Bring ideas to life
VIA University College

Project management system

The system that helps companies to manage their time and projects in more easy and innovative way

Baicoianu Ioan-Sorin, 304145 Lukas Juskevicius, 305575

Titas Savelskis, 304475 Rytis Ziaukas, 305573

Supervisors: Mona Wendel Andersen and Steffen Vissing Andersen

VIA University College

VIA University
College

[27,254 characters]

Software technology engineering

1st semester

18/12/2020

Table of content

Abstract

The purpose was to make a functional application that will help companies manage their workload easier. By doing that, workers will be more satisfied by their work. Also, the future projects will be more cost-efficient. The system was created using Java and its system design was implemented in JavaFX. The system does not fulfil few requirements, but with reading and writing information to a file, the application would match all necessary points for a software management system in order to be used in a company.

1 Introduction

This project is an innovation to every business that has projects to be done. To take a brief look at the problem we are facing, let's take an IT company as a sample. Every IT company has certain projects with their own deadlines and tasks. For even a bit bigger company, who has more than 3 projects at the time, managing all of the tasks would be really challenging. That is why top companies have a solution to this problem and it helps them manage their time and distribute tasks.

The aim with this project is to create a project managing system, that would help small businesses to adopt newer and better solution with their work and optimize it as much as possible.

When developing such a solution, we were trying to meet all the requests and requirements for the system. Although the requirements are not that advanced, we will still face difficulties such as implementation of a user-friendly interface and saving information to file efficiently. The decision was to make an interface that is not complex and easy to implement for beginners. Customers can expect more advanced interface for more advanced use, but with the certain time limit and resources the decision was made to delimitate complexity from our design. It is not efficient to rewrite file every time a consumer changes something, but with the small company with several ongoing projects this will not create any problems.

2 Analysis

The main problem starts with optimization of project management. The customer wants an easy and simple solution which would be understandable to use for his employees. The main objects are simplicity and reliability. These two factors are critical to any company including our customer. Simplicity and reliability are keys to uninterrupted work. Simplicity comes with an uncomplicated design. The decision was made to make very basic structure, which contains several members who are linked to certain requirements and tasks. All tasks and projects have deadlines and status. With this simple design we fulfill the customers requirements and also keep it fairly simple.

2.1 Requirements

The requirements we encountered were plain and simple. Overall, the customer wants full control over the created projects and requirements with tasks in it. Modification of these items should not be limited with certain administrative access nor by the program itself. Also, the graphical user interface should be developed in JavaFX and the project information should be stored in one file.

2.2 Functional Requirements

Critical priority:

1. As a project creator, I want to create a project in the system, so that team members could see and work on it.
2. As a team member, I want every project to have a function to add a task so that team members would be able to add tasks after the requirement is created.
3. As a team member, I want to receive information about ongoing projects, such that I can check status, acceptance and rejections of tasks.
4. As a team member, I want to see the deadline of the project, such that I could manage my work.
5. As a team member, I want to see the start date of the project, so that I could manage time spent on the project.
6. As a team member, I want to register the time that I have spent on an assigned task, so that it would be visible in the system.
7. As a company owner, I want to see how much time each team member spent on different projects, such as I can track team members' productivity.
8. As a team member, I want to see future projects, such that I can see who is responsible for what.

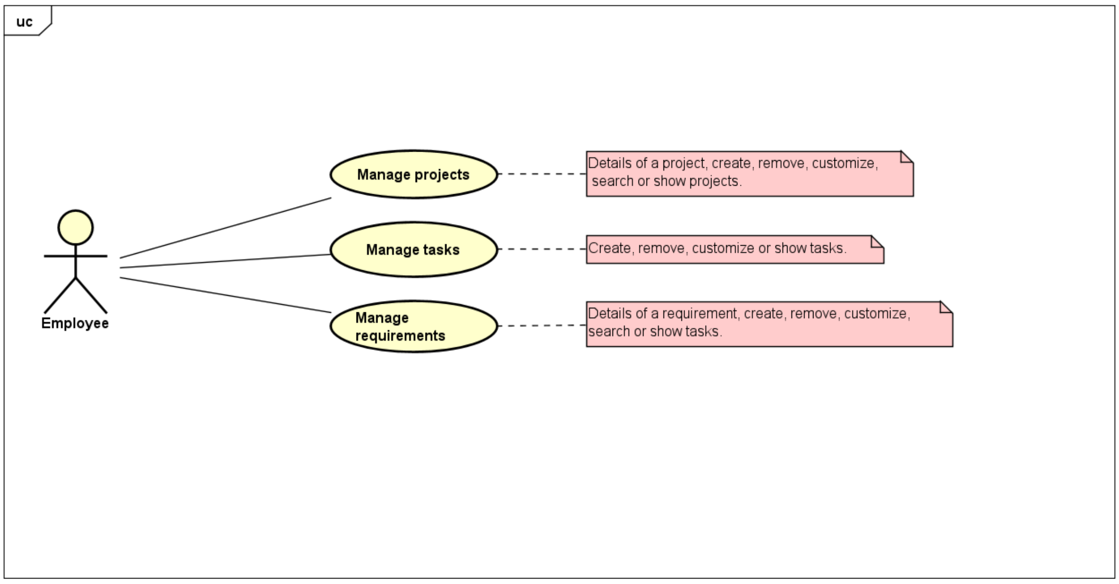
9. As a team member, i want to see the start date of a requirement, such that i could manage my tasks for every requirement I have created for project.
10. As a team member, i want to see a list of all ongoing requirements, such that I could manage my work.
11. As a team member, i want to have a add and remove button for every requirement and every new requirement should have “Not started” status, such that i will have a better workflow.
12. As a team member, i want to have a button for customize every requirement, such that, when tasks are done, i should be able to change the status of a requirement from “Started” to “Ended” or when a requirement is started, then change status to “Started” from “Not started”.
13. As a team member, i want to see the task done and undone for every requirement, such that i will know how to manage my workflow.
14. As a team member, i want to see the time spent for every requirement registered from time spent on every task, such that i will be able to see how many hours the members worked on the requirement.

High priority:

15. As a team member, I want to register a new project with a unique code, so that other team members can find the project by code.
16. As a project creator, i want to have access to assign responsible team member for requirement, such that only one person could do that.
17. As a project creator, i want to have access to assign members for every task, such that only one person could do that.
18. As a team member I want to be able to manage projects deadlines.
19. As a team member, I want to change status of a project like “Not started, Started, Ended, Approved or Rejected”, so that it would be easier to track all projects by status.
20. As a team member, I want to search for projects, such that, by entering project id, I can find the project.
21. As a team member, I want to access the entire project information, such that it can be accessed when searching for historical data.
22. As a customer, I want to see my project status and progress, such that I can track changes and work.

Low priority:

23. As a project owner, I want to have three roles: “Team member, Scrum master, Product owner” for every project and also “Project creator”, so that everybody would know their function.
24. As a company owner, I want the project team to be able to see a short description of the project, so that they would understand the goal of the project.
25. As a customer, I want to see estimated project completion time, such that I can plan implementation of the system.


uc
Employee
Manage projects
Manage tasks
Manage requirements
Details of a project, create, remove, customize, search or show projects.
Create, remove, customize or show tasks.
Details of a requirement, create, remove, customize, search or show tasks.

