IUPUI

INFO-C451

PROJECT: Student Management System(Pythagor)

Fabrice Ngahadjo Tchouamo

I. General Introduction

Sufficient and varied levels of assistance must accompany cooperative information systems. Multi-agent systems make it possible to coordinate the behavior of agents interacting and communicating in society to perform tasks or solve problems. Therefore, it seems well suited for designing a mediation intended for the users of these complex systems – each agent representing a level of assistance. To illustrate our approach, we will present an assistance system integrated into the learning environment Pythagor (Student management system). With our App, we can learn how student success and lifecycle management technologies help improve student outcomes. Higher education institutions are increasingly applying predictive analytics to student success projects. Several technologies were necessary to realize our project; we will cite the HTML language for completing static pages, a conceptual study carried out by the UML modeling language, the PHP language for the dynamic part, and MySQL for the elaboration of database query requests.

II. customer problem statements and system requirements

No.	**Priority Weight**	**Description**
REQ-1 (show the unique label of the requirement)	The priority weight of the requirement	Describe the requirement briefly

No.	Priority Weight	Description
REQ-1	High	Only the administrator could create the class
REQ-2	High	Only the Administrator could create professor and student account
REQ-3	High	Only The professor could Add and change the grade for his class

REQ-4	High	The student should have one or more classes
REQ-5	Medium	The Student could add or delete the class
REQ-6	High	Only the Professor could add/Delete assignment
REQ-7	Medium	The system should lock student access to the assignment after the due date.

A. Functionality

Pythagor is a simple Web App with three principal accounts. The admin account can manage, change, and delete an account; The professor account for managing classes and students, and the student account can take the assignment and manage the course. We Have some functionality like Search.

B. Usability

This Application could help trace the student lifecycle and professor work. It could help the professor get the eagle eyes on student works.

C. Reliability

Pythagor Have a solid and secure Maria DB database http://173.255.197.34/phpmyadmin/

Sitting in www.cloud.linode.com. This database is replicated in the western region, meaning we will have a backup recovery in a disaster case. Pythagor Web App

http://173.255.197.34/index.php is stored on a secure subnet behind the firewall and app load balancer, which means all the security setup is in place. The repository for the Version control system is on GitHub https://github.com/ngahadjo/pythagor; the pipeline is not in place yet but will be for the next deployment.

D. Performance

The Pythagor app could help the user manage and search want they want in the database

E. Supportability

The app is in PHP and can run with chrome, Mozilla, or browse. It is light and could be accessible on the phone too. The DNS is not ready and in place yet, but the Server could support up to 10000 users

III. functional requirement specification

- **Actors and Goals**

Primary actors

- Students: This actor can Register with the Faculty and take a class under his major.

- Professor: This actor could have one or more classes where he Teaches. I can have access to the student profile in his class.

Secondary actors:

- Admin: This can add, remove, or update faculty, student, and professor accounts.

- **Casual Description**

Here we use 1 or 2 to indicate the submenu or sublink because each point will be the link to access the issue.

Admin (Total: 26)

- Login/Logout: login/logout to the admin account (2)

- Check/add/search Student: Add/Remove student to the faculty (4)

- Check/search/Update Professor: Add/ Remove to the Faculty (3)

- Add/ Remove Major: Add/ Remove to the Faculty (2)

- Add/ Remove Classes: Add/ Remove Classes to the Major (2)

- Add/ Remove Homework: Add/ Remove Homework to the class (2)
- Add/ Remove Grade: Add/ Remove Grade to the Student (2)
- Add/ Update Graduation: Add/ Remove Graduation and Graduate students (4)
- Add/ Update Tutor: Add/ Remove Tutor to the class (2)
- Check/search/Update Internship: Update internship for Students (3)

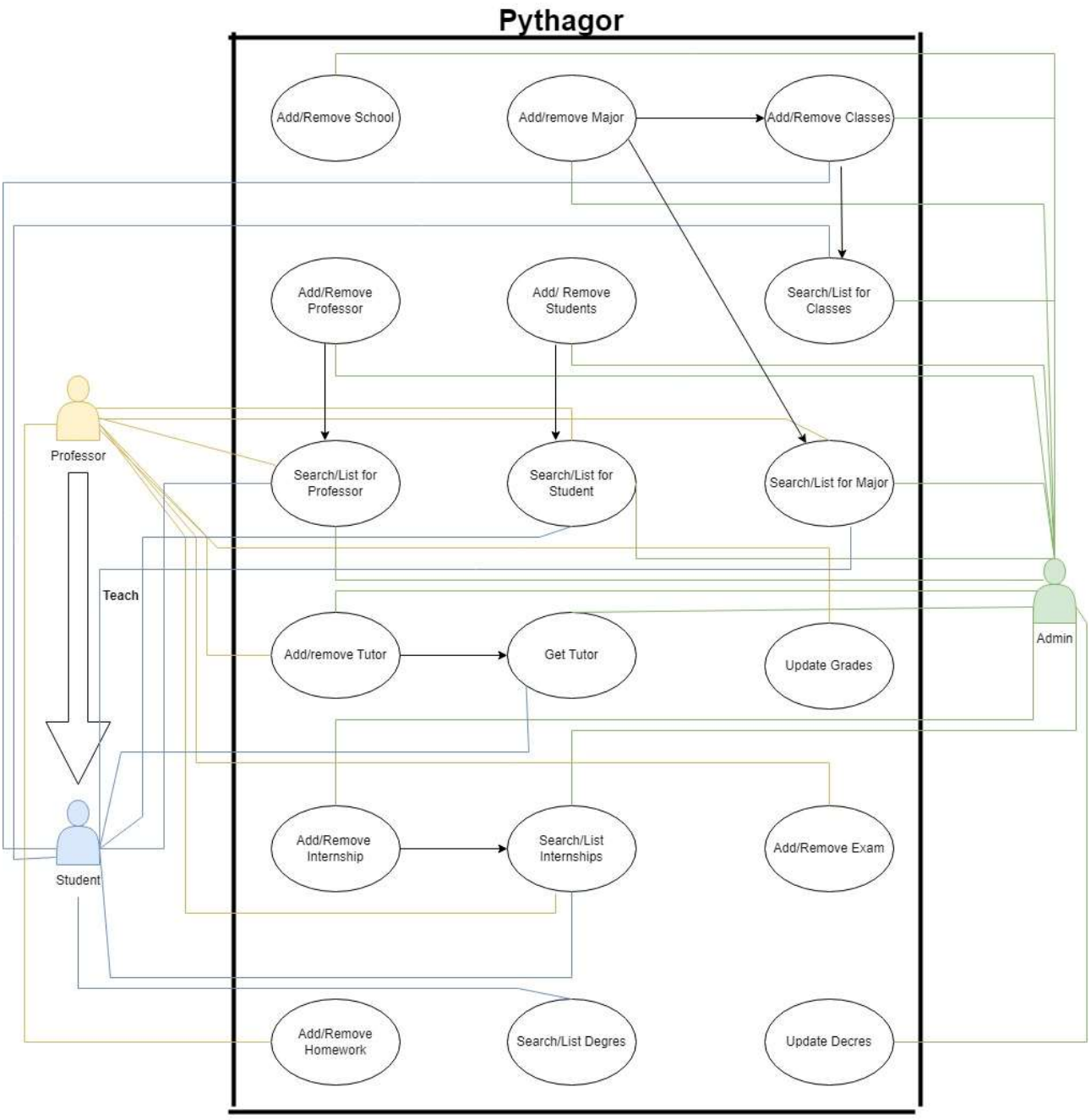
Student (total: 14)

