Project Management System
Project Report

Lia Cicati, 304134

Rickie Nielsen, 293624

Loredana Cicati, 304135

Siyu Xia, 305554

Supervisors:

Mona Wendel Anderson

Steffen Vissing Andersen

VIA University College

VIA University College

Software Technology Engineering

1st Semester

18.12.2020

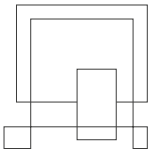
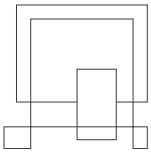
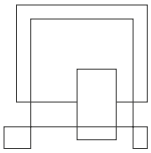


Table of content



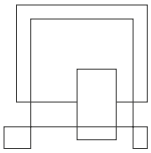


Abstract

The aim of the project is to develop and document a single user system for the small company Colour IT that currently has no project management system to handle tasks and time for their IT projects. The objectives are to fulfill the customer's requirements by making a quality final product yet easy to use and with a simple interface.

The main technical choices used for this project are implementing the system in Java making a graphical user interface with the GUI in JavaFX in order for it to be easily modified.

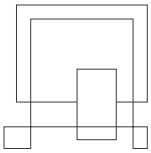
The result of this project is a stand-alone application managing projects for internal use at Colour IT and a website.



1 Introduction

Colour IT is a small company developing and implementing IT systems mostly for private customers.
The owner, Indigo Turquoise has issued a problem for Software Engineering students at VIA University College which is that the company has no project management system to handle tasks and time for their IT projects.
Their current procedure is seen as not effective because everything is registered manually something that can lead to project setbacks.
Companies have goals to be achieved, which can be used to estimate their level of success and growth. To reach them, a specific amount of tasks must be accomplished. This can be done through effective project management. “76% of high-performing organizations use strategic initiatives to meet business intent and original project goals.” (Project Management Institute, 2014)
The main problem for Colour IT is that the progress of the projects in the company currently cannot be monitored on any occasion, for example how the team members are performing on their assigned tasks or the precise time spent on completing the work. This results in being unable to guarantee the efficiency of the staff as well as the quality of completed tasks.
Due to the scope of the project some problems will not be solved. This includes no registration account to log in the system, no use of database to store the information and no log in on the website.
To fulfil the customer’s wishes the work towards this project will be fully concentrated on using both waterfall and iterative approach.
Using waterfall approach helped split the project into several phases. Starting from the Project Description and building up on Analysis where the stakeholder expectations were determined, and research was made by all team members certain Design specifications were established and evaluated to understand what the final product should look like and what actions are needed to get there. Taking the information from the previous stages provided the opportunity to create a functional product by coding in the Implementation phase and arrive at the last phase Testing where several actions were taken to ensure everything was working as planned.
For constant improvement and adjusting the iterative methodology was applied throughout the development cycle.

For Project Description see Appendix A



2 Analysis

In contrast to the interview, Project Description and upon further analyzing it was concluded that the customer wants a system to handle tasks and time for their IT projects which can result in improving the speed of their projects and service delivery. In addition, a web access to customers where they may find information about projects like description and all the requirements with a status.

2.1 Requirements

The following will present the distinct features specified in the customer interview which have to be implemented in the system. All requirements are arranged regarding their priority and significance to satisfy the customer's needs.
Functional requirements are the ones which could be tested and that reflect the most work in providing solutions delivered to the customer. Non-functional requirements are the ones which do not fulfil those conditions but had to be included to ensure the usability of the entire system.

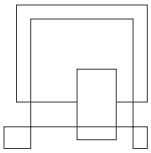
2.2 Functional Requirements

Critical Priority:

1. As a Project creator, I want to be able to create new projects in which I can assign team members, give roles for each of them so a new project can be added in the system.
2. As a Product owner, I want to add new requirements to a specific project at any time so that the team assigned to the project can see what they will be working on.
3. As a Product owner, I want to be able to add to each requirement an id, the user story text (who-what-why) template structure, estimated time for the process work, deadline, team member responsible for the requirement so that the team members can see an overview of the work needed.
4. As a team member, I want to be able to select a requirement or more in order to add specific tasks to it that contain an id, title or description, estimated time, deadline, name for the responsible team member, so that all needed information can be stored in the system.
5. As a team member, I want to be able to register the time spent on a specific task so that the precise time worked on a task can be updated in the system.

High Priority:

6. As a customer, I want access to general information about my project showing a short description and all requirements with a status to keep track of the progress.
7. As a Product owner, I want to remove or reorder a specific requirement so that the requirements can be changed between iterations.
8. As a team member, I want to be able to set the current status of a requirement to "Not Started", "Started", "Ended" so that the progress of work can be analyzed by other team members and therefore tested.



9. As a Product owner, I want to see if a requirement is in “Ended” state when all the tasks are completed so that the requirement can be tested, and the status changed to “Approved” if the feature is accepted, otherwise is set to “Rejected”.
10. As a Scrum master, I want to see on which task each team member is working on so that the progress of the work can be handled.
11. As a Scrum master, I want to see the status of a specific task so that it can be shown whenever a task is completed or not.
12. As a Scrum master, I want to see if a team member worked on multiple tasks so that the data can be registered in the system.
13. As a Project creator, I want to be able to change team member’s roles in a project and add new members to a project team so that if changes occur the information can be updated in the system.
14. As a team member, I want to be able to search for a project so the information can be seen in a form of a list of requirements with status, estimated and used time and all additional information related to individual tasks.
15. As a team member, I want to see how many hours were spent in total for all the tasks belonging to one requirement so that a team knows how much time they spent on developing, implementing, testing, and documenting a requirement.

Low Priority

16. As a Project creator, I want to be able to associate a specific project to a customer so that the data can be managed in the system.
17. As a team member, I want to be able to search by team members in order to see the employee’s productivity and the time spent related to the estimated time and with whom he worked the most.
18. As a team member, I want to store the entire project information in a single file so it can be accessed when a project is done by searching for historical data.
19. As a team member, I want to be able to see all tasks related to an Approved requirement in a separated view from ongoing tasks so that the information could be more organized in the system.

2.3 Non-Functional Requirements

20. The system uses files for persistence to store the information.
21. The graphical user interface should be implemented in Java with the GUI in JavaFX.
22. The system must be a single user software.

2.4 Use Case Diagram

In relation to the requirements a use case diagram was made to help achieve the scope of the system. It sets out what the system should be able to do and describes the interactions between the users, also known as actors with the system.
The use cases represent the different usages that a user might have. As shown in *Figure 1*, the project creator can have access to the following functionalities such as creating a new project and assigning a team as well as changing the team of an already created project.
The team member can manage requirements, tasks, register time spent and search for project information.
It is also important to highlight that the customer can have a web access to their projects showing the description and requirements.