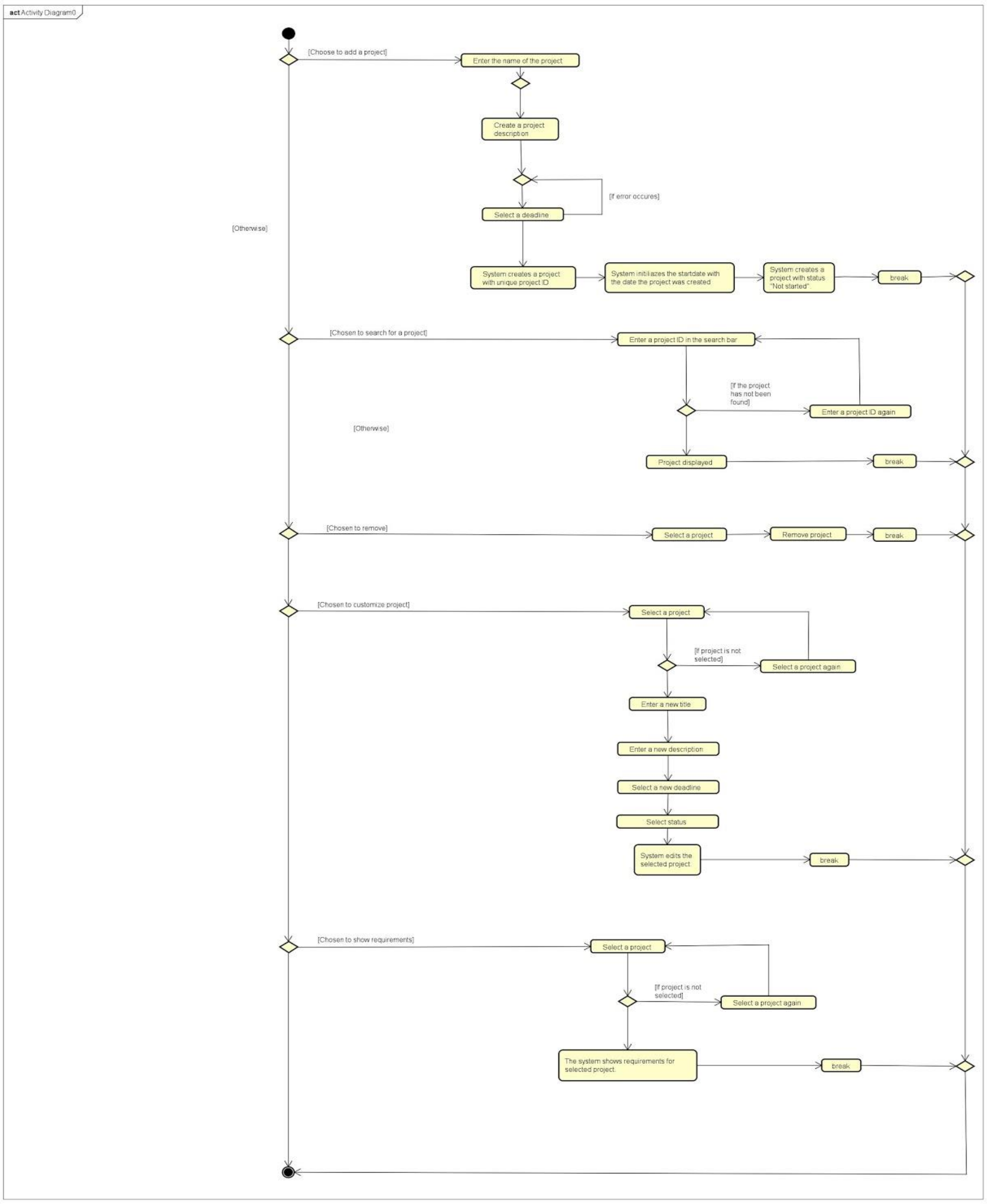
In this use case diagram are displayed a basic functionality of our system. It consists of three parts which are the full system. Manage projects let you create, remove, customize and search for projects. Inside those projects, there are requirements and in requirements there are tasks. Both of them you can create, remove and manage to your needs.

Use case	Manage projects
Summary	Details of a project, create, remove, customize, search or show projects.
Actor	Team member
Precondition	For show/create projects, GUI needs to be started. For remove/customize a project needs to be selected first. For show requirements, a project needs to be selected first.
Postcondition	A project is removed, created, or showed with deadline, members count, status, title and description details. If “Requirements”, requirements of a project are showed. If “search”, a project is showed in the table.

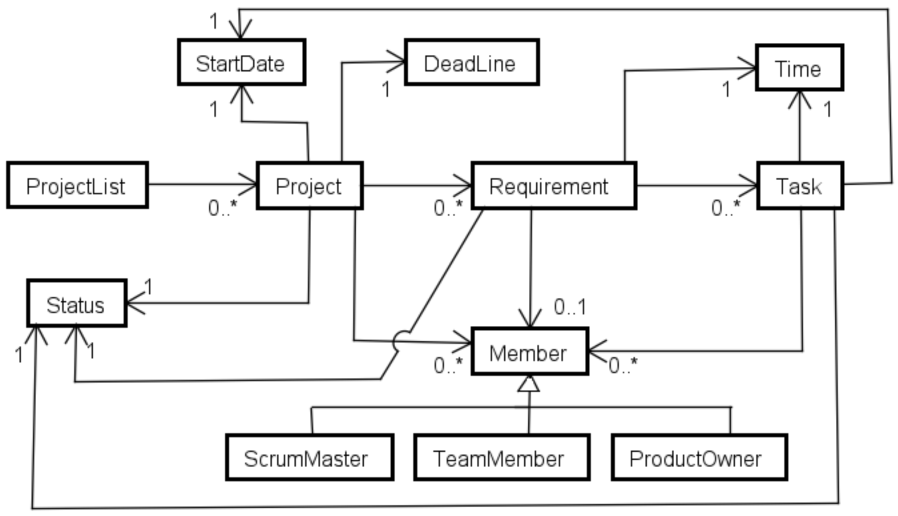
Base sequence	1. GUI started, the projects are showed. 2. If you want to add a project, then just press “Add”. 3. If chosen to “Add” then: 1. Enter a title. 2. Enter a description. 3. Select a date that represents the deadline. 4. Press “Add” button. 4. If you want to remove/customize a project, then first select it. 5. If chosen to “Remove” button: a. A confirmation window will appear. b. Press “Yes” for removing the project or “No” for cancelling. 6. If chosen to “Customize” button: . Enter a new title. a. Enter a new description. b. Select a new deadline. c. Change the status from the current one. 7. If you want to search for a project, a. Enter a project id in a search bar. b. If the project does not exist, go to step 1. c. If the project was successfully found, it will be displayed. 8. If you want to see details of a project: . Press “Requirements” button. a. A list of all requirements of project selected will be displayed.
Exception sequence	1. If you don’t select a project and you press “Remove”, “Customize”, “Requirements”, a error label will be showed with the “Select a project first.” text. 2. If you don’t feel all boxes for creating or editing a project, then an error will be showed and will not allow you to take the action chosen. 3. If project searched does not exist, then an error will be showed with text “This project does not exist”.
Note	The project creation and edit can be cancelled any time and not saved, by pressing cancel button. You cannot assign member for a project. Member count will be changed every time when you assign a member for every task or a responsible team member for every requirement. Project has requirements who has task. You will be able to add requirements later on, by selecting “Requirement” button. You cannot change project id. When you create a project the start date will be initialized with the date the project was created. When you create a project the status will be initialized with “Not started”. Only one person can manage projects at the time despite his role.

	This Use case covers requirements 1, 4, 8, 15, 18, 19, 20, 21, 22, 23, 24, 25.

This use case description refers to one of the main functions in our system – managing projects. Here it is explained in detail how can you manage your projects, what you can do and what you cannot do.
All other use case descriptions are contained in the appendices where the idea is the same, just for different use cases.


act Activity Diagram0
[Choose to add a project]
Enter the name of the project
Create a project description
[If error occures]
Select a deadline
[Otherwise]
System creates a project with unique project ID
System initilíazes the startdate with the date the project was created
System creates a project with status "Not started".
break
[Chosen to search for a project]
Enter a project ID in the search bar
[If the project has not been found]
Enter a project ID again
[Otherwise]
Project displayed
break
[Chosen to remove]
Select a project
Remove project
break
[Chosen to customize project]
Select a project
[If project is not selected]
Select a project again
Enter a new title
Enter a new description
Select a new deadline
Select status
System edits the selected project
break
[Chosen to show requirements]
Select a project
[If project is not selected]
Select a project again
The system shows requirements for selected project.
break

In this activity diagram it is displayed how class manage project works. Diagram is divided into 5 main functions: add a project – which describes how adding a project works and what criteria is presented in order to add a project, project search – it defines what a customer needs to search for a project, project removal – how does project removal work, project customization – what requirements are presented for a project to be customized and edited, displaying requirements – how displaying of requirements inside the project works.



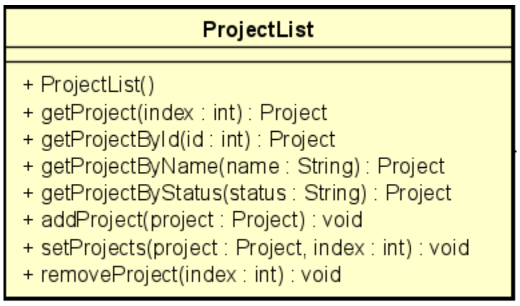
This domain model presents classes association and how they collaborate with each other.

2.3 Non-Functional Requirements

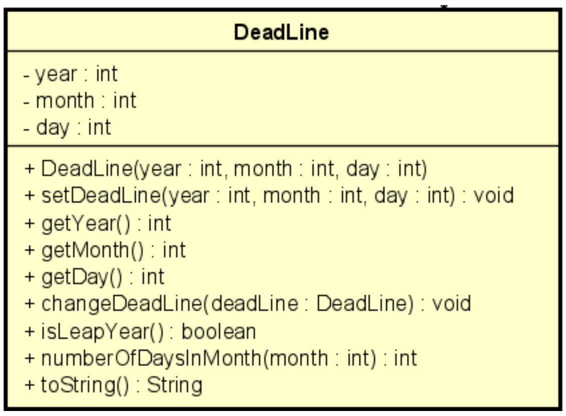
25. The system GUI should be implemented in JavaFX.
26. The project information should be in one file.