- Login/Logout: To log in/logout to the Student account (2)
- Search/Check: Search the class's student or professor or, consul his grade (3)
- Search/check professor: Search professor by class or by name (3)
- Search: Search his classes (1)
- Search/check internships: Search available internships (2)
- Search/check classes: Search available classes (2)
- check classes: Check an accomplishment degree (1)

Professor (Total: 18)

- Login/Logout: To log in/log out to the Professor account (2)
- Update/Search/check: Update/Search/check classes, students, and grades (3)
- Search/check Professor: Search available Professor (2)
- Search/check classes: Search classes (2)
- Search/check tutor: Search tutor (2)
- Search/check internship: Search internship (2)
- Add/Remove Exam: Add/Remove Exam for class (2)
- Add/Remove Homework: Add/Remove Homework for class (2)

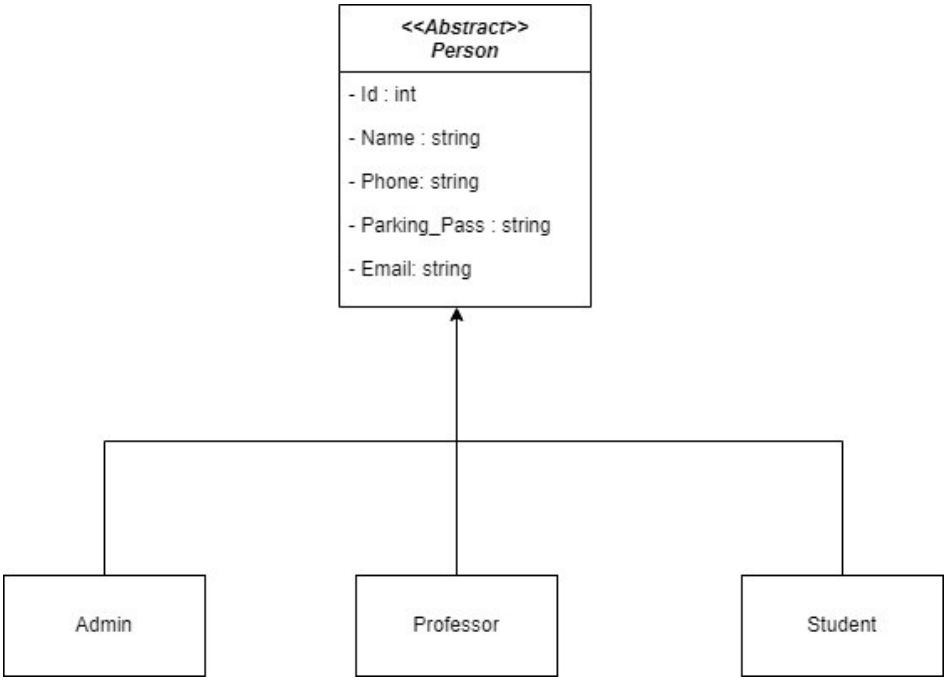
- **Use Case Diagram**


Pythagor
Add/Remove School
Add/remove Major
Add/Remove Classes
Add/Remove Professor
Add/ Remove Students
Search/List for Classes
Professor
Search/List for Professor
Search/List for Student
Search/List for Major
Teach
Admin
Add/remove Tutor
Get Tutor
Update Grades
Add/Remove Internship
Search/List Internships
Add/Remove Exam
Student
Add/Remove Homework
Search/List Degres
Update Decres

- **Class Diagram**

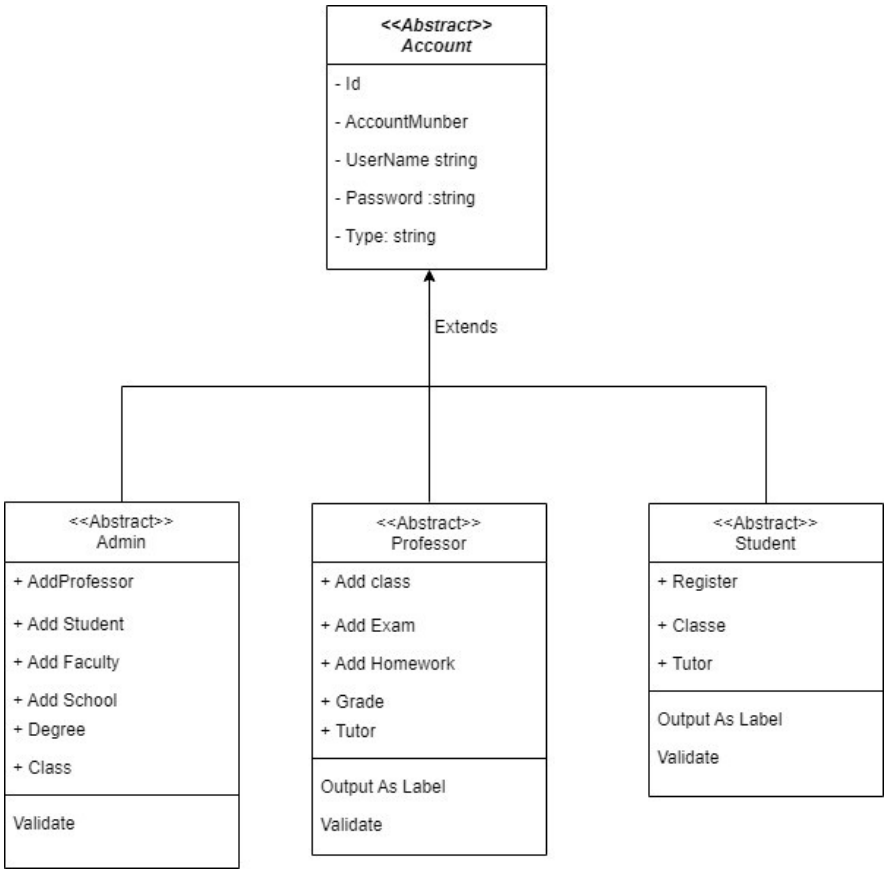
1- Person

We have 3 People who interact with our system, The Admin, The professor, and the student.

