****

IT Project Management System

Project Report

Group 4

Group Members: Henrik Koster (305916), Kim Tranberg (172394), Aleksandrs Bistrovs (304542)

Supervisors: Astrid Hanghøj, Steffen Vissing Andersen

VIA University College, Campus Horsens

Character Count:

Software engineering

Semester 1

Date: 18/12/2020

[**Abstract**](#_ro8d7eqlkyea) **2**

[**1. Introduction**](#_ocyocysk54a) **3**

[**2. Analysis**](#_62ghrny0u5eb) **4**

[2.1. Functional requirements](#_k7pmprch7b4r) 4

[2.2. Non-functional Requirements](#_my1zd1r9vnv6) 6

[2.3. Use Case Diagram](#_a852ck9y488x) 7

[2.4. Use Case Descriptions](#_yy0fr32rguj1) 8

[2.5. Activity Diagrams](#_at4j8tohaifk) 10

[2.6. Domain Model](#_mdf8hsuqzjbh) 12

[**3. Design**](#_ll4ouhp6s3zq) **13**

[3.1. Class diagram](#_xzlhbibt7u9w) 14

[3.2. Sequence diagram](#_ak01jd4ifo9c) 16

[3.3. GUI views](#_e0fhhu7vvnb4) 17

[3.4 Website](#_7f1zjhgfj3i4) 20

[**4. Implementation**](#_qv6dlu96che) **20**

[**5. Test**](#_29jnxxy8qmtg) **22**

[5. 1. J-Unit test](#_knoq8wmwpewd) 23

[5. 2. Use Case test](#_e4v891a0feef) 24

[**6. Results and discussion**](#_vga27mo3davb) **28**

[**7. Conclusions**](#_yoncmaim4sbd) **29**

[**8. Project Future**](#_8xsxkputcp4e) **30**

[**9. Sources of information**](#_s0hx9qmanf1x) **31**

# 

# Abstract

The aim of the following report is to go through the process of creating an IT project management system by going through the main stages of the report - analysis, design and implementation. It begins by the development team gathering and analysing factual data from the customer company “Colour IT” to document the needs of the customer, as well as, the requirements for the system. The analysis is continued by the design, where careful thoughts and considerations are put forward by the team in order to achieve the desired result and meet the customers expectations. When analysis and design are completed, and the team has made considerable amounts of thought behind the desired product, the implementation phase begins.

Once the program is implemented, a variety of tests are performed to ensure that the program functions as intended, while identifying the features which are defective or missing. At this point, the product is considered to be in its ‘finished’ state and the team assembles to discuss the results and determine whether the project was a success or not.

The system was implemented in Java with the help of JavaFX to create a GUI. Implementation of the code was chosen to be done in a git repository, hosted on the free git hosting service called GitHub

In the end, the project management system achieved basic functionality as intended.

# 

# 1. Introduction

At the beginning of the first semester, software engineering students were contacted by a company named “Colour IT”. The company has provided the students with a task to design and develop a system for managing IT projects, as well as the website for their customers with the information about their projects. The customer - Mr. Colour has specifically asked for a system with a functionality to create projects, split the work between the team members and track the progress of the projects (Mr.Colour, 2020).

In order to gain a better understanding of the problem, the development team has decided to introduce the reader to what exactly is an IT management system, its importance in today's world, advantages/disadvantages and possible challenges to be encountered during the development of such a system.

IT management system is a tool to assist with the scheduling of the projects, task management and team coordination.The most important function of the mentioned system is to help the managers with day-to-day management responsibilities. It is especially important today in the age of technology to keep yourself within the range of competition, as nearly every industry relies on IT solutions (Kashyap Vartika, 2020).

The advantage of using such a system is to stay afloat with your competitors, reduce the time waste by eliminating the updating the schedules and timetables by hand, ensure that all the team members are on board with the project, as well as ensure that all the necessary information is in one place. (Mavenlink, 2020)

Disadvantages to consider are the high costs of implementation of an IT system in the workplace, adoption to the technology by the employees, as well as the time spent training the employees to use the system. It can all be very costly and time consuming, which in turn creates a challenge for the developers to make the system affordable and easy to use. The goal of the IT management system is to speed up the processes within the company instead of creating an extra burden. (Mavenlink, 2020)

Taking all of the above to consideration and, given little to-no experience and the time restrictions, the development team will be only able to carry out IT solutions for small company projects. It is important to mention that this project report will not include any budget considerations and extra occurring costs in case of not meeting the requirements of the customer.

The following report is split into seven parts:

1. Analysis, where the reader will be able to find an interpretation of the system by the authors based on the interview with Mr. Colour. Also, read through the requirements and use cases. This chapter will result in the Domain model of the system, which will be the base for the following part.
2. Design, where the reader will be able to get familiar with the Class diagram and Sequence diagram in order to get a more in-depth understanding of the methods and developed IT system itself.
3. Implementation, where the reader will be guided through the implementation of the system with the code snippets and their descriptions.
4. Testing, where the reader will be able to see if the content of the requirements have been fulfilled during the project. (do we need this part?)
5. Results and discussion, where the reader will get familiar with the outcome of the project and achieved results by the authors.
6. Conclusions, where the reader will see the compilation of results from each chapter of this project.
7. Project future, where the reader will see the authors reflections on the technical viewpoint of the project in the future.

# 

# 2. Analysis

The analysis was the most time consuming stage of the project because it establishes the framework and context of the product. As such, a great deal of care went into deciphering exactly what the consumer wanted their program to do and determining which features were fundamentally required.

## 2.1. Functional requirements

Firstly, a list of requirements was formulated. The requirements were arranged in chronological order from the most critical features to the least critical ones. These requirements were continually revised over a long-period of time with changes being made even as far as the design and implementation stage. Lastly, a list of non-functional requirements were created. As the name suggests, these requirements were not necessary for the general functionality of the product, however they would only enhance it further given enough development time. For this reason, they would only be implemented once all the functional requirements are fulfilled.

**Critical priority:**

1. As a project creator, I want to be able to create new projects in the system, so that our projects will become easier to manage.
2. As a project creator, I want to be able to assign team members to a project, such that they can work on projects.
3. As a product owner, I want to be able to add and remove requirements, to make sure the project will meet the customers specifications.
4. As a scrum master, I want to be able to add tasks to the project, such that the rest of the team members know what tasks they have to work on.
5. As a project creator, I want all information from the system stored in a single file, so that information will not be lost when the system is closed.

**High priority:**

1. As a project creator, I want to be able to assign roles to team members, so it will be clear what kind of responsibility they have.
2. As a product owner, I want to be able to prioritize requirements based on their importance(Low, Medium, High, Critical), such that features which are critical for the system to function will be developed first.
3. As a product owner, I want to change who is responsible for a requirement in case another team member is more suited for handling that requirement.
4. As a product owner I want requirements to automatically get marked with “Ended” when all tasks for a requirement are done, so that I can see what requirements I should be testing.
5. As a scrum master, I want change who is responsible for a task in case another team member is more suited for handling that task.
6. As a product owner, I want requirements to contain an id, user stories in who, what, why template, estimated time, a deadline, who is responsible, status, total hours spent, such that I can easily get an overview of all relevant information for a requirement.
7. As a product owner, I want to be able to approve or reject requirements, such that the project can reach a finished state, and make sure it meets the customers needs.
8. As a scrum master, I want each task to contain all information (Requirement ID, task ID, title, time estimation, deadline, responsible team member, status, hours spent) such that I can easily get an overview of all relevant information for a task.
9. As a team member, I want to be able to register a total amount of hours to the system whenever a task has been finished, so that I can keep track of its progress and see our productivity in regards to how well we can estimate time for tasks.
10. As a product owner, I want requirements to display total time spent, which should automatically be a total of time spent from all related tasks, so it will be possible to track the progress of a requirement.
11. As a product owner, I want the total estimated time of all tasks related to a specific requirement, be exactly the same amount as the estimated time for that requirement, such that the project will stay on schedule.
12. As a scrum master, I want to mark the status of each task in the form of: started, not started, finished; such that team members know which tasks should be worked on.