3 Design

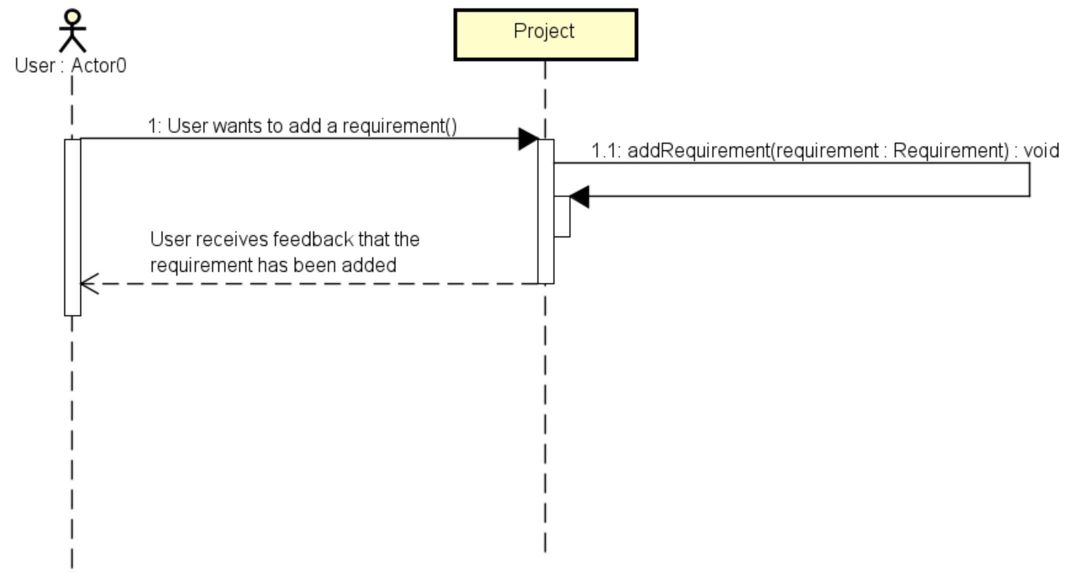
The system structure is designed to be not complicated to understand and develop. The system has the main class called “ProjectList” which stores all of the data that is necessary for the end user.


ProjectList
+ ProjectList()
+ getProject(index : int) : Project
+ getProjectById(id : int) : Project
+ getProjectByName(name : String) : Project
+ getProjectByStatus(status : String) : Project
+ addProject(project : Project) : void
+ setProjects(project : Project, index : int) : void
+ removeProject(index : int) : void

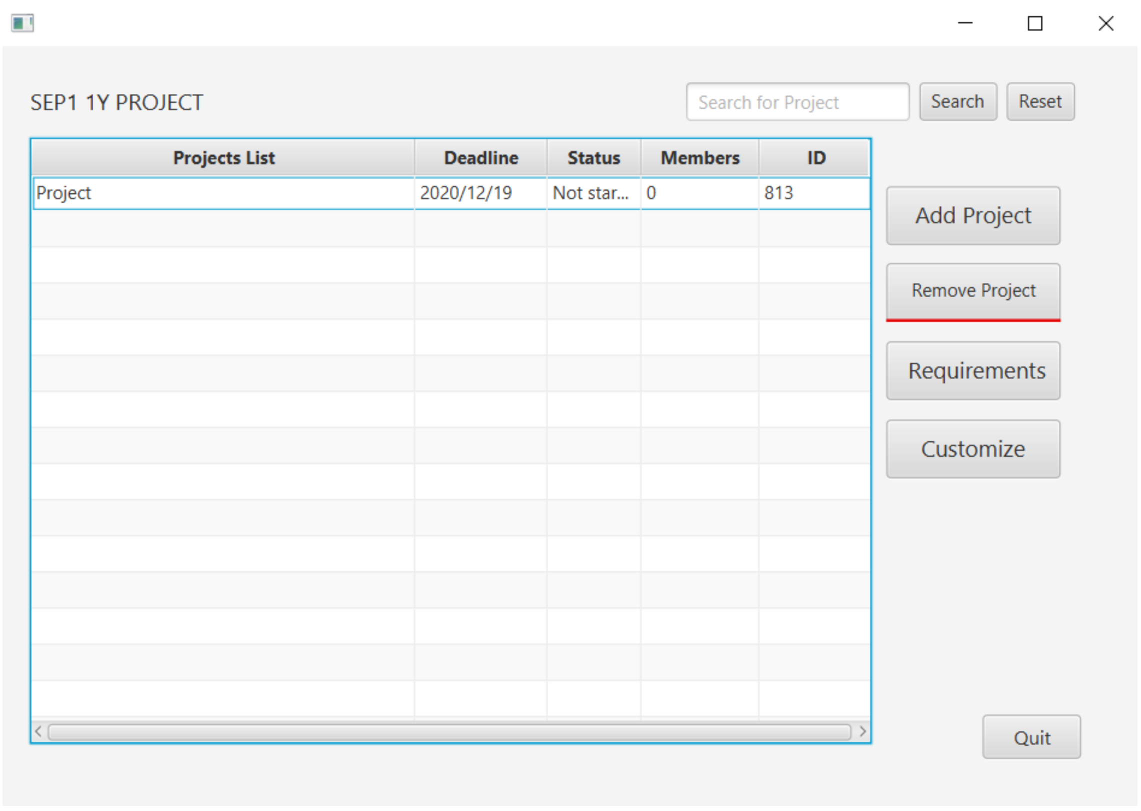
This project list class diagram represents all the classes which stores all of the other components inside it. It does not have a lot of methods, but it has all the getters and set methods necessary to sustain a working program.


DeadLine
- year : int
- month : int
- day : int
+ DeadLine(year : int, month : int, day : int)
+ setDeadLine(year : int, month : int, day : int) : void
+ getYear() : int
+ getMonth() : int
+ getDay() : int
+ changeDeadLine(deadLine : DeadLine) : void
+ isLeapYear() : boolean
+ numberOfDaysInMonth(month : int) : int
+ toString() : String

This class diagram represents class deadline. It is a very primitive class, but used in almost every window. It contains simple integers which represent a full date. It has all the getters and set methods necessary.

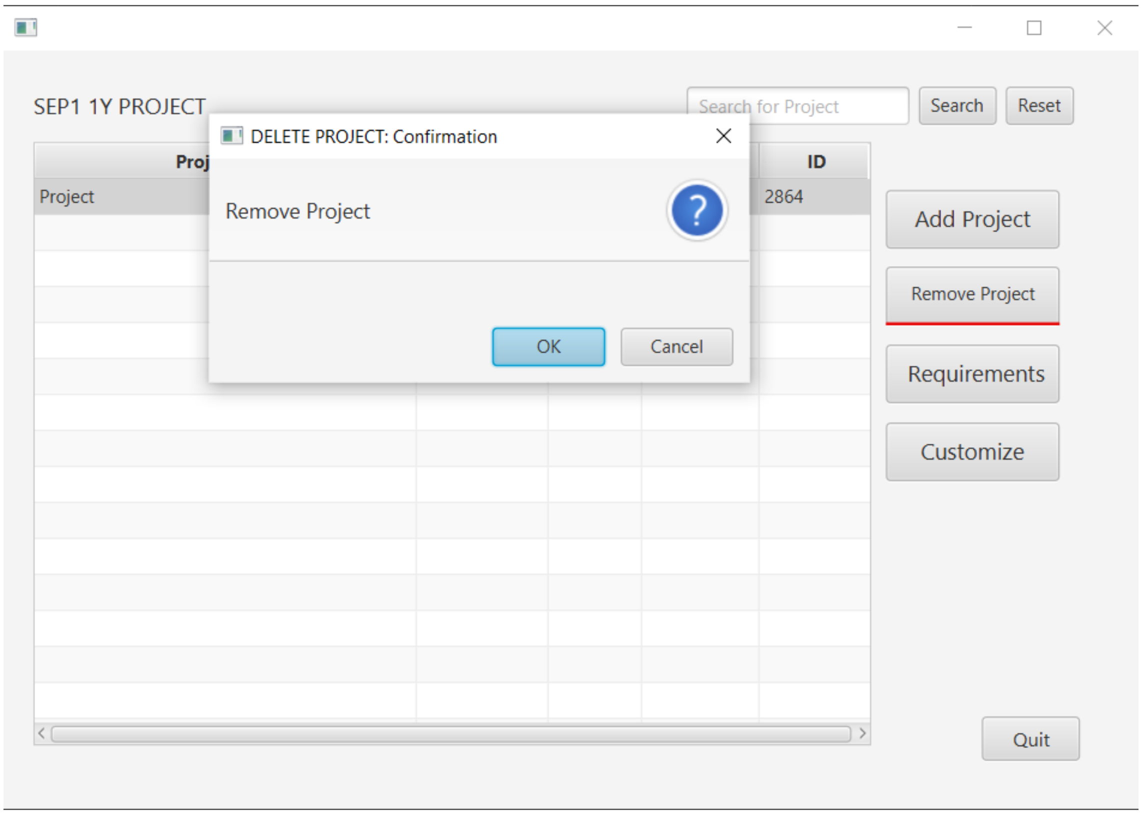

User : Actor0
Project
1: User wants to add a requirement()
1.1: addRequirement(requirement : Requirement) : void
User receives feedback that the requirement has been added

In this sequence diagram is displayed the requirement adding process.