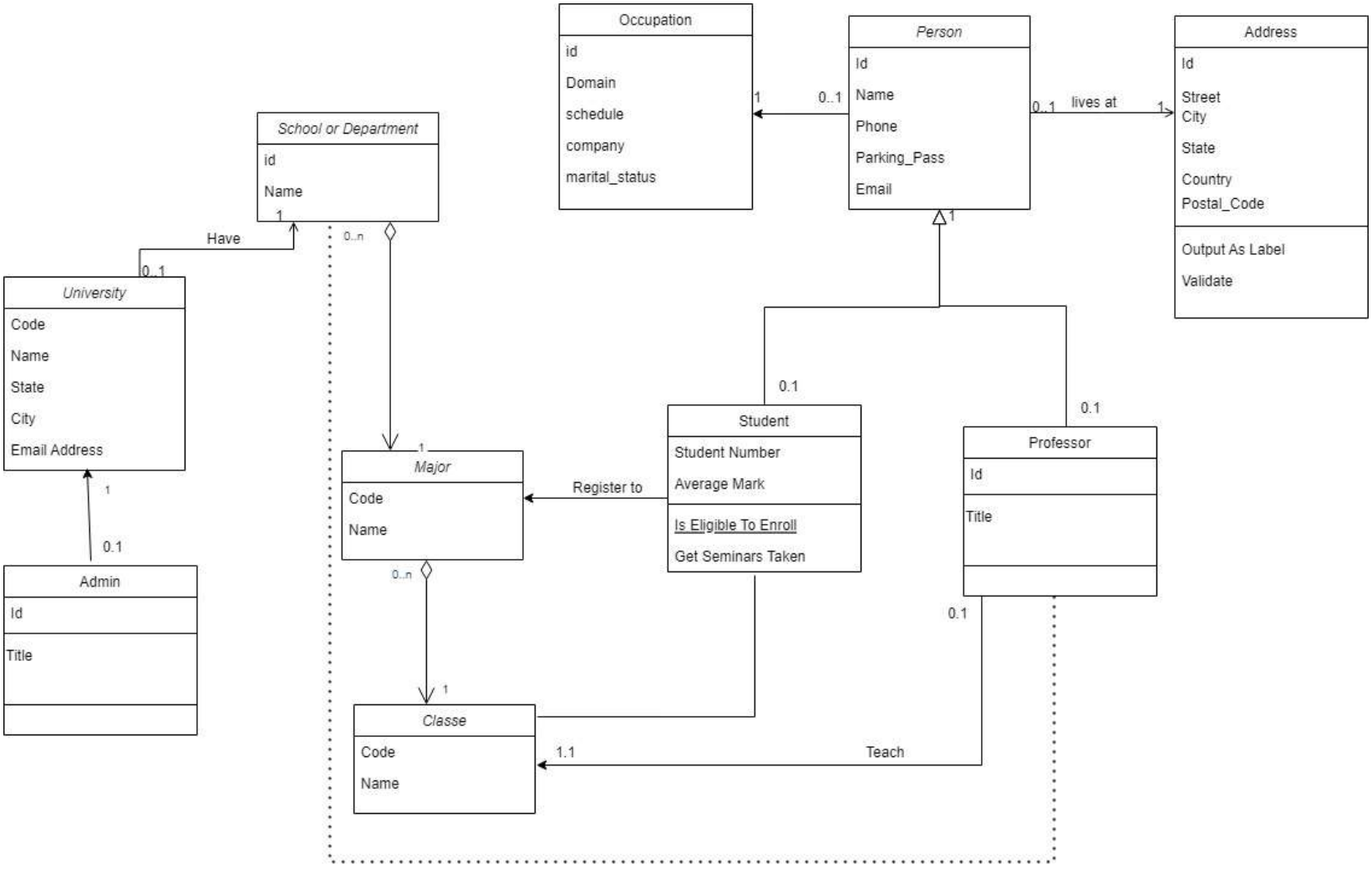


2- Account

We will have three types of accounts. However, the admin will be the one who will manage the funds.


<<Abstract>>
Account
- Id
- AccountMunber
- UserName string
- Password :string
- Type: string
Extends
<<Abstract>>
Admin
+ AddProfessor
+ Add Student
+ Add Faculty
+ Add School
+ Degree
+ Class
Validate
<<Abstract>>
Professor
+ Add class
+ Add Exam
+ Add Homework
+ Grade
+ Tutor
Output As Label
Validate
<<Abstract>>
Student
+ Register
+ Classe
+ Tutor
Output As Label
Validate

Inheritance



IV. system sequence diagram

In our project, we have defined two sequence diagrams. These diagrams represent all the actions and interferences between actors (Professor/student) and system and system.

State Actor and Object of this sequence:

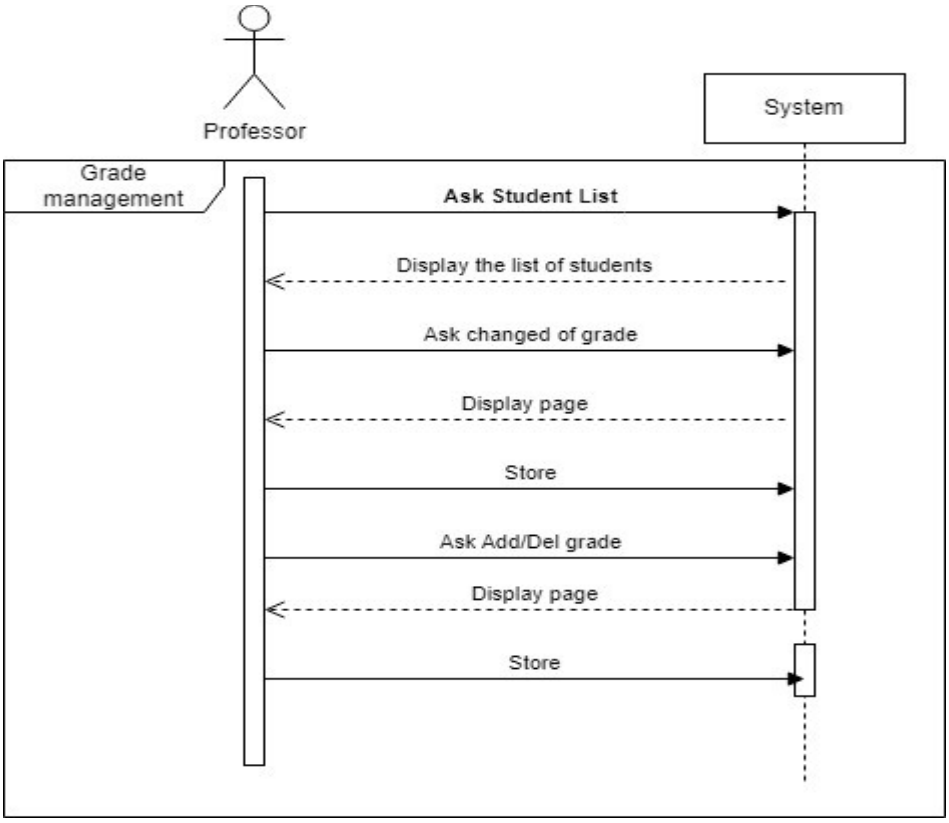
Actor: Professor

Object: Obtain the list of students absent in a session

Steps for Grade management:

The Professor can

- The Professor asks to edit grades.
- The system displays the page to access student grades.
- The Professor accesses the notes. And do the actions of changes.
- The Professor requests to store the changes.
- The Professor asks the system to add or delete notes.
- The system displays the page.
- The Professor completes the form.
- the Professor requests to store the information.
- The system is recording.



