Cairo University
Faculty of Computers and Information Department of Computer Sciences

Graduation Project Organizer

Supervised by

Prof. Dr. Hesham Ahmed Hassan

TA. Sarah Ahmed Elnady

Implemented by

20160066	Ameer Mohamed Abd Elwahab
20170110	Sarah Samer Moustafa
20170111	Sarah Saeed Ebraheem
20170115	Salma Essam Soliman

Graduation Project

Academic Year 2020-2021

Final Documentation

Abstract

Forming a graduation project team is a difficult process especially when the number of registering students is increasing. This makes reaching out to professors and teaching assistants harder because the more teams formed, the more emails professors get and they may not have a chance to read them all or to see the team's capabilities and proposed projects. So, the process of selecting a professor or a teaching assistant to supervise the team's project is becoming effort and time consuming. To solve this problem, we propose developing an online platform to allow professors to view all project ideas of all teams and even request to supervise a specific idea if they like it or add some ideas they would prefer to supervise, search for a specific technology and set a maximum number of teams to work with. This also makes it easier for administrators to see the registered teams, their supervisors and their proposed ideas and to suggest a supervisor for unsupervised teams. The main reason why we wanted to try and solve this problem and facilitate this process is that we also had to deal with it and it took us quite some time to determine our idea and to find a supervisor. For this project, we will use several technologies such as web development, natural language processing and Restful API using tools and technologies like React, MongoDB, Node.js and python.

Table of Contents

Chapter 1: Introduction 10

1.1. Motivation 10

1.2. Problem Definition 11

1.3. Project Objective 11

1.4. Gantt Chart of Project Time Plan 12

1.5. Project Development Methodology 13

1.6. The Used Tools in the Project 14

1.7. Report Organization 15

Chapter 2: Related Work 16

Chapter 3: System Analysis and Design 17

3.1. Project Specification 17

3.1.1. Functional Requirements 17

3.1.2. Non-functional Requirements 18

3.2. Use Case Diagram and Sample Use Cases 19

Chapter 4: System Design 33

4.1. System Component Diagram 34

4.2. System Class Diagram 35

4.3. Sequence Diagrams 44

4.4. Project ERD 54

4.5. System GUI Design 55

Chapter 5: Implementation and Testing 64

Chapter 6: Conclusion and Future Work 73

References 74

List of Figures

Figure 1: Tasks Gantt Chart	12
Figure 2: Use Case Diagram	19
Figure 3: System Architecture Diagram	33
Figure 4: System component Diagram	34
Figure 5: System class Diagram	35
Figure 6: “Register” Sequence Diagram	44
Figure 7: “Student Login” Sequence Diagram	44
Figure 8: “Get All Public Projects” Sequence Diagram	45
Figure 9: “Search and filter students” Sequence Diagram	45
Figure 10: “Get Profile” Sequence Diagram	46
Figure 11: “Edit Supervisor Profile” Sequence Diagram	46
Figure 12: “Get Student Team Info” Sequence Diagram	47
Figure 13: “Get All Teams Projects” Sequence Diagram	47
Figure 14: “Student Add Project” Sequence Diagram	48
Figure 15: “Edit Project” Sequence Diagram	48
Figure 16: “Get Notifications” Sequence Diagram	49
Figure 17: “Delete Project” Sequence Diagram	49
Figure 18: “request to join another student's team” Sequence Diagram	50

Figure 19: "Ask Team Members to Take Dr's Project" Sequence Diagram	50
Figure 20: "Ask Team members to make supervisor take an idea" Sequence Diagram	51
Figure 21: "Students vote to take Dr's idea" Sequence Diagram	52
Figure 22: "Get All Projects Supervised by a Supervisor" Sequence Diagram	52
Figure 23: "Get Admin Statistics" Sequence Diagram	53
Figure 24: Project ERD	54
Figure 25: Admin Login Page	55
Figure 26: Students & Supervisors Login Page	55
Figure 27: Control Center (Statistics Page)	56
Figure 28: Students Data Table	57
Figure 29: Doctors Data Table	58
Figure 30: Departments Data Table	58
Figure 31: Student's own Profile	59
Figure 32: Add Project	59
Figure 33: Team Information	60
Figure 34: Search All Students	60
Figure 35: List of Old Ideas	61
Figure 36: View other Student's Profile	61

Figure 37: View other Student's Team	62
Figure 38: Student ask supervisor to supervise a project	62
Figure 39: View other Supervisor's Profile	63
Figure 40: Student login pass output	64
Figure 41: student login fail output	64
Figure 42: Add team project pass output	65
Figure 43: Add team project fail output	65
Figure 44: Get team info pass output	66
Figure 45: Search students pass output	66
Figure 46: Search students fail output	67
Figure 47: Get supervisor's project pass output	67
Figure 48: take supervisor's project pass output	68
Figure 49: take supervisor's project fail output	68
Figure 50: get student profile pass output	69
Figure 51: get student profile fail output	69
Figure 52: admin login pass output	70
Figure 53: admin login fail output	71
Figure 54: supervisor login pass output	71

Figure 55: supervisor login fail output	72

List of Tables

Table 1: Ecom authentication	**21**
Table 2: register	**21**
Table 3: user login	**22**
Table 4: get and filter old projects	**22**
Table 5: search and filter users	**23**
Table 6: get profile	**23**
Table 7: edit profile	**24**
Table 8: get student team info	**24**
Table 9: get all student team's projects	**25**
Table 10: add project	**25**
Table 11: edit project	**26**
Table 12: get notifications	**26**
Table 13: delete project	**27**
Table 14: request to join team	**27**
Table 15: ask team to get doctor's project	**28**
Table 16: ask team to have a supervisor	**28**
Table 17: vote to take doctor's project	**29**
Table 18: vote to send request to doctor to supervise on project	**29**
Table 19:get supervisor's supervised projects	**30**
Table 20: TA requests to supervise team	**31**
Table 21: request to supervise team	**31**

Table 22: Dr request to supervise team project	**32**
Table 23: get statistics	**32**
Table 24: Student login	**64**
Table 25: Add project to student team	**65**
Table 26: Get team info	**66**
Table 27: Search students	**66**
Table 28: get supervisor projects	**67**
Table 29: take supervisor project	**68**
Table 30: get user profile	**69**
Table 31: admin login	**70**
Table 32: supervisor login	**71**

List of Abbreviations

1. **GP:** Graduation Project
2. **TA:** Teaching assistant
3. **NLP:** Natural language processing
4. **API:** Application programming interface
5. **MongoDB:** Mongo database
6. **Node JS:** open-source development platform for executing JavaScript code server-side
7. **TF-IDF:** Term frequency-inverse document frequency
8. **Doc2Vec:** Document to vector, an unsupervised algorithm to generate vectors for documents
9. **ERD:** Entity relation diagram
10. **GUI:** Graphical user interface
11. **Ecom:** Faculty of computers and artificial intelligence official website
12. **ML:** Machine learning
13. **JSON:** JavaScript Object Notation.

Chapter 1: Introduction

The majority of practical faculties, if not all of them, requires their senior students to develop a graduation project which is essential for them to graduate. The graduation project is a cornerstone of the academic life, and professional career, of many students, and so whatever time is invested in such a project is truly worthwhile. Furthermore, students' extending themselves beyond their limitations in this endeavor is essential to enhancing the concepts they learned in previous courses. Taking that into consideration, we find that the students have to organize and divide their tasks well according to the time they have. However, The process of making a graduation project starting with gathering team members of the same department, searching for valid supervisors and requesting their supervision, and searching for ideas, till reaching the implementation stage, consumes both time and effort. Each team needs a supervisor(s) to supervise and help with the graduation project. Unfortunately, the available means used for getting a confirmation for choosing an idea or a supervisor are limited. The most common means of communication are by sending mails and/or personally talking to the supervisor. This causes some difficulties and issues as some professors are not available all the time in the faculty or to check their mails quickly, some professors want to supervise specific ideas with specific technologies, and some professors may have completed their list of teams and can't supervise any extra teams anymore. Sending mails to multiple supervisors and waiting for a reply then changing the idea if needed is time consuming for the student. It is also very exhausting for the supervisors to receive large amounts of mails and having to reply to all of them as quickly as possible. So making a platform that is dedicated to manage and arrange this process may hopefully be helpful and useful for all its users.

1.1. Motivation

Forming and searching for a team to join in the graduation project (GP) has been a tedious task for students and what adds up to this is that when students are trying to contact professors/doctors. The students may receive many late replies, sometimes no replies at all if some doctors were busy. Some doctors may not be taking any teams and some may be maxed out on their teams and can't have any more teams to supervise. Also, some teams sometimes find out late that their ideas are similar or even the exact same. If this discovery was too late and one of the teams had to change their idea, it will cause a problem for them as they will have to restart searching for a new idea, get approval for this idea and repeat most of the steps of the graduation process which consumes a lot of their limited time that they are expected to have managed well, and instead of organizing the next step, they would have to repeat the current steps and even the previous steps as well. All these problems and issues can make the GP team process a lot more time consuming. This can waste a lot of opportunities for a lot of students to be able to work on their desired graduation project idea since it is not possible to easily reach out to doctors and present their ideas. This motivated us to develop an online platform to automate the whole graduation project process.

1.2. Problem Definition

Senior students face some hardships while choosing their graduation project idea and supervisors that consume time which could have been spent working on the actual project. They begin sending mails to multiple supervisors asking them to supervise their projects simultaneously since some emails may not be noticed or may be noticed too late which affect the speed of replying to those emails. Also, the latest emails sent are placed on top of the list of emails that were already sent to doctors, so the latest emails would probably be opened first which isn't fair to students who sent emails earlier. Moreover, some doctors prefer choosing to supervise specific projects that use certain technologies, and some doctors may already have some ideas in mind that they want the students to choose from. Also, students don't always know which doctors in their department are willing to supervise projects and which aren't this year, and which doctors are already done with choosing their teams. In addition, some ideas may be repeated and it may be found out too late causing the team to work from the start on a new idea.

Recently, with the increase in the number of GP teams and due to the reasons above, sending mails has proven to not be an efficient way of communication between the supervisors and the teams, and it can waste much time for students. Having a specific platform for this process would reduce the consumed time to only getting the response from a doctor to supervise the project which will make more time for the next steps in the GP process.

1.3. Project Objective

We propose developing an online platform to help students form teams and allow students in teams to ask doctors and TAs to be their supervisors. Doctors and TAs can accept or reject a request and can view all project ideas of all teams and even request to supervise a specific idea if they like it, or add some ideas they would prefer to supervise, search for a specific technology and set a maximum number of teams to work with and supervise. This platform should also help students choose team members or supervisors that are experienced in a specific field of technology through recommendations in order to provide the needed support during the project stages and development. In addition, any duplicate projects should be detected and the involved students should be notified early so that they can change their idea early and begin working on the new one. This also makes it easier for administrators to see the registered teams, their supervisors and their proposed ideas, and to suggest a supervisor for unsupervised teams.

This project will be a useful tool for senior students, professors/doctors, TAs and administrators in university. It will help students find other students and form a team for their graduation project since students will be able to search for other students or doctors/TAs based on their preferred technologies. Furthermore, students in teams can ask doctors and TAs to be their supervisors, meanwhile the supervisors will be presented with a list of team members, including their project ideas and the technologies that will be used in them, from which the doctor/TA can select projects to supervise. Doctors and TAs can accept or reject a request and can even search for a specific idea or

technology to work with and offer that team to supervise them if they are interested in their idea. Doctors can even add a list of projects that they would like to work with and any team can take up that project to be supervised by that doctor. Team registration reports can be generated for the administrators with team members' names, IDs, supervisors, etc. Overall, this tool will save time and will make the process easier and more organized for students, admins, doctors and TAs.

1.4. Gantt chart of project time plan

Figure 1: Tasks Gantt Chart part 1

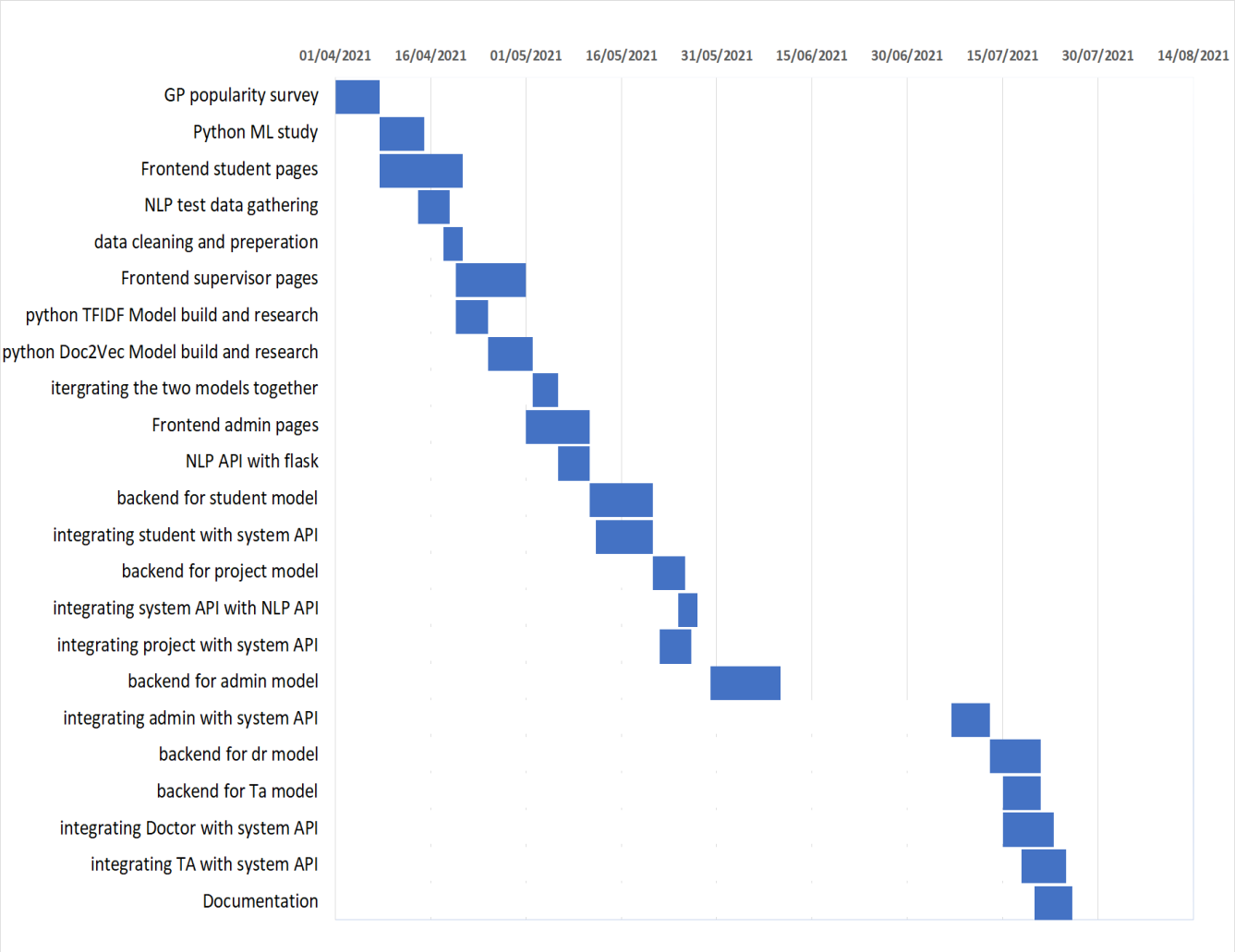


Figure 1: Tasks Gantt Chart part 2

1.5. Project development methodology

Developing a project can be a difficult process, so there are some methodologies that exist to help the developers design and develop their project efficiently without missing any steps and make it harder on themselves. The most common used methodologies are:

1. Waterfall methodology.
2. Agile methodology.
3. Agile-Waterfall hybrid methodology.

The used methodology in this project is the Agile-Waterfall hybrid.

Waterfall methodology

It's a methodology based on a sequential design process. The steps of the design process are arranged in a sequential order so that the project flows from requirement gathering and documentation to system design, then to implementation and coding, followed by testing then deployment and finally maintenance. This methodology is best used with small projects with requirements that are unlikely to change. The methodology is easy to understand and manage, but it is inflexible so it is nor recommended with complex projects.

Agile methodology

This methodology's first priority is development through evolution. It is centered around the idea of iterative development and involves a continuous improvement at every stage of the development process. The developers using this methodology finish all the stages of the design process for each functional requirement before beginning working on the next one.It is a flexible methodology that is best used with the software that can evolve with iterations.

Agile-Waterfall hybrid methodology

This methodology mixes the previous two methodologies and incorporates the best of them both. The reason why this methodology exists is that Agile is more of an approach than a methodology, so it is possible to combine it with Waterfall.

In this project we preferred using the third methodology, the Agile-Waterfall hybrid. We used Agile on enterprise level and Waterfall for requirements gathering, design and implementation. We preferred using this method because it was the most suitable method in terms of the given time and the system has high dependency so we couldn't finish completely a component before making another. For example: we couldn't finish the student model alone before as it heavily depends on adding projects, joining a team and searching for supervisors, so we had to finish the system design completely before beginning with the implementation.

1.6. The used tools in the project

For this project, we will use several technologies such as web development as we want the platform for the project to be a website so that the needed hardware would be any device that is connected to the internet whether it is a mobile phone, laptop, PC, etc. We will also be using NLP and Restful API using React, MongoDB, NodeJS, TypeScript, docker and python with Flask as they are the most popular and the most used tools nowadays by a lot of people working in the field.

Using machine learning will be necessary as it will help us in developing some functionalities such as detecting similar projects to notify the students. We will be using both TF-IDF and Doc2vec which are natural language processing methods to detect near duplicate project ideas introduced by students or Doctors which will reduce the common mistake of having two teams proposing the same project idea.

We prefer web development rather than making a mobile application as some students may not have enough memory on their mobile phones to download the application or they may have an old mobile phone that doesn't have the privilege of downloading mobile applications. Also, it allows the user to use it anytime from any device without having to pre-download anything in case of using different devices.

We used Docker with MongoDB, System API and NLP python API.

1.7. Report Organization

In the rest of the report, we will be introduced to the related ideas and related work. we will be able to see the similarities and the main differences between their project and ours in the terms of main idea and functionalities that can be found in chapter 2

Then, in chapter 3 we will be able to know more about our project by being introduced to the system analysis which helps in knowing more about the system's functional and nonfunctional requirements. We will also be introduced to the use case diagram which will make it easier to know who the users of our software are and what their main functionalities are so that we can imagine the flow of the project and its sequence from the user logs in till he logs out.

Afterwards, in chapter 4 we will be introduced to the system design which will help in understanding the project's software design through some elements such as the class diagram, sequence diagrams and ERD. They help in showing many things such as the classes, attributes, and functions, as well as the flow of the code when implementing each functional requirement. It will also display some of the used GUI.

Finally, we will be introduced to some samples of the applied test cases of the implemented system that can be found in chapter 5.

Chapter 2: Related Work

Unfortunately, there were no applications developed to solve/facilitate the GP team formation problem. A research was performed on organizing undergraduate research projects and it mentioned 2 models. The authors declared that the first model, the 'student-led' model, is the most used model. It indicates that the students design their own research project and identify and approach a member of staff to be their supervisor. Students contact supervisors directly via email or in person, and it is up to the staff member to agree to supervise the student or refer them to someone else to supervise them instead. This model raises several issues for the undergraduates. First, most of the undergraduates do not know where to start in choosing a research topic to take as their project idea as most have no practical research experience. Generally, prior to their senior year, students have not tackled projects of the magnitude of the graduation project. They are accustomed to short-term projects and assignments that do not involve deep analysis or the need to address real-world problems and that makes choosing an appropriate graduation project harder for them. In addition, doctors and TAs might have to supervise projects about which they have little knowledge or interest, where they learn one step ahead of the students or revise their knowledge in order to help them and guide them when they need.

An alternative to the student-led model is a student-choice model that is concerned with the creation of a catalogue of project summaries designed by the supervisors either as new projects, or as upgraded versions of old projects. In this model, students receive a catalogue, attend a project day in which they learn more about the idea of the projects and a brief description about them, and have an opportunity to ask questions to the supervisors in order to know more about the projects. They then select a top-five list of projects, and email this to the person responsible for assigning ideas to the teams. Students can talk to supervisors before they choose, but importantly staff are not allowed to independently decide to take on a student. Instead, students are allocated based on a number of factors. Among the issues in this model is that the allocation evidently takes some staff time and potentially creates tension within the staff team and conflicts may arise.

The most familiar software that is similar to our project idea is a software that manages the projects allowing the students to organize their tasks and deadlines with minimum effort, enables supervisors to easily keep track of students and their progress, and ultimately permit graduation project committee members to monitor all transactions taking place to ensure that all is well. However, this software focuses on organizing the steps that follow the team formation and project idea choosing. It focuses on what comes after choosing all team members, supervisors, and the project idea depending solely on exchanging emails between the students and the supervisors as the means of choosing all of that. However, we couldn't find any faculty that has a specific tool in its website or application that incorporates a blend of both models in the GP formation process. The only method used in this area is emails.

Chapter 3: System Analysis and Design

This chapter contains the project specifications and the use case diagram for the proposed project:

3.1. Project Specification

The Following list contains the stakeholders:

1. Senior Students
2. Professors/ Doctors
3. TAs
4. Admins (Faculty staff members)

3.1.1. Functional Requirements

The Following list contains the functional requirements:

1. Students and supervisors can register using a link provided by the faculty's Ecom website.
2. Students and supervisors can update their experienced technologies, add notes and specify the maximum number of teams to take in case of a supervisor in their personal profile to be seen by other stakeholders.
3. Students can view the feeds announced by the admin.
4. Students can view a list of the old accepted GP ideas of any specified year and filter them.
5. Students can view a list of all ideas presented by all the doctors in their department.
6. Students can search for other students/doctors/TAs, filter the search results and view their profile.
7. Students can send requests to join other students in their team.
8. Students can view their team information and their team projects list.
9. Students can add, edit and delete a project from their team projects list.
10. Students can not add a duplicate GP idea or edit an old idea to become a duplicate.
11. Students can view a list of all the received requests to join their team, to choose a supervisor, or choose a main GP idea and vote to accept or decline these requests.
12. Students can ask their team members to request to be supervised by a certain supervisor or choose their main GP idea and vote to accept or decline before taking these actions.
13. Students and supervisors can leave the team and rejoin another one before the project registration deadline and the rest of the team should be notified.
14. Students and supervisors should be notified if anyone sends them requests, announcements, and team information updates.
15. Supervisors can view all projects provided by all teams in all departments.
16. Doctors can ask to supervise a specific GP idea of any team with no Doctor to supervise yet.
17. TAs can ask to supervise any team without choosing a specific idea.
18. Supervisors can view a list of all the students and see their profile.
19. Supervisors can view a list of the teams that they already agreed to supervise.

20. Supervisors can search for an idea by filtering all the ideas with a specific technology as a keyword.
21. Doctors can add ideas to their own profile to be visible for all students to choose from in case the team wants to work with one of those ideas under the supervision of that doctor.
22. More than one doctor can supervise the same team. In order for that to happen, the team has to be supervised by at least one doctor then request other doctors to co-supervise them.
23. Admin can view data tables of all the students, teams, departments, doctors, and TAs.
24. Admin can add, edit, and delete any row in the data tables of doctors, TAs, students, departments, and teams.
25. Admin can search and filter all the data tables.
26. Admin can add announcements to be seen by other stakeholders.
27. Admin can list and view all the teams in all the departments and see their members, supervising doctors, supervising TA, the main project idea, and its technologies.
28. Admin can view statistics about the number of students and supervisors in the system, number of teams lacking a supervisor and the number of formed teams with the supervisors in each department.
29. Admin will have the ability of matchmaking supervisors with empty team(s) with the teams that lack supervisor(s) (Doctor/ TA/ both).

3.1.2. Non-Functional Requirements

The following list contains the non-functional requirements:

1. **Ease of Use:** The average training hours needed by the actor to use the website is 4 hours.
2. **Fault Tolerance:** The Number of faults when performing an action shouldn't exceed 3.
3. **Efficiency:** The average time needed for screen refresh and event response is 1 sec depending on the efficiency of the internet connection.
4. **Portability:** The number of platforms that the website can run on exceeds 2 (windows, Linux, Android, etc.), It can run on any platform connected to the internet.
5. **Reliability:** The system should be available 24 hours a day. The rate of failure occurrence should be less than 3 times per month. Mean time of failure shouldn't exceed 1 hour.
6. **Robustness:** The time needed to restart after failure shouldn't exceed 30 minutes.
7. **Flexibility:** The system is flexible as the micro service can be replaced by any other service easily.
8. **Scalability:** The website can handle many requests without server failure as processes are divided across the micro services.
9. **Compatibility:** The website can be used without affecting any other running program.

3.2. Use Case Diagram and Sample Use Cases

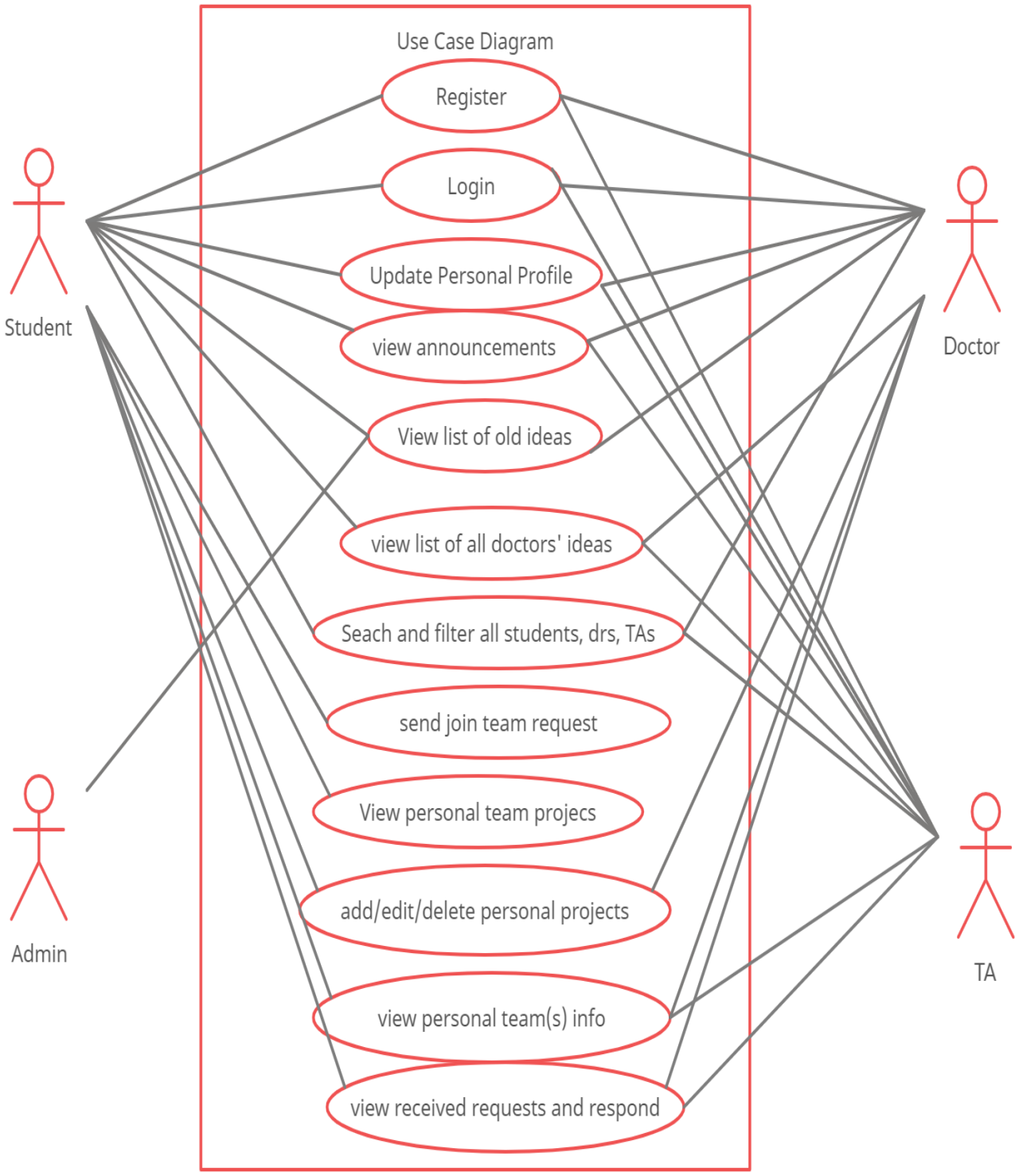


Figure 2: Use Case Diagram_part 1

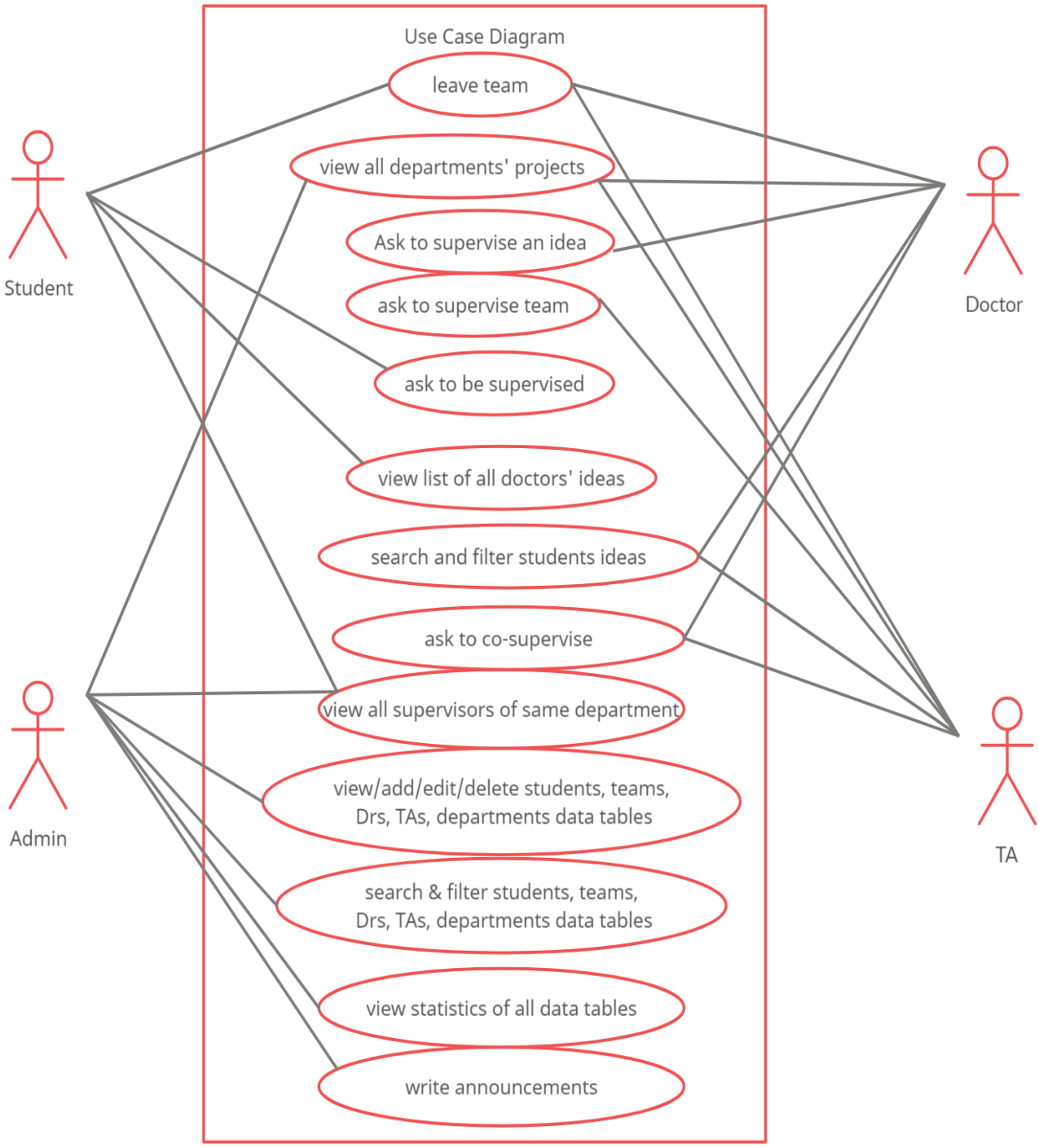


Figure 2: Use Case Diagram_part 2

Use Case Tables

Use Case ID	1
Use Case Name	ecom authentication
Overview	The actor gets a link generated from the ecom that has a token used for registering the user into the system
Actors	Student and supervisors
Pre-conditions	1-The actor must have an account in ecom
Sequence Flow	

User Action	System Action
user login in ecom and press the authentication link	
ecom generates a token with the user ID, name, department & user type (student, Dr, or TA)	
ecom sends the token to the system	
	receive & decrypt the token & the check the key
	the user is asked to register

Post-conditions	user can register his information to the system
Exceptions	if the key is wrong: the user can't register
Notes	None

Use Case ID	2
Use Case Name	register
Overview	the user with the data received from the sent token is required to enter his personal data before logging in for the first time.
Actors	Students and supervisors
Pre-conditions	1-The actor must have an account in ecom 2- the token key must be correct
Sequence Flow	

User Action	System Action
user enters the required data and password	
	validate the user input data
	register the user with the entered data & the data from the token.
	transfer the user to the login page

Post-conditions	the user can login with the password entered and his id
Exceptions	if the required data violates the API rules
Notes	None

Use Case ID	3
Use Case Name	user login
Overview	Providing the actor to login
Actors	Student and supervisors
Pre-conditions	1-The actor must have an account in ecom
Sequence Flow	

User Action	System Action
User enter ecom ID, password and check box for students	
	Enter the user to the system

Post-conditions	The user will be logged in
Exceptions	System will show message to user if data that user is entered are rong
Notes	None

Use Case ID	4
Use Case Name	Get and filter old projects
Overview	Doctors, TAs, students and admins can get old projects and can filter them using department, technologies, description or/and year.
Actors	Doctors, TAs, students and admins
Pre-conditions	user must be logged in
Sequence Flow	

User Action	System Action
User press on old projects	
	View old projects with filter
User filter with words using department, technologies, description or/and year.	
	view old projects that return from filtering.

