Project Report

Malai Alexandru [304189]
George-Eduard Andronache [304460]
Marwa Nizar Alsaduni [304221]
Adrian-Gabriel Vaitis [304486]
Dmitrii Pinzari [304618]

Supervisors:

Steffen Vissing Andersen
Astrid Hanghøj
Klaus Bøgestrand

Via University College

Software Technology Engineering
I Semester
18/12/2020

Table of content

Abstract

The aim of our project is to provide support to “Colour-IT” company in order to help manage the projects and monitor the progress, which includes - productivity of the employees, division of the tasks, time spent working and processes that help in achieving the goal.

The program has been developed using Java programming language, Scene builder and JavaFX , following the Waterfall method. The website has been made using HTML, CSS, and JavaScript.

The result of the project is a managing projects system that helps to manage projects and reduce the hustle with easy-to-use structure, explicit documentation, and a user guide. Furthermore, besides Java application, there is a website presenting functionality of the project, the features, gives the possibility to contact the team and gives a main idea about software possibilities.

1. Introduction

A company “Colour IT” has reached out to the 1. Semester students of Software Engineering with a task to develop and implement an IT project management system. In addition, the company has asked to develop a website to complement the above mentioned system (Mr. Colour, 2020).

To start off, it is important to understand what exactly is the project management system, why is it important, why there is a need for such, and what are the main challenges in creating one.

Project management system is the tool that helps organizations in planning, organizing, tracking, scheduling and improving the work. It can assist in preparing for difficult projects and also improve productivity of the company as a whole (Vartika Kashyap, 2020).

Due to the given time restrictions, as well as the fact that the project is carried out by the first semester students with little to no programming experience, the project is not going to include a complicated system with a database. This being said, the system that the team will develop is a basic and simple single user system (ColorIT - The interview).

In today's world many organizations rely on online IT solutions in order to get better results at work and satisfy the customers. The main reasons why organizations use project management tools are (Wrike, 2020):

1) **Visibility.** Enables the company and their customers to see the overview of the projects, view the progress and timeline of accomplished work.
2) **Accountability**. Sets clear division of tasks and responsibilities for employees, thus avoiding confusions in splitting the tasks, makes a clear overview of scheduled work

and deadlines, and helps with timely updates to all levels in the organization, also including the customers.

3) **Organization.** Keeps everything in one place and is secure. Informs about the details and if the updates were made, thus keeping every team member and also the customer on board.

However, due to the high costs that could arise during the implementation of such systems, not every company is able to pay for the software management systems and their maintenance. That can lead to the companies refusing to use the mentioned systems, which of course shows on a company's overall results, because it becomes harder to compete in the market with other companies which do use it (Wrike, 2020). Another challenge worth mentioning is that there is no guarantee that a specific management system will improve the overall performance of a firm and its customers satisfaction. This happens due to the fact that it takes time to get comfortable with the softwares interface and there is also no guarantee that it might be well-designed or suited for the specific needs of a specific firm. Therefore it is very important to properly define the requirements of such a system, which can be hard if the customer is unsure, or if there is a misunderstanding between the firm and the customer (Alp Group, 2019).

2. Requirements

2.1 Functional Requirements

Critical priority:

1. As a project creator, I want to create a new project ,to start the project work;
2. As scrum master, I want to be able to check every project status such that I want to keep track;
3. As scrum master, I want to be able to assign tasks to each team member so that I know every member is responsible for a task;
4. As a scrum master, I want to add and remove requirements so that I can keep the track of the project;
5. As a scrum master, I want to add and remove tasks so that I can sort them out whether they are good or not;

6. As a team member I want to search about a specific task by id or name ,to manage it or just show it.
7. As scrum master, I want to handle the process reports by searching the deadline date and if there are all the methods when the requirements are done such that I can check if all is going right;
8. As a project creator, I want to be able to manage projects.

High priority:

9. As a team member, I want to be able to view projects such that I can have an overview at them;
10. As a customer I want to search for a project by name or id , so I can see all the project details and task status . .
11. As a scrum master, I want to edit total hours for a task so that everyone keep tracking the work progress;
12. As a product owner I want to edit a type of requirement (functional, not functional, project requirement), so that they will be sorted out.
13. As a scrum master, I want to add deadlines for existing tasks.
14. As a product owner,I want to approve the requirement when it's done, such that I can see all the projects that are not done yet.

Low priority:

15. As a scrum master, I want to edit available tasks in case something changes in a project or in the case that customer changes his preferences;
16. As a customer I want to have access to projects, tasks, time spent on them, so I can see the progress and how long it took to be ready;

2.2 Non-functional requirements:

17. The system must be compatible with each device that is running Windows 7, Windows 8, Windows 8.1 and Windows 10;

18. As a scrum master, I want to assign tasks to team members so that they can handle the projects.

3. Analysis

The purpose of the analysis section is to outline an understanding of the problem domain and specifically what the stakeholders want.

3.1 Use case diagram and description

The use case diagram is based on the requirements mentioned above. Project managing system must manage IT projects. A Project creator to create a project with team members, one being a Scrum master and one being a Product owner. A project has a list of requirements, some are functional requirements formulated as user stories with the 3 parts who, what and why.

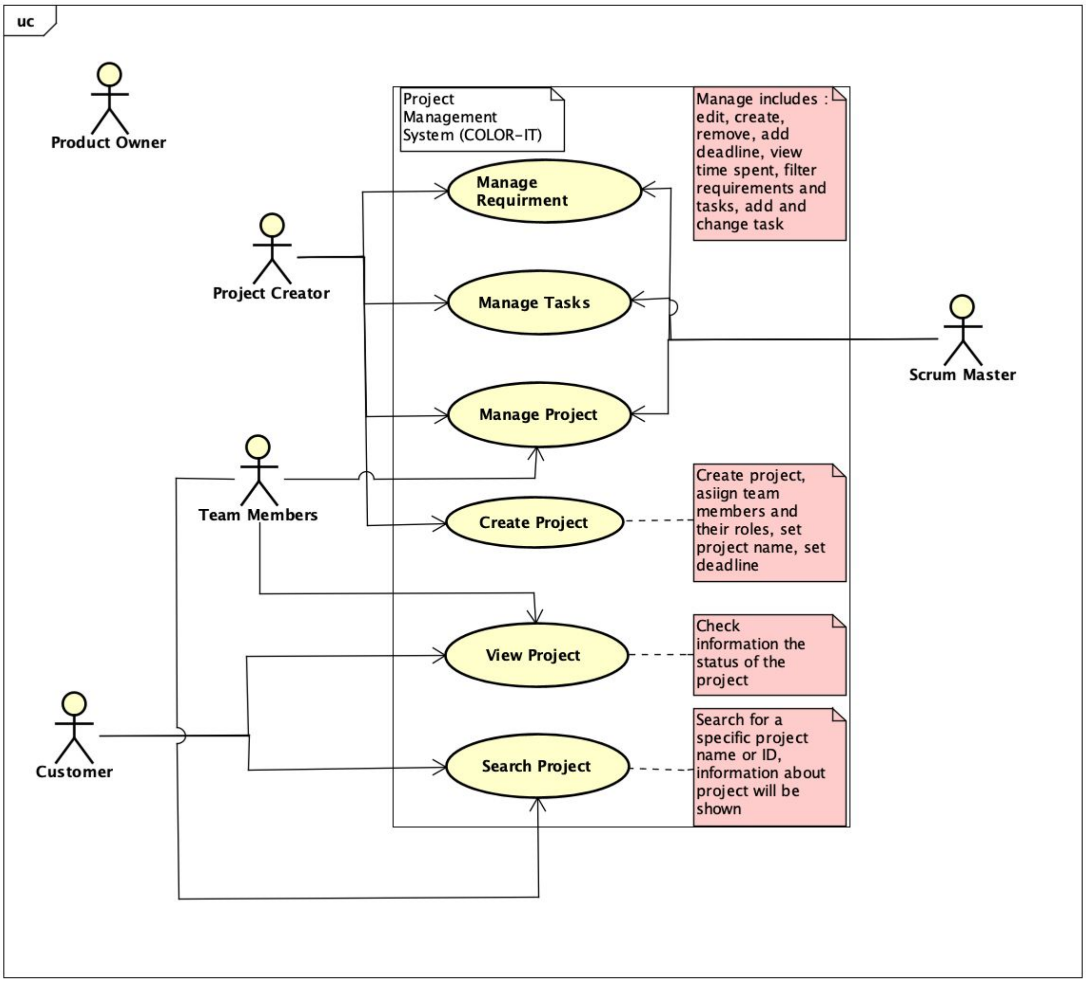
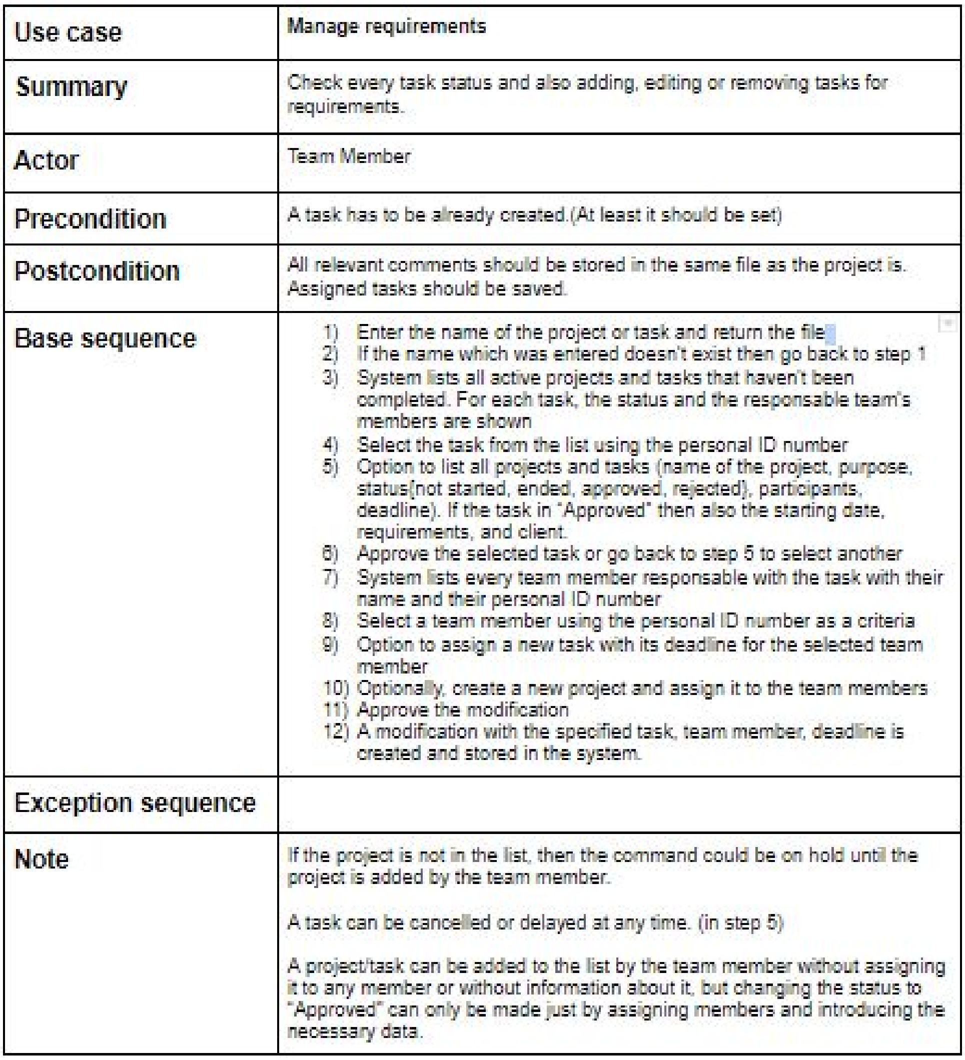

uc
Product Owner
Project Creator
Team Members
Customer
Scrum Master
Project Management System (COLOR-IT)
Manage Requirment
Manage Tasks
Manage Project
Create Project
View Project
Search Project
Manage includes : edit, create, remove, add deadline, view time spent, filter requirements and tasks, add and change task
Create project, asiign team members and their roles, set project name, set deadline
Check information the status of the project
Search for a specific project name or ID, information about project will be shown

Figure 1

Each use case has a corresponding use case description which depicts the process of each functionality and shows the steps the user must follow when using the system. Every use case description has also an activity diagram, which shows the flow of activities in a process including concurrent activities and branches



Use case	Manage requirements
Summary	Check every task status and also adding, editing or removing tasks for requirements.
Actor	Team Member
Precondition	A task has to be already created.(At least it should be set)
Postcondition	All relevant comments should be stored in the same file as the project is. Assigned tasks should be saved.
Base sequence	1) Enter the name of the project or task and return the file 2) If the name which was entered doesn't exist then go back to step 1 3) System lists all active projects and tasks that haven't been completed. For each task, the status and the responsable team's members are shown 4) Select the task from the list using the personal ID number 5) Option to list all projects and tasks (name of the project, purpose, status(not started, ended, approved, rejected), participants, deadline). If the task in "Approved" then also the starting date, requirements, and client. 6) Approve the selected task or go back to step 5 to select another 7) System lists every team member responsable with the task with their name and their personal ID number 8) Select a team member using the personal ID number as a criteria 9) Option to assign a new task with its deadline for the selected team member 10) Optionally, create a new project and assign it to the team members 11) Approve the modification 12) A modification with the specified task, team member, deadline is created and stored in the system.
Exception sequence	
Note	If the project is not in the list, then the command could be on hold until the project is added by the team member. A task can be cancelled or delayed at any time. (in step 5) A project/task can be added to the list by the team member without assigning it to any member or without information about it, but changing the status to "Approved" can only be made just by assigning members and introducing the necessary data.

Figure 2

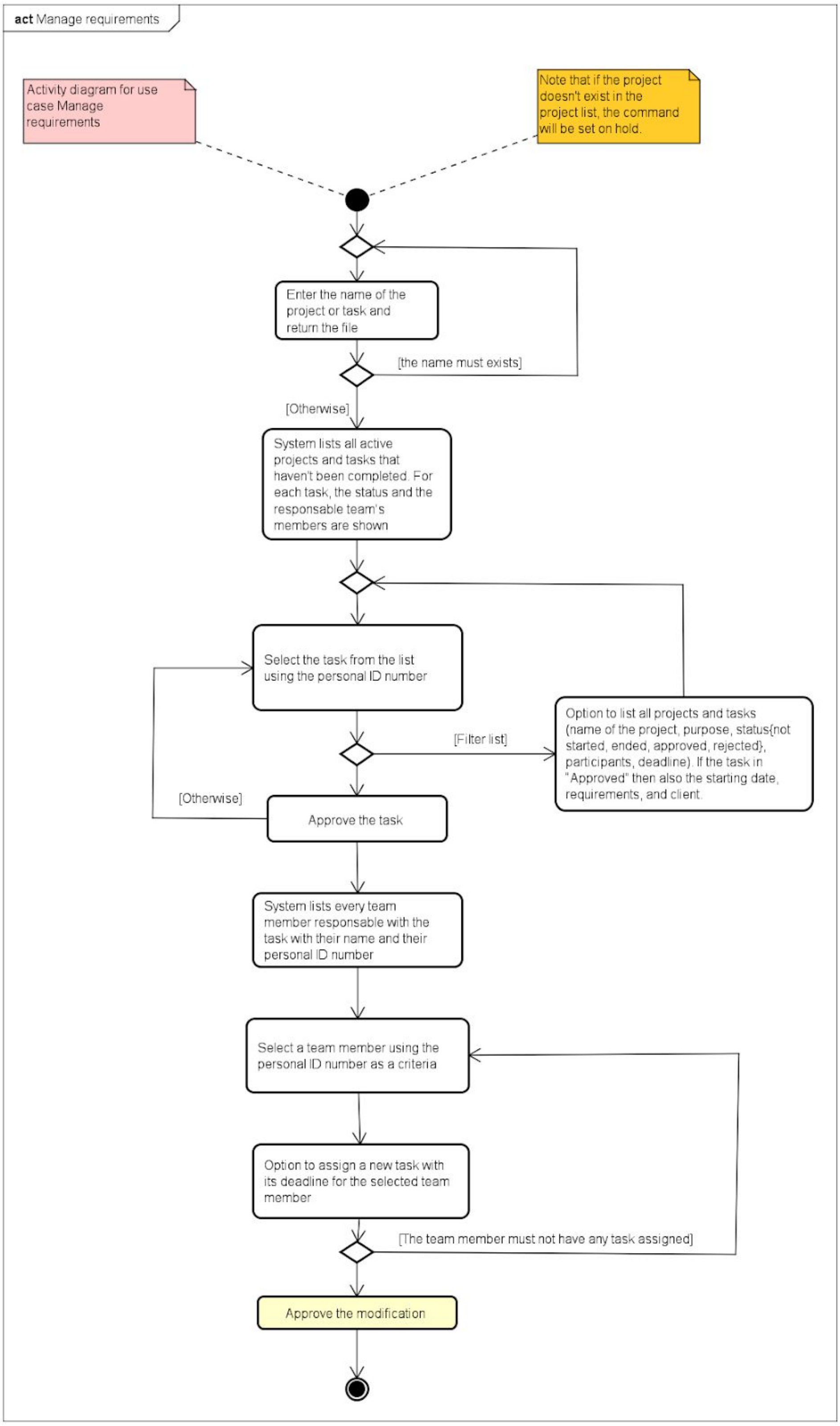
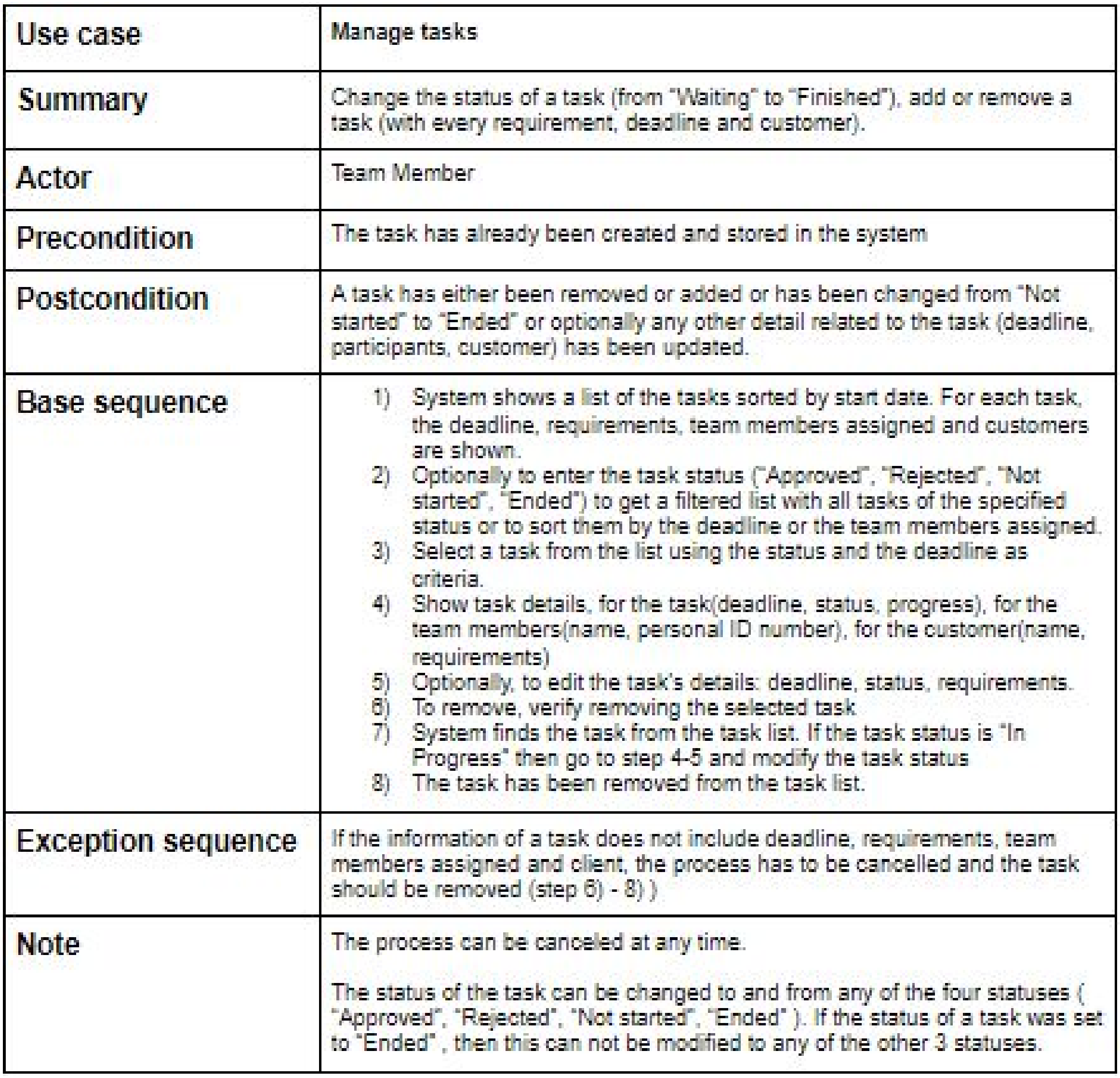

act Manage requirements
Activity diagram for use case Manage requirements
Note that if the project doesn't exist in the project list, the command will be set on hold.
Enter the name of the project or task and return the file
[the name must exists]
[Otherwise]
System lists all active projects and tasks that haven't been completed. For each task, the status and the responsable team's members are shown
Select the task from the list using the personal ID number
[Filter list]
Option to list all projects and tasks (name of the project, purpose, status{not started, ended, approved, rejected}, participants, deadline). If the task in "Approved" then also the starting date, requirements, and client.
[Otherwise]
Approve the task
System lists every team member responsable with the task with their name and their personal ID number
Select a team member using the personal ID number as a criteria
Option to assign a new task with its deadline for the selected team member
[The team member must not have any task assigned]
Approve the modification

Figure 3



Use case	Manage tasks
Summary	Change the status of a task (from "Waiting" to "Finished"), add or remove a task (with every requirement, deadline and customer).
Actor	Team Member
Precondition	The task has already been created and stored in the system
Postcondition	A task has either been removed or added or has been changed from "Not started" to "Ended" or optionally any other detail related to the task (deadline, participants, customer) has been updated.
Base sequence	1) System shows a list of the tasks sorted by start date. For each task, the deadline, requirements, team members assigned and customers are shown. 2) Optionally to enter the task status ("Approved", "Rejected", "Not started", "Ended") to get a filtered list with all tasks of the specified status or to sort them by the deadline or the team members assigned. 3) Select a task from the list using the status and the deadline as criteria. 4) Show task details, for the task(deadline, status, progress), for the team members(name, personal ID number), for the customer(name, requirements) 5) Optionally, to edit the task's details: deadline, status, requirements. 6) To remove, verify removing the selected task 7) System finds the task from the task list. If the task status is "In Progress" then go to step 4-5 and modify the task status 8) The task has been removed from the task list.
Exception sequence	If the information of a task does not include deadline, requirements, team members assigned and client, the process has to be cancelled and the task should be removed (step 6) - 8))
Note	The process can be canceled at any time. The status of the task can be changed to and from any of the four statuses ("Approved", "Rejected", "Not started", "Ended"). If the status of a task was set to "Ended" , then this can not be modified to any of the other 3 statuses.