State Actor and Object of this sequence:

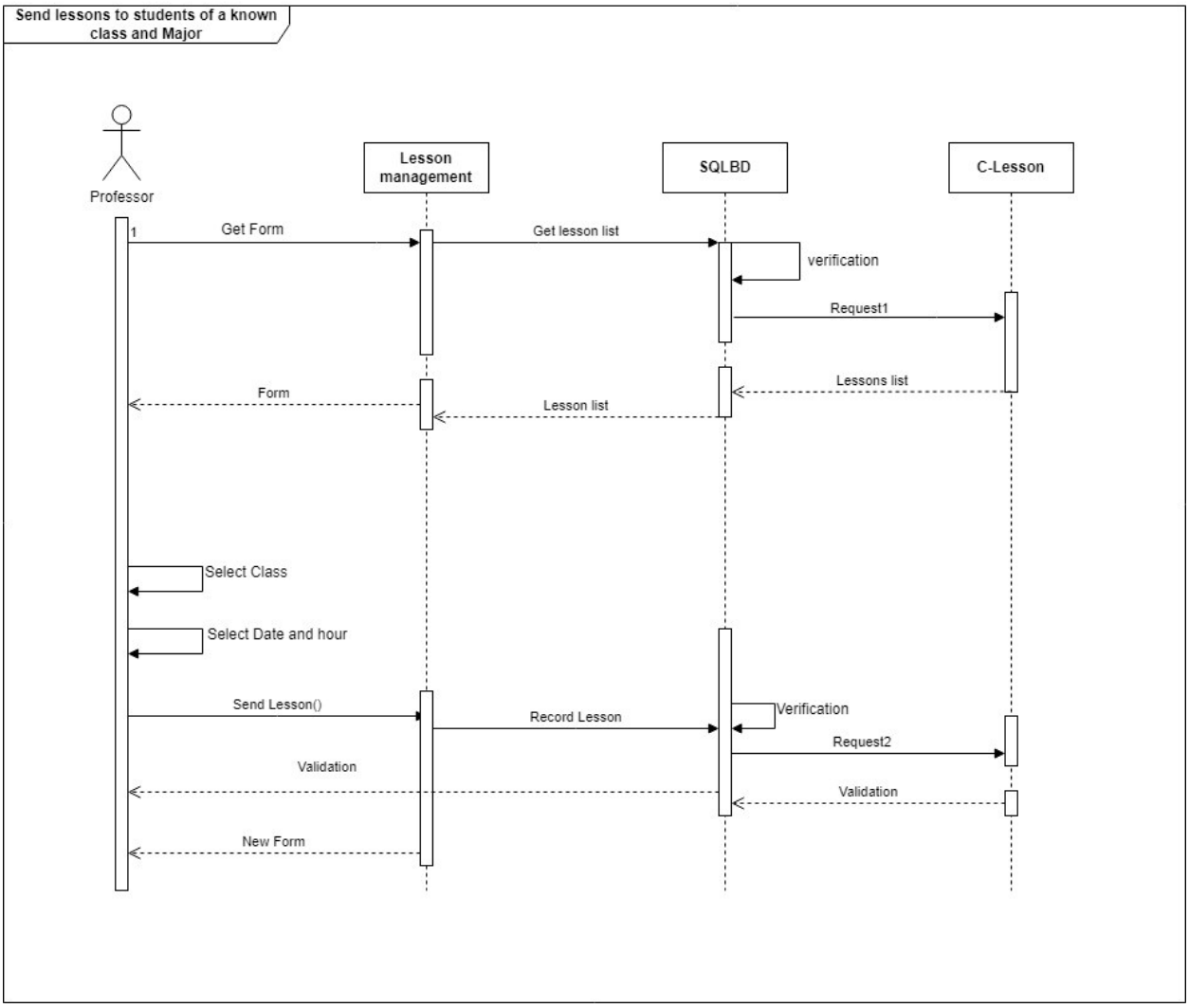
Actor: Professor

Object: Send lessons to students of a known level and group

Steps for students' Lesson interaction:

The Professor can

- The Professor asks to consult the courses and personal work sent to the students.
- The system displays the page.
- The Professor asks to add or delete lessons or assignments.
- The system displays the form.
- The Professor stores.
- The system is recording.
- The Professor asks to consult the works sent by the students. Send lessons to students of a known class and Faculty.



Absence

State Actor and Object of this sequence:

Actor: Professor

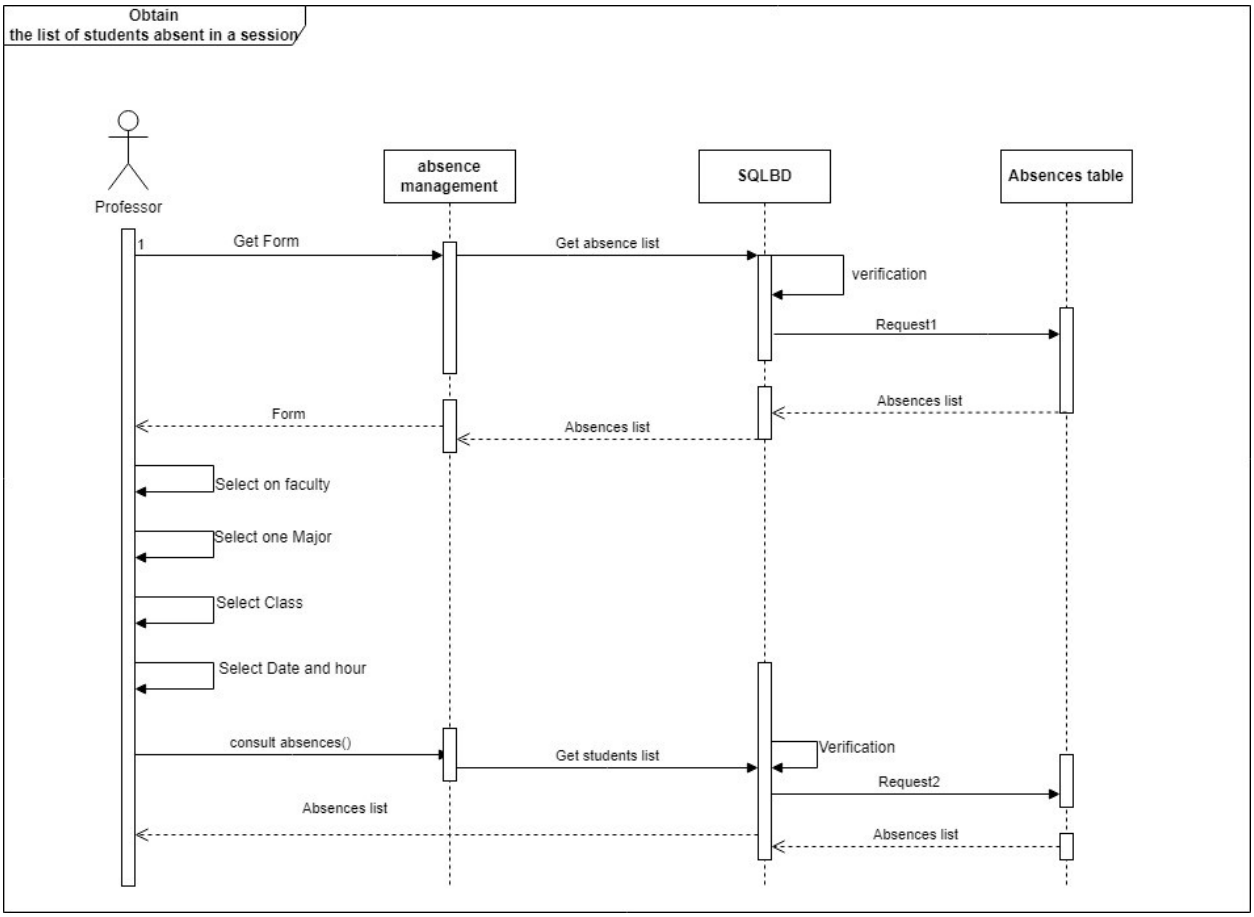
Object: Obtain the list of students absent in a session