Post-conditions	The user will get old projects
Exceptions	if system failed to get data will show massage
Notes	the same in get and filter supervisor projects the only different user will pressed on staff projects

Use Case ID	5
Use Case Name	search & filter users
Overview	The actor can search all users of a specific type to view them or filter to get specific user(s)
Actors	Students and supervisors
Pre-conditions	1-The actor must be logged in
Sequence Flow	

User Action	System Action
enters the name of a user and choose their type or only choose the type	
	fetch the inputs, if the name is specified it will get all the users of the same name and type & show the filter form
	else it will get all the users of the specified type & show the filter form
can enter data in the filter form to filter the results	
	in case of filtration: it takes the input and fetch the data from DB of specified input

Post-conditions	the user can view the search results
Exceptions	if user filters with wrong inputs, the system will return nothing
Notes	the same filtration process is used with filtering old ideas and supervisors' ideas

Use Case ID	6
Use Case Name	Get profile
Overview	Doctors, TAs and students can get their profile.
Actors	Doctors, TAs and students.
Pre-conditions	user must be logged in
Sequence Flow	

User Action	System Action
User press on profile	
	View user's profile

Post-conditions	Users will get their profile.
Exceptions	if system failed to get data, it will show message
Notes	None

<table>
<tr><td>Use Case ID</td><td>7</td></tr>
<tr><td>Use Case Name</td><td>Edit profile</td></tr>
<tr><td>Overview</td><td>Doctors, TAs and students can edit (notes, technologies) in their profile.
Doctors and TAs can also edit the max number of teams that they can take.</td></tr>
<tr><td>Actors</td><td>Doctors, ta and students.</td></tr>
<tr><td>Pre-conditions</td><td>user must be logged in</td></tr>
<tr><td>Sequence Flow</td><td><table>
<tr><th>User Action</th><th>System Action</th></tr>
<tr><td>1-User press on profile</td><td></td></tr>
<tr><td></td><td>2- View user’s profile</td></tr>
<tr><td>case 1: Edit note and press on add notes
case 2: The user chooses technology then presses on the plus button.
case 3: user press on “x” button (to delete technology)
case 4: if user is dr or ta :
change max number of teams</td><td></td></tr>
<tr><td></td><td>4- update data and view message</td></tr>
</table></td></tr>
<tr><td>Post-conditions</td><td>The users will edit their profile</td></tr>
<tr><td>Exceptions</td><td>if user input violates the rules or becomes a duplicate project, the project won’t be updated.</td></tr>
<tr><td>Notes</td><td>None</td></tr>
</table>

<table>
<tr><td>Use Case ID</td><td>8</td></tr>
<tr><td>Use Case Name</td><td>get student team info</td></tr>
<tr><td>Overview</td><td>students can get their team info.</td></tr>
<tr><td>Actors</td><td>Students.</td></tr>
<tr><td>Pre-conditions</td><td>user must log in</td></tr>
<tr><td>Sequence Flow</td><td><table>
<tr><th>User Action</th><th>System Action</th></tr>
<tr><td>User press on team info</td><td></td></tr>
<tr><td></td><td>View user’s team info</td></tr>
</table></td></tr>
<tr><td>Post-conditions</td><td>Users will get their team info.</td></tr>
<tr><td>Exceptions</td><td>if system failed to get data will show message</td></tr>
<tr><td>Notes</td><td>None</td></tr>
</table>

Use Case ID	9
Use Case Name	get all student team's projects
Overview	The actor view all their previously added projects if exists
Actors	Student
Pre-conditions	1-The actor must be logged in as a student
Sequence Flow	<table><tr><th>User Action</th><th>System Action</th></tr><tr><td>click on team projects button to view</td><td></td></tr><tr><td></td><td>view all the student team's projects</td></tr></table>
Post-conditions	the user can view their team's projects
Exceptions	if user has no previously added projects, the system will return nothing
Notes	the same process is used with getting the doctors' projects

Use Case ID	10
Use Case Name	add project
Overview	students and doctors can add projects.
Actors	Students and doctors.
Pre-conditions	user must log in
Sequence Flow	<table><tr><th>User Action</th><th>System Action</th></tr><tr><td>case 1: user is student - User presses on team project case 2: user is doctor - User presses on my projects</td><td></td></tr><tr><td></td><td>View team's projects if user is student, view doctor's projects id user is doctor</td></tr><tr><td>user presses on Add project</td><td></td></tr><tr><td></td><td>view model has a title, description and button for the user.</td></tr><tr><td>user enters title and description then press on add project</td><td></td></tr><tr><td></td><td>view message</td></tr></table>
Post-conditions	Users will get their team info.
Exceptions	if system failed to get data will show message
Notes	None

Use Case ID	11
Use Case Name	edit project
Overview	The actor can edit a previously added project
Actors	Students and doctors
Pre-conditions	1-The actor must be logged in 2-The project must be added by them or a project of the same team
Sequence Flow	

User Action	System Action
click on the edit button specified for the wanted project	
update the project's content and press update	
	updates the new information

Post-conditions	the users can view the new edited projects
Exceptions	if user input violates the rules or becomes a duplicate project, the project won't be updated.
Notes	None

Use Case ID	12
Use Case Name	get notifications
Overview	users can get their notifications.
Actors	Students, supervisors.
Pre-conditions	user must log in
Sequence Flow	

User Action	System Action
User press on notifications	
	View user's notifications

Post-conditions	Users will get their notifications.
Exceptions	if system failed to get data will show message
Notes	None

Use Case ID	13
Use Case Name	delete project
Overview	The actor can delete a previously added project
Actors	Students and doctors
Pre-conditions	1-The actor must be logged in 2-The project must be added by them or a project of the same team
Sequence Flow	

User Action	System Action
click on the delete button specified for the wanted project	
delete the project's from the projects list	
	updates the new projects list

Post-conditions	the users can view the new projects list
Exceptions	None
Notes	None

Use Case ID	14
Use Case Name	request to join team
Overview	The actor requests to join a team
Actors	Students and doctors
Pre-conditions	1-The actor must be logged in 2-The team must still have available positions
Sequence Flow	

User Action	System Action
choose any user in the wanted team and press ask to join	
	system notifies the team members

Post-conditions	the team members are notified
Exceptions	None
Notes	None

Use Case ID	15
Use Case Name	ask team to get doctor’s project
Overview	Users can ask their team to get a doctor's project .
Actors	Students.
Pre-conditions	1- user must log in 2- fetch doctor’s projects
Sequence Flow	<table><tr><th>User Action</th><th>System Action</th></tr><tr><td>User press on ask to supervise</td><td></td></tr><tr><td></td><td>send requests to all other members’ in the same team and notify them.</td></tr></table>
Post-conditions	Users will receive requests to get this doctor project and notification.
Exceptions	if system failed, it will show message
Notes	None

Use Case ID	16
Use Case Name	ask team to have a supervisor
Overview	Users can ask their team to ask for a specific supervisor’s supervision.
Actors	Student
Pre-conditions	1- user must be logged in 2- choose a project from project list to ask supervisor to supervise it
Sequence Flow	<table><tr><th>User Action</th><th>System Action</th></tr><tr><td>User press on ask to supervise</td><td></td></tr><tr><td></td><td>send requests to all other team members and notify them with the chosen project and chosen supervisor.</td></tr></table>
Post-conditions	Users receive requests and notifications then vote.
Exceptions	if having no projects in the projects lists, the student won’t be able to ask for the supervisor.
Notes	None

<table>
<tr><td>Use Case ID</td><td>17</td></tr>
<tr><td>Use Case Name</td><td>vote to take doctor’s project</td></tr>
<tr><td>Overview</td><td>Users can vote to take on a doctor's project or not .</td></tr>
<tr><td>Actors</td><td>Students.</td></tr>
<tr><td>Pre-conditions</td><td>1- user must log in
2- fetch requests</td></tr>
<tr><td>Sequence Flow</td><td>
<table>
<tr><th>User Action</th><th>System Action</th></tr>
<tr><td>case 1:
- user presses on accept button
case 2:
- user presses on decline button</td><td></td></tr>
<tr><td></td><td>if all team’s member vote :
- If the majority accepts, the system will assign this project to the team and send notification to the doctor and this team member.
- else system doesn’t do anything.
if not all team vote:
- system don’t do anything</td></tr>
</table>
</td></tr>
<tr><td>Post-conditions</td><td>Users will be assigned a doctor's project to them or not assigned and get notification.</td></tr>
<tr><td>Exceptions</td><td>if system failed, it will show message</td></tr>
<tr><td>Notes</td><td>None</td></tr>
</table>

Use Case ID	18
Use Case Name	vote to send request to doctor to supervise on project
Overview	Users can vote to send a request to a doctor to supervise their project or not.
Actors	Students.
Pre-conditions	1- user must log in 2- fetch requests

<table>
<tr><td>Sequence Flow</td><td>
<table>
<tr><th>User Action</th><th>System Action</th></tr>
<tr><td>case 1:
- user presses on accept button
case 2:
- user presses on decline button</td><td></td></tr>
<tr><td></td><td>if all team’s member vote :
- If the majority accepts, the system will assign this project to the team and send notification to the doctor and this team member.
- else system doesn’t do anything.
if not all team vote:
- system doesn’t do anything</td></tr>
</table>
</td></tr>
<tr><td>Post-conditions</td><td>send a request to the doctor or not send.</td></tr>
<tr><td>Exceptions</td><td>if system failed, it will show message</td></tr>
<tr><td>Notes</td><td>None</td></tr>
</table>

<table>
<tr><td>Use Case ID</td><td>19</td></tr>
<tr><td>Use Case Name</td><td>get supervisor’s supervised projects</td></tr>
<tr><td>Overview</td><td>the supervisor views their own supervised projects.</td></tr>
<tr><td>Actors</td><td>Supervisor</td></tr>
<tr><td>Pre-conditions</td><td>1- the user must be logged in as a doctor or a TA.
2- the user must have previously agreed to supervise project(s).</td></tr>
<tr><td>Sequence Flow</td><td>
<table>
<tr><th>User Action</th><th>System Action</th></tr>
<tr><td>User press on own supervised projects</td><td></td></tr>
<tr><td></td><td>view all the projects in the supervised projects list</td></tr>
</table>
</td></tr>
<tr><td>Post-conditions</td><td>User can view all their own chosen to supervise projects</td></tr>
<tr><td>Exceptions</td><td>if having no projects in the supervised project list, the list will be empty and the system will show nothing</td></tr>
<tr><td>Notes</td><td>None</td></tr>
</table>

Use Case ID	20
Use Case Name	TA requests to supervise team
Overview	User can ask to supervise a specific team
Actors	TA
Pre-conditions	1- user must be logged in as a TA
Sequence Flow	<table><tr><th>User Action</th><th>System Action</th></tr><tr><td>User press on ask to supervise</td><td></td></tr><tr><td></td><td>send requests to all team members and notify them with the TA that wants to supervise.</td></tr></table>
Post-conditions	Users receive requests and notifications then vote.
Exceptions	if the team already have a TA, the system won't accept sending the request
Notes	None

Use Case ID	21
Use Case Name	request to supervise team
Overview	Users can vote to send a request to a doctor to supervise their project or not .
Actors	TAs and doctors.
Pre-conditions	1- user must log in 2- fetch requests
Sequence Flow	<table><tr><th>User Action</th><th>System Action</th></tr><tr><td>case 1: - user presses on accept button</td><td></td></tr><tr><td></td><td>system will supervise this user to the team that sent this request and notify them.</td></tr><tr><td>case 2: - user press on decline button</td><td></td></tr><tr><td></td><td>system will notify team's members that sent this request.</td></tr></table>
Post-conditions	system will notify team's members if user request or reject if user accept, he will be supervisor this team
Exceptions	if system failed, it will show message
Notes	None

Use Case ID	22
Use Case Name	Dr request to supervise team project
Overview	the user can choose a specific idea and ask its team to supervise it.
Actors	Doctor
Pre-conditions	1- The user must be logged in as a dr. 2- choose a project from project list to ask team to supervise it
Sequence Flow	

User Action	System Action
User press on ask to supervise	
	send requests to all team members and notify them with the chosen project and the supervisor.

Post-conditions	a request is sent, and users receive requests and notifications then vote.
Exceptions	if having no projects in the projects lists, the doctor won't be able to ask for supervision.
Notes	None

Use Case ID	23
Use Case Name	get statistics
Overview	Users can view all statistics about all doctors, TAs, students, etc.
Actors	Admin
Pre-conditions	1- user must be an admin
Sequence Flow	

User Action	System Action
User press on admin control center	
	view all the statistics about the number of students, doctors, TAs, teams of each department, etc.

Post-conditions	view all the statistics
Exceptions	if all the data tables is empty, the statistics will be all zeroes
Notes	None

Chapter 4: System Design

The system uses client server system architecture which is a computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client. Client(s) computers are connected to a central server over a network or internet connection. The front-end will be a react single page application and will send requests to a restful API. The restful API has a python micro service for near duplicate projects detection. The system will use MongoDB to store users' data.