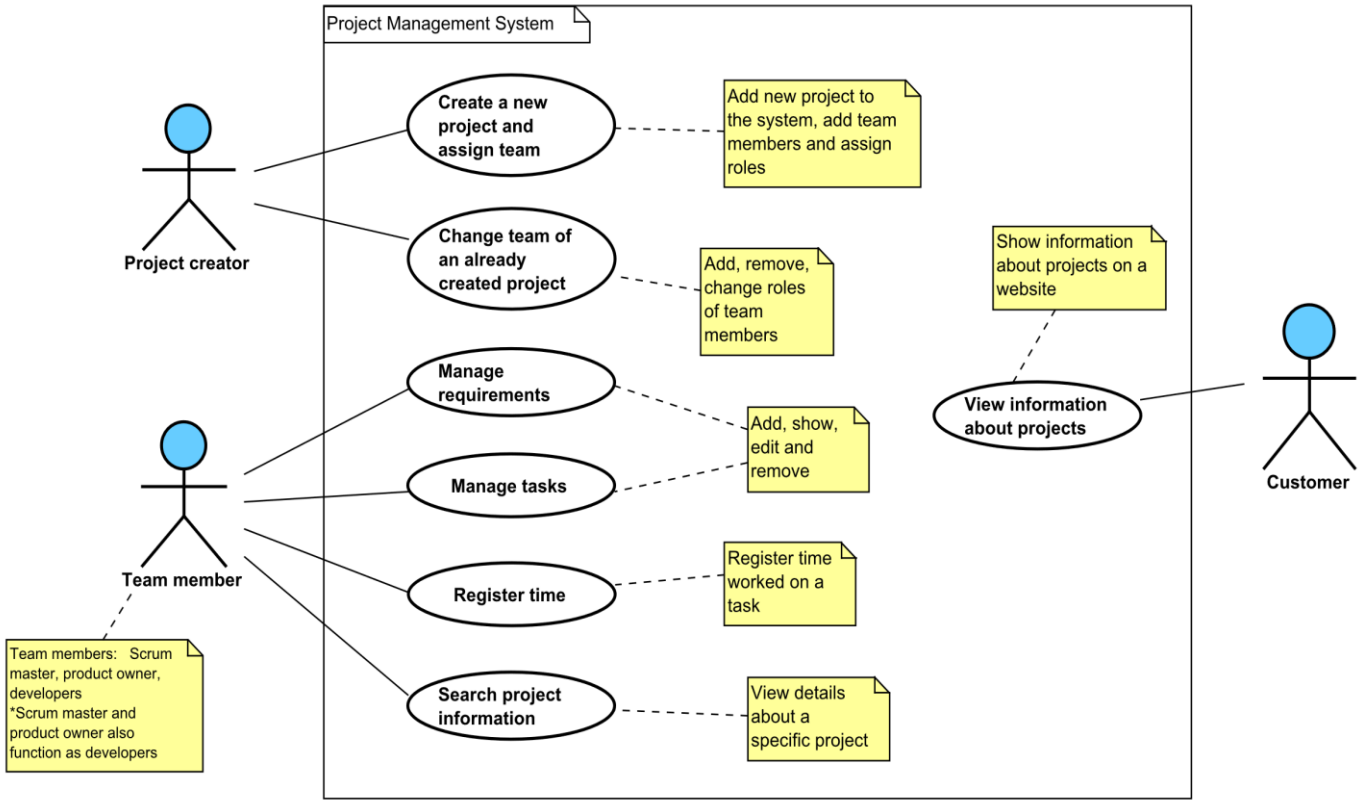
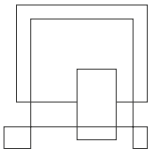


Figure 1



2.5 Use Case Description

In correspondence to each use case there is a use case description explaining the process of each function by showing the steps that must be followed when accessing the system.

In *Figure 2* is presented the use case description showing how to create a new project and assign a team. As it is the first step when accessing the system, it is considered the most important functionality. The following functions can only be implemented if this one is executed.

Use case	**Create a new project and assign team**
Summary	Add a new project to the system, add team members and assign their roles
Actor	Project creator
Precondition	There is a list where the information can be added
Postcondition	A project has been created. Team member(s) has been added. Roles of members have been set.
Base sequence	**1.** Show overview of projects created or empty list if no projects have been added If ADD go to step **2** If ASSIGN go to step **7** ADD **2.** Select to create project **3.** Enter a title, deadline, a unique ID for customer and a description for the project **4.** System validates data and prompts for illegal values or if the given title/ID is already used. In this case repeat step 3 again, typing or editing input **5.** Save the project **6.** System displays project data End of use case for ADD ASSIGN **7.** Select the created project **8.** Select to add a team member **9.** Enter a name and ID for team member **10.** System validates data and prompts for illegal values or if the given name/ID is already used. In this case repeat step 8 again, typing or editing input **11.** Select to assign a role from a list containing: product owner, scrum master, team member **12.** Save changes **13.** System displays team member data in form of a list End use case for ASSIGN
Exception sequence	-
Note	Changes can be made to the project information after creation if needed. The process can be canceled at any time. This Use case covers requirements 1, 16

Figure 2

For other use case descriptions see Appendix B

2.6 Activity Diagram

Subsequently to describe the functionality in a Use Case and have a better understanding activity diagrams were created.

Similarly, to use case descriptions, the activity diagrams are another important part that allow to represent the flow from one activity to another activity. Each of them is used for visualizing the dynamic aspects of a system.

In *Figure 3*, can be seen the Activity Diagram for Manage Requirements (ADD). The reason it was chosen is because the steps were later followed diligently to develop the system.

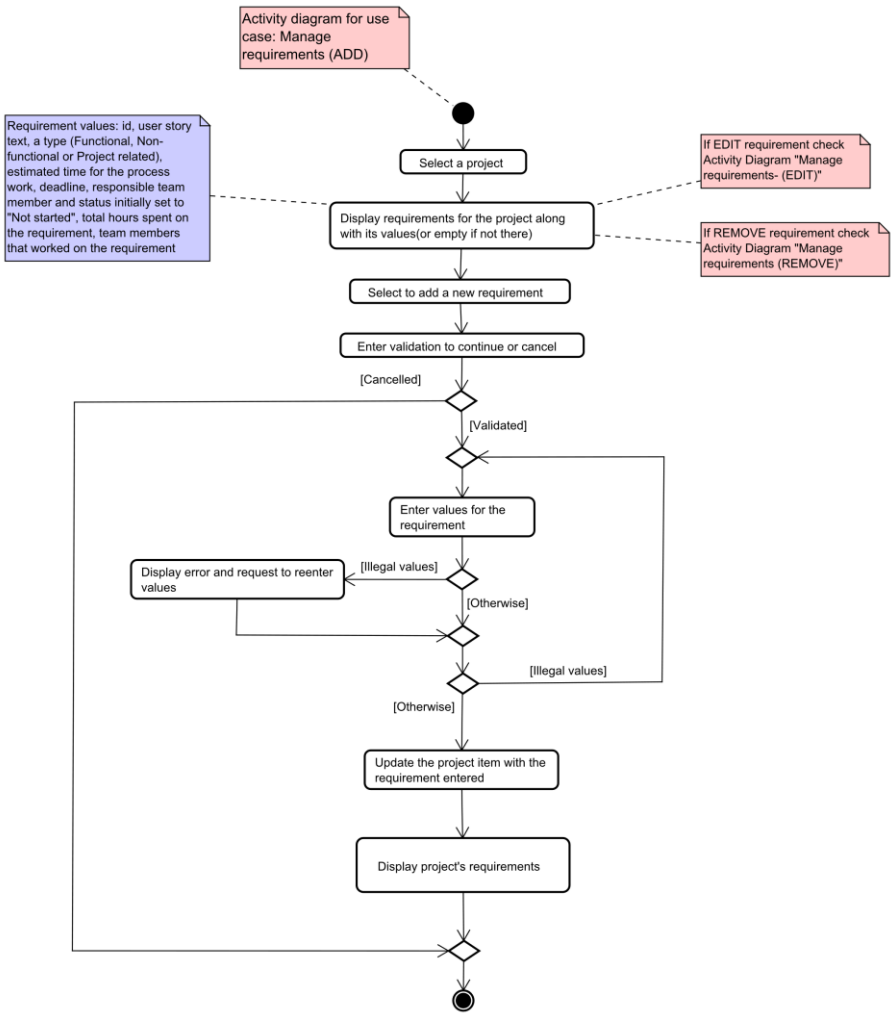


Figure 3

For other activity diagrams see Appendix C

2.7 Domain Model

As a result of the requirements that include relevant information from the interview and the use cases, for the last step of the analysis phase a domain model was created.

The main purpose of this domain model is to outline in a selective and structured representation the domain knowledge relevant for the given system. Also, it defines a general overview of the conceptual classes and the relationships between them.

In *Figure 4*, it is seen that one Project has association to a specific Customer and it can have from zero to many requirements. Besides that, an assigned team member list having three specific roles: Team Member, Scrum Master and Product Owner.

More specifically, a requirement is split up into tasks. It has a specific status assigned beforehand, a responsible team member and apart from that team members who worked on the requirement.

The task is assigned a responsible team member, a given status, multiple team members who worked on the selected task and the time registered.