Figure 4

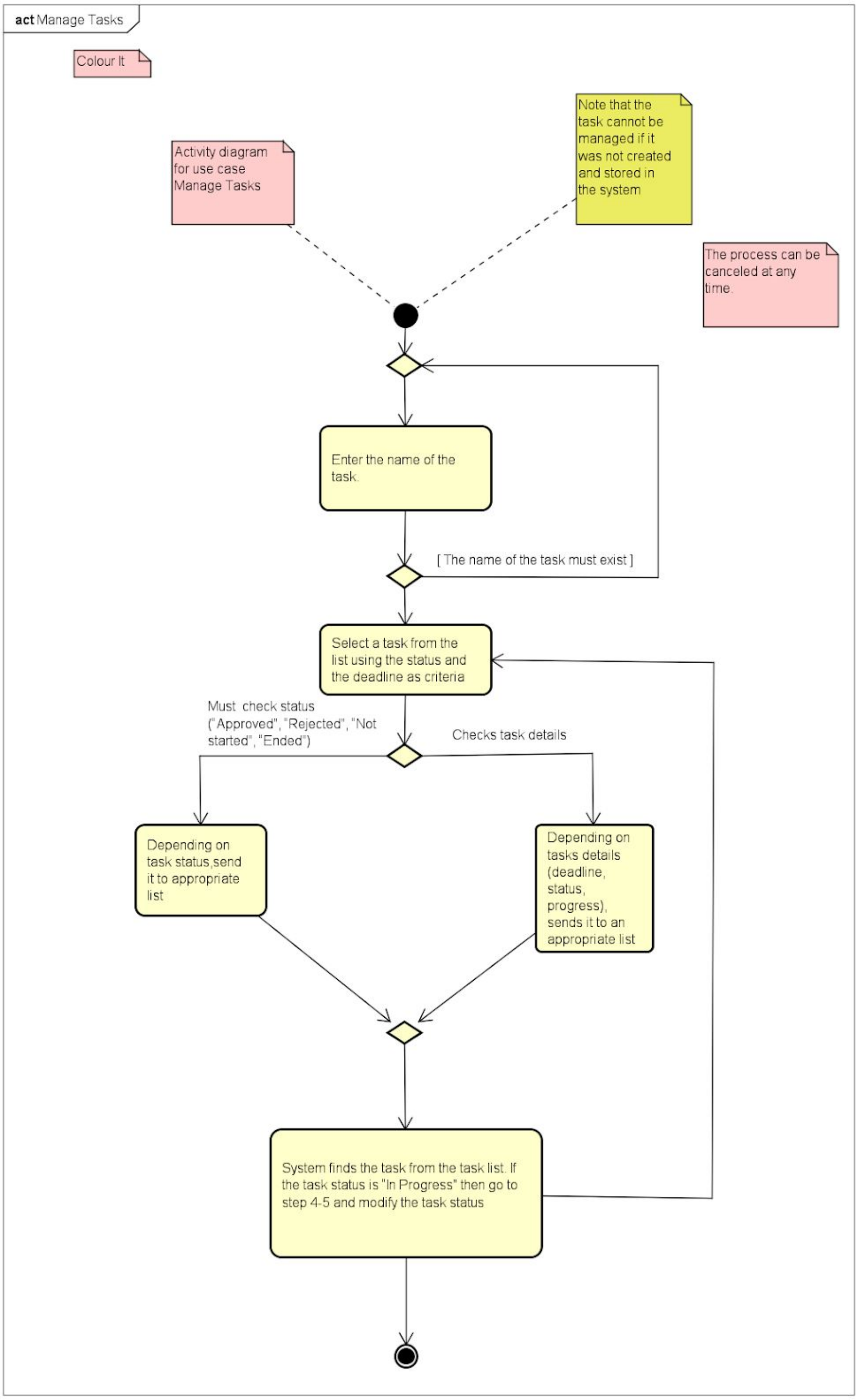
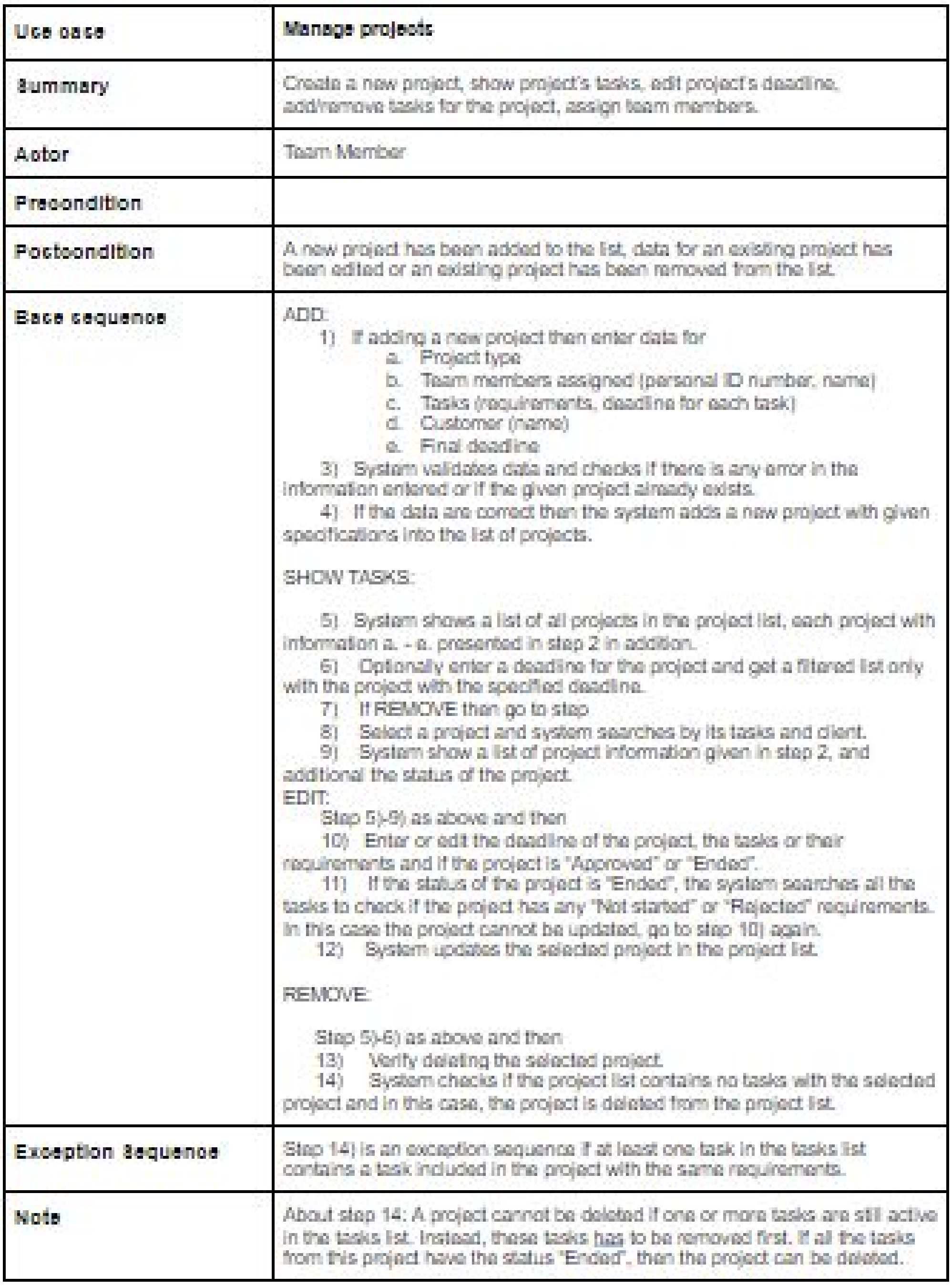

act Manage Tasks
Colour It
Activity diagram for use case Manage Tasks
Note that the task cannot be managed if it was not created and stored in the system
The process can be canceled at any time.
Enter the name of the task.
[The name of the task must exist]
Select a task from the list using the status and the deadline as criteria
Must check status ("Approved", "Rejected", "Not started", "Ended")
Checks task details
Depending on task status,send it to appropriate list
Depending on tasks details (deadline, status, progress), sends it to an appropriate list
System finds the task from the task list. If the task status is "In Progress" then go to step 4-5 and modify the task status

Figure 5



Use case	**Manage projects**
Summary	Create a new project, show project's tasks, edit project's deadline, add/remove tasks for the project, assign team members.
Actor	Team Member
Precondition	
Postcondition	A new project has been added to the list, data for an existing project has been edited or an existing project has been removed from the list.
Base sequence	ADD: 1) If adding a new project then enter data for a. Project type b. Team members assigned (personal ID number, name) c. Tasks (requirements, deadline for each task) d. Customer (name) e. Final deadline 3) System validates data and checks if there is any error in the information entered or if the given project already exists. 4) If the data are correct then the system adds a new project with given specifications into the list of projects. SHOW TASKS: 5) System shows a list of all projects in the project list, each project with information a. - e. presented in step 2 in addition. 6) Optionally enter a deadline for the project and get a filtered list only with the project with the specified deadline. 7) If REMOVE then go to step 8) Select a project and system searches by its tasks and client. 9) System show a list of project information given in step 2, and additional the status of the project. EDIT: Step 5)-9) as above and then 10) Enter or edit the deadline of the project, the tasks or their requirements and if the project is "Approved" or "Ended". 11) If the status of the project is "Ended", the system searches all the tasks to check if the project has any "Not started" or "Rejected" requirements. In this case the project cannot be updated, go to step 10) again. 12) System updates the selected project in the project list. REMOVE: Step 5)-6) as above and then 13) Verify deleting the selected project. 14) System checks if the project list contains no tasks with the selected project and in this case, the project is deleted from the project list.
Exception Sequence	Step 14) is an exception sequence if at least one task in the tasks list contains a task included in the project with the same requirements.
Note	About step 14: A project cannot be deleted if one or more tasks are still active in the tasks list. Instead, these tasks has to be removed first. If all the tasks from this project have the status "Ended", then the project can be deleted.

Figure 6

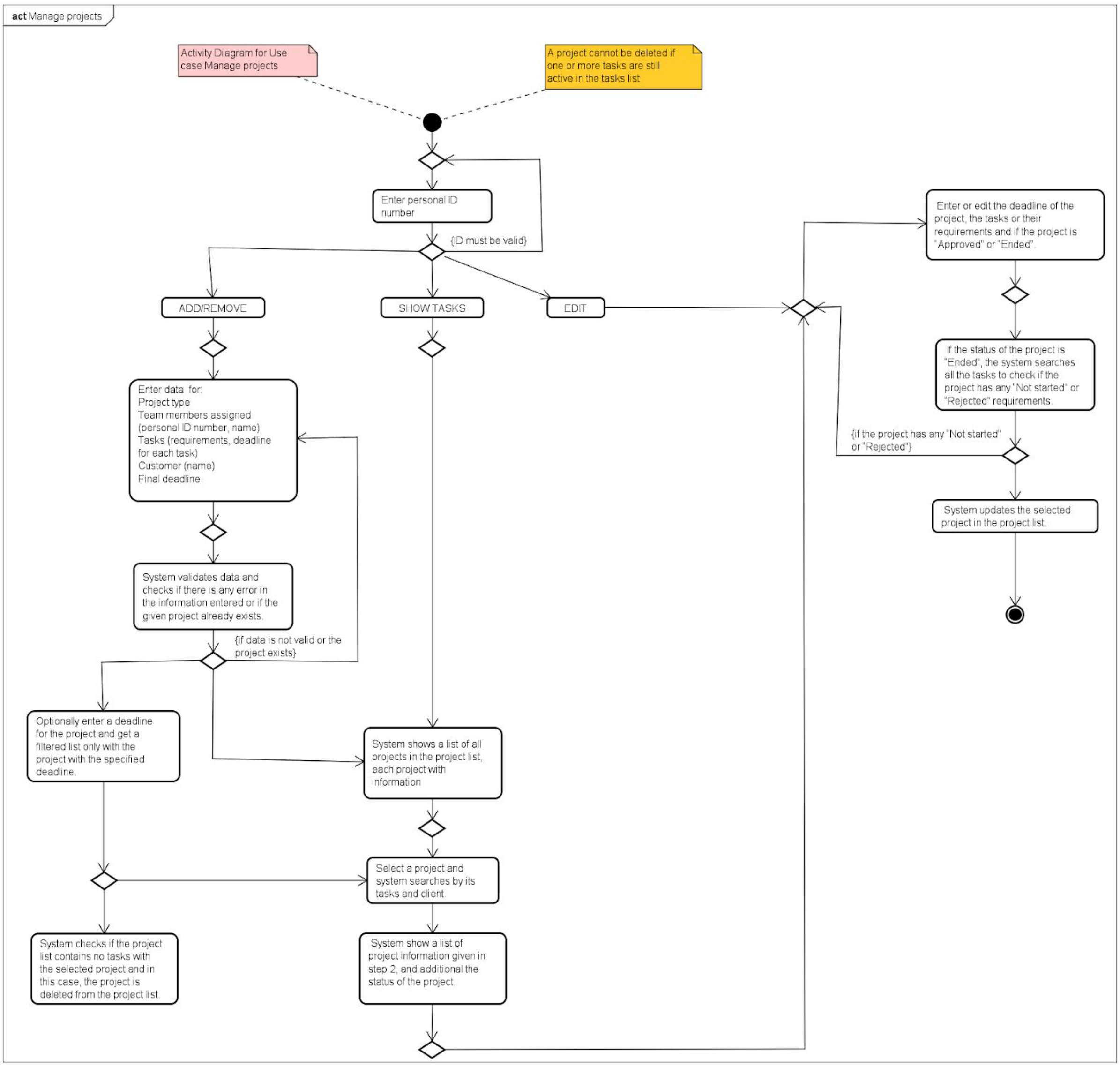
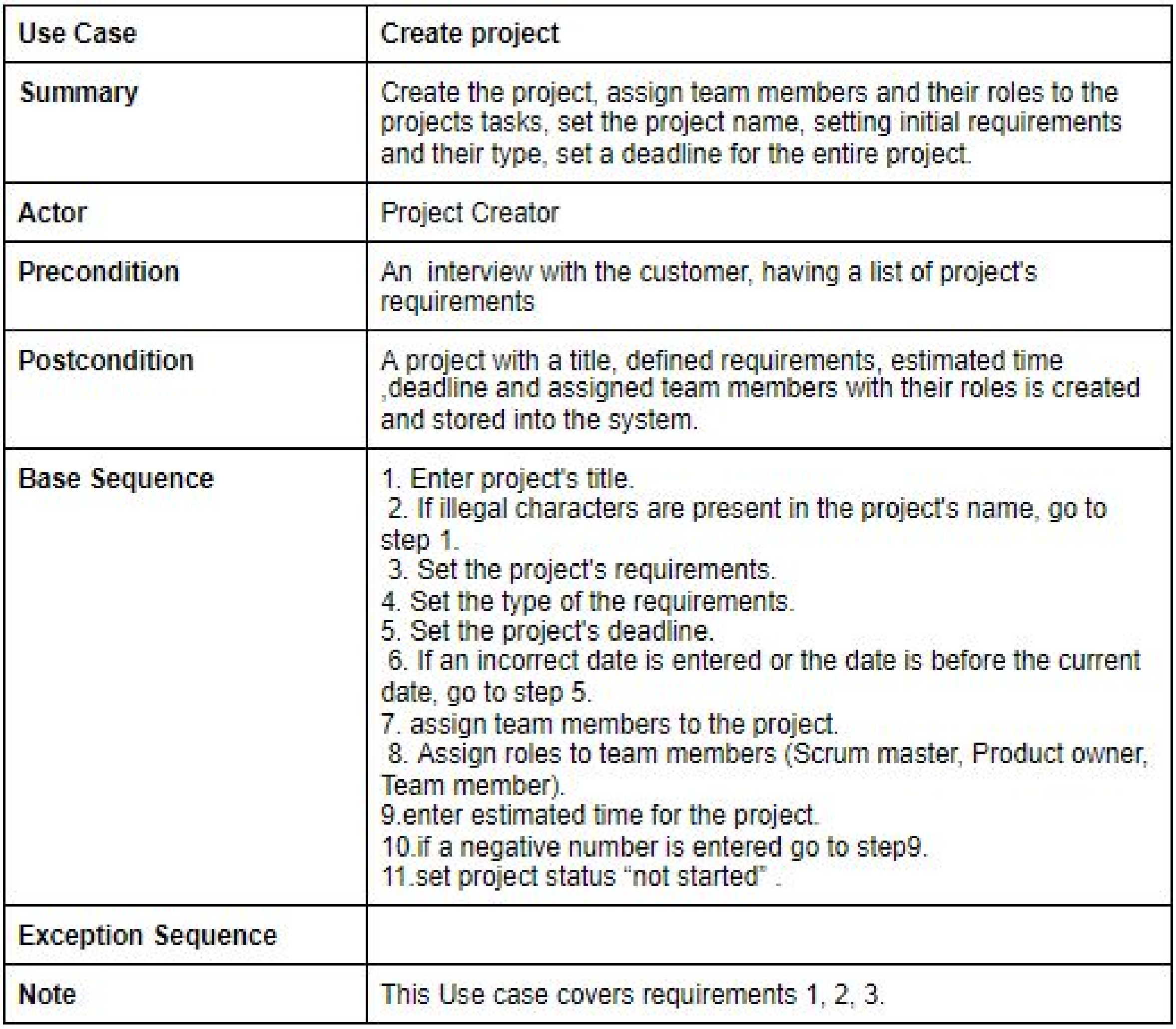

act Manage projects
Activity Diagram for Use case Manage projects
A project cannot be deleted if one or more tasks are still active in the tasks list
Enter personal ID number
{ID must be valid}
ADD/REMOVE
SHOW TASKS
EDIT
Enter data for:
Project type
Team members assigned (personal ID number, name)
Tasks (requirements, deadline for each task)
Customer (name)
Final deadline
System validates data and checks if there is any error in the information entered or if the given project already exists.
{if data is not valid or the project exists}
Optionally enter a deadline for the project and get a filtered list only with the project with the specified deadline.
System checks if the project list contains no tasks with the selected project and in this case, the project is deleted from the project list.
System shows a list of all projects in the project list, each project with information
Select a project and system searches by its tasks and client.
System show a list of project information given in step 2, and additional the status of the project.
Enter or edit the deadline of the project, the tasks or their requirements and if the project is "Approved" or "Ended".
If the status of the project is "Ended", the system searches all the tasks to check if the project has any "Not started" or "Rejected" requirements.
{if the project has any "Not started" or "Rejected"}
System updates the selected project in the project list.

Figure 7



Use Case	Create project
Summary	Create the project, assign team members and their roles to the projects tasks, set the project name, setting initial requirements and their type, set a deadline for the entire project.
Actor	Project Creator
Precondition	An interview with the customer, having a list of project's requirements
Postcondition	A project with a title, defined requirements, estimated time ,deadline and assigned team members with their roles is created and stored into the system.
Base Sequence	1. Enter project's title. 2. If illegal characters are present in the project's name, go to step 1. 3. Set the project's requirements. 4. Set the type of the requirements. 5. Set the project's deadline. 6. If an incorrect date is entered or the date is before the current date, go to step 5. 7. assign team members to the project. 8. Assign roles to team members (Scrum master, Product owner, Team member). 9.enter estimated time for the project. 10.if a negative number is entered go to step9. 11.set project status “not started” .
Exception Sequence	
Note	This Use case covers requirements 1, 2, 3.