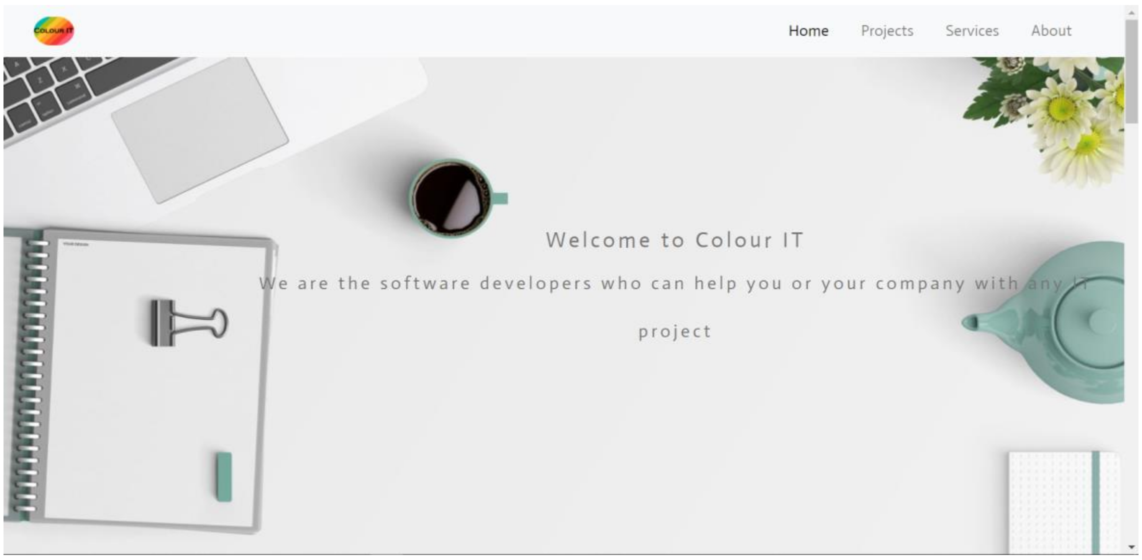


Projects List	Deadline	Status	Members	ID
Project	2020/12/19	Not star...	0	813

This is one of the most important windows in the program. Although it is simple. From this window you can access all created projects, requirements inside of it and tasks inside the requirement. This design was chosen to not to confuse the consumer when choosing what to edit and what to select. It is clear what project you are editing and inside of that, what requirements and tasks are presented.



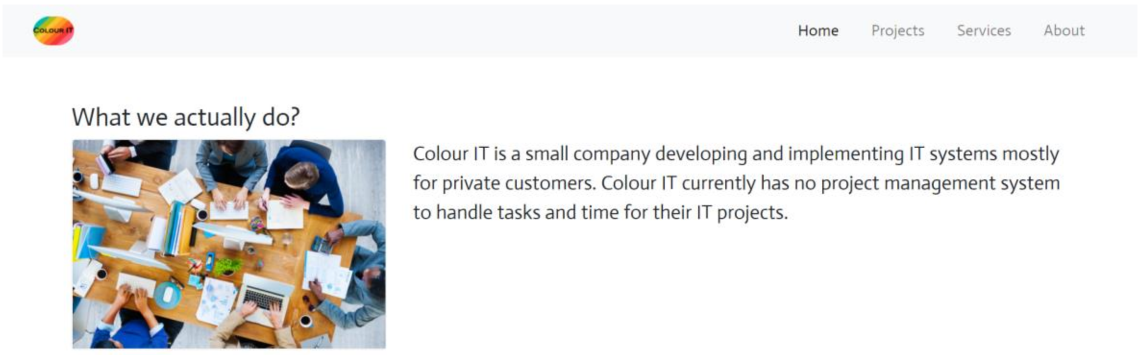
This screen shot represents an important part, because we think that it is vital for our program to have confirmation button on removal to prevent accidental removal of the project/requirement/task.



Welcome to Colour IT

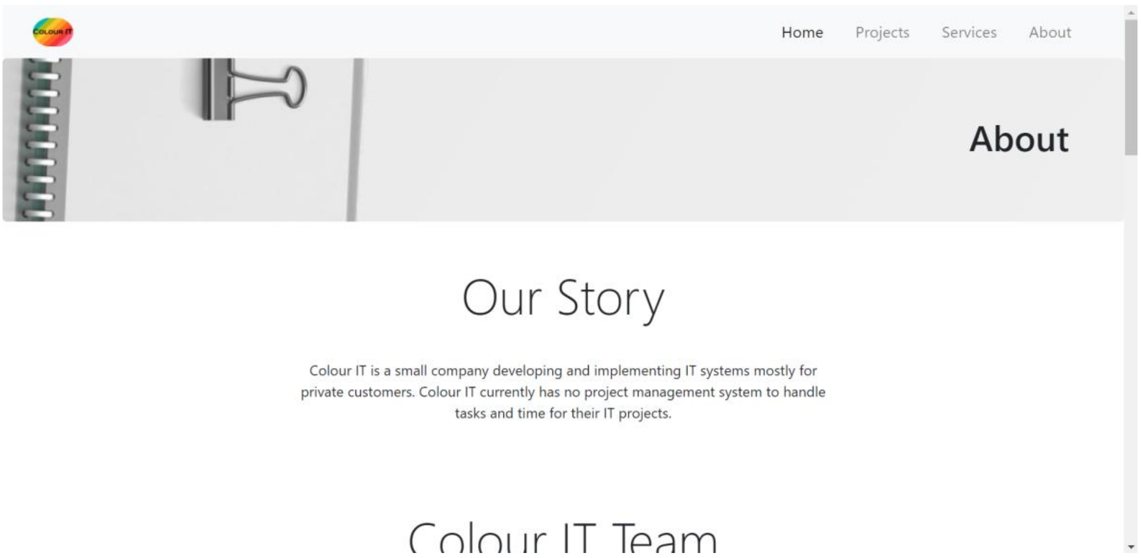
We are the software developers who can help you or your company with any IT project

With the website the decision was made to go with a simple – easy to read design. The pages do not contain a lot of information to not overwhelm the consumer.



What we actually do?

Colour IT is a small company developing and implementing IT systems mostly for private customers. Colour IT currently has no project management system to handle tasks and time for their IT projects.



About

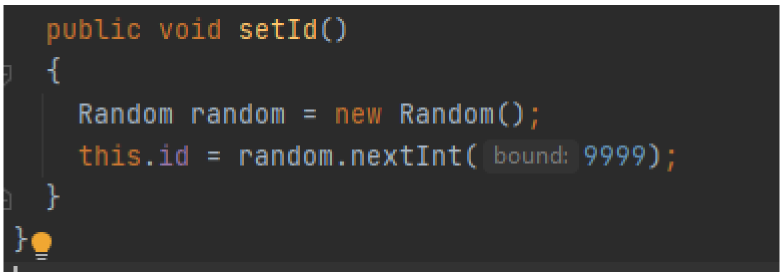
Our Story

Colour IT is a small company developing and implementing IT systems mostly for private customers. Colour IT currently has no project management system to handle tasks and time for their IT projects.

Colour IT Team

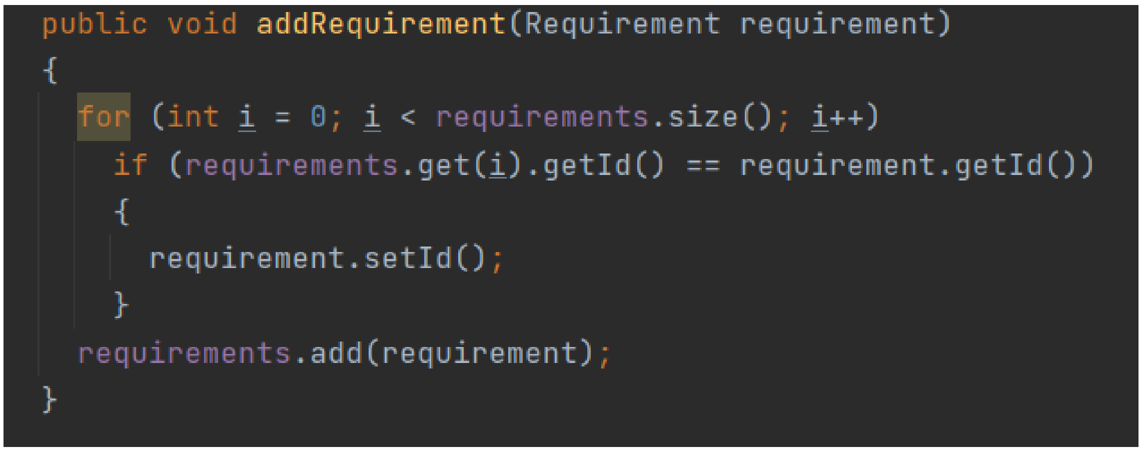
In about section there is all the necessary information displayed in one page for easy navigation.

