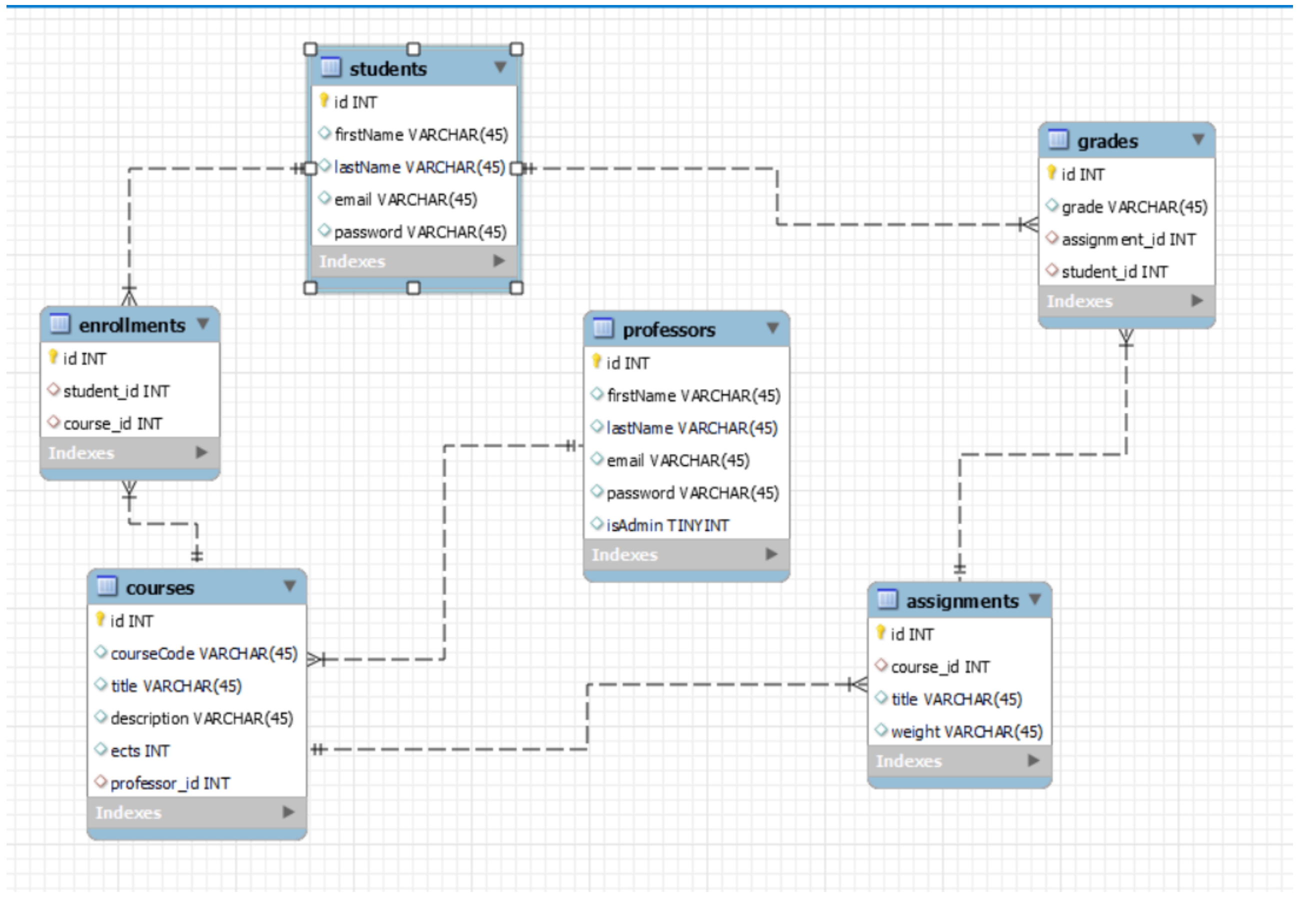
2.2. Architectural Pattern

The architectural pattern followed in this project is the Layered Architecture Pattern. A clear division of responsibilities is encouraged when software is designed using a layered architectural pattern, which improves maintainability and scalability. Because every layer has a different job to do, debugging is made easier and independent scalability is made possible. The system can develop and adjust to new requirements with the least amount of disturbance because of this modular design, which also makes components more flexible, reusable, and testable.


```
rest
  dao
  routes
  services
tests
.DS_Store
.gitignore
.htaccess
composer.json
composer.lock
index.php
```


```
public
  vite.svg
src
  assets
  components
  pages
  styles
  App.css
  App.jsx
  index.css
  main.jsx
.eslintrc.cjs
.gitignore
README.md
index.html
package-lock.json
package.json
vite.config.js
```

2.3. Database Entities



- **students** → Stores information about students, including their identification and contact details. Used to identify and track students within the system.
- **professors** → Contains data about professors, such as their names and department affiliations.Manages professor information and links them to courses.
- **enrollments** → Records student enrollments in various courses. Tracks which students are enrolled in which courses and when.
- **grades** → Stores students' grades for different courses and assignments.Used to record and analyze students' academic performance.
- **courses** → Contains information about courses offered, including course names and descriptions.Manages course details and links them to professors and students.
- **assignments** → Records details of assignments given in courses, including due dates and descriptions. Tracks assignments and links them to courses and grades.

2.4. Design Patterns

In our Grading Management System, we utilized several design patterns to ensure code maintainability, efficiency, and scalability. Below is a list of the design patterns we employed, along with where they can be found in the source code and a brief explanation of their use and benefits.

1. Singleton Pattern

File Location:

- grading-management-system/rest/dao/BaseDao

Since we needed a single point of access to the database, we chose the Singleton pattern for our BaseDao class. The Singleton pattern ensures that only one instance of the BaseDao class is created and used throughout the application, providing a consistent way to access database connections and preventing resource conflicts.

Using the Singleton pattern helped us manage database connections efficiently, avoiding multiple open connections and ensuring that all database operations are coordinated through a single instance, which simplified debugging and maintenance.

2. Concrete Class for BaseDao

File Location:

- grading-management-system/rest/dao/StudentDao
- grading-management-system/rest/dao/ProfessorsDao
- grading-management-system/rest/dao/CoursesDao
- grading-management-system/rest/dao/Assignments Dao
- grading-management-system/rest/dao/EnrollmentsDao
- grading-management-system/rest/dao/GradesDao

For our data access layer, we created concrete classes extending the BaseDao class for specific entities such as **StudentDao**, **ProfessorsDao**, and **CoursesDao**. Each concrete class inherits common database operations from BaseDao and implements entity-specific methods. This structure allows for code reuse and consistency across different DAO classes.

By using concrete classes, we avoided redundant code and provided a clear, modular structure for handling data operations. This approach made it easier to extend functionality and maintain the codebase, ensuring that changes in database handling are localized and do not affect other parts of the application.