Figure 8

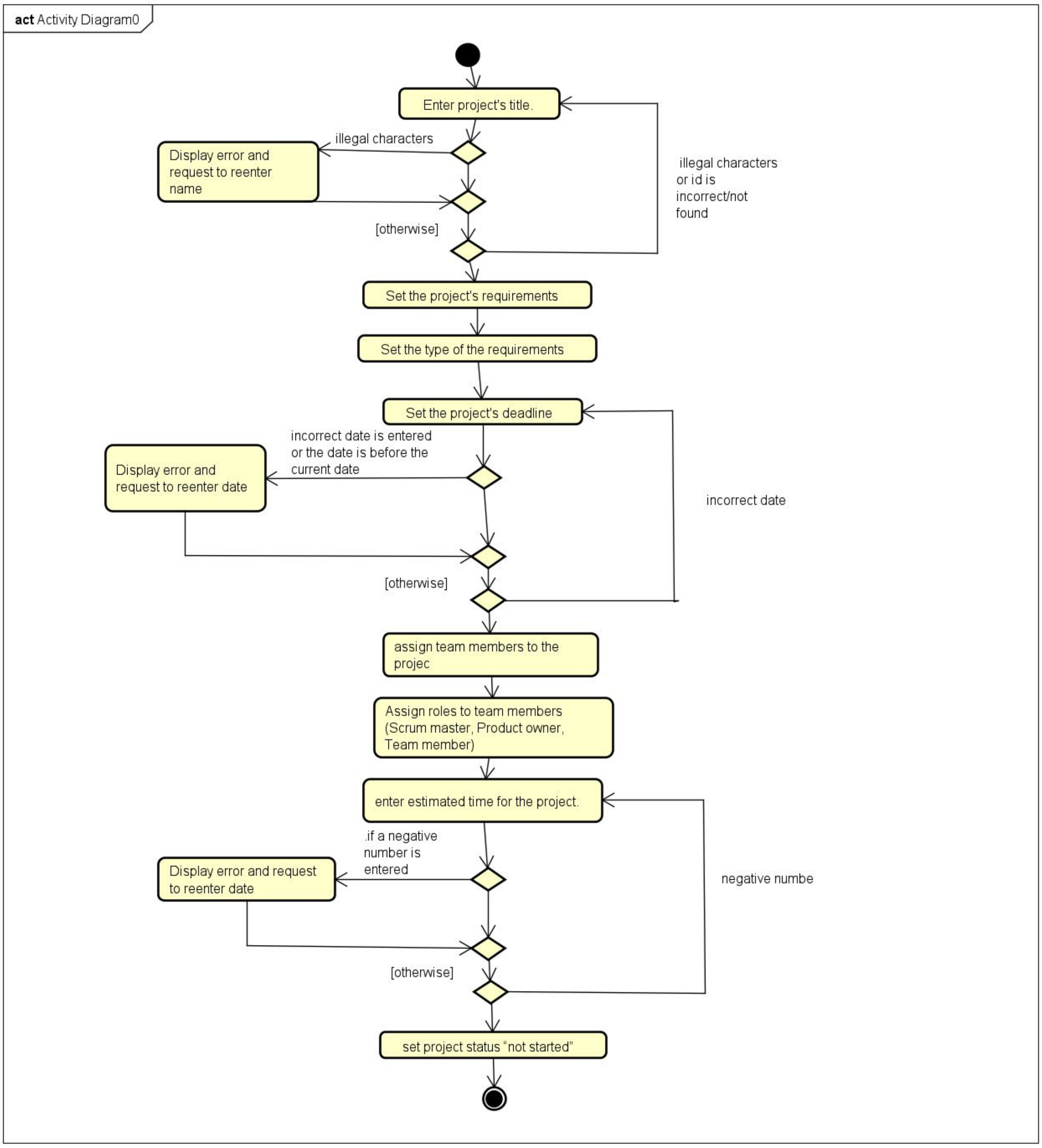
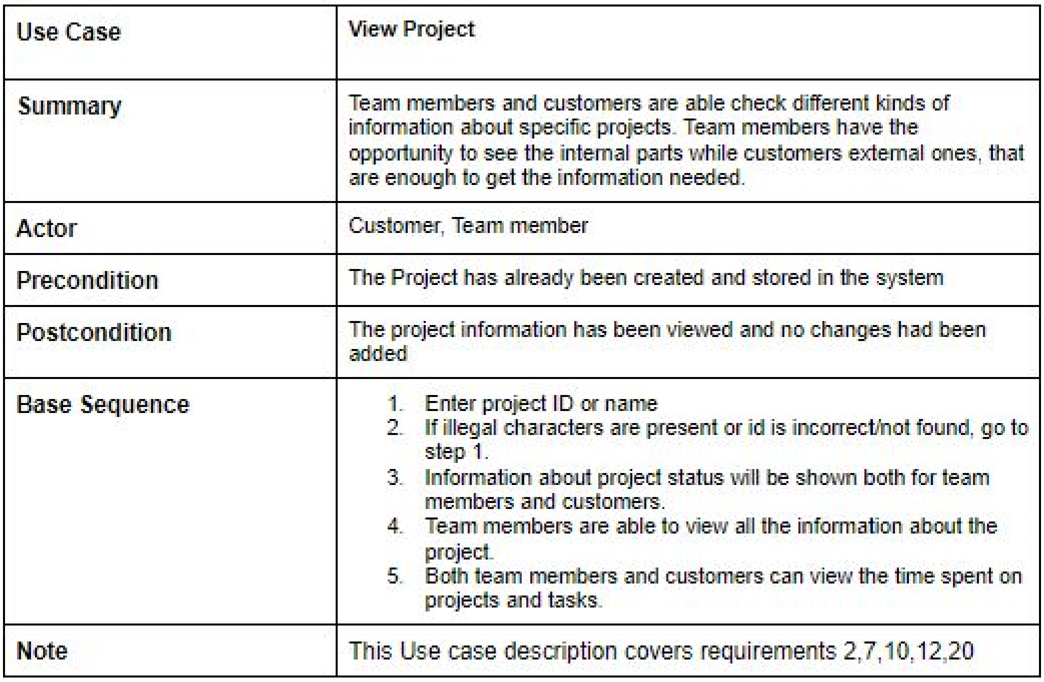

act Activity Diagram0
Enter project's title.
illegal characters
Display error and request to reenter name
[otherwise]
illegal characters or id is incorrect/not found
Set the project's requirements
Set the type of the requirements
Set the project's deadline
incorrect date is entered or the date is before the current date
Display error and request to reenter date
incorrect date
[otherwise]
assign team members to the projec
Assign roles to team members (Scrum master, Product owner, Team member)
enter estimated time for the project.
.if a negative number is entered
Display error and request to reenter date
negative numbe
[otherwise]
set project status "not started"

Figure 9



Use Case	**View Project**
Summary	Team members and customers are able check different kinds of information about specific projects. Team members have the opportunity to see the internal parts while customers external ones, that are enough to get the information needed.
Actor	Customer, Team member
Precondition	The Project has already been created and stored in the system
Postcondition	The project information has been viewed and no changes had been added
Base Sequence	1. Enter project ID or name 2. If illegal characters are present or id is incorrect/not found, go to step 1. 3. Information about project status will be shown both for team members and customers. 4. Team members are able to view all the information about the project. 5. Both team members and customers can view the time spent on projects and tasks.
Note	This Use case description covers requirements 2,7,10,12,20

Figure 10

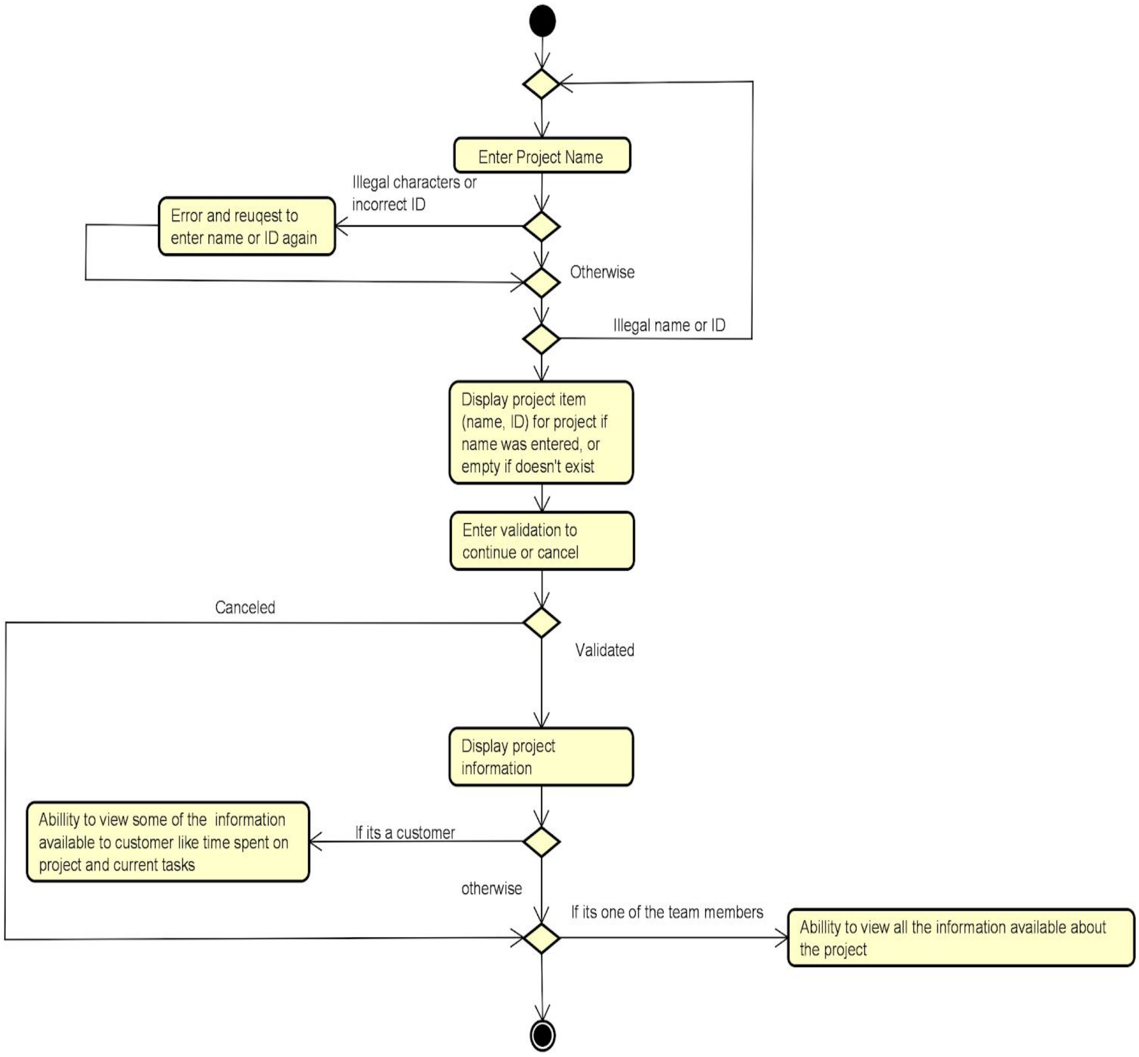
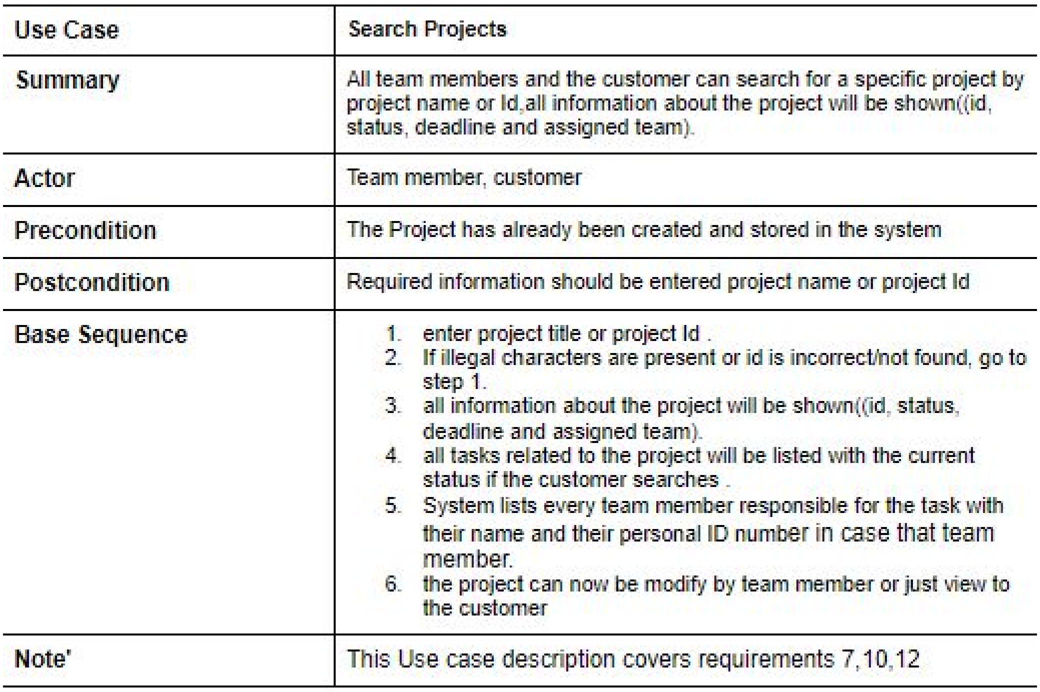

Enter Project Name
Illegal characters or incorrect ID
Error and reuqest to enter name or ID again
Otherwise
Illegal name or ID
Display project item (name, ID) for project if name was entered, or empty if doesn't exist
Enter validation to continue or cancel
Canceled
Validated
Display project information
Abillity to view some of the information available to customer like time spent on project and current tasks
If its a customer
otherwise
If its one of the team members
Abillity to view all the information available about the project

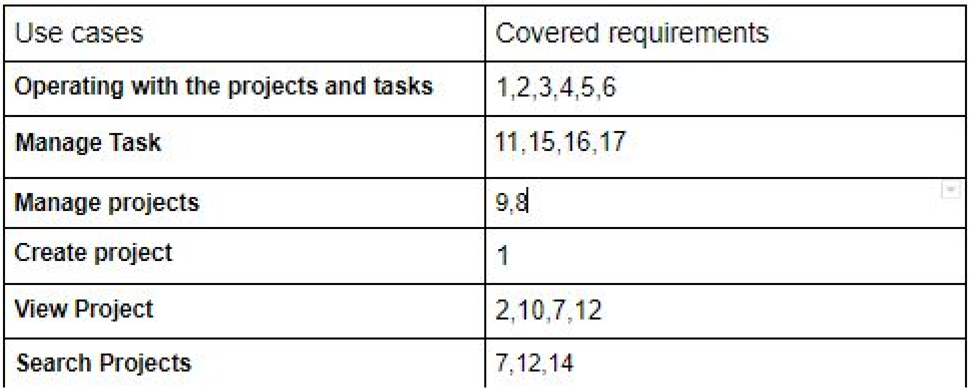
Figure 11



Use Case	Search Projects
Summary	All team members and the customer can search for a specific project by project name or Id,all information about the project will be shown((id, status, deadline and assigned team).
Actor	Team member, customer
Precondition	The Project has already been created and stored in the system
Postcondition	Required information should be entered project name or project Id
Base Sequence	1. enter project title or project Id . 2. If illegal characters are present or id is incorrect/not found, go to step 1. 3. all information about the project will be shown((id, status, deadline and assigned team). 4. all tasks related to the project will be listed with the current status if the customer searches . 5. System lists every team member responsible for the task with their name and their personal ID number in case that team member. 6. the project can now be modify by team member or just view to the customer
Note'	This Use case description covers requirements 7,10,12

Figure 12

As use cases and requirements correspond a clear link can be seen between them. In the following table the use cases and its' covered requirements are presented:



Use cases	Covered requirements
Operating with the projects and tasks	1,2,3,4,5,6
Manage Task	11,15,16,17
Manage projects	9,8
Create project	1
View Project	2,10,7,12
Search Projects	7,12,14

Figure 13

3.2 Domain Model

To provide an overview of the classes and relations between them, there was created, as a last step of analysis, a domain model.

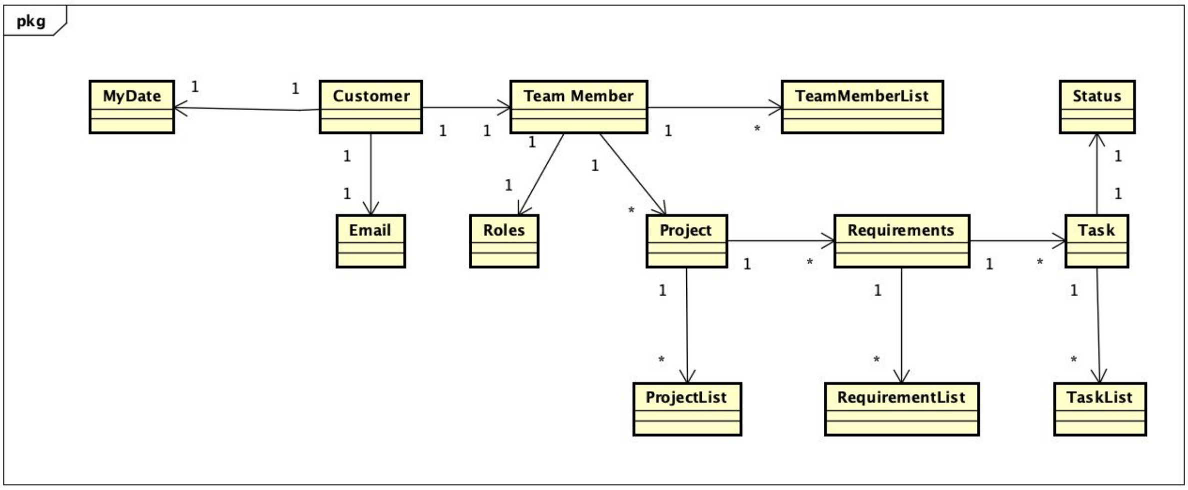


Figure 14

The diagram represents the connection between classes and there are some important facts that must be stated based on the domain:

- Product Owner and Scrum Master are a part of Team Members
- Requirements contain tasks which have a status
- Each project contains requirements

4. Design

The purpose of the design section is to outline how the system is structured; i.e. to transform the artefacts of the analysis into a model that can be implemented. The design section is relevant for the programmer, whereas the analysis is relevant for the stakeholder.

4.1 Class Diagram

(zoom in for better quality)

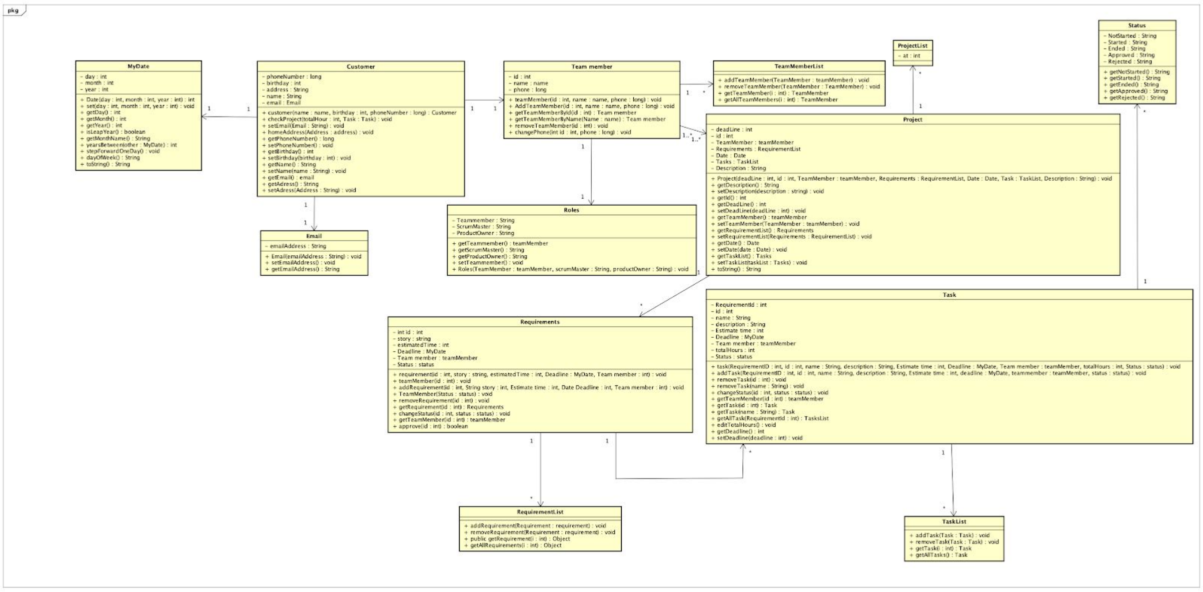

pkg
MyDate
- day : int
- month : int
- year : int
+ Date(day : int, month : int, year : int) : int
+ set(day : int, month : int, year : int) : void
+ getDay() : int
+ getMonth() : int
+ getYear() : int
+ isLeapYear() : boolean
+ getMonthName() : String
+ yearsBetween(other : MyDate) : int
+ stepForwardOneDay() : void
+ dayOfWeek() : String
+ toString() : String
Customer
- phoneNumber : long
- birthday : int
- address : String
- name : String
- email : Email
+ customer(name : name, birthday : int, phoneNumber : long) : Customer
+ checkProject(totalHour : int, Task : Task) : void
+ setEmail(Email : String) : void
+ homeAddress(Address : address) : void
+ getPhoneNumber() : long
+ setPhoneNumber() : void
+ getBirthday() : int
+ setBirthday(birthday : int) : void
+ getName() : String
+ setName(name : String) : void
+ getEmail() : email
+ getAdress() : String
+ setAdress(Address : String) : void
Email
- emailAddress : String
+ Email(emailAddress : String) : void
+ setEmailAddress() : void
+ getEmailAddress() : String
Team member
- id : int
- name : name
- phone : long
+ teamMember(id : int, name : name, phone : long) : void
+ AddTeamMember(id : int, name : name, phone : long) : void
+ getTeamMemberById(id : int) : Team member
+ getTeamMemberByName(Name : name) : Team member
+ removeTeamMember(id : int) : void
+ changePhone(int id : int, phone : long) : void
Roles
- Teammember : String
- ScrumMaster : String
- ProductOwner : String
+ getTeammember() : teamMember
+ getScrumMaster() : String
+ getProductOwner() : String
+ setTeammember() : void
+ Roles(TeamMember : teamMember, scrumMaster : String, productOwner : String) : void
TeamMemberList
+ addTeamMember(TeamMember : teamMember) : void
+ removeTeamMember(TeamMember : TeamMember) : void
+ getTeamMember(i : int) : TeamMember
+ getAllTeamMembers(i : int) : TeamMember
ProjectList
- at : int
Status
- NotStarted : String
- Started : String
- Ended : String
- Approved : String
- Rejected : String
+ getNotStarted() : String
+ getStarted() : String
+ getEnded() : String
+ getApproved() : String
+ getRejected() : String
Project
- deadLine : int
- id : int
- TeamMember : teamMember
- Requirements : RequirementList
- Date : Date
- Tasks : TaskList
- Description : String
+ Project(deadLine : int, id : int, TeamMember : teamMember, Requirements : RequirementList, Date : Date, Task : TaskList, Description : String) : void
+ getDescription() : String
+ setDescription(description : string) : void
+ getId() : int
+ getDeadLine() : int
+ setDeadLine(deadLine : int) : void
+ getTeamMember() : teamMember
+ setTeamMember(TeamMember : teamMember) : void
+ getRequirementList() : Requirements
+ setRequirementList(Requirements : RequirementList) : void
+ getDate() : Date
+ setDate(date : Date) : void
+ getTaskList() : Tasks
+ setTaskList(taskList : Tasks) : void
+ toString() : String
Requirements
- int id : int
- story : string
- estimatedTime : int
- Deadline : MyDate
- Team member : teamMember
- Status : status
+ requirement(id : int, story : string, estimatedTime : int, Deadline : MyDate, Team member : int) : void
+ teamMember(id : int) : void
+ addRequirement(id : int, String story : int, Estimate time : int, Date Deadline : int, Team member : int) : void
+ TeamMember(Status : status) : void
+ removeRequirement(id : int) : void
+ getRequirement(id : int) : Requirements
+ changeStatus(id : int, status : status) : void
+ getTeamMember(id : int) : teamMember
+ approve(id : int) : boolean
Task
- RequirementId : int
- id : int
- name : String
- description : String
- Estimate time : int
- Deadline : MyDate
- Team member : teamMember
- totalHours : int
- Status : status
+ task(RequirementID : int, id : int, name : String, description : String, Estimate time : int, Deadline : MyDate, Team member : teamMember, totalHours : int, Status : status) : void
+ addTask(RequirementID : int, id : int, name : String, description : String, Estimate time : int, deadline : MyDate, teammember : teamMember, status : status) : void
+ removeTask(id : int) : void
+ removeTask(name : String) : void
+ changeStatus(id : int, status : status) : void
+ getTeamMember(id : int) : teamMember
+ getTask(id : int) : Task
+ getTask(name : String) : Task
+ getAllTask(RequirementId : int) : TasksList
+ editTotalHours() : void
+ getDeadline() : int
+ setDeadline(deadline : int) : void
RequirementList
+ addRequirement(Requirement : requirement) : void
+ removeRequirement(Requirement : requirement) : void
+ public getRequirement(i : int) : Object
+ getAllRequirements(i : int) : Object
TaskList
+ addTask(Task : Task) : void
+ removeTask(Task : Task) : void
+ getTask(i : int) : Task
+ getAllTasks() : Task
1
*
1..*