**Low priority:**

1. As a team member, I want to be able to search information regarding the projects by ID, responsible team members, deadlines, such that I can easily access specific information of a given topic, without having to go through the entire system.
2. As a project creator, I want to have the ability to change the roles of the team members, so they can work on more suitable tasks.
3. As a customer, I want the project’s description, requirements and their status, displayed on a website, such that I can track its progress.
4. As a project creator, I want to be able to remove team members from a project, in case the team size needs to be adjusted.

## 2.2. Non-functional Requirements

1. The GUI should be implemented using Java/JavaFx.
2. The website has to function using Google Chrome (version 86.0.4240.193, release date 2020-11-10) Mozilla Firefox (version Firefox 82, release date 2020-10-20) Safari (version Safari 14.0, release date 2020-9-16).

## 

## 2.3. Use Case Diagram

The use case diagram portrays the relationship between the system’s use cases and their respective actors. This makes it easier for new users to understand the hierarchy of access within the project management system. The names of the use cases represent the user-goal for a particular actor. In other words, the shared purpose of a particular group of requirements. For instance, requirements 14, 17 and 19 all concern the updating of tasks. Therefore, their use case can be simply represented as ‘Update Tasks’. The same principle goes for the rest of the use cases shown below.

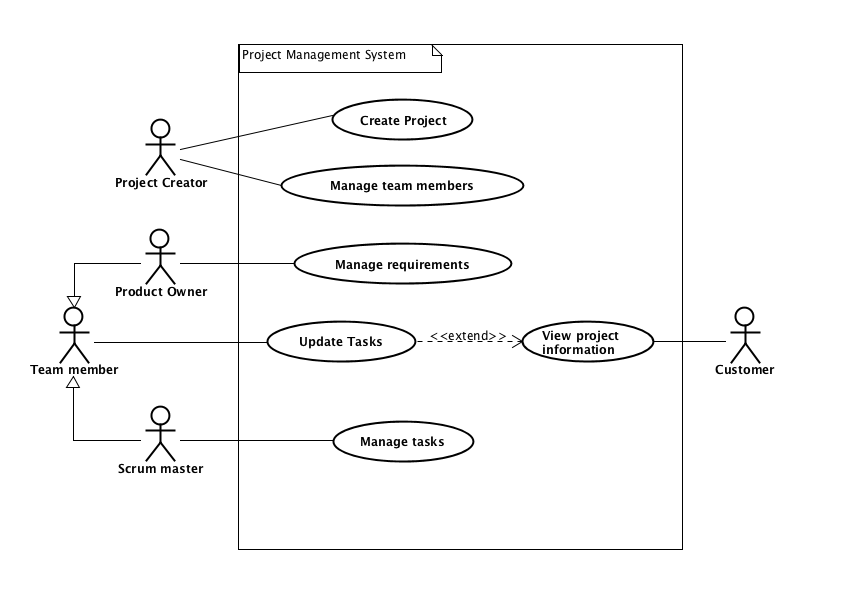


Fig. 1. Project management system - Use case diagram.

## 2.4. Use Case Descriptions

Following the formation of all requirements and use cases, a meticulous set of use-case descriptions was required. All the information one would need to know about a particular use case, such as preconditions and postconditions can be found in the use case descriptions. The base sequence is especially helpful because it establishes the step-by-step process a user of the system would go through to execute a particular function, as well as any possible exceptions and limitations. Much like the list of requirements, the use-case diagrams were a continuous work in progress. Initially, these were engineered based on how we envisioned the program to work. However, once the time came to implement it, the base sequence for each process was vastly different from the original. Although the intended users of the system would be trained beforehand, the use case descriptions help to accommodate new users and provide a form assistance for navigating the program. Many processes are only accessible by specific roles, for example only a project creator can create a project. For this reason, use-case descriptions help users of the system to distinguish which processes are relevant to them, thus preventing any confusion in regards to system access.

| **Use case** | **Create Project** |
| --- | --- |
| **Summary** | Adding a new project to the system, with all relevant information(requirements, tasks, deadline, customer, etc.) |
| **Actor** | Project Creator |
| **Precondition** | Analysis should be accepted by the customer and all requirements and tasks should be specified |
| **Postcondition** | The new project is added to the system, and its information is saved to a file. |
| **Base sequence** | 1. Create a project 2. Enter project title, customer name, project description and set a project deadline 3. Save project |
| **Exception sequence** |  |
| **Note** | This use case covers requirements 1, 5 |

Fig. 2. Use case description table - Create project

| **Use case** | **Manage requirements** |
| --- | --- |
| **Summary** | Add or remove requirements for a given project, changing their priority or responsible team member. |
| **Actor** | Product Owner |
| **Precondition** | A project should already be created in the system. |
| **Postcondition** | The requirements are now updated. |
| **Base sequence** | 1. Open Project.    1. Editing or removing a requirement go to step 6 2. Add Requirement. 3. Give Requirement an estimated time, user story, deadline, priority, responsible team member. (edit requirement) 4. Save requirement 5. Go back to project    1. Adding a new requirement, go to step 2.    2. Done with managing requirements go to step 12 6. Select the requirement to be edited. 7. If the requirement is to be removed go to step 11 8. Open requirement. 9. Edit requirement information (priority, responsible team member, approve or reject) 10. Save Requirement. 11. Go to step 5 12. Remove requirement     1. If more requirements needs to be removed go to step 6 13. Save Project. |
| **Exception sequence** |  |
| **Note** | This use case covers requirements 3, 6, 7, 8, 11, 12, 18 |

Fig. 3. Use case description table - Update tasks.

## 

## 2.5. Activity Diagrams

Once use-case descriptions were finalized, activity diagrams were produced for each system process in order to demonstrate their logical operations. Activity diagrams provide a better way of visualizing a process as a whole. Instead of portraying a specific process in a linear fashion like a base sequence, they are represented as branching functions with multiple outcomes depending on what the user wants to do and in what order they do it. They are simple to follow and provide a means to evaluate all the possible outcomes of executing a function.

For example, the base sequence for the ‘Create Project’ use case involves three key steps: 1. ‘Create a project’ 2. ‘Enter project information’ 3. ‘Save project’. This process is represented in the activity diagram directly below by three action nodes (Rectangle nodes). However between these action nodes are two decision nodes (Diamond-shaped nodes). In this particular example, they represent a decision, forming a loop. The condition being that the information entered must be correct. Therefore, if the information is wrong, the process must be repeated until the information entered is correct. This information would not be conveyed as effectively with use case descriptions alone. However, when combined with an activity diagram, the logical operation is clearly visible.