4 Implementation



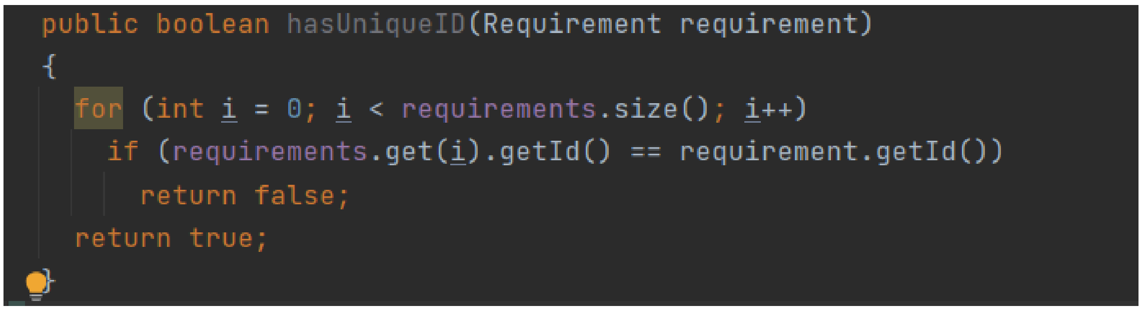
```
public void setId()
{
    Random random = new Random();
    this.id = random.nextInt( bound: 9999);
}
}
```

Figure 1.a public setid() – Requirement



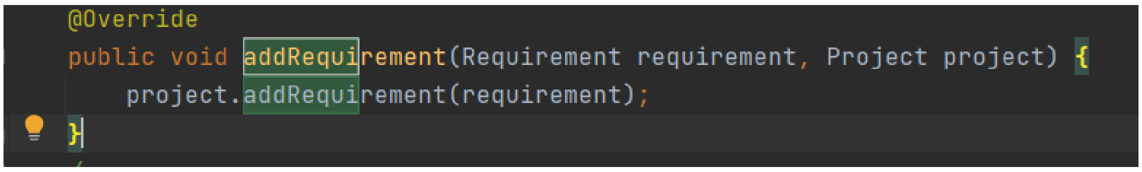
```
public void addRequirement(Requirement requirement)
{
  for (int i = 0; i < requirements.size(); i++)
    if (requirements.get(i).getId() == requirement.getId())
    {
      requirement.setId();
    }
  requirements.add(requirement);
}
```

Figure 1.b public addRequirement(Requirement requirement) – Project



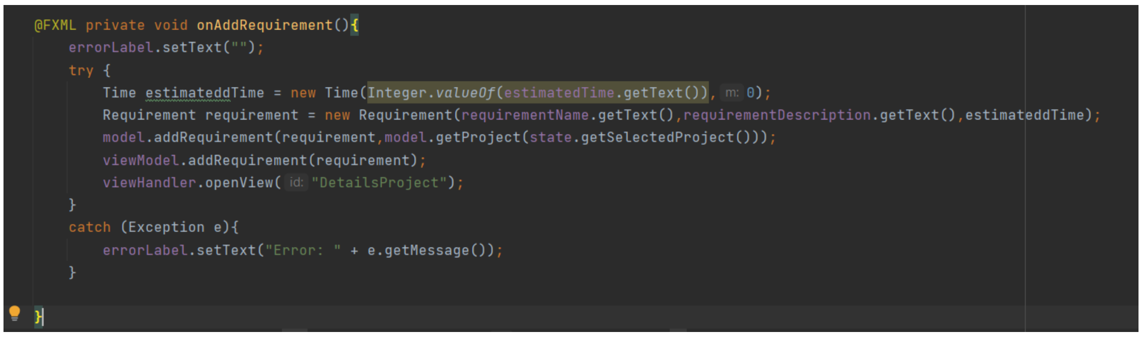
```
public boolean hasUniqueID(Requirement requirement)
{
  for (int i = 0; i < requirements.size(); i++)
    if (requirements.get(i).getId() == requirement.getId())
      return false;
  return true;
}
```

Figure 1.c public boolean hasUniqueID(Requirement requirement) – Project



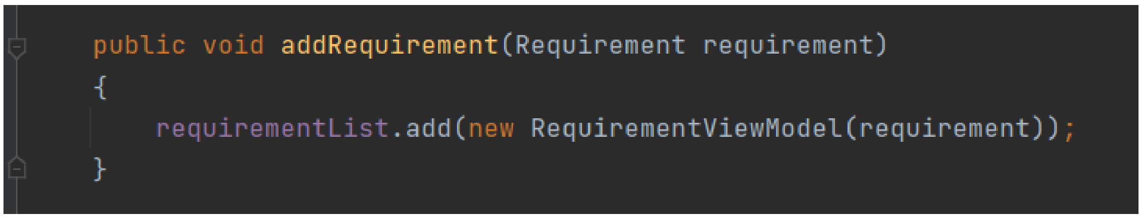
```
    @Override
    public void addRequirement(Requirement requirement, Project project) {
        project.addRequirement(requirement);
    }
```

Figure 1.d public void addRequirement(Requirement requirement, Project project) – ProjectModelManager



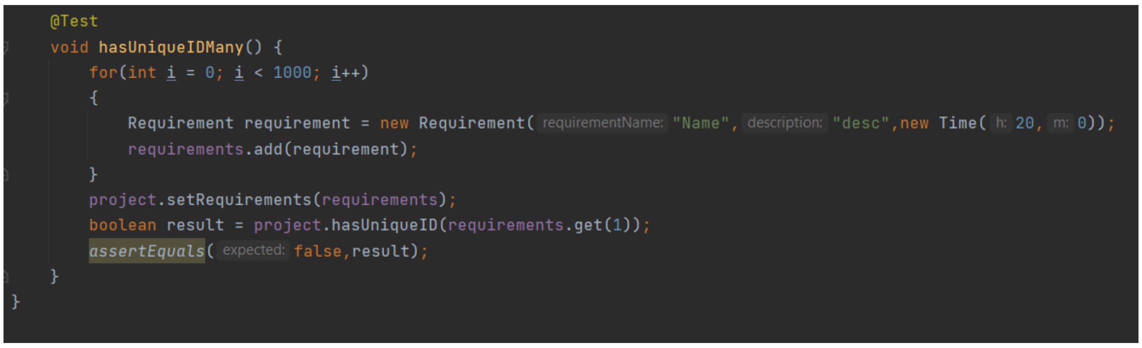
```
@FXML private void onAddRequirement(){
    errorLabel.setText("");
    try {
        Time estimateddTime = new Time(Integer.valueOf(estimatedTime.getText()), m: 0);
        Requirement requirement = new Requirement(requirementName.getText(),requirementDescription.getText(),estimateddTime);
        model.addRequirement(requirement,model.getProject(state.getSelectedProject()));
        viewModel.addRequirement(requirement);
        viewHandler.openView( id: "DetailsProject");
    }
    catch (Exception e){
        errorLabel.setText("Error: " + e.getMessage());
    }
}
```

Figure 1.e @FXML private void onAddRequirement() – AddRequirementViewController



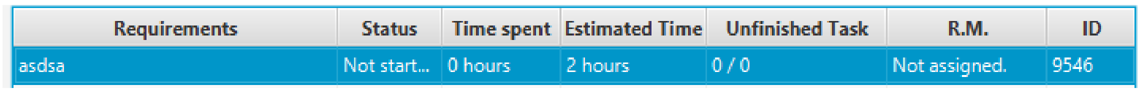
```
public void addRequirement(Requirement requirement)
{
    requirementList.add(new RequirementViewModel(requirement));
}
```

Figure 1.f public void addRequirement(Requirement requirement) – RequirementListViewModel



```
    @Test
    void hasUniqueIDMany() {
        for(int i = 0; i < 1000; i++)
        {
            Requirement requirement = new Requirement( requirementName: "Name", description: "desc",new Time( h: 20, m: 0));
            requirements.add(requirement);
        }
        project.setRequirements(requirements);
        boolean result = project.hasUniqueID(requirements.get(1));
        assertEquals( expected: false,result);
    }
}
```