Figure 15

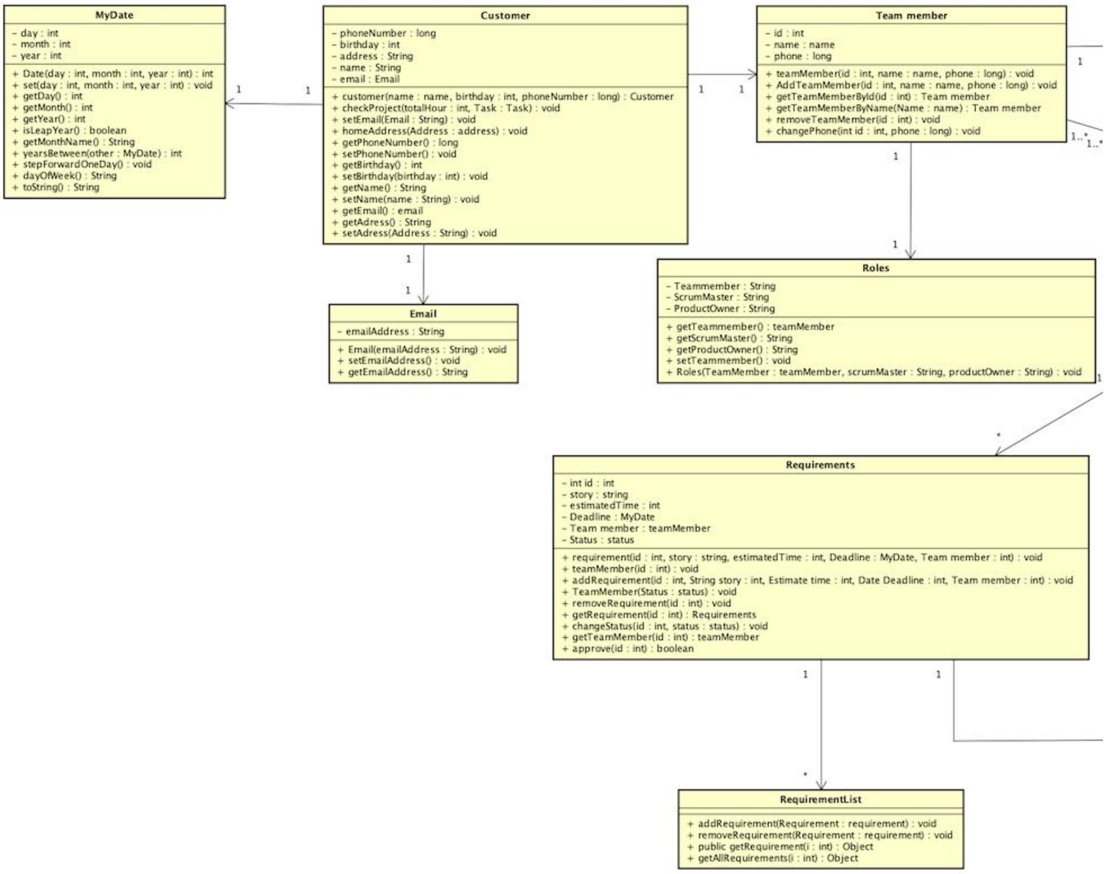

MyDate
- day : int
- month : int
- year : int
+ Date(day : int, month : int, year : int) : int
+ set(day : int, month : int, year : int) : void
+ getDay() : int
+ getMonth() : int
+ getYear() : int
+ isLeapYear() : boolean
+ getMonthName() : String
+ yearsBetween(other : MyDate) : int
+ stepForwardOneDay() : void
+ dayOfWeek() : String
+ toString() : String
Customer
- phoneNumber : long
- birthday : int
- address : String
- name : String
- email : Email
+ customer(name : name, birthday : int, phoneNumber : long) : Customer
+ checkProject(totalHour : int, Task : Task) : void
+ setEmail(Email : String) : void
+ homeAddress(Address : address) : void
+ getPhoneNumber() : long
+ setPhoneNumber() : void
+ getBirthday() : int
+ setBirthday(birthday : int) : void
+ getName() : String
+ setName(name : String) : void
+ getEmail() : email
+ getAdress() : String
+ setAdress(Address : String) : void
Email
- emailAddress : String
+ Email(emailAddress : String) : void
+ setEmailAddress() : void
+ getEmailAddress() : String
Team member
- id : int
- name : name
- phone : long
+ teamMember(id : int, name : name, phone : long) : void
+ AddTeamMember(id : int, name : name, phone : long) : void
+ getTeamMemberById(id : int) : Team member
+ getTeamMemberByName(Name : name) : Team member
+ removeTeamMember(id : int) : void
+ changePhone(int id : int, phone : long) : void
Roles
- Teammember : String
- ScrumMaster : String
- ProductOwner : String
+ getTeammember() : teamMember
+ getScrumMaster() : String
+ getProductOwner() : String
+ setTeammember() : void
+ Roles(TeamMember : teamMember, scrumMaster : String, productOwner : String) : void
Requirements
- int id : int
- story : string
- estimatedTime : int
- Deadline : MyDate
- Team member : teamMember
- Status : status
+ requirement(id : int, story : string, estimatedTime : int, Deadline : MyDate, Team member : int) : void
+ teamMember(id : int) : void
+ addRequirement(id : int, String story : int, Estimate time : int, Date Deadline : int, Team member : int) : void
+ TeamMember(Status : status) : void
+ removeRequirement(id : int) : void
+ getRequirement(id : int) : Requirements
+ changeStatus(id : int, status : status) : void
+ getTeamMember(id : int) : teamMember
+ approve(id : int) : boolean
RequirementList
+ addRequirement(Requirement : requirement) : void
+ removeRequirement(Requirement : requirement) : void
+ public getRequirement(i : int) : Object
+ getAllRequirements(i : int) : Object
1
*
1..*

Figure 16

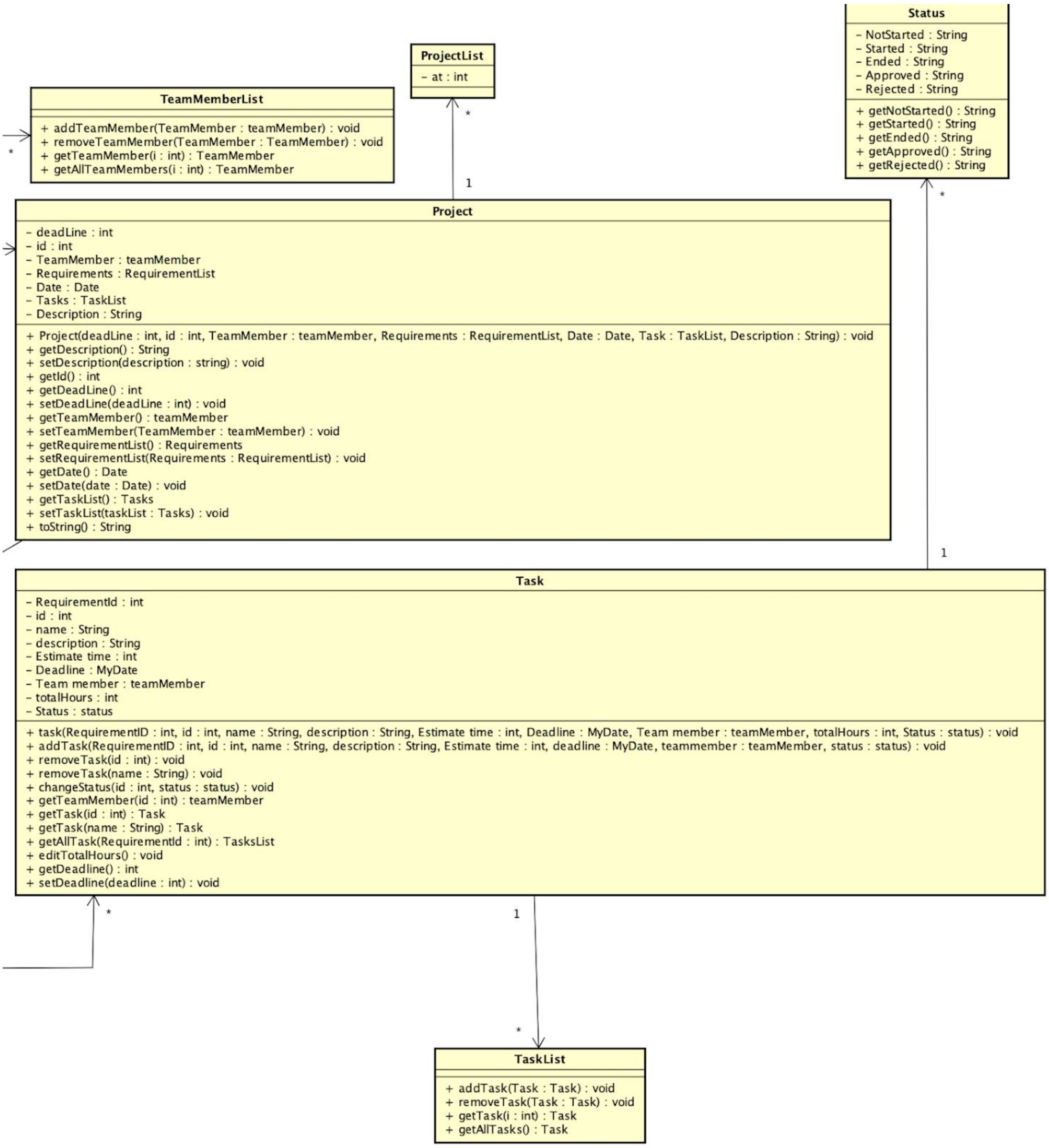

Status
- NotStarted : String
- Started : String
- Ended : String
- Approved : String
- Rejected : String
+ getNotStarted() : String
+ getStarted() : String
+ getEnded() : String
+ getApproved() : String
+ getRejected() : String
ProjectList
- at : int
TeamMemberList
+ addTeamMember(TeamMember : teamMember) : void
+ removeTeamMember(TeamMember : TeamMember) : void
+ getTeamMember(i : int) : TeamMember
+ getAllTeamMembers(i : int) : TeamMember
*
*
1
Project
- deadLine : int
- id : int
- TeamMember : teamMember
- Requirements : RequirementList
- Date : Date
- Tasks : TaskList
- Description : String
+ Project(deadLine : int, id : int, TeamMember : teamMember, Requirements : RequirementList, Date : Date, Task : TaskList, Description : String) : void
+ getDescription() : String
+ setDescription(description : string) : void
+ getId() : int
+ getDeadLine() : int
+ setDeadLine(deadLine : int) : void
+ getTeamMember() : teamMember
+ setTeamMember(TeamMember : teamMember) : void
+ getRequirementList() : Requirements
+ setRequirementList(Requirements : RequirementList) : void
+ getDate() : Date
+ setDate(date : Date) : void
+ getTaskList() : Tasks
+ setTaskList(taskList : Tasks) : void
+ toString() : String
*
1
Task
- RequirementId : int
- id : int
- name : String
- description : String
- Estimate time : int
- Deadline : MyDate
- Team member : teamMember
- totalHours : int
- Status : status
+ task(RequirementID : int, id : int, name : String, description : String, Estimate time : int, Deadline : MyDate, Team member : teamMember, totalHours : int, Status : status) : void
+ addTask(RequirementID : int, id : int, name : String, description : String, Estimate time : int, deadline : MyDate, teammember : teamMember, status : status) : void
+ removeTask(id : int) : void
+ removeTask(name : String) : void
+ changeStatus(id : int, status : status) : void
+ getTeamMember(id : int) : teamMember
+ getTask(id : int) : Task
+ getTask(name : String) : Task
+ getAllTask(RequirementId : int) : TasksList
+ editTotalHours() : void
+ getDeadline() : int
+ setDeadline(deadline : int) : void
*
1
*
TaskList
+ addTask(Task : Task) : void
+ removeTask(Task : Task) : void
+ getTask(i : int) : Task
+ getAllTasks() : Task

Figure 17

Class diagram consists of 13 classes. The most important classes are: Requirements, Tasks and Projects. They represent the wireframe of the program. Each team member has an assigned role which is stored in the team member list. One or more team members can work on one or more projects. For instance, there can be a project where 5 team members are working at the same time and can be another one where there is only one. Every project contains a list of requirements which are stored in RequirementsList. Each requirement has a task which, at its turn is stored in a task list.

4.2 GUI Class Diagram

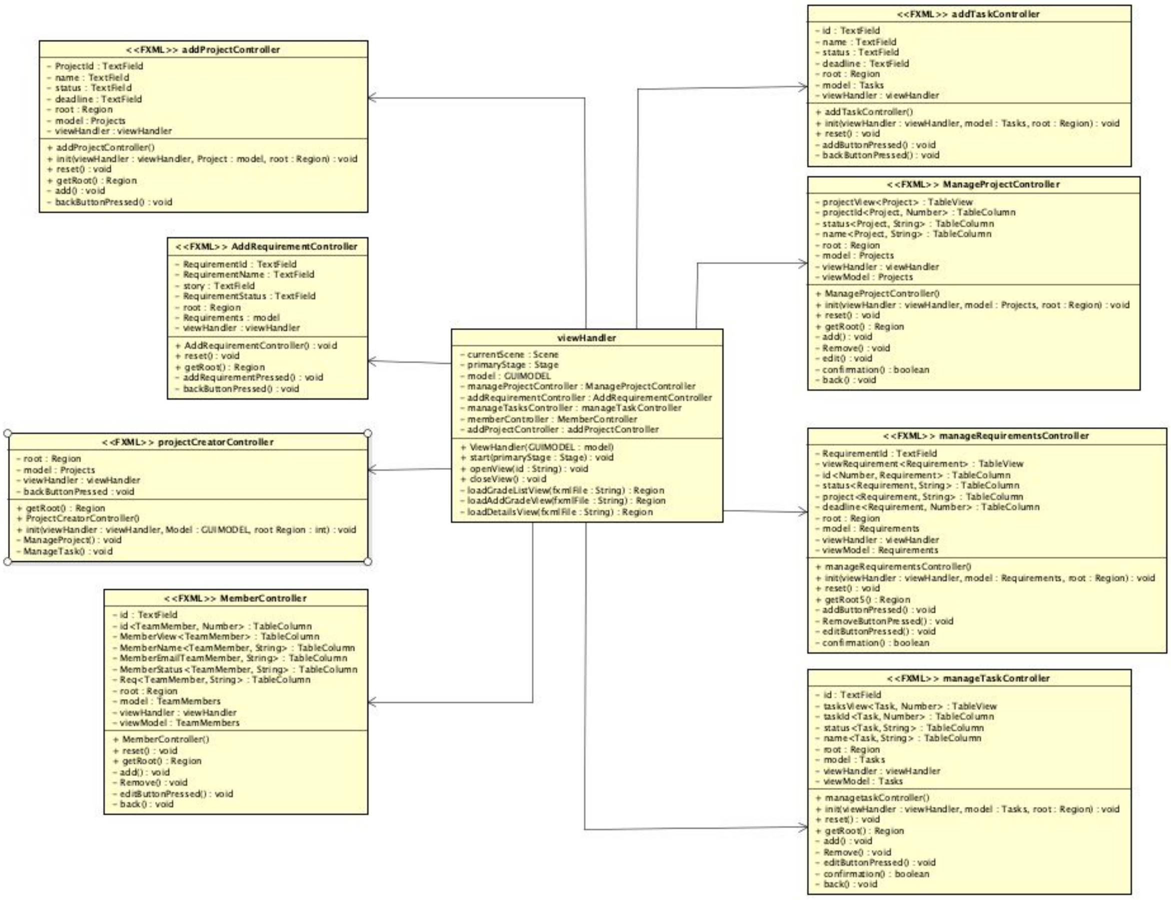

<<FXML>> addProjectController
- ProjectId : TextField
- name : TextField
- status : TextField
- deadline : TextField
- root : Region
- model : Projects
- viewHandler : viewHandler
+ addProjectController()
+ init(viewHandler : viewHandler, Project : model, root : Region) : void
+ reset() : void
+ getRoot() : Region
- add() : void
- backButtonPressed() : void
<<FXML>> AddRequirementController
- RequirementId : TextField
- RequirementName : TextField
- story : TextField
- RequirementStatus : TextField
- root : Region
- Requirements : model
- viewHandler : viewHandler
+ AddRequirementController() : void
+ reset() : void
+ getRoot() : Region
- addRequirementPressed() : void
- backButtonPressed() : void
<<FXML>> projectCreatorController
- root : Region
- model : Projects
- viewHandler : viewHandler
- backButtonPressed : void
+ getRoot() : Region
+ ProjectCreatorController()
+ init(viewHandler : viewHandler, Model : GUIMODEL, root Region : int) : void
- ManageProject() : void
- ManageTask() : void
<<FXML>> MemberController
- id : TextField
- id <TeamMember, Number> : TableColumn
- MemberView<TeamMember> : TableColumn
- MemberName<TeamMember, String> : TableColumn
- MemberEmailTeamMember, String> : TableColumn
- MemberStatus<TeamMember, String> : TableColumn
- Req<TeamMember, String> : TableColumn
- root : Region
- model : TeamMembers
- viewHandler : viewHandler
- viewModel : TeamMembers
+ MemberController()
+ reset() : void
+ getRoot() : Region
- add() : void
- Remove() : void
- editButtonPressed() : void
- back() : void
viewHandler
- currentScene : Scene
- primaryStage : Stage
- model : GUIMODEL
- manageProjectController : ManageProjectController
- addRequirementController : AddRequirementController
- manageTasksController : manageTaskController
- memberController : MemberController
- addProjectController : addProjectController
+ ViewHandler(GUIMODEL : model)
+ start(primaryStage : Stage) : void
+ openView(id : String) : void
+ closeView() : void
- loadGradeListView(fxmlFile : String) : Region
- loadAddGradeView(fxmlFile : String) : Region
- loadDetailsView(fxmlFile : String) : Region
<<FXML>> addTaskController
- id : TextField
- name : TextField
- status : TextField
- deadline : TextField
- root : Region
- model : Tasks
- viewHandler : viewHandler
+ addTaskController()
+ init(viewHandler : viewHandler, model : Tasks, root : Region) : void
+ reset() : void
- addButtonPressed() : void
- backButtonPressed() : void
<<FXML>> ManageProjectController
- projectView<Project> : TableView
- projectId<Project, Number> : TableColumn
- status<Project, String> : TableColumn
- name<Project, String> : TableColumn
- root : Region
- model : Projects
- viewHandler : viewHandler
- viewModel : Projects
+ ManageProjectController()
+ init(viewHandler : viewHandler, model : Projects, root : Region) : void
+ reset() : void
+ getRoot() : Region
- add() : void
- Remove() : void
- edit() : void
- confirmation() : boolean
- back() : void
<<FXML>> manageRequirementsController
- RequirementId : TextField
- viewRequirement<Requirement> : TableView
- id <Number, Requirement> : TableColumn
- status<Requirement, String> : TableColumn
- project<Requirement, String> : TableColumn
- deadline<Requirement, Number> : TableColumn
- root : Region
- model : Requirements
- viewHandler : viewHandler
- viewModel : Requirements
+ manageRequirementsController()
+ init(viewHandler : viewHandler, model : Requirements, root : Region) : void
+ reset() : void
+ getRoot5() : Region
- addButtonPressed() : void
- RemoveButtonPressed() : void
- editButtonPressed() : void
- confirmation() : boolean
<<FXML>> manageTaskController
- id : TextField
- tasksView<Task, Number> : TableView
- taskId <Task, Number> : TableColumn
- status <Task, String> : TableColumn
- name<Task, String> : TableColumn
- root : Region
- model : Tasks
- viewHandler : viewHandler
- viewModel : Tasks
+ managetaskController()
+ init(viewHandler : viewHandler, model : Tasks, root : Region) : void
+ reset() : void
+ getRoot() : Region
- add() : void
- Remove() : void
- editButtonPressed() : void
- confirmation() : boolean
- back() : void

Figure 18

The Class Diagram for the GUI was made with an initial concept in mind. The main idea was to keep it tidy and simple. The diagram is easy to read and understand and it was implemented succesfully. All the diagrams can be seen as Astah files.

4.3 Sequence Diagram

A sequence diagram shows how objects are working together. For this reason, there was made a sequence diagram about a method from Email class, which first of all has an if statement which basically validates the email by checking the presence of "@", as well as initializes the variable email Address, and a method getAllTasks, which has a loop through whole Tasks Arraylist, returning every each of them available and printing out the number and the task itself.

