



SOFTWARE DOCUMENTATION

V6.0

For Productivity Manager System

Abstract

Submitted in partial fulfillment of the requirements of the software engineering course

Habiba Amr	2020/08121
Noran Essam	2020/07406
Rana Ehab	2020/15051
Mariam Maged	2020/00559
Belal Adel	2020/11213

Thursday, 24 November 2022

CONTENTS

Introduction	3
Purpose	3
Scope	3
Glossary	3
Document Overview	4
Overall Specification	5
System Environment	5
System Context Diagram	5
Use Case Diagram	6
Functional Requirements Specification	7
Non-Functional Requirements	7
Use Cases	8
Log In	8
Add User	9
View Users	10
Update User	11
Delete User	12
Add Employee	14
View Employees	16
Update Employee	17
Delete Employee	18
View Cross-utilized Employees	19
View Over-utilized Employees	20
Add Project	21
View Projects	22
Update Project	23
Delete Project	24
Assign Project Leaders	25
Add employees to project teams	26
Mark Project as done	27
Update Project Capacity	28
Update Employee Utilization on Project	29

View statistical details	30
Sequence Diagrams	31
Add new Project	31
Add New User	32
Assign Employee to project	33
Update employee utilization On Project	34
View employees.....	35
View Projects	36
View project statistics.....	37
View Employee Statistics	38
Designs.....	39
Class Diagram	39
Object Diagram	40
Database Physical Design	41
Prototype.....	42
1- Admins can manage users' data.	42
2- Admins can manage projects' data.	44
3- Team leaders can add or remove employees to their team.	45
4- Team leaders can add the required employee capacity to their team members.	46
5- Team leaders and Admins can view a List of all the projects and their data	46
6- Admins and team leaders can view statistical details	47
7- Admins can manage employees	47

INTRODUCTION

PURPOSE

The purpose of this document is to present a detailed description of the Productivity Manager. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system.

This document defines the system design for this productivity management system. It should illustrate how classes will be structured and arranged and how they will communicate.

SCOPE

This system is a productivity enhancing tool that keeps track of projects data, employees' work utilization. This system is a web-based application that provides interfaces for various stake holders (project leaders, managers/admins, employees).

This system's admins can add new recruits to the employee pool where they'd be assigned to projects according to the respective project leader's request. The admins can monitor the current work utilization of employees and view over-utilized and cross-utilized employees to better divide their workloads if possible and/or necessary. This system's admins can also view the status and punctuality of projects. The system uses a lazy delete implementation to save historic data. The system aggregates relevant and useful statistics to support data driven decision making regarding how workload is managed within the department.

GLOSSARY

Term	Definition
User/ Actor	The end user of the system. This user can either be an admin or a project leader.
Project leader	The user in charge of managing projects. This user can operate on their project data as well as view other projects and employees.
Admin	The user in charge of the system. This user can operate on user, employee, and project data.
Employee	The human resource which is utilized for projects.
Utilization	The amount by which the employee is busy with work.
Cross-utilization	The state where an employee is working on multiple projects at once.
Over-utilization	The state where an employee is overtasked.
Project Capacity	The number of employees needed for the project.
Statistical details	Collections of information extracted from saved data.

DOCUMENT OVERVIEW

The Overall Description section of this document gives an overview of the functionality of the product. The system environment chapter describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The Requirements Specification section of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

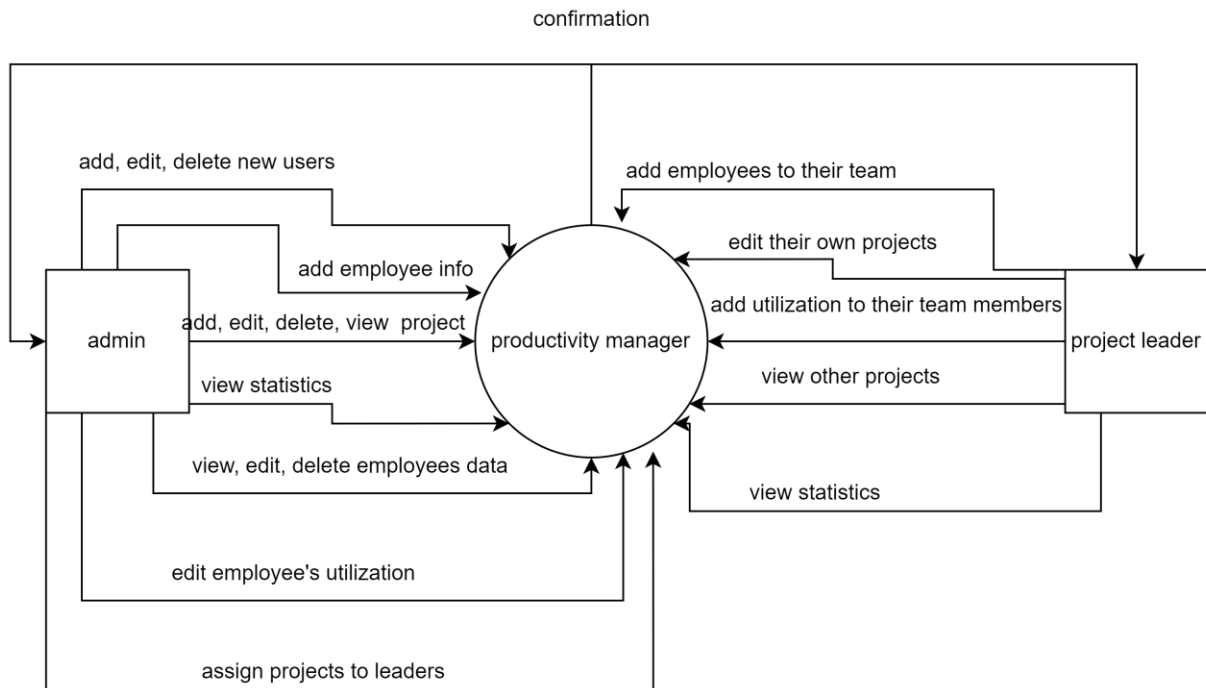
Both sections of the document describe the same software product in its entirety but are intended for different audiences and thus use different language.

The system structure is described in the class diagram. An example of instances is described in the Object Diagram. The physical database ERD is also included.

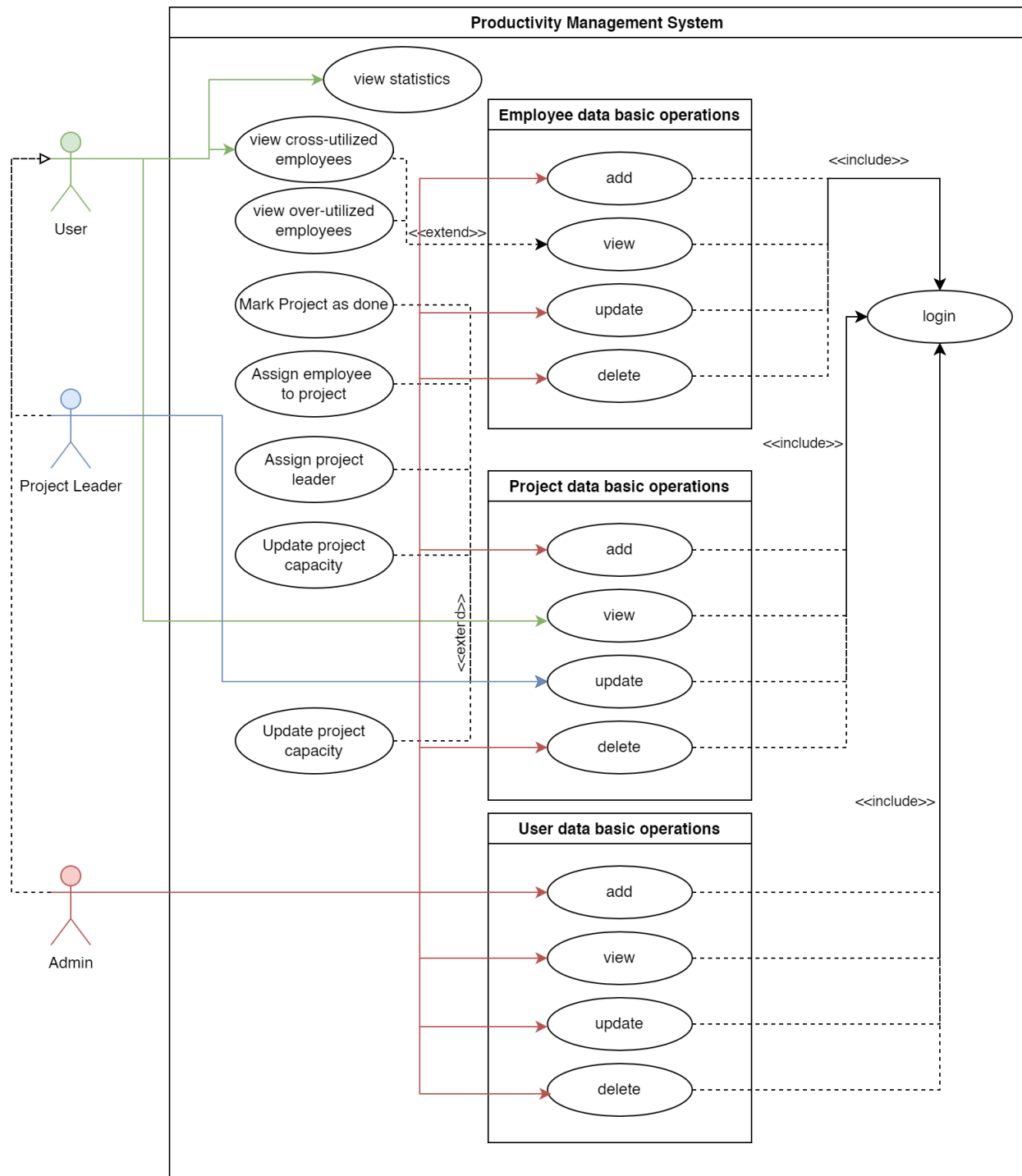
OVERALL SPECIFICATION

SYSTEM ENVIRONMENT

SYSTEM CONTEXT DIAGRAM



USE CASE DIAGRAM



FUNCTIONAL REQUIREMENTS SPECIFICATION

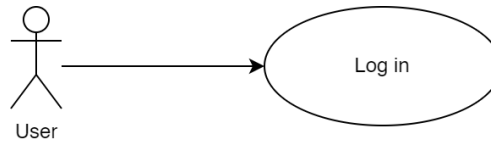
ID	Description	Comments
R0	Admins can manage users' data.	
R1	Admins can manage projects' data.	
R2	Team leaders can add or remove employees to their team.	
R3	Team leaders can add the required employee capacity to their team members.	
R4	Team leaders and Admins can view a List of all the employees and their data	
R5	Team leaders and Admins can view a List of all the projects and their data	
R6	Admins and team leaders can view statistical details	Statistical details include: Employees' utilization distribution, programming languages, average utilization of programming languages, projects distribution by regions, cross utilization distribution.

NON-FUNCTIONAL REQUIREMENTS

- User friendly interface with simple readable and simple statistics.
- Our system has high speed efficiency.
- Our system is combatable with other operating systems.

USE CASES

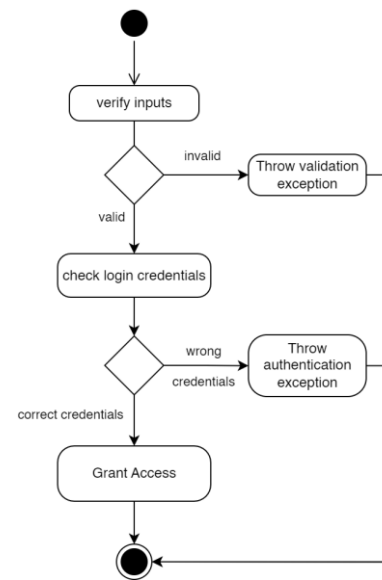
LOG IN



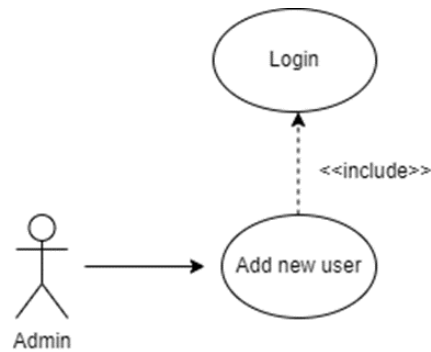
Property	Value
ID, Name	UC1, Log in
Goal	A user should be granted access to the system through giving their email and password.
Initiator	Any user
Precondition	None
Postcondition	<ul style="list-style-type: none">The given user credentials will be granted access if their credentials were verified.