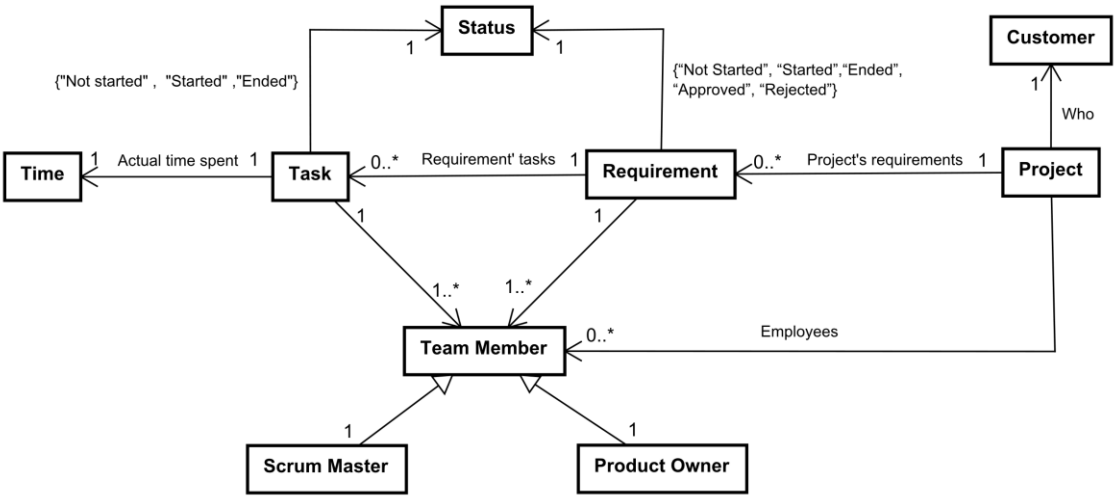


Figure 4

3 Design

Naturally following the steps from the previous phase, it was possible for everything to be transformed into a model that can be implemented. As in the Analysis the main focus was on what the system can do, in the Design section the purpose is to indicate how the system is structured.
In order to show a conceptual detailed modeling that can translate the models into programming code Class Diagrams, Graphic User Interface, Sequence Diagrams were created.

3.1 Class Diagram

The basis for the class diagram comes from requirements, use cases and domain model which were defined during analysis.
Firstly, the class diagram was structured into three packages: Model, Mediator and View where general classes were separated from specific classes.

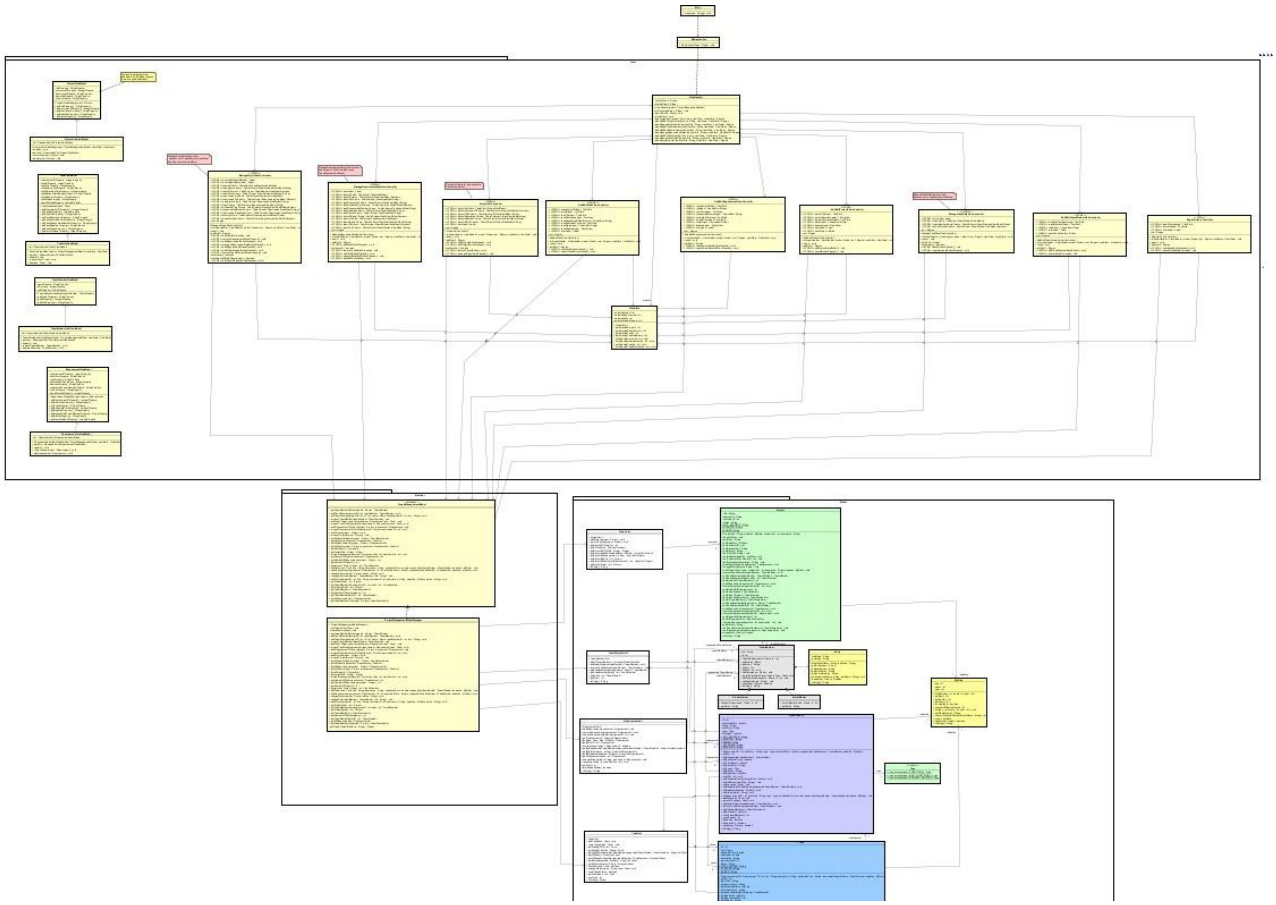
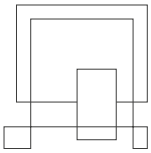


Figure 5

For the full diagram see Appendix D



Model

The Model package consists of 13 general model classes that encapsulate necessary attributes and methods making them reusable. For example, setters, getters, toString, equals, final fields and so on.
Some of the most significant classes are Project, Requirement, Task and TeamMember.
The first three mentioned classes have access to the class MyDate setting and storing a specific deadline.
The TeamMember has access to class Name and is a super class to ScrumMaster and ProductOwner which are roles that must be assigned later when implementing the system.
List classes were made for Project, Requirement, Task and TeamMember.
Lastly an enum class Type was created containing three constants for the type of requirement and a switch method where the value specified as case is being switched on and checked for each case.

Mediator

The Mediator package has a model and model manager class.
The ProjectManagementModel contains an interface with methods to be called from the view and the ProjectManagementModelManager implementing this interface and delegating to the domain model classes.

View

The Graphic User Interface (GUI) is introduced in the View package.
Prior to proceeding with implementing the files for each window, a design template, which can be seen in *Figure 6*, was made beforehand as a first draft. These blueprints were followed when creating the FXML files- “an XML-based markup language” generated by JavaFX SceneBuilder – “a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding.” (Oracle,2020)