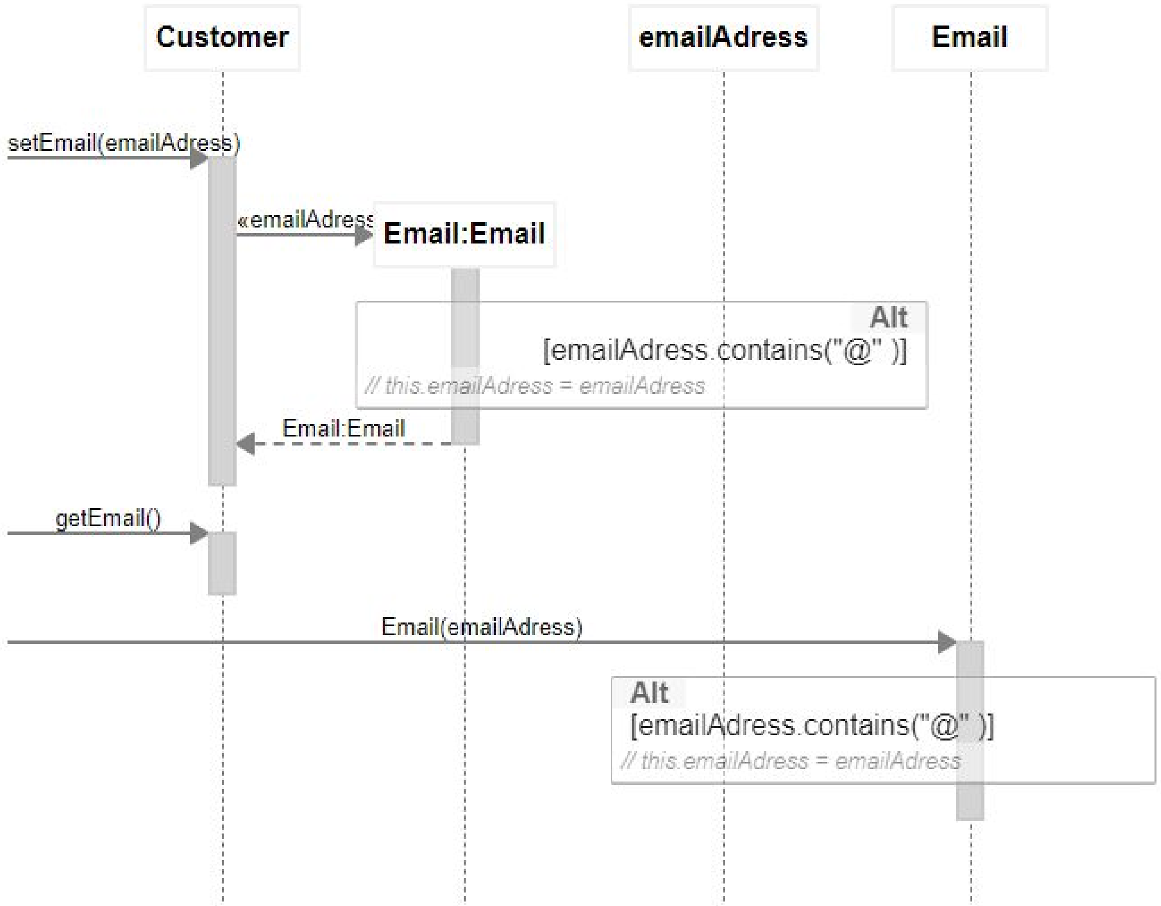

Customer
emailAdress
Email
setEmail(emailAdress)
«emailAdress
Email:Email
Alt
[emailAdress.contains("@")]
// this.emailAdress = emailAdress
Email:Email
getEmail()
Email(emailAdress)
Alt
[emailAdress.contains("@")]
// this.emailAdress = emailAdress

Figure 19

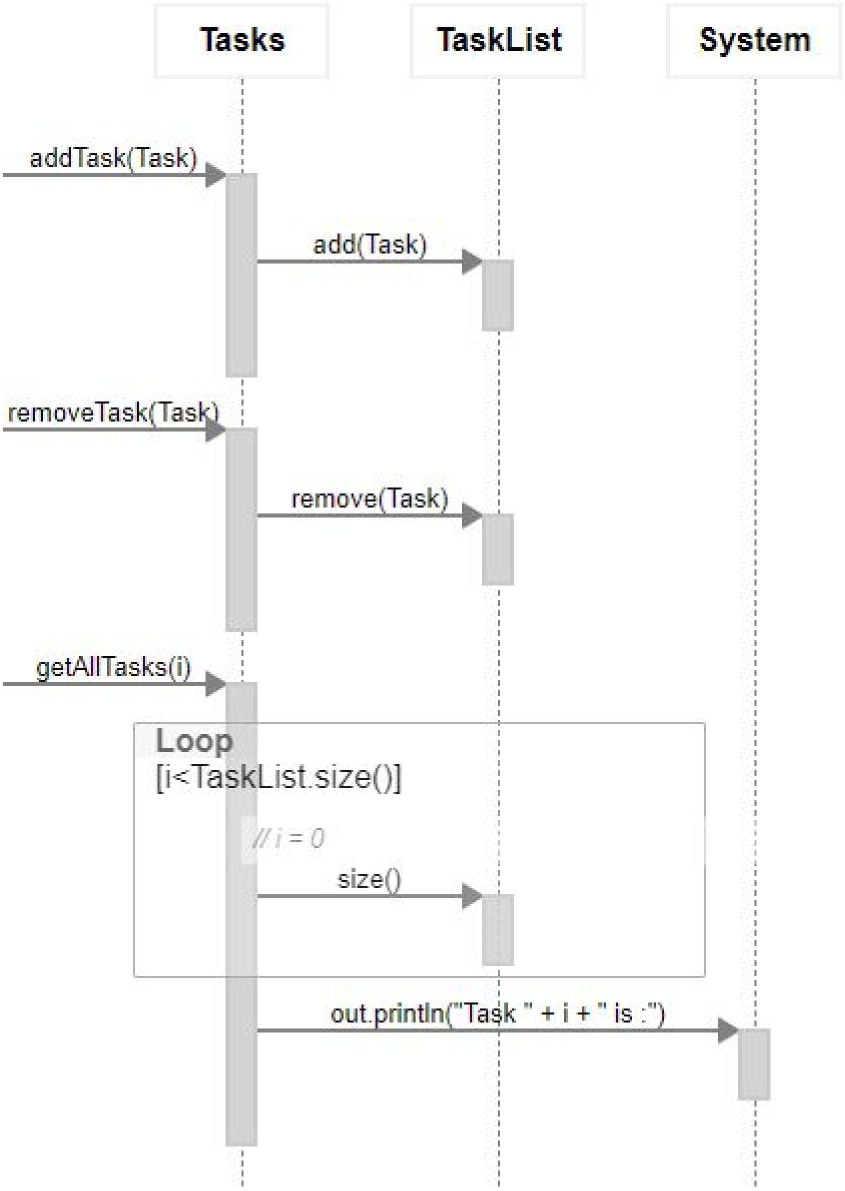

Tasks
TaskList
System
addTask(Task)
add(Task)
removeTask(Task)
remove(Task)
getAllTasks(i)
Loop
[i<TaskList.size()]
// i = 0
size()
out.println("Task " + i + " is :")

Figure 20

4.4 User Interface

The purpose of the user interface is to make an output for the user that is simple and easy and comfortable to use and enables to follow the predetermined activity diagrams.

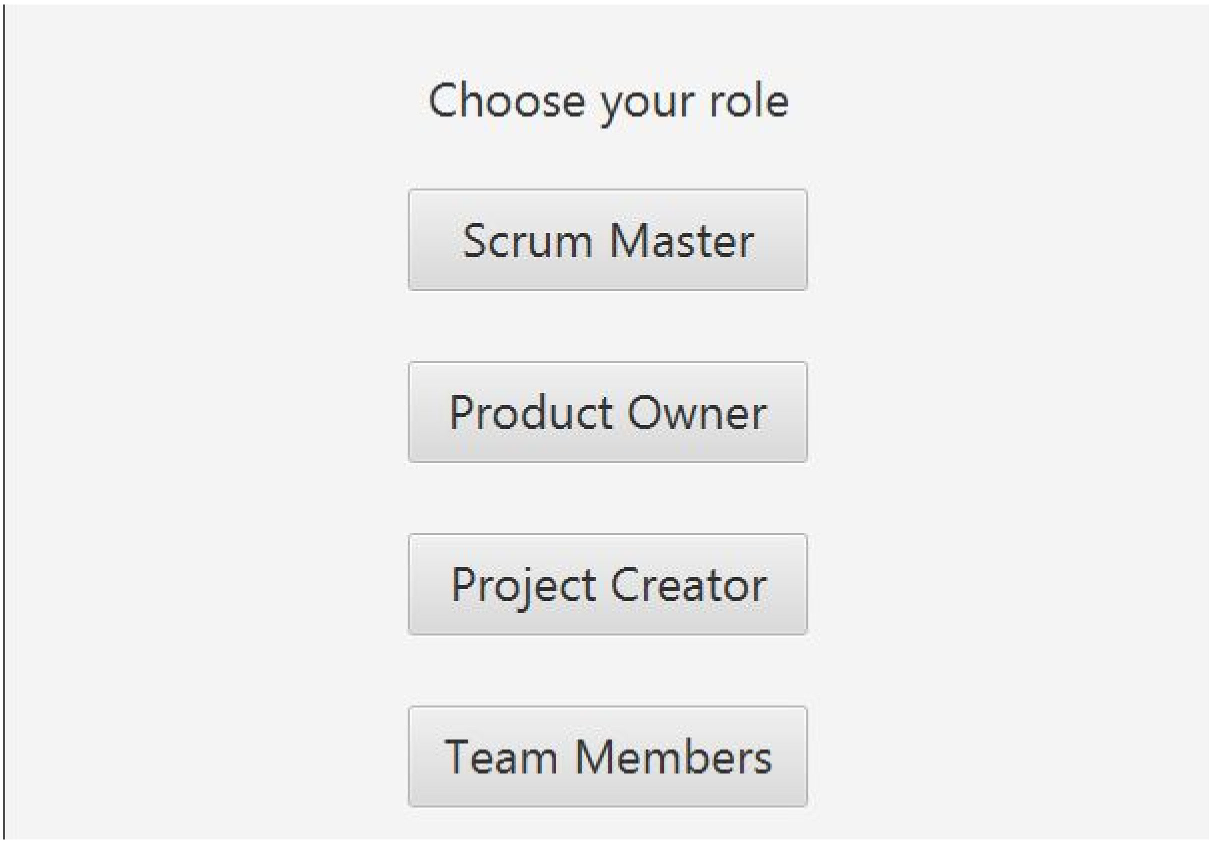
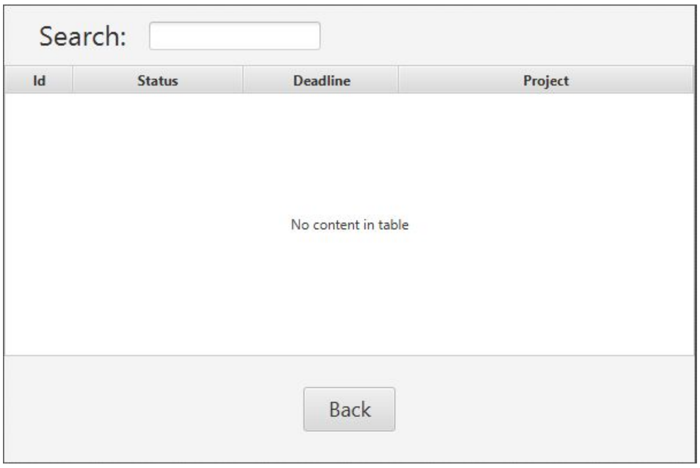


Figure 21

Firstly, the user should choose his role. Then, depending on which one he has chosen, he goes to another window, where he can choose his next actions.
Product owner can only view and search projects so his window looks like this:



Search:

Id	Status	Deadline	Project

No content in table

Figure 22

Project Creator can manage projects, requirements, tasks and add or remove members.

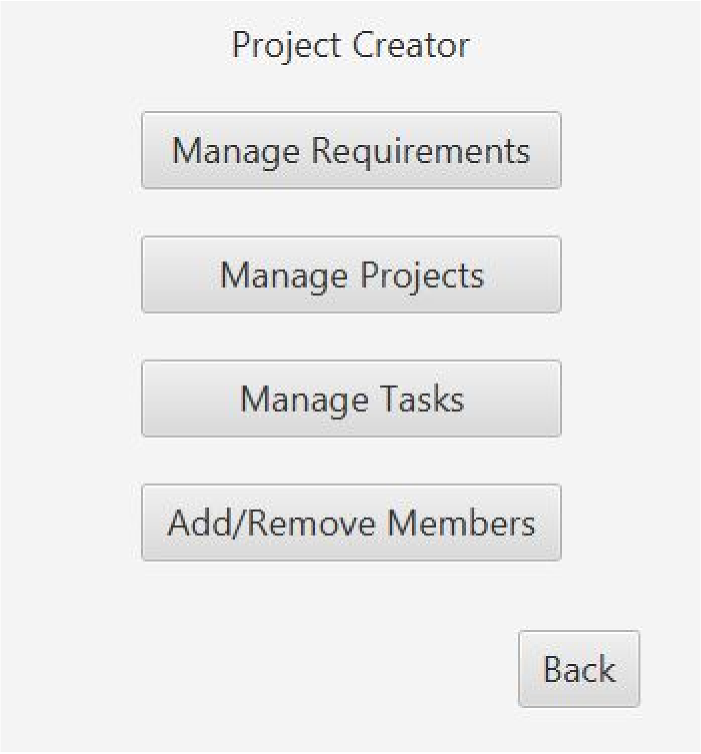


Figure 23

According to which operation he chooses, there will be shown different windows with relevant actions.

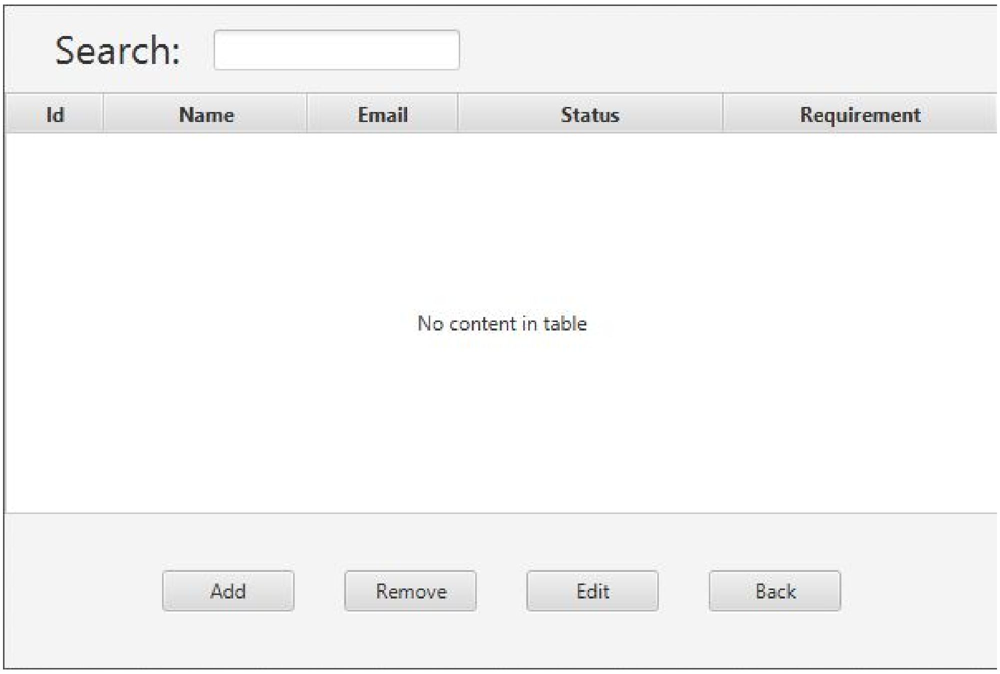


Figure 24

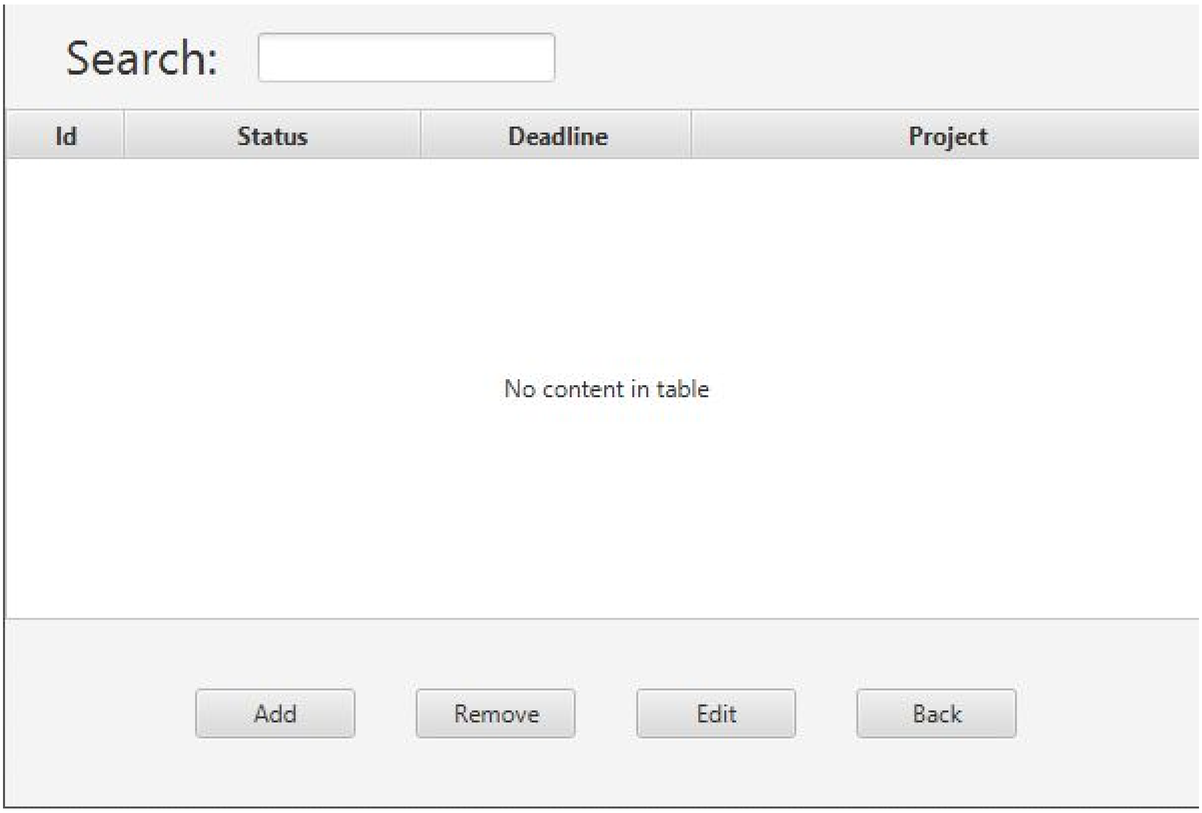


Figure 25

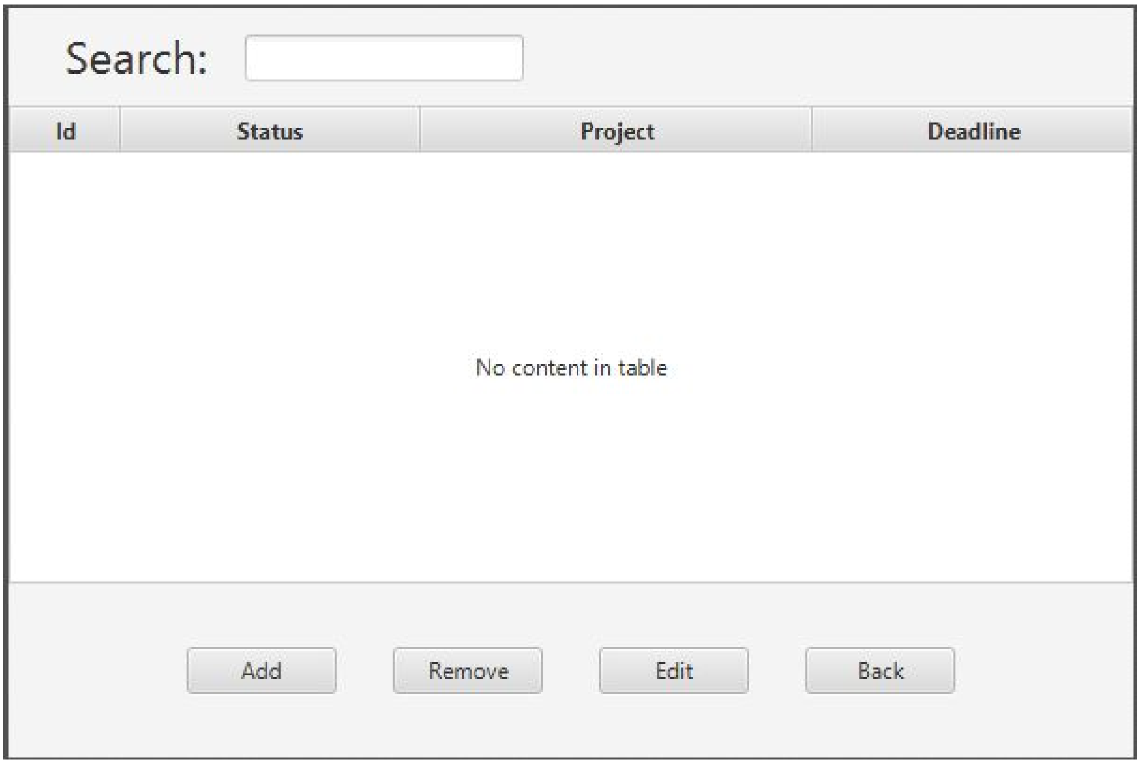


Figure 26

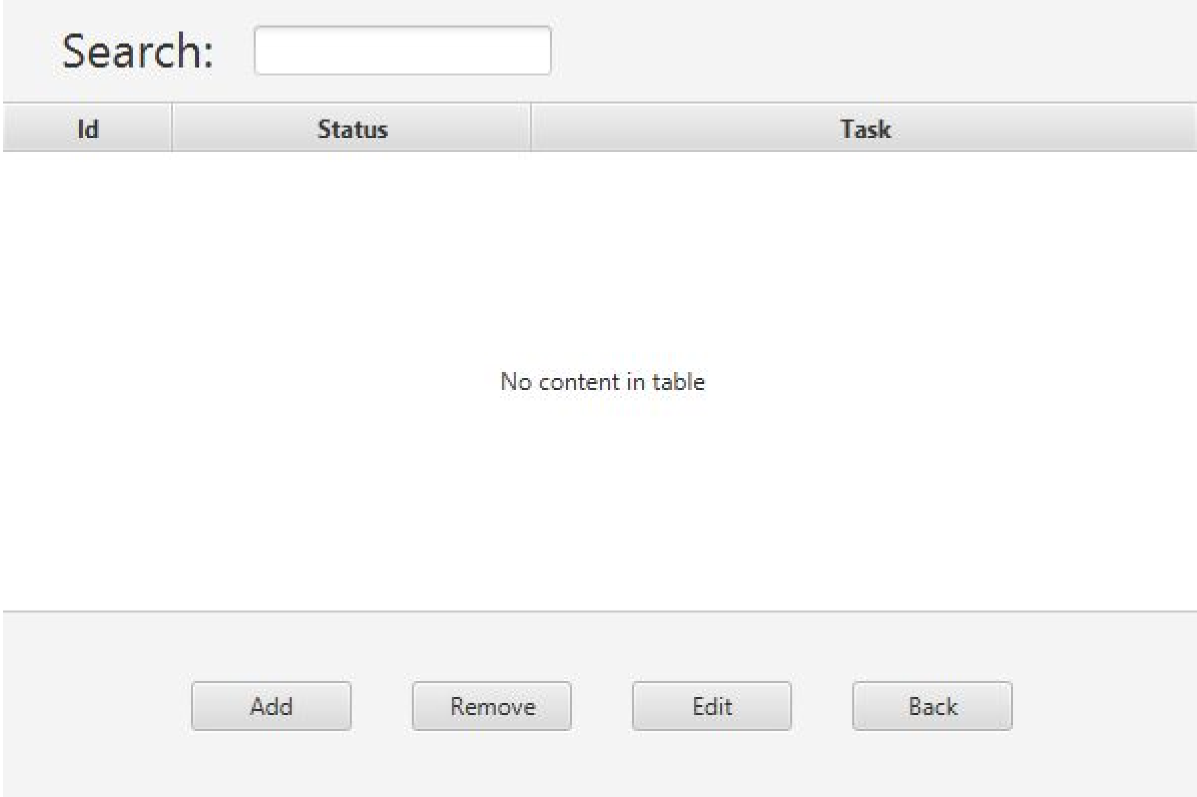


Figure 27

There also will be shown different windows for add and edit with corresponding actions.