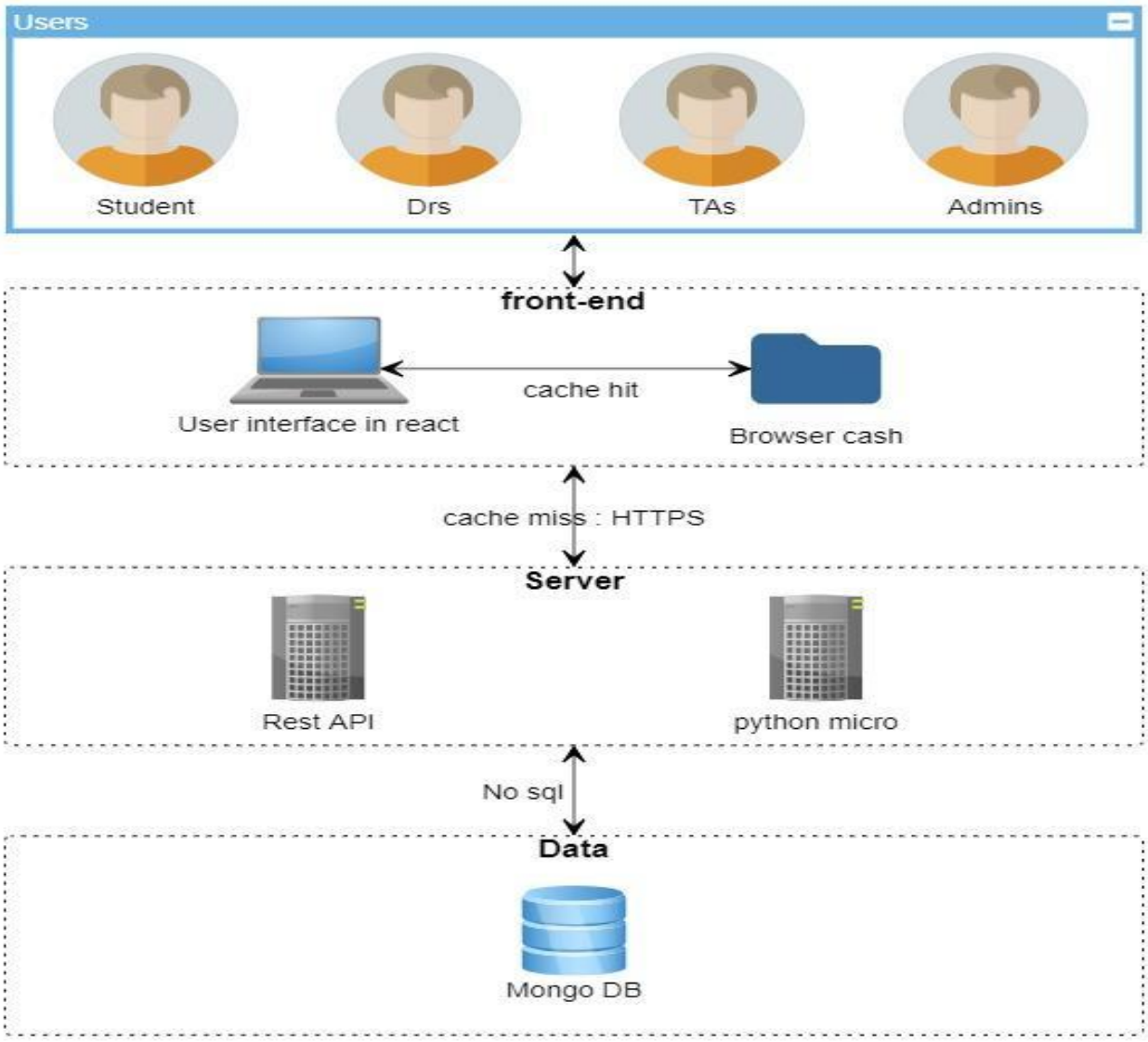


Figure 3: System Architecture Diagram

4.1. System Component Diagram

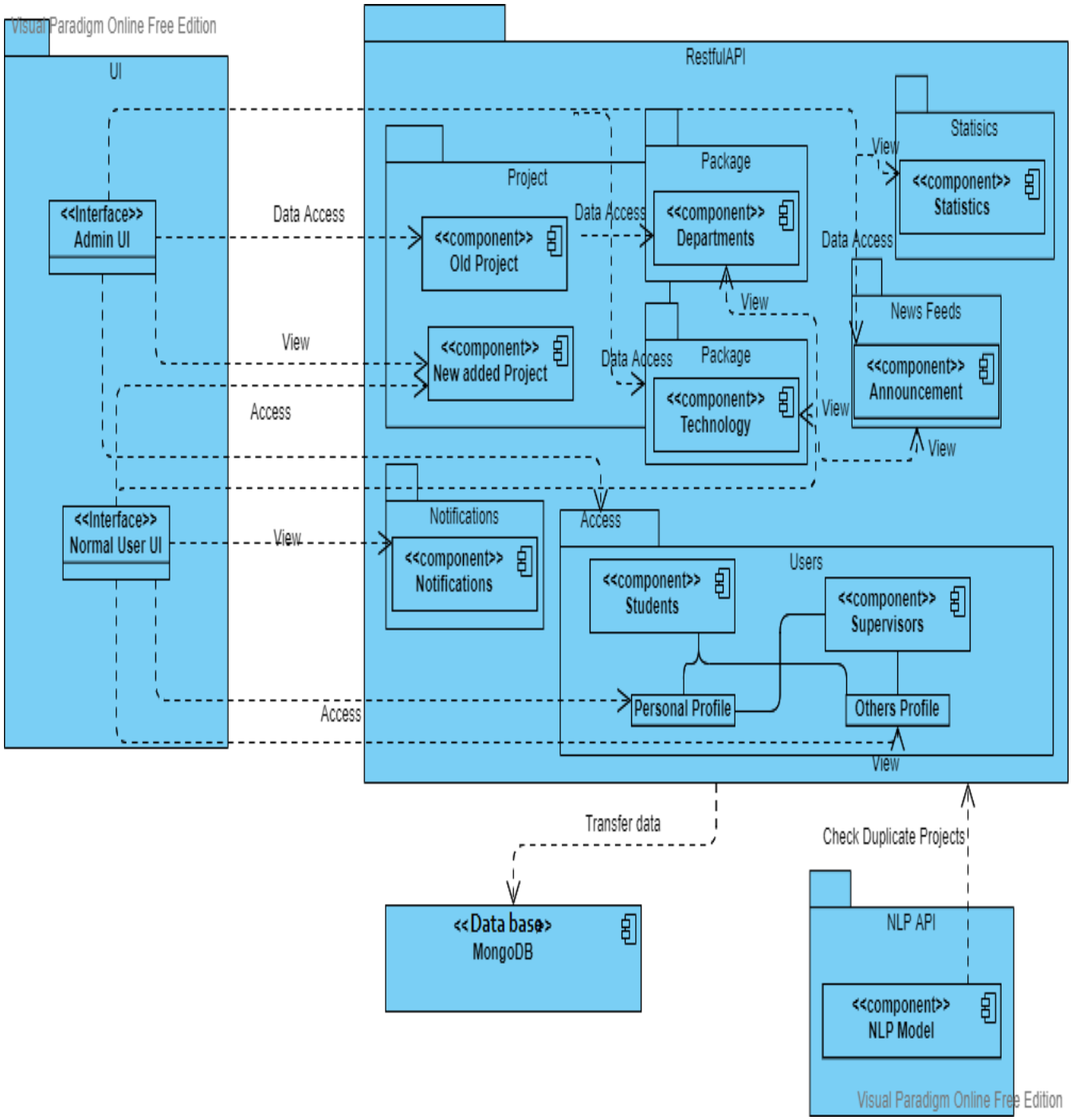


Figure 4: System Component Diagram

4.2. System Class Diagram

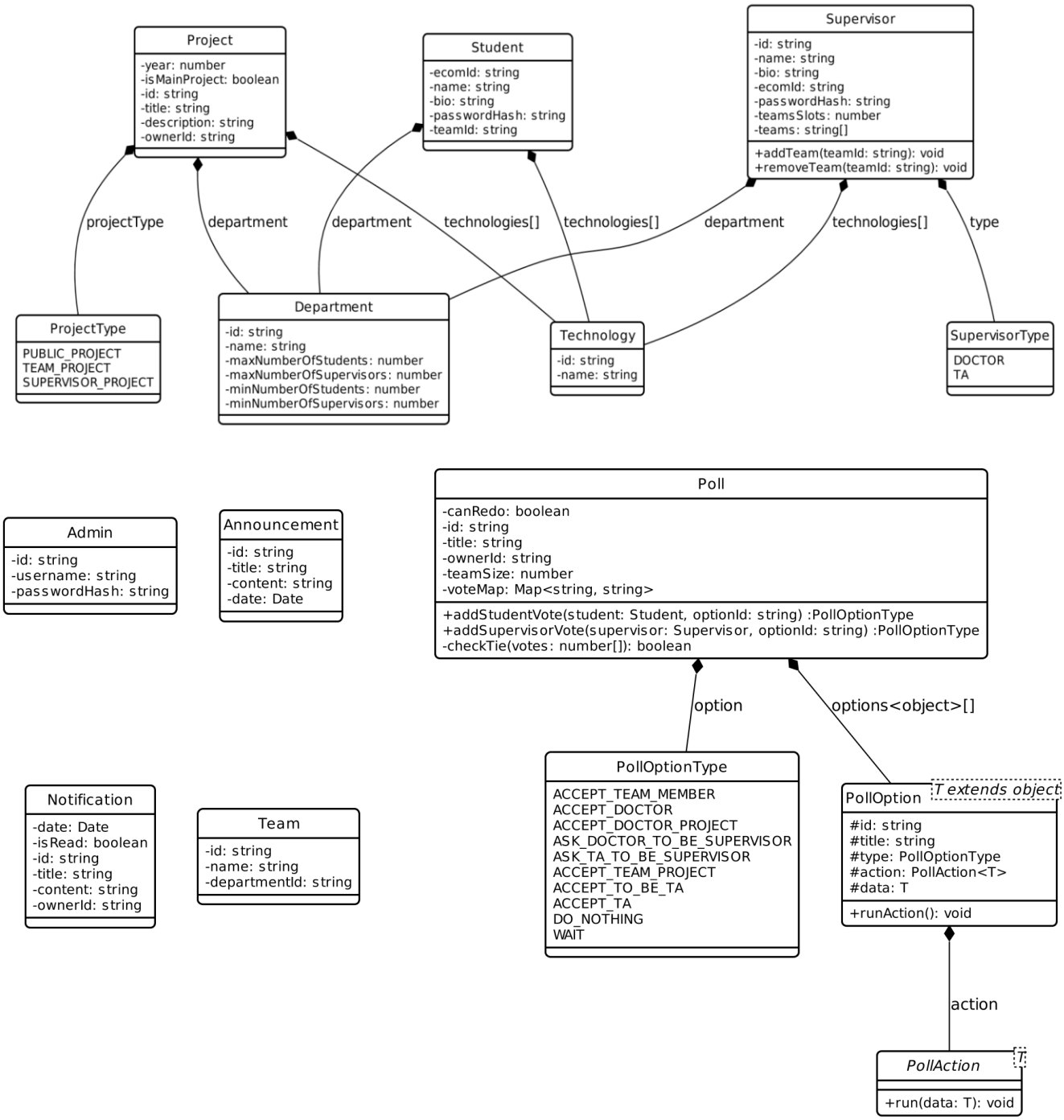


Figure 5: System Class Diagram part 1

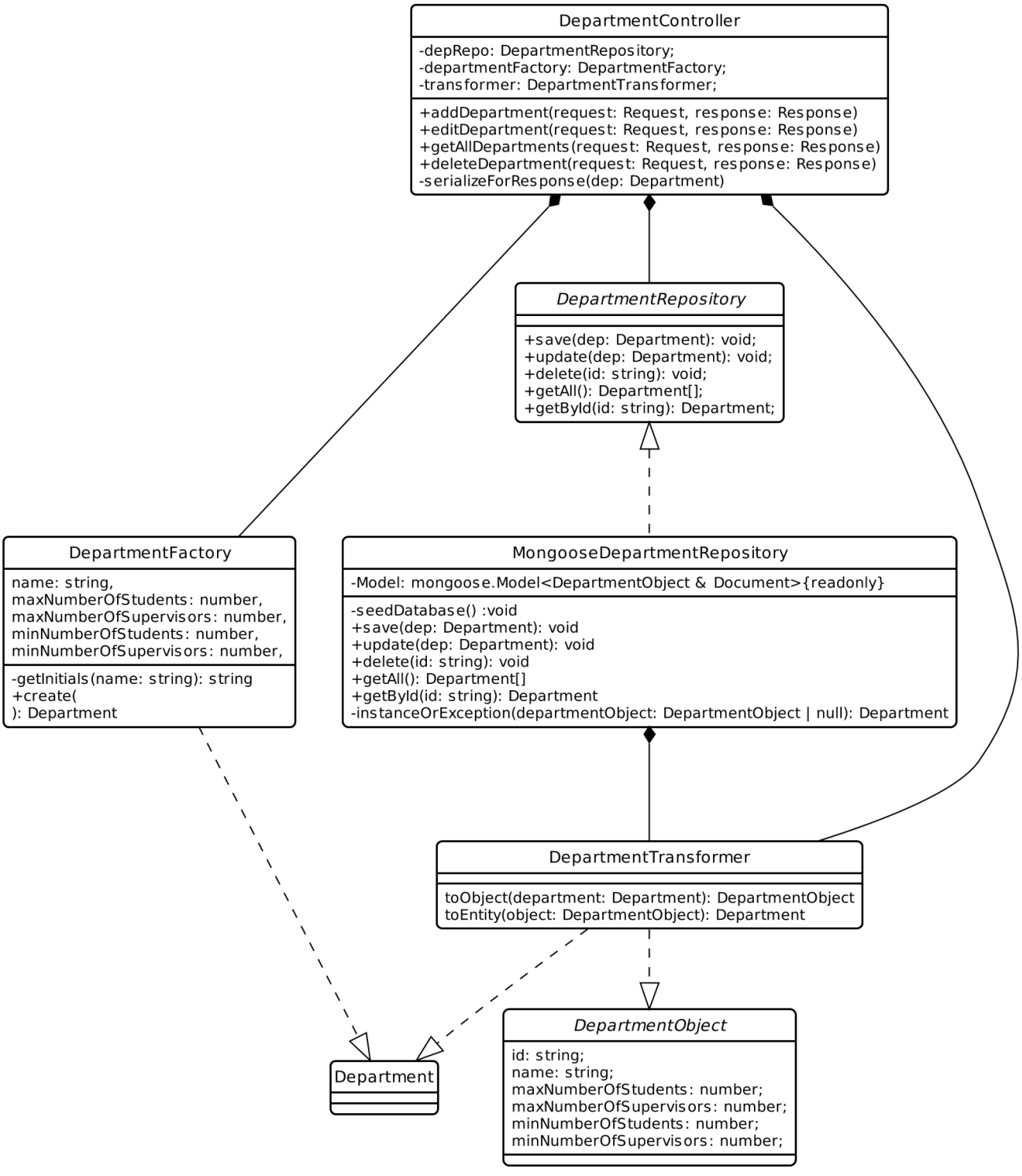


Figure 5: System Class Diagram part 2

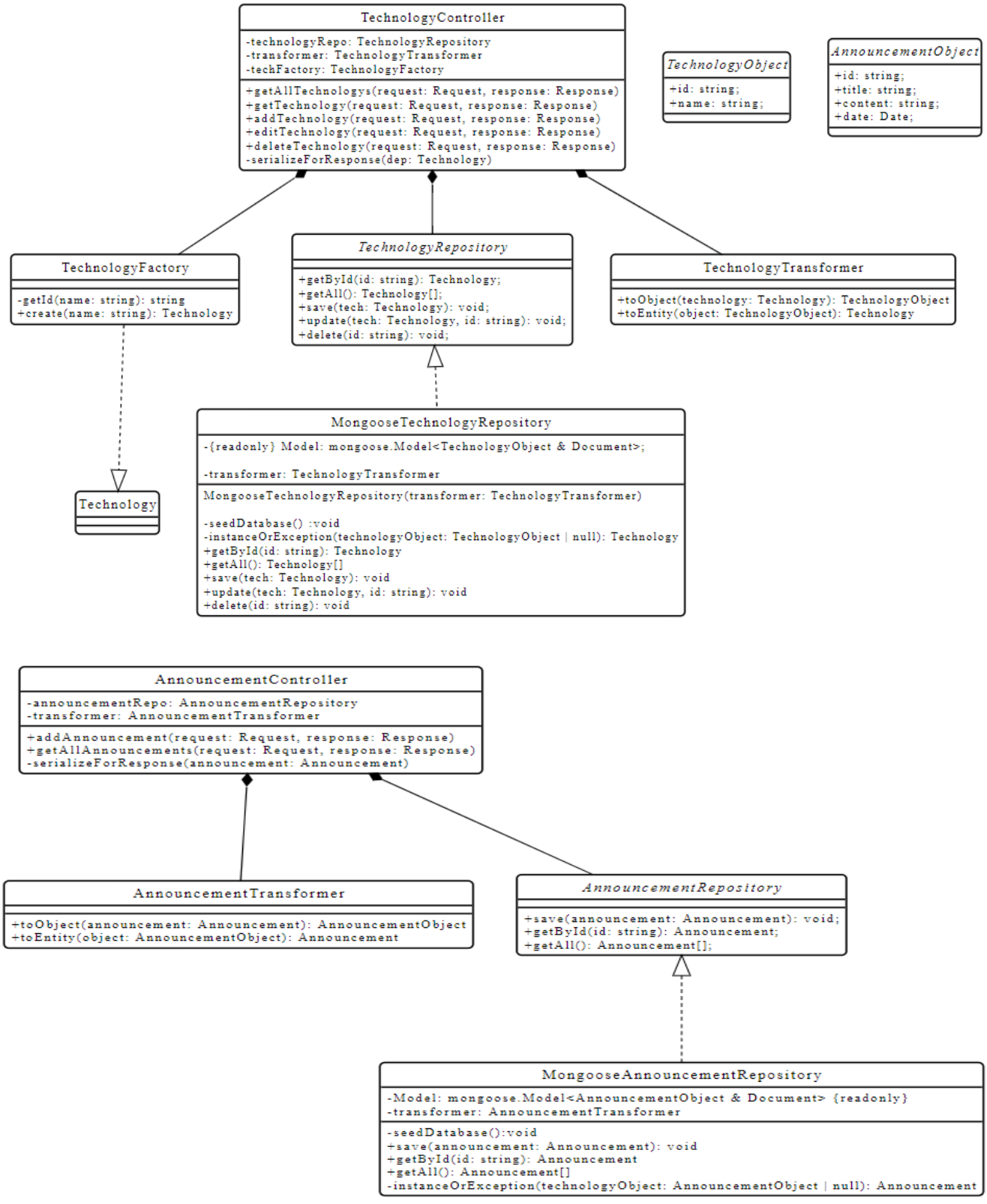


Figure 5: System Class Diagram part 3

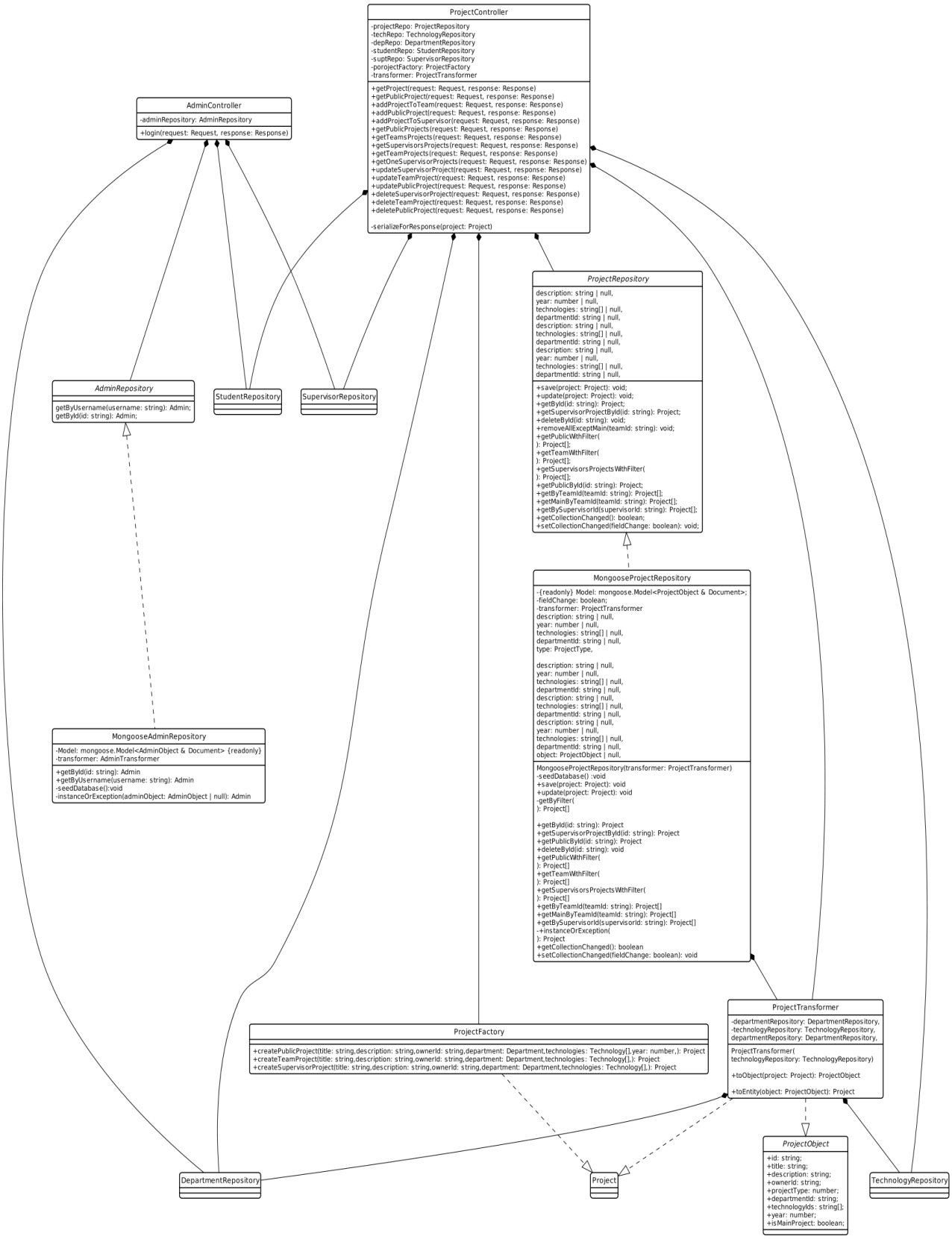


Figure 5: System Class Diagram part 4

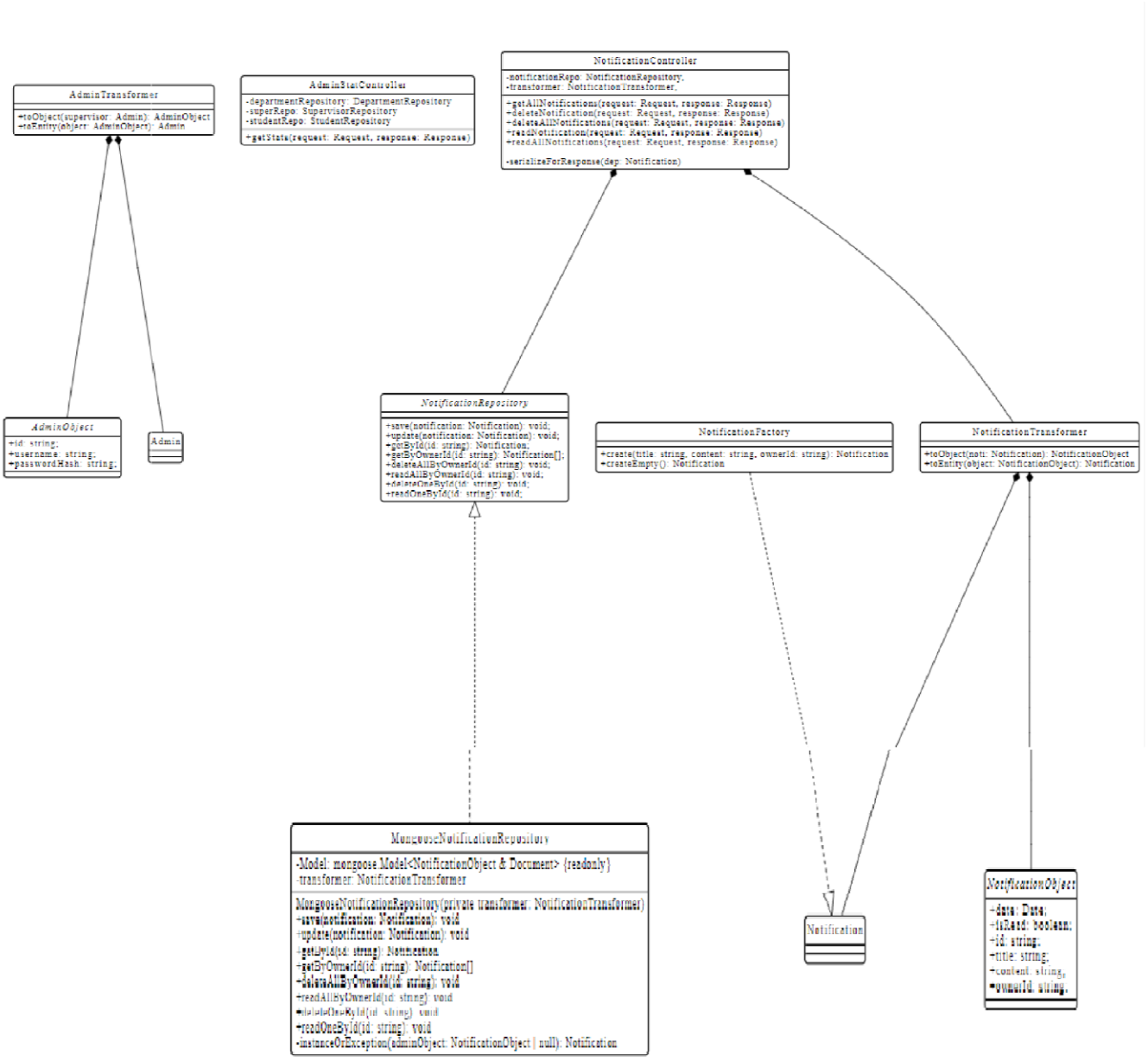


Figure 5: System Class Diagram part 5

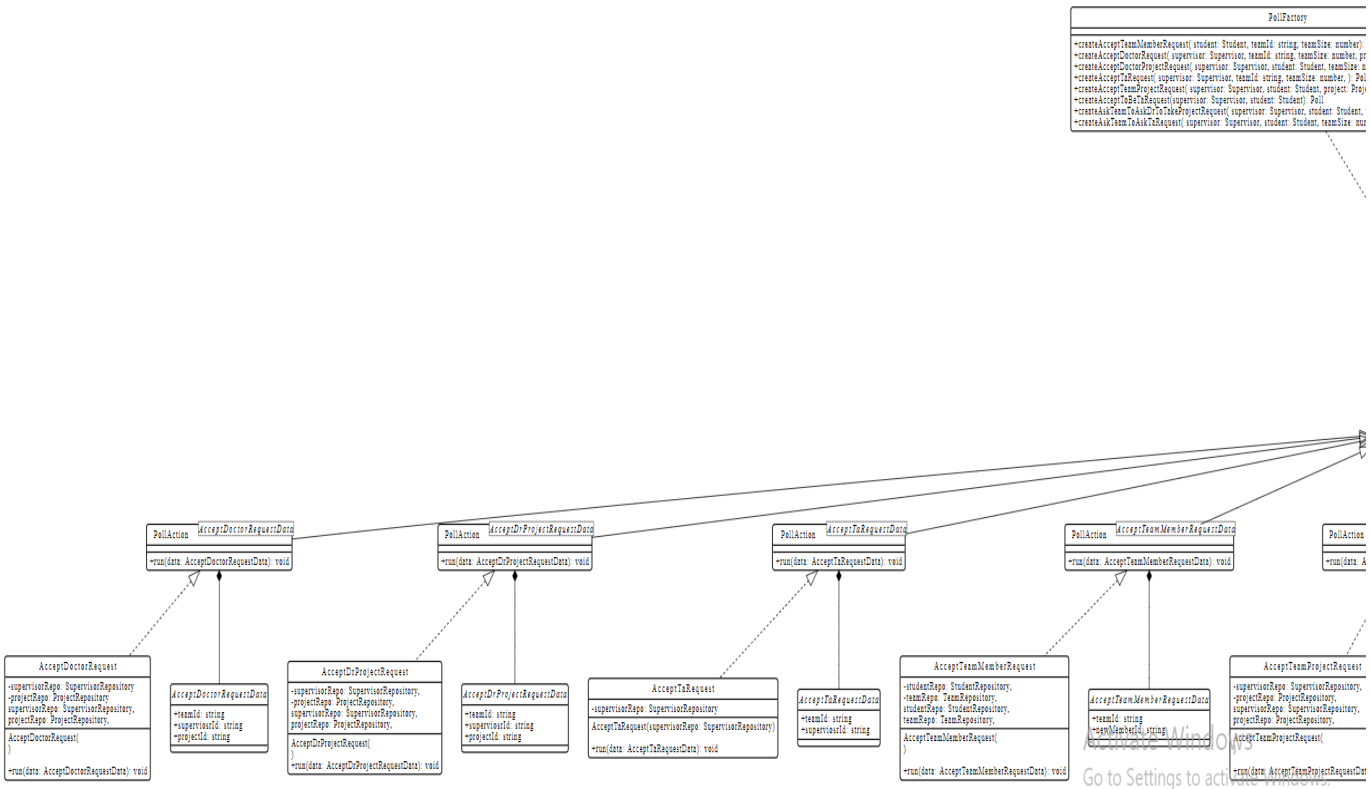


Figure 5: System Class Diagram part 6

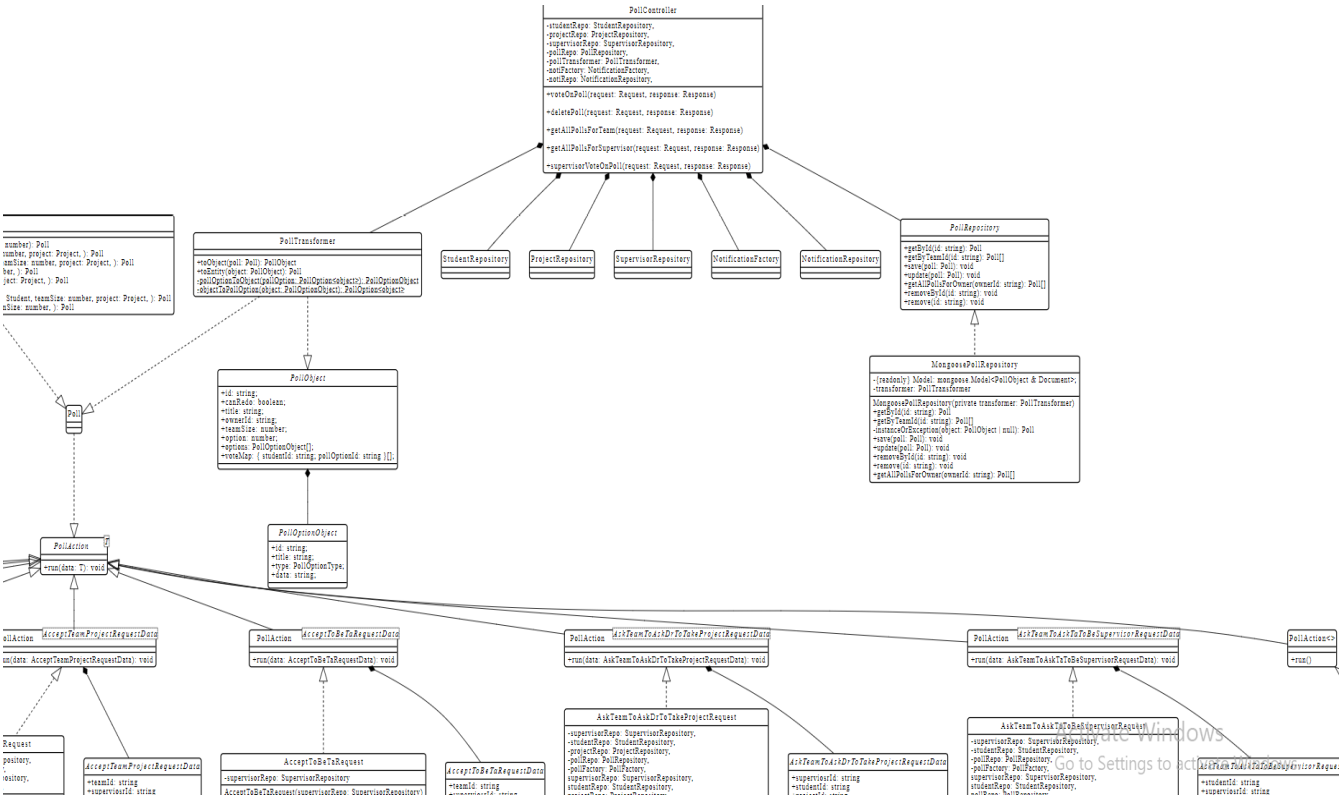


Figure 5: System Class Diagram part 7

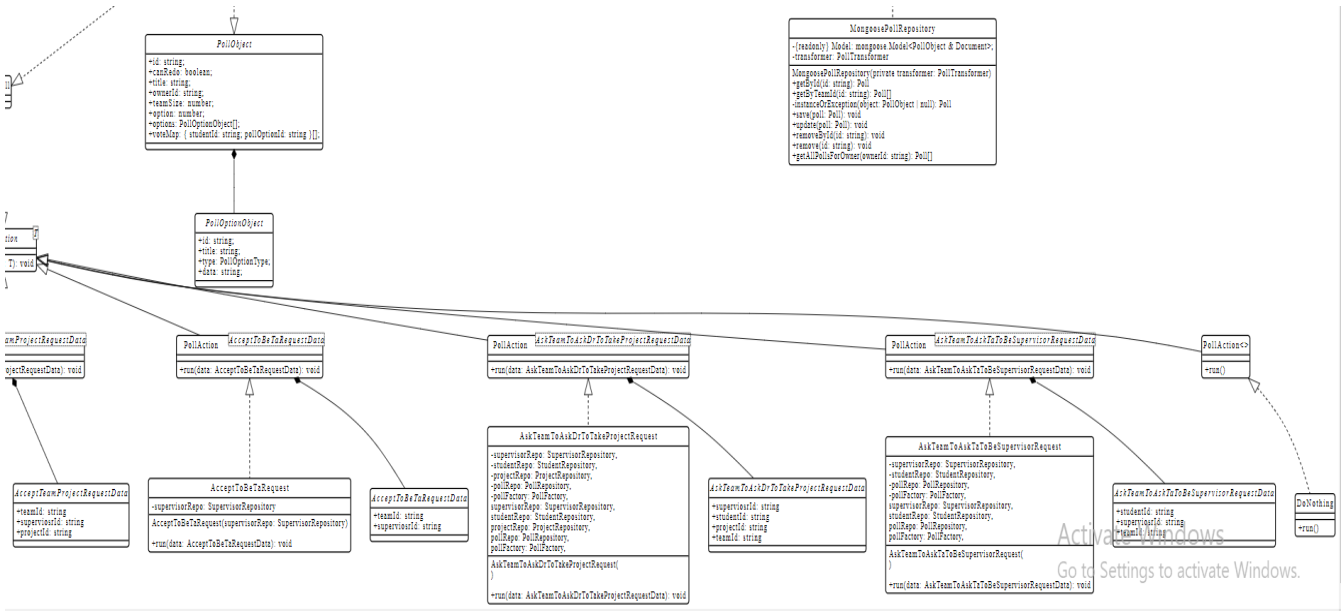


Figure 5: System Class Diagram part 8

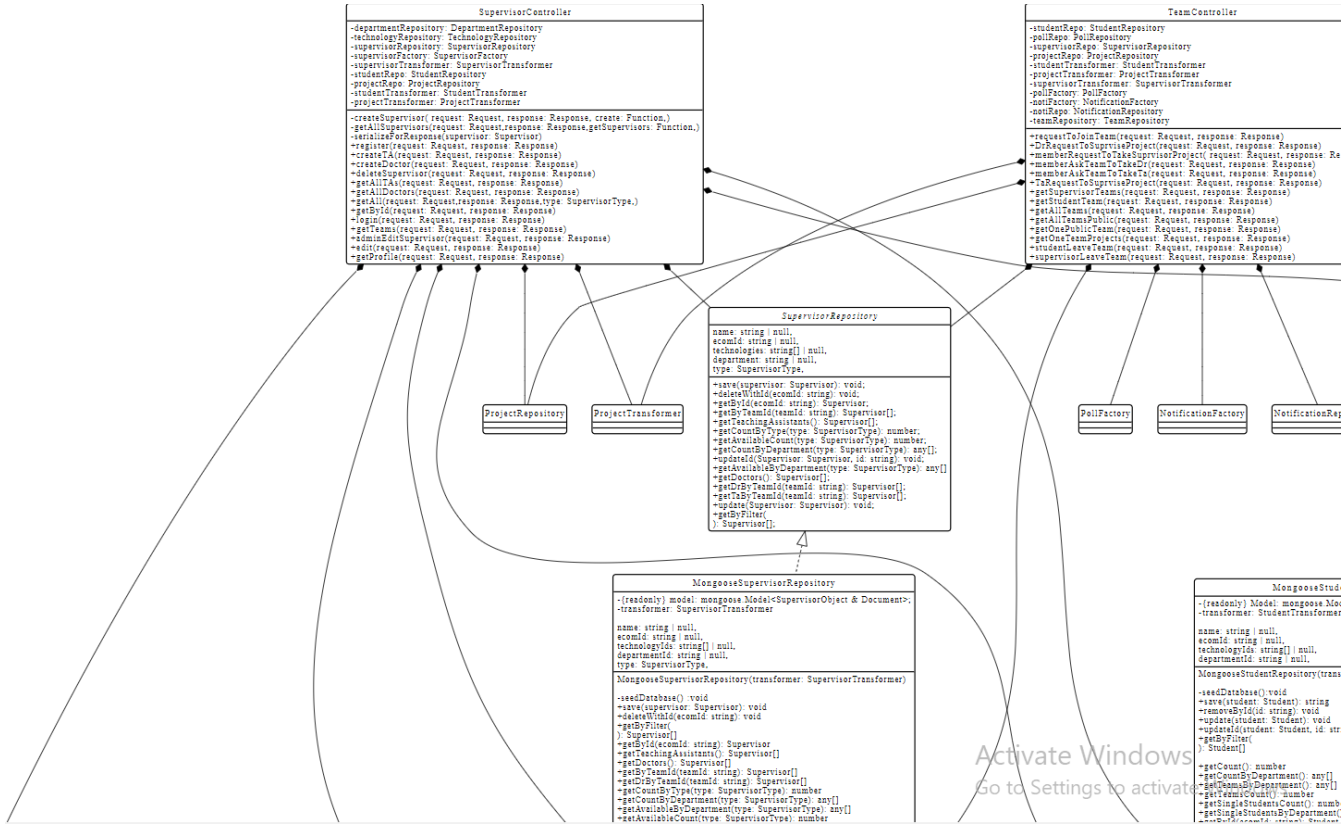


Figure 5: System Class Diagram part 9

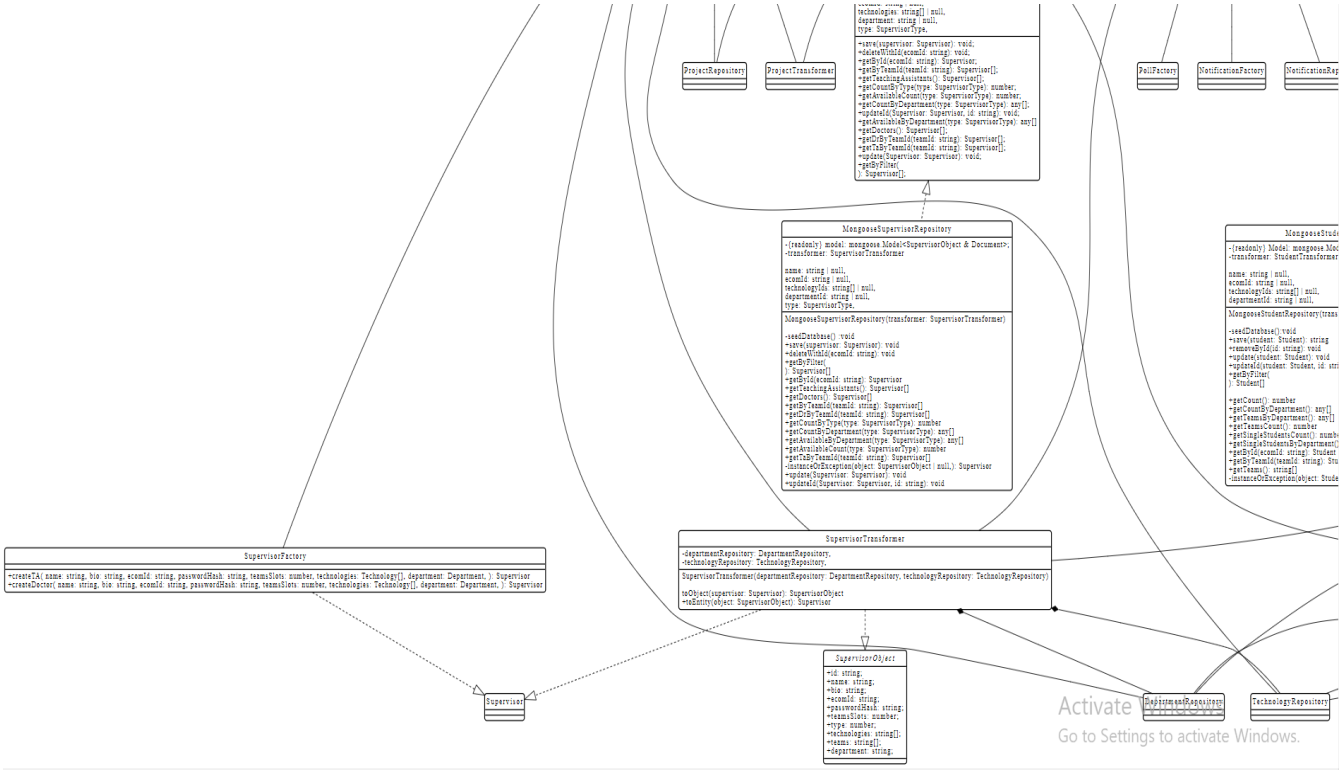


Figure 5: System Class Diagram part 10