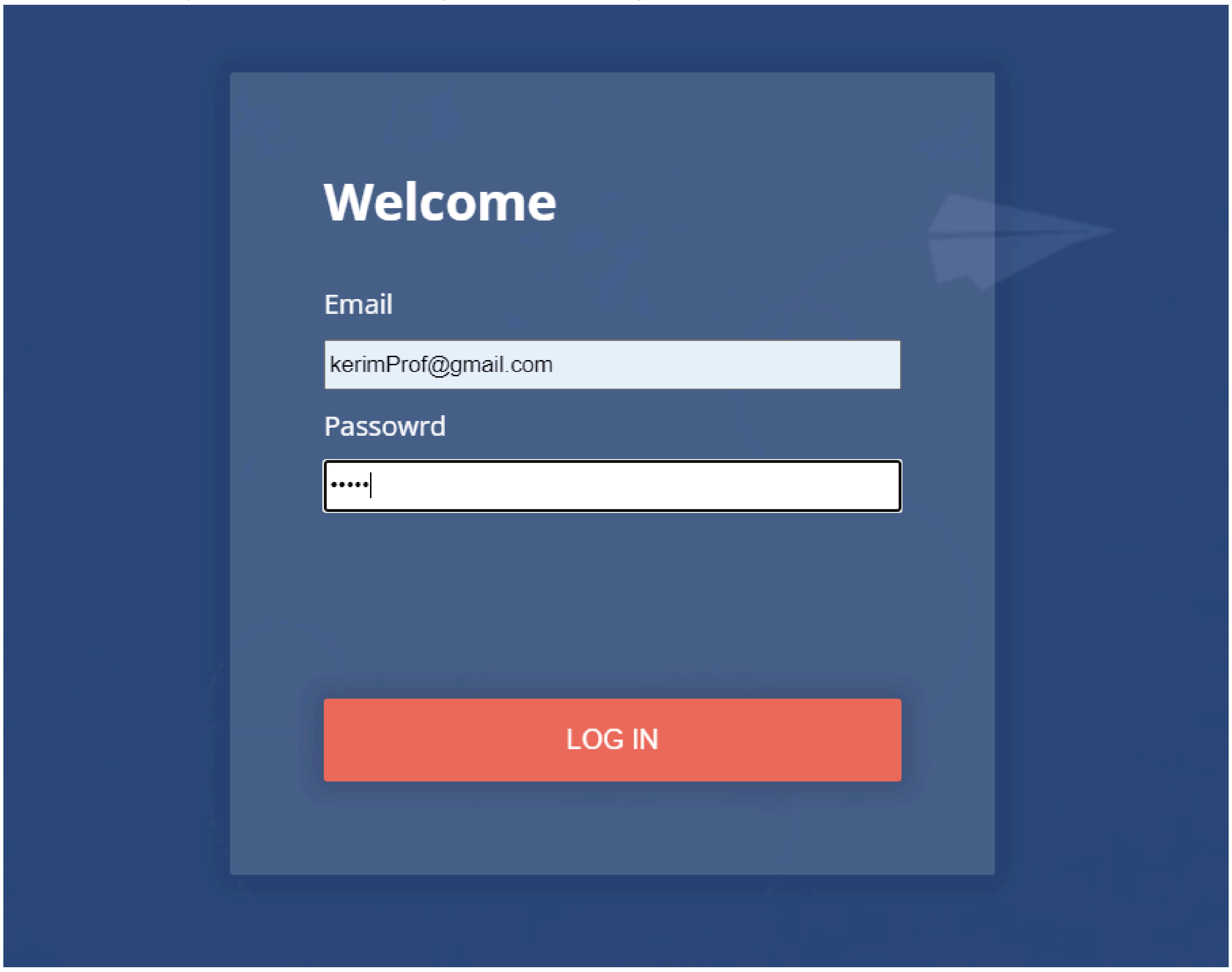
Singleton is crucial for managing global resources like database connections or configuration settings, while Concrete classes handle specific operations and data management tasks related to grading, ensuring flexibility and maintainability in the system architecture.

2.5. Project Functionalities and Screenshots

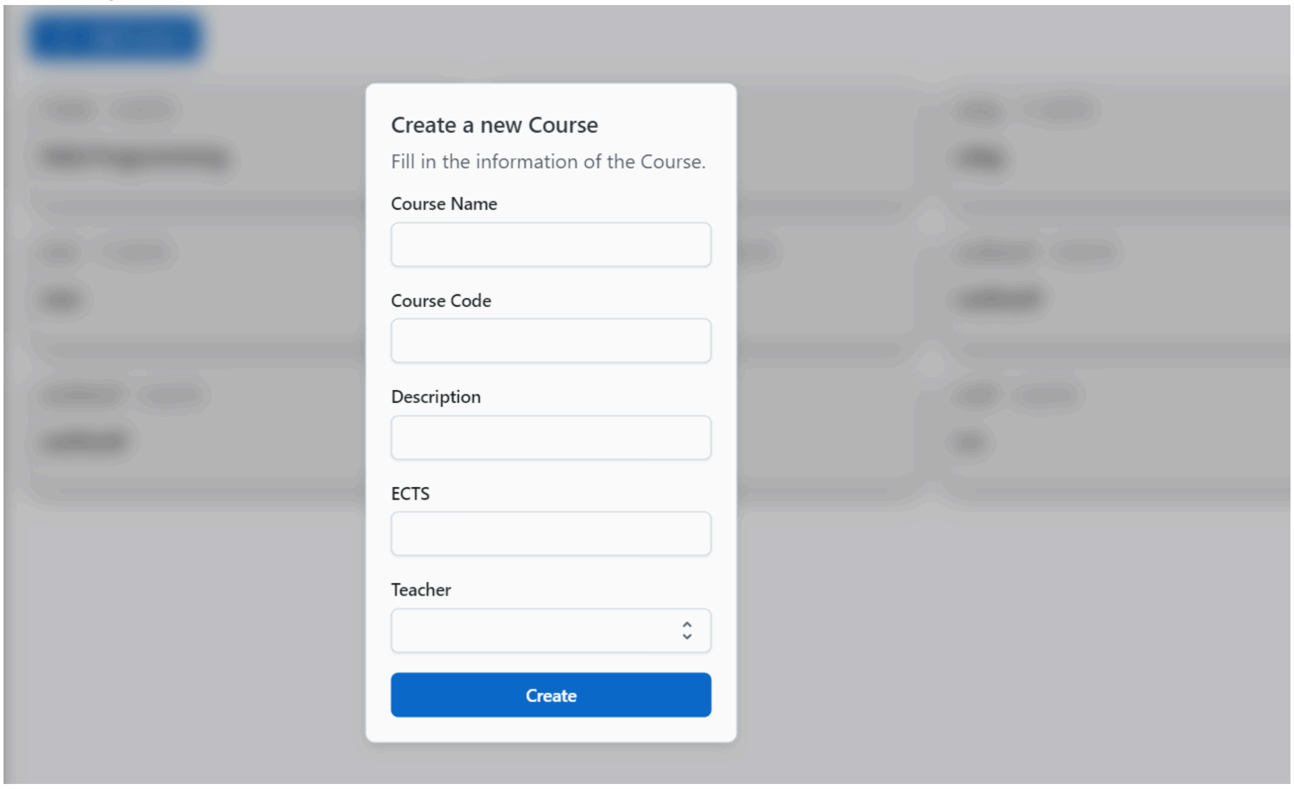
Here are some of the main features of our Grading Management System**:**