## 

Fig. 4. Activity diagram - Create project

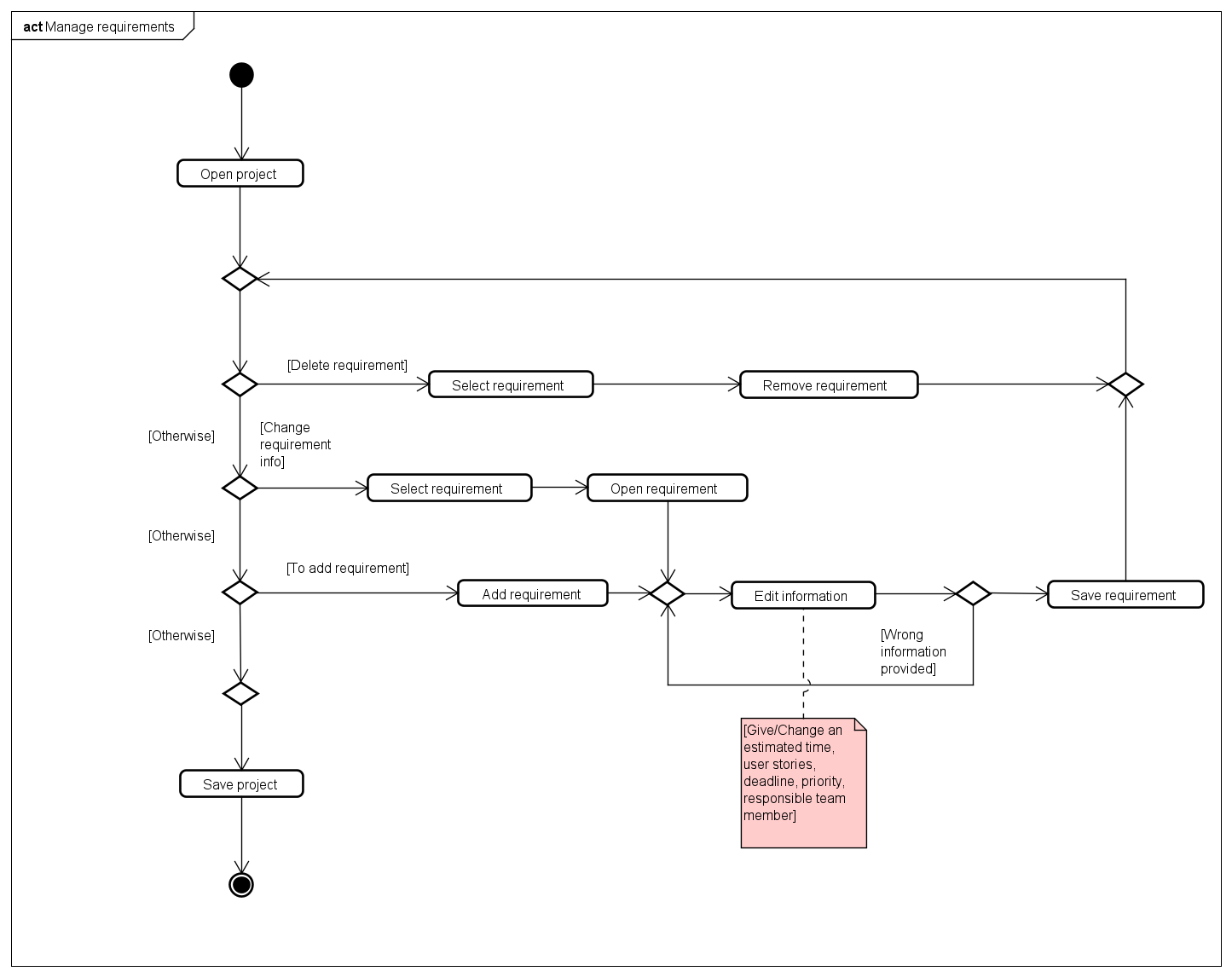


Fig. 5. Activity diagram - Manage requirements

In this activity diagram, a switch was employed to provide the user with multiple options - these options are called cases. Each new case is represented first by a decision node and they all lead back to the start of the switch. For this reason, these cases can be repeated as many times as the user wants considering they meet that case’s conditions. For example, when a user wants to remove a requirement (Case A), after they have done so, they can choose to remove another requirement (Case A), change requirement info (Case B) or add a requirement (Case C). When the user is ready to save their project, they are essentially exiting the switch.

With closer examination of the activity diagram, one can see that when a user adds a requirement, the ‘Add requirement’’ node leads to a merge node which is being intercepted by the ‘Open requirement’ node from Case B. The reason for this is that the user is editing the requirements information in both Case B and Case C. Therefore, only one ‘Edit information’ node is required as opposed to two. The merge node accomplishes this by merging the two cases into only one. Essentially, this makes the diagram simpler and helps prevent unnecessary repetition of nodes.

## 2.6. Domain Model

The final step in the analysis stage was creating a domain model. This establishes the relationship between the conceptual classes and knowledge of the problem, while showing the key concepts of the object behaviour (Philip Brown, 2014). This was challenging in its own right because classes had to be compiled into a single diagram in order to make sense as a union and come together in a cohesive manner. In the domain model shown in figure 6. *Domain model - Project management system* below, the reader is able to find the main objects that interact within the system, which helps in understanding and reaching a solution to a given problem. In the case of this report, the problem is an IT project management system to manage the projects and track the progress. In the model there are 9 conceptual classes or objects with the main one being - Project, which is standing in the middle of the model and which has a relationship to all the other objects shown in the diagram. The Project has a Date or deadline, it has a Customer who ordered the project, and a Team member who is working on the project. The project consists of Requirements, which in turn consist of Tasks. The Requirements also have Priority, that is to show which requirements are more critical for the project completion and need to be worked on first. Both tasks and requirements have Status, it tracks the progress of the project. By viewing this model, the user should have a better understanding of the description of the given problem.

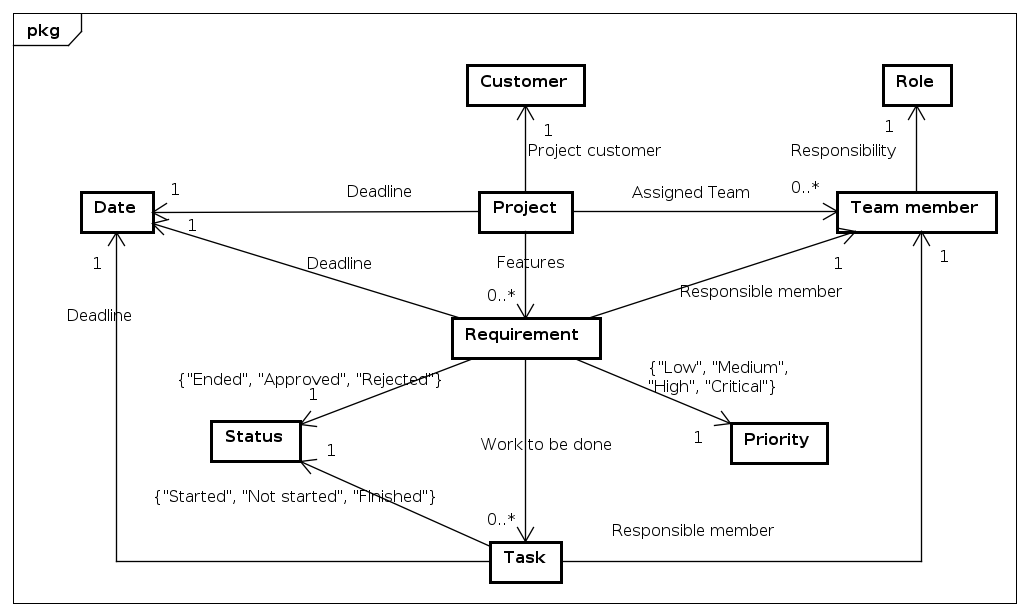


Fig. 6. Domain model - Project management system.

# 3. Design

Up until this point the development team has gathered and analyzed the data regarding the needs of the customer, that way getting an understanding of the requirements for the system and its functionality. The next step, which is a system design, was carried out in order to set the guidelines for the development team itself. Design acts as a set of instructions for the programmers, same way as the architectural design for the construction managers. Also, it is an effective way to communicate the product to both - customers and fellow employees. With the help of the design it is easier to split the work and envision what the product will look like in the future.

*“The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.”* (Odhiambo Didacus, 2018)

The design of the current system consists of a Class diagram, Sequence diagram, and some of the GUI screenshots to grasp a deeper understanding of how the system will look like and how it is going to function.

When working with the diagrams and Graphical User Interface(GUI), the sketches of the system were brainstormed and used. In order to make the design presentable for the reader, the team required help from the software development tools, to be more specific, the programme called Astah were used. All the diagrams and models presented in this report were designed in the mentioned software modelling tool. While working on the systems GUI, the team has used yet another software development tool called Scene Builder. It made the work way easier, because the mentioned tool enables to design user interfaces without actual coding.

## 

## 3.1. Class diagram

The Class diagram is the skeleton of the system - it is used to visualize and construct the system (Visual Paradigm, 2020). Here the reader will be able to see the Classes it consist of and the methods which are going to be implemented for each single Class. To make it more observable, the team has decided to split the Class diagram into several parts, this was done because of the quantity of the classes the team has agreed to implement for the full functionality of the system. The Class diagram was split into two parts, namely the model and the view part.