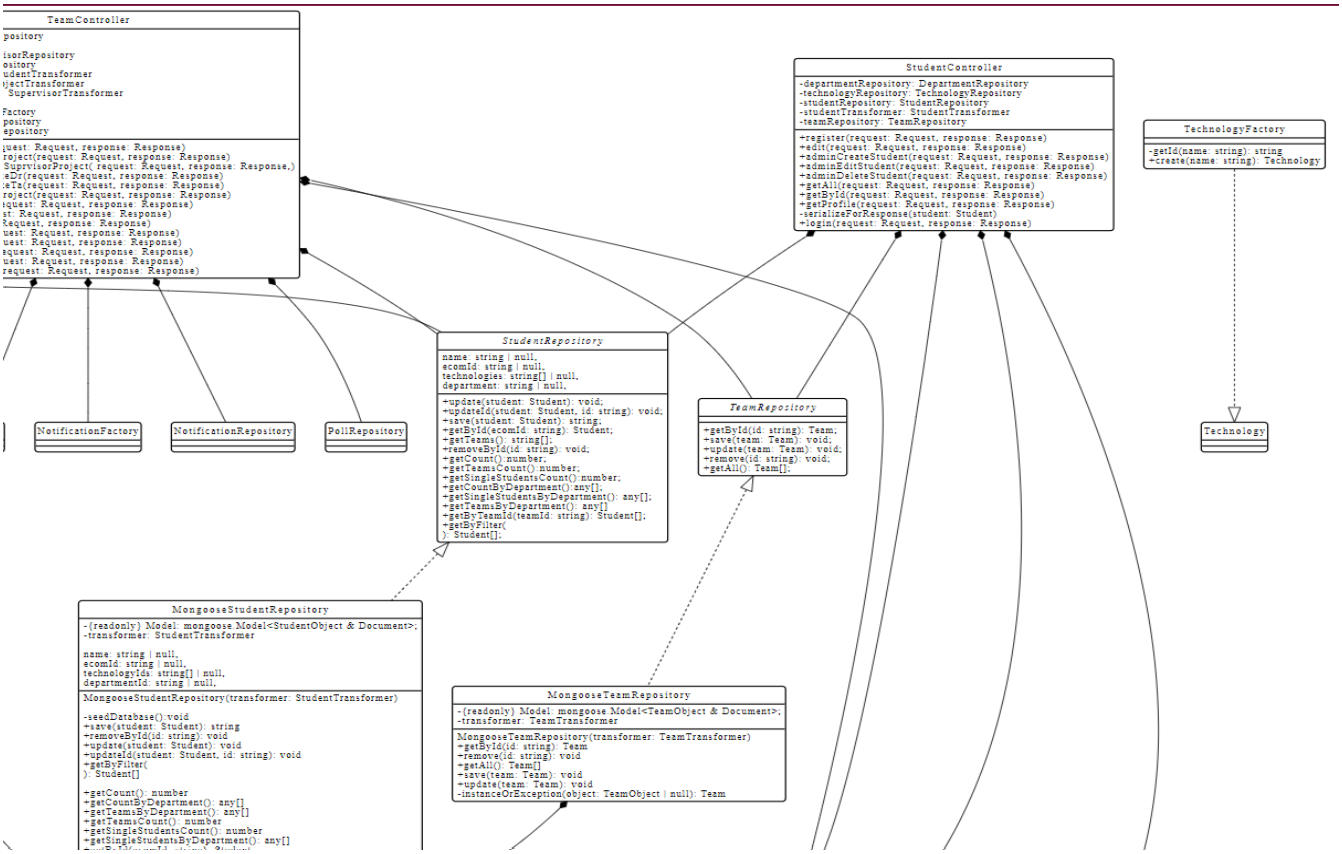


Figure 5: System Class Diagram_ part 11

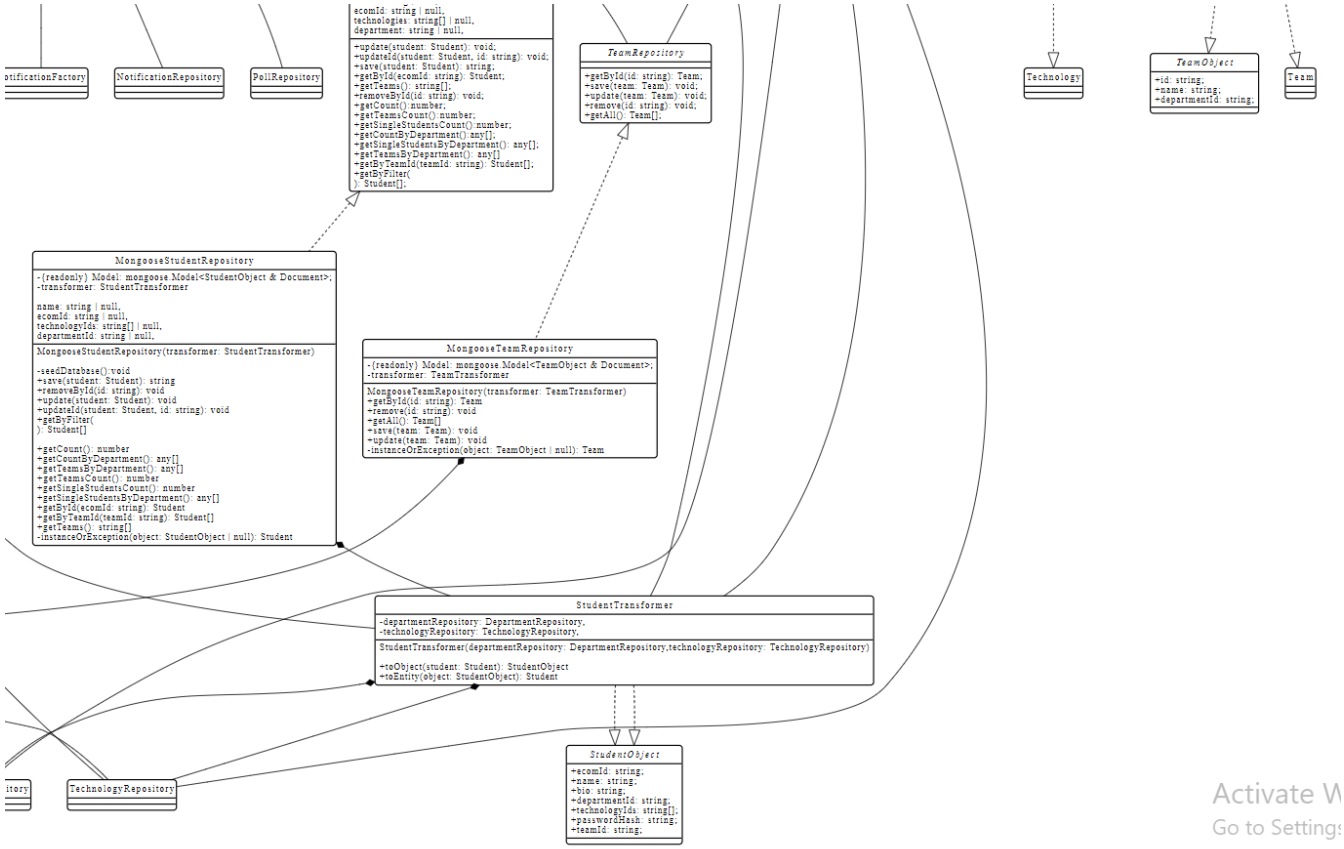


Figure 5: System Class Diagram part 12

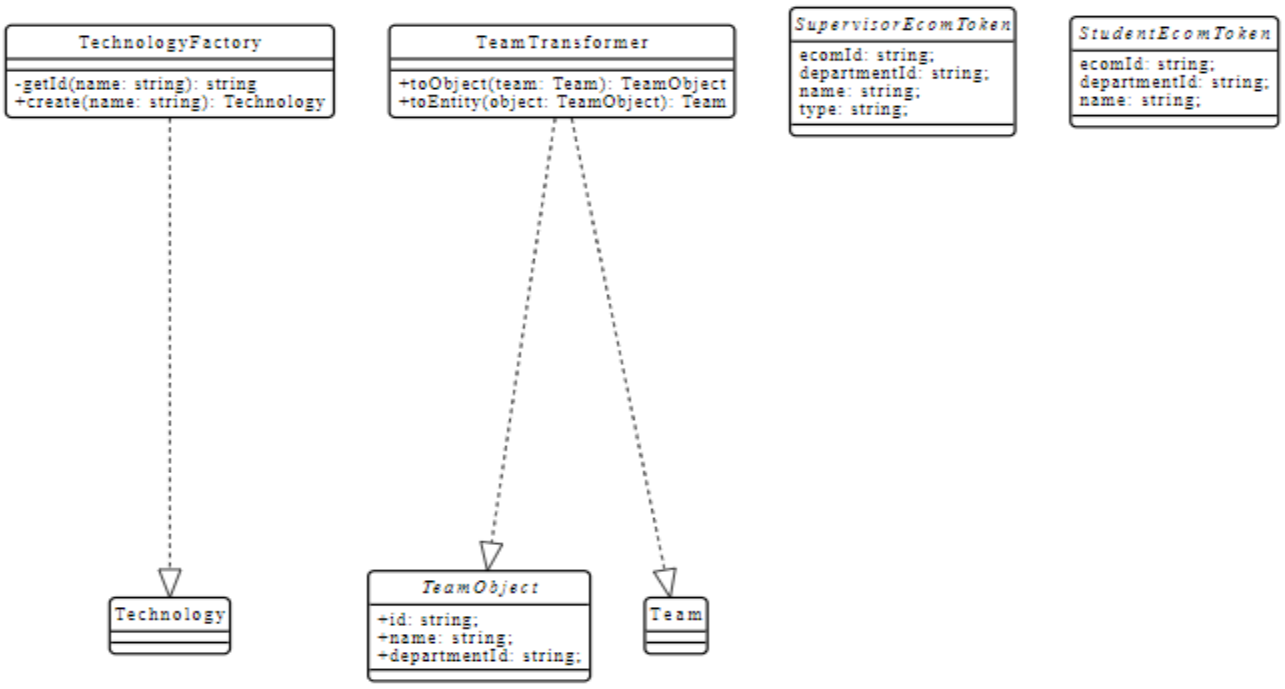


Figure 5: System Class Diagram part 13

4.3. Sequence Diagrams

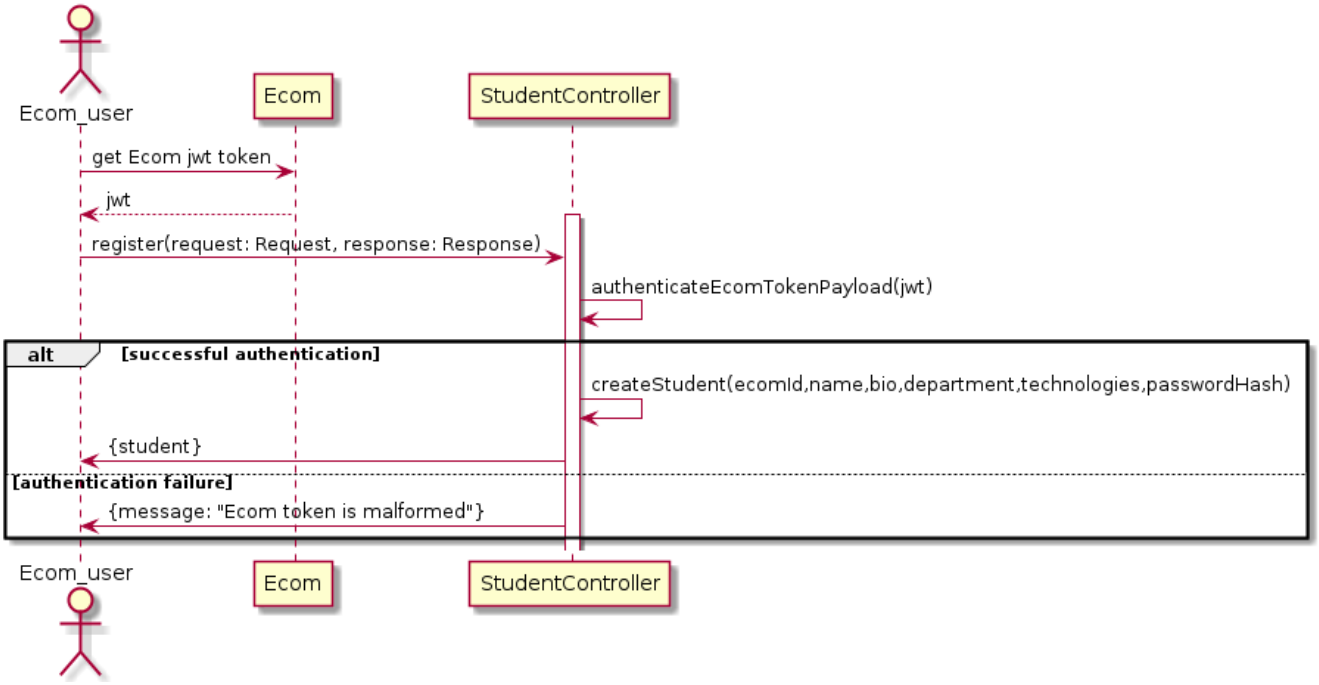


Figure 6: "Register" Sequence Diagram

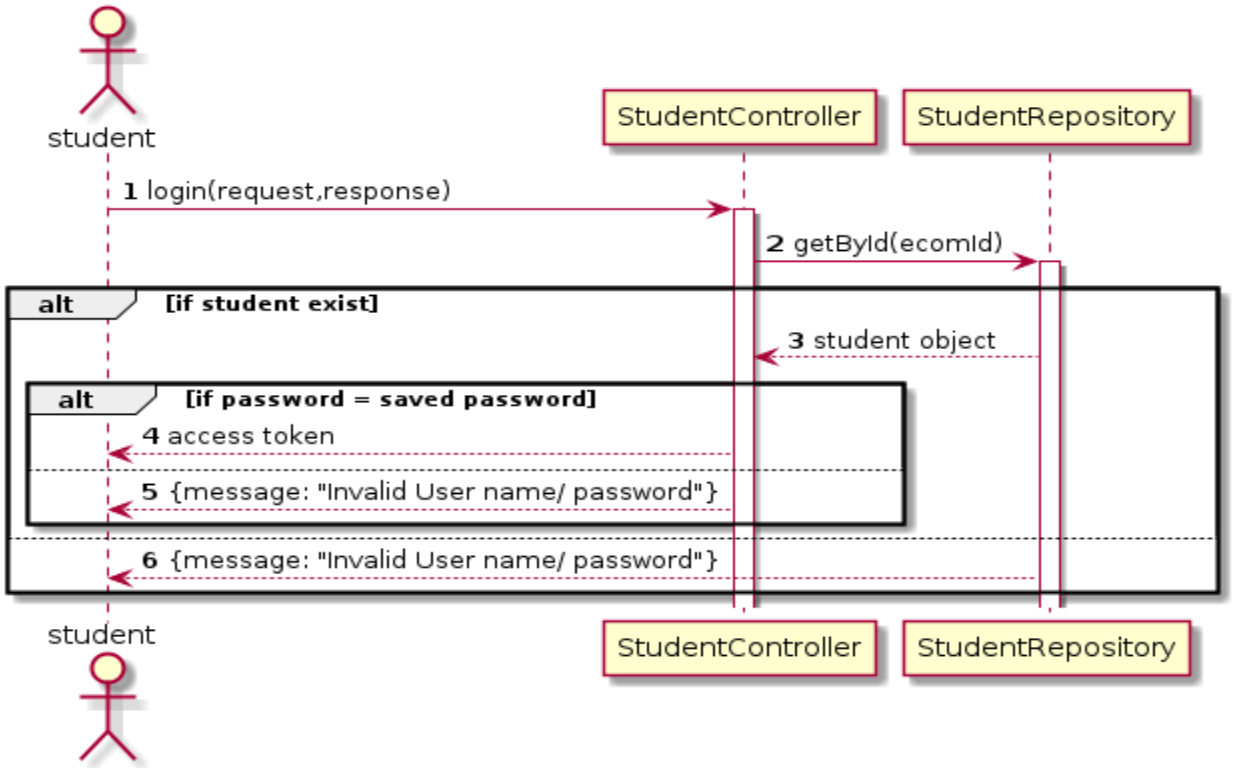


Figure 7: "Student Login" Sequence Diagram

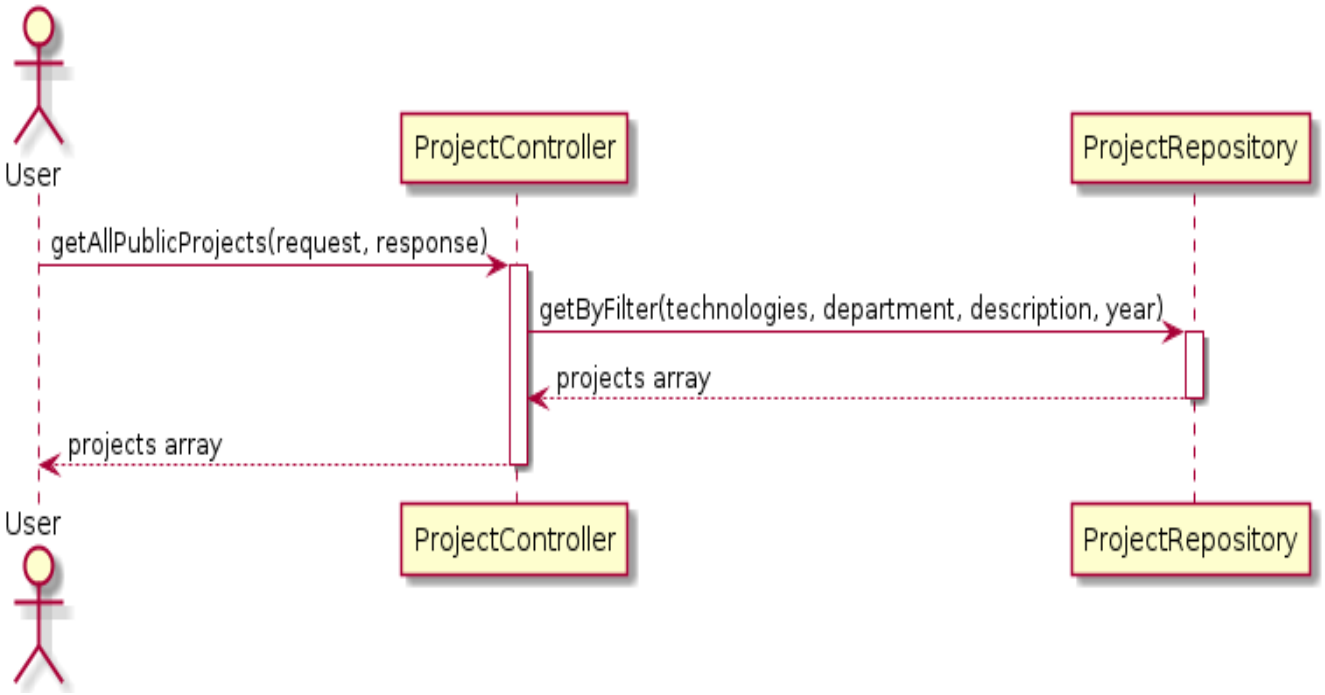


Figure 8: "Get All Public Projects" Sequence Diagram

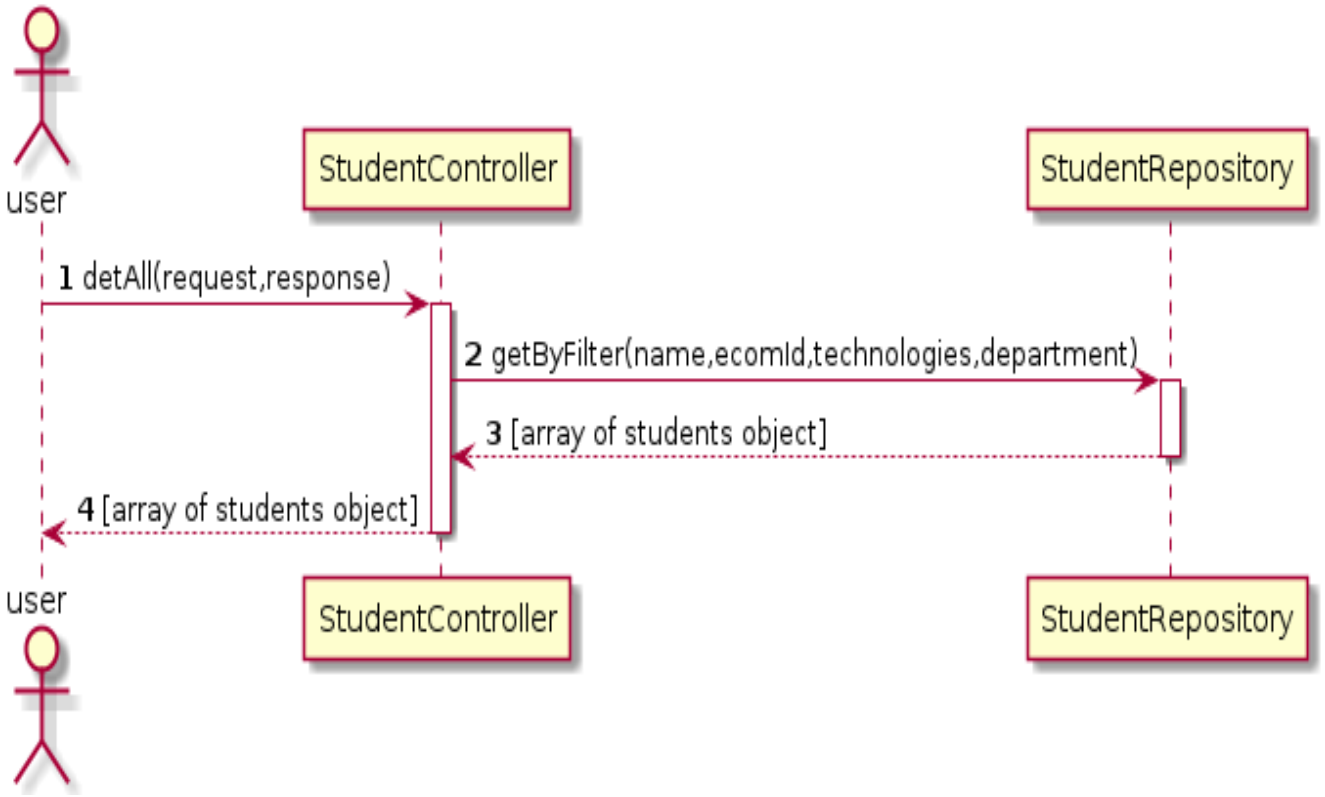


Figure 9: "Search and filter students" Sequence Diagram

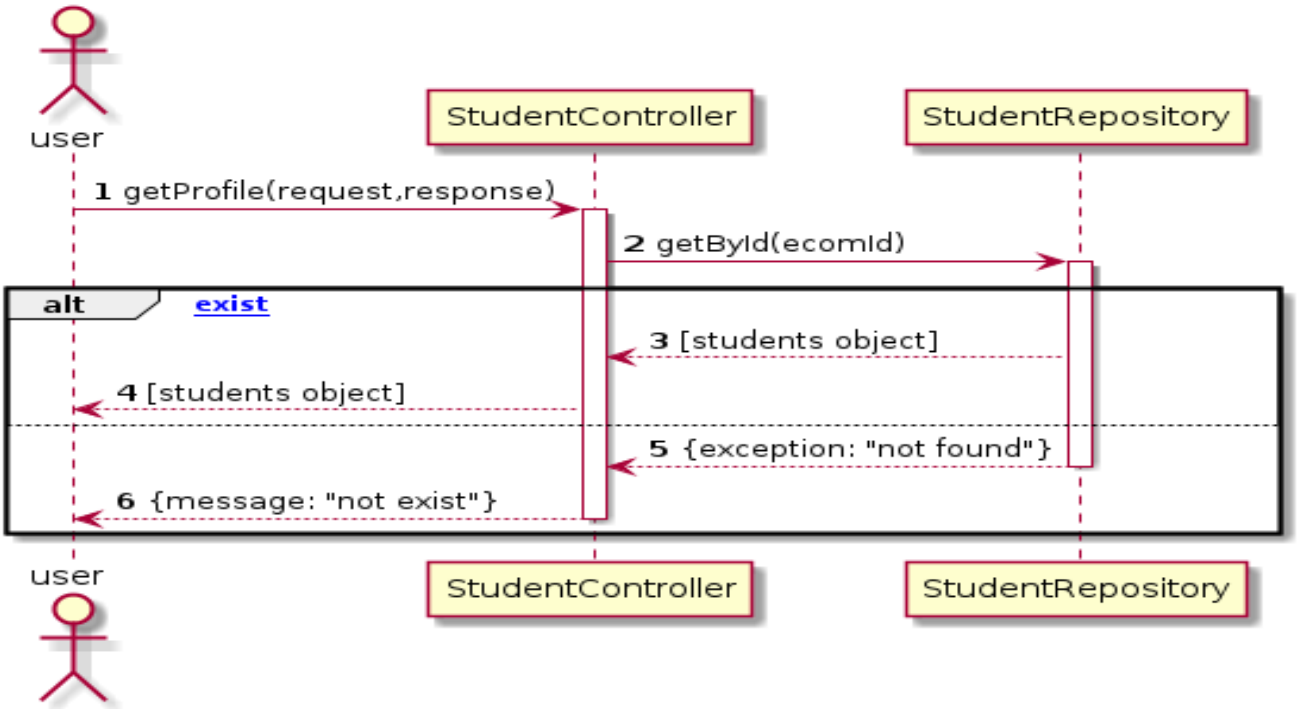


Figure 10: "Get Profile" Sequence Diagram

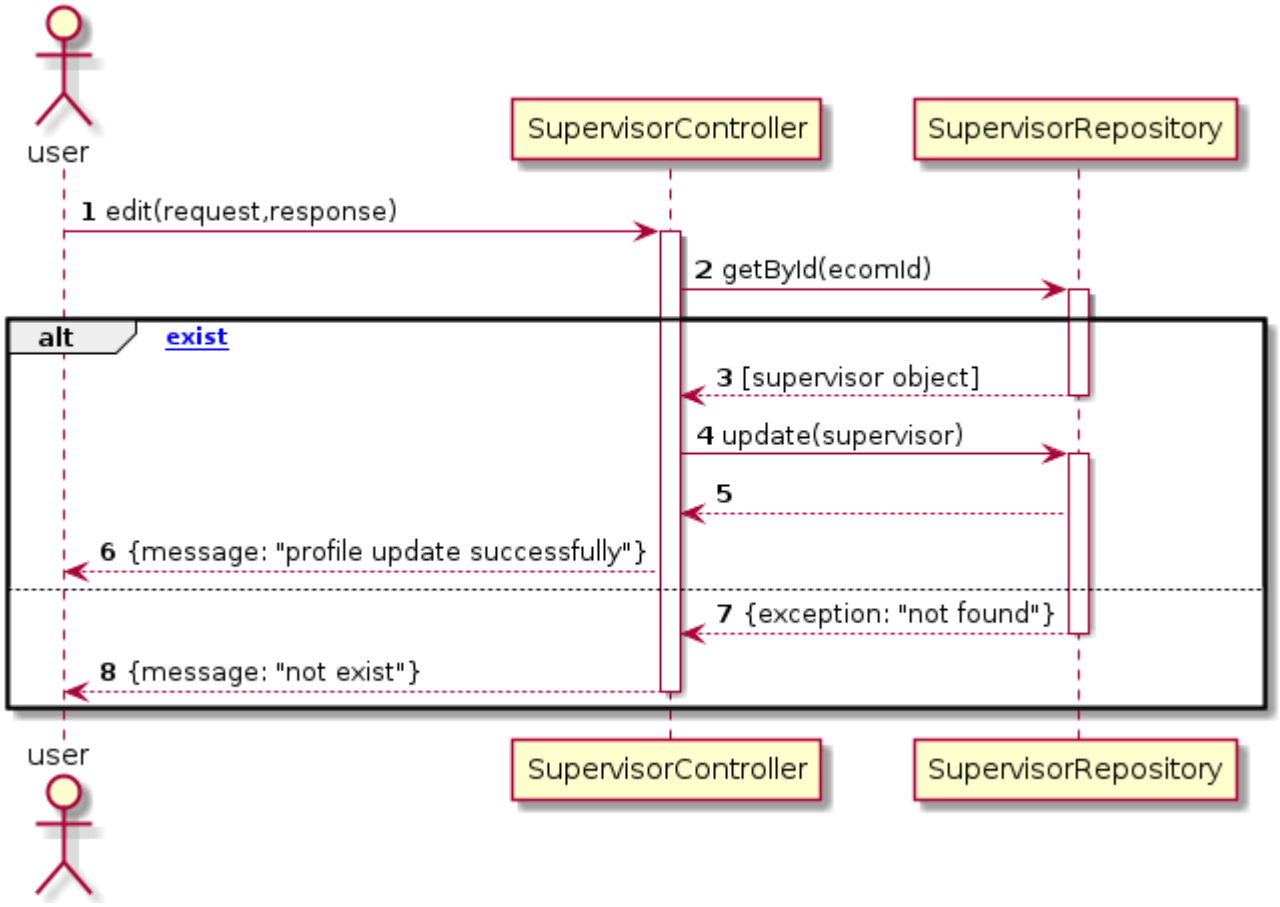


Figure 11: "Edit Supervisor Profile" Sequence Diagram

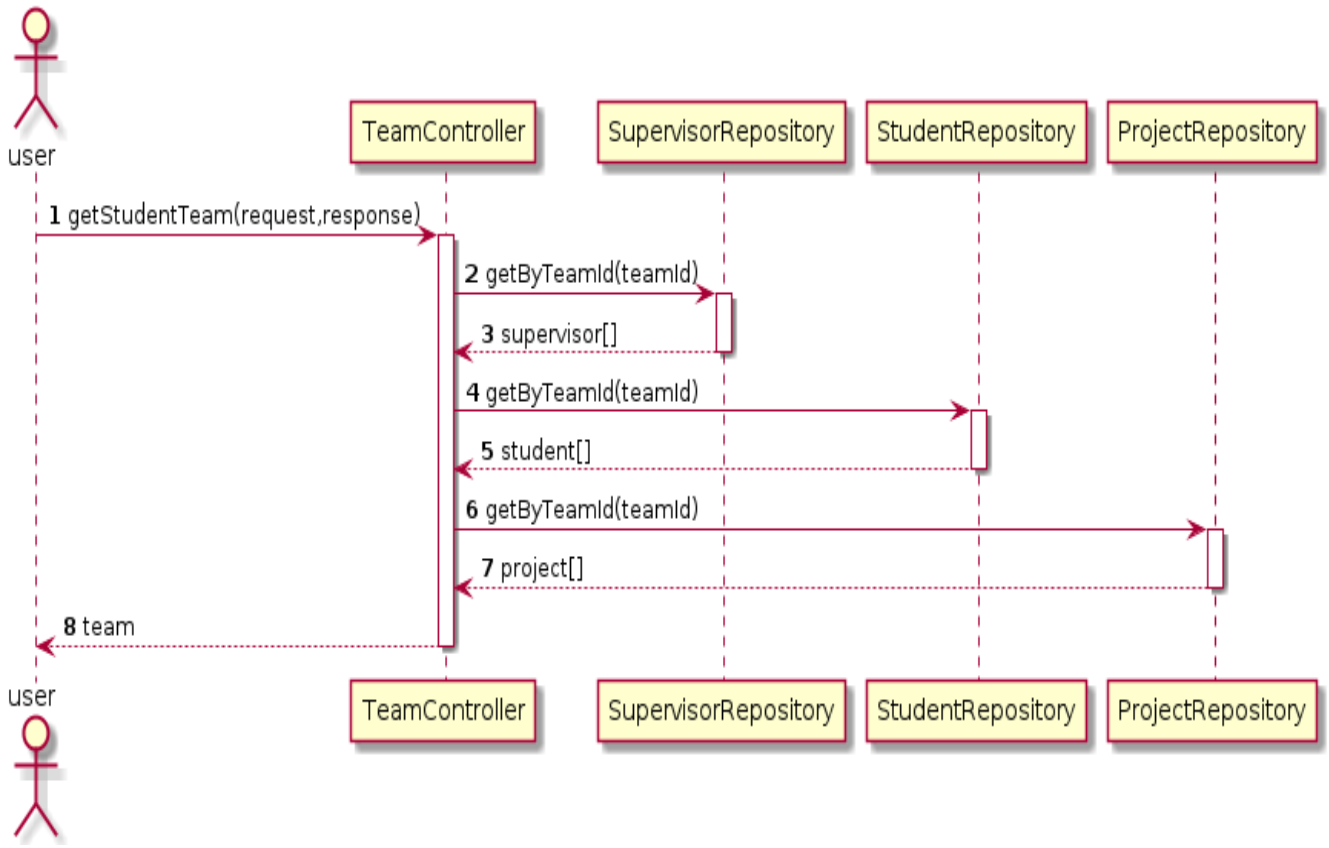


Figure 12: "Get Student Team Info" Sequence Diagram

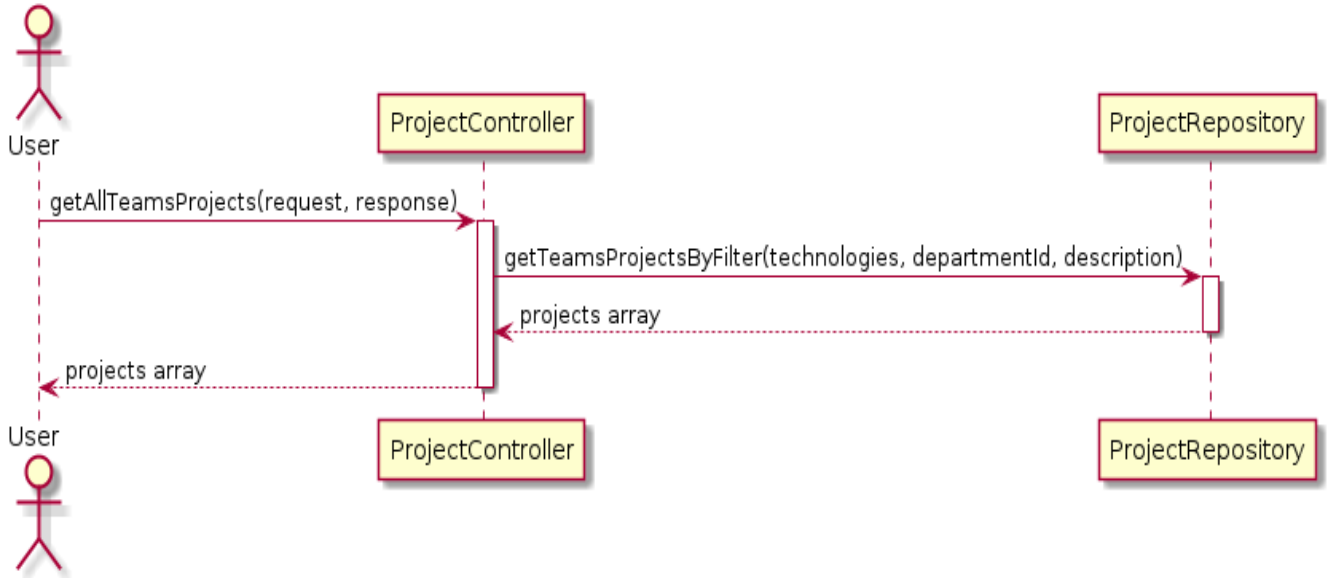


Figure 13: "Get All Teams Projects" Sequence Diagram

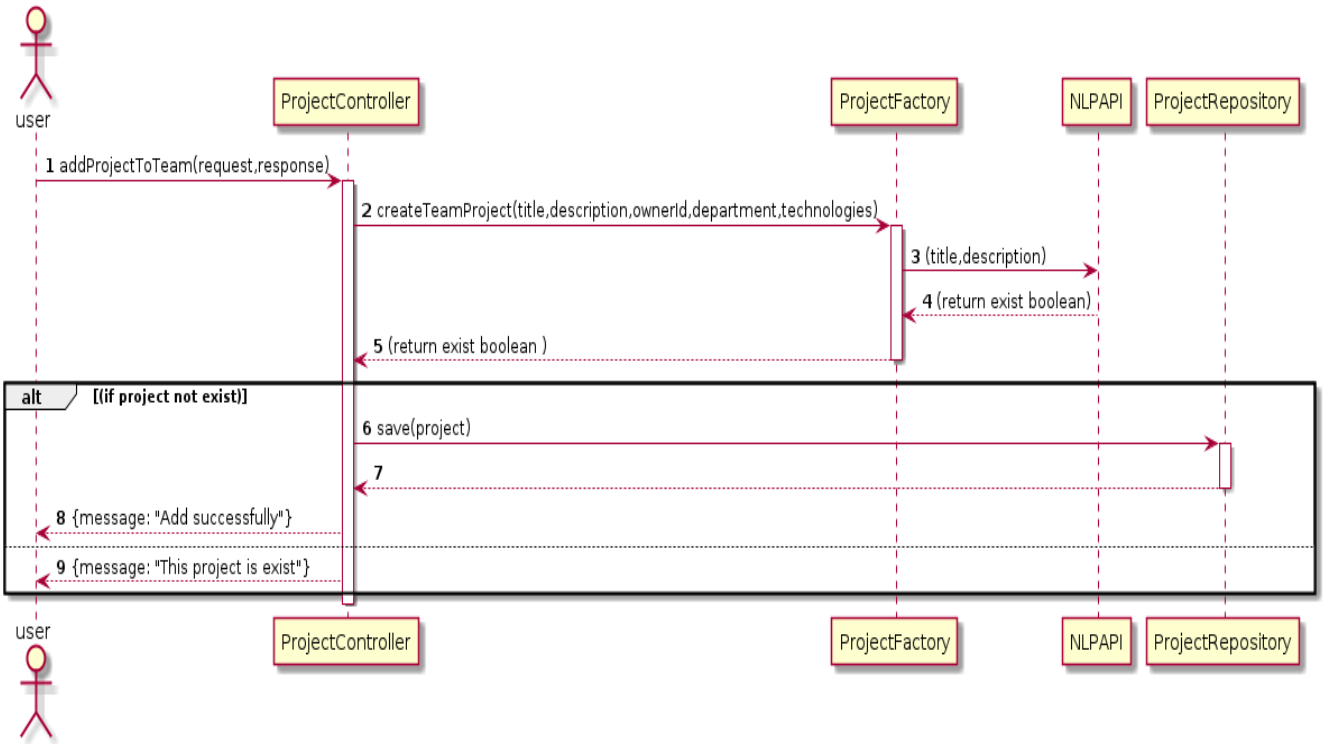