STEP-BY-STEP DESCRIPTION

1. The actor submits a login request with their email and password.
2. The system hashes the given password to match the hashed value in the database.
3. The system checks that a user with the given email and password exists.
 - a. On success, the user is granted access to the site.
 - b. On failure, a message appears to the user notifying



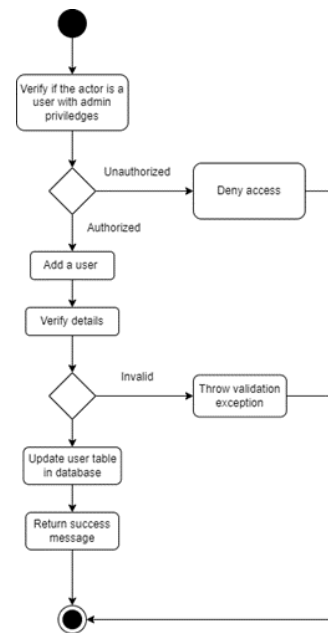
ADD USER



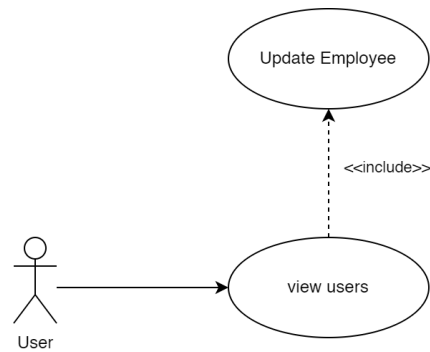
Property	Value
ID, Name	UC2.1, Add new user.
Goal	Users should be added to the list of users.
Initiator	Admin.
Precondition	<ul style="list-style-type: none">• A user with admin privileges should be logged in.
Postcondition	<ul style="list-style-type: none">• Users should be added to the list of users.

STEP-BY-STEP DESCRIPTION

1. The system checks that the actor is a user with admin privileges.
 - a) If the actor isn't an admin, the system denies their access.
 - b) If the actor is an admin, the system redirects them to the "add user page."
2. The admin proceeds to add a user to the list and enters all the details related to the user.
3. The user is then added to the database in a new row.
4. A success message is sent to the admin if the project is successfully added to the database.



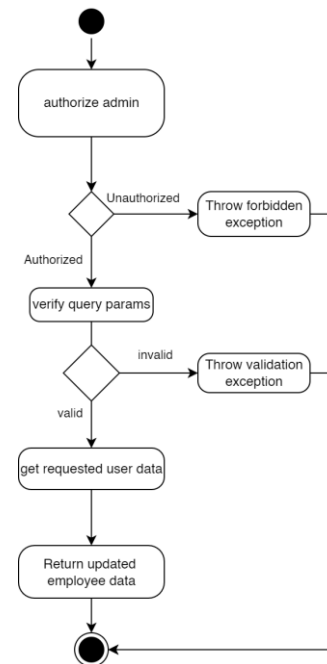
VIEW USERS



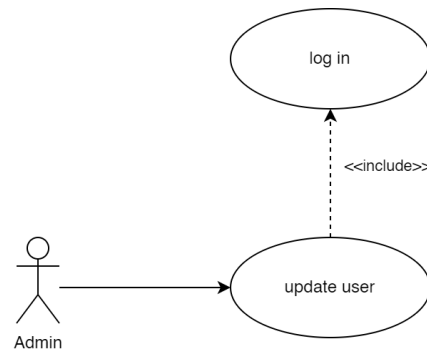
Property	Value
ID, Name	UC2.2, View users
Goal	view data of users for the user
Initiator	Admin
Precondition	<ul style="list-style-type: none"> A user with admin privileges should be logged in.
Postcondition	<ul style="list-style-type: none"> Employees data is returned to the user if found

STEP-BY-STEP DESCRIPTION

1. The actor requests to view employees.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system verifies the query parameters of the request.
3. The system queries the database to get employees data with the given filter if exists.
4. The system returns a response carrying a success code and all employee's data.



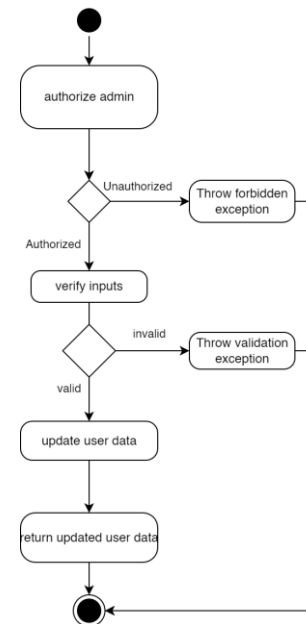
UPDATE USER



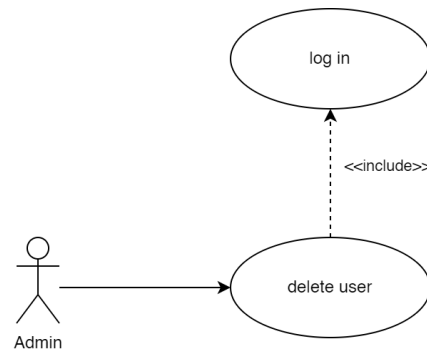
Property	Value
ID, Name	UC2.3, Update user
Goal	Inputted changes should apply to the table of users.
Initiator	Admin
Precondition	<ul style="list-style-type: none"> A user with admin privileges should be logged in.
Postcondition	<ul style="list-style-type: none"> The user's properties will be updated. The new user data will be returned in response.

STEP-BY-STEP DESCRIPTION

- The actor submits an update request to update one of the users.
- The system runs validation checks, upon the failure of any, throws an exception.
 - The system checks the actor's login validity.
 - The system checks that the actor is either the project leader assigned to this project or a user with higher privileges.
 - The system verifies the input format of the request.
- The system queries the database to update the given user's properties with the new data.
- The system returns a response carrying a success code and the updated user's data.



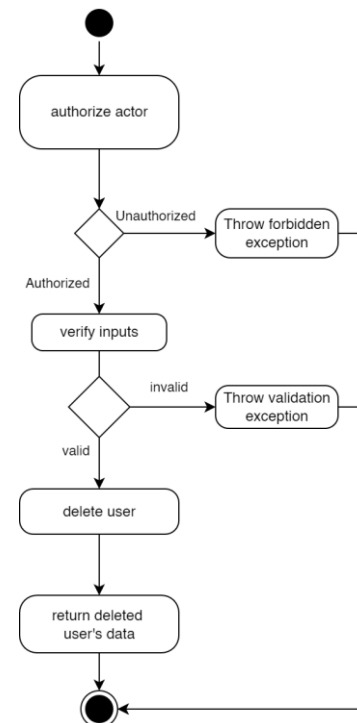
DELETE USER



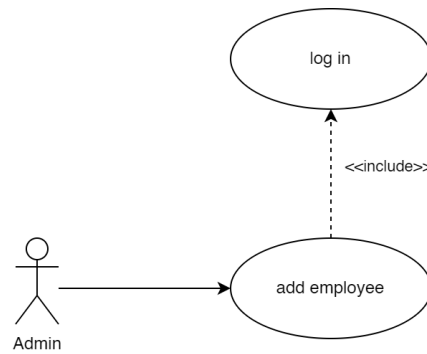
Property	Value
ID, Name	UC2.4, Delete user
Goal	The given user should be deleted
Initiator	Admin
Precondition	<ul style="list-style-type: none"> A user with admin privileges should be logged in.
Postcondition	<ul style="list-style-type: none"> The user is deleted from the database. The deleted user's data is returned.

STEP-BY-STEP DESCRIPTION

1. The actor submits a delete request to delete one of the users.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system checks that the actor is an admin.
 - c. The system verifies the input format of the request.
3. The system queries the database to delete the given user.
4. The system returns a response carrying a success code and the delete user data.



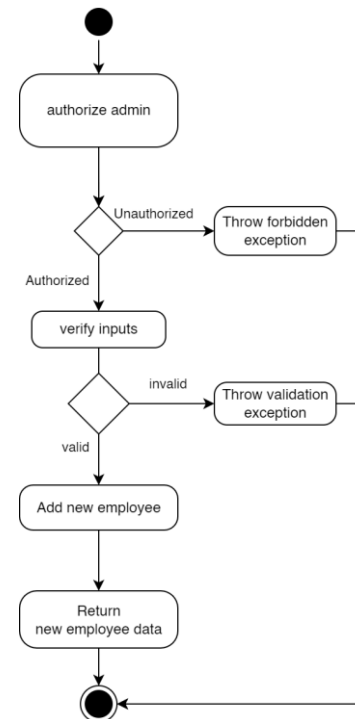
ADD EMPLOYEE

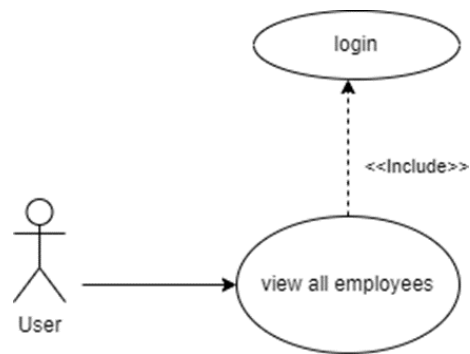


Property	Value
ID, Name	UC3.1, Add new Employee.
Goal	Employee should be added to the database.
Initiator	Admin.
Precondition	<ul style="list-style-type: none"> A user with admin privileges should be logged in.
Postcondition	<ul style="list-style-type: none"> An employee should be added to the database. The new employee's data should be returned.

STEP-BY-STEP DESCRIPTION

1. The actor sends a request to add a new employee.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system checks that the actor is an admin.
 - c. The system verifies the input format of the request.
3. The system adds the new employee to the database.
4. The system returns a response with a success code and the new employee's data

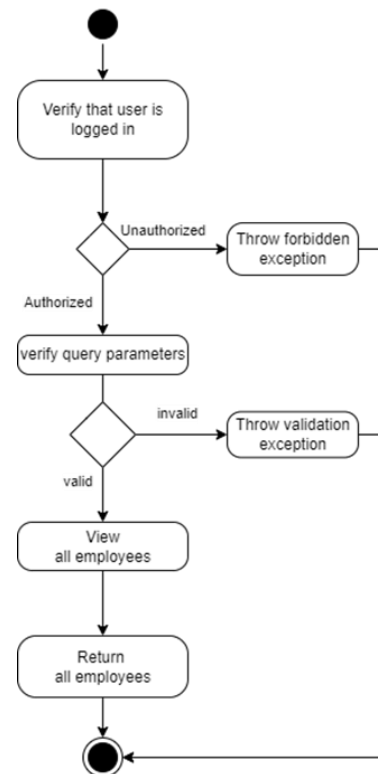




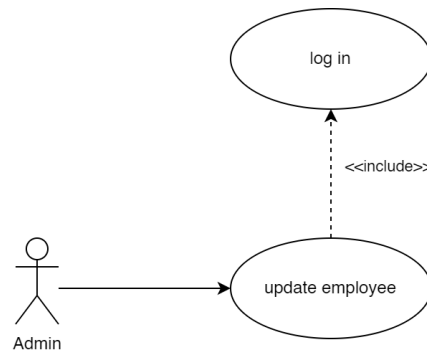
Property	Value
ID, Name	UC3.2, View employees
Goal	view data of employees for the user
Initiator	User
Precondition	<ul style="list-style-type: none"> The user should be logged in
Postcondition	<ul style="list-style-type: none"> Employees data is returned to the user if found

STEP-BY-STEP DESCRIPTION

1. The actor requests to view employees.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system verifies the query parameters of the request.
3. The system queries the database to get employees data with the given filter if exists.
4. The system returns a response carrying a success code and all employee's data.