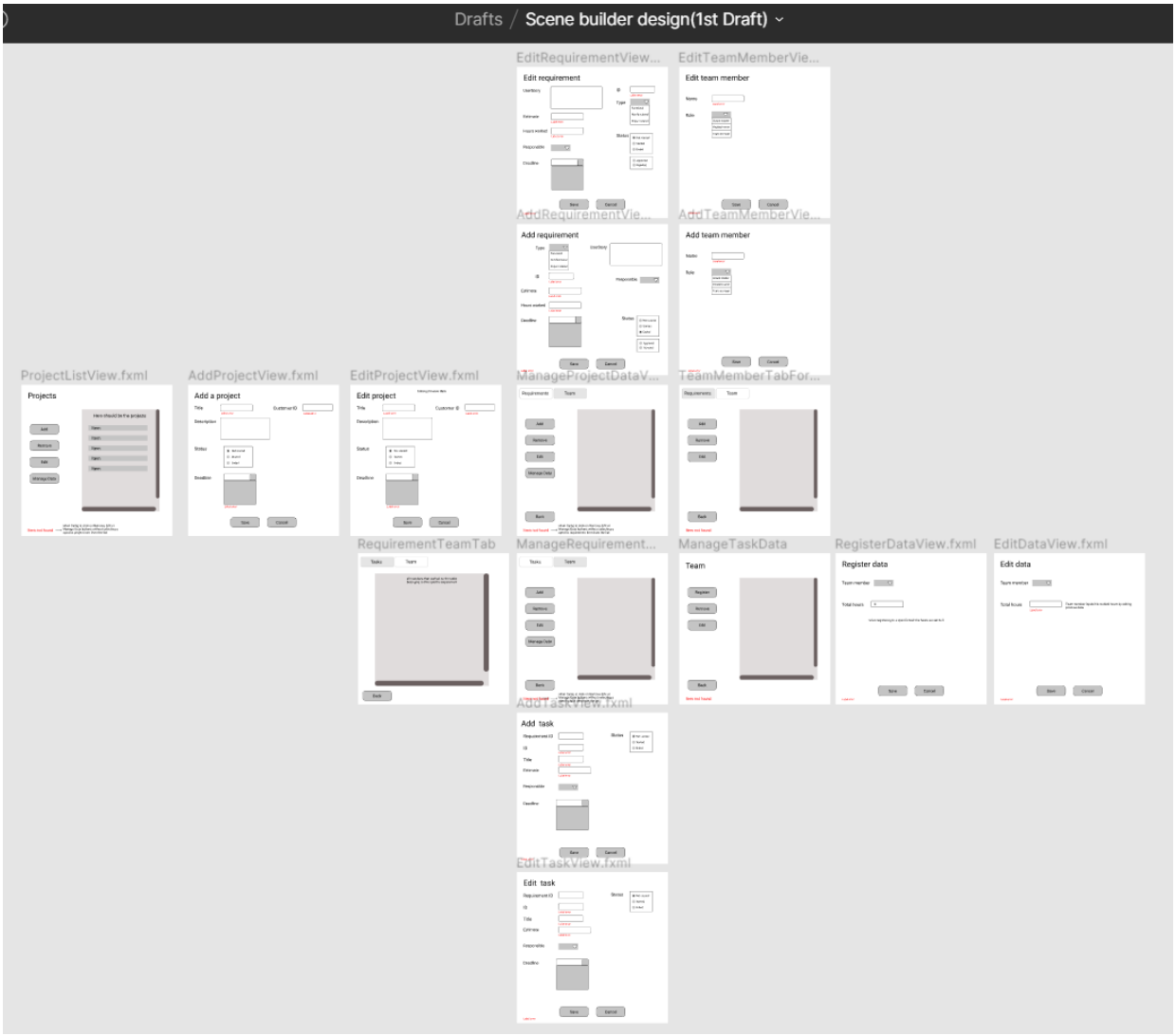


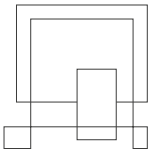
Figure 6

Later on, some FXML files were merged together such as Add and Edit to enable editing the existing information that was added in the system.

As a result, the View ended up having nine FXML files containing the component for the window.

As a first step, a list of all projects *"ProjectListView".* From this window there are options to open another window to either add, remove or edit a project *"AddEditProjectView".* For adding it is required to select Add whereas for editing a project has to be selected from the list and then all project information that was imputed is shown.

Apart from that, when selecting to Manage data, another window is displayed *"ManageProjectDataView"* with two tabs containing lists with the team and all requirements related to the project.



Both tabs in this window also have the option to either add, remove or edit *"AddEditTeamMemberView"* and *"AddEditRequirementView".*

From the tab with Requirements, selecting a requirement and ManageData button will open another window *"ManageRequirementDataView".* More specifically this window will contain two tabs for Tasks and Team.

The first tab will follow the same logic as the previous windows such as adding, removing and editing a task by opening a new window *"AddEditTaskView".* Similarly, it will contain the possibility to select an added task and open another window for managing data *"ManageTaskDataView"*. Finally, here the last window will be opened for registering hours *"RegisterHoursView".*

Each of these files has a controller class that provides functionality implemented as methods.

The ViewHandler class is in charge of creating the Scene and the Stage and switching between windows. It has methods to load different FXML files. Additionally, the ViewHandler is creating the object ViewState and has an option to send it to each of the controllers. First in the load method passing it as an argument and eventually calling the init method.

Therefore, the ViewState class enables sending states between windows.

More classes were implemented to make possible to display the stored data such as: ProjectViewModel, ProjectListViewModel, RequirementViewModel, RequirementListViewModel, TeamMemberViewModel, TeamMemberListViewModel, TaskViewModel and TaskListViewModel.

3.2 Sequence Diagram

Taking as source a Class diagram, a sequence diagram was made to show how objects are interacting and working together. The method below in *Figure 7* is complex therefore it was chosen to be presented in a sequence diagram. *(See Appendix E)*

Based on this method the status of a project will always be "Not Started" if no requirements are added to it. Later, if requirements are added by a team member to a specific project its status will automatically be set to Started.

As a next step, if changes are made and the Product Owner sees that a requirement is in Ended state when all tasks are completed, and the status is changed to Approved after testing, then the project's status will be Ended.

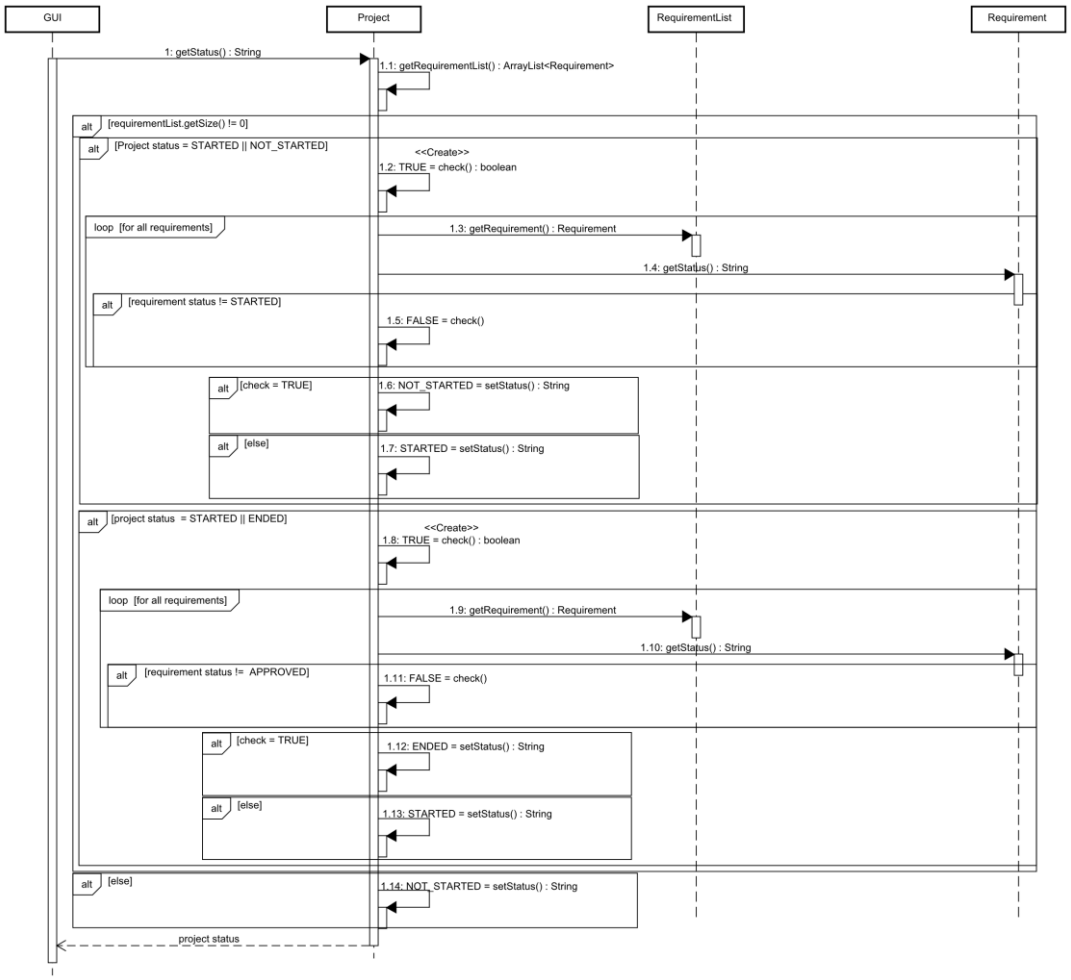
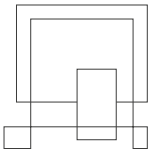


Figure 7

For more information see in the source code- package Model-Project class the code implemented based on this sequence.