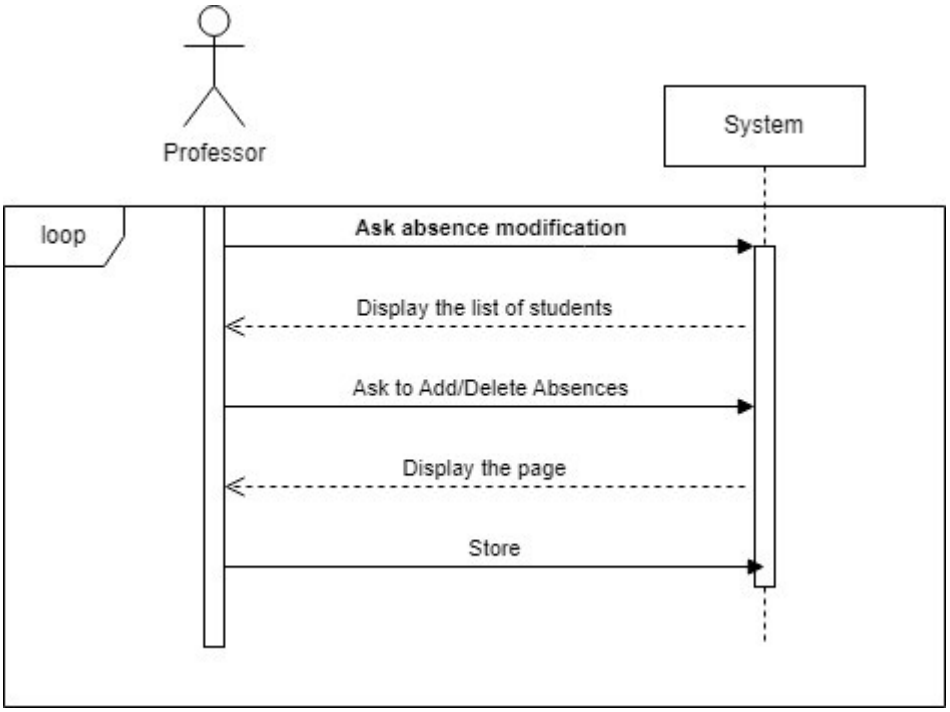
Steps for students' absent interaction:

The Professor can

- consult the absences of the students.
- modify the absences of the students.
- add the absences of the students.
- delete the absences of the students.
- validates the justifications.
- consults the excluded modules.

Based on the interactions discussed above, the sequence diagram could be:





State Actor and Object of this sequence:

Actor: student

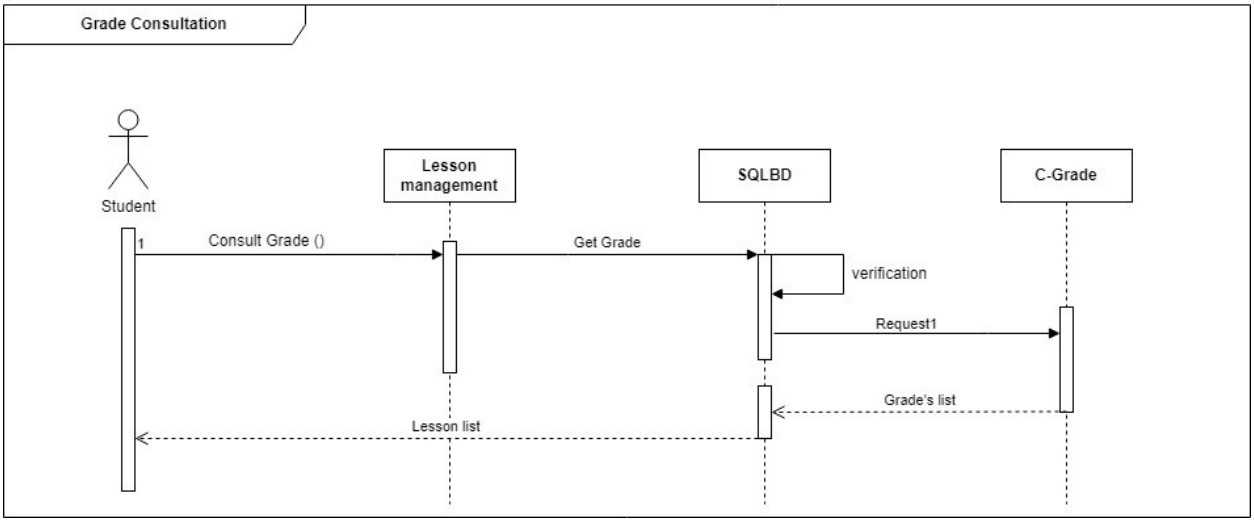
Object: Proposition assignment interaction diagram

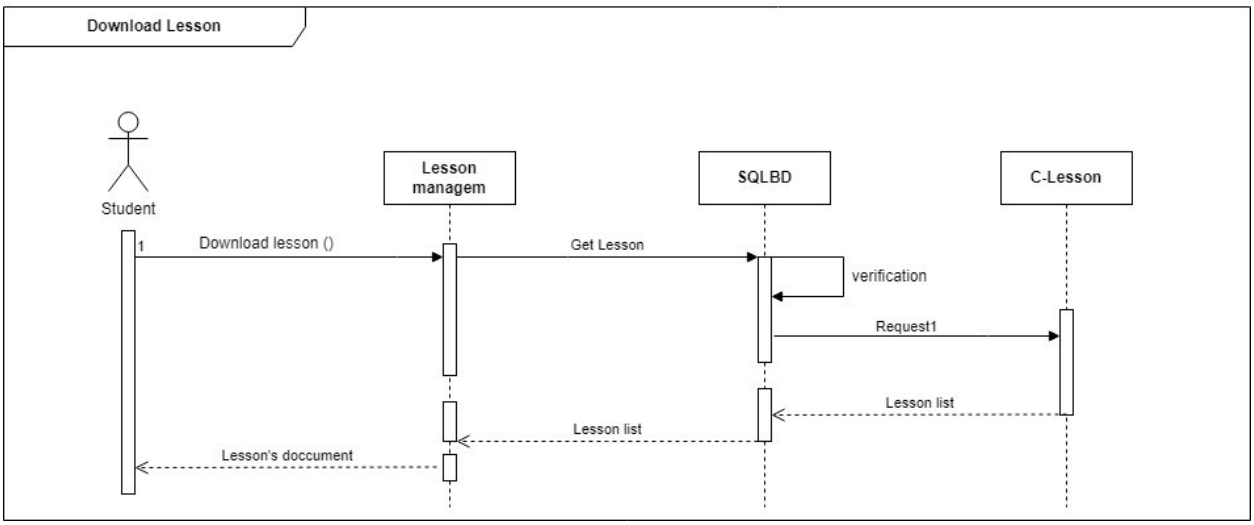
Steps for Grade consultation:

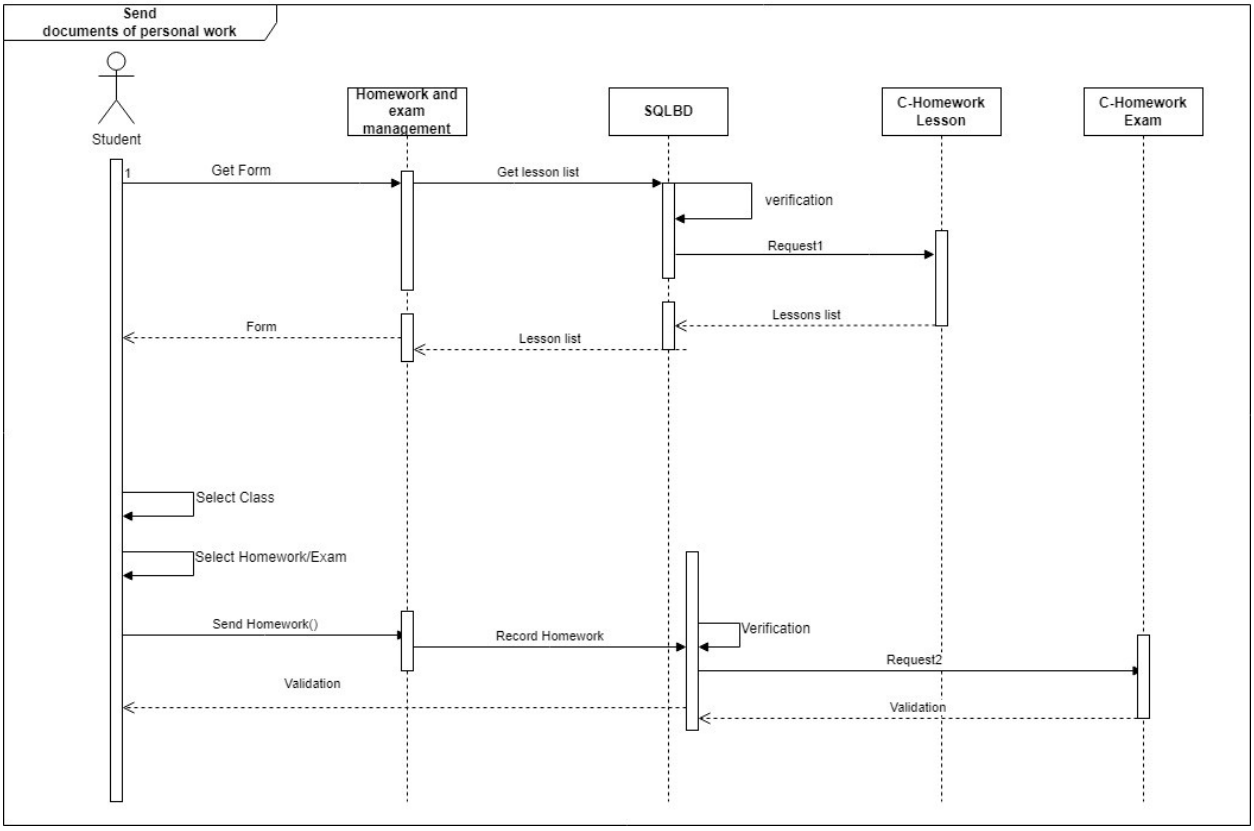
The student can

- The student asks to consult the Courses and personal work sent by teachers.
- The student creates the corresponding work documents.
- The system displays the page.
- The student sends the documents.
- The system is recording.

Based on the interactions discussed above, the sequence diagram could be:







The administrator manages the events on the site by adding new ones and deleting old ones or, in case of cancellation, modifying their names and descriptions...

State Actor and Object of this sequence:

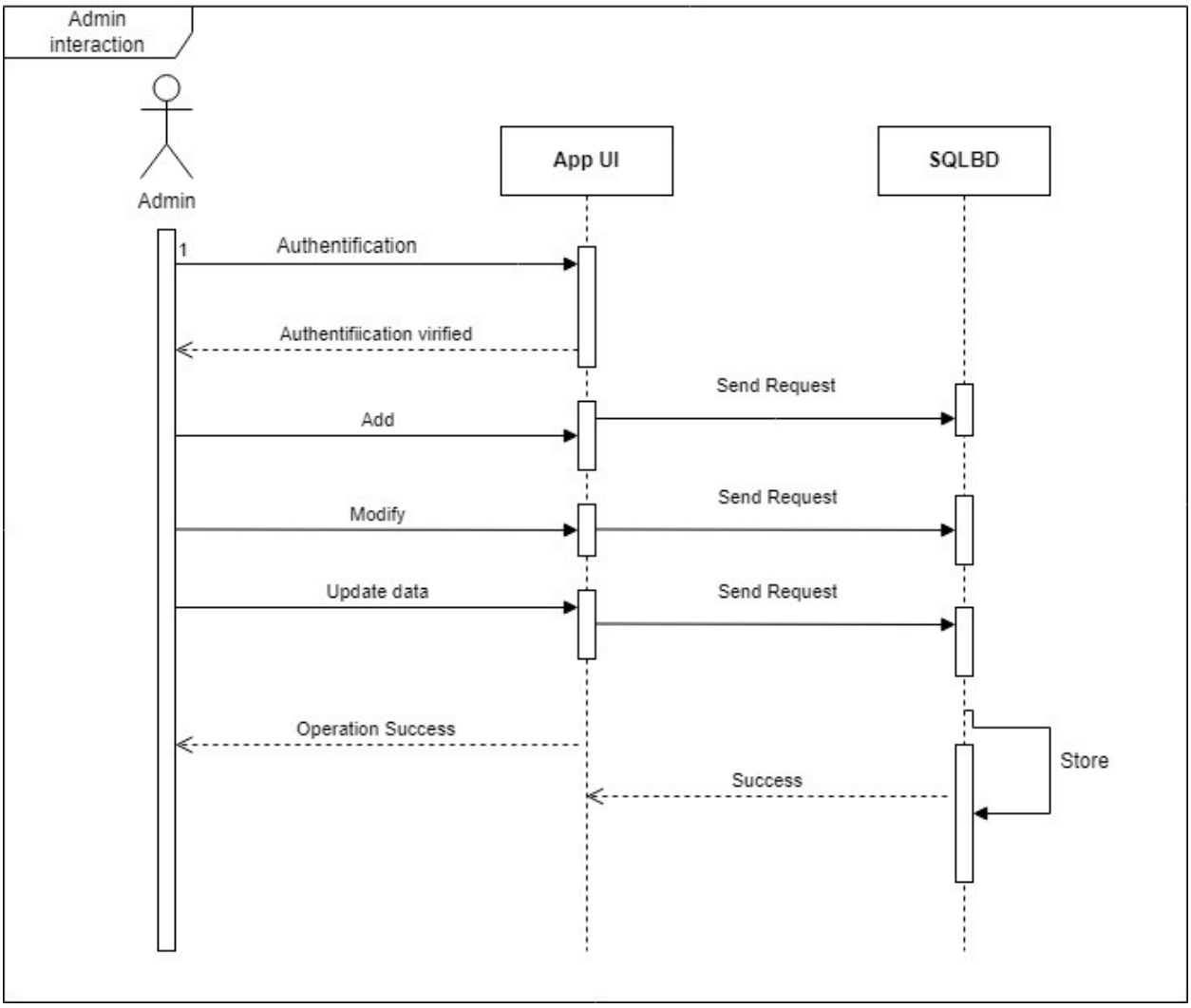
Actor: **Admin**

Object: Proposition assignment interaction diagram

Steps for Admin interaction:

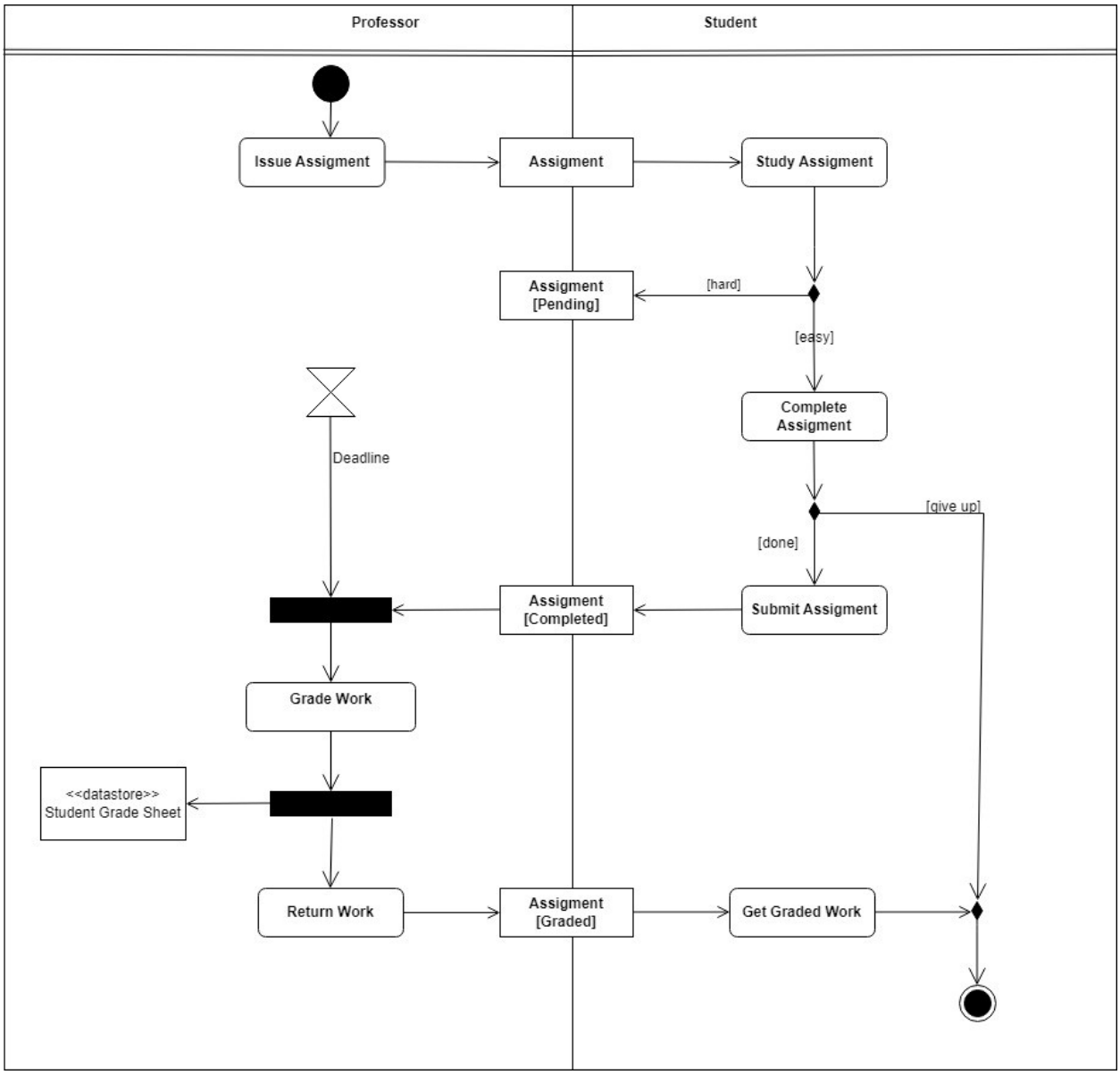
The admin can

- Register one or more students in the system.
- View their information and update or modify it.
- Update their level at the end of the year.
- Add or modify timetables (semester, control, exam, and remedial).
- Add one/several modules with their level, credit, and coefficient.
- Add one/several grades (control, exam, practical work, reset) of a module according to its level and the academic year.
- Modify the grade of any module according to its level in the academic year.
- Add all types of lesson formats (image, a text document, PDF…) and have the possibility to modify them. Based on the interactions discussed above, the sequence diagram could be:


Admin interaction
Admin
App UI
SQLBD
1
Authentification
Authentifiication virified
Send Request
Add
Send Request
Modify
Update data
Send Request
Operation Success
Store
Success

V. activity diagram

The application is a website. It has two interfaces. The first is dedicated to the Professor who manages the various data concerning the courses, the students, the modules, the timetables, and the events. The Professor interface includes a home page. The latter has easy-to-use interfaces. The organization of the pages is achieved using tables and CSS style sheets.