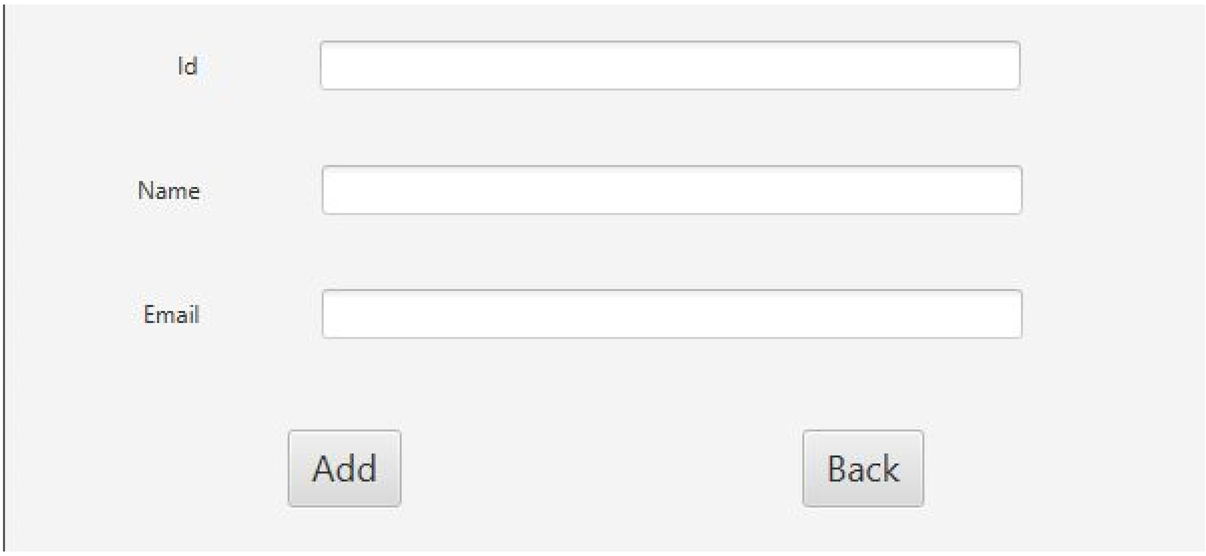


Figure 28

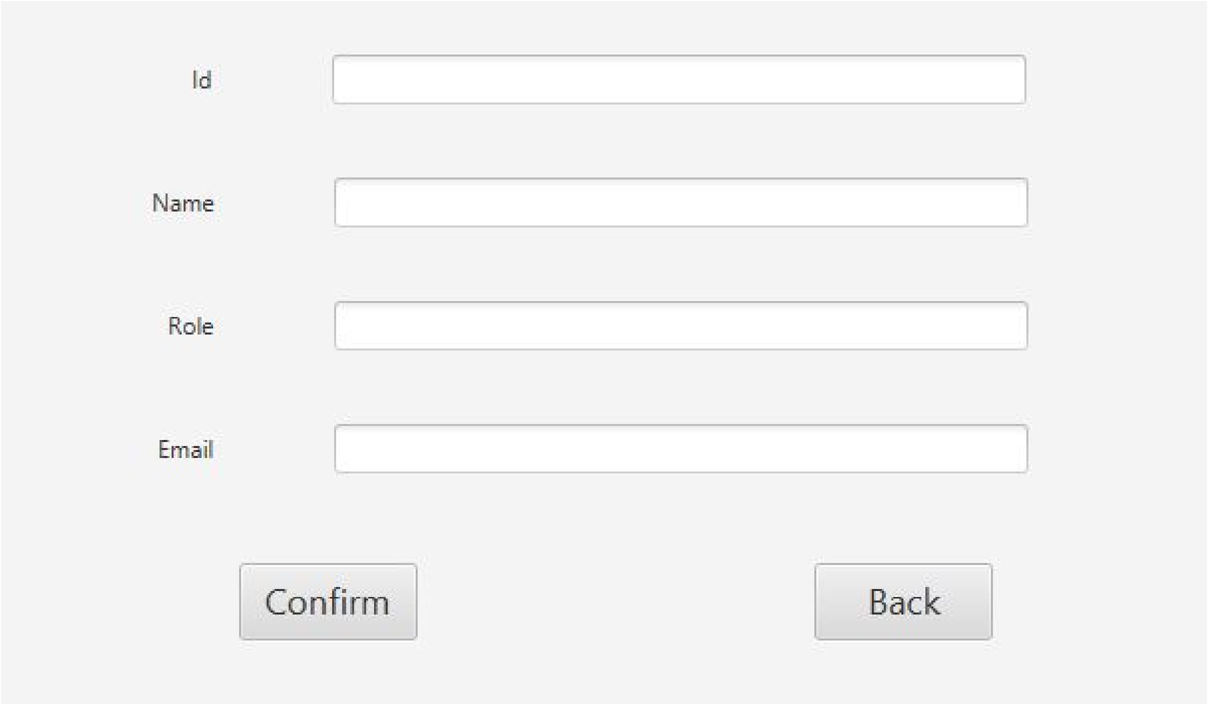


Figure 29

Scrum Master can only manage requirements and tasks with corresponding actions for any of options

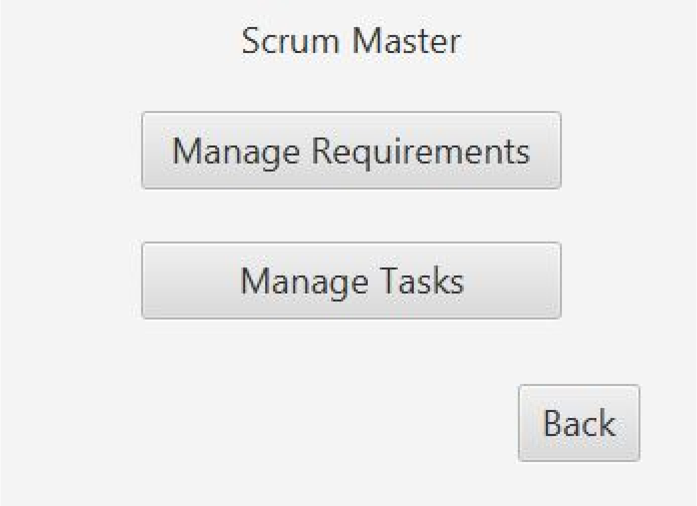
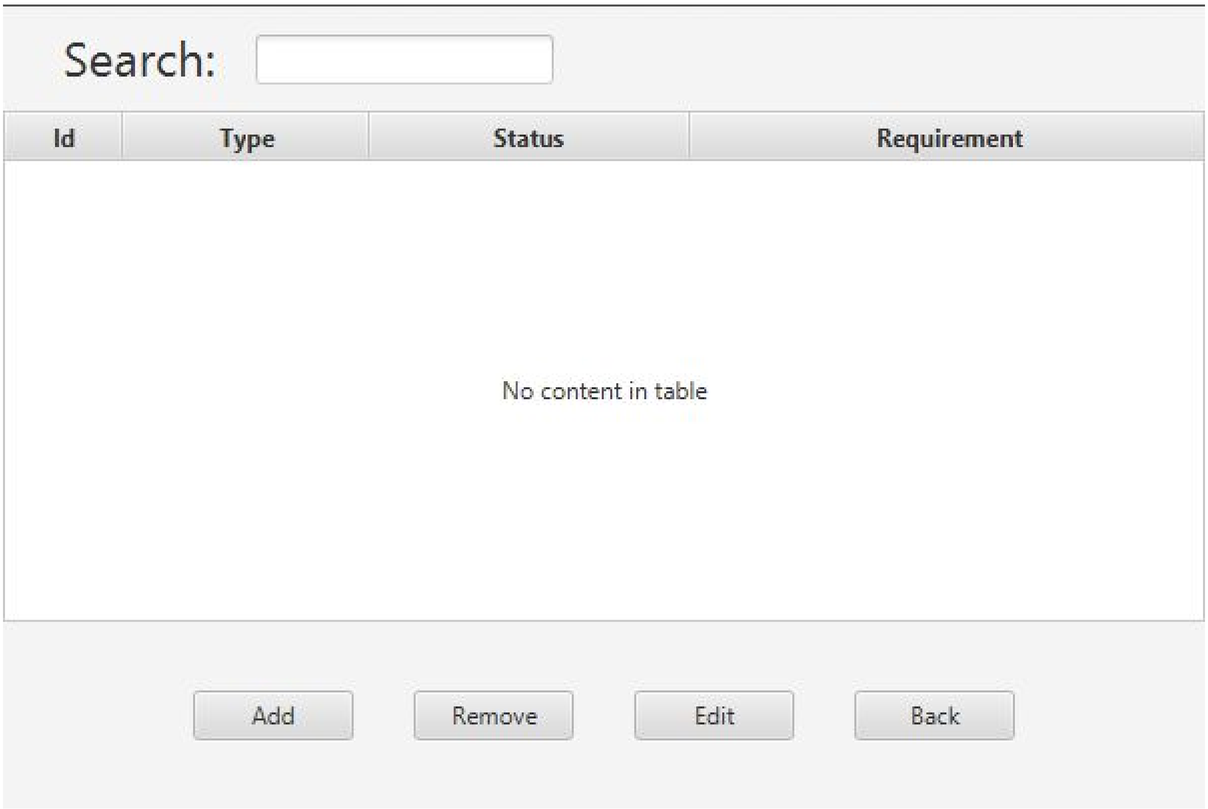


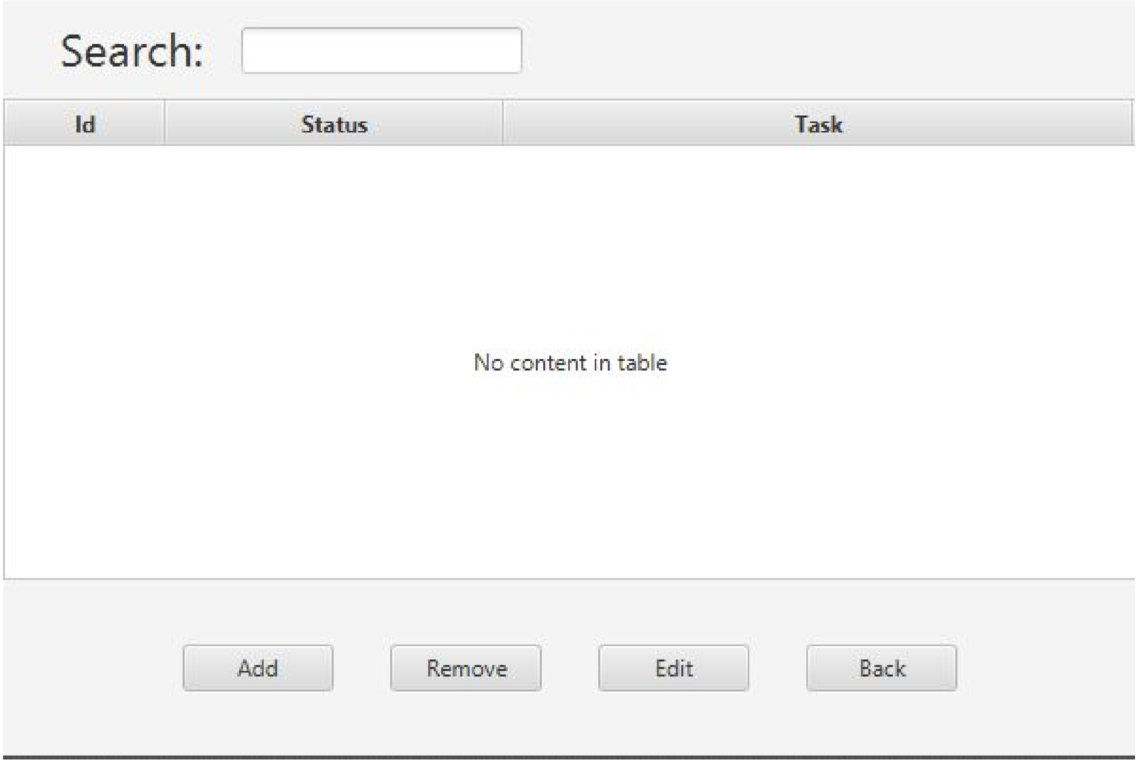
Figure 30



Id	Type	Status	Requirement

No content in table

Figure 31



Id	Status	Task

No content in table

Figure 32

Team member can manage, view and search projects.

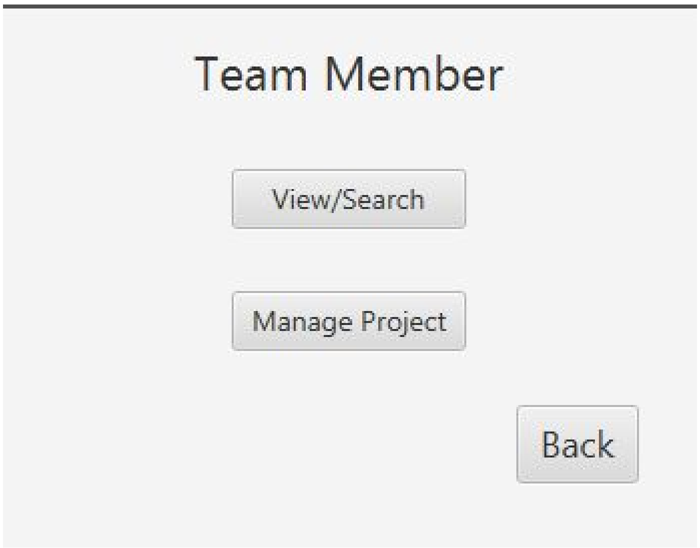
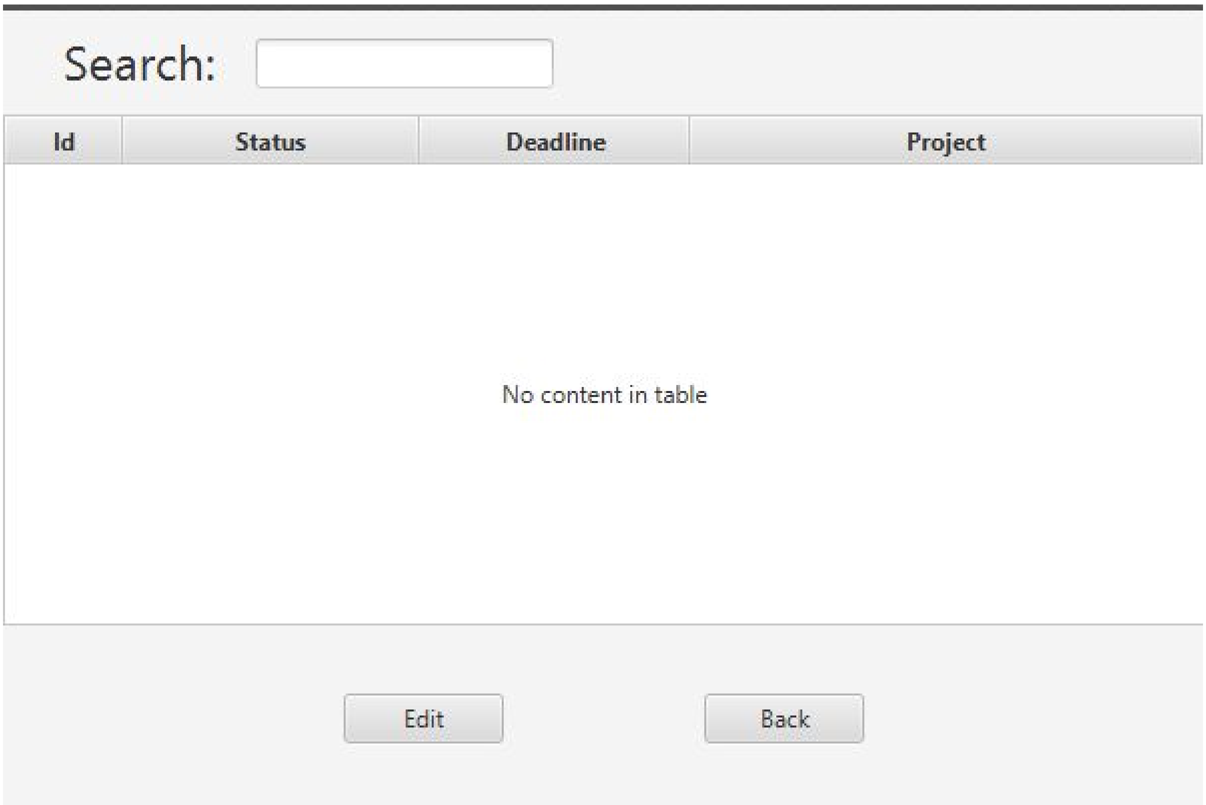


Figure 33

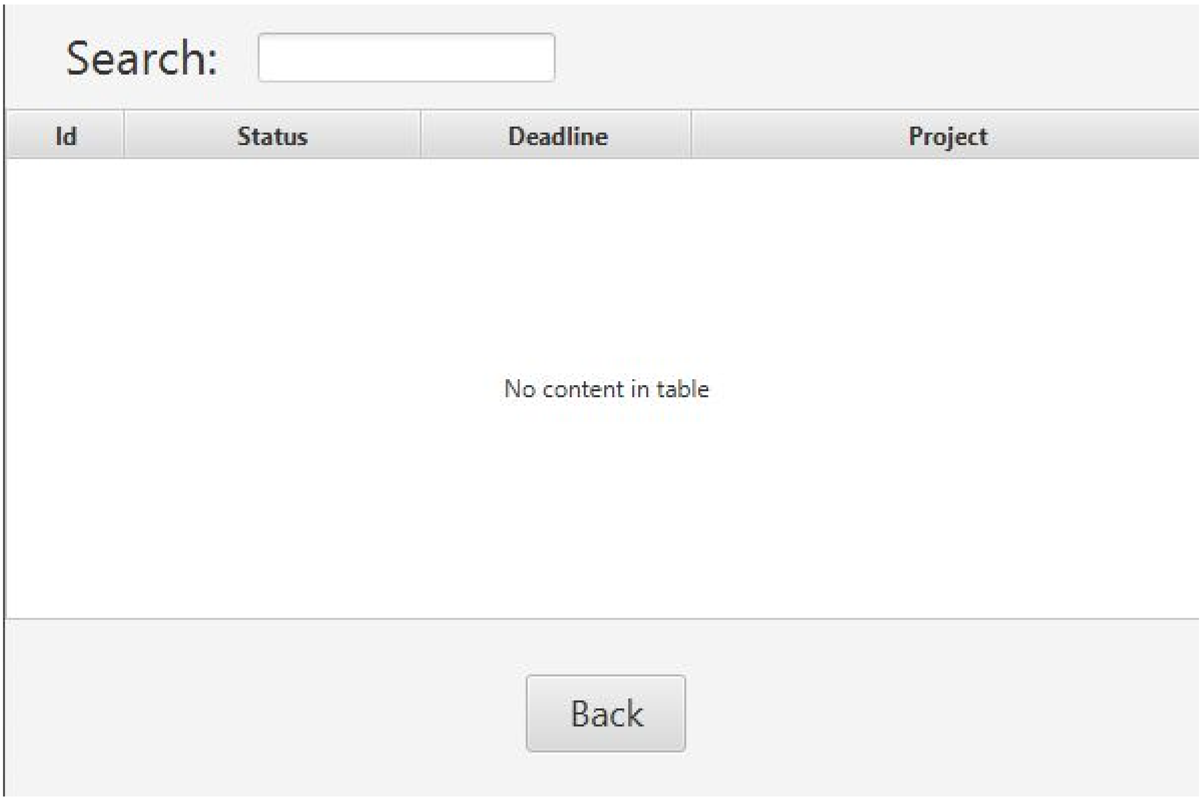
Manage project window differs from the previous one due to less actions that team members can make.



Id	Status	Deadline	Project

No content in table

Figure 34



Id	Status	Deadline	Project

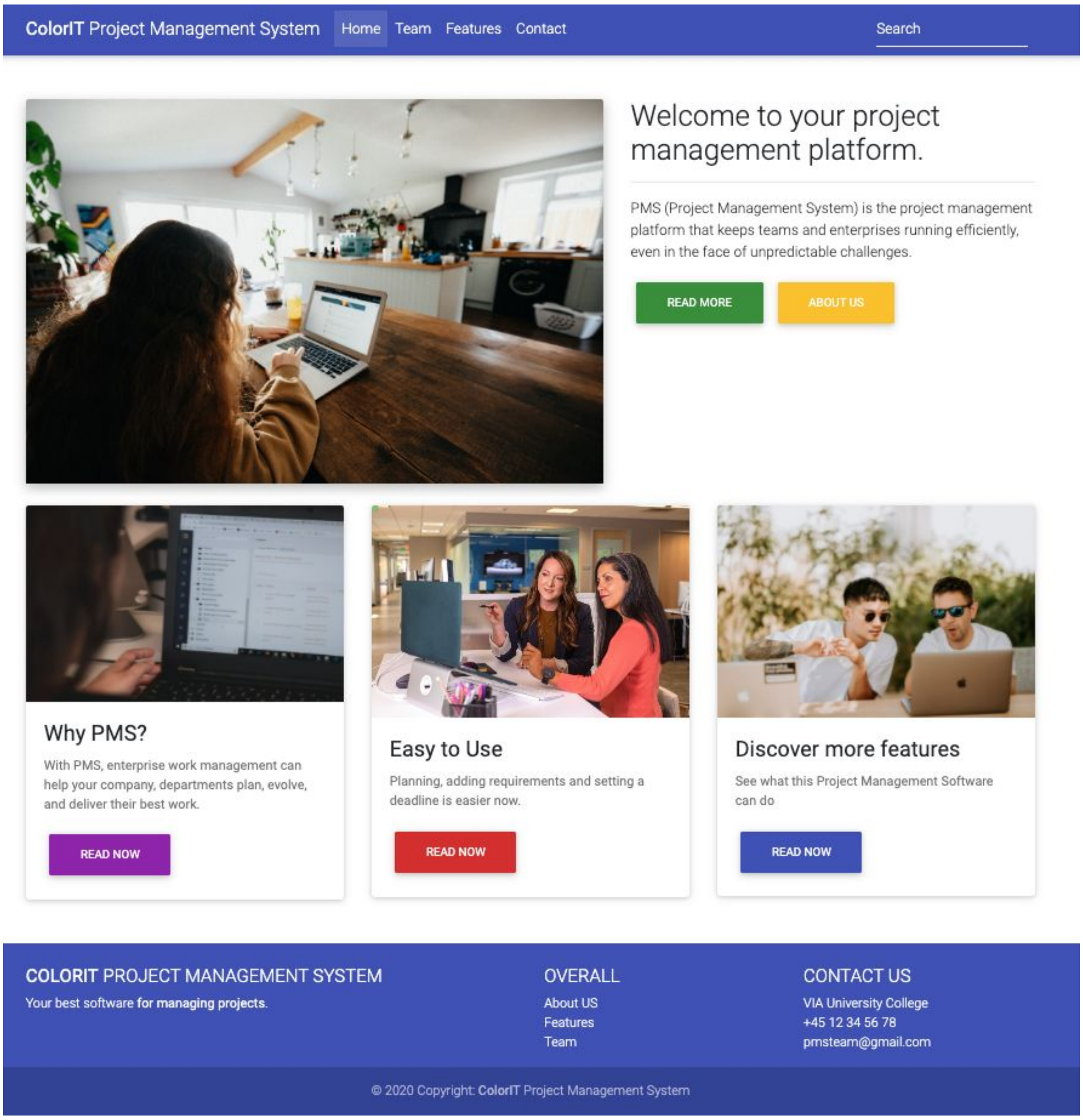
No content in table

Figure 35

4.5 Website

The final result of the website design is made to be simple and navigable by each user who uses the website. The website is made in a modern way by it's material design elements such as square corners and flat colors. Also the color schemes of the buttons are related to the main's customer theme: colored because of the name of the company, ColorIT.