3.3 User Interface

The main goal of the UI Design is to create interfaces which users find easy to use and pleasurable. Due to the fact that it mainly focuses on the needs of the platform and its user expectations (User Interface Design, 2020), the activity diagrams made before based on functionality requirements gathering were followed to create the needed output of the application.

In *Figure 8* is represented the Add a project view which can be considered one of the starting points of the program as following actions are possible once a project can be added to the system. Following the first part of the related activity diagram “Create a new project and assign team (ADD)”, the process of creating a project can be followed sequentially.

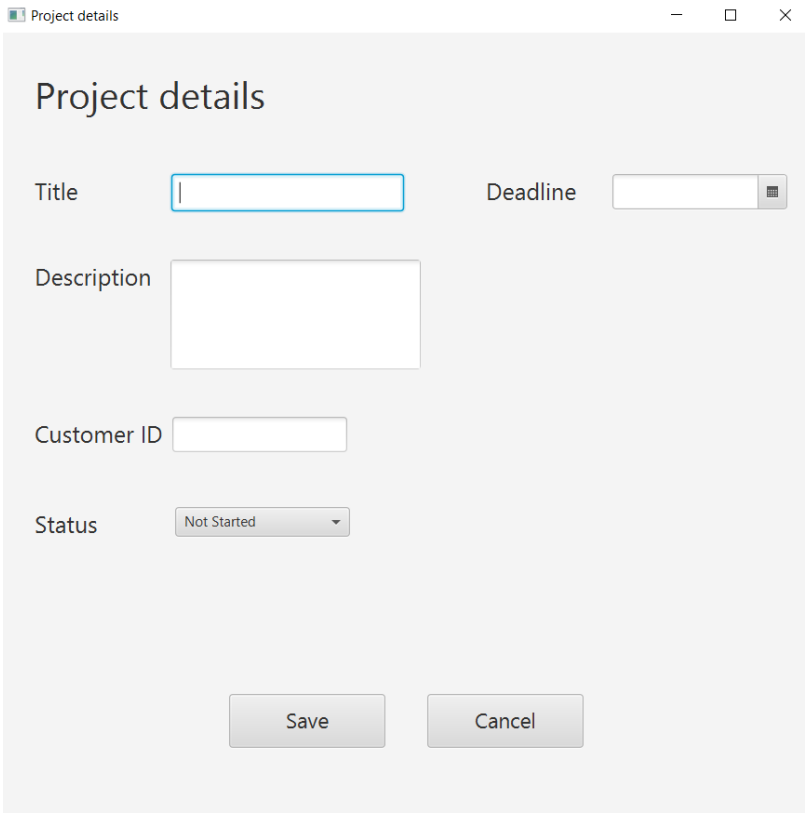


Figure 8

Considering that a project has association to the team and requirements, *in Figure 9* and *10* is represented the Manage Project Data View consisting of two tabs. In the first tab “Team” (*Figure 9)* is presented the result of following the “Create a new project and assign team (ASSIGN)” activity diagram as well as “Change team of an already created project-EDIT/REMOVE”. In the second tab “Requirements” (*Figure 10*) is presented the result of following the Manage requirements add, edit and remove activity diagram.

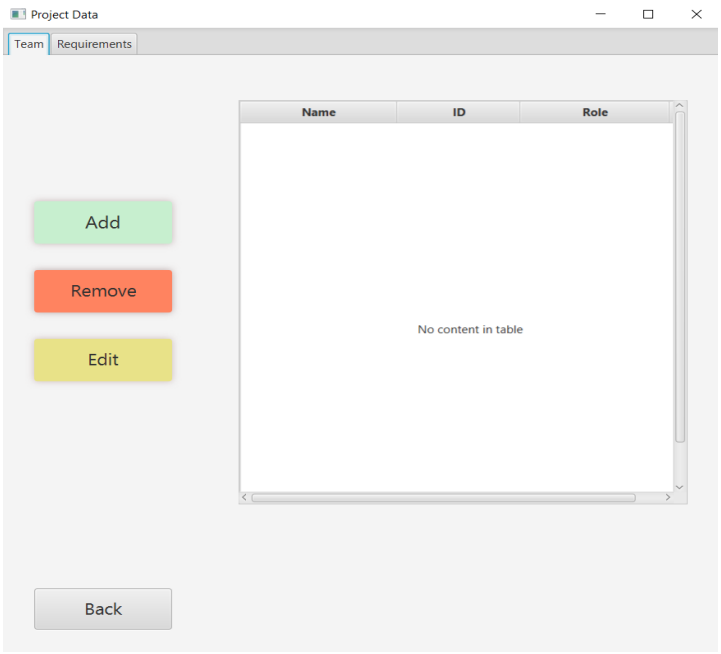


Figure 9

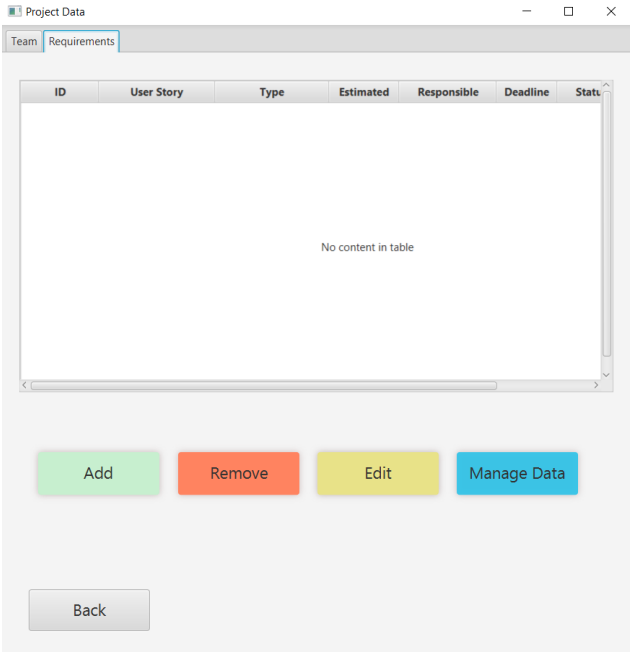


Figure 10

For User Guide see Appendix F

3.4 Website

Based on the customer interview the client wanted to have a web access to their projects. To fulfil the needed requirement a website was created.
Furthermore, the costumer drew a design sketch at a meeting of an idea for layout of the front page. However, the freedom to be creative was given.
The picture from the meeting is included below in *Figure 11*.

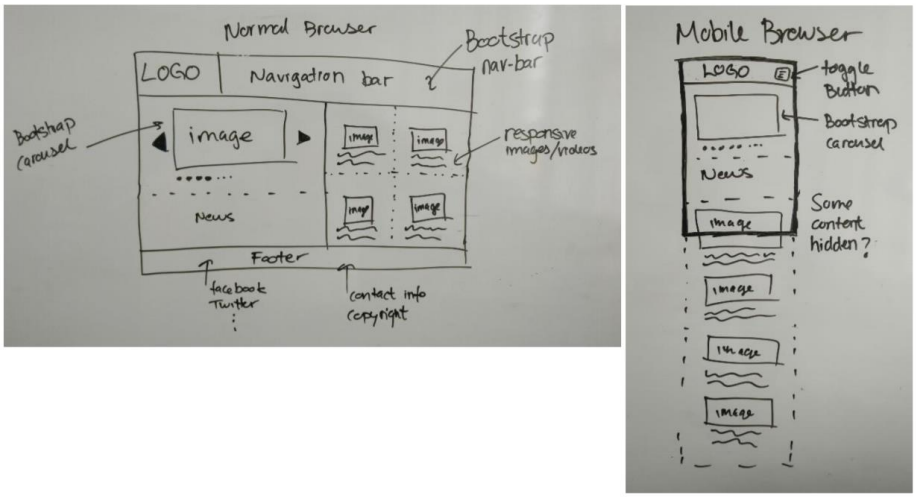


Figure 11

Accordingly, a template (*Figure 12*) was made beforehand in Figma - an interface design application, to create a better understanding of the overall concept for the website.