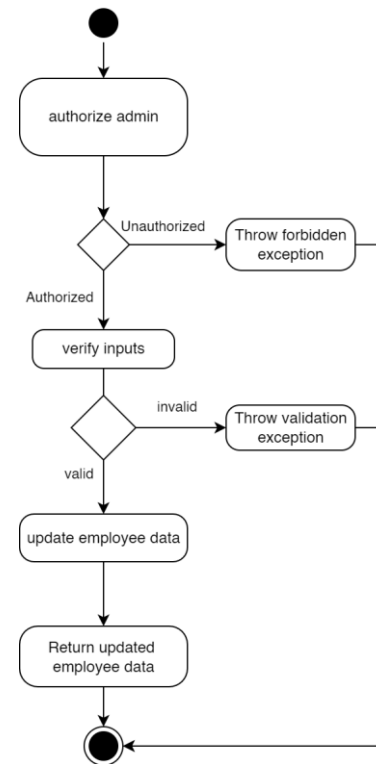
UPDATE EMPLOYEE



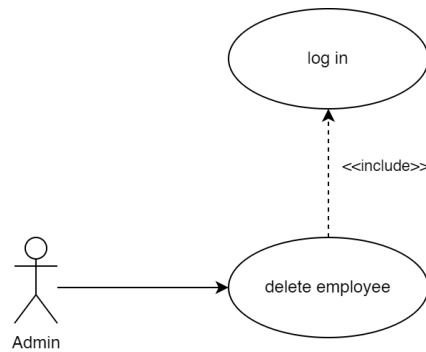
Property	Value
ID, Name	UC3.3, Update Employee.
Goal	Employee data will be updated in the database.
Initiator	Admin.
Precondition	<ul style="list-style-type: none"> A user with admin privileges should be logged in.
Postcondition	<ul style="list-style-type: none"> An employee should be added to the database. The updated employee's data should be returned.

STEP-BY-STEP DESCRIPTION

1. The actor sends a request to update an employee.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system checks that the actor is an admin.
 - c. The system verifies the input format of the request.
3. The system updates the employee's data in the database.
4. The system returns a response with a success code and the updated employee's data.



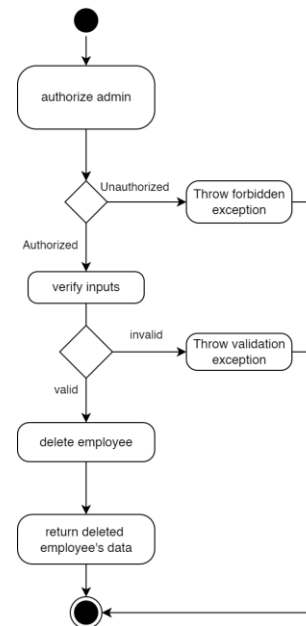
DELETE EMPLOYEE



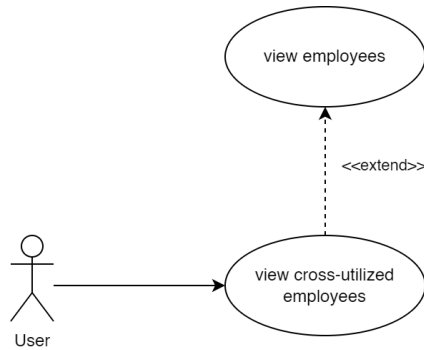
Property	Value
ID, Name	UC3.4, Delete Employee.
Goal	Employee data will be deleted.
Initiator	Admin.
Precondition	<ul style="list-style-type: none"> A user with admin privileges should be logged in.
Postcondition	<ul style="list-style-type: none"> The employee is deleted from the database. The deleted employee's data is returned

STEP-BY-STEP DESCRIPTION

1. The actor submits a delete request to delete one of the employees.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system checks that the actor is an admin.
 - c. The system verifies the input format of the request.
3. The system queries the database to delete the given employee.
4. The system returns a response carrying a success code and the deleted employee's data.



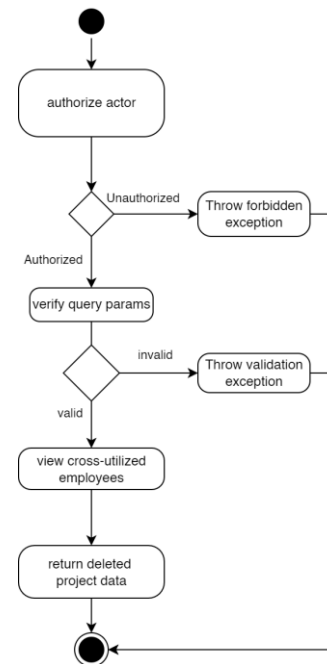
VIEW CROSS-UTILIZED EMPLOYEES



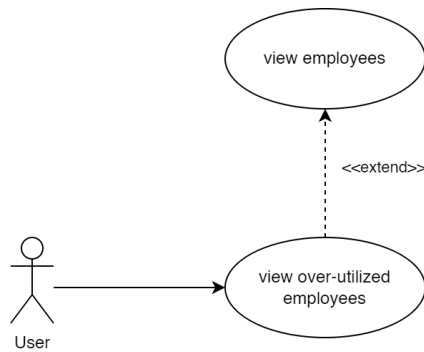
Property	Value
ID, Name	UC3.2.1, View cross-utilized employees
Goal	view data of cross-utilized employees
Initiator	User
Precondition	<ul style="list-style-type: none"> The user should be logged in
Postcondition	<ul style="list-style-type: none"> data of cross-utilized employees is returned to the user if found

STEP-BY-STEP DESCRIPTION

1. The actor requests to view cross-utilized employees.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system verifies the query parameters of the request.
3. The system queries the database to get data of cross-utilized employees with a given filter if exists.
4. The system returns a response carrying a success code and data of cross-utilized employees if found.



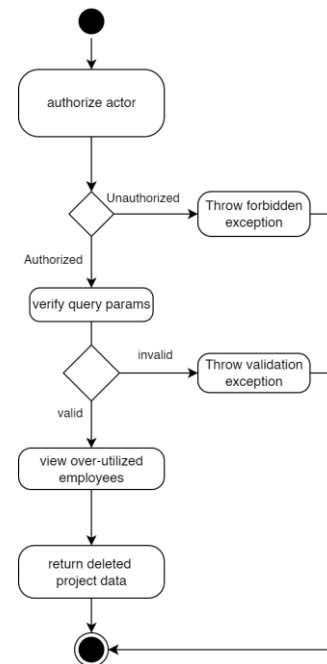
VIEW OVER-UTILIZED EMPLOYEES



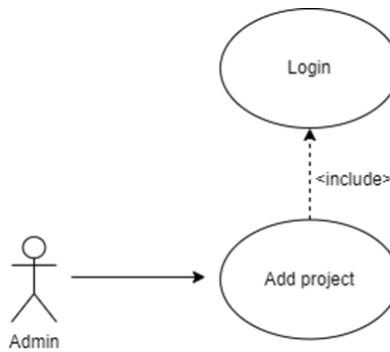
Property	Value
ID, Name	UC3.2.2, View over-utilized employees
Goal	view data of over-utilized employees
Initiator	User
Precondition	<ul style="list-style-type: none"> The user should be logged in
Postcondition	<ul style="list-style-type: none"> data of over-utilized employees is returned to the user if found

STEP-BY-STEP DESCRIPTION

1. The actor requests to view over-utilized employees.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system verifies the query parameters of the request.
3. The system queries the database to get data of over-utilized employees with a given filter if exists.
4. The system returns a response carrying a success code and data of over-utilized employees if found.



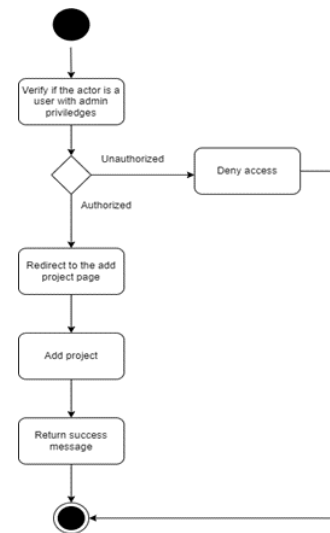
ADD PROJECT



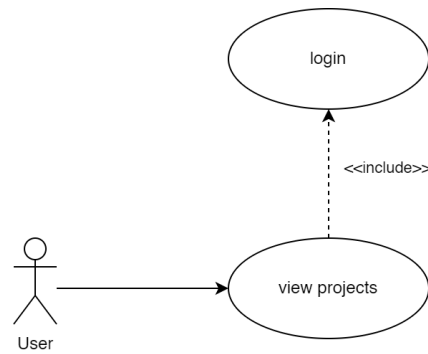
Property	Value
ID, Name	UC4.1, Add project
Goal	Adding a project to the project list
Initiator	Admin
Precondition	<ul style="list-style-type: none"> A user with admin privileges should be logged in.
Postcondition	<ul style="list-style-type: none"> A project should be added to the project list by the logged in admin.

STEP-BY-STEP DESCRIPTION

- The system checks that the actor is a user with admin privileges.
 - If the actor isn't an admin, the system denies their access.
 - If the actor is an admin, the system redirects them to the "add project page."
- The admin proceeds to add a project to the list and enters all the details related to the project.
- The project is then added to the database in a new row.
- A success message is sent to the admin if the project is successfully added to the database.



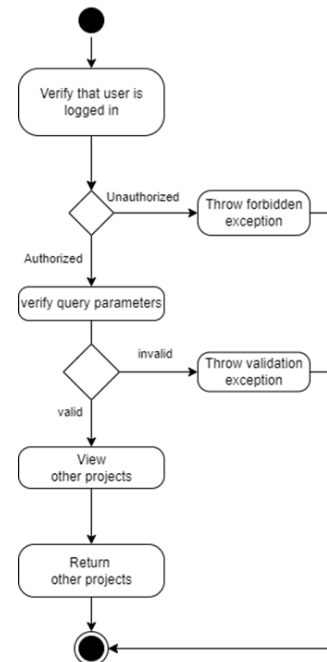
VIEW PROJECTS



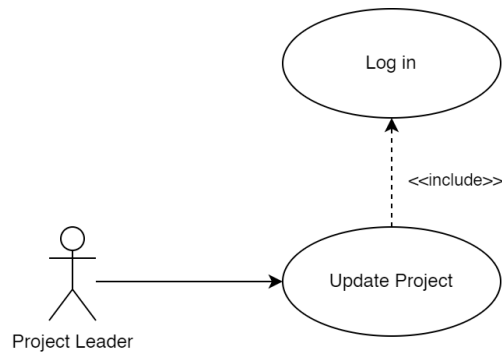
Property	Value
ID, Name	UC4.2, View projects
Goal	view data of projects for the user
Initiator	User
Precondition	<ul style="list-style-type: none"> The user should be logged in
Postcondition	<ul style="list-style-type: none"> Projects data is returned to the user if found

STEP-BY-STEP DESCRIPTION

1. The actor requests to view other projects request.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system verifies the query parameters of the request.
3. The system queries the database to get projects data.
4. The system returns a response carrying a success code and projects data.



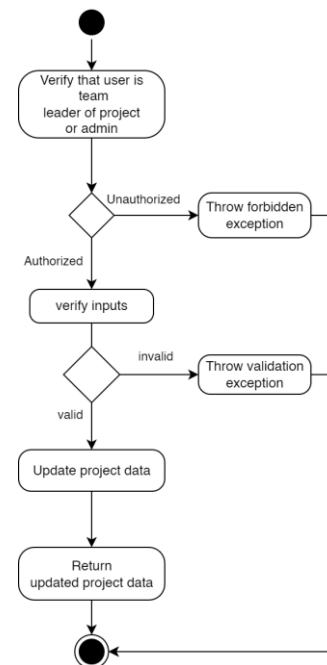
UPDATE PROJECT



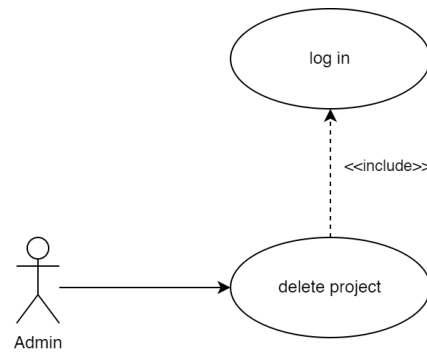
Property	Value
ID, Name	UC4.3, Update project
Goal	Inputted changes should apply to the project table.
Initiator	Project Leader, Admin
Precondition	<ul style="list-style-type: none"> A user with project leader or higher privileges should be logged in. The project leader must be the team leader of this project.
Postcondition	<ul style="list-style-type: none"> The project's properties will be updated. The new project data will be returned in response.