With such a concept in mind, the website is fulfilling the needs of presenting the product and also the features of the software such as navigation and the roles which can be controlled.



Welcome to your project management platform.

PMS (Project Management System) is the project management platform that keeps teams and enterprises running efficiently, even in the face of unpredictable challenges.

Why PMS?

With PMS, enterprise work management can help your company, departments plan, evolve, and deliver their best work.

Easy to Use

Planning, adding requirements and setting a deadline is easier now.

Discover more features

See what this Project Management Software can do

COLORIT PROJECT MANAGEMENT SYSTEM

Your best software for managing projects.

OVERALL

About US
Features
Team

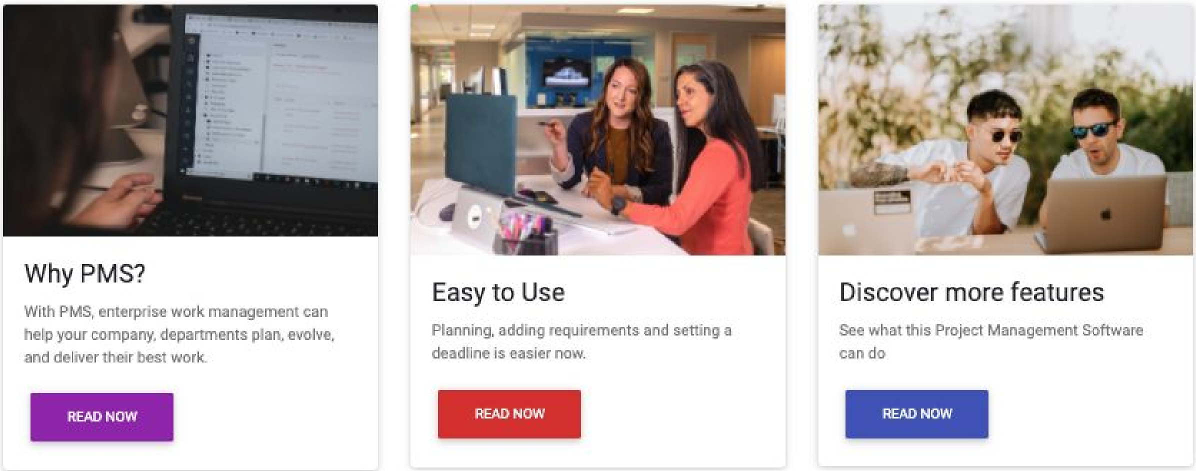
CONTACT US

VIA University College
+45 12 34 56 78
pmsteam@gmail.com

© 2020 Copyright: ColorIT Project Management System

Figure 36

On the main page called the “Home” page, the customer can see a generous menu that directs to different pages like Team which is showing up the team that developed the final product and and the About Us page shows information about the educational background of the developers. Also there is a page called “Contact Us” that redirects to a form that needs to be filled to actually send a message to the team for further information about the project, reporting bugs and future collaborations. The contact details can be found on the footer as well.



Why PMS?

With PMS, enterprise work management can help your company, departments plan, evolve, and deliver their best work.

Easy to Use

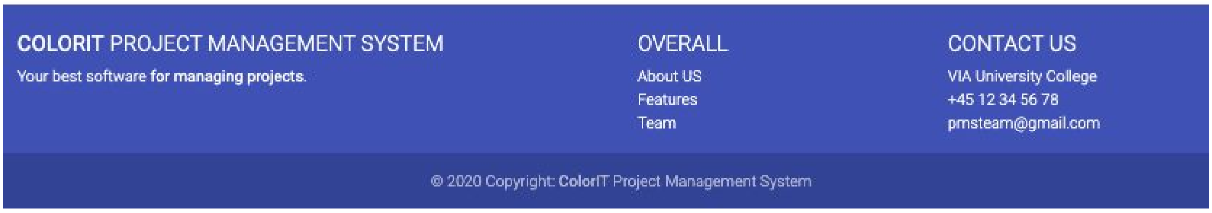
Planning, adding requirements and setting a deadline is easier now.

Discover more features

See what this Project Management Software can do

Figure 37

This section of the page highlights the main features and is promoting the product on different scenarios. It overviews the main characteristics and it also contains buttons that direct to the “Features” page where you can learn more about the product.



COLORIT PROJECT MANAGEMENT SYSTEM

Your best software for managing projects.

OVERALL

About US
Features
Team

CONTACT US

VIA University College
+45 12 34 56 78
pmsteam@gmail.com

© 2020 Copyright: ColorIT Project Management System

Figure 38

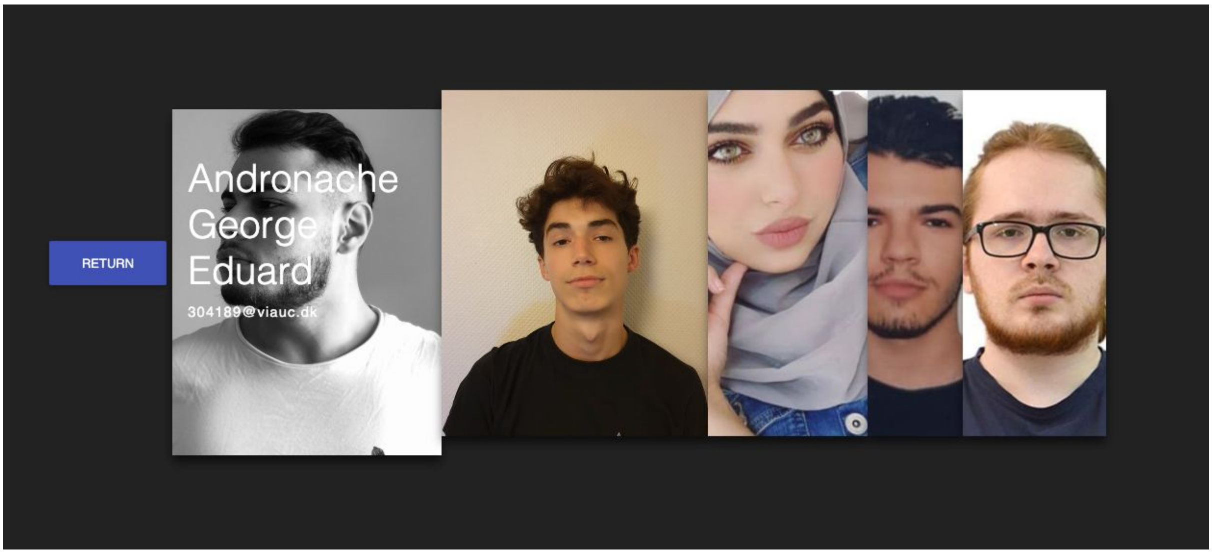


Figure 39

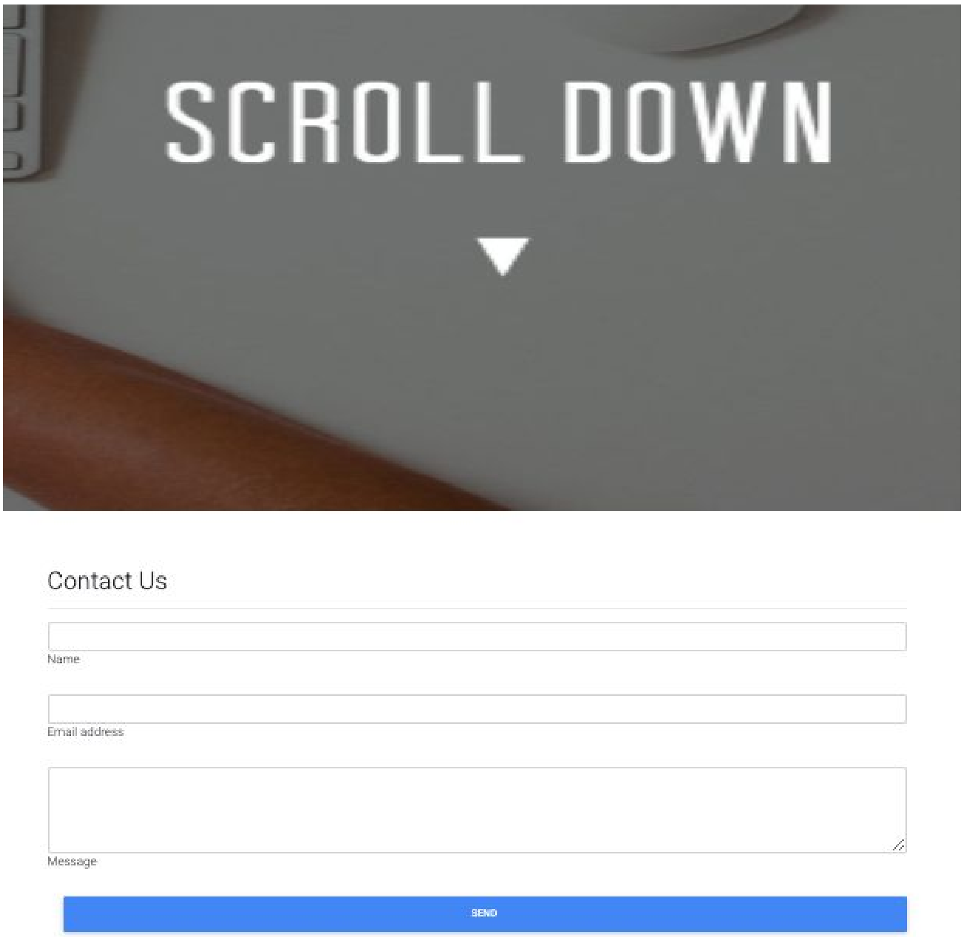
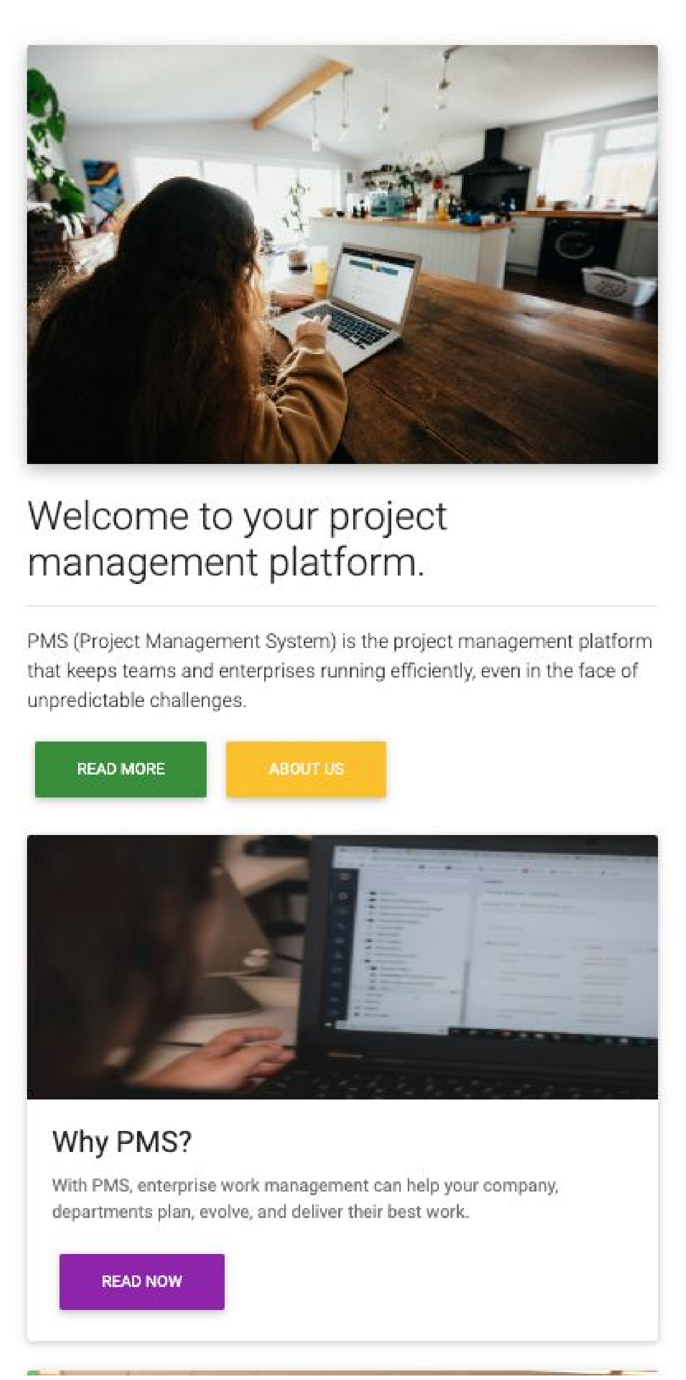


Figure 40

Due to implementation of MaterialDesign which uses the latest version of Bootstrap (Bootstrap 5) the website is fully responsive and it can be viewed on mobile devices such as tablets and smartphones, not only on desktop browsers.



Welcome to your project management platform.

PMS (Project Management System) is the project management platform that keeps teams and enterprises running efficiently, even in the face of unpredictable challenges.

Why PMS?

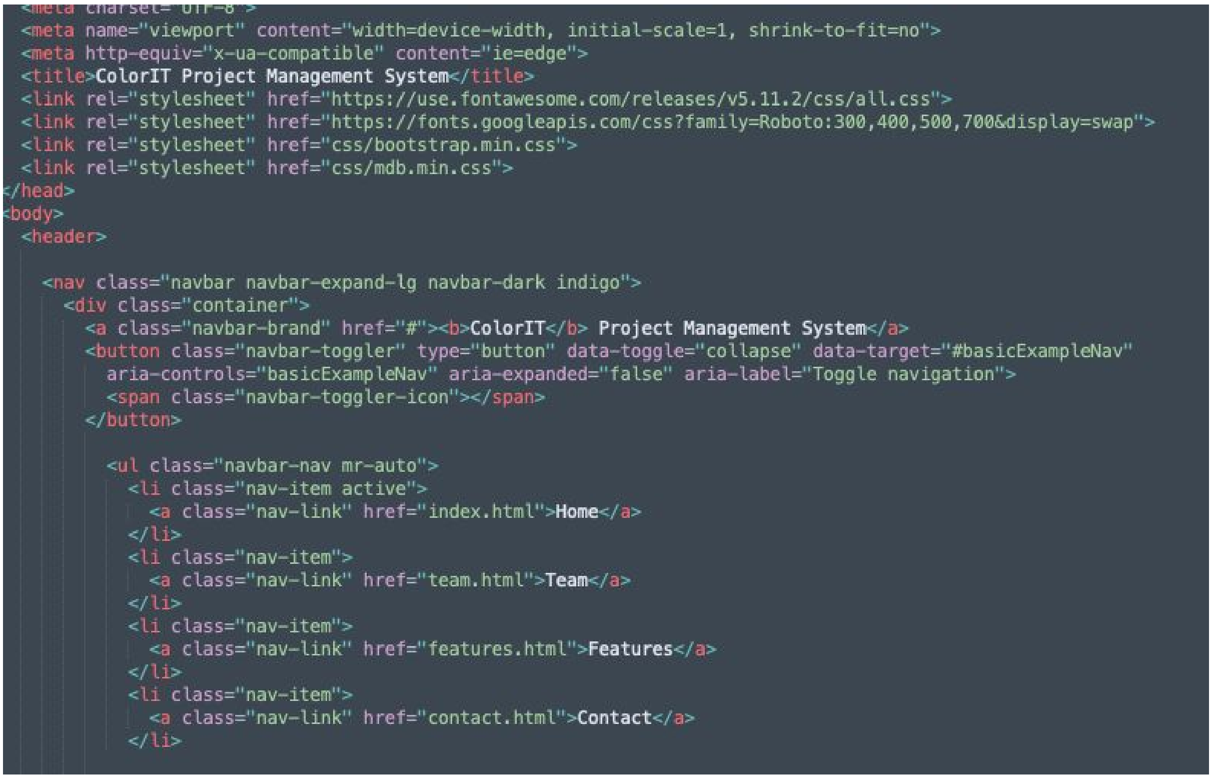
With PMS, enterprise work management can help your company, departments plan, evolve, and deliver their best work.

Figure 41

5.Implementation

The purpose of the implementation section is to explain interesting code snippets.

5.1 Website

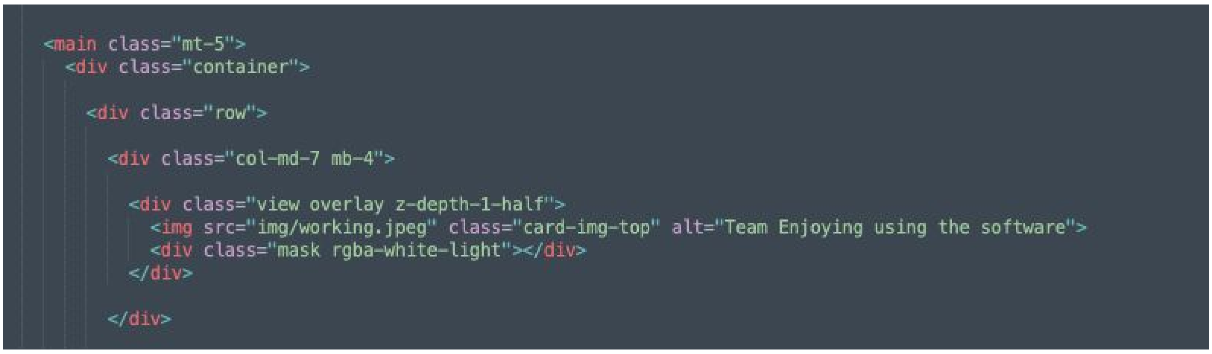


```
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta http-equiv="x-ua-compatible" content="ie=edge">
  <title>ColorIT Project Management System</title>
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.11.2/css/all.css">
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap">
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link rel="stylesheet" href="css/mdb.min.css">
</head>
<body>
  <header>

    <nav class="navbar navbar-expand-lg navbar-dark indigo">
      <div class="container">
        <a class="navbar-brand" href="#"><b>ColorIT</b> Project Management System</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#basicExampleNav"
          aria-controls="basicExampleNav" aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>

          <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
              <a class="nav-link" href="index.html">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="team.html">Team</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="features.html">Features</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="contact.html">Contact</a>
            </li>
```

Figure 42

In order to make the website responsive, as said before, MaterialDesign was implemented using Bootstrap 5 and it’s specific style given by the keywords tag to make the CSS more organised and more efficient in terms of organising.

Before the coding process of the website, the design was done using wireframes in Balsamiq Mockup. The concept was planned before to make the website usable on other devices other than desktop. (Figure 40)



```
<main class="mt-5">
  <div class="container">

    <div class="row">

      <div class="col-md-7 mb-4">

        <div class="view overlay z-depth-1-half">
          <img src="img/working.jpeg" class="card-img-top" alt="Team Enjoying using the software">
          <div class="mask rgba-white-light"></div>
        </div>

      </div>
```

Figure 43

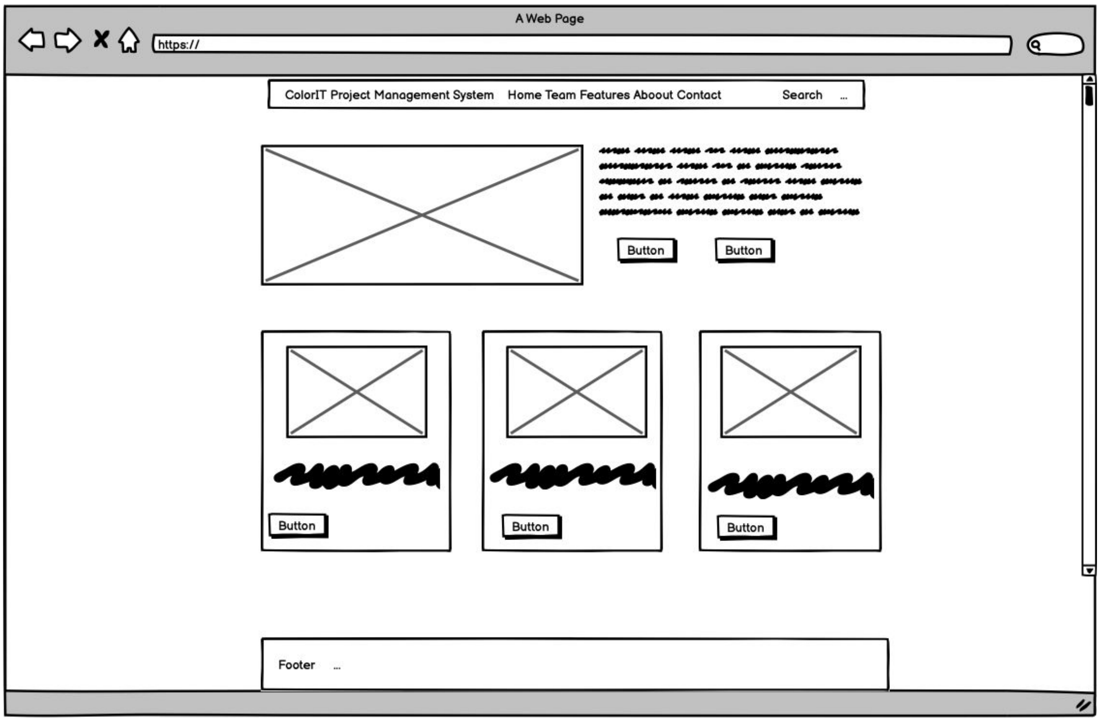


Figure 44

Each page was made with the concept in mind and after that implemented in code.