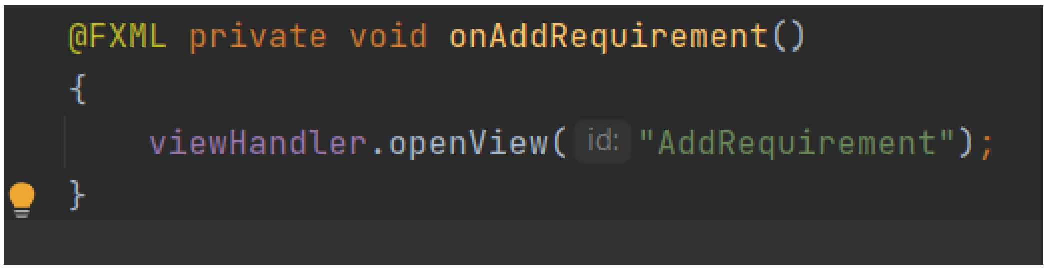
Figure 1.g public void hasUniqueIDMany() – ProjectTest



Requirements	Status	Time spent	Estimated Time	Unfinished Task	R.M.	ID
asdsa	Not start...	0 hours	2 hours	0 / 0	Not assigned.	9546

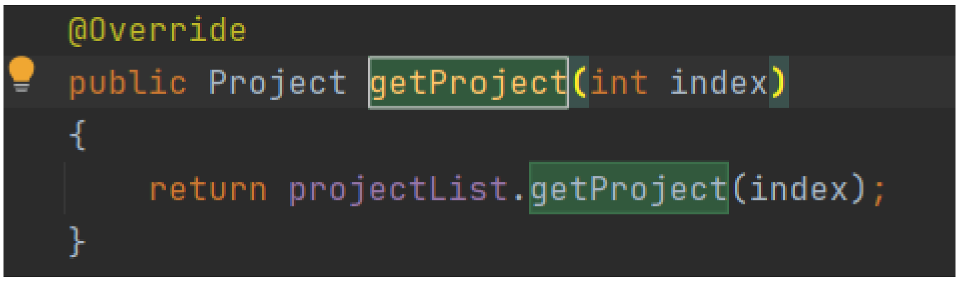
Figure 1.h – Requirements list shown in JavaFX GUI represented by DetailsProjectViewController.

Figure 1.i – button that goes in “Add requirementViewController” by implementation of figure 1.j.



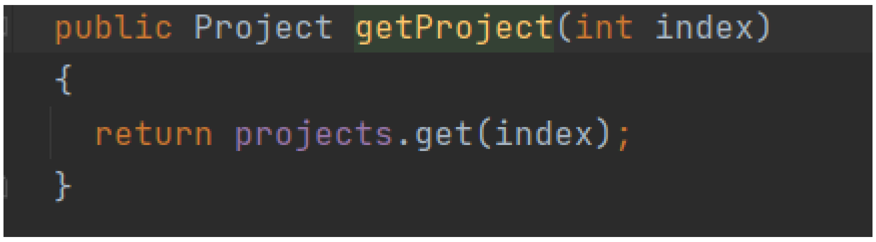
```
@FXML private void onAddRequirement()
{
    viewHandler.openView( id: "AddRequirement");
}
```

Figure 1.j - @FXML private boid onAddRequirement() – DetailsProjectViewController



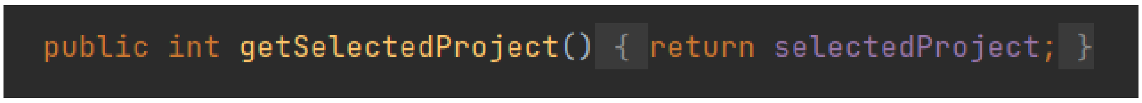
```java
@Override
public Project getProject(int index)
{
    return projectList.getProject(index);
}
```

Figure 1.k – public Project getProject(int index) – ProjectModelManager



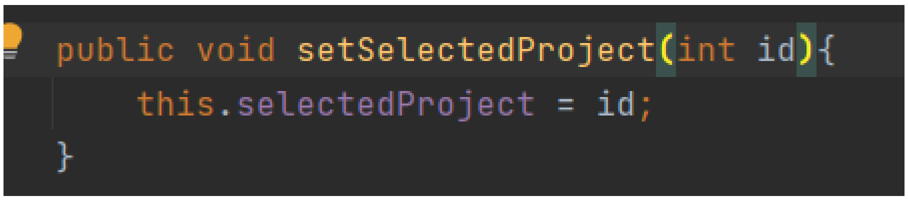
```java
public Project getProject(int index)
{
  return projects.get(index);
}
```

Figure 1.l – public Project getProject(int index) – ProjectList



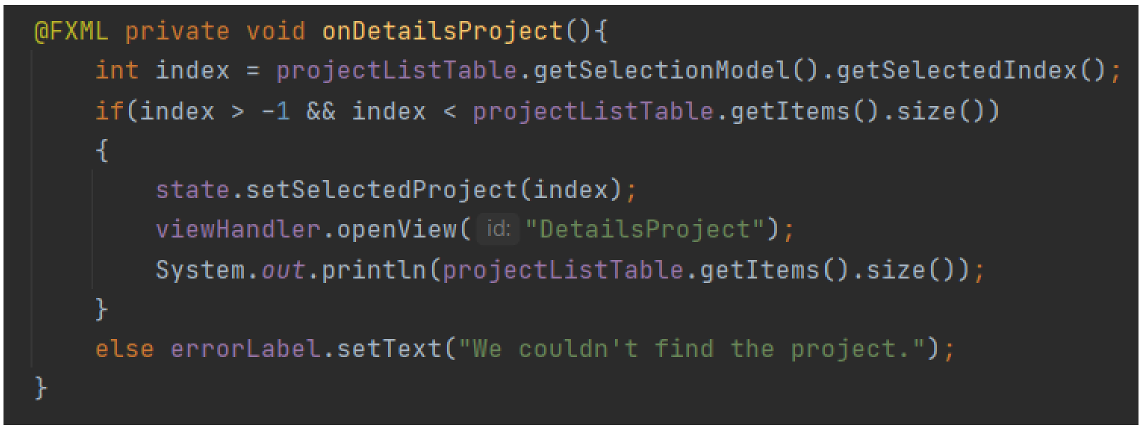
```java
public int getSelectedProject() { return selectedProject; }
```

Figure 1.m – public int getSelectedProject() - ViewState



```java
public void setSelectedProject(int id){
    this.selectedProject = id;
}
```

Figure 1.n – public void setSelectedProject(int id) - ViewState



```java
@FXML private void onDetailsProject(){
    int index = projectListTable.getSelectionModel().getSelectedIndex();
    if(index > -1 && index < projectListTable.getItems().size())
    {
        state.setSelectedProject(index);
        viewHandler.openView( id: "DetailsProject");
        System.out.println(projectListTable.getItems().size());
    }
    else errorLabel.setText("We couldn't find the project.");
}
```

Figure 1.o – private void onDetailsProject() – ProjectListViewController

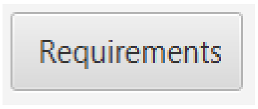
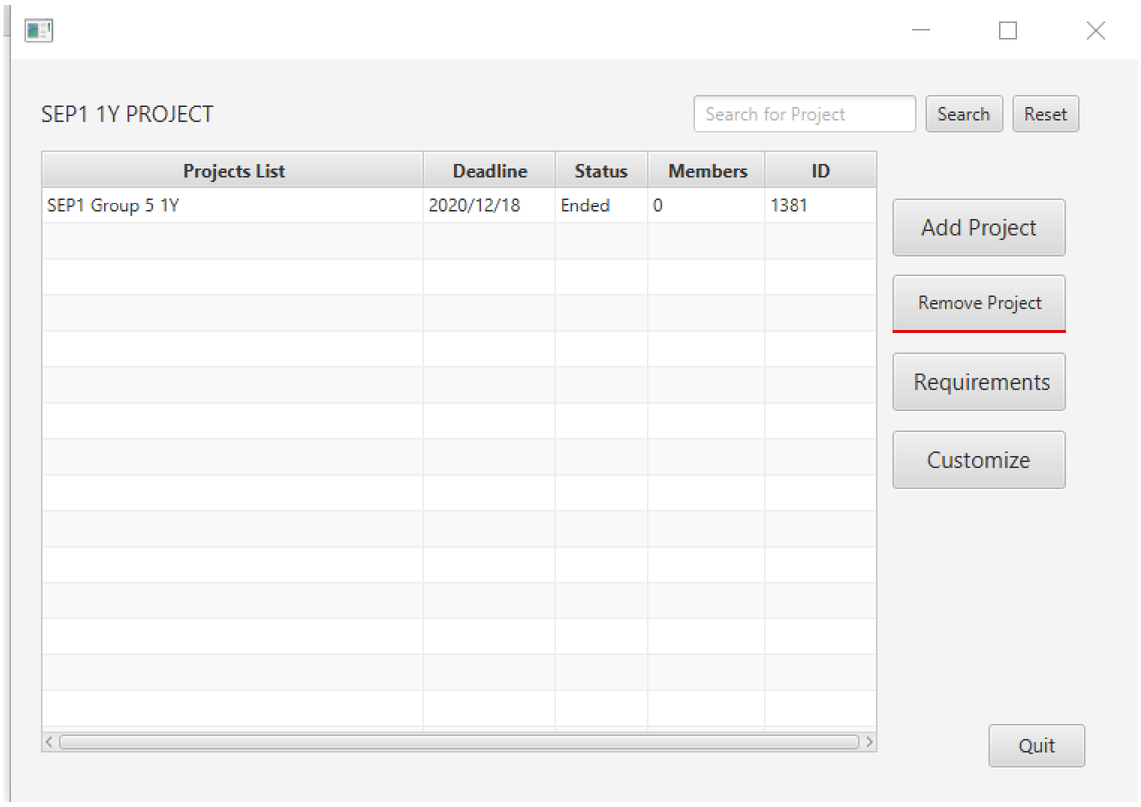
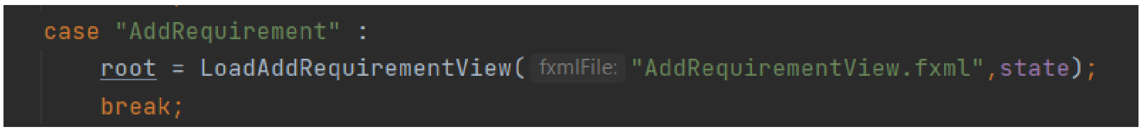