1. Login
2. Logout
3. Adding Courses
4. Updating Courses
5. Deleting Courses
6. Adding Students to Course
7. Removing Student from Course
8. Adding an Assignment to Course
9. Adding Students to Assignments
10. Grading Student Assignments
11. View My Profile
12. Admin and Professor are able to update their information and password
13. Students are only able to update their password

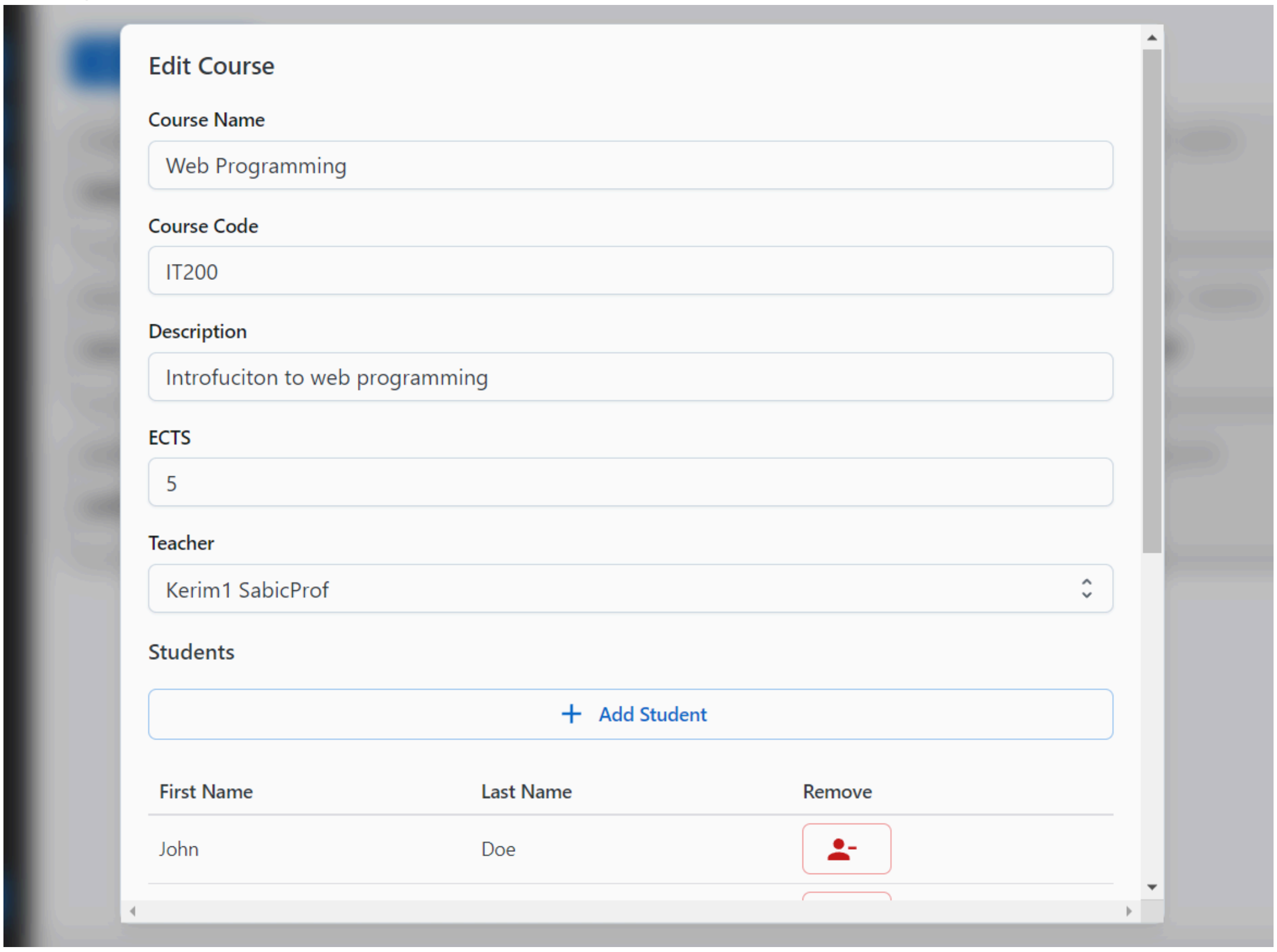
14. View System Information (About Section)



Creating a new course:



Editing a course:



Edit Course

Course Name

Web Programming

Course Code

IT200

Description

Introfuciton to web programming

ECTS

5

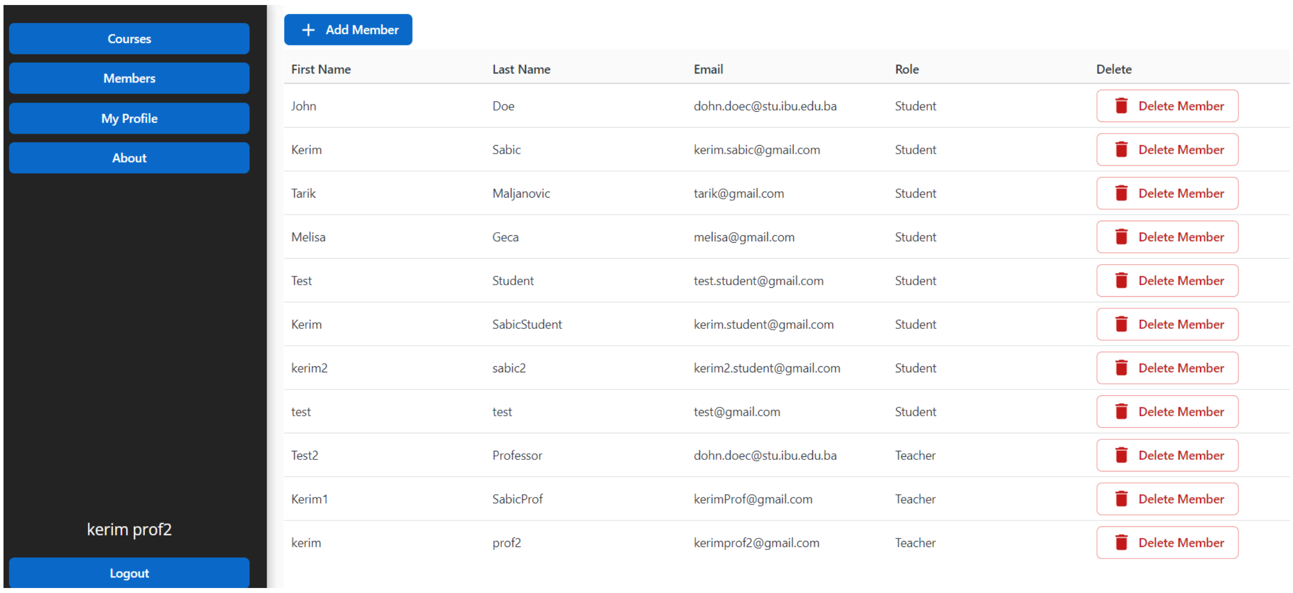
Teacher

Kerim1 SabicProf

Students

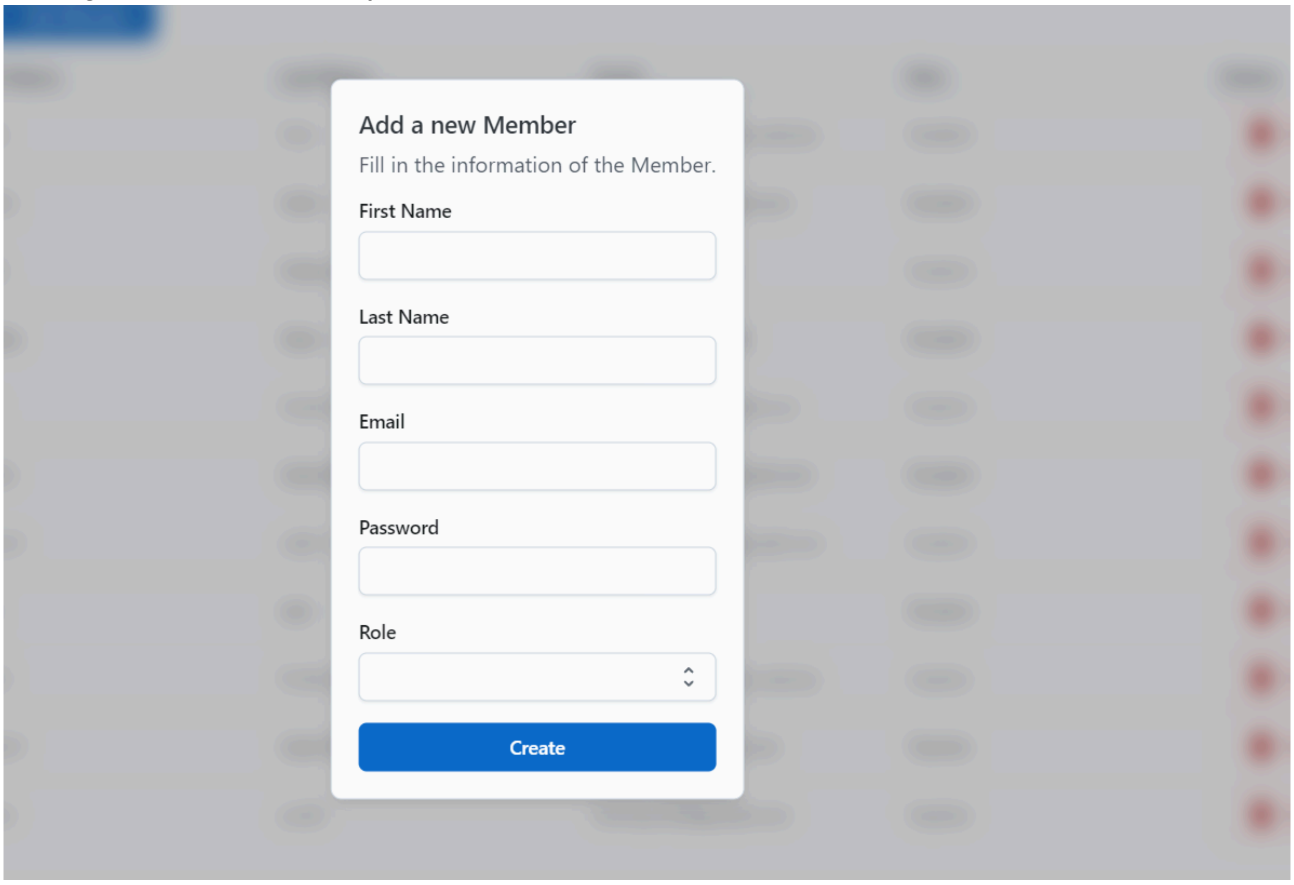
First Name	Last Name	Remove
John	Doe	

View members:



First Name	Last Name	Email	Role	Delete
John	Doe	dohn.doec@stu.ibu.edu.ba	Student	Delete Member
Kerim	Sabic	kerim.sabic@gmail.com	Student	Delete Member
Tarik	Maljanovic	tarik@gmail.com	Student	Delete Member
Melisa	Geca	melisa@gmail.com	Student	Delete Member
Test	Student	test.student@gmail.com	Student	Delete Member
Kerim	SabicStudent	kerim.student@gmail.com	Student	Delete Member
kerim2	sabic2	kerim2.student@gmail.com	Student	Delete Member
test	test	test@gmail.com	Student	Delete Member
Test2	Professor	dohn.doec@stu.ibu.edu.ba	Teacher	Delete Member
Kerim1	SabicProf	kerimProf@gmail.com	Teacher	Delete Member
kerim	prof2	kerimprof2@gmail.com	Teacher	Delete Member

Adding a member to the system:



Add a new Member

Fill in the information of the Member.

First Name

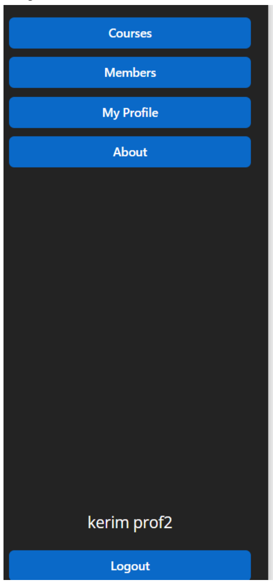
Last Name

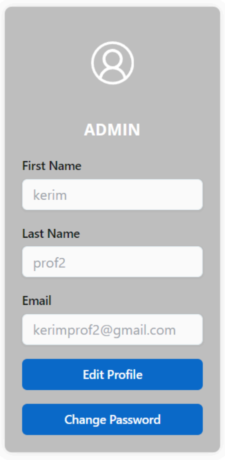
Email

Password

Role

My Profile for Professors and Administrators:





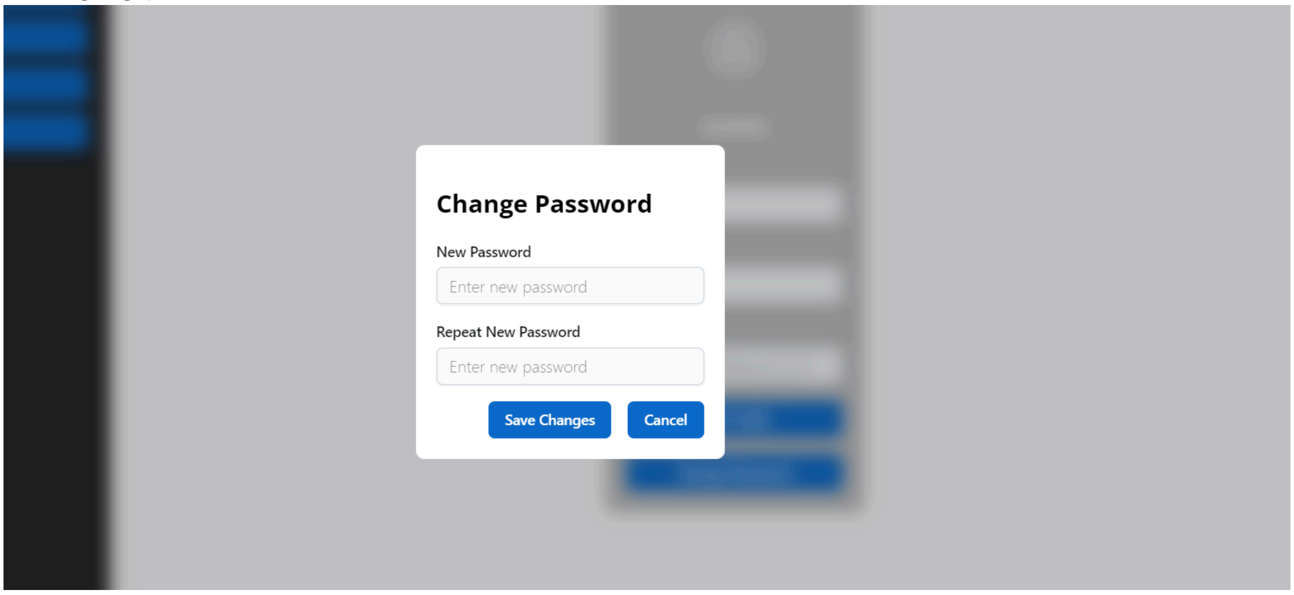
ADMIN

First Name: kerim

Last Name: prof2

Email: kerimprof2@gmail.com

Changing password:



Change Password

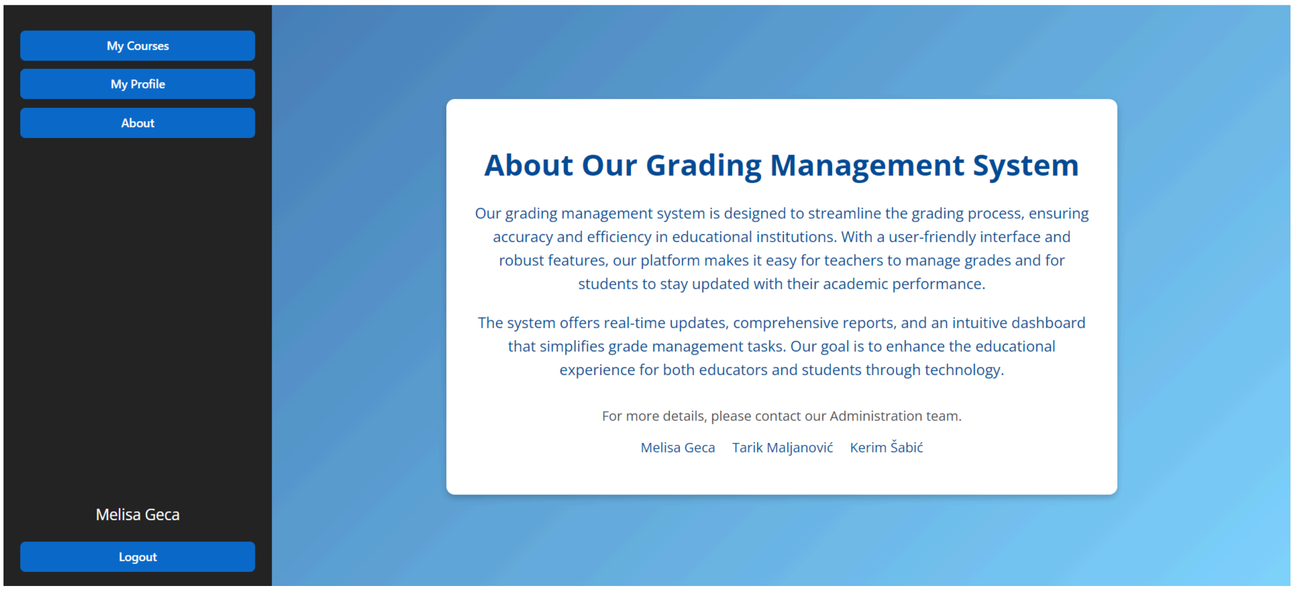
New Password

Enter new password

Repeat New Password

Enter new password

View System Information:



About Our Grading Management System

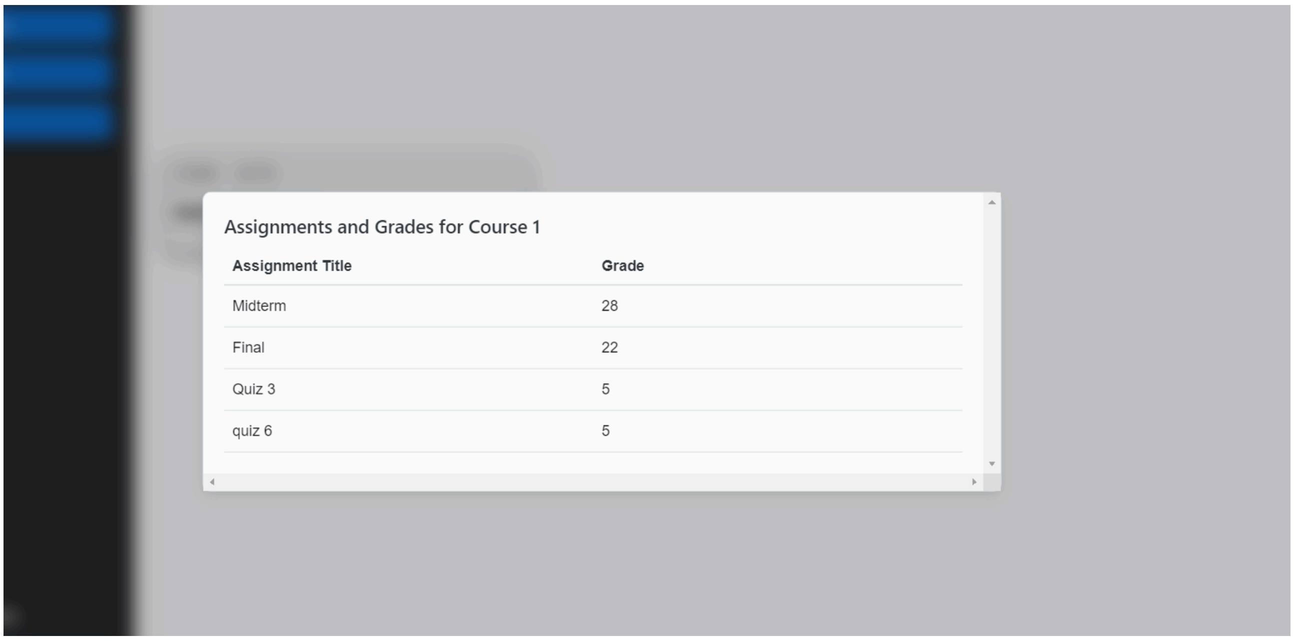
Our grading management system is designed to streamline the grading process, ensuring accuracy and efficiency in educational institutions. With a user-friendly interface and robust features, our platform makes it easy for teachers to manage grades and for students to stay updated with their academic performance.