STEP-BY-STEP DESCRIPTION

1. The actor submits an update request to update one of his projects.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system checks that the actor is either the project leader assigned to this project or a user with higher privileges.
 - c. The system verifies the input format of the request.
3. The system queries the database to update the given project's properties with the new data.
4. The system returns a response carrying a success code and the updated project data.



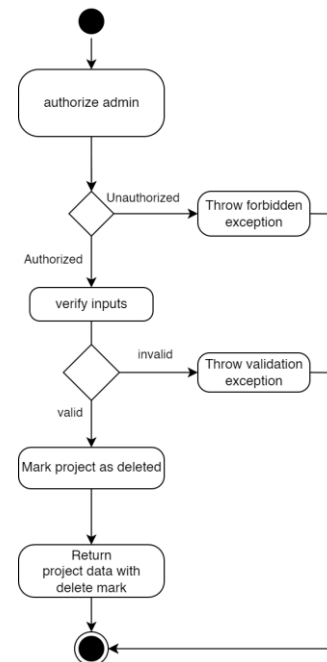
DELETE PROJECT



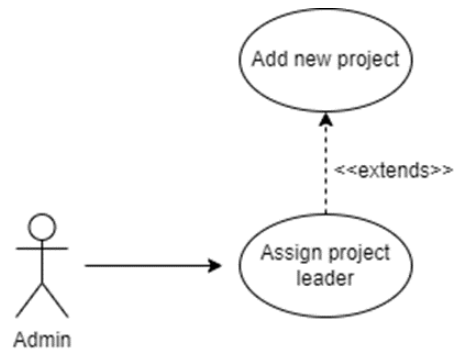
Property	Value
ID, Name	UC4.4, Delete project
Goal	The given project should be deleted
Initiator	Admin
Precondition	<ul style="list-style-type: none"> A user with admin privileges should be logged in.
Postcondition	<ul style="list-style-type: none"> The project is deleted from the database. The deleted project's data is returned

STEP-BY-STEP DESCRIPTION

1. The actor submits a delete request to delete one of the projects.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system checks that the actor is an admin.
 - c. The system verifies the input format of the request.
3. The system queries the database to delete the given project.
4. The system returns a response carrying a success code and the deleted project's data.



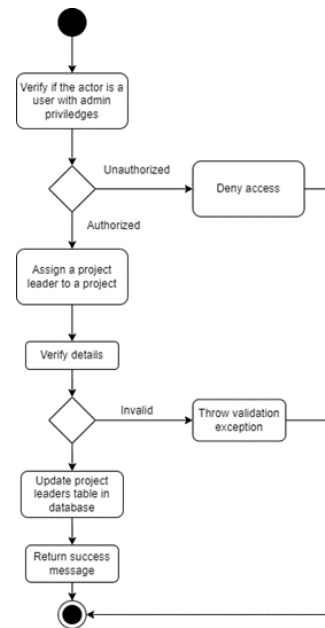
ASSIGN PROJECT LEADERS



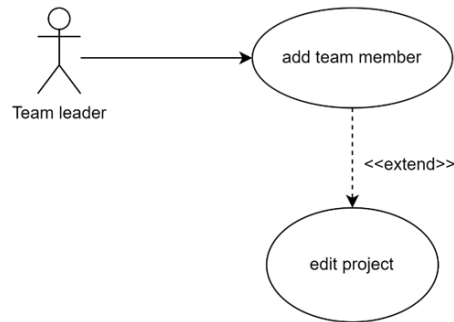
Property	Value
ID, Name	UC4.1.1, Assign project leaders.
Goal	A project leader should be assigned to a project.
Initiator	Admin.
Precondition	<ul style="list-style-type: none"> A user with admin privileges should be logged in.
Postcondition	<ul style="list-style-type: none"> A project leader should be assigned to a project.

STEP-BY-STEP DESCRIPTION

- The system checks that the actor is a user with admin privileges.
 - If the actor isn't an admin, the system denies their access.
 - If the actor is an admin, the system redirects them to the "assign project leaders page."
- The admin proceeds to assign a project leader to a project and enters all the details required.
- The project leader is then added to the list of project leaders in the database table.
- A success message is sent to the admin if the project leader is successfully added to the database.



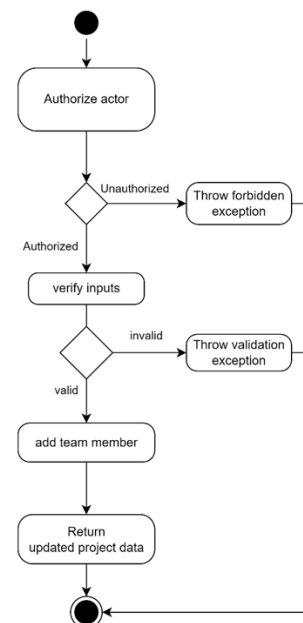
ADD EMPLOYEES TO PROJECT TEAMS



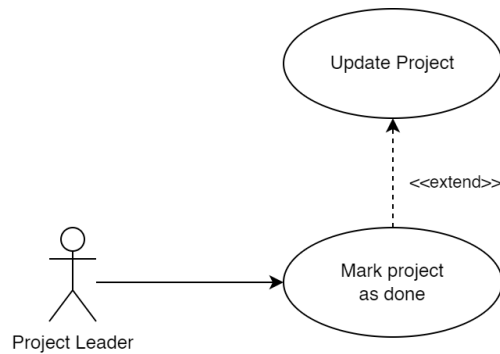
Property	Value
ID, Name	UC4.3.1, Add Employees To Project Teams
Extends	Update Project
Goal	An employee is added to the team of the given project.
Initiator	Project Leader
Precondition	<ul style="list-style-type: none"> • A user with project leader should be logged in. • The project leader must be the team leader of the given project.
Postcondition	<ul style="list-style-type: none"> • The employee is added to the given project's team. • The project team data will be returned in response.

STEP-BY-STEP DESCRIPTION

1. The project leader searches for employees using search filters
2. The system shows the team leader a list of employees based on the filters
3. The project leader submits an update request to add a team member to one of the projects that they're leading. By selecting an employee from the list of employees with optional filters.
4. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system checks that the inputted project's leader is the logged in actor.
 - c. The system verifies the input format of the request.
5. The system queries the database to update the given project's team members and add the new member
6. The system returns a response carrying a success code and the updated project data.



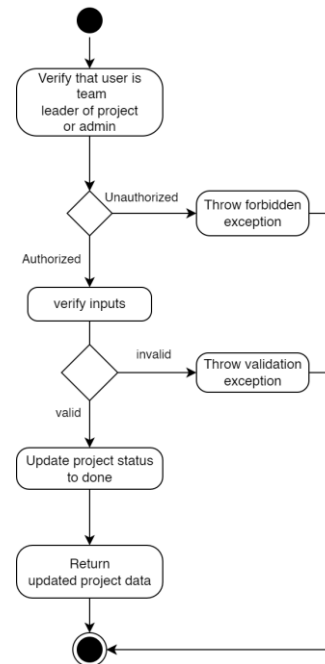
MARK PROJECT AS DONE



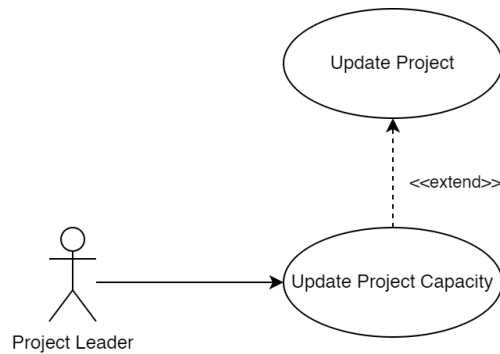
Property	Value
ID, Name	UC4.3.2, Mark project as done
Extends	Update project
Goal	A project's status is updated to done.
Initiator	Project Leader
Precondition	<ul style="list-style-type: none"> A user with project leader or higher privileges should be logged in. A project leader must be the team leader of this project.
Postcondition	<ul style="list-style-type: none"> The project's status will be updated. The new project's data will be returned in response.

STEP-BY-STEP DESCRIPTION

1. The project leader submits an update request to update the status field of one of the projects that they're leading.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system checks that the inputted project's leader is the logged in actor.
 - c. The system verifies the input format of the request.
3. The system queries the database to update the given project's status with the done status that indicates that this project does not currently utilize any employees.
4. The system returns a response carrying a success code and the updated project data.



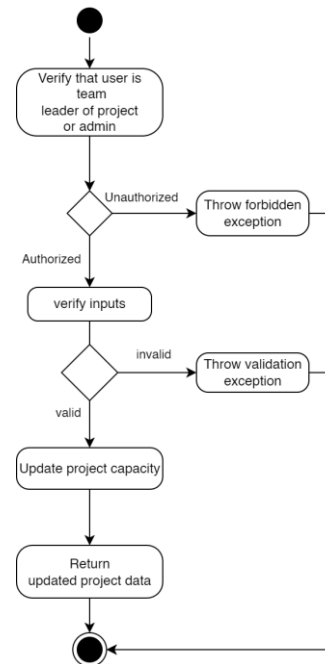
UPDATE PROJECT CAPACITY



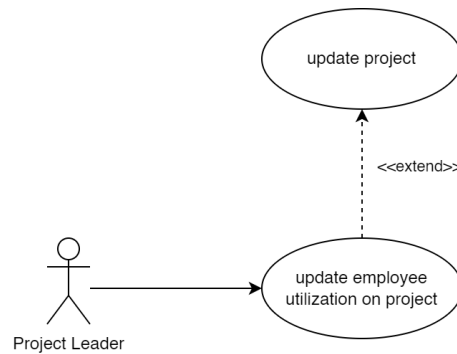
Property	Value
ID, Name	UC4.3.3, Update project capacity
Extends	Update project
Goal	A project's status is updated to done.
Initiator	Project Leader
Precondition	<ul style="list-style-type: none"> A user with project leader or higher privileges should be logged in. A project leader must be the team leader of this project.
Postcondition	<ul style="list-style-type: none"> The project's capacity is updated. The new project data will be returned in response.

STEP-BY-STEP DESCRIPTION

- The project leader submits an update request to update the capacity field of one of the projects associated with one of his projects.
- The system runs validation checks, upon the failure of any, throws an exception.
 - The system checks the actor's login validity.
 - The system checks that the inputted project's leader is the logged in actor.
 - The system verifies the input format of the request.
- The system queries the database to update the given project's capacity with the inputted capacity that indicates that this project needs a given number of employees on its team.
- The system returns a response carrying a success code and the updated project data.



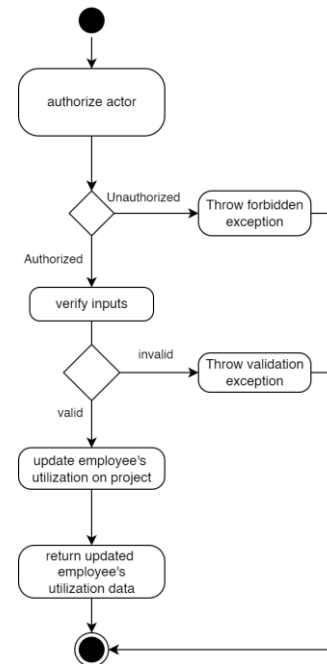
UPDATE EMPLOYEE UTILIZATION ON PROJECT

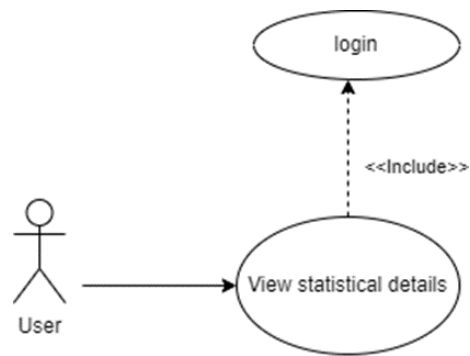


Property	Value
ID, Name	UC4.3.4, Update project capacity
Extends	Update project
Goal	A project's status is updated to done.
Initiator	Project Leader
Precondition	<ul style="list-style-type: none"> A user with project leader or higher privileges should be logged in. A project leader must be the team leader of this project.
Postcondition	<ul style="list-style-type: none"> The employee's utilization on the given project is updated. The new employee's utilization data will be returned in response.

STEP-BY-STEP DESCRIPTION

1. The project leader submits an update request to update the utilization field of one of the employees on one of his projects.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system checks that the inputted project's leader is the logged in actor.
 - c. The system verifies the input format of the request.
3. The system queries the database to update the utilization field of the employee's utilization on the project.
4. The system returns a response carrying a success code and the updated employee's utilization data.