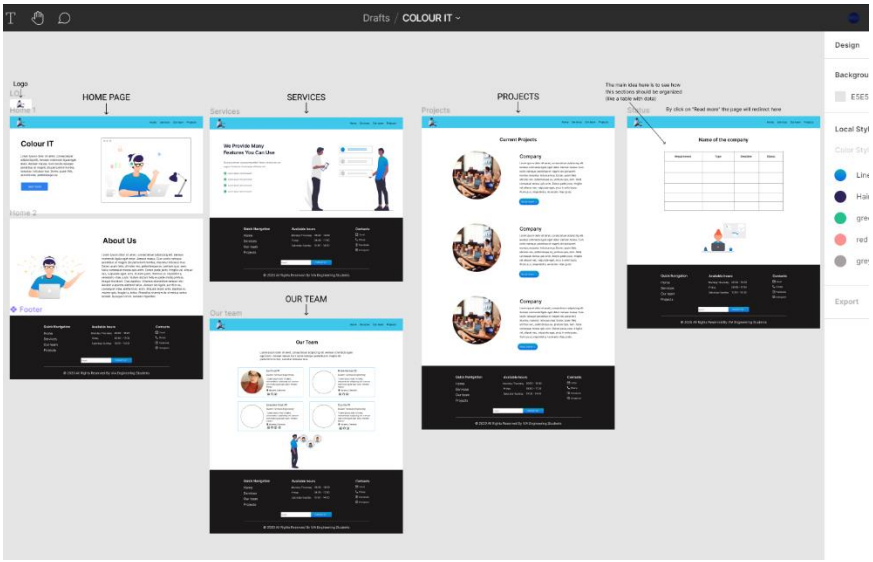


Figure 12

The website had to look nice and professional. For that reason, the colour palette consisting of blue, white and black was chosen providing a nice and clean view to the customers.
According to the customer the website needed to have an organised layout containing relevant information about the company.
To achieve that, five general pages were designed.
For instance, a Home page to provide an overview of Colour IT, Services page to show information about services that Colour IT provides and Our Team page to present information about the development team. Lastly, a Projects page to include a short description of different available projects. By clicking on read more it will redirect the user to a page called Project where a table will be shown displaying all the requirements for the project, if the requirement is functional or non-functional, the deadline, and the status of the requirement.
The following design in *Figure 13* and *Figure 14* was done to meet the customer's wishes.
The final result of the website can be accessed on this link
https://liacicati.github.io/ProjectManagementSystem/Website/

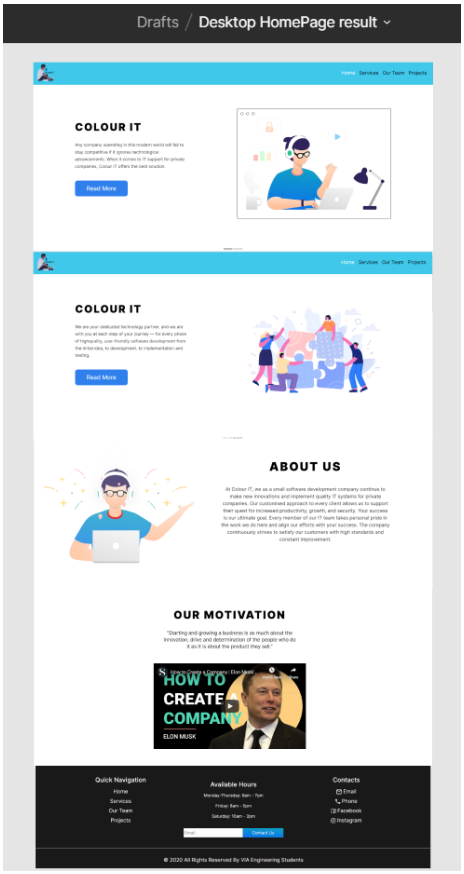


Figure 13

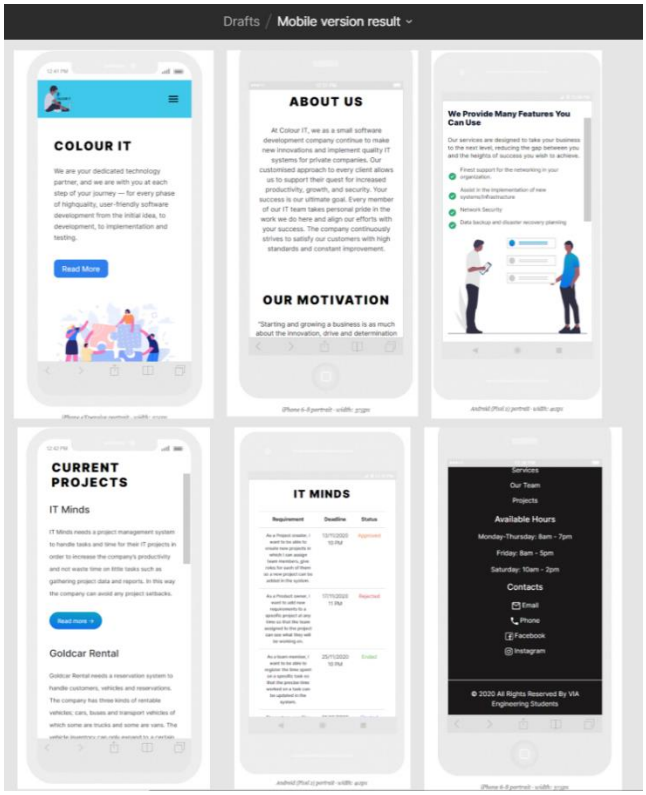
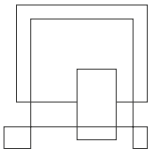


Figure 14



4 Implementation

In relation to the Design, the detailed models could be translated into code during Implementation phase. To present a better understanding of the developed system and web access several code snippets will be shown.

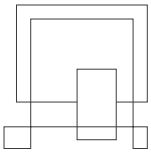
4.1 System

The system was implemented and developed based on the analysis and design.
Considering that one of the critical requirements for the system was to be able to select a requirement that has been added to a specific project so that a new task can be added to it, the following method has been chosen to be further explained.
The method shown below (*Figure 15*) is from the *ProjectManagementModelManager* class, in the Mediator package, overriding the method declared in the *ProjectManagementModel*, the interface including all methods that are used from GUI classes.
To be able to add a task to a specific requirement, a project must be created first and a requirement added to it. Therefore, the method (*Figure 15*) calls the instance variable projectList taken from the ProjectList class in the Model package, searches the project by its id, gets the array list of all requirements that have been added to the specified project, searches the requirement the task will be added to by its id and finally adds the task to the system.

```
@Override public void addTask(int requirementID, int projectID, Task task)
{
  projectList.getProjectById(projectID).getAllRequirements()
      .getByID(requirementID).addTask(task);

}
```

Figure 15



To be able to send states between windows in the ViewState class that contains the information needed to be send, 4 integers representing the id's : selectedProject, selectedRequirement, selectedTeamMember and selectedTask were initialized in the constructor to -1 indicating that there is no selection.
The code snippet below (*Figure 16*) shows the part of the method triggered when a user is trying to save valid data entered to the system. In case an existing task was selected from the list, the editTask method from the ProjectManagementModelManager will be called so that the data will be changed accordingly to the user's input, otherwise a new task will be added to the system calling the method shown above (*Figure 15*)
In both cases when editing or adding a new task the new information will be saved to the specified project and the data will be updated in the XML file related to it by calling the saveToDisk method from the Project class.

```
if (viewState.getSelectedTask() > -1) // edit
{
  model.editTask(task, taskID, viewState.getSelectedProject(),
      viewState.getSelectedRequirement(), status);

  model.getProjectByID(viewState.getSelectedProject()).saveToDisk();
}
else // add
{
  model.addTask(viewState.getSelectedRequirement(),
      viewState.getSelectedProject(), task);

  model.getProjectByID(viewState.getSelectedProject()).saveToDisk();
}
```

Figure 16

See full Source code in SingleUserSystem

4.2 Website

Following the template made during the Design phase the required website was implemented using two of the core technologies for building Web pages: Hyper Text Markup Language (HTML) and Cascading Style Sheets (CSS). Additionally, for making the website responsive a CSS Framework called Bootstrap was applied.