The system offers real-time updates, comprehensive reports, and an intuitive dashboard that simplifies grade management tasks. Our goal is to enhance the educational experience for both educators and students through technology.

For more details, please contact our Administration team.

Melisa Geca Tarik Maljanović Kerim Šabić

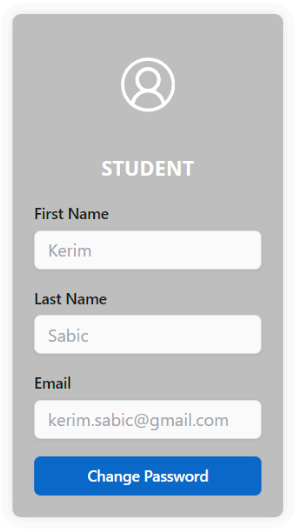
Student Views Course Grades:



Assignments and Grades for Course 1

Assignment Title	Grade
Midterm	28
Final	22
Quiz 3	5
quiz 6	5

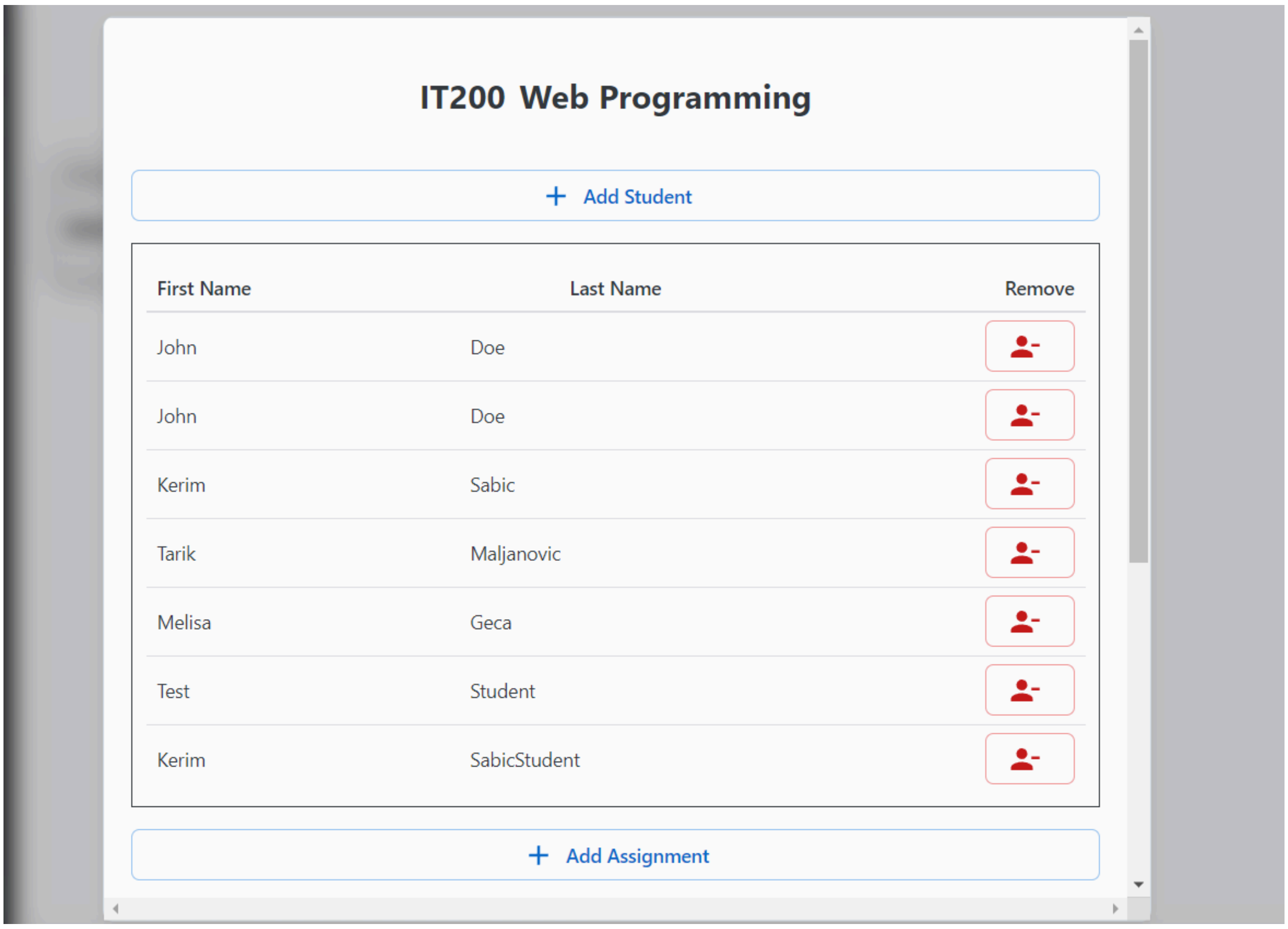
Student My Profile (can only modify password):



STUDENT

First Name	Kerim
Last Name	Sabic
Email	kerim.sabic@gmail.com

Professors view of the course:

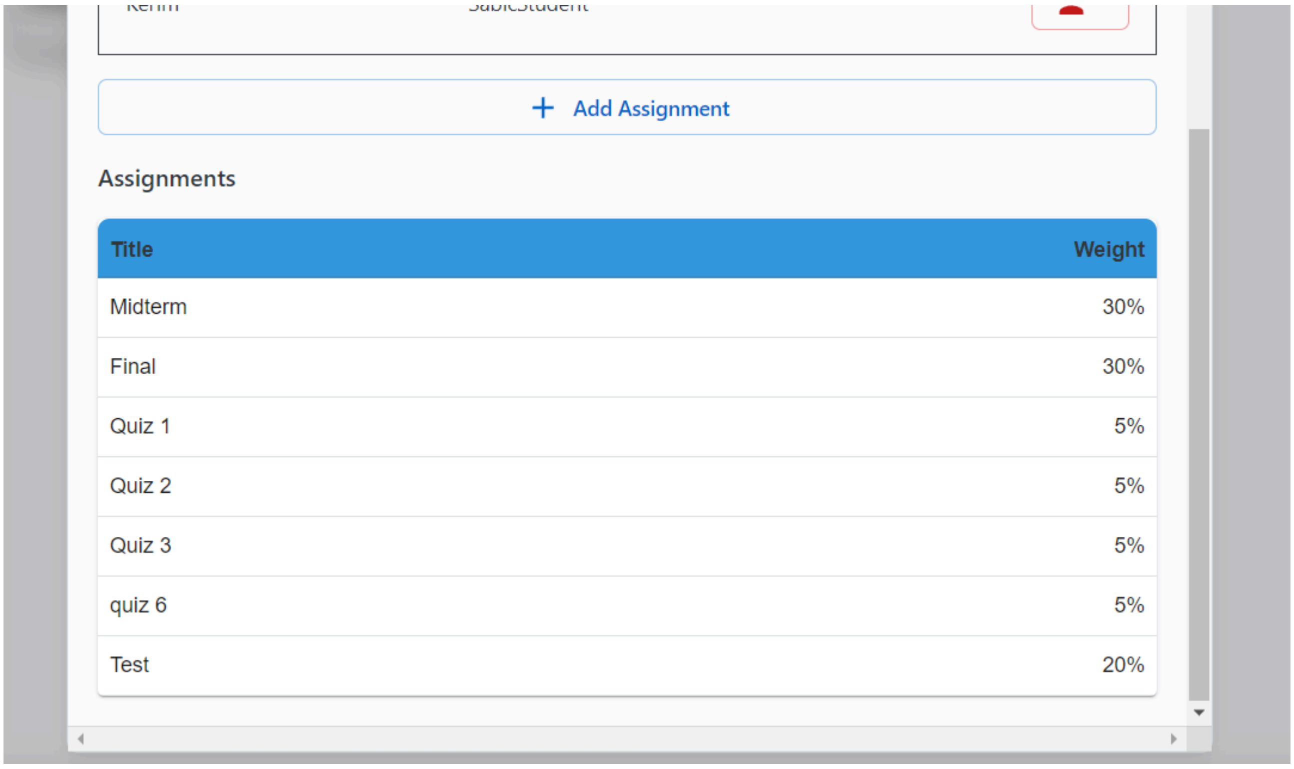


IT200 Web Programming

+ Add Student

First Name	Last Name	Remove
John	Doe	
John	Doe	
Kerim	Sabic	
Tarik	Maljanovic	
Melisa	Geca	
Test	Student	
Kerim	SabicStudent	

+ Add Assignment

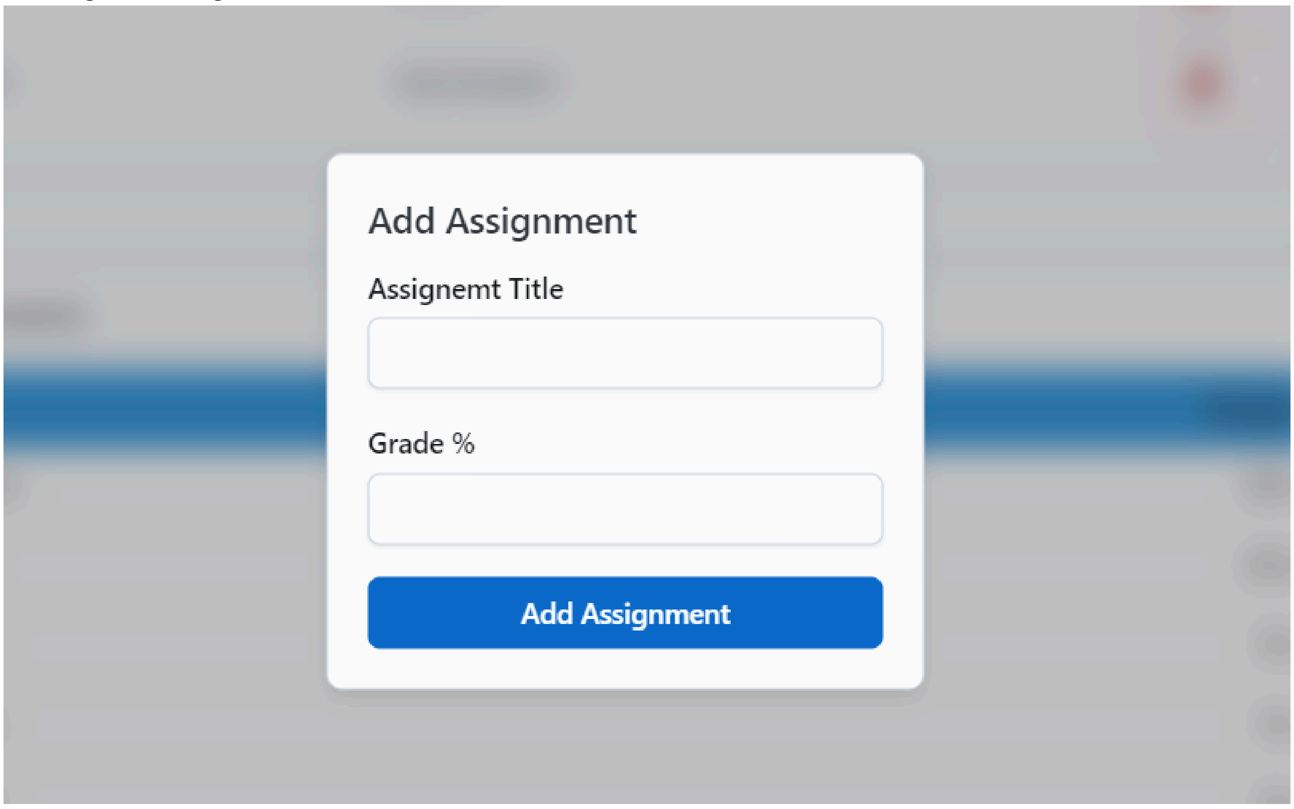


+ Add Assignment

Assignments

Title	Weight
Midterm	30%
Final	30%
Quiz 1	5%
Quiz 2	5%
Quiz 3	5%
quiz 6	5%
Test	20%