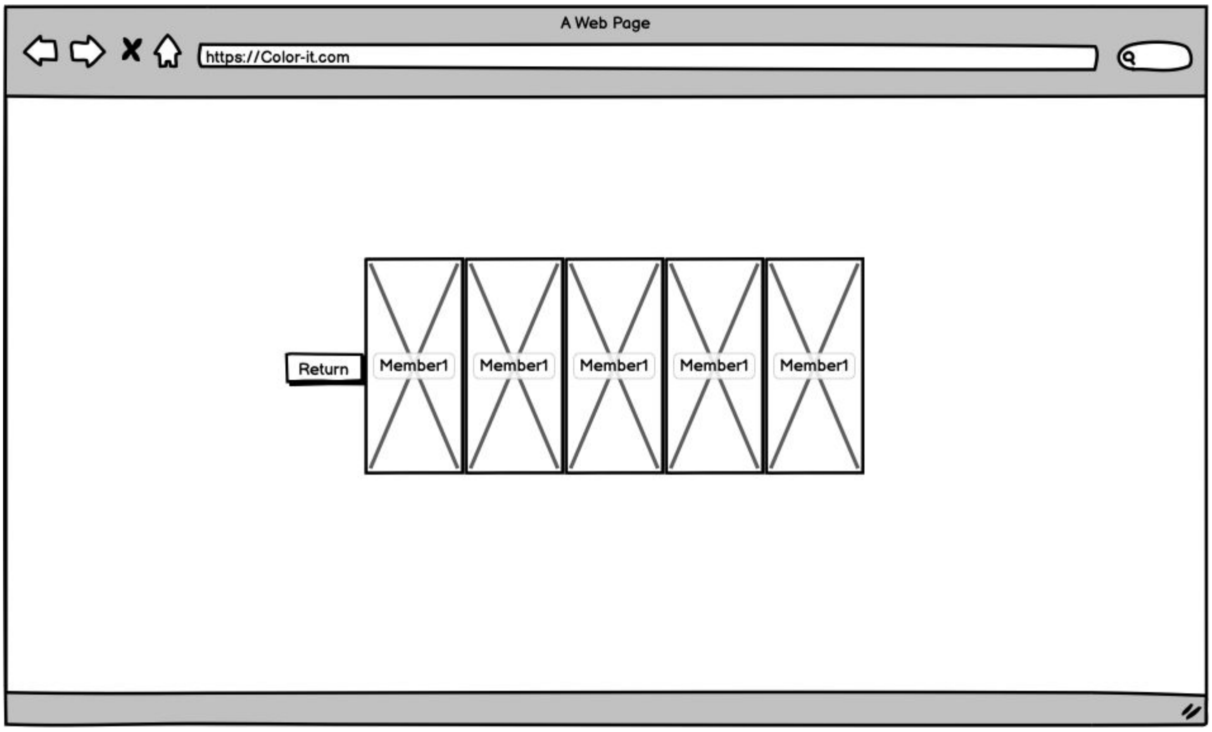


Figure 45

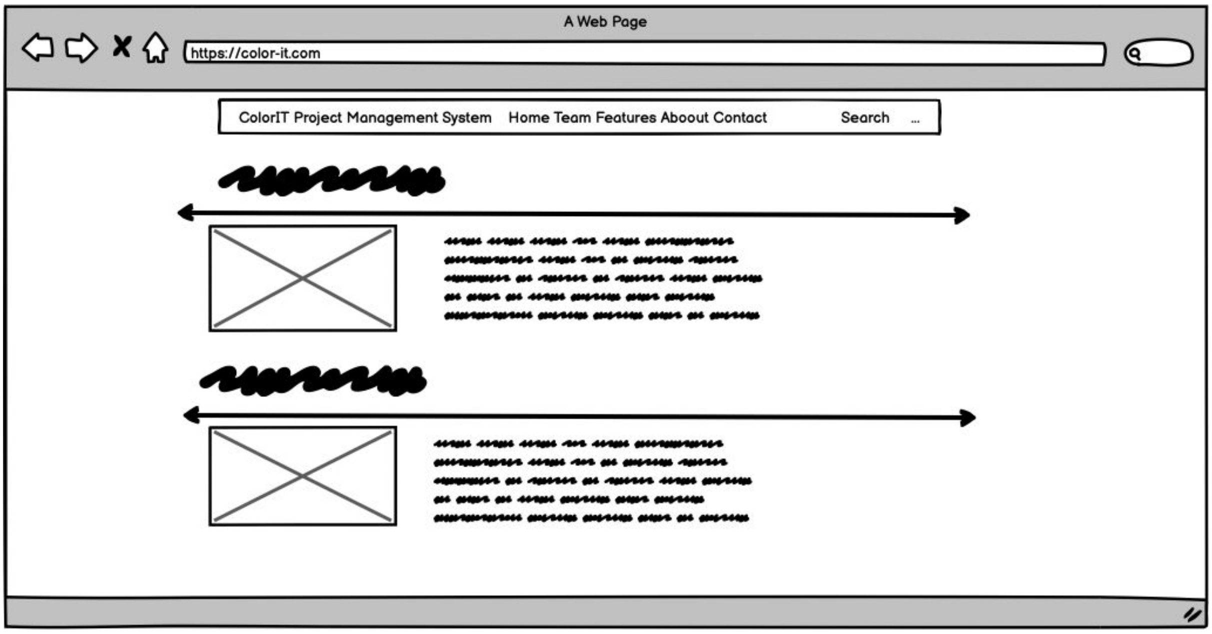

A Web Page
https://color-it.com
ColorIT Project Management System
Home Team Features Aboout Contact
Search
...

Figure 46

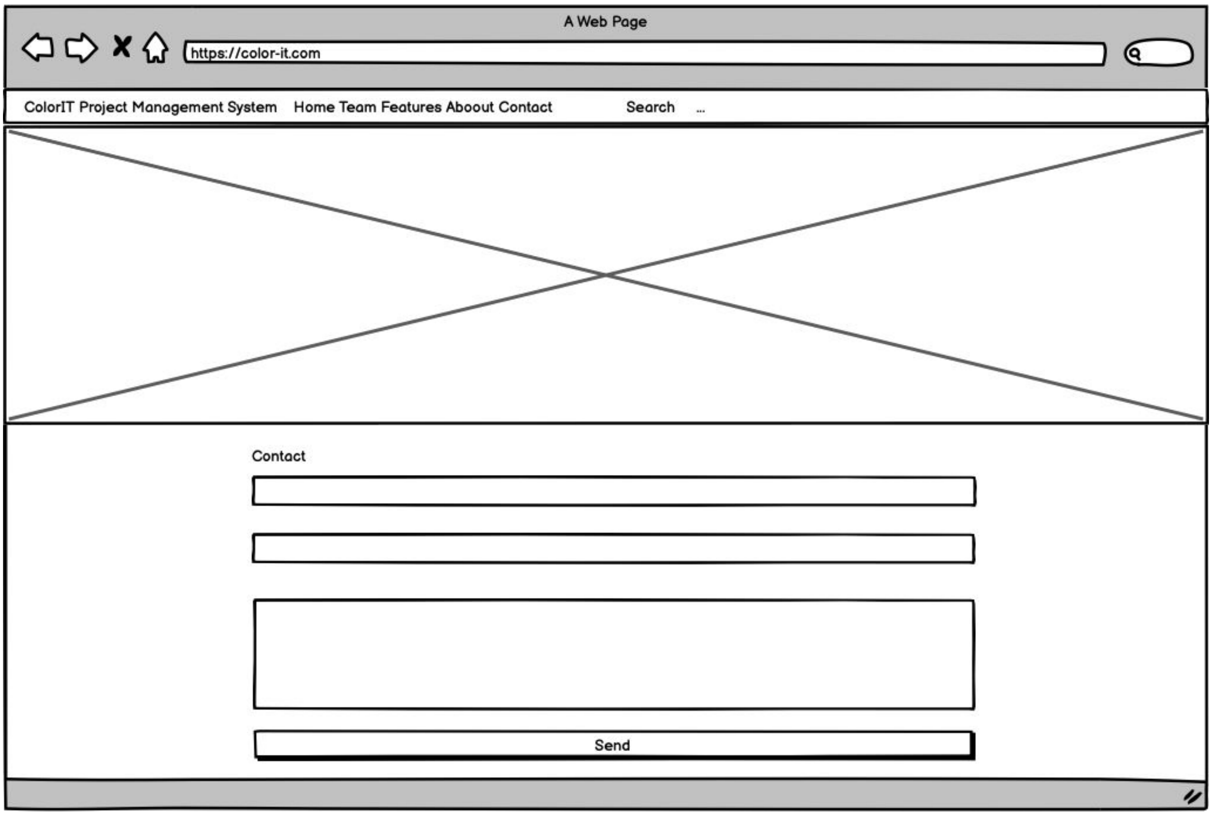

A Web Page
https://color-it.com
ColorIT Project Management System
Home Team Features Aboout Contact
Search ...
Contact
Send

Figure 47

Mobile friendly user interface was adapted to each customer who wants to visit the website without worrying of any possible problems.

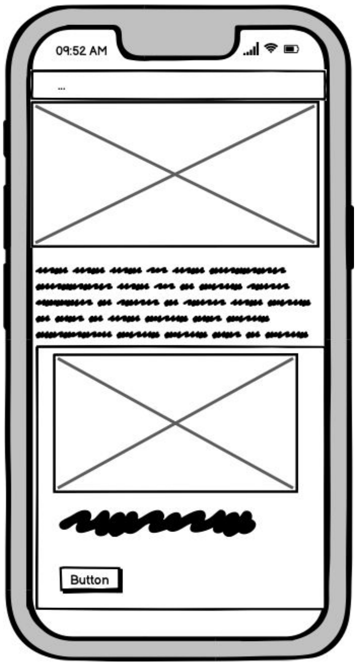


Figure 48

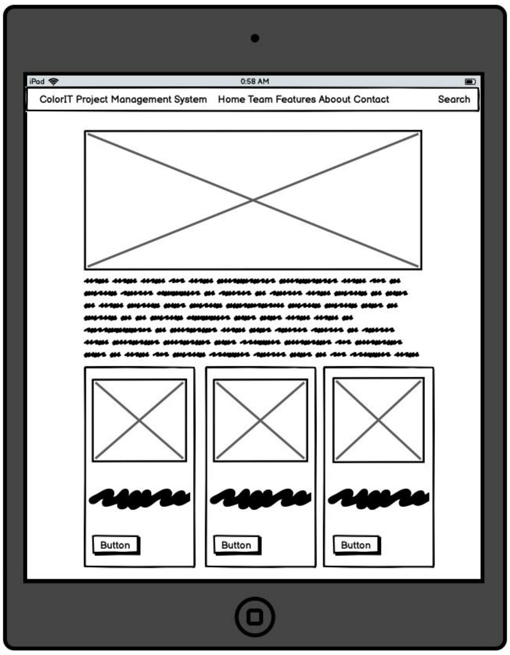

iPad
0:58 AM
ColorIT Project Management System
Home Team Features Aboout Contact
Search
Button
Button
Button

Figure 49

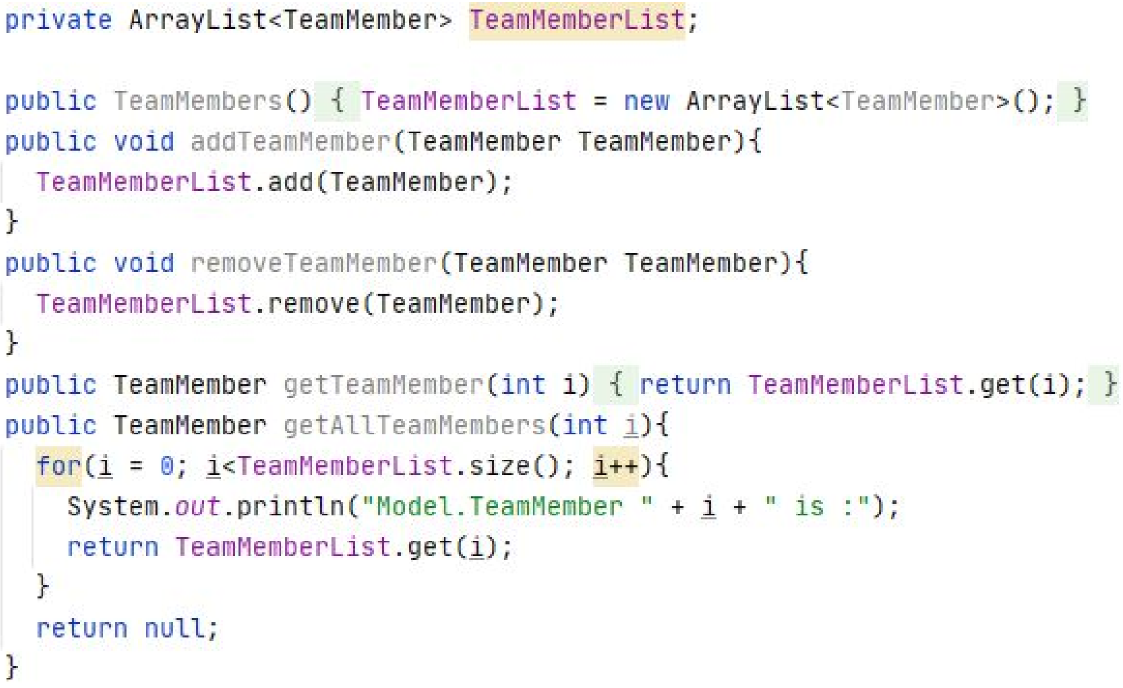
5.2 Java Program

The code was based not only on documentation that we did such as Color-It Analysis, project Description, etc, but on templates and interview too. In this way we could make a comprehensive structure of the code so that there are more actions for every role.

Despite the fact that the program appeared to be very vast, it was done in the simplest way in order to have no problems with implementation.

The most important classes are: Requirements, Team Members and Tasks because the majority of the roles have access to these classes.

Team Member is not a difficult class but it is needed to get track of team members and it is used to manage team members.

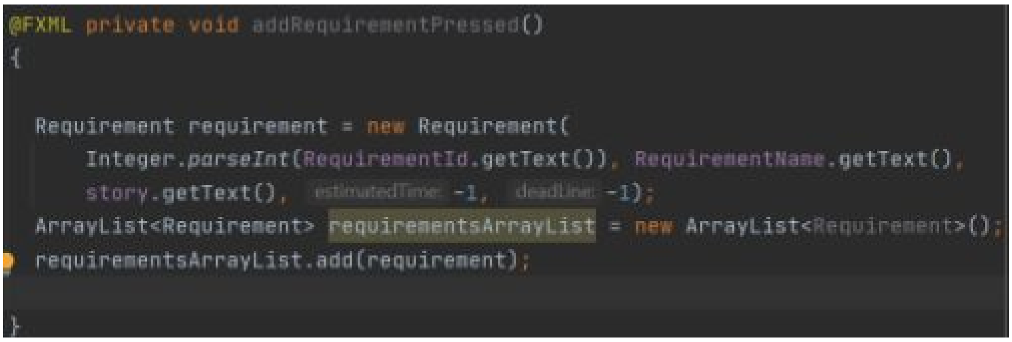


```java
private ArrayList<TeamMember> TeamMemberList;

public TeamMembers() { TeamMemberList = new ArrayList<TeamMember>(); }
public void addTeamMember(TeamMember TeamMember){
  TeamMemberList.add(TeamMember);
}
public void removeTeamMember(TeamMember TeamMember){
  TeamMemberList.remove(TeamMember);
}
public TeamMember getTeamMember(int i) { return TeamMemberList.get(i); }
public TeamMember getAllTeamMembers(int i){
  for(i = 0; i<TeamMemberList.size(); i++){
    System.out.println("Model.TeamMember " + i + " is :");
    return TeamMemberList.get(i);
  }
  return null;
}
```

Figure 50

The biggest but not the hardest class is Task. It evolves around the main idea of the project: getting tasks done.It contains information about tasks including its deadline, descrition, team member etc.

In the controller part, there was made a method called ”addRequirementPressed()”. This method adds requirements to an existing arraylist where they are stored.



```
@FXML private void addRequirementPressed()
{

  Requirement requirement = new Requirement(
      Integer.parseInt(RequirementId.getText()), RequirementName.getText(),
      story.getText(), estimatedTime: -1, deadline: -1);
  ArrayList<Requirement> requirementsArrayList = new ArrayList<Requirement>();
  requirementsArrayList.add(requirement);

}
```

Figure 51

6.Test

The purpose of the test part is to describe the result of the testing and to verify if the content of the requirements section has been fulfilled.

Not all parts of the program were tested because of time constraint. The table below shows the test of the most important use case: Manage Tasks.

Precondition	An interview with the customer, having a list of project's requirements				
Manage Tasks					
Step	Test	Expected Result	Actual Result	Status	Notes
1	Enter Project's title	Title entered	As expected	Passed	-
2	If illegal characters are present go to1)	Character check	As expected	Passed	-
3	Set project requirements	Project requirements set	As expected	Passed	-
4	Set the requirement type	Requirement set	Not working as expected	Partially Passed	Needs figuring out the implementation
5	Set the project's deadline	Deadline set	As expected	Passed	-
6	If an incorrect date is entered go to 5)	Incorrect date	Currently not working	Not Passed	Needs figuring out the

		recognised			implementation
7	Assign Team Members	Team members assigned	As expected	Passed	-
8	Assign roles to team members (Scrum, Owner, Team Member)	Roles assigned	As expected	Passed	-
9	Enter estimated time	Estimate time entered	Not working as expected	Not Passed	Needs figuring out the implementation
10	If a negative number is entered go to 9)	Negative number shown	Not working as expected	Not Passed	Needs figuring out the implementation
11	Set project status "not started"	Project Status set	Not working as expected	Not Passed	Needs figuring out the implementation

7.Results

As a result of the system, there was created a program that can manage projects for a company in order to make the following of the project route easier.

Requirements	Status	Solution if not implemented
1) As a project creator, I want to create a new project ,to start the project work;	Done	
2) As scrum master, I want to be able to check every project status such that I want to keep track;	Done	

3) As scrum master, I want to be able to assign tasks to each team member so that I know every member is responsible for a task;	Done	
4) As a scrum master, I want to add and remove requirements so that I can keep the track of the project	Done	
5) As a scrum master, I want to add and remove tasks so that I can sort them out whether they are good or not;	Done	
6))As a team member I want to search about a specific task by id or name ,to manage it or just show it.	Done	
7) As scrum master, I want to handle the process reports by searching the deadline date and if there are all the methods when the requirements are done such that I can check if all is going right;	Done	
8) As a project creator, I want to be able to manage projects.	Done	
9) As a team member, I want to be able to view projects such that I can have an overview at them;	Done	
10) As a customer I want to search for a project by name or id , so I can see all the project details and task status . .	Done	
11) As a scrum master, I want to edit total hours for a task so that everyone keep tracking the work progress;	Done	
12) As a product owner I want to edit a type of requirement (functional, not functional, project requirement), so that they will be	Done	

sorted out.		
13) As a scrum master, I want to add deadlines for existing tasks.	Done	
14) As a product owner,I want to approve the requirement when it's done, such that I can see all the projects that are not done yet.	Undone	Further development and documentation about the subject will be involved to create the method
15) As a scrum master, I want to edit available tasks in case something changes in a project or in the case that customer changes his preferences;	Done	
16) As a customer I want to have access to projects, tasks, time spent on them, so I can see the progress and how long it took to be ready;	Done	
17) The system must be compatible with each device that is running Windows 7, Windows 8, Windows 8.1 and Windows 10;	Done	
18) As a scrum master, I want to assign tasks to team members so that they can handle the projects.	Done	

8 Conclusion

The system did not fulfil all requirements but the requirements that were not fulfilled were replaced or implemented in another way so that there are no left requirements.

After the initial interview with Mr.Color, the customer's requirements and desires were analysed and put together in an Analysis Document File which contains use case diagrams, use case descriptions, a domain model, a class diagram with all the variables and methods with all the related relations between them.

The system's design concept was implemented with JavaFX to create the GUI's software. The main part of the software was created using the SceneBuilder software to create windows of the project by generating automatically the .FXML files.

The website was made with a modern framework called Bootstrap 5 which makes the CSS code more easy to implement and simplifies the coding part. At the same time, this framework provides a modern design because within this plugin a material design style was implemented, making the site easy to navigate and easy to understand for the customer. The website contains essential parts like features of the software, contact us page and a home page with all the important information.

The final software was tested manually to see if the methods are working.

9.Project future

As possibilities are concerned, there are still a lot of improvements that could be done in order to make the managing system easier and better

Firstly, there are still a lot of bugs in the program that should have been fixed but due to lack of time they were not fixed.

Secondly, despite the fact that the program is being easy and simple, it could be more optimised. For example, the birthday or email methods are not needed and deleting them will help the program to run faster.

Also it will be a good idea to improve user interface into a much more pleasant user experience and user interface.

In the future, more methods will be implemented and it will make the software fully functional.

10.Sources of information

- Studienet.via.dk.(2018).*Engineering_Project_Metodology.*[online] Available at: https://studienet.via.dk/projects/Engineering__project_methodology/General/Guidelines/4-2018%20Project%20Report%20(Appendix%203)%20-%20VIA%20Engineering%20Guidelines.pdf [Accesed 10 Dec. 2020]

- Studienet.via.dk.(2018).*Engineering_templates.*[online] Available at: https://studienet.via.dk/projects/Engineering__project_methodology/_layouts/15/WopiFrame.aspx?sourcedoc=/projects/Engineering__project_methodology/General/Templates/Documents%20in%20English/2018%20Project%20Description%20Template%20-%20VIA%20Engineering%20Guidelines.docx&action=default [Accesed 9 Dec. 2020]

- Studienet.via.dk.(2018).*Project_Report_Template.*[online] Available at: https://studienet.via.dk/projects/Engineering__project_methodology/_layouts/15/WopiFrame.aspx?sourcedoc=/projects/Engineering__project_methodology/General/Templates/Documents%20in%20English/2018%20Project%20Report%20Template%20-%20VIA%20Engineering%20Guidelines.docx&action=default [Accesed 9 Dec. 2020]

- Via.itslearning.com.*(2018).Formal and informal language Flinders.*[online] Available at: https://via.itslearning.com/ContentArea/ContentArea.aspx?LocationID=14231&LocationType=1 [Accesed on 10 Dec. 2020]

- Via.itslearning.com.(2018).*PRESENTATION Report Writing Project Reports*.[online] Available at: https://via.itslearning.com/ContentArea/ContentArea.aspx?LocationID=14231&LocationType=1

- Openjfx.io.(2019).*JavaFX.*[online] Available at: https://openjfx.io/ [Accessed 6 Dec. 2020]

- Getbootstrap5.com(2020).*Bootstrap 5.*[online] Available at: https://getbootstrap.com/ [Accessed 10 Dec. 2020]

- Mdbootstrap.com(2020).*MaterialDesignBootstrap.*[online] Available at: https://mdbootstrap.com/ [Accessed 10 Dec. 2020]

- Balsamiq.com(2020).*Balsamiq_Mockups.*[online] Available at: https://balsamiq.com/ [Accessed 12 Dec. 2020]

11.Appendices

- Background Description
- Sequence diagram
- Class diagram
- Domain model
- Color IT - Analysis Document
- Delimitation
- Problem statement
- Project description
- Manage tasks activity diagram
- Manage requirement activity diagram
- Code
- Fxml files