(*Note: in the figures below only the part of the both diagrams is visible, to access the full version of the Class diagrams see Appendix D*)

Following the waterfall approach in the beginning of the project, the development team quickly realized that it is not the best approach, when creating a functioning system.

This was concluded when the team had to go back several times and re-model the Class diagram, because of the difficulty to get all the methods described in the older version of the Class diagram to be working properly. If the strict waterfall approach would be followed, there would be many flaws in the implementation phase, thus making it hard to achieve the desired result.

In the model part of the diagram, the reader is able to find the main classes of the system. This part is responsible for all the back-end operations and general functionality of the system. The figure 7. *Project management system - Model Class diagram* shows only a fraction of the implemented classes and methods. For example, in the figure the reader can see the Project class and its relation in regards to the other classes. The Project class has a customer, team member list, id, a date, and the requirement list. It also has a constructor called Project, which holds its own variables - title and description, variable from Customer class called customer, and variable from MyDate class called deadline. In addition, the Project class has composition type of relation to the Id class and also MyDate class, meaning that Project cannot exist without Id or deadline, and the Id and deadline cannot exist on its own. In this case both of the mentioned classes should have also had a .copy() method, so that information remains unchangeable.

Given the time pressure and constant re-modeling of the Class diagram, the development team did not manage to set the right relations between the classes, to be more precise, mention where are the Association, Aggregation or Composition to be found. However, it is something that the team is aware of and what could be changed in the future development of this project.

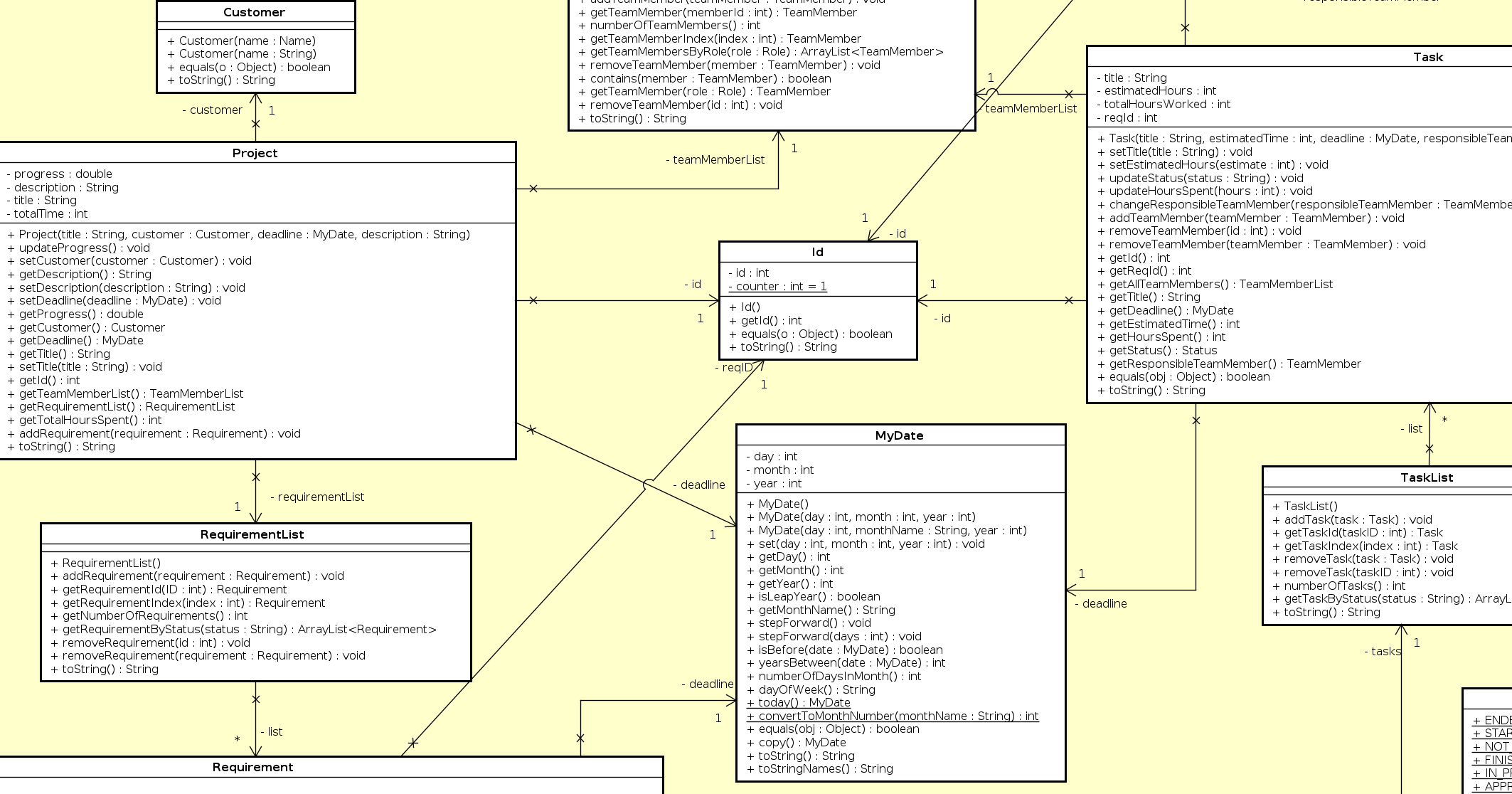
****

Fig. 7. Project management system - Model Class diagram.

In the figure 8. *Project management system - View Class diagram* below, there are four classes visible. Two of them are view controllers and the other two are view handler and view state. The ‘ViewHandler’ class is responsible for launching windows in the program. As a result, whenever a view controller needs to open a new view, it will need to access a method within the ViewHandler to do so. The ‘ViewState’ is responsible for how the system handles the states between the windows. Its main goal is to store the data and restore the results which impacted the user interface by user actions.