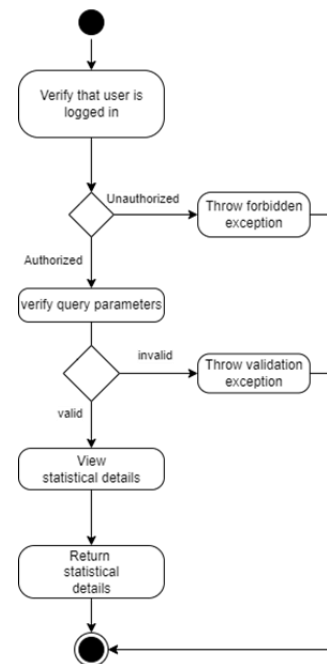




Property	Value
ID, Name	UC5, View statistical details
Goal	views statistical data of sorts for the user
Initiator	User
Precondition	<ul style="list-style-type: none"> The user should be logged in
Postcondition	<ul style="list-style-type: none"> The system will aggregate statistical analysis and send it to user

STEP-BY-STEP DESCRIPTION

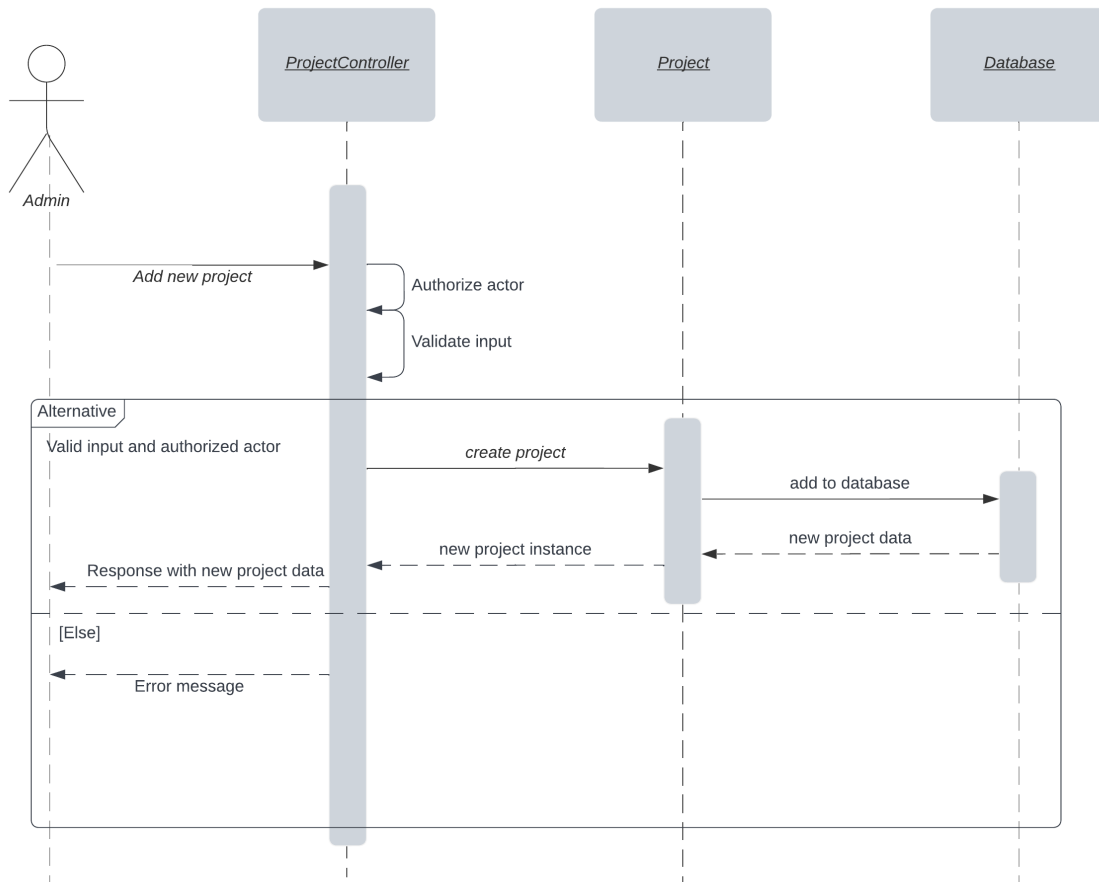
1. The actor submits a view statistical details request.
2. The system runs validation checks, upon the failure of any, throws an exception.
 - a. The system checks the actor's login validity.
 - b. The system verifies the query parameters of the request.
3. The system queries the database to get statistics data.
4. The system returns a response carrying a success code and the statistics data.