Figure 1.p – button "Requirements". Represents the code of **figure 1.o**.



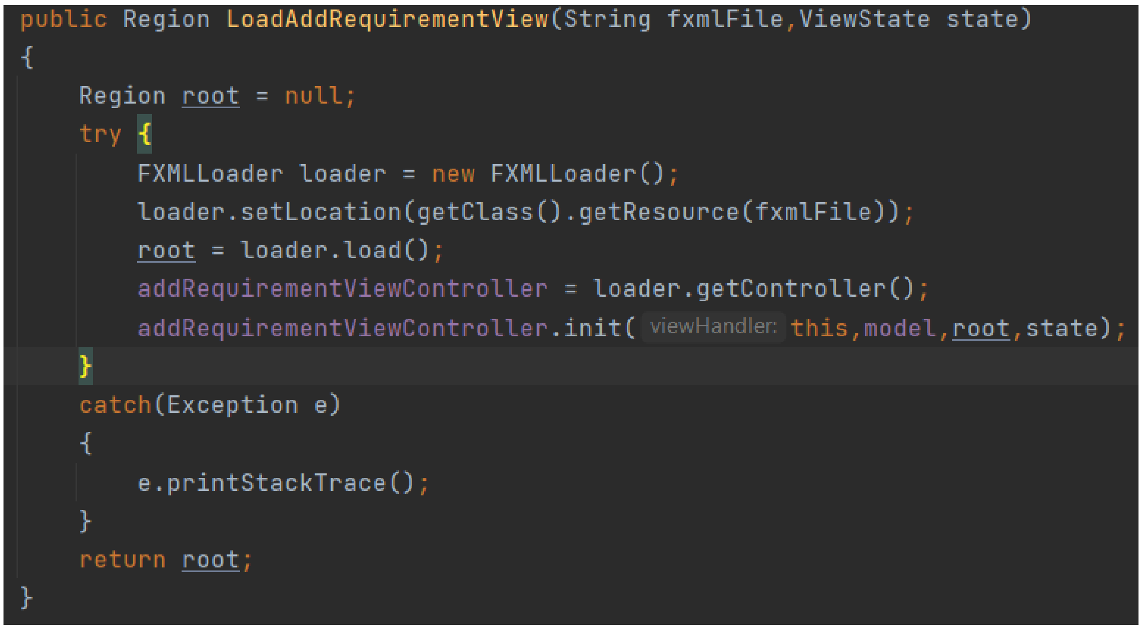
Projects List	Deadline	Status	Members	ID
SEP1 Group 5 1Y	2020/12/18	Ended	0	1381

Figure 1.q – JavaFX GUI for ProjectListController



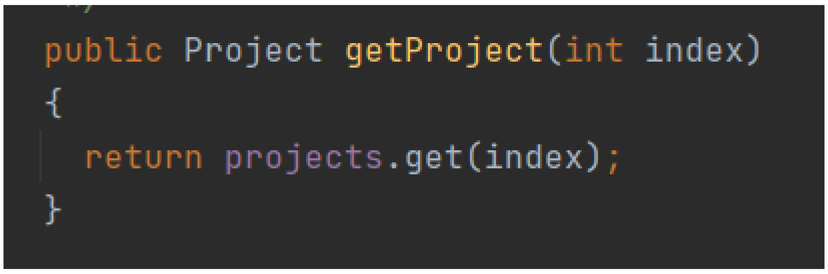
```
case "AddRequirement" :
    root = LoadAddRequirementView( fxmlFile: "AddRequirementView.fxml",state);
    break;
```

Figure 1.r – "Add Requirement" case for switch from ViewHandler.



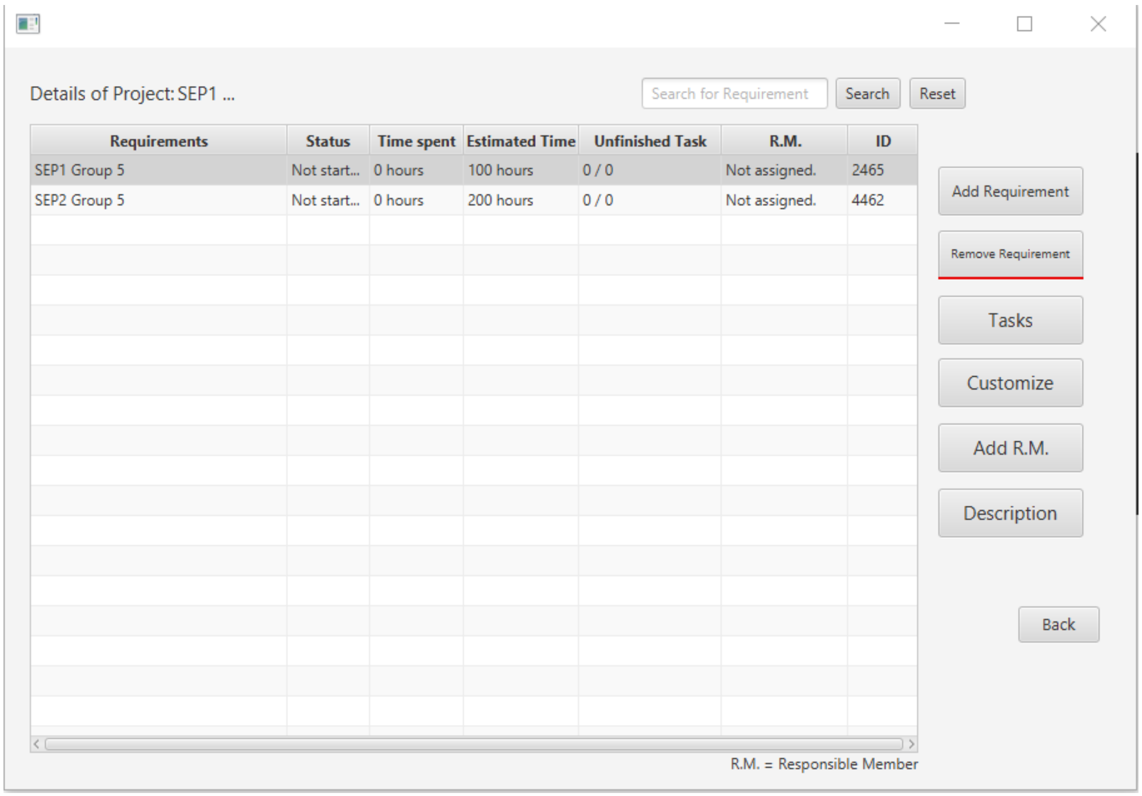
```
public Region LoadAddRequirementView(String fxmlFile,ViewState state)
{
    Region root = null;
    try {
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(getClass().getResource(fxmlFile));
        root = loader.load();
        addRequirementViewController = loader.getController();
        addRequirementViewController.init( viewHandler: this,model,root,state);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return root;
}
```

Figure 1.s – public Region LoadAddRequirementView(String fxmlFile, ViewState state)



```
public Project getProject(int index)
{
  return projects.get(index);
}
```

Figure 1.t – public Project getProject(int index)


Details of Project: SEP1 ...