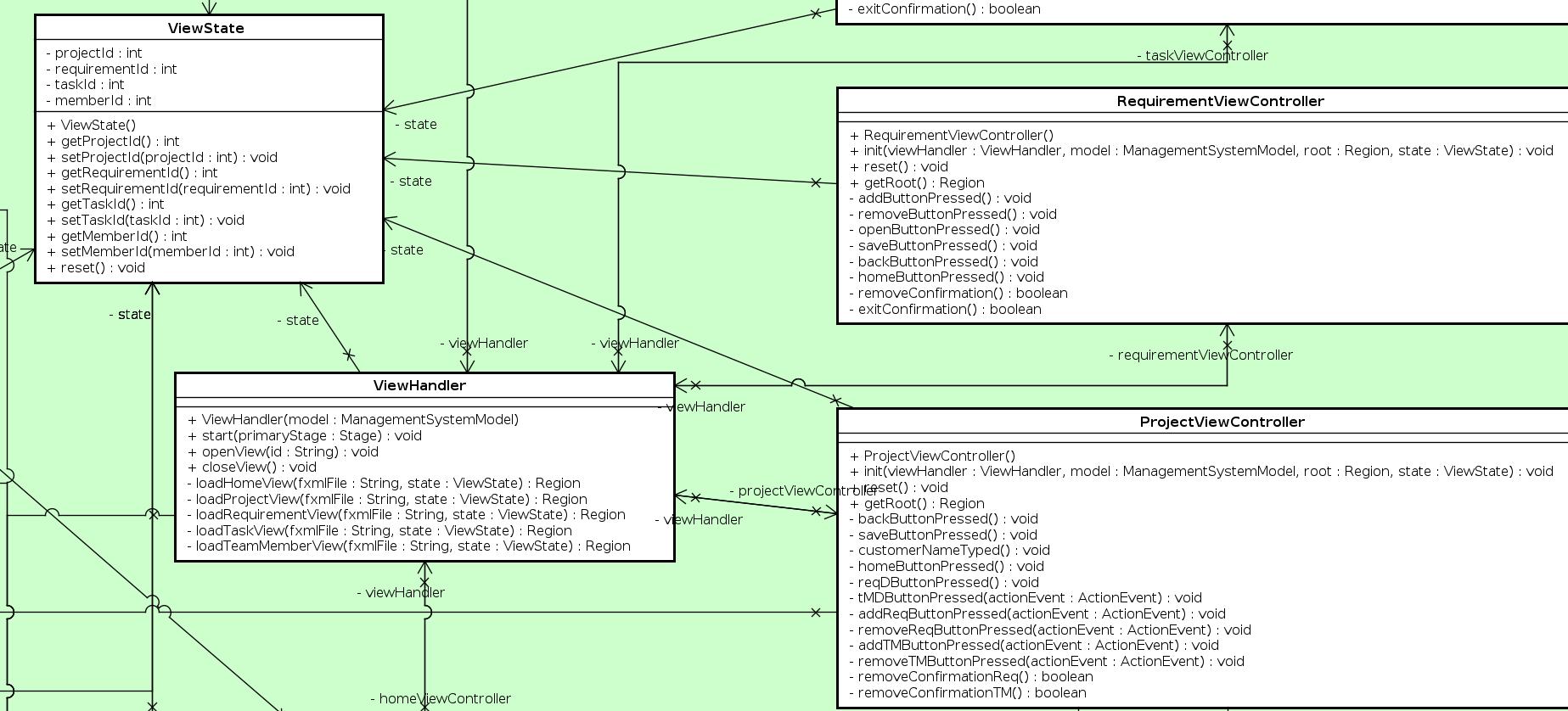


Fig. 8. Project management system - View Class diagram.

## 3.2. Sequence diagram

The sequence diagram was carried out for one of the more difficult methods by the opinion of the group. It depicts the sequence of events how the objects interact in order to achieve the desired functionality of the method to cover the system’s requirement (Lucidchart, 2020). Especially useful for the programmers to understand the logic behind the given method.

The sequence diagram in figure 9 (*Note: for the bigger sized picture see Appendix E sequence diagram, or appendix Sequence diagram updateProjectProgress.svg file*) describes the process of updating a project's progress. First the updateProjectProgress is called inside the ProjectViewController, where it is passed an id of a project to update. Then the ManagementSystem class asks the ProjectList class to return the project with the given id, where after updateProgress is called on said project.

Project loops through all related requirements and updates their status, if current status of a requirement is different from approved. A requirement has to check the status of all related tasks before it can update its own status. All related tasks are then looped through. If any of them has a different status than finished, the loop ends, which means the requirements status is in progress. In the other scenario where all tasks are finished, the requirements status will be changed to ended. Finally the project will then count the amount of requirements that has the status of ended or approved, and updates its progress as a percentage of requirements that are finished/approved.

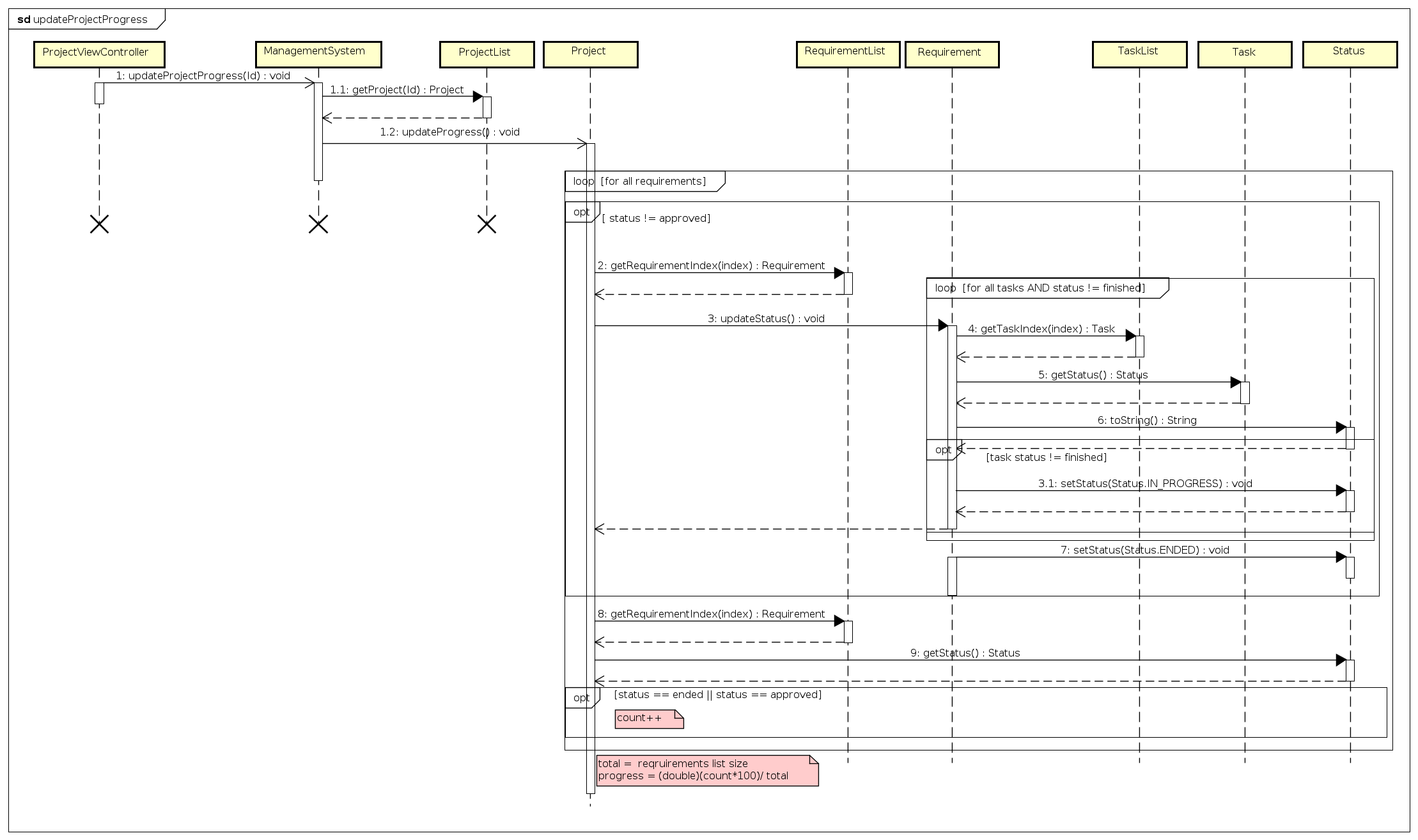


Fig. 9. *updateProjectProgress* method - Sequence diagram.

## 3.3. GUI views

The goal of the following section was to design and create a Graphical User Interface, which should include all the functionality for the system and at the same time should be easy to use. Focus is put here mostly on a visual aspect of the system. The key is that the navigation through the system should be understandable and intuitive, thus making the system user friendly. As mentioned in the introduction of this report, adoption to the technology by the employees, as well as the time spent training the employees to use the system can be very time consuming and costly. Therefore, a careful thought and considerations should be put behind the design of any system.