SEQUENCE DIAGRAMS

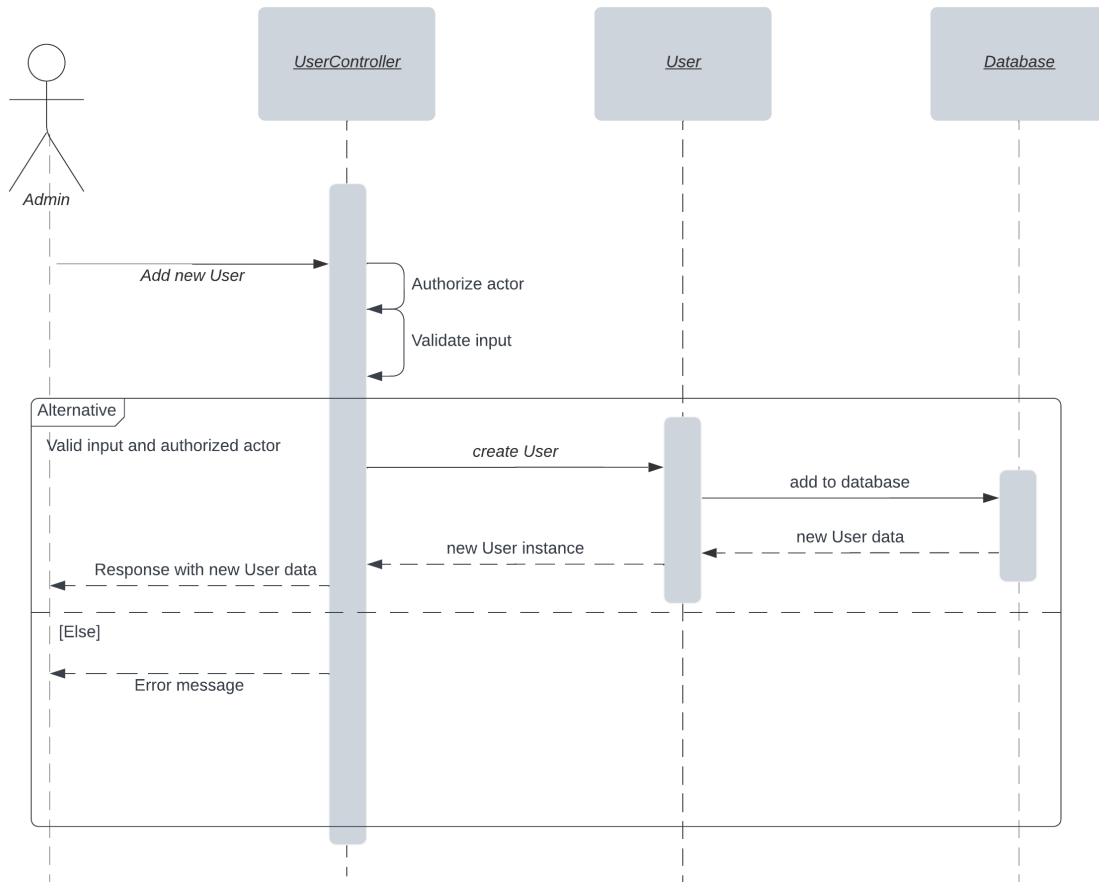
ADD NEW PROJECT

- See use case: UC4.1



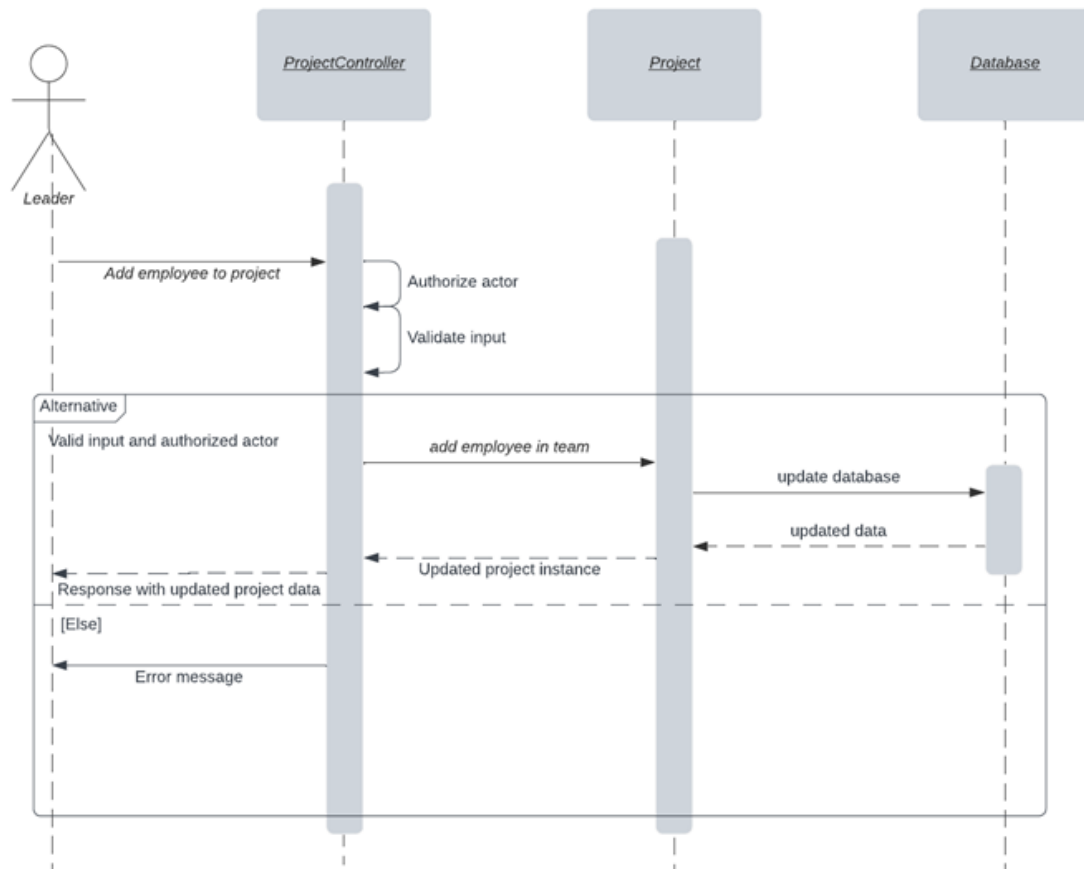
ADD NEW USER

- See use case: UC2.1



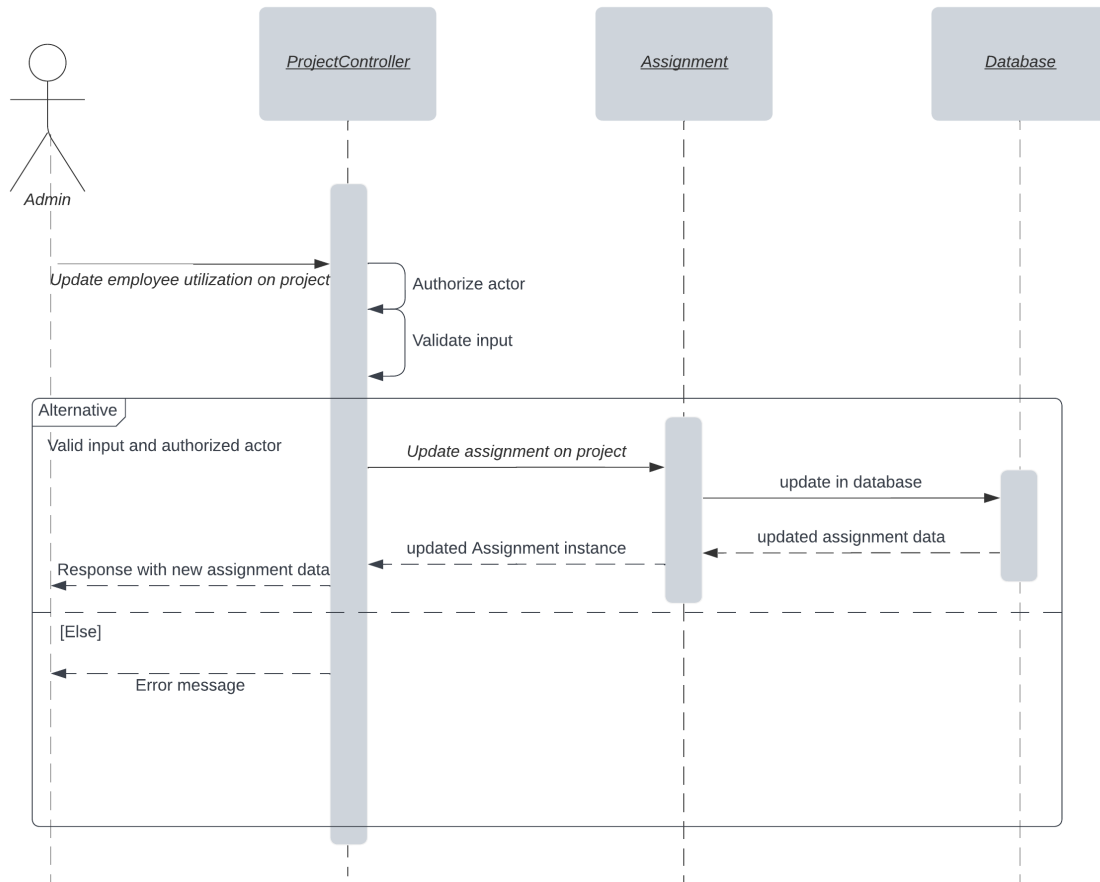
ASSIGN EMPLOYEE TO PROJECT

- See use case: UC4.3.1



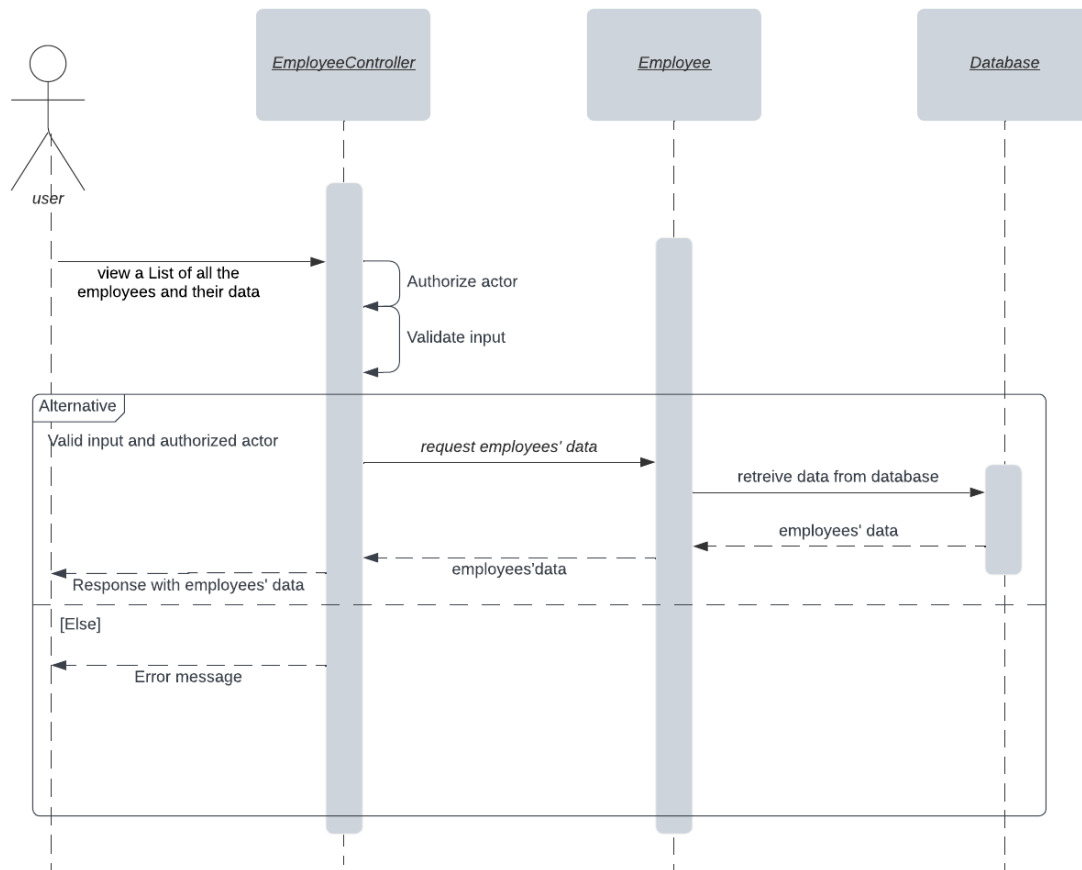
UPDATE EMPLOYEE UTILIZATION ON PROJECT

- See use case: UC4.3.4



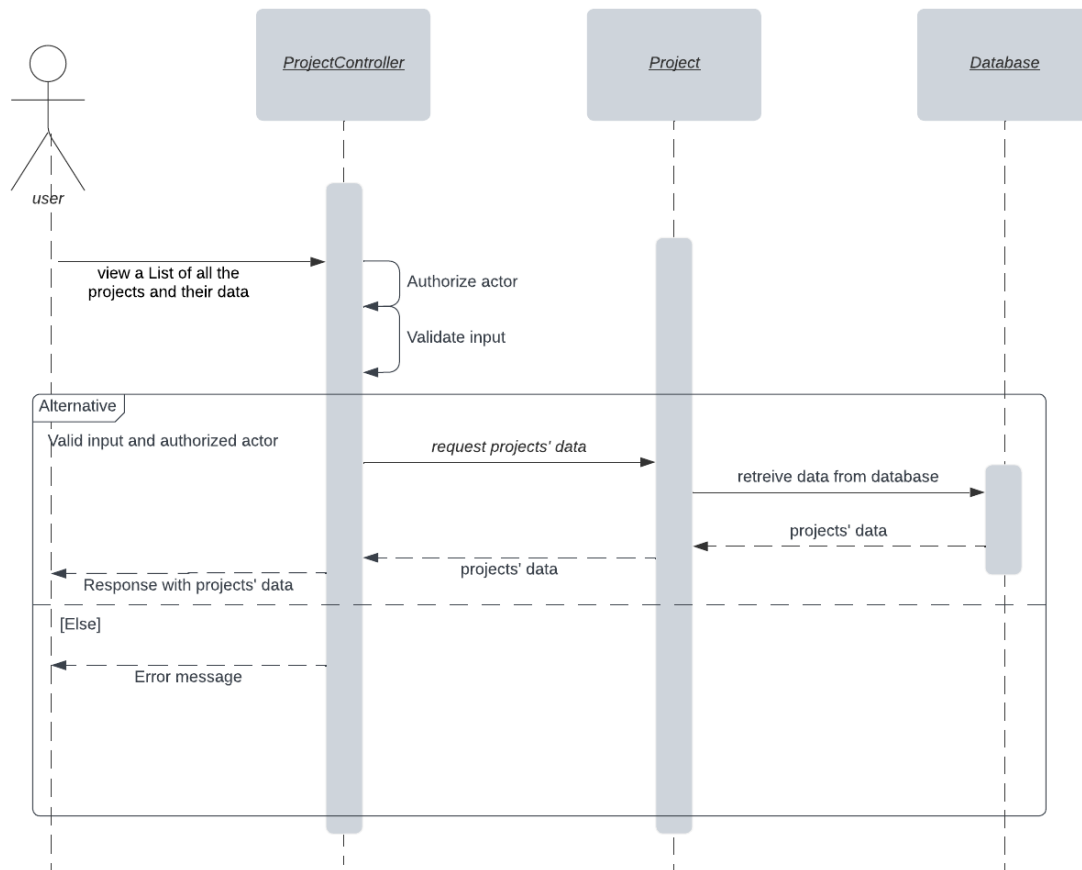
VIEW EMPLOYEES

- See use case: UC3.2



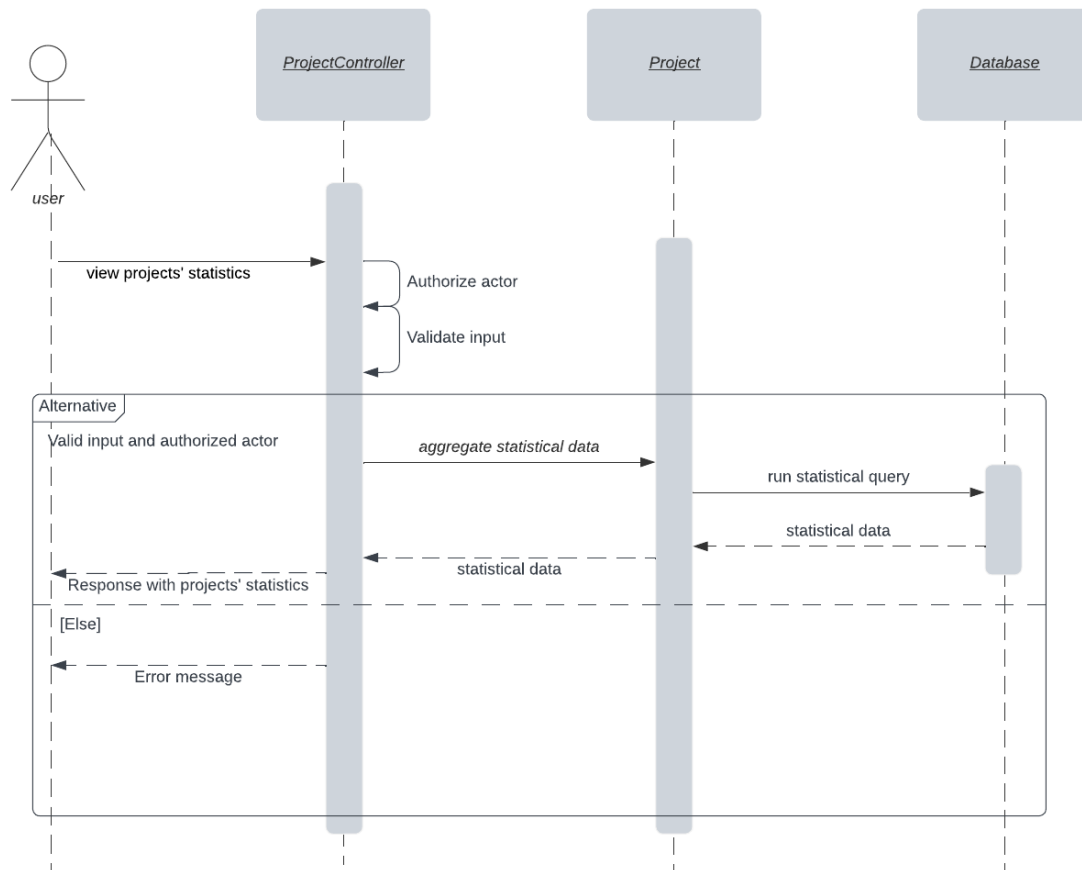
VIEW PROJECTS

- See use case: UC4.2



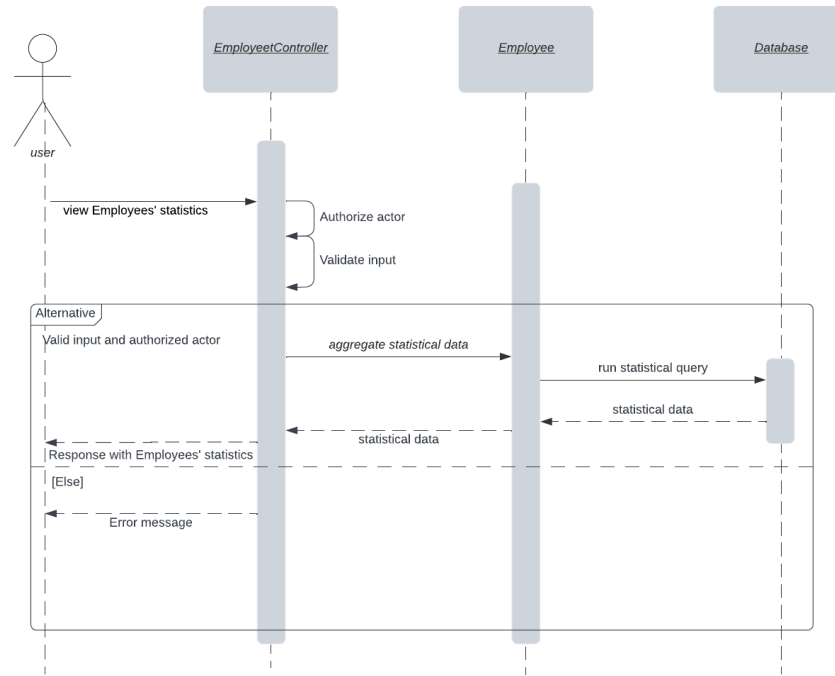
VIEW PROJECT STATISTICS

- See use case: UC5



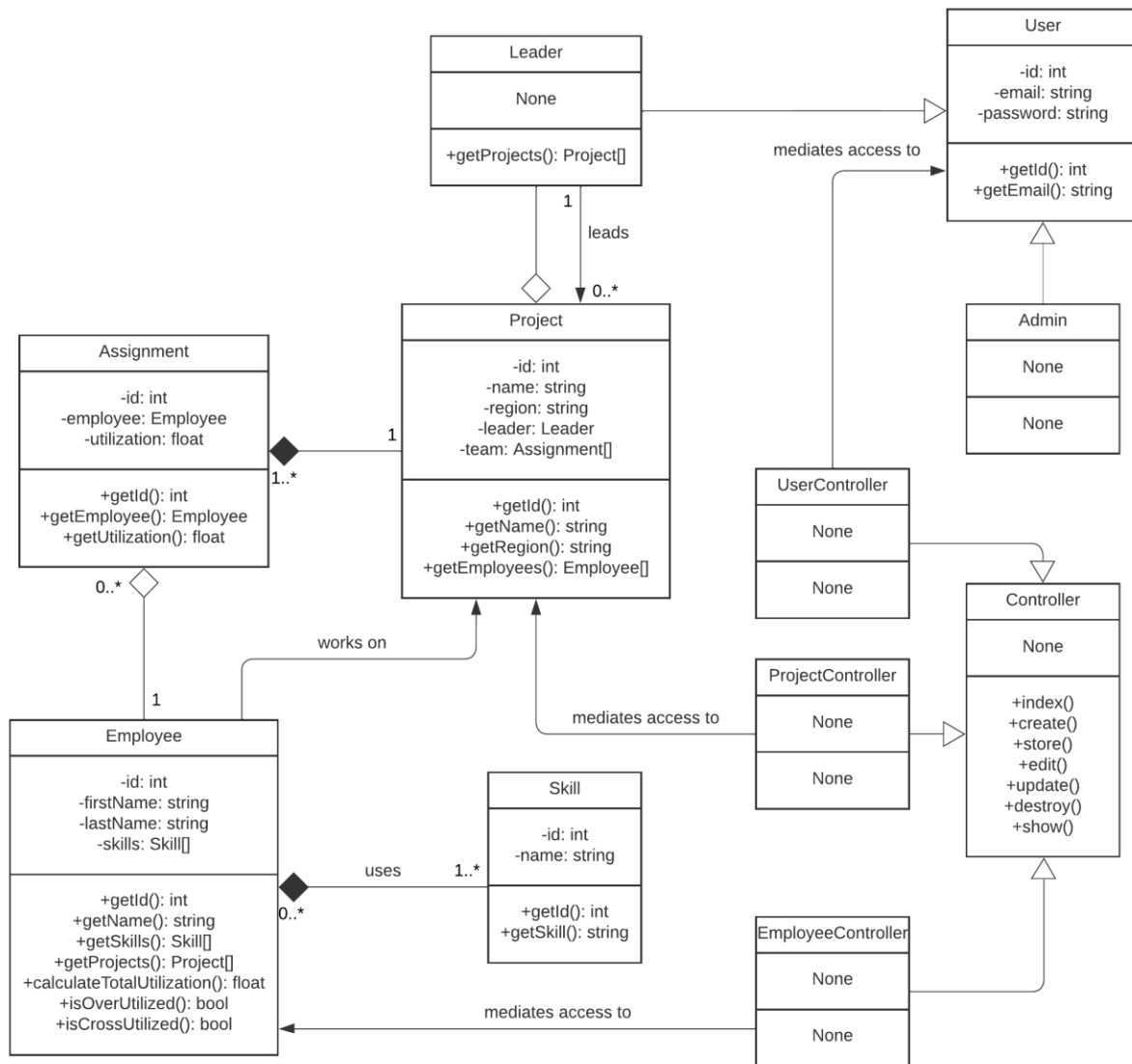
VIEW EMPLOYEE STATISTICS

- See use case: UC5

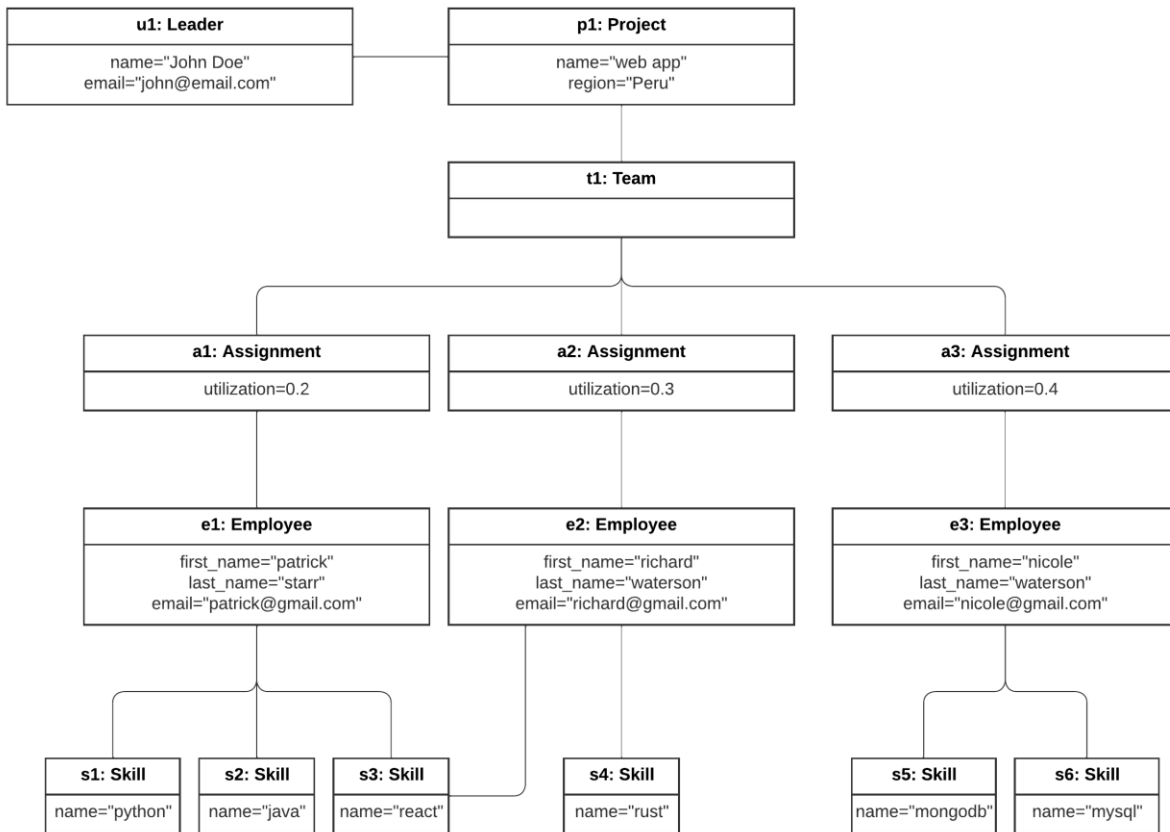


DESIGNS

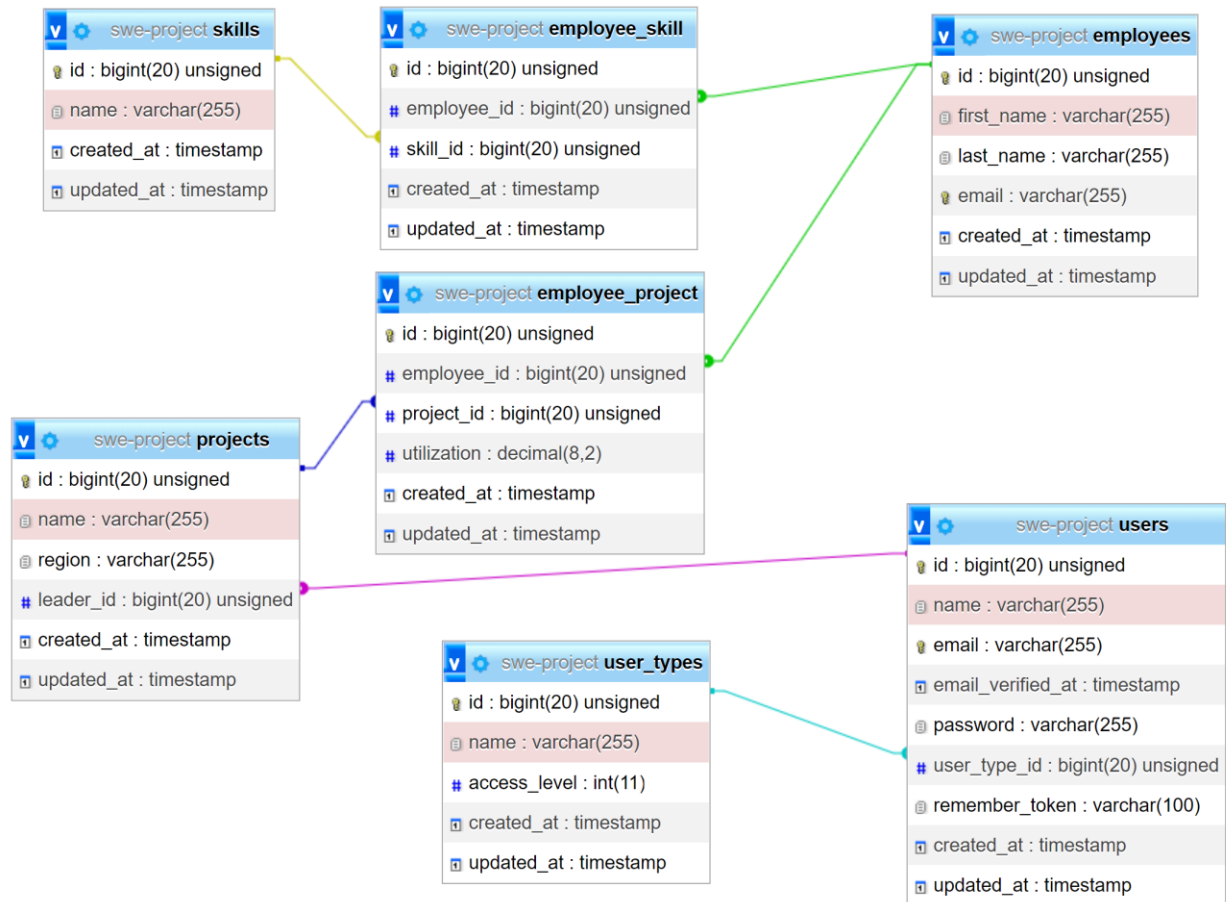
CLASS DIAGRAM



OBJECT DIAGRAM



DATABASE PHYSICAL DESIGN



1- ADMINS CAN MANAGE USERS' DATA.

- Add a user:

The screenshot shows the 'Add a user' form within the PM application. The interface includes a sidebar with icons for home, settings, tasks, and users. The main content area is titled 'Users welcome | Roderick' and contains the 'Add a user' form. The form has the following fields:

- user's name *: Heller-Whit
- user's email *: heller@gmail.com
- user's password *:
- user type: admin (dropdown menu)

Below the form is an 'ADD USER' button. The bottom left corner of the application shows the version number '1.27.0.0 1.8000'.

- Edit user:

The screenshot shows the 'edit a user' form within the PM application. The interface is similar to the 'Add a user' form, but with different field values and additional buttons. The form has the following fields:

- user's name *: Roderick
- user's email *: admin@example.com
- user's password *:

Below the form are two buttons: 'EDIT USER' and 'DELETE USER'. The bottom left corner of the application shows the version number '1.27.0.0 1.8000'.