Figure 14: "Student Add Project" Sequence Diagram

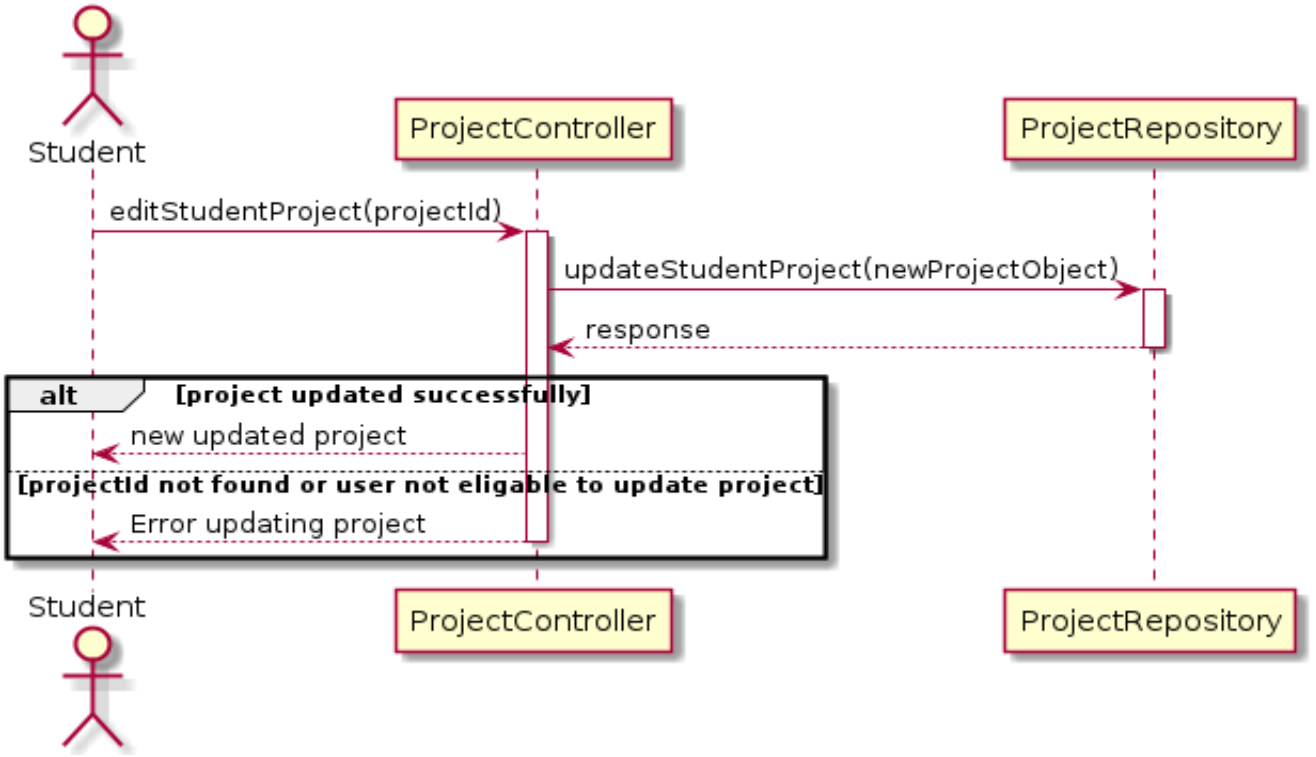


Figure 15: "Edit Project" Sequence Diagram

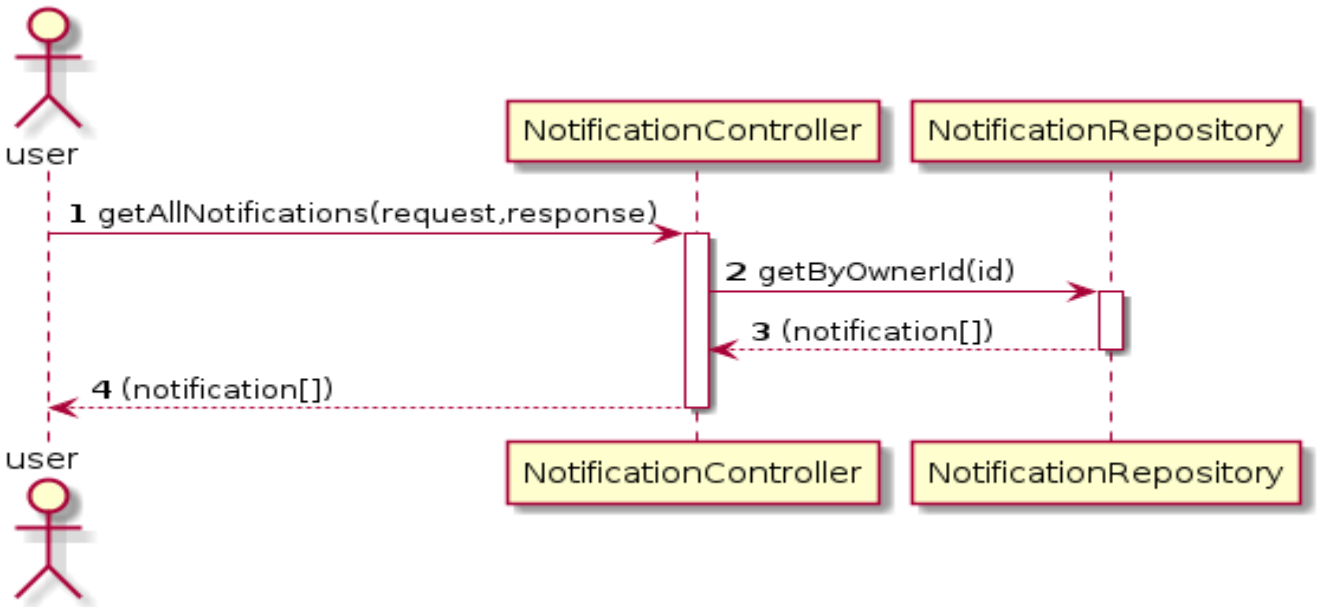


Figure 16: "Get Notifications" Sequence Diagram

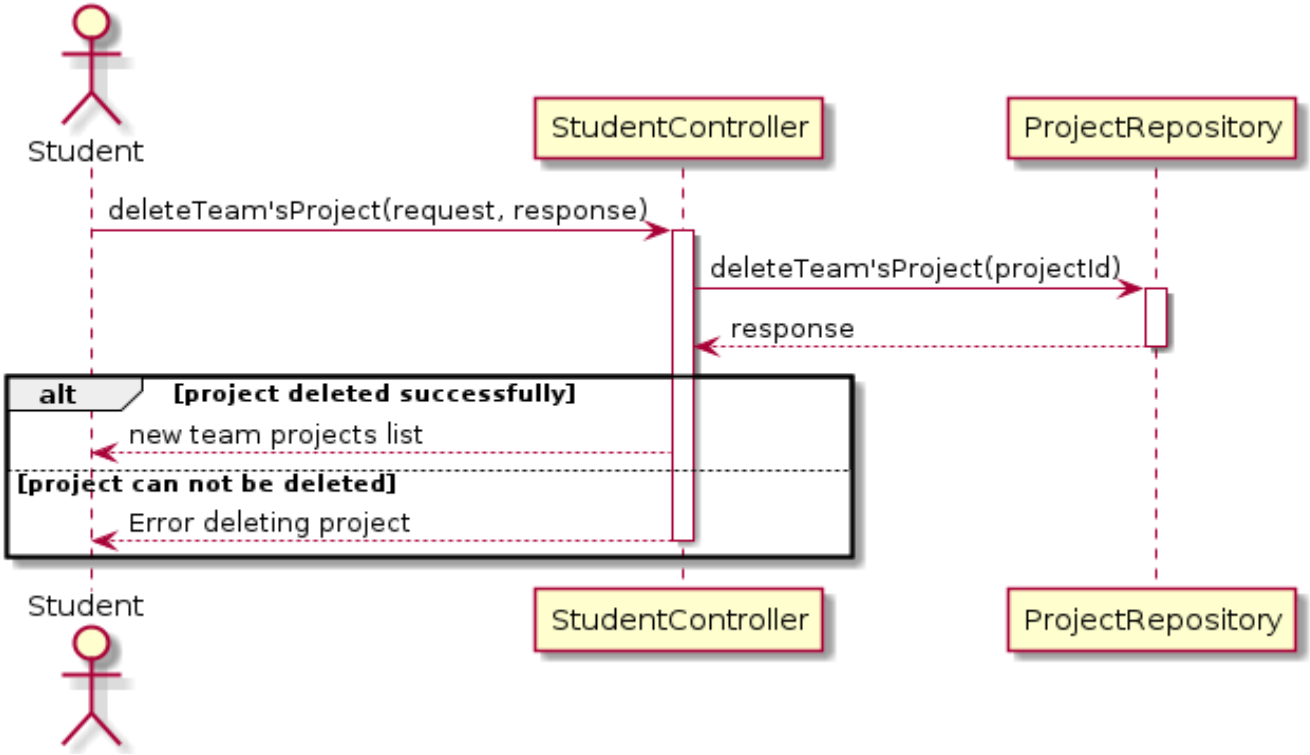


Figure 17: "Delete Project" Sequence Diagram

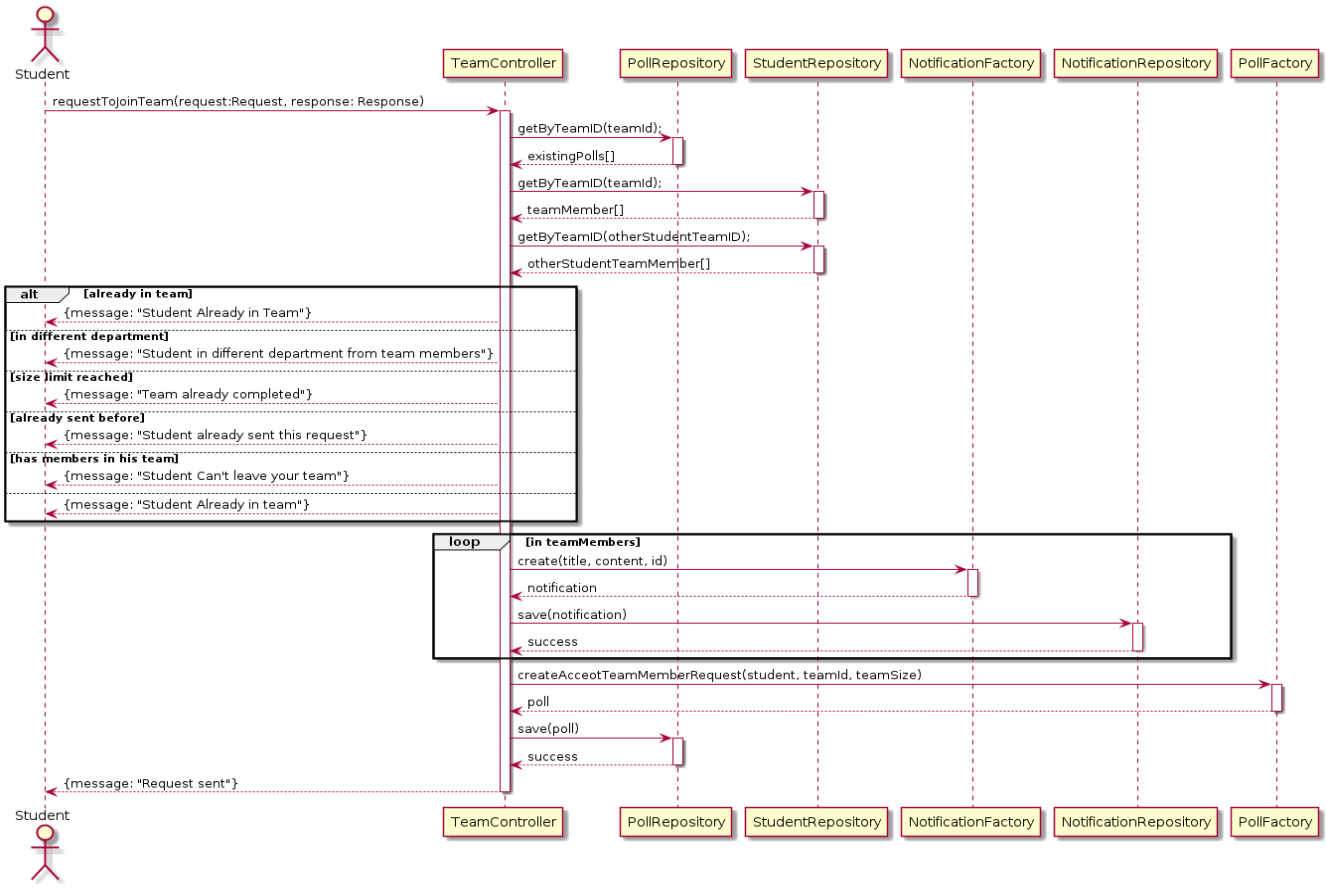


Figure 18: "request to join another student's team" Sequence Diagram

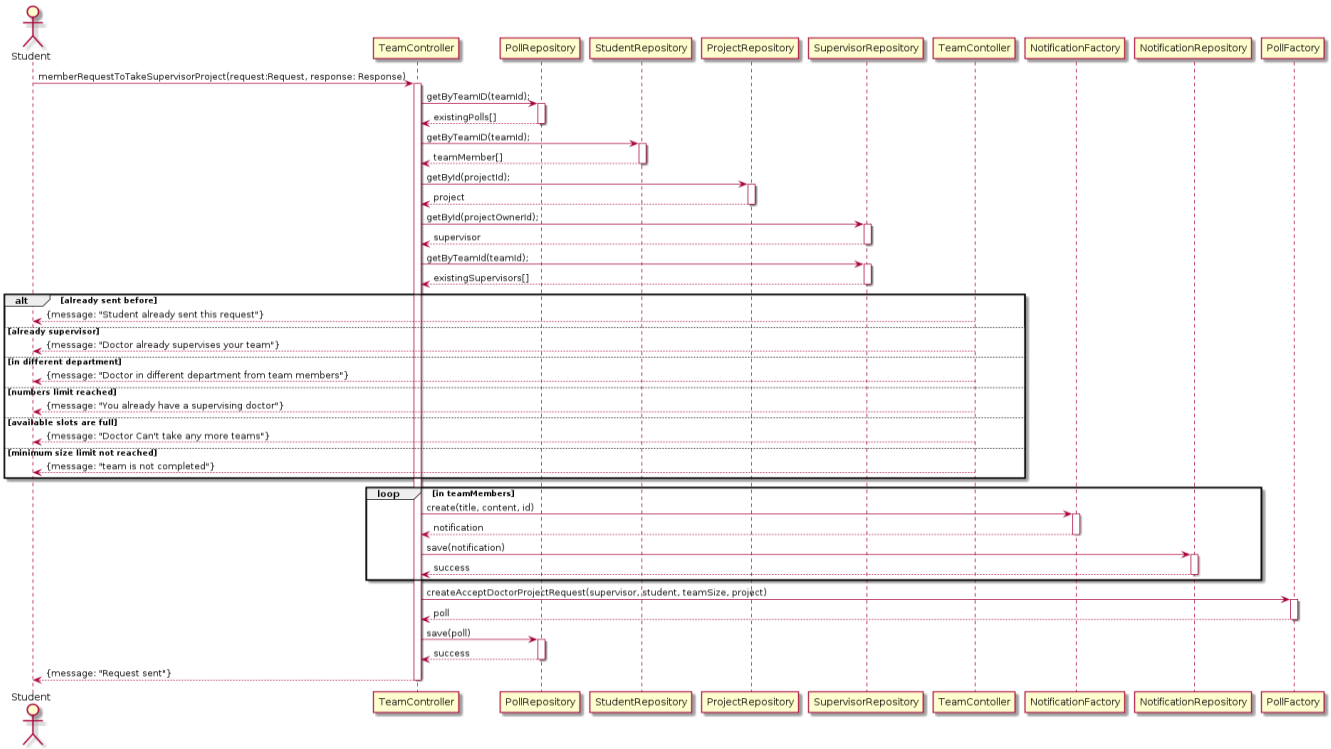


Figure 19: "Ask Team Members to Take Dr's Project" Sequence Diagram

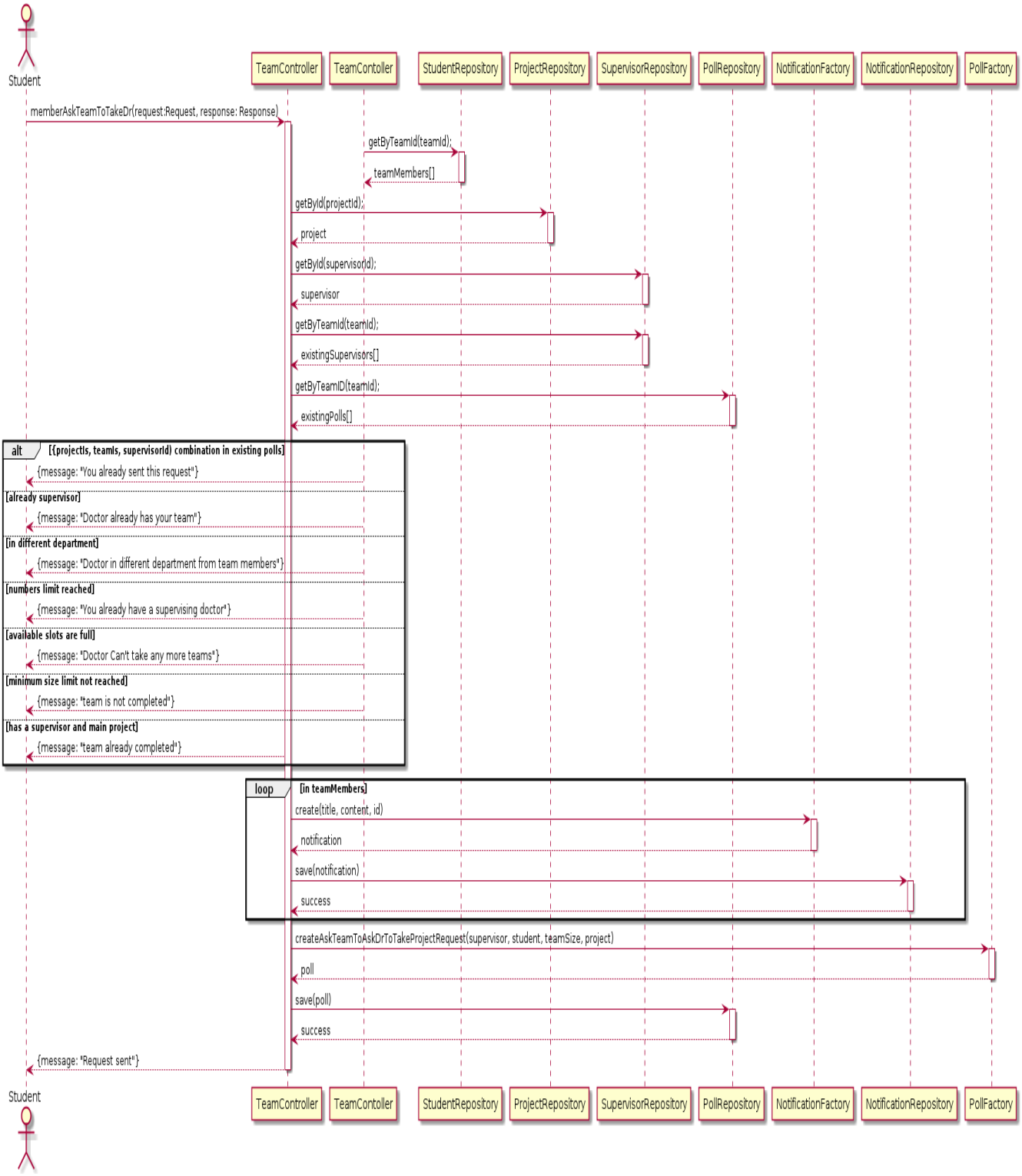


Figure 20: "Ask Team members to make supervisor take an idea" Sequence Diagram

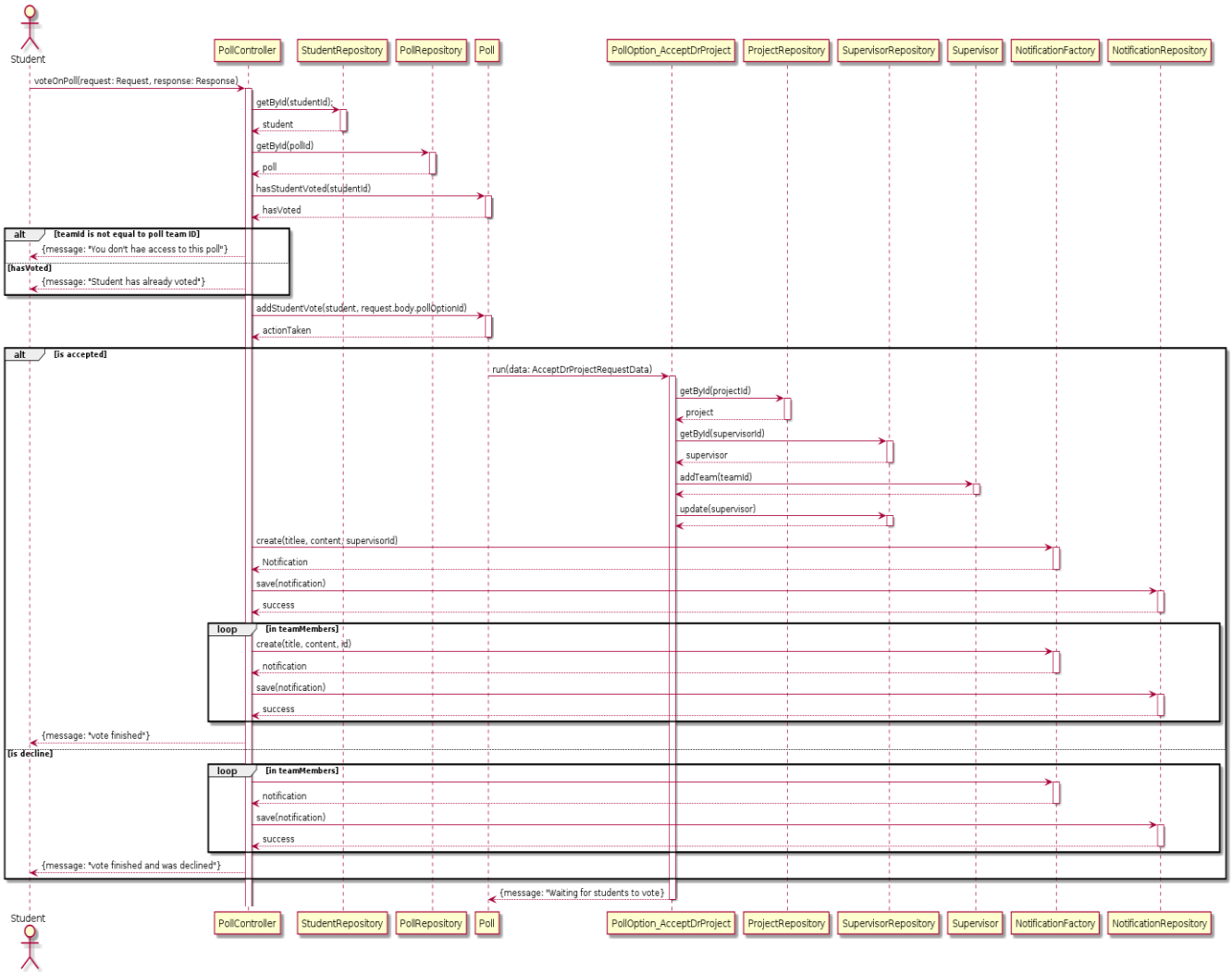


Figure 21: "Students vote to take Dr's idea" Sequence Diagram

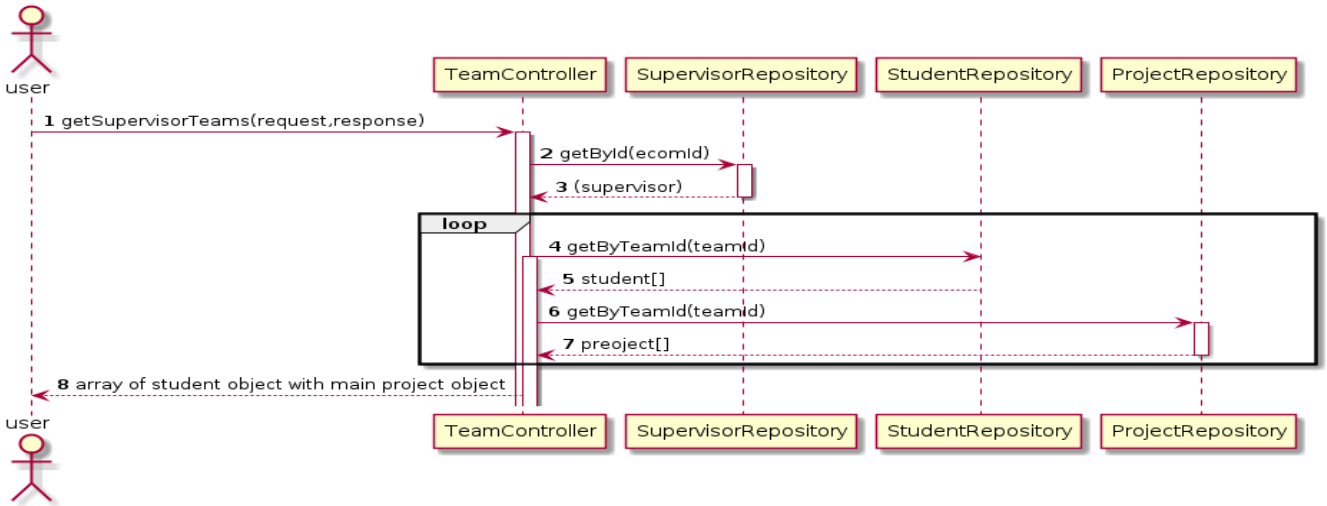


Figure 22: "Get All Projects Supervised by a Supervisor" Sequence Diagram

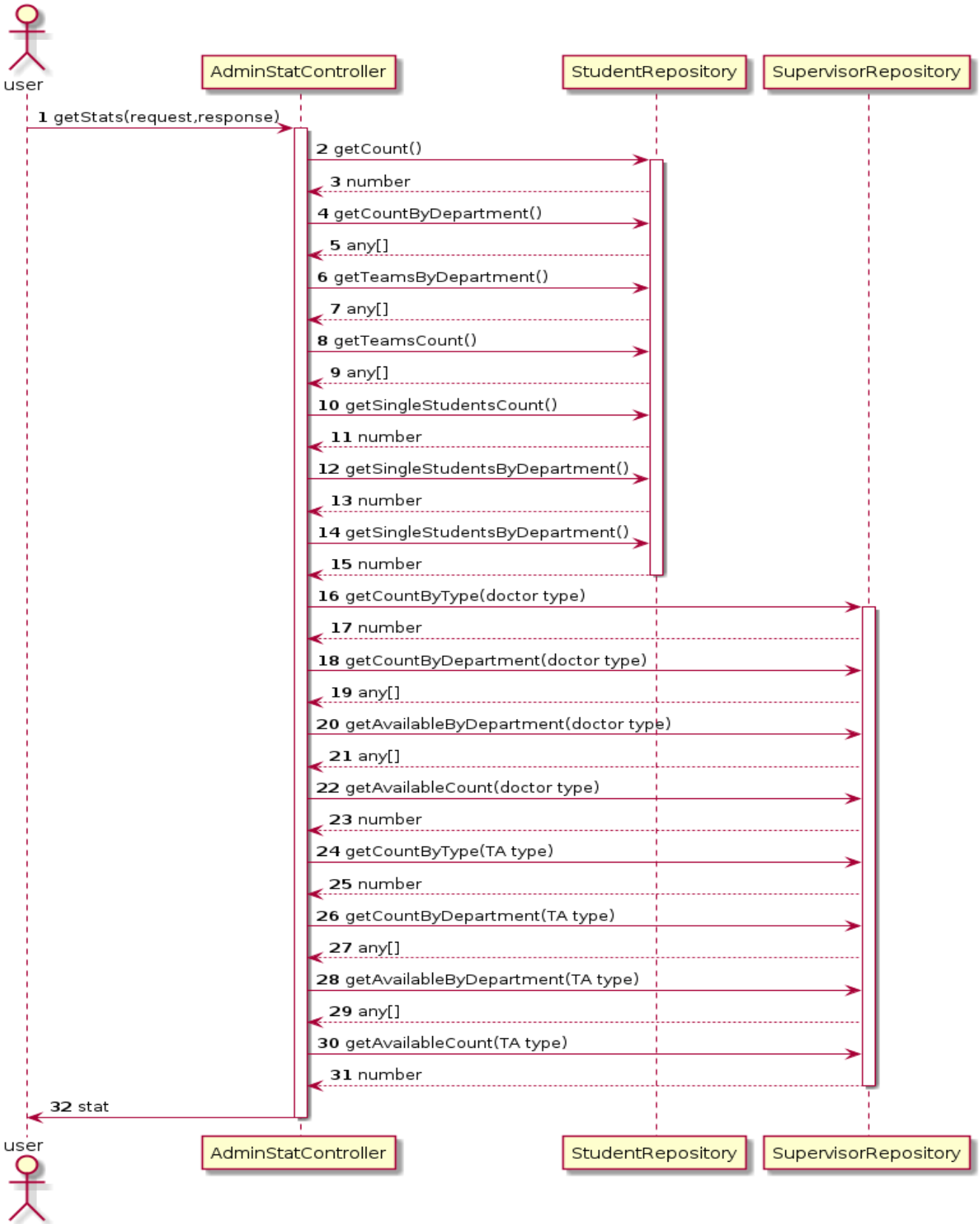


Figure 23: "Get Admin Statistics" Sequence Diagram

4.4. Project ERD