When planning on how the system would look, the development team went through a rather long process. Things to consider were the layout, the number of windows, if the views should include embedded menus or tabs, what happens when the displayed windows get resized, pop-up windows and more. It is important to mention that during the development process, the team had to cardinally change the look of the system. It was done due to impracticality of the previously designed GUI. It consisted of twelve windows in total, which defeated the purpose of the system being user friendly and easy to navigate through. Having realized that during the implementation phase of the project, the team had to quickly change and readjust the GUI of the system. It resulted in the five different view windows and solved the impracticality issue. Further below the reader will be able to find the comparison between the two designs. (refer to the *Appendix B - Old GUI Sketch* to see the old design*)*

The following two figures are the screenshots of the new GUI design. In the figure 10. *Project - current system view*, the reader can see how the team decided to merge several windows together and make it one whole. In comparison to the old sketch, the newly designed view contains three windows of the old design in just one window, to be more precise, it is the list of requirements, the list of the team members and the project information itself. By making such a design the team got rid of the unnecessary amount of windows, thus making the system easier to navigate through and less confusing for the customer. Also, it minimized the view controller class count and allowed for the less written code. The layout of the current window is simple to understand and gives a good overview of the information like what project is being worked on, who is the customer, the description of the project, when is the deadline, what is the progress (given in percentage), total hours spent working on it and ID to be easily distinguishable among the other projects. In the same window on the right side, the user can see everyone who is working on the project by having an access to the team member list with a possibility to add, edit or remove the team members for the current project. In addition, all the requirements for the project are listed on the left side of the window with the same possibility to add, remove or edit the requirements. To access more information, the user has a possibility to open the requirement by clicking on “Requirement details”, that in turn opens the next window shown in figure 11. *Requirement - current system view*, where he/she will be able to see and edit the list of tasks for a chosen requirement, as well as edit the requirement itself. This layout allows the user to have access to all the required information within just one click, whereas when compared to the old design, the user needed to go through multiple windows to find the same information, while forgetting what the information was related to.

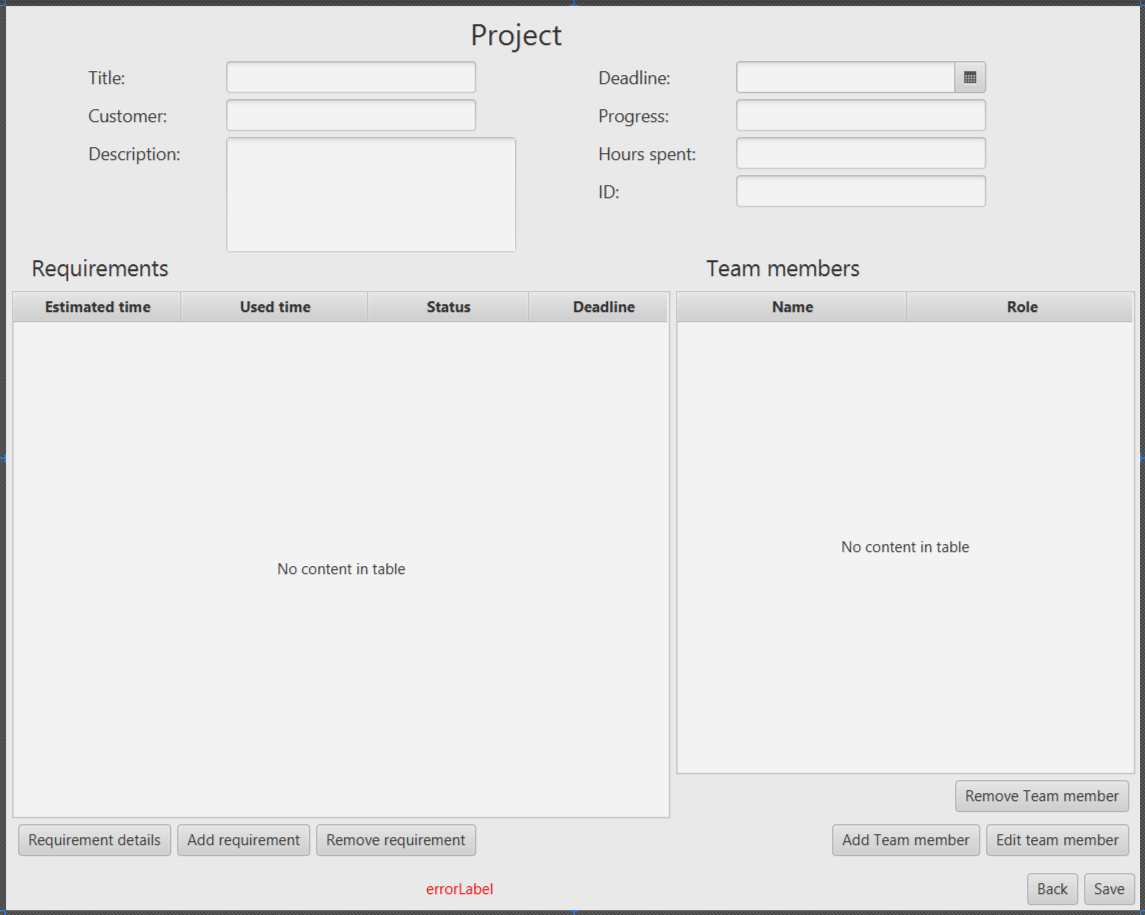


Fig. 10. Project - current system view.

The requirement window seen in figure 11.  *Requirement - current system view* consists of two main sections, where one of them is the information about the given requirement in the upper side of the view, and all the related tasks for that requirement in the bottom side of the view. This again gives a quick access to a handful of information, which is concentrated in one place in comparison to the old design. Here the user can find all the data regarding the requirement like its ID, responsible team member, description of the given requirement, deadline, its status, total hours spent working on it, priority and estimated time for the completion of that specific requirement. Most of the information is also editable, except ID and hours spent (ID is assigned automatically by the system, and hours spent are calculated based on the time spent working on tasks). As it was already mentioned above, at the bottom of the window the user is presented with the list of tasks, where the tasks can be added, removed or edited.

In addition, to make the system more user friendly, the development team decided to add pop-up windows. When there has been changes done to any of the information, and the user wishes to quit or go back without saving that information, the pop-up window comes up, which is asking for the confirmation of the chosen action. Also, when the wrong information is entered in any of the fields, the program gives an error and specifies what it is, that way ensuring the user to put the correct type of information in the specific fields.

# 

Fig. 11. Requirement - current system view.

## 

## 

## 3.4 Website

One of the requirements with a low priority was to make a website for the customers of the “Colour IT” with a possibility to be able to view the information regarding their ordered projects and track the progress. To be more precise, it was a requirement number 20 from the requirement list presented in the Analysis part of the project. Due to several reasons this task could not be accomplished. First and foremost, the development team has put their full focus on critical and high priority requirements, as without them there would be no chance to make the system at all. Second, followed by the time pressure to meet the deadline, the team needed to choose what is of most importance in the current project, which obviously was the IT management system and the reports. On top of all, the constant running into the problems while making the project and not receiving the full support from all the team members, all resulted in the website being left out of the current project.

# 4. Implementation

The following chapter shows how the above discussed system has been implemented. Here the reader will be able to find the path that the team has followed in order to make the system come to reality. To guide the reader through the implementation process, the development team will present and explain some of the most important code snippets below.

First the model was implemented, since the functionality of it is the foundation of the entire system. The model class diagram(*Appendix D.1. Model class diagram*) was used as a template, by exporting it as Java classes from Astah and importing them into an IntelliJ project. Implementation started at the “edges” by implementing simple classes, with few and simple methods. Like the Role class in figure 12

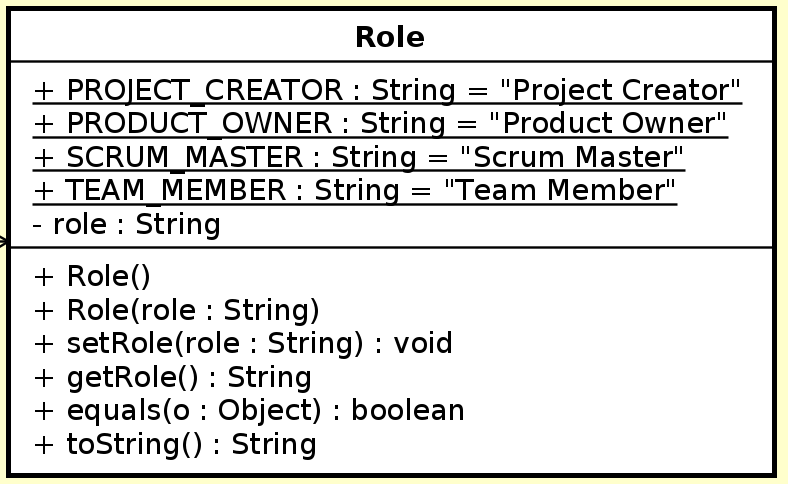


Fig. 12. Role - class diagram.