- View user:

Users

welcome | Roderick

↗

↶

Search

+ Create user

Name	Email	Position	Details
Mann	test@example.com	leader	Show
Roderick	admin@example.com	admin	Show
Miss Rachael Walsh V	mrz.pascale@example.com	leader	Show
Prof. Emmanuelle Dach Sr.	altenwerth.herta@example.net	leader	Show
Mr. Emilio Green	fgibson@example.org	leader	Show
Cordelia Strosin	mozelle56@example.net	leader	Show
Ms. Julie Schullist	ftorp@example.com	leader	Show
Prof. Connor Mann	pschimmel@example.org	leader	Show
Mr. Derek Fendall	duff@scots.com	leader	Show

User Details

welcome | Roderick

↗

↶

Mann Details

Name

Mann

Email

test@example.com

User type

leader

Projects

Mosciski, Kiehn and Hettinger >

Mraz, Gutmann and Langworth >

McKenzie-Klocko >

127.0.0.1:8000

2- ADMINS CAN MANAGE PROJECTS' DATA.

- Add a project:

The screenshot shows the 'Create a project' form in the PM application. The form is titled 'Create a project' and contains three input fields: 'Project name *', 'team leader', and 'region'. Each field has a corresponding icon (a document for name, a person for leader, and a location pin for region). Below the fields is a purple button labeled 'ADD PROJECT'. The form is set against a light blue background with a sidebar on the left containing icons for home, search, and users. The top navigation bar shows 'Projects welcome | Admin' and a refresh icon.

- Edit a project:

The screenshot shows the 'Edit the project' form in the PM application. The form is titled 'Edit the project' and contains three input fields: 'Project name *', 'team leader', and 'region'. Each field has a corresponding icon (a document for name, a person for leader, and a location pin for region). Below the fields are two purple buttons labeled 'EDIT PROJECT' and 'DELETE PROJECT'. The form is set against a light blue background with a sidebar on the left containing icons for home, search, and users. The top navigation bar shows 'Projects welcome | Admin' and a refresh icon.

- View a project:

The screenshot shows the 'View a project' page in the PM application. The page features a search bar, a 'Skills' dropdown, a 'Regions' dropdown, and a '+ Create project' button. Below these elements is a table with three columns: 'Name', 'Leader', and 'Details'. The table contains two rows of data: 'CPP project' and 'Web app', both led by 'Dr. Easter Langworth'. Each row has a 'Show' link in the 'Details' column. The page is set against a light blue background with a sidebar on the left containing icons for home, search, and users. The top navigation bar shows 'Projects welcome | Admin' and a refresh icon.

Name	Leader	Details
CPP project	Dr. Easter Langworth	Show
Web app	Dr. Easter Langworth	Show

PM

Project Details welcome | Admin ↗

↶

🏠
🔄
📋
👥

CPP project Details ✎

Name
CPP project

Region
Cairo

Leader
Dr. Easter Langworth

Team
None

+ add team member

3- TEAM LEADERS CAN ADD OR REMOVE EMPLOYEES TO THEIR TEAM.

PM

Add team member welcome | Admin ↗

↶

🏠
🔄
📋
👥

add a new team member

👤 Team member name ▼

📅 project capacity

SAVE CHANGES

Page 45 | 49

4- TEAM LEADERS CAN ADD THE REQUIRED EMPLOYEE CAPACITY TO THEIR TEAM MEMBERS.

PM

Add team member welcome | Admin ↩

add a new team member

Team member name

project capacity

SAVE CHANGES

5- TEAM LEADERS AND ADMINS CAN VIEW A LIST OF ALL THE PROJECTS AND THEIR DATA

PM

Projects welcome | Admin ↩

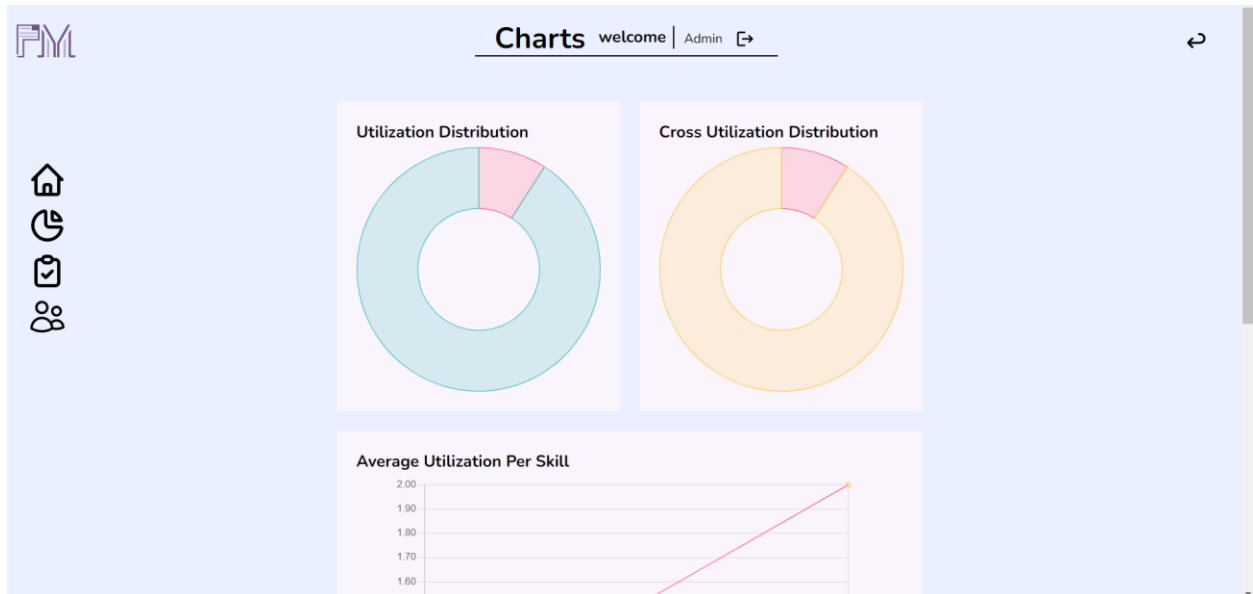
Search

Skills Regions

+ Create project

Name	Leader	Details
CPP project	Dr. Easter Langworth	Show
Web app	Dr. Easter Langworth	Show

6- ADMINS AND TEAM LEADERS CAN VIEW STATISTICAL DETAILS



7- ADMINS CAN MANAGE EMPLOYEES

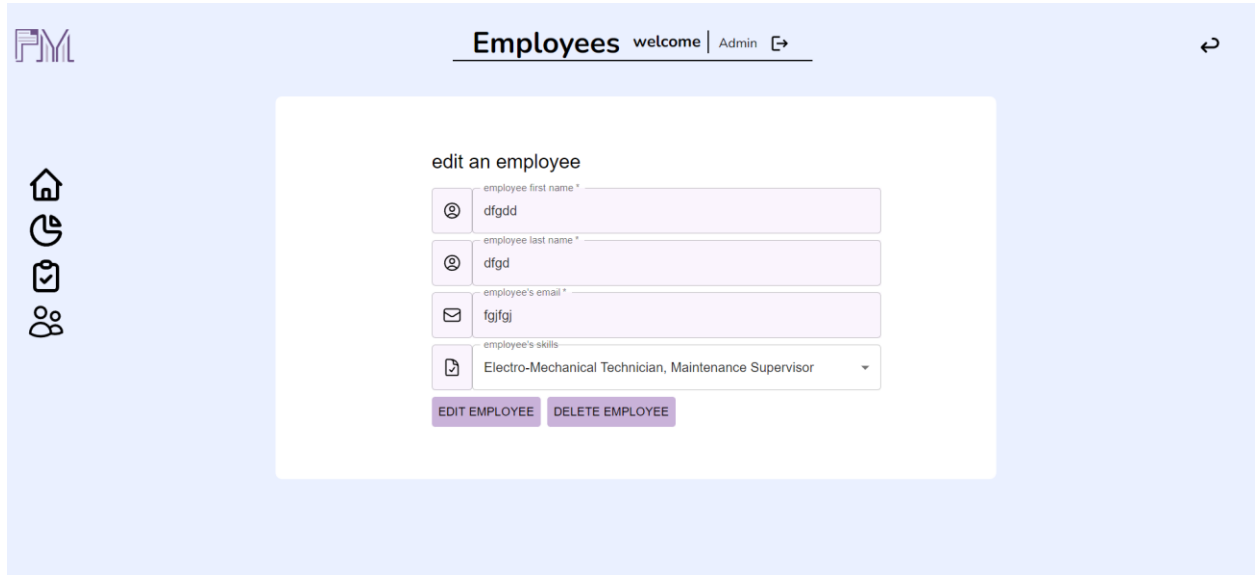
1- Add employee:

The screenshot displays a dashboard titled "Employees" with a user profile "welcome | Admin" and a refresh icon. The dashboard features a form titled "Add an employee" with the following fields:

- employee first name *
- employee last name *
- employee's email *
- Skills (dropdown menu)

Below the form is a purple button labeled "ADD EMPLOYEE". A sidebar on the left contains icons for home, charts, tasks, and users.

2- Edit employee:



PM Employees welcome | Admin ↗

edit an employee

employee first name * dfgdd

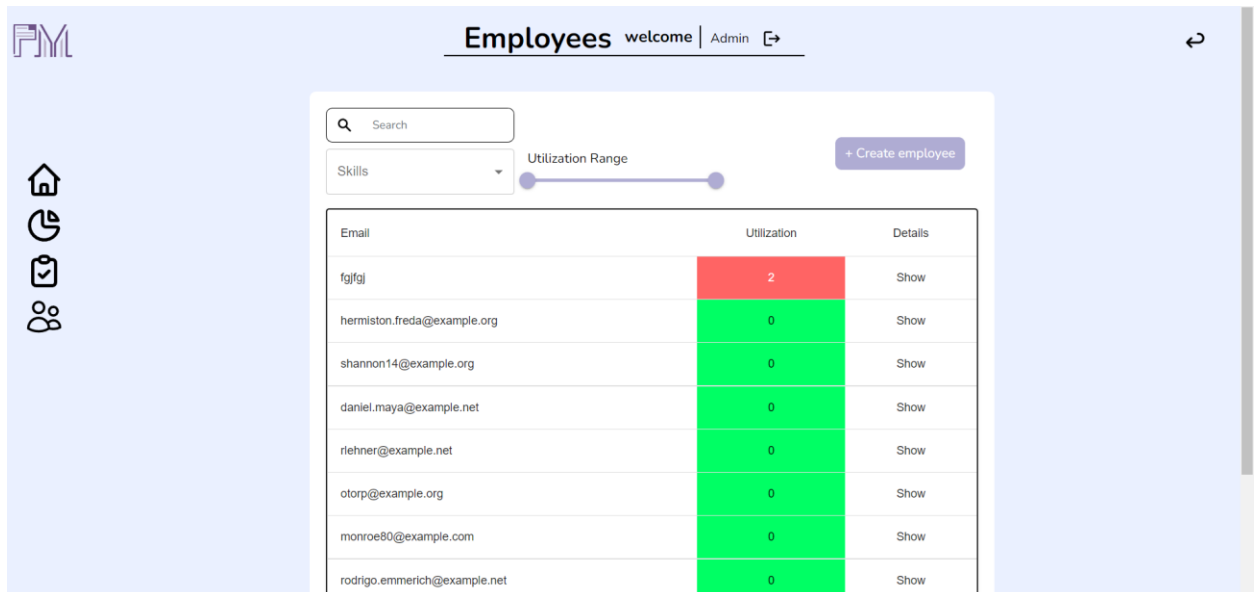
employee last name * dfgd

employee's email * fgjfgj

employee's skills Electro-Mechanical Technician, Maintenance Supervisor

EDIT EMPLOYEE DELETE EMPLOYEE

3- View employee:



PM Employees welcome | Admin ↗

Search

Skills Utilization Range + Create employee

Email	Utilization	Details
fgjfgj	2	Show
hermiston.freda@example.org	0	Show
shannon14@example.org	0	Show
daniel.maya@example.net	0	Show
riehner@example.net	0	Show
otorp@example.org	0	Show
monroe80@example.com	0	Show
rodrigo.emmerich@example.net	0	Show



dfgdd Details



Name

dfgdd dfgd

Email

fgjfgj

Skills

Electro-Mechanical Technician

Maintenance Supervisor

Project

Web app



CPP project



Utilization