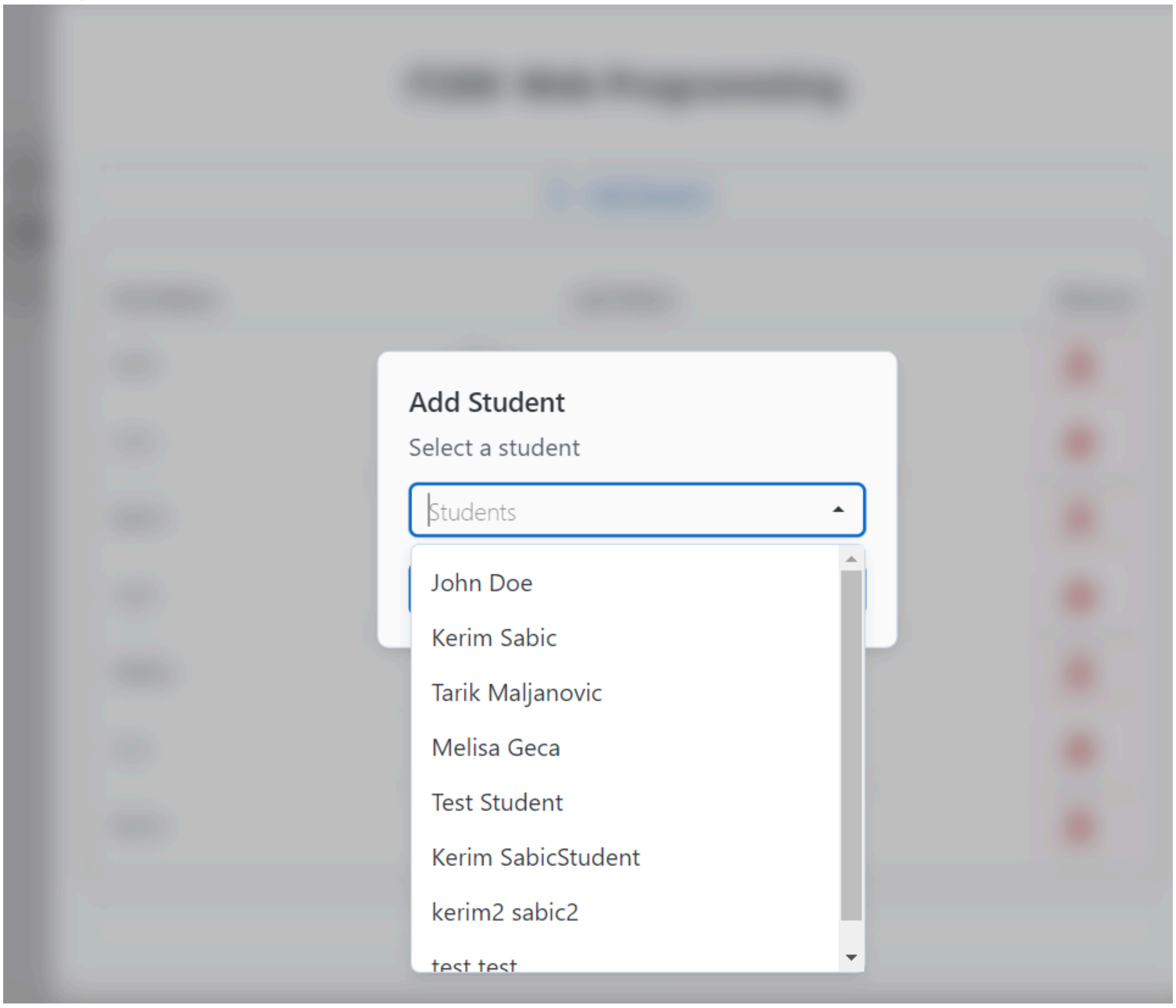
Adding an assignment to the course:


Add Assignment

Assignemt Title

Grade %

Adding Student to the course:

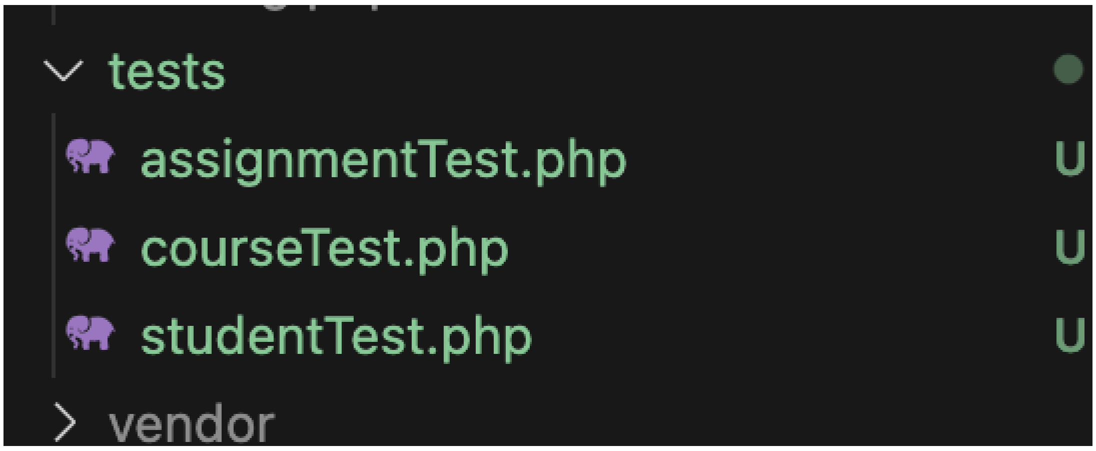


2.6. Tests

We have conducted unit tests on 3 of our most critical entities (Students, Courses and Assignments). The test cases and code can be found in the backend repository in the "tests" directory.
The following functionalities were tested:

1. Retrieving grades of a course
2. Retrieving grades of a specific student for a specific course
3. Retrieving all the courses of a professor
4. Retrieving the students of a course
5. Retrieving the assignments of a course
6. Adding a new student
7. Retrieving the courses of a student
8. Deleting a student

All tests have passed with status 200 “OK”.


```
⌄ tests                        ●
    assignmentTest.php         U
    courseTest.php             U
    studentTest.php            U
> vendor
```

3. Conclusion

API URL: https://king-prawn-app-66oof.ondigitalocean.app/
Website URL: https://sea-lion-app-txixi.ondigitalocean.app/

Our website is currently deployed but the deployment is turned off to avoid high costs. Please notify us when you are ready to review the project, and we will activate the deployment so that the website can be accessed over the internet.

Overall, we are pleased with the final version of our project, which aimed to enhance the school experience for both students and professors. Developing a system that meets the needs of these diverse user groups posed significant challenges in design and development.

Coordinating the functionalities required by students, professors, and administrators while maintaining a user-friendly experience was particularly challenging. However, we successfully navigated these complexities to create a system that supports a better school experience for all parties involved.

Moving forward, potentially we could aim to further improve user experience and enhance the scalability of the application to accommodate growth and potential feature integrations. These efforts will ensure our system continues to meet the evolving needs of educational institutions effectively.

Backend repository: additional routes, tests + deployment:
https://github.com/tarikmaljanovic/grading-management-system

Frontend repository:
https://github.com/tarikmaljanovic/grading-management-system-frontend/tree/faza3

Release(s):
https://github.com/tarikmaljanovic/grading-management-system/releases/tag/r2
https://github.com/tarikmaljanovic/grading-management-system-frontend/releases/tag/r2