In the next phase where implementation began on some of the more complex classes, flaws in the design started to appear. A redesign of the class model diagram had to be made. When the changes had been implemented, work continued on more complex classes and methods. The last part of the model was implementing functionality of interface class, that handles communication between the view and model.

Implementation was approached by basing it on the model class diagram design, but quickly realized it was not very well designed, so a decision was made to finish implementing the view to figure out what methods actually would be needed. After the GUI was implemented, the next phase was dealing with implementation of functionality in view controllers.

Finally, a way to store the systems information would be implemented. A choice of saving all information in the form of a .json file was made, based upon one of the requirements described a website, and importing information from a .json file with javascript would be easy. With a finished system here is an explanation of how one path through the system should function, in theory. A scenario where a user wants to add a team member to a task, and how the system handles this. User clicks add team member in task window, code checks if task is saved, if not user is told to save first.

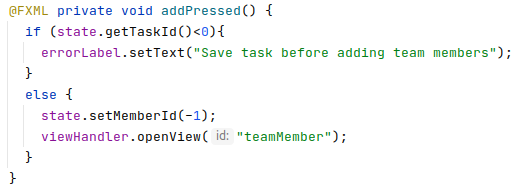


Fig. 13. Event handling when add button is pressed

When a task is saved, the viewhandler is told to open the team member window, where the initialization of the window changes depending if it was accessed from a task or project. Team member view controller, checks what team members are assigned to the related project, while at the same time excluding team members already assigned to the task, and adds them to an observable list. The list is then assigned to the drop down selection, where the user can choose who to add to the task. When the choice is made, the user clicks save. Code then adds the selected team member to the tasks team member list. Task view is now loaded by the viewhandler and the team member list is updated, with the newly added team member. User clicks on save, code then saves all information from the system to a json file.

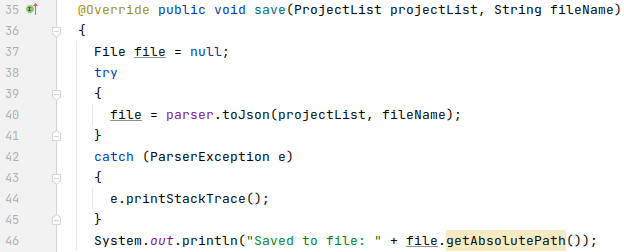


Fig. 14. Saving to a json file

This scenario and many others will be touched upon in the next section, where the system will be tested, to make sure everything is functioning as expected.

# 5. Test

After the implementation of the system, the team needed to make sure that everything is working and all the requirements are met. This was being done by making J-Unit tests for some of the class methods, as well as, testing the use cases. J-Unit tests are made to test the functionality of the method, to be more precise, if the method is doing what was intended by the programmer. Testing use cases on the other hand, are useful for broader testing of the system - it covers step by step activities from start to finish of a specific action and tests if the system functions as requested by the customer.

## 5. 1. J-Unit test

Ideally, J-Unit tests are done for all the methods, as it helps to identify the exact problem within the method if there are any and gives the opportunity to fix it. However, for this exact project the development team has only done a few J-Unit tests to present the newly acquired skill during the semester of studies. In addition, given that the coding program automatically checks for the errors itself, and the development team is always testing the system on the go, there is a slim chance of having a major error. Besides that, the mentioned errors could be fixed during the Maintenance stage of the system development, which the development team will go through in their studies in the near future. Below the reader will be able to find some of the J-Unit test screenshots followed by a short description and explanation.

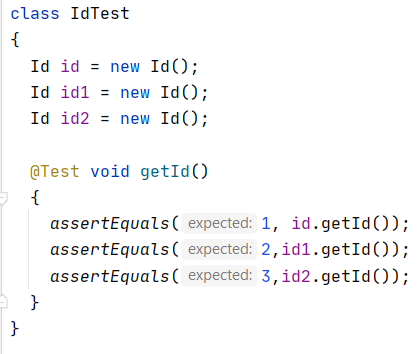


Fig. 15. J-Unit test for the method getId()

In the figure 12. *J-Unit test for the method getId()* above, the reader can see a J-Unit test done to check if the method, which was created to automatically assign the ID to the projects, requirements and/or tasks is functioning as intended by the development team. The intention was to automatically assign an integer, based on the order in which the ID was created (If it was the first ID, then the ID of it would be number 1, If it was the 206. ID, then the ID of it would be number 206). In the test there were 3 IDs created, therefore expected results would be 1, 2, and 3. In case of not correct implementation of the code, the numbers would not match, which in turn would specify the incorrect test case. This enables the programmers to go back and check the mistake and find different solutions for the previously implemented code.

In the figure 13. *J-Unit test for the method stepForward()* below, the reader is able to see if the date goes up one day to the future. As in the example, if it is the last day of the month, by calling the stepForward method it should set the day to 1, set the number of the next month, as well as, set the number of the next year (if the date was 31/12/2020, then the new date should result in 1/1/2021). In this case the development team has written the code correctly, however if the opposite would be true, the date could result in 32/13/2021, which is impossible given that there are 31 days in the month of December and there are 12 month in a year. If any of the expected numbers in the example would change, the test would fail, while also printing out the message “Not correct ...” depending in which case the mistake was found.

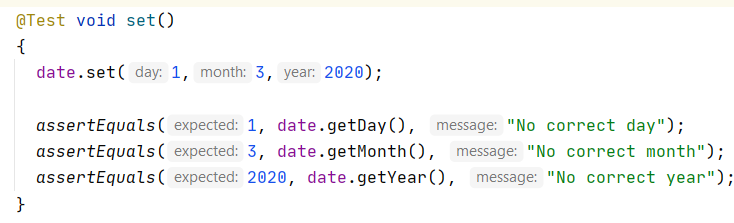


Fig. 16. J-Unit test for the method stepForward()

## 5. 2. Use Case test

Use case tests were performed to evaluate whether our functionality works as the use cases intended. These are assessed with a simple PASS or FAIL depending on whether they were accomplished with success. The program is started and a particular use case is attempted, afterwhich observations of the functionality and what order is required are made. These observations are then compared side by side with the expected results to determine whether they do in fact match the desired result. If they do, they are marked as ‘PASS’ and if they are not, they are marked as ‘FAIL’.

After a use case test has been performed for the main scenario of the use case, another one is performed for the exception scenario. In other words, the outcome of an incorrect execution. The assessment is the same as with the main scenario, either a ‘PASS’ is awarded or a ‘FAIL’ depending on whether it meets the expected criteria.

| Use case under test | Use case ”Create project” | | | |
| --- | --- | --- | --- | --- |
| Scenario | Main scenario | | | |
| Precondition | System should be running | | | |
|  | | | | |
| Step | Action | Expected result/observation | Actual result/observation | Assessment(PASS/FAIL) |
| 1 | Click on add project | New project window will open | New project windows opens | PASS |
| 2 | Enter project title, customer name, project description and set a project deadline, then save project | Information will be saved, the project window should now be updated with entered information. The project should also have been given an id, progress off 0% and 0 hours spent | After entering information and saving, the window is updated with entered information and progress is 0%, hours spent 0 and id is updated | PASS |

Fig. 17. Use case test (Main scenario) - Create project

In figure 14, a use case test for the ‘Create project’ use case has been performed. For step 1, the observed results match the desired results and so, a ‘PASS’ is awarded. The same can be said for step 2, meaning this use case functions as intended.

| Use case under test | Use case ”Create project” | | | |
| --- | --- | --- | --- | --- |
| Scenario | Exception scenario | | | |
| Precondition | System should be running | | | |
|  | | | | |
| Step | Action | Expected result/observation | Actual result/observation | Assessment(PASS/FAIL) |
| 1 | Click on add project | New project window will open | New project windows opens | PASS |
| 2 | Enter customer name and nothing else | Error will be displayed, explaining what information is missing | Nothing happens | FAIL |