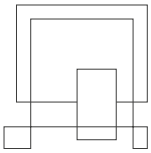
To make the interface development easy and fast the BEM (Block, Element, Modifier) Methodology was used to divide the user interface into independent blocks so that it was easier to reuse the existing code without copying and pasting. (Official BEM Documentation, 2020) Furthermore, it makes the code easy to follow and understand.

The code shown below (*Figure 17*) represents the services section of the Services HTML file.

```
<section class="services container-fluid">
    <div class="row">
        <div class="col-lg-6">
            <div class="services__information">
                <h2 class="section-title">We Provide Many Features You Can Use</h2>
                <p class="section-subtitle">Our services are designed to take your business to the next level,
                    reducing the gap between you and the heights of success you wish to achieve.</p>
                <ul class="services__list">
                    <li class="service__item">
                        <div class="service__icon"></div>
                        <p class="service__text">Finest support for the networking in your organization.</p>
                    </li>
                    <li class="service__item">
                        <div class="service__icon"></div>
                        <p class="service__text">Assist in the implementation of new systems/infrastructure</p>
                    </li>
                    <li class="service__item">
                        <div class="service__icon"></div>
                        <p class="service__text">Network Security</p>
                    </li>
                    <li class="service__item">
                        <div class="service__icon"></div>
                        <p class="service__text">Data backup and disaster recovery planning</p>
                    </li>
                </ul>
            </div>
        </div>
        <div class="col-lg-6">
            <img class="img-fluid services__illustration" src="../images/Illustration-services.svg"
                alt="Illustration">
        </div>
    </div>
</section>
```

Figure 17

For instance, “section-title” and “section-subtitle” are an example of reusable interface components as each HTML file had a heading and a subheading, therefore, to avoid repetitive styles in each CSS file, a general CSS file named “page.css” was created to store the styles that were applied to all pages (*Figure 18*) and afterwards imported to other CSS files.



Classes “services__information”, “services__list”, etc. represent elements, composite parts of the block “services” and cannot be used separately from it, therefore these elements were organized in a separate CSS file “services.css” as they were unique and described the content of the particular HTML file “services.html”.

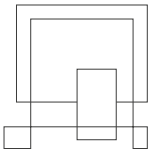
```css
/*Repetitive styles for the main title, subtitle and buttons on all pages were classified
in the following classes */
.section-title {
    margin: 0 0 30px;
    font-size: 36px;
    font-weight: 900;
    color: #000000;
    letter-spacing: 0.10em;
    text-transform: uppercase;
}


.section-subtitle {
    margin: 0;
    font-weight: 400;
    font-size: 18px;
    color: #000;
    line-height: 1.4;
    padding-bottom: 20px;
}

.button {
    display: flex;
    justify-content: center;
    align-items: center;
    transition: opacity .4s ease-in;
    outline: none;
}

.button:hover {
    opacity: .7;
}
```

Figure 18



Key word class names from Bootstrap “row”, “container-fluid”, “img-fluid” etc. were applied to make the website user-friendly meaning that the website can be accessed on different types of devices. Additionally, were used media queries to add responsiveness control by doing targeted styling with breakpoints. (*Figure 19*)

```
@media screen and (max-width: 992px) {
    .services {
        margin-top: 50px;
    }

    .services__information {
        margin: 0 auto;
    }

}

@media screen and (max-width: 600px) {
    .services__title {
        font-size: 22px;
        line-height: 25px;
    }

    .service__text {
        font-size: 14px;
    }

}
```

Figure 19

The whole code can be seen in the Website folder

5 Test

To test the system and its functionalities several tests were performed during the implementation and after it.

The strategy that was applied was White Box (Unit Test). This was done to verify the flow of input-output and see if the results are the same as expected.

During testing different possible aftermaths were predicted and handled afterwards to make the system as efficient as possible. However, because some of the methods were too complex some scenarios have not been taken into consideration due to time restrictions this meaning that more tests are required when the system is completely implemented.

Testing was executed by using test cases and JUnit. In *Figure 20* can be seen some test results. (For more detailed version see the Tests class in the Model package.

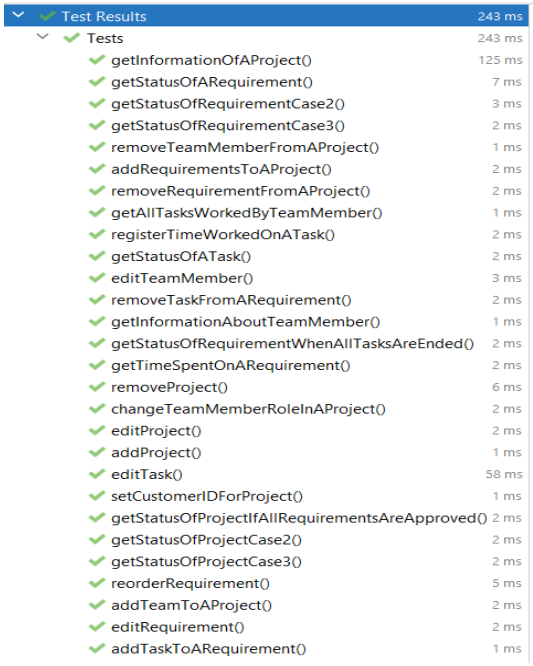
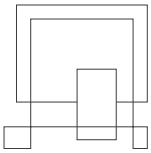


Figure 20

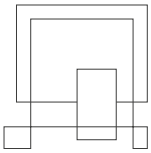


5.1 Test Use Cases

To see if a use case was implemented as expected tests were run and in order to state a test successful, the result from the test should be identical to the expected results.
The table below shows the test of the use case: Create a new project and assign team.

Create a new project and assign team			
Step	Test steps	Expected Result	Actual Result
1	Run the application	A table with project data is shown	As expected
2	Add button selected	A window with project data to be added is displayed	As expected
3	Data is entered in the text fields	The user can insert needed data	As expected
4	Insert illegal values and press Save button	System validates data and prompts for illegal values, checks if deadline is not set in the past or if the given title/ID is already used.	As expected
5-6	Insert legal values and confirm adding the data by pressing button Save	The project is added to the table data	As expected
7	A project is selected from the list	Project selected	As expected
8	Another window is displayed and Add button is selected	A window with team member data to be added is displayed	As expected
9	Name and ID is entered	The user can insert needed data	As expected
10	Insert illegal values	System validates data and prompts for illegal values or if the given name/ID is already used.	As expected
11	The choice box is selected	A list containing 3 types of roles is displayed	As expected
12-13	Confirm the data entered by pressing button Save	The team member is assigned to the project and displayed in the table	As expected

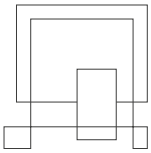
General status:	Notes
Pass	Changes can be made to the project information after creation if needed by selected a specific project from the list and pressing “Edit” button. A project can be removed from the list if needed The process of any action can be cancelled at any time.



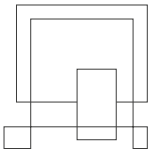
6 Results

As a result of Implementation and Testing in this section results of the system will be discussed. The outcome and achieved results of the project will be reviewed by going over the critical and high priority requirements. The lower requirements have not been implemented due to time restrictions.