In MongoDB, data is stored as documents. These documents are stored in MongoDB in JSON format. JSON documents support embedded fields, so related data and lists of data can be stored with the document instead of an external table.

Since MongoDB is a document database and does not have an ERD, we made a conceptual ERD of what the database is designed like.

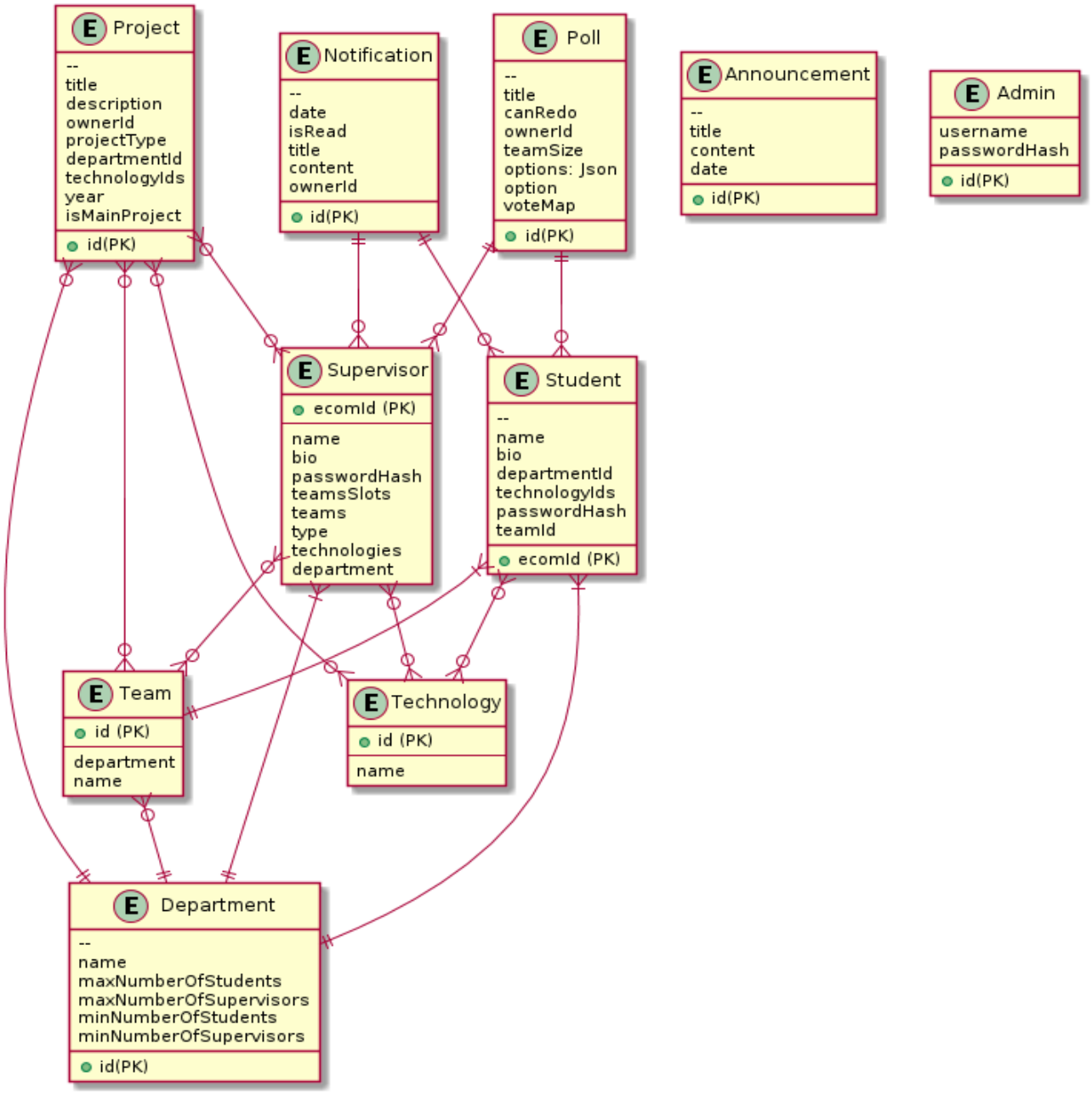


Figure 24: Project ERD

4.5. System GUI Design

In this section, we can be introduced to some pages that the stakeholders view and interact with:

Login pages:

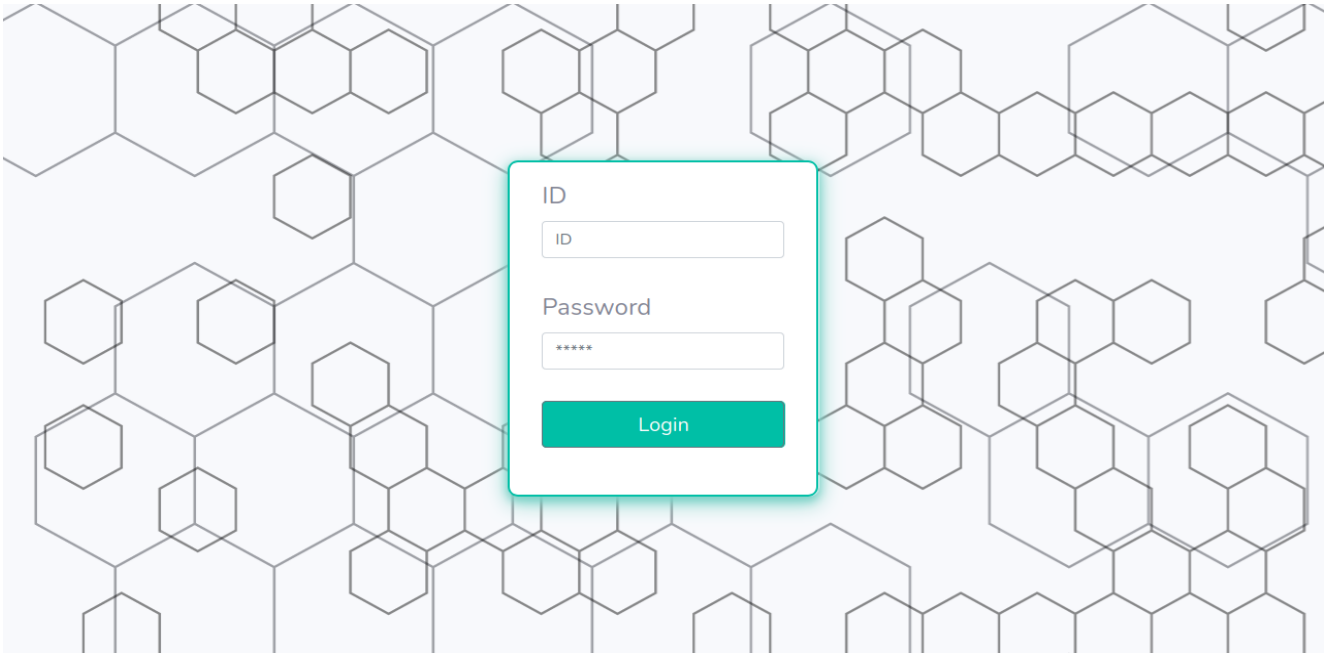


Figure 25: Admin Login Page

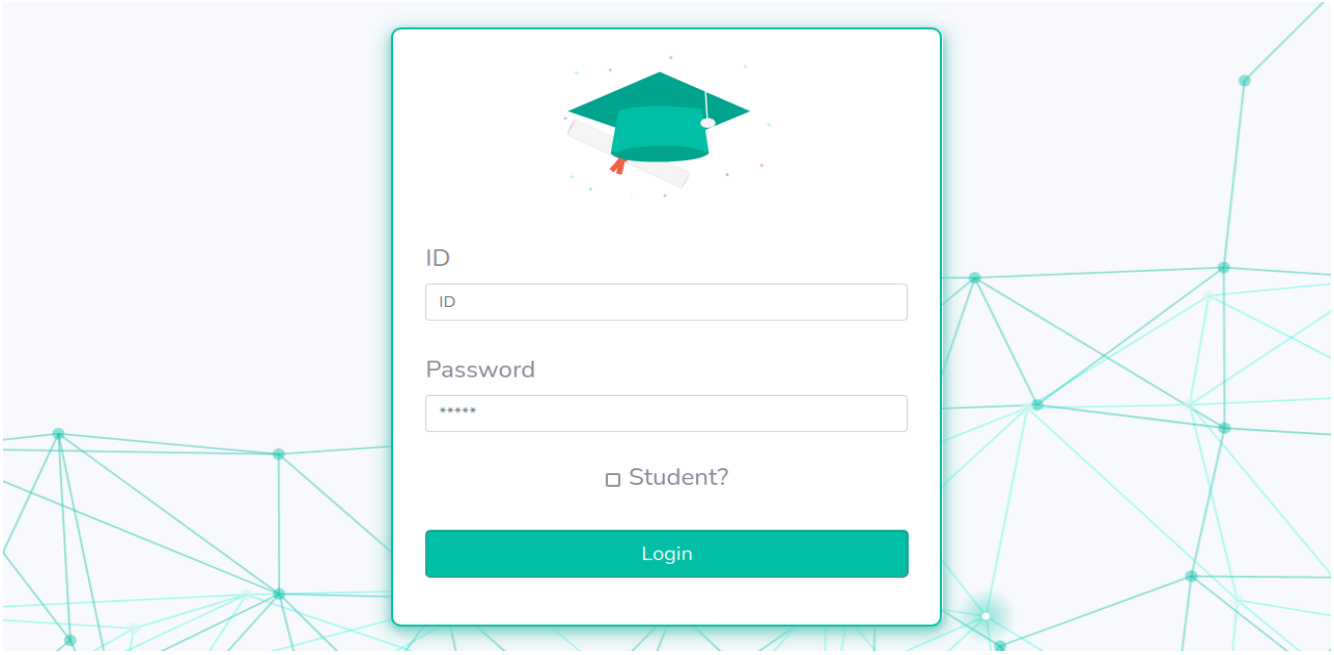


Figure 26: Students & Supervisors Login Page

Admin Pages:

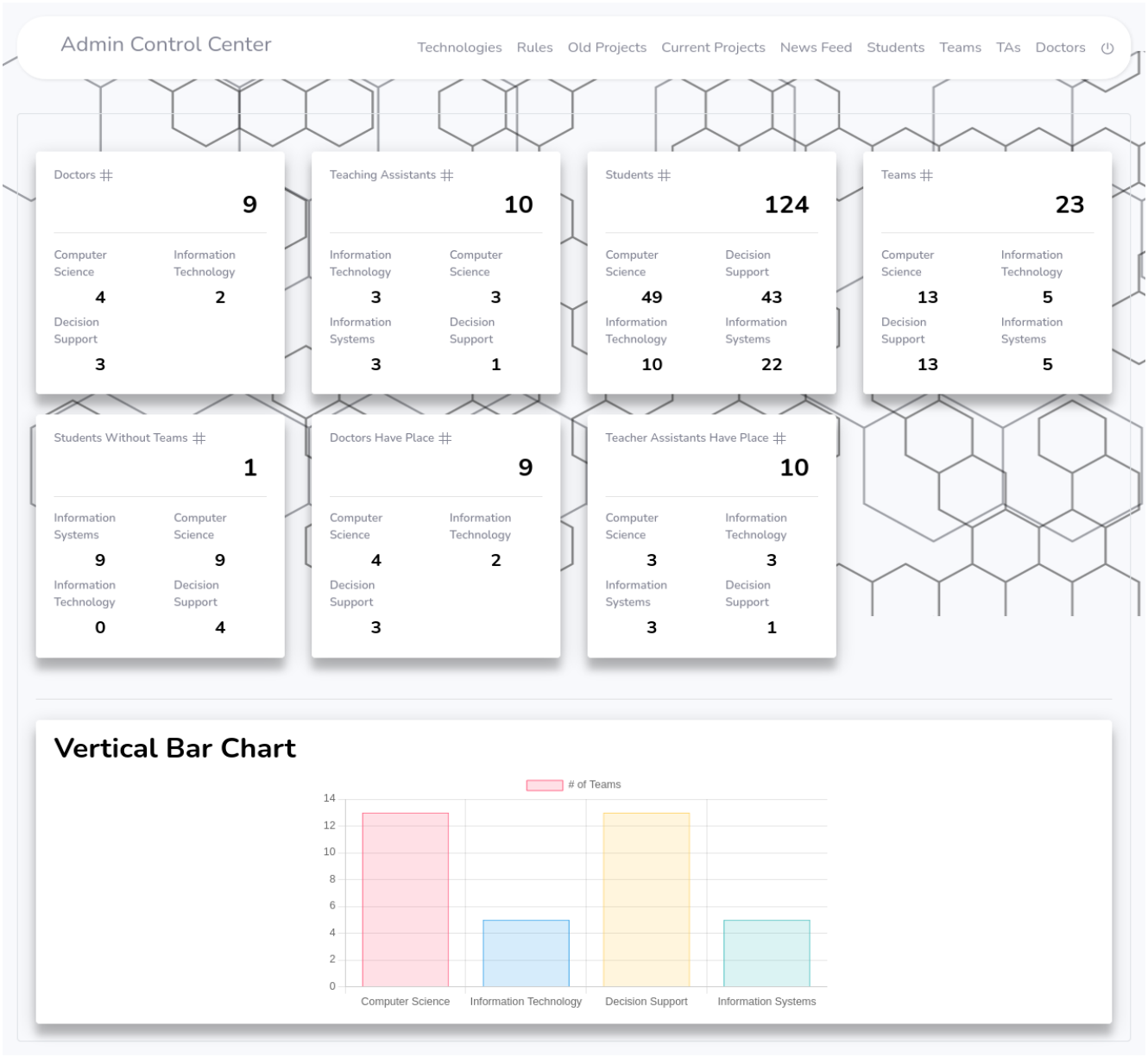


Figure 27: Control Center (Statistics Page)

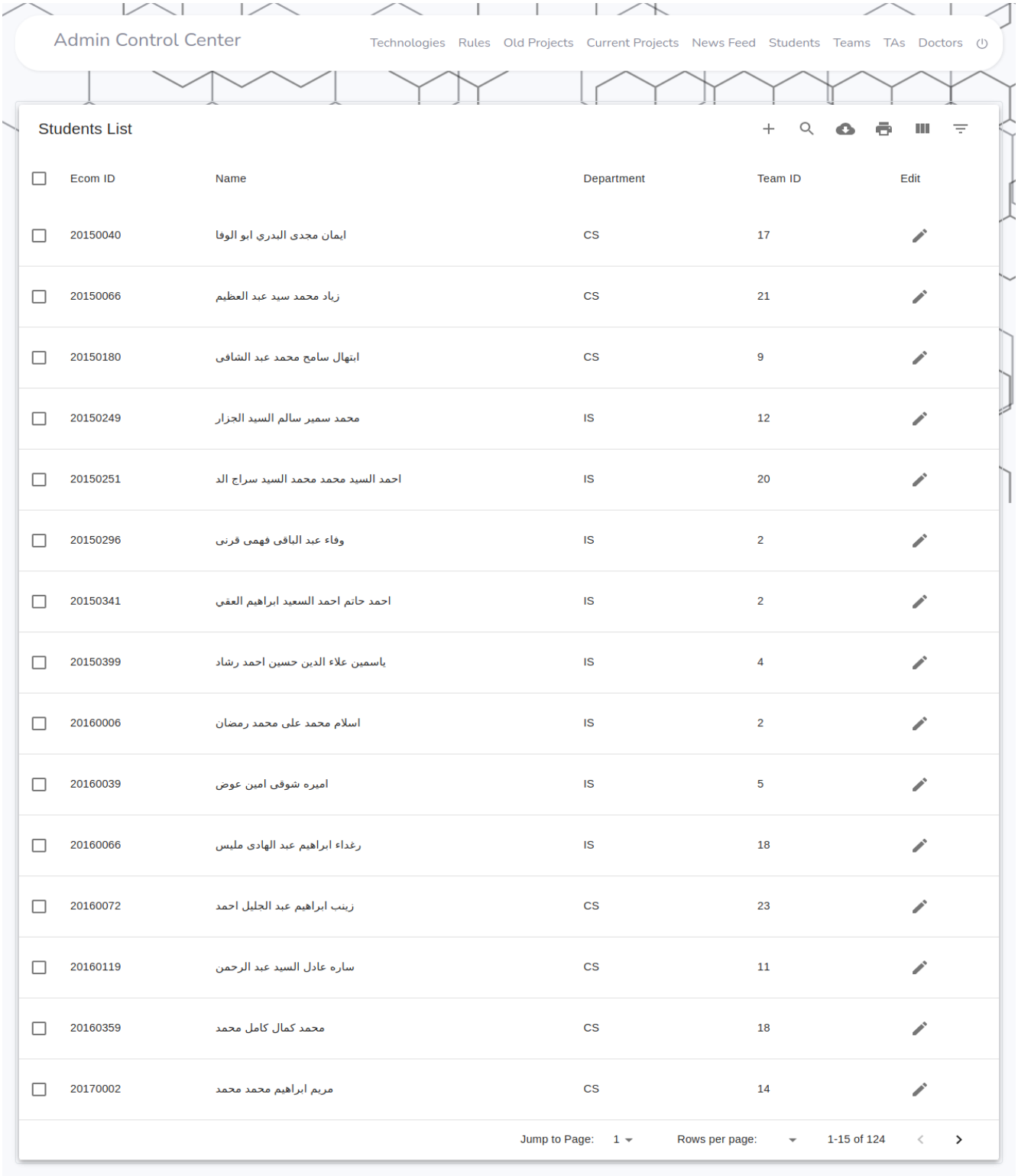


Figure 28: Students Data Table

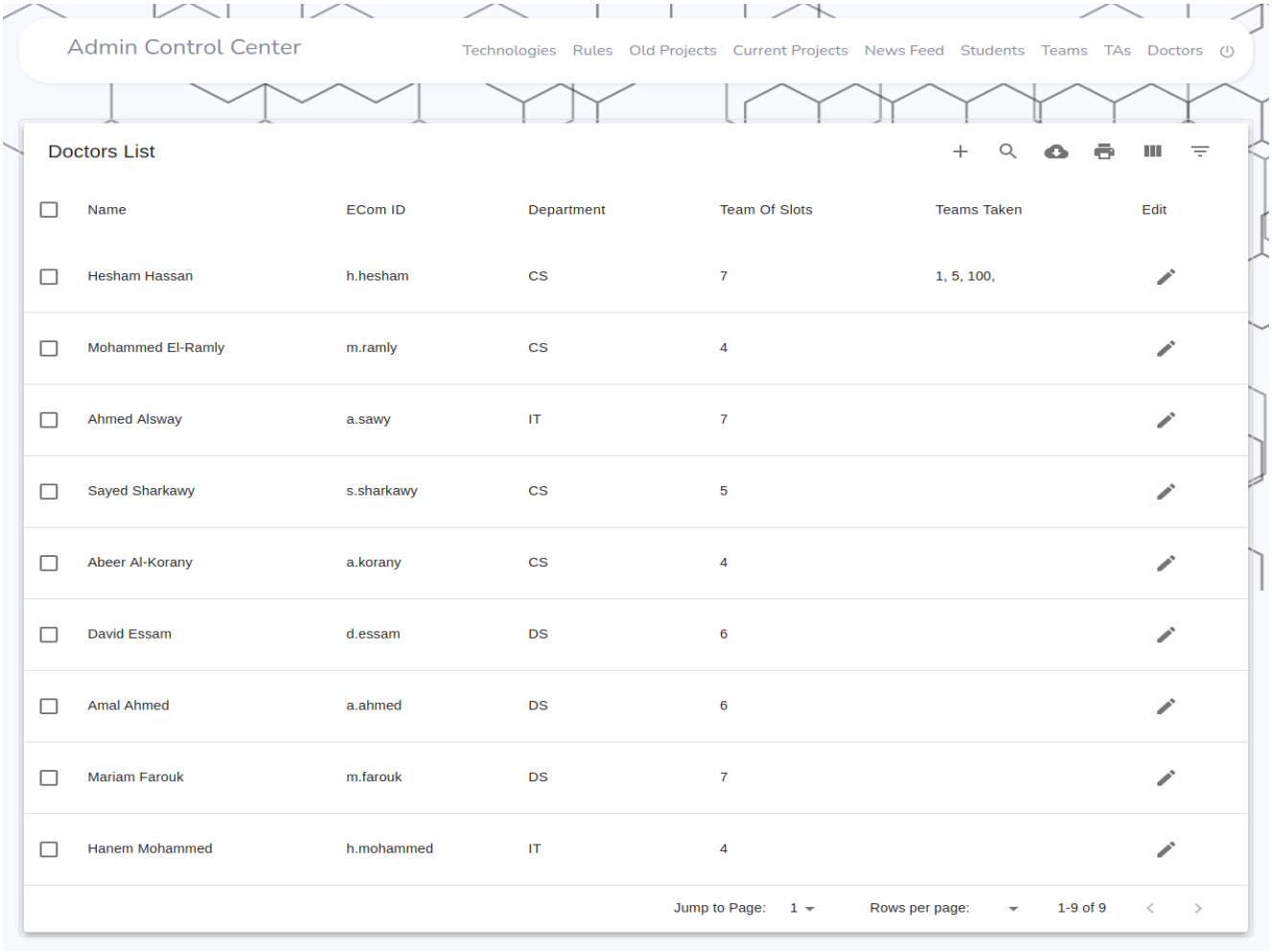


Figure 29: Doctors Data Table

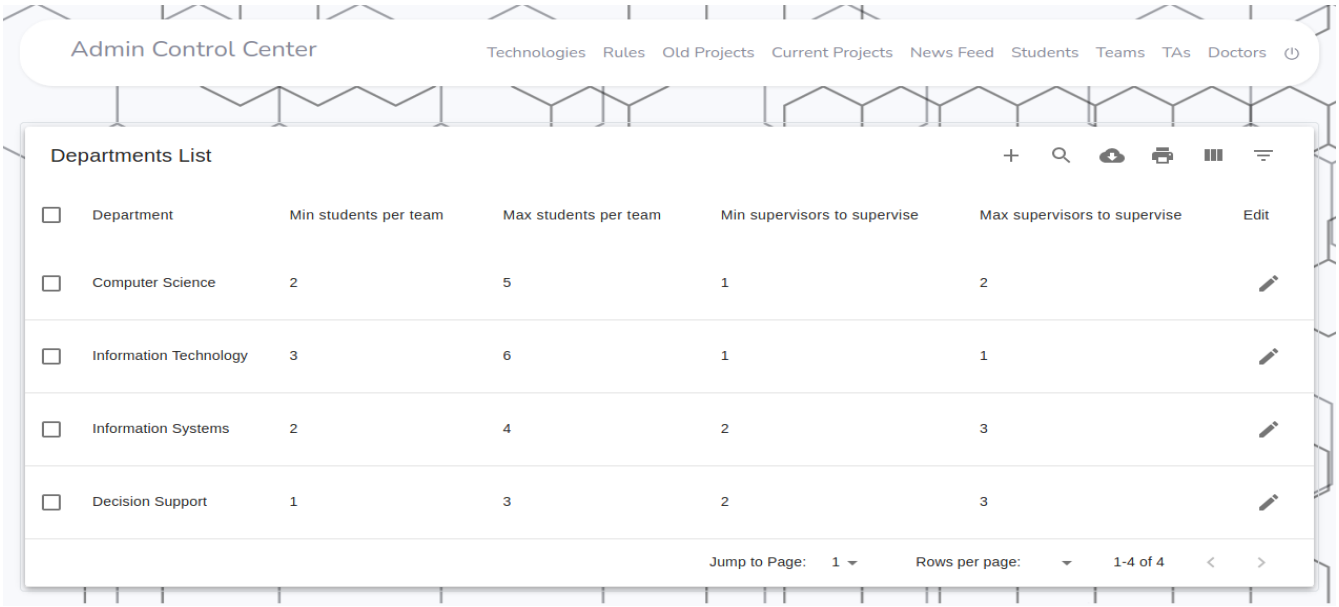


Figure 30: Departments Data Table

Students and supervisors' pages:

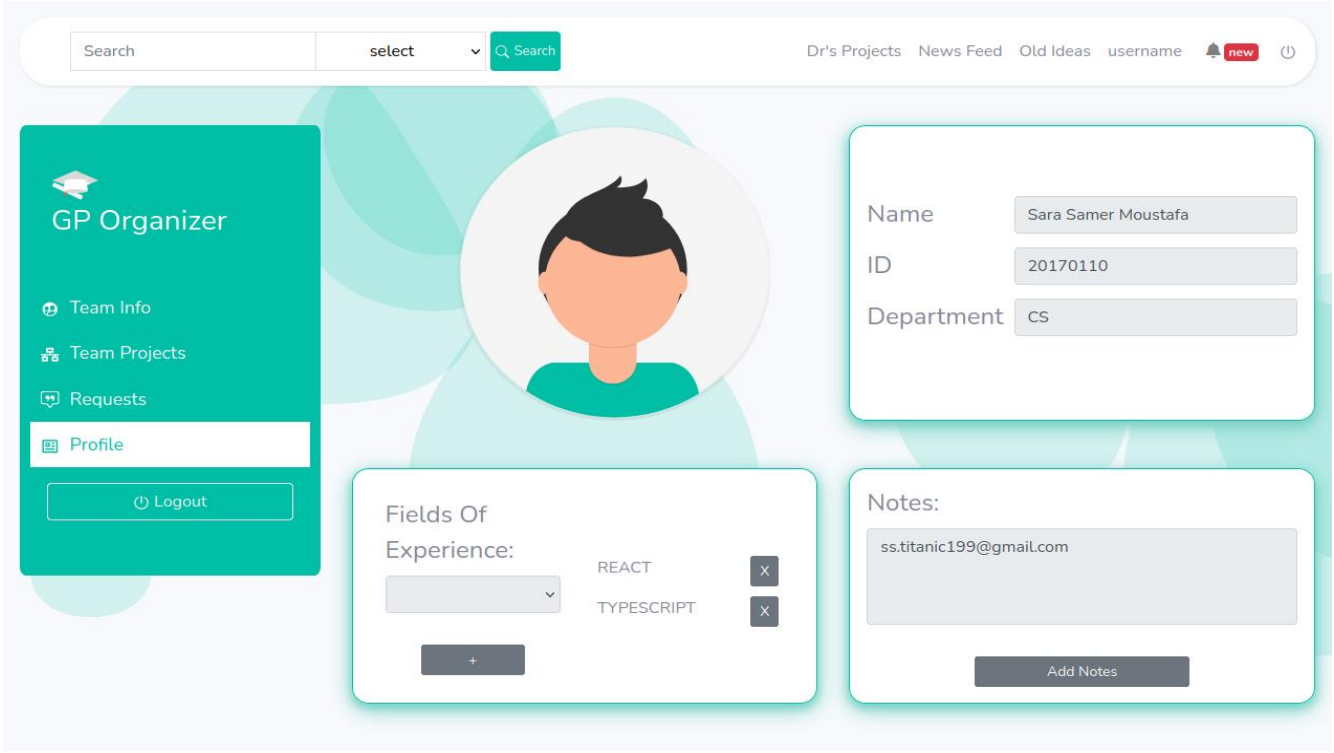


Figure 31: Student's own Profile

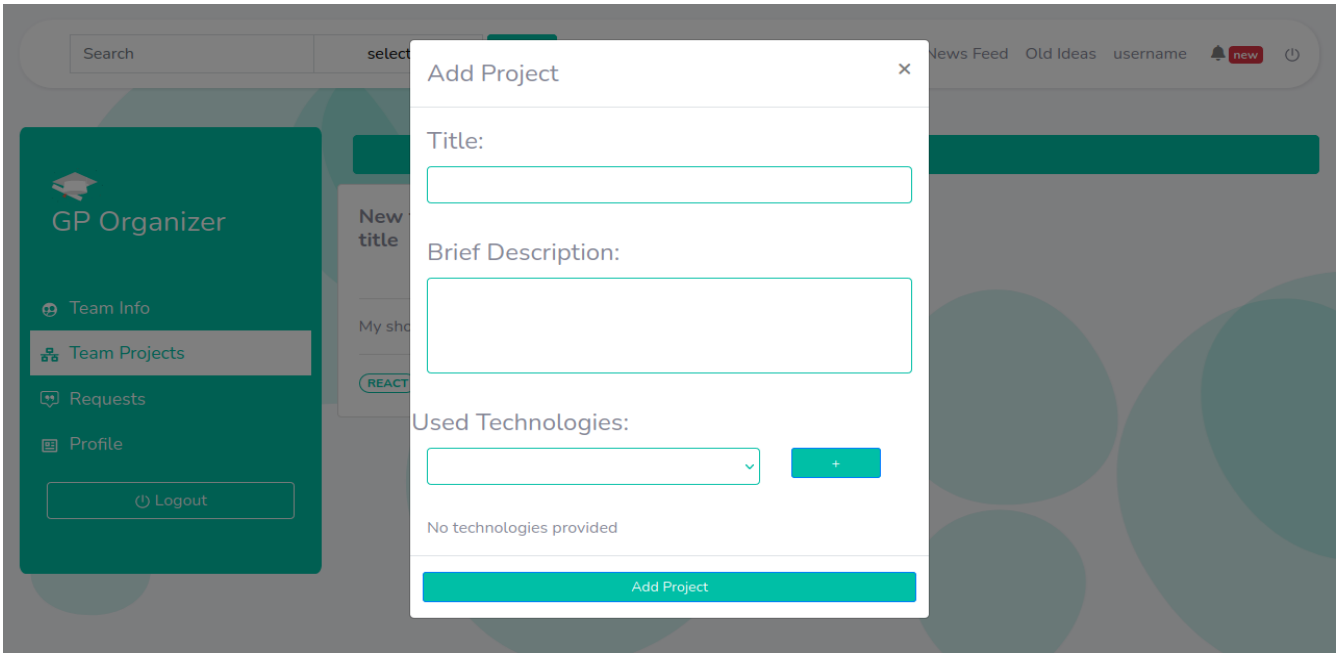


Figure 32: Add Project

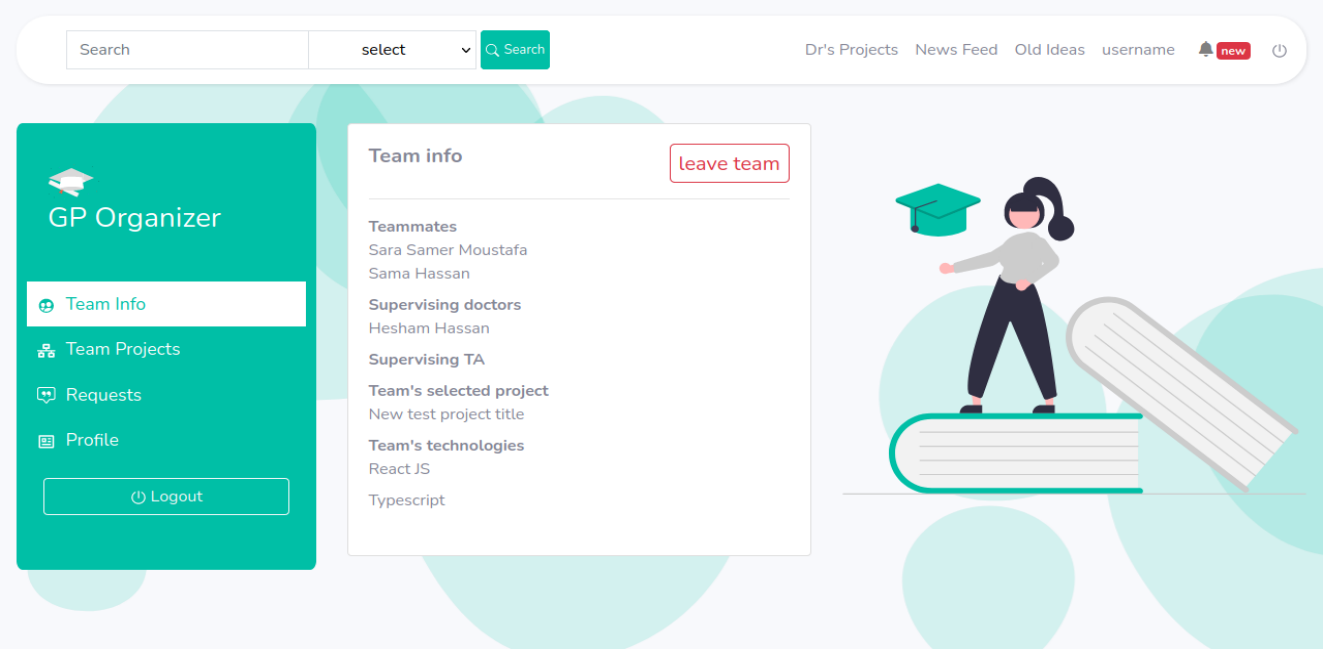


Figure 33: Team Information

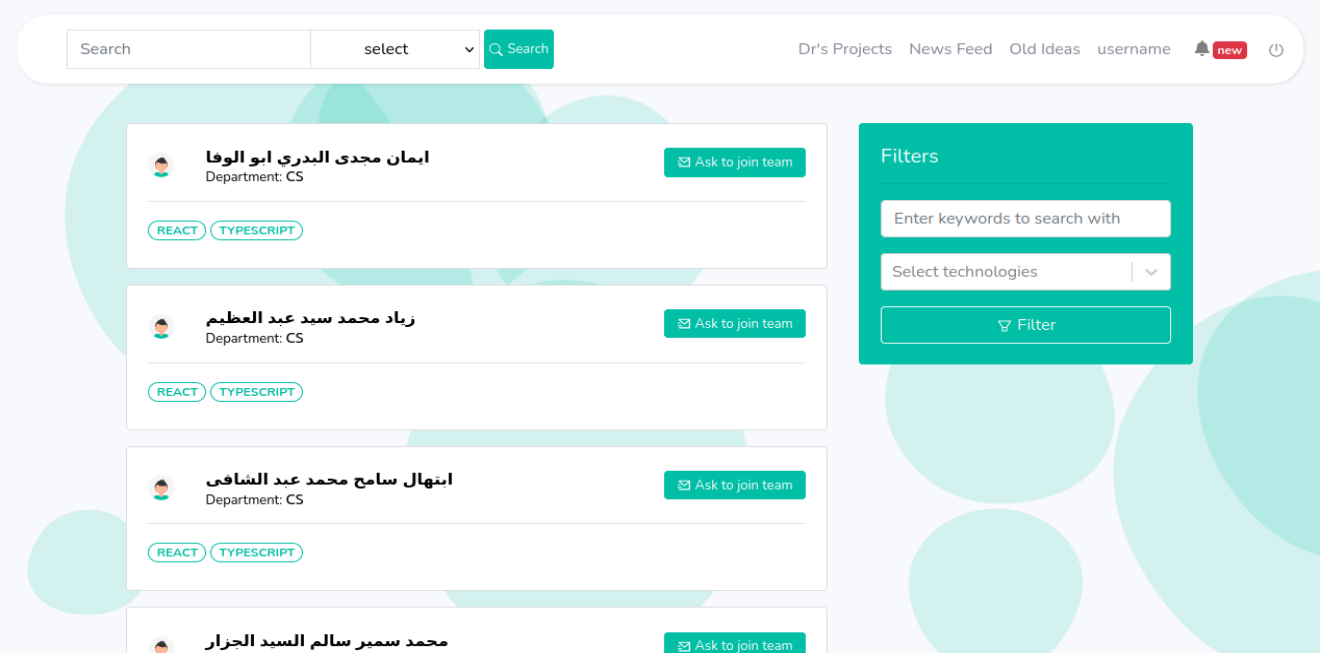


Figure 34: Search All Students

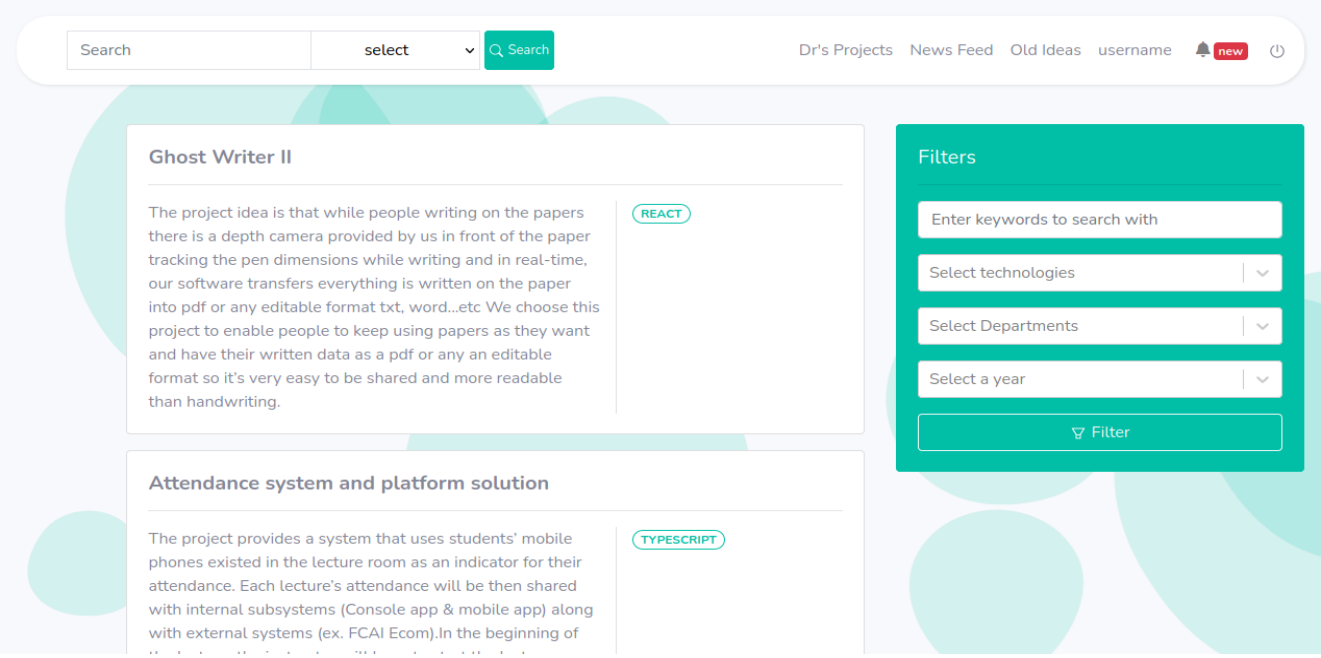


Figure 35: List of Old Ideas

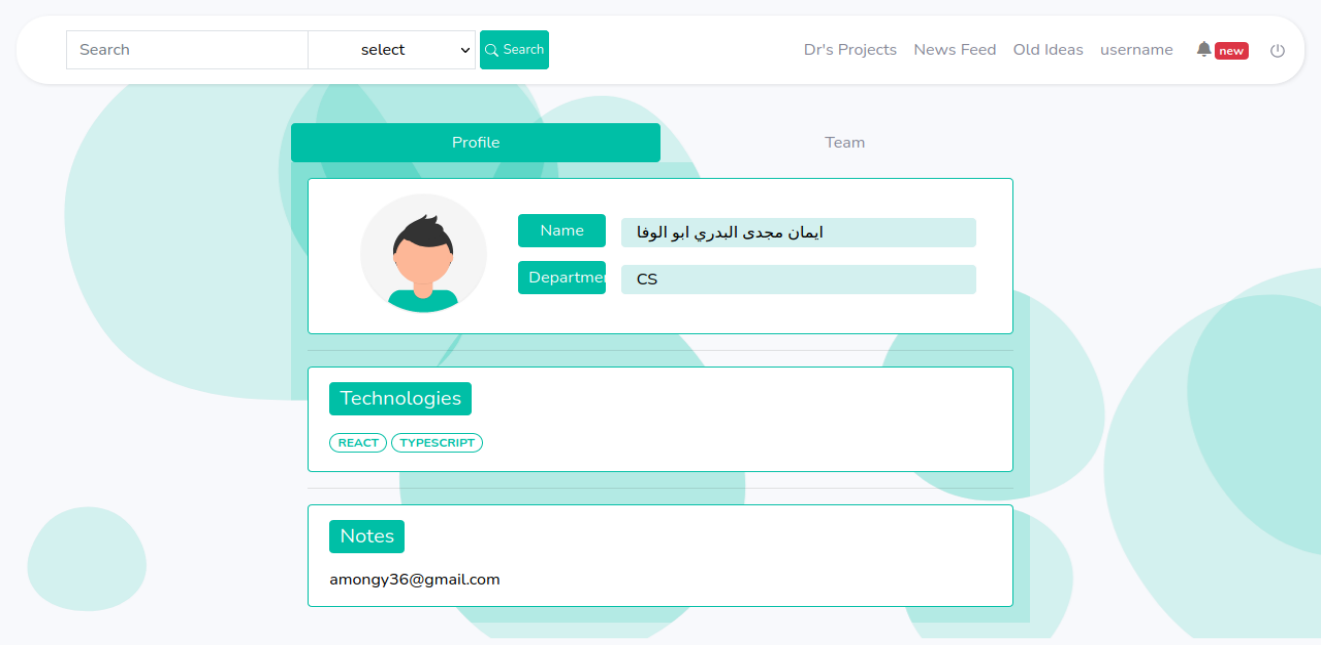


Figure 36: View other Student's Profile

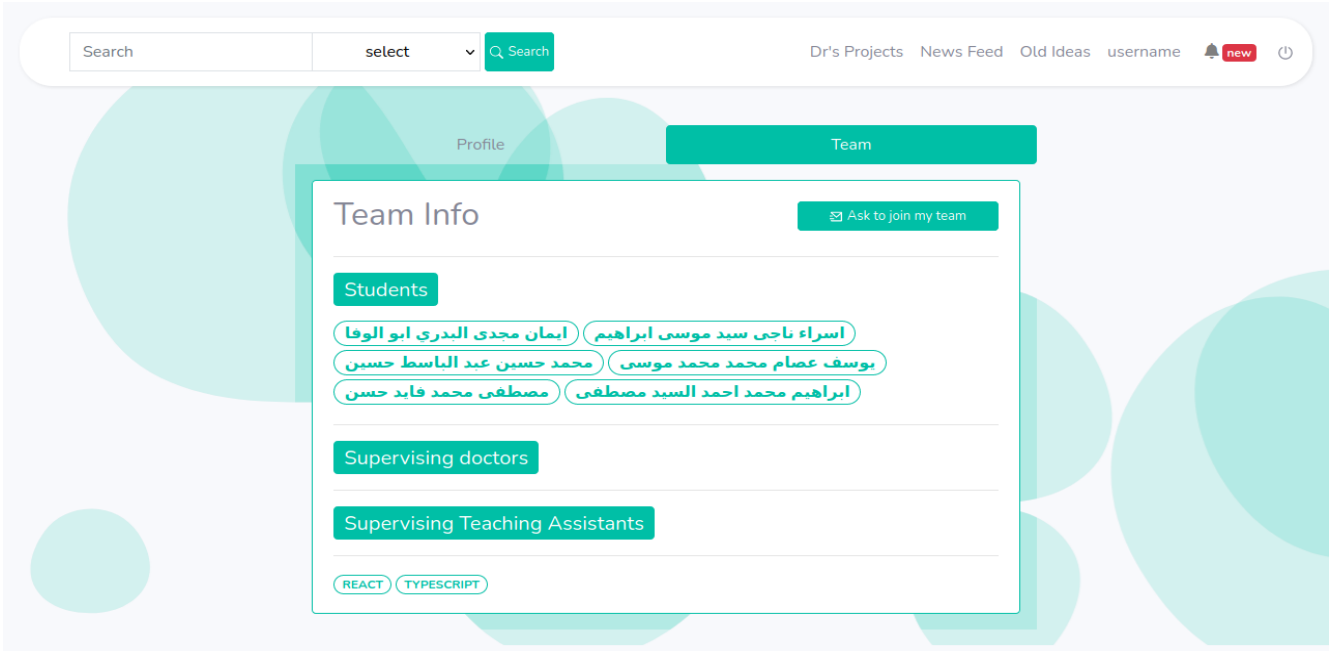


Figure 37: View another Student's Team

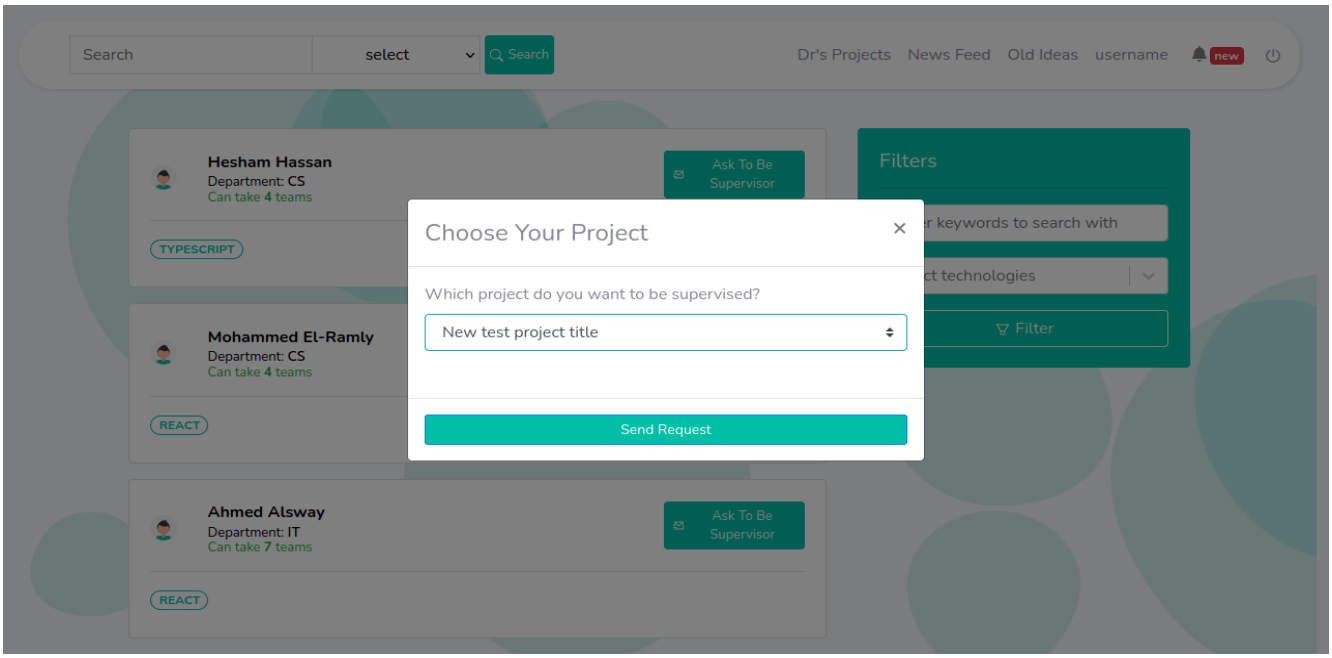


Figure 38: Student ask supervisor to supervise a project

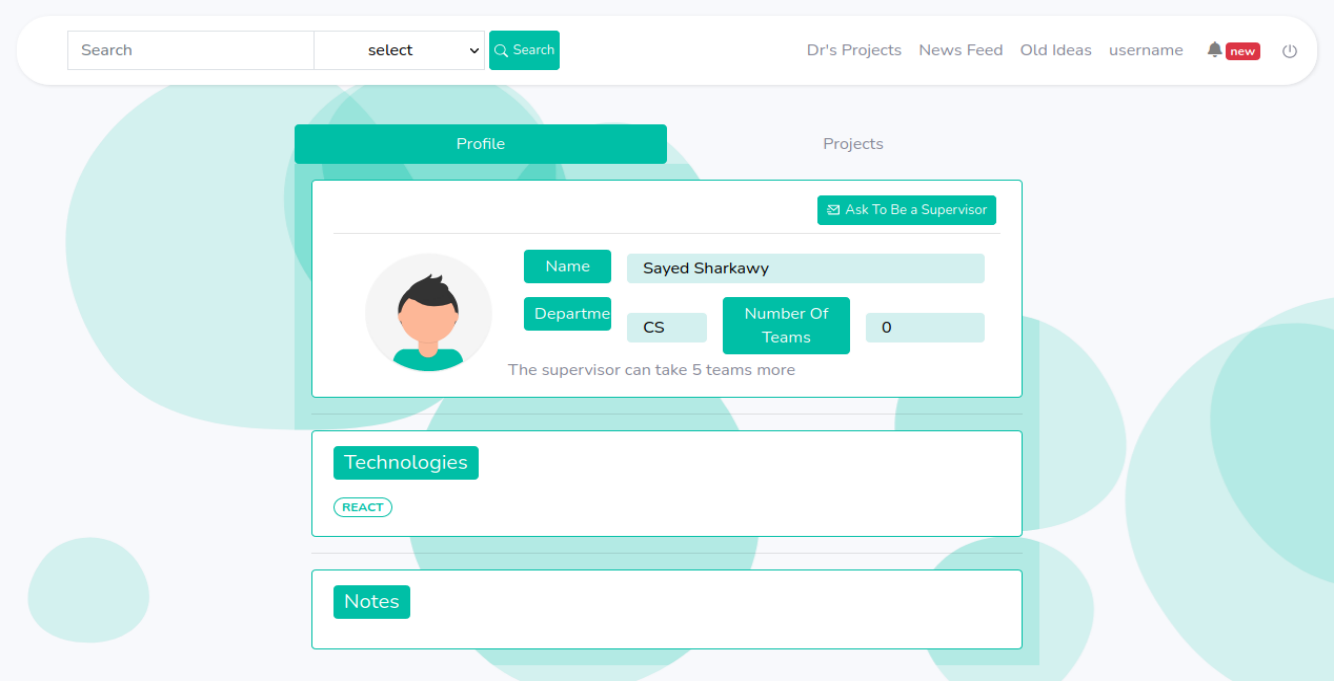
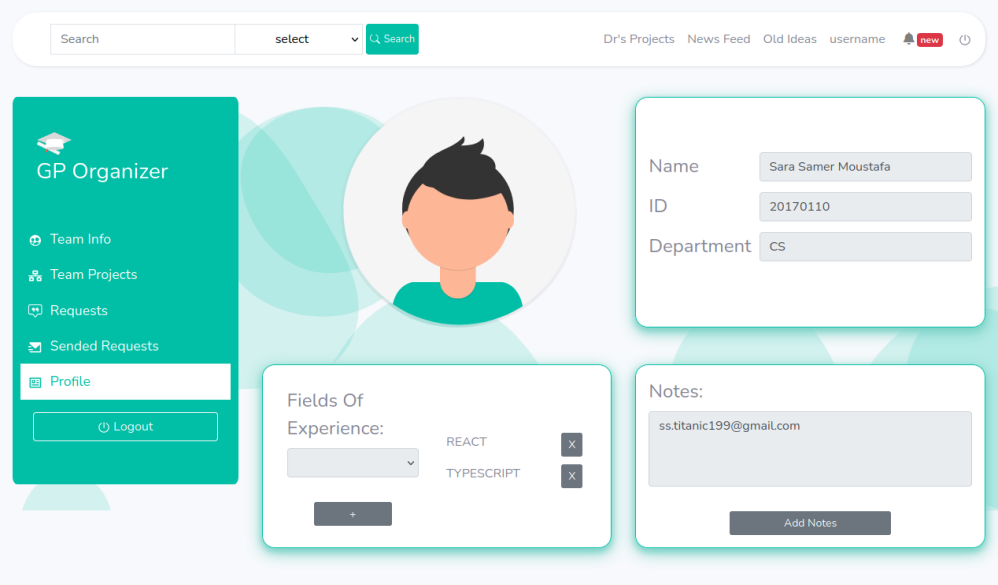
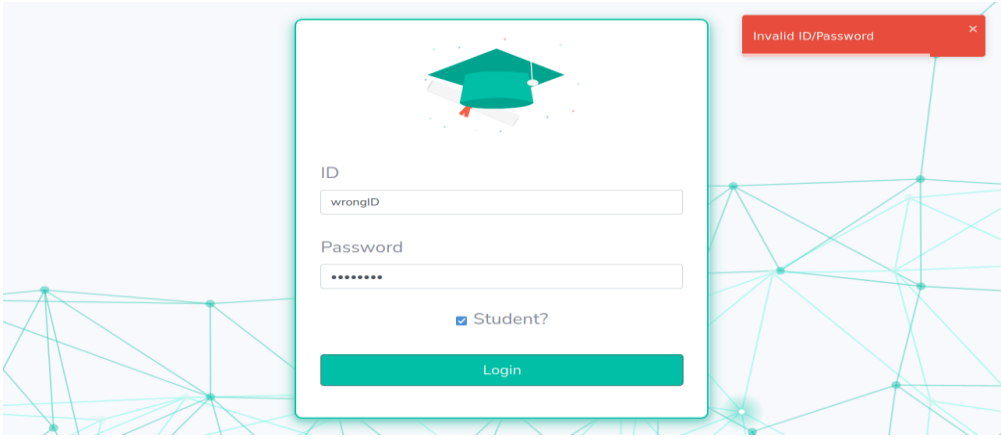
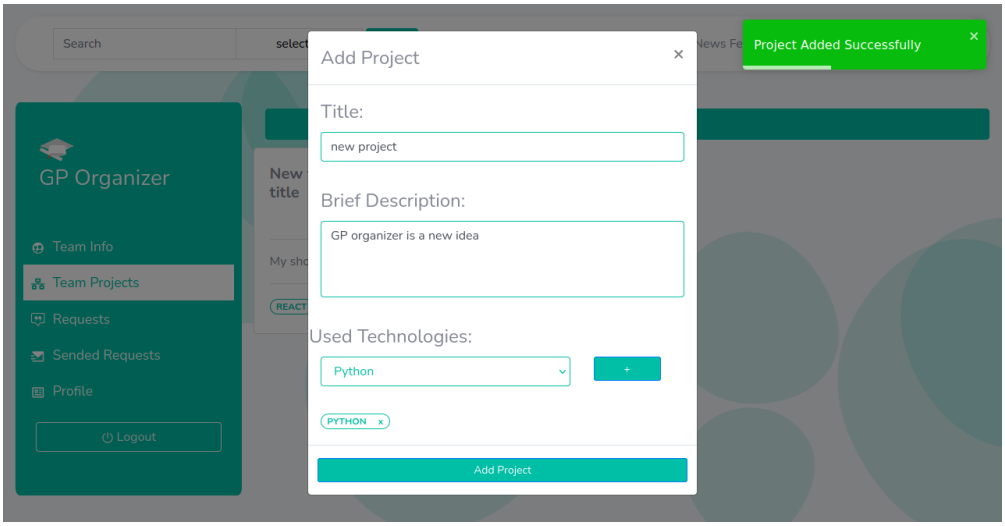
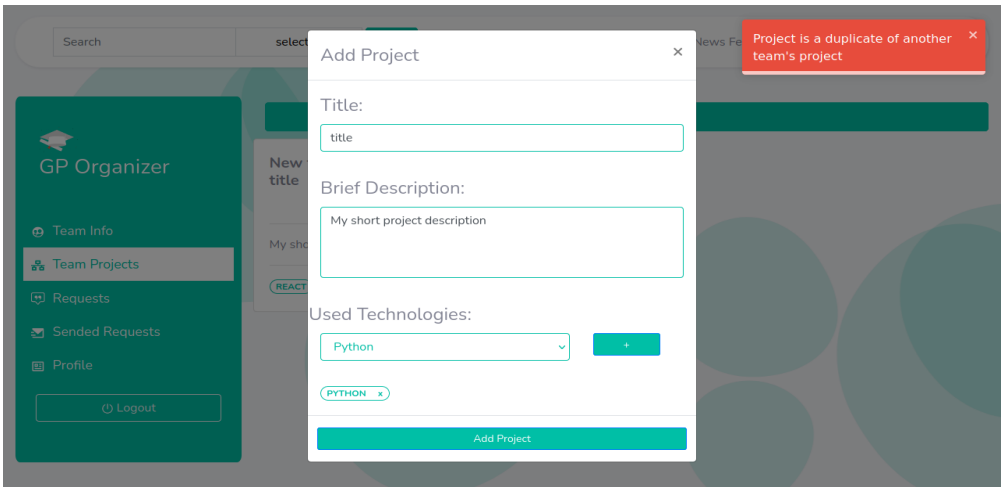


Figure 39: View another Supervisor's Profile

Chapter 5: Implementation and Testing

In this section, we will be testing some functionalities using valid and invalid input to see how the program will react.

Test title & ID	24:Student login
Functionality	Login
Passing input	ID: 20170110 Password: 12341234
Pass output	 ***Figure 40: Student login pass output***
Failing input	ID: wrongID Password: 12341234
Fail output	 ***Figure 41: student login fail output***

Test title & ID	25: Add Project to student team
Functionality	Add project
Passing input	Title: new title Description: GP organizer is a new idea Technologies: Python
Pass output	 ***Figure 42: Add team project pass output***
Failing input	Title: new title Description: GP organizer is a new idea Technologies: Python
Fail output	 ***Figure 43: Add team project fail output***

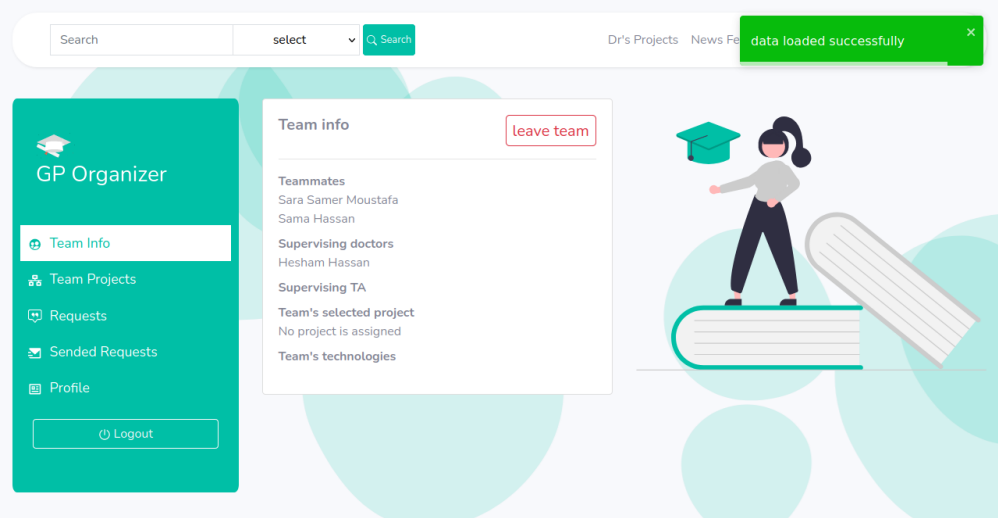
Test title & ID	26: Get Team info
Functionality	Get all members and supervisors
Passing input	Valid access token
Pass output	
Failing input	Invalid access token
Fail output	

Figure 44: Get team info pass output

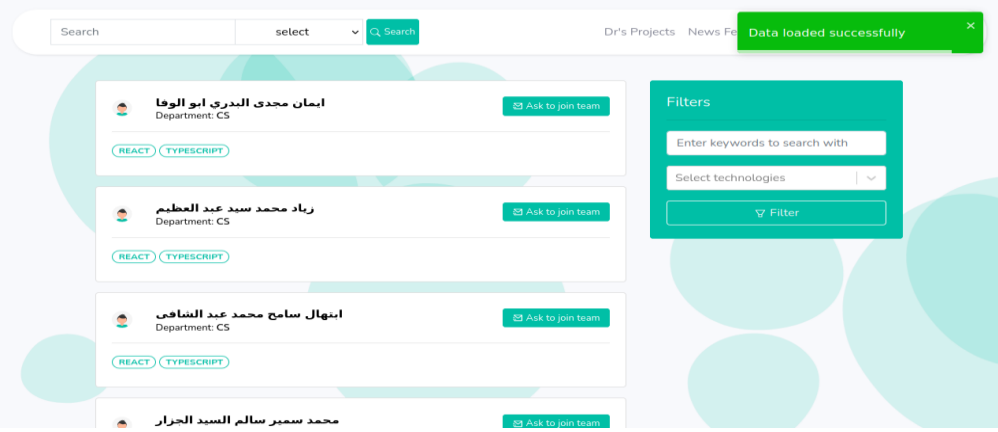
Test title & ID	27: Search students
Functionality	Search
Passing input	Search query: “”
Pass output	

Figure 45: Search students pass output

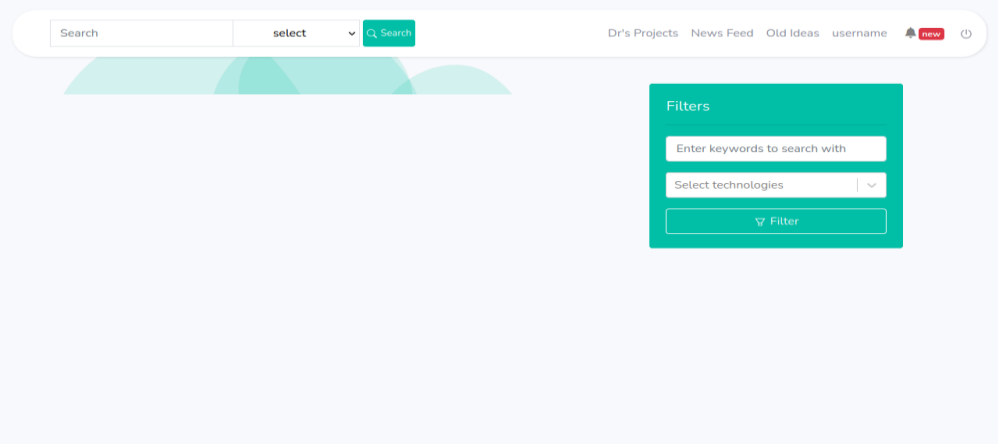
Failing input	Search query: no student
Fail output	

Figure 46: Search students fail output

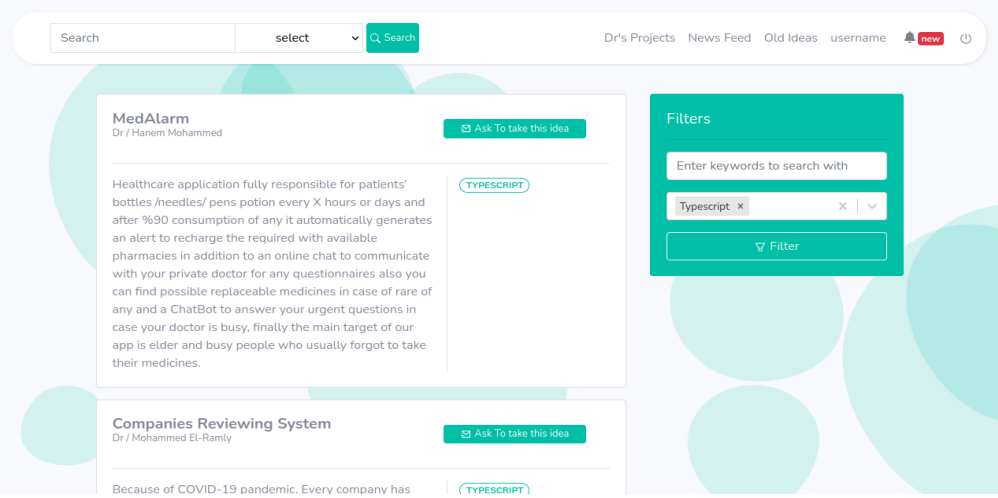
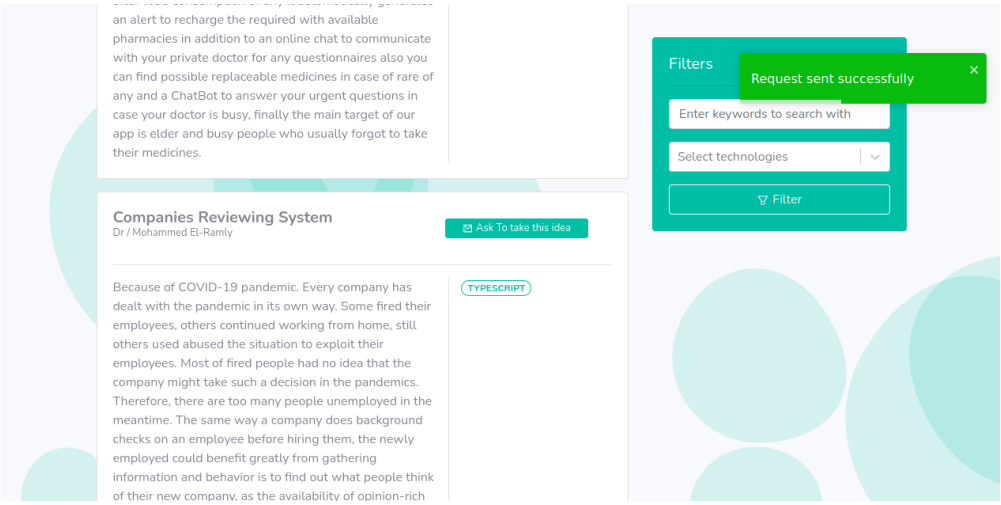
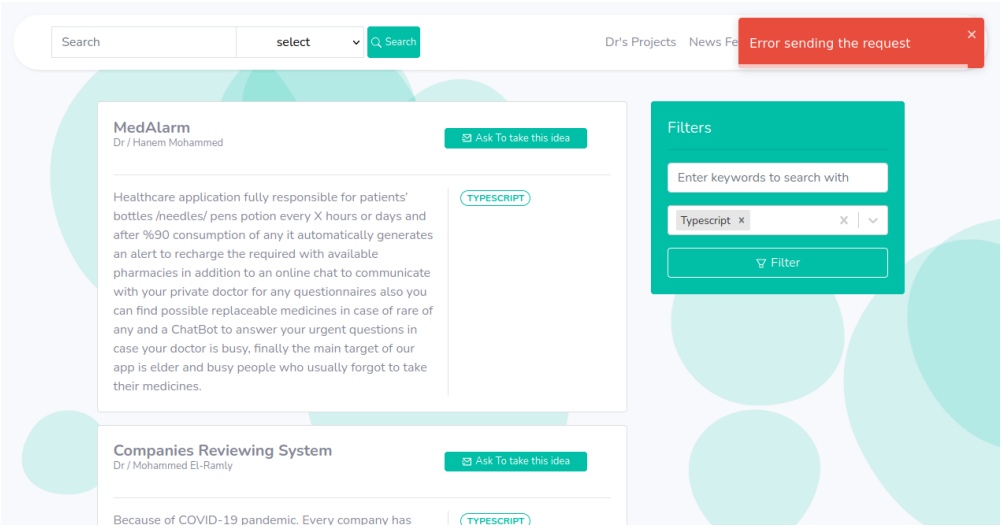
Test title & ID	28: Get supervisor projects
Functionality	Get supervisor project with filter
Passing input	Technologies: TypeScript
Pass output	
Failing input	No invalid input exists
Fail output	Not available

Figure 47: Get supervisor's project pass output

Test title & ID	29: Take supervisor project
Functionality	Send request to other team members to agree tp take on that supervisor
Passing input	Student that has no supervisor & in same department as Dr