Fig. 18. Use case test (Exception scenario) - Create project

In figure 15, a use case test for the exception scenario of the ‘Create project’ use case has been performed. For step 1, the observed results match the desired results and so, a ‘PASS’ is awarded. However, the observed results for step 2 differ greatly from that of the expected result and so, step 2 is assessed as a ‘FAIL’.

| Use case under test | Use case ”Manage requirements” | | | |
| --- | --- | --- | --- | --- |
| Scenario | Main scenario | | | |
| Precondition | System should be running, and a project should already be created in the system. The project in question should have at least one team member assigned | | | |
|  | | | | |
| Step | Action | Expected result/observation | Actual result/observation | Assessment(PASS/FAIL) |
| 1 | Select the project and open it | Project information will be displayed | Project window opens and information is displayed | PASS |
| 2 | Click on add requirement | New requirement window will be displayed | New requirement window is opened | PASS |
| 3 | Enter estimated time, user story, deadline, priority, responsible team member and save the requirement | Requirement window will be update with entered information, and will be given a unique id | Requirement window is updated, but window title is still “New Requirement” | FAIL |
| 4 | Click on back and choose ok in confirmation box | Project window will open, the newly added requirement will be displayed in the requirement list. | Project window opens, and the requirement is listed | PASS |
| 5 | Select the requirement and click on requirement details | Requirement window will open and show information from selected requirement | Requirement window opens and displays information of selected requirement | PASS |
| 6 | Edit estimated time, user story, priority, responsible team member and save the requirement | Requirement window will be update with entered information | Requirement window is updated with edited information | PASS |
| 7 | Click on back and choose ok in confirmation box | Project window will open, the requirement will be displayed in the requirement list, with the edited information | Project window opens, and the requirement is listed with edited changes displayed | PASS |
| 8 | Select requirement and remove it | Confirmation box will appear and ask if you want to remove selected requirement | Confirmation box appears | PASS |
| 9 | Choose Ok | Selected requirement will now be removed from the projects requirement list | Requirement is removed from the list | PASS |

Fig. 19. Use case test (Main scenario) - Manage requirements

In figure 16, a use case test was performed for the ‘Manage requirements’ use case. The observed results of steps 1,2,3,4,5,6,7,8 and 9 satisfy their respective expected results, and are therefore marked as ‘PASS’. However, the observed result for step 3, although very similar to their expected result, is not sufficient to be considered successful, and is therefore assessed as a ‘FAIL’.

| Use case under test | Use case ”Manage requirements” | | | |
| --- | --- | --- | --- | --- |
| Scenario | Exception scenario | | | |
| Precondition | System should be running, and a project should already be created in the system, with at least one requirement. The project in question should have at least one team member assigned | | | |
|  | | | | |
| Step | Action | Expected result/observation | Actual result/observation | Assessment(PASS/FAIL) |
| 1 | Select the project and open it | Project information will be displayed | Project window opens and information is displayed | PASS |
| 2 | Select a requirement and click on details | Requirement window will be displayed with information from selected requirement | Requirement window is opened and displays information about selected requirement | PASS |
| 3 | Enter characters into the estimate time box | Information will not be saved and an error text will be displayed with information about what is wrong | No information is save, error text is displayed, but it does not explain in an understandable way what went wrong | FAIL |

Fig. 20. Use case test (Exception scenario) - Manage requirements

In figure 17, a use case test for the exception scenario of the ‘Manage requirements’ use case has been performed. For the first two steps, the observed results match the desired results and so, a ‘PASS’ is awarded to both of them. However, the observed results for step 3 differ greatly from that of the expected result and so, step 3 is assessed as a ‘FAIL’.

# 6. Results and discussion

The goal of this project was to develop a project management system which enables its users to create projects, manage various requirements, tasks and their team members. This was clearly achieved as the final product allows users to do all of those things with all the detail one would expect from a standard management system. It is also easy to use and lets its users work collaboratively without any complications.

As the use case tests show, the basic functionality is sound. Furthermore, the system aids its users by informing them when they have skipped a particular step or when they are missing required inputs.

However, the biggest shortcoming of the project was undoubtedly the lack of a website for displaying the project information. As such, customers would not have a way of directly checking their project’s progress themselves.

Furthermore, the option to search for projects by their unique ID was never finalized and was ultimately discarded. This would have made it easier for users with a large number of projects to search for particular projects quickly.

Nevertheless, the program functions as it was expected and has a standard of quality consistent with what we envisioned.

# 7. Conclusions

All aspects and stages of the project were challenging in their own rights. However, the success of the project is undoubtedly attributed to the meticulous planning that went into it. This can be seen most prominently in the Analysis and Design stages. These were the stages that presented the most problems because they required a considerable amount of foresight. Subsequently, the slightest mistake could cause problems down the line. As a result, a great deal of care went into these stages. They were revisited continuously, with more and more corrections and additions being made the further development progressed.

Structurally speaking, the design was solid from the beginning, with only minor changes being made towards the end. This careful planning facilitated implementation considerably because there was no need to go back and redo any aspects of the fundamental design. Ultimately, this led to there being more time for ensuring the program worked as intended.

Compromises were made in the final stages of the project in regards to the final look of the program, with some features being stripped down for better functional quality. However, none of these changes affected the final product in any negative way. Therefore, they should be seen as improvements rather than downgrades.

When assessing the project as a whole, it is evident that the goals which were set out were achieved with consistency.

# 8. Project Future

There are a variety of which would have improved the systems functionality and final look. However, due to several factors, primarily a lack of time, these ideas were inevitably discarded. The majority of these are non-functional requirements, as such, they were of low priority and ultimately overlooked.

One of these regards how to system handles exceptions. Initially, methods were meant to test and throw exceptions based on if input is wrong. This was worked on for a brief amount of time before work was directed to more important functionality.

Another possible feature would have been to save the date on which a project and its requirements and tasks is created and started. Technically speaking, this is easy to achieve but like most the features in this section, they were only thought of in the very final stages of development. Therefore, it was simply never attempted.

Additionally, an option to backup project files was another possible feature. Essentially, whenever a new project was created, a copy of that project's file is saved in another location so in the case of missing or accidentally deleted projects, a project can be easily recovered.

To improve the structure of projects, it would have been helpful if requirements, tasks and team members had their own files as opposed to having all of a project's information on a single file.

Due to a lack of time, the website for displaying project information to customers was not made. Had there been more time, a website with all the appropriate information would have been made. This would have accessed information stored on the project’s file and displayed it in the form of an interactive table.

Lastly, the program would have benefited from an option to search for projects by their unique ID. This was not possible within the time provided because of the way the system handles new states.

# 

# 9. Sources of information

1. Mr. Colour, September 2020. *Colour IT–The interview.* [pdf] Available at: <<https://via.itslearning.com/ContentArea/ContentArea.aspx?LocationID=14220&LocationType=1>>[Accessed 9 December 2020]
2. Kashyap Vartika, 2020. *Project Management System: Definition & Features.* [online] <<https://www.proofhub.com/articles/project-management-system>> [Accessed 9 December 2020]
3. Mavenlink, 2020. *What is project management software?* [online] <<https://www.mavenlink.com/resources/what-is-project-management-software>> [Accessed 9 December 2020]
4. Odhiambo Didacus, 2018. *System Design in Software Development.* [online] <<https://medium.com/the-andela-way/system-design-in-software-development-f360ce6fcbb9>> [Accessed 12 December 2020]
5. Visual Paradigm, 2020. *UML Class diagram tutorial.* [online] <<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>> [Accessed 12 December 2020]
6. Lucidchart, 2020. *UML Sequence diagram tutorial.* [online] <<https://www.lucidchart.com/pages/uml-sequence-diagram#:~:text=A%20sequence%20diagram%20is%20a,to%20document%20an%20existing%20process>> [Accessed 12 December 2020]
7. Philip Brown, 2014. *What is the Domain Model in Domain Driven Design?* [online] <<https://culttt.com/2014/11/12/domain-model-domain-driven-design/>> [Accessed 17 December 2020]