Requirements	Priority	Status	Solution
1. As a Project creator, I want to be able to create new projects in which I can assign team members, give roles for each of them so a new project can be added in the system.	Critical	Done	
2. As a Product owner, I want to add new requirements to a specific project at any time so that the team assigned to the project can see what they will be working on.	Critical	Done	
3. As a Product owner, I want to be able to add to each requirement an id, the user story text (who-what-why) template structure, estimated time for the process work, deadline, team member responsible for the requirement so that the team members can see an overview of the work needed.	Critical	Done	
4. As a team member, I want to be able to select a requirement or more in order to add specific tasks to it that contain an id, title or description, estimated time, deadline, name for the responsible team member, so that all needed information can be stored in the system.	Critical	Partially Implemented Tasks can be added to a requirement but: 1) A team member can select one requirement at a time and add tasks to it 2)The responsible team member cannot be chosen from the actual team members that are added to the system	1) One way could be to set the table view to multi selection mode 2) Make the ComboBox to display the actual list of team members that have been previously assigned to the project so that a responsible member can be selected
5. As a team member, I want to be able to register the time spent on a specific task so that the precise time worked on a task can be updated in the system.	Critical	Partially Implemented	Make a better version of the method that registers hours to a specific task from Model Manager class so that it can be used in the GUI also or create a separate class for time registration



6. As a customer, I want access to general information about my project showing a short description and all requirements with a status to keep track of the progress.	High	Partially functional. The system creates lists of projects and saves them in a file	Use AJAX technique to load the XML documents, sending requests to the server so that the needed data can be displayed on the website
7. As a Product owner, I want to remove or reorder a specific requirement so that the requirements can be changed between iterations.	High	Remove - Done Reorder - Partially Done (Methods implemented but not connected to GUI)	Connect the logic with the GUI part.
8. As a team member, I want to be able to set the current status of a requirement to “Not Started”, “Started”, “Ended” so that the progress of work can be analyzed by other team members and therefore tested.	High	Done	
9. As a Product owner, I want to see if a requirement is in “Ended” state when all the tasks are completed so that the requirement can be tested, and the status changed to “Approved” if the feature is accepted, otherwise is set to “Rejected”.	High	Done	
10. As a Scrum master, I want to see on which task each team member is working on so that the progress of the work can be handled.	High	Partially done	The methods from the Model classes should be reconsidered and implemented in the GUI
11. As a Scrum master, I want to see the status of a specific task so that it can be shown whenever a task is completed or not.	High	Done	
12. As a Scrum master, I want to see if a team member worked on multiple tasks so that the data can be registered in the system.	High	Not done	Adding new methods that will search through all the tasks related to a project by a specific team member
13. As a Project creator, I want to be able to change team member’s roles in a project and add new members to a project team so that if changes occur the information can be updated in the system.	High	Done	
14. As a team member, I want to be able to search for a project so the information can be seen in a form of a list of requirements with status, estimated and	High	Not done	The search function should be implemented in the GUI



	used time and all additional information related to individual tasks.			
15.	As a team member, I want to see how many hours were spent in total for all the tasks belonging to one requirement so that a team knows how much time they spent on developing, implementing, testing, and documenting a requirement.	High	Partially done	The methods from the Model classed should be implemented in the GUI

Discussion

Requirements that are essential to make the system work were implemented and tested.

Parts that are working well and were implemented successfully consist of the main functionalities such as creating a project, assigning a team to the specific project, adding requirements and tasks with their values. Several exceptions were made to make sure the data stored is valid.
Apart from adding it was made possible to edit or remove projects, requirements, tasks and team members so the information could be handled and updated in the system.

However, due to the time constraint some requirements that are considered of critical or high priority were either partially implemented or not supported by the system.
As an example, for the partially implemented requirements methods were made but their functionality was not connected with the GUI part, thus could not be tested.

Another example is the method which sets the status of a Requirement to Ended after all tasks belonging to it are ended which was met with some minor errors. Despite the fact that the status sets automatically it does not update from the first time but requires more steps to be made in the system.

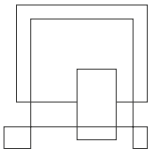
Writing to a .xml file:

This has been implemented with the method “saveToDisk” in the class “Project” using XmlJsonParser-1.4, however when it writes the .xml file it will create an empty subNode of type “teamMembers”, this creates a problem when trying to read and display the data.

In summary the system can save a project along with the corresponding data to a .xml file but will also create some data that the XmlJsonParser cannot interpret when trying to read the file.

Reading from a .xml file:

This has been implemented using two methods in the class “ProjectManagementModelManager” first is the method “initProjectsFromFiles” that looks for all .xml files in the “SingleUserSystem” folder and then adds them to a “projectList” with the method “getSingleProject” by parsing from xml using XmlJsonParser-1.4.



Problems:

When saving new files or when overwriting existing files, the xml parser will create empty subNodes of type “teamMember” that the parser is unable to handle when trying to read the files. This means that the current version of the system is unable to read from xml files unless you manually go in into the files and remove the subNodes giving errors.

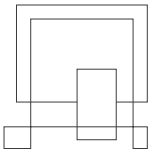
Another problem that was found using the XmlJsonParser is that it tends to give errors when there is added only one element to an ArrayList since it is given an Object when it is expecting an ArrayList. (This was fixed using DOM parser)

Potential solution:

To fix these problems, it was tried to use DOM (Document Object Model) parser since it would allow for more control over choosing how to handle the .xml files. However, another error appeared at the end using this method that had to do with the fact that when creating a new task in the system, it has to be assigned to a requirement in its constructor.

After almost two whole days of trying to come up with a feasible solution to these problems there was no time left and had to resort back to using the XmlJsonParser even though it does not function properly. Given more time to fix this method it would have been the desired solution to the problem since DOM parser allows for a lot more control then simply using the XmlJsonParser.

The (unfinished) method using DOM parser can be seen in Appendix G



7 Conclusions

Starting from the introduction the project began with a case presentation where it was discussed that Colour IT has no system to handle tasks and time for their IT projects.

The solution was further discussed in the analysis where it was decided that a Project Management System should be developed for the customer as well as a web access so relevant information about project could be found.

At this stage requirements were formulated and prioritized based on the stakeholder's specifications.

Building on that, Use Cases were made to set out the relation between the users with the system.

Later on, use case descriptions were written explaining the process of each function by showing the steps that must be followed when accessing the system.

To better describe the functionality in a Use Case and have a better understanding activity diagrams were created.

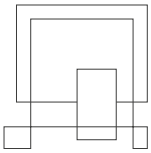
As a result of the requirements and the use cases, for the last step of the analysis phase a domain model was created.

Everything being successfully fulfilled made it possible to move to the Design phase where models where designed in order to show a conceptual detailed modeling that can translate the models into programming code. For that reason, Class Diagrams, Graphic User Interface, Sequence Diagrams were established.

Using the models from the design, implementation of the system could be started. Along with implementing the methods that represent the logic behind the system the GUI, visual part of the system was made.This provided for the system to have basic functionality.

However, despite the scope of the project being to fulfill all requirements due to time constraint it could not be done.

Due to customer's wishes a web access was created successfully to provide an overview of the company as well as its projects. A further improvement will be to export the same data to the website instead of registering it manually.



8 Sources of information

Methodology-Quick Start. [online] Available at:

https://en.bem.info/methodology/quick-start/ [Accessed 10 November 2020]

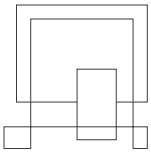
What Is Figma? (2018) *Web Design Figma* [online] Available at:

https://webdesign.tutsplus.com/articles/what-is-figma--cms-32272 [Accessed December 8, 2020]

Oracle, 2020. JavaFX Scene Builder. [online] Available at:

https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html

[Accessed 12 December 2020]



9 Appendices

- Appendix A: Project Description
- Appendix B: Use Case and Use Case Descriptions
- Appendix C: Activity Diagrams
- Appendix D: Full Class Diagram
- Appendix E: Sequence Diagram
- Appendix F: User Guide
- Appendix G: DomParser (unfinished method)
- Source code: SingleUserSystem
- Website code: Website