Pass output	 ***Figure 48: take supervisor's project pass output***
Failing input	Student in different department
Fail output	 ***Figure 49: take supervisor's project fail output***

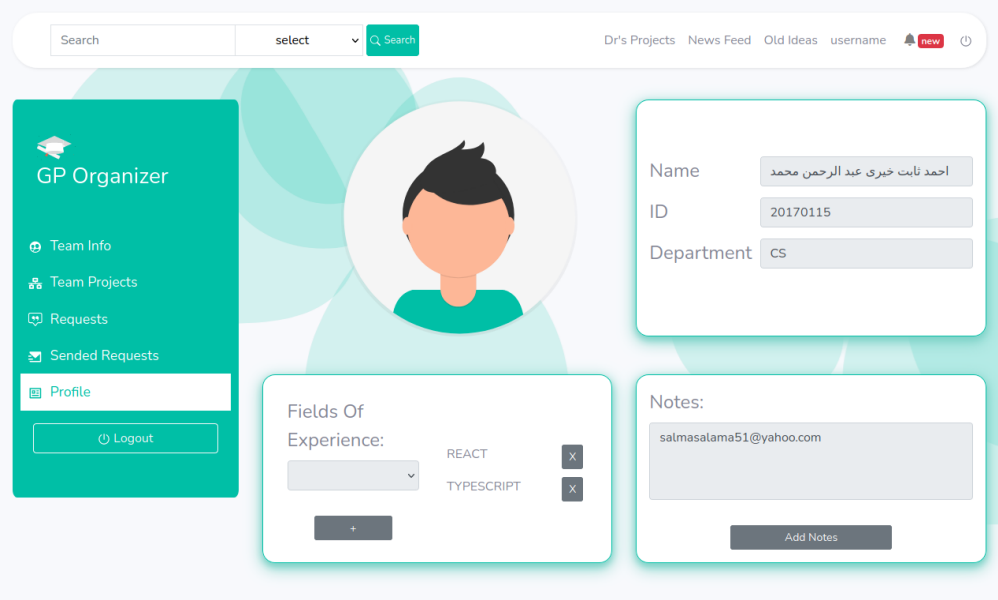
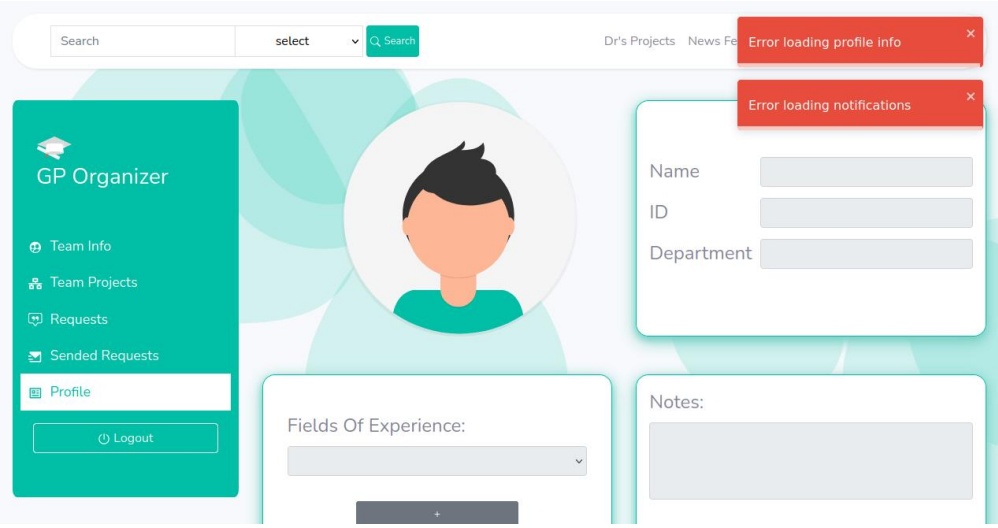
Test title & ID	30: Get User Profile
Functionality	Fetch user profile from backend
Passing input	Valid access token
Pass output	
Failing input	Invalid access token
Fail output	

Figure 50: get student profile pass output

Figure 51: get student profile fail output

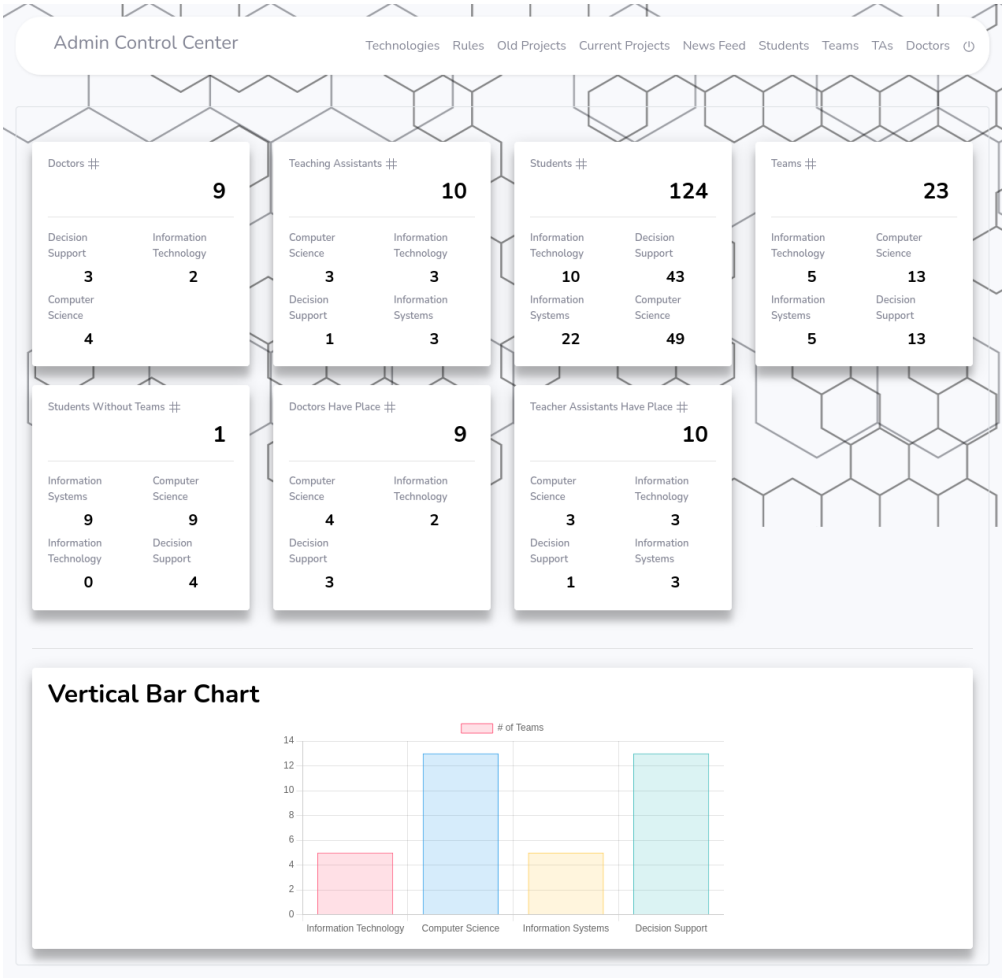
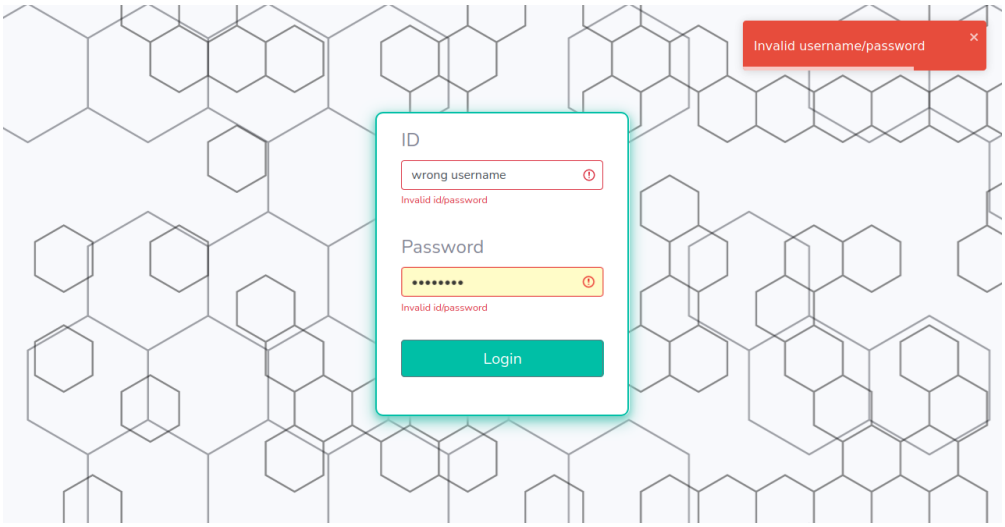
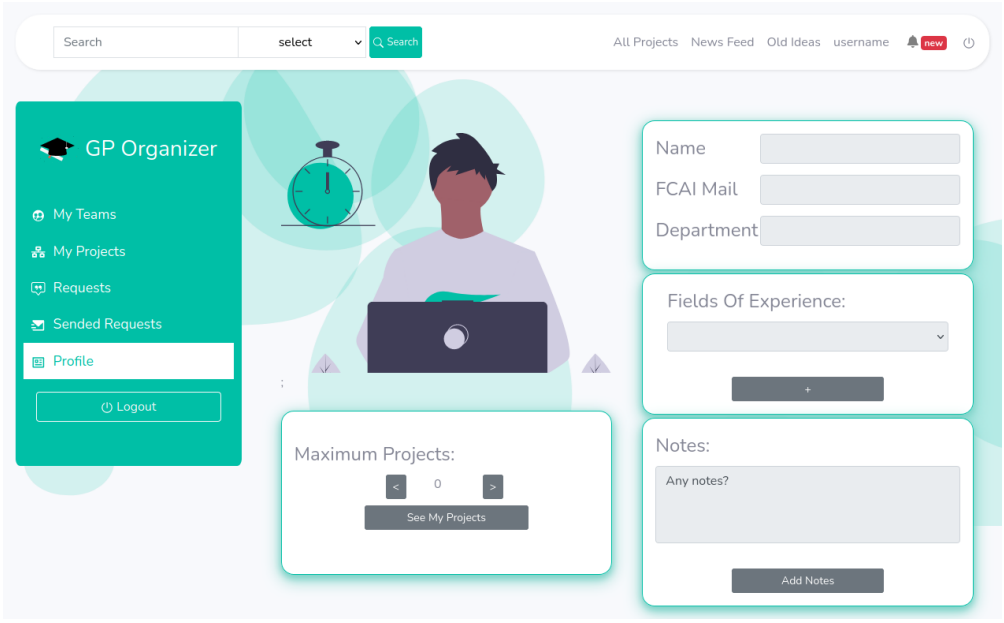
Test title & ID	31: Admin login
Functionality	Login
Passing input	Username: admin Password: pasword
Pass output	
Failing input	Username: wrogn username Password: password

Figure 52: admin login pass output

Fail output	 ***Figure 53: admin login fail output***

Test title & ID	32: Supervisor login
Functionality	Login as supervisor
Passing input	ID: h.hesham Password: 12341234
Pass output	 ***Figure 54: supervisor login pass output***
Failing input	ID: wrong Password: 12341234

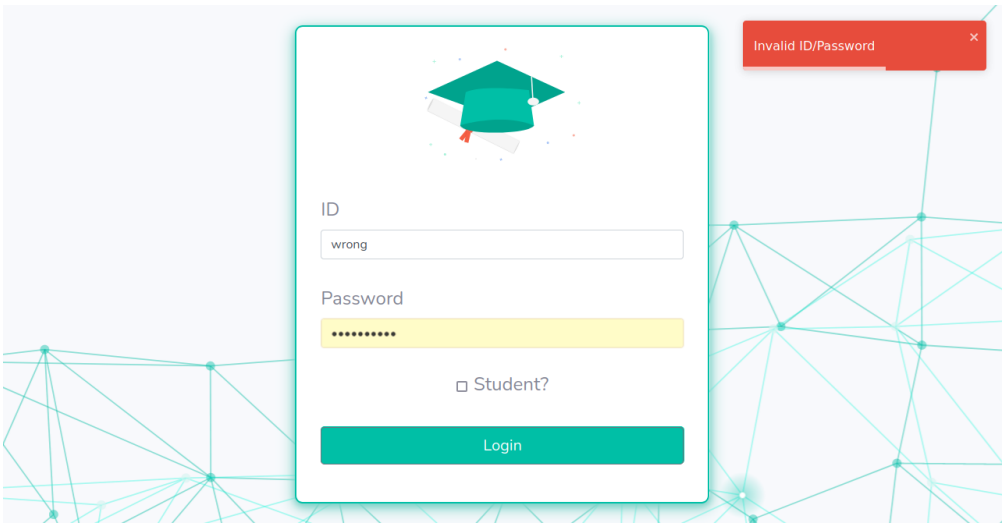
Fail output	

Figure 55: supervisor login fail output

Chapter 6: Conclusion and Future Work

We can conclude from this documentation the need for having a platform for organizing the process of making a graduation project. The existence of a platform will facilitate a lot of steps that need to be done by all the stakeholders, especially the senior students and the doctors. The existence of this platform will save a lot of both time and effort from choosing the idea or changing it in case it turns out to be a duplicate and will convert the saved time to be used in the other steps especially the manufacture of system architecture and the implementation which need a lot of time.

Some functionalities have been left out due to lack of time and test data. Future work concerns matchmaking for admin, a supervisor writing announcement for a specific team, adding co-supervision for teams and setting graduation project registration deadline for admin.

References

[1] Knight, Rachael-Anne & Botting, Nicola. (2016). Organizing undergraduate research projects: student-led and academic-led models. Journal of Applied Research in Higher Education. 8. 455-468. 10.1108/JARHE-07-2015-0054.

[2] L'Anson, R., & Smith, K. (2004). Undergraduate Research Projects and Dissertations: Issues of Topic Selection, Access and Data Collections amongst Tourism Management Students. Journal of Hospitality, Leisure, Sport And Tourism Education 3 (1) 19-3

[3] Khelifi, Adel & Al-Rajab, Murad & Karem, Sara & Subhi, Lamia. (2011). Graduation project online management system ALHOSN university case study.

[4] (Agile-Waterfall Hybrid: Is It Right for Your Team? | Lucidchart Blog, 2021)