VI. user interface specification

Welcome page

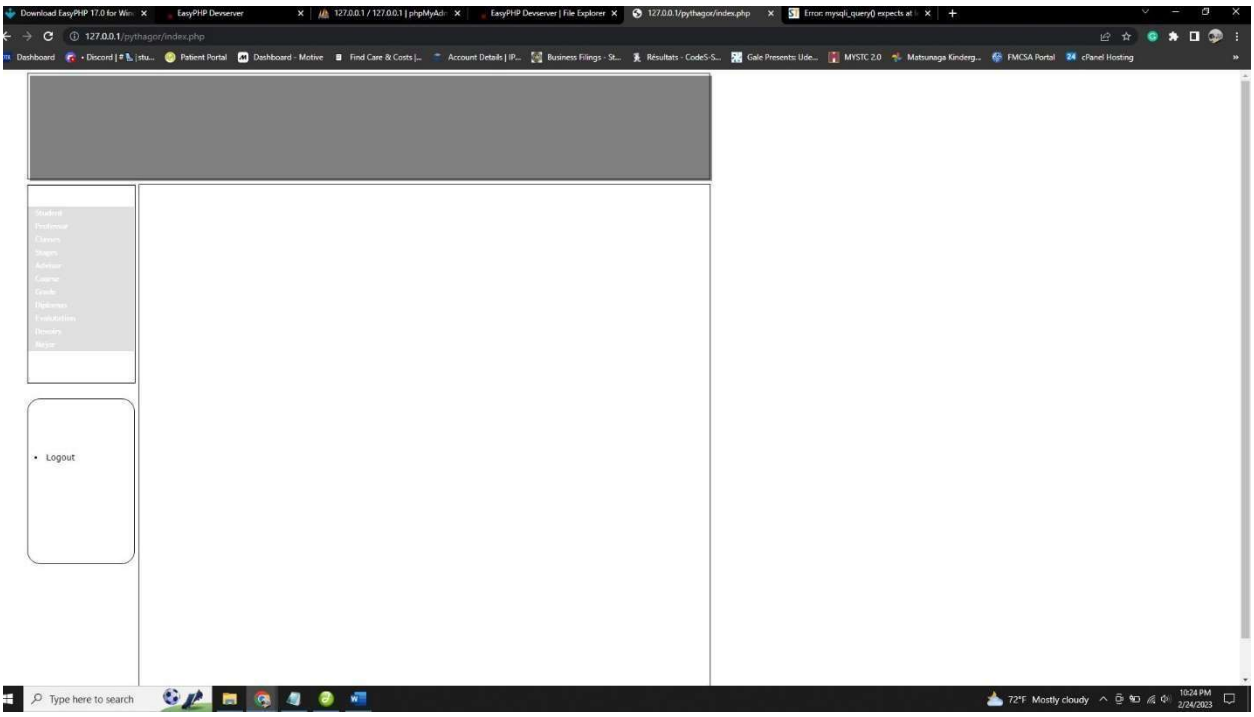
Menu:

- Student
- Professor
- Classe
- Internship
- Courses
- Diplomas
- Major

Admin Page

Menu:

- Students
- Professor
- Classe
- Internship
- Courses
- Diplomas
- Major



Professor page

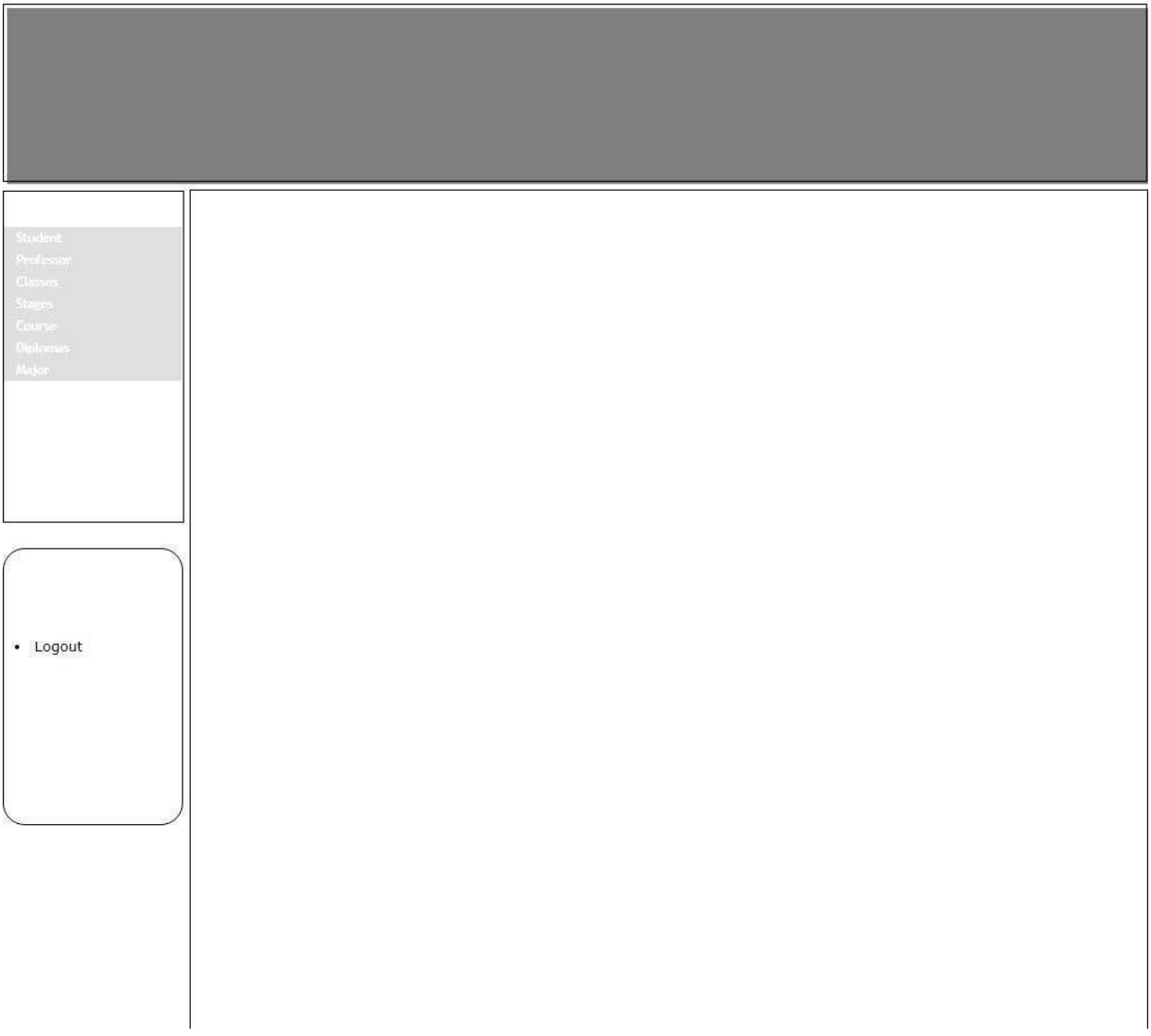
Menu:

- Student
- Professor
- Classes
- Internship
- Advisor
- Course
- Grade
- Diplomas
- Homework
- Major

Student page

Menu:

- Student
- Professor
- Classes
- Internship
- Degre
- Major



VII. project plan

Up to now, we have been done with the functionality. The next step is to focus on the graphist, set up the Pipeline for Dev and Prod environment, set up a separate server with a database, work on security issues, and set up the DNS for the app.

VIII. reference

Web App: http://173.255.197.34/index.php

Admin account : User: admin Password: admin

Professor account: User Professor Password: abc123

Student account: User student Password: abc123

Database: http://173.255.197.34/phpmyadmin/

User: oracle Password: abc123

Repository: https://github.com/ngahadjo/pythagor

Public