Requirements	Status	Time spent	Estimated Time	Unfinished Task	R.M.	ID
SEP1 Group 5	Not start...	0 hours	100 hours	0 / 0	Not assigned.	2465
SEP2 Group 5	Not start...	0 hours	200 hours	0 / 0	Not assigned.	4462

R.M. = Responsible Member

Figure 1.u – represents table of requirements list of a selected project.

The most relevant function that has been pointed is implementation for unique IDs. **Figure 1.a** represents a void function because It does not have to return something. It is a public method because will be used in the next future for the anothers figures. It represents a setter for requirement's id. It sets a random number of Integer type that is bounded from 0 to 9999. This function was created for testing part.
Figure 1.b represents a void function because it does not have to return something. It goes in a loop and searches firstly if requirement, that is about to be added, has the same id with another requirement already added. If true, then function "void setId()" it will be called and id of the requirement that is in the paranthesis will be changed with another id that is different from another requirement's id added in the past. If false, then it will go simply to last part of function, adding the requirement and the same thing will be done if it is true, after it goes throught the loop.

Figure 1.c represents boolean function that returns the state of truth. It is a public function because it was used for testing part from **figure 1.g**. It checks if every requirement has an unique ID. In figure 1.c it checks, in a for loop, the id of every requirement already added and requirement that is in paranthesis. If the state finds a requirement with the same ID, then will return true, otherwise, false.
Figure 1.d shows a public function and because of that it is used in model interface for next future implementations for GUI. It is void because I does not require to return something. The two parameters, requirement and project are used to add requirement wanted in the current project selected in JavaFX Application. It adds a new requirement, using the function “addRequirement” explained in **figure 1.b**.
Figure 1.f is part for code from **RequirementListViewModel**. It is a public function because it will be used for part of code in **figure 1.e**. It has void type because the function does not return something, it adds a new requirement on **requirementList** which is a **ObservableList** parameter and uses the function “add” from **java.util.list library**.
Figure 1.e represents a FXML function because it is called by a button from JavaFX GUI. It has private because it has to be called just in **AddRequirementViewController**.
Firstly, it initializes **errorLabel** which is a **FXML Label** used for showing errors that could appear while the application is running.
Secondly, it creates an estimatedTime of Time class type with hours that are written. It represents the time estimated of finishing a requirement. This parameter is created because it will be needed for creating a new requirement parameter. Time has aggregation with Requirement.The column that shows the estimated hours is illustrated in **figure 1.g**. After that, requirement is created with parameters required: name of requirement, description and time. Furthermore, unique ID does not have to be created by hand. It is hardcoded. **Name of requirement** is hardcoded by **requirementName TextField FXML** and **requirementDescription TextField FXML**.
Thirdly, the requirement is created by calling “addRequirement” function explained in **figure 1.d**, using instance variable “model” from **AddRequirementViewController** class. It gets the project as parameter of Project type using ViewState class, see **figure**

1.m. The id of the state, for getting project, it gets when you select a project and press "Requirement" button (see **figure 1.p**).

Fourthly, after adding the requirement in the **Requirement List Table**, it also needs to be added too in the viewmodel by **"viewModel.addRequirement(requirement)"** function, see **figure 1.f**, then it gets back to principal window where all requirement are showed (**figure 1.q**).

Figure 1.g represents a void method that does not have to return something, it has to be tested instead. This function is used for the testing part of **"hasUniqueID(Requirement requirement)"** from **Project class** (see **figure 1.c**).

First part it is represented by a loop who creates and adds 1000 requirements in an ArrayList of Requirement type in order to be tested, to see if every requirement has **Unique ID**.

Figure 1.j represents a FXML void method that implements function for **Add Requirement button** (**see figure 1.i**).

This method is private because it is used just for JavaFX GUI on "**AddRequirementViewController**". It calls method viewHandler.openView("Add Requirement") represented by **figure 1.r** and **figure 1.s**.

Here it does not require to use ViewState methods because it just adds another requirement, not customizes or removes it.

Figure 1.k – public Project getProject(int index) – ProjectModelManager

Figure 1.k represents a method from **ProjectModelManager class.** It is public because it is used for implementation in **view package**. It returns a project.

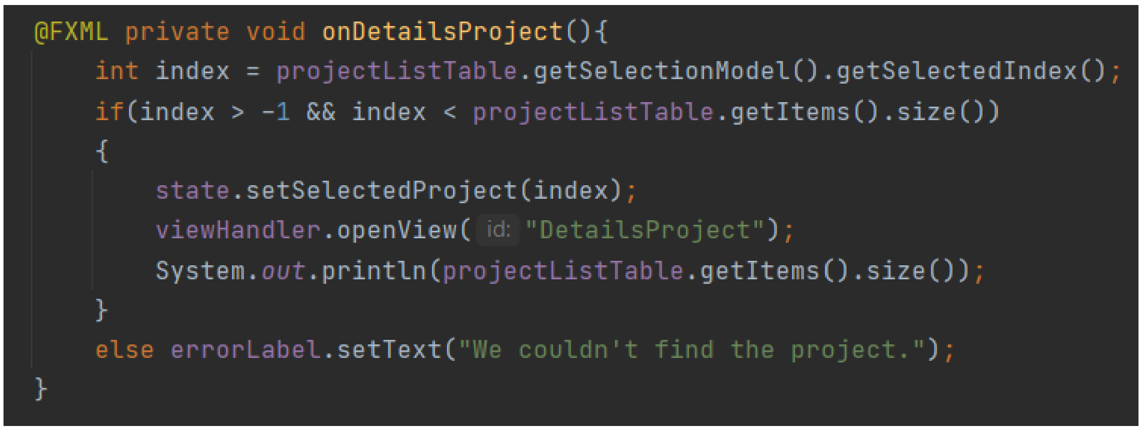
It uses method **public Project getProject(int index)** from **ProjectList class** (see **figure 1.t**).

Because **projects** parameter it is an ArrayList, the method "**get**" it is called from **java.util.ArrayList library**. It has an **Integer** parameter **index** which will be used for function "**projects.get(index)"**.

Figure 1.m represents a method from **ViewState class** that returns an integer as parameter. It is a public method because it will be used in **package view**.

Figure 1.n represents a setter method from **ViewState class**. It does not have to return something. It has to set the id when requirements for a project are opened.

It calls the **selectedProject** parameter and equals it with **ID** used in paranthesis.



```
@FXML private void onDetailsProject(){
    int index = projectListTable.getSelectionModel().getSelectedIndex();
    if(index > -1 && index < projectListTable.getItems().size())
    {
        state.setSelectedProject(index);
        viewHandler.openView( id: "DetailsProject");
        System.out.println(projectListTable.getItems().size());
    }
    else errorLabel.setText("We couldn't find the project.");
}
```

Figure 1.o – private void onDetailsProject() – ProjectListViewController

Figure 1.o represents a FXML method that opens requirements for a project selected. It is private because it is used just in **ProjectListViewController**.

Firstly, it equals the index of Integer type with index of selected project from table list.

Secondly, if index is -1 (if it is -1 then no project was selected) or more then size of entire project list, then an errorLabel, which is a **FXML Label** for errors will appear and message will be set with “**We could not find the project**”.

Thirdly, if index is between parameters given, **state parameter of ViewState type** will be setted with the index that was setted in the beginning of method and a list with requirements of selected project will appear (**see figure 1.u**).

The entire process it starts from **ProjectModelManager**.

Before creating a new requirement, it requires to have a project already created and then selected. Pressing the “Requirement” button a windows with all requirements will be shown. The entire process it goes from **ProjectListViewController** to **DetailsProjectViewController** were **“Add Requirement”** is implemented.

Pressing “**Add Requirement”** button requires filling all boxes shown by **AddRequirementViewController**. After filling all boxes, it requires to press on **“Add”** button and requirement is created.

5 Test

Use Case	Test result	Comments
Manage projects	Works	Projects can be created, removed, their names, description, deadline can be added. Projects can be customized, viewed, in that section status can be changed and added. However, projects are not read from or written to a file.
Manage tasks	Works	Tasks can be created, removed and searched. Task names, descriptions, time span can be added. Tasks can be customized and in this section status can be added, or customized. Time spent on tasks can be added, members responsible for tasks and their roles can be selected, added, customized, viewed and removed. However, tasks and their content are not being read from, or written to a file.
Manage requirements	Works	Requirements can be created, removed. Their names, descriptions, estimated time can be added. Requirements names, descriptions, can be customized, changed, responsible team members can be added. Requirements can be viewed and searched. However, requirements are not being read from, or written to a file.

6 Results and Discussion

The system we developed is not completely done. Although it is not done, we still managed to create a working prototype which has almost all working functionality of desired system. It can add projects, delete them, save projects to file, edit projects, search them and so on. The flaws are that requirements and tasks for the project cannot be saved to file and the status is not linked with the website we designed.

Requirement number	Status
1	Implemented
2	Implemented
3	Implemented
4	Implemented
5	Not implemented
6	Implemented
7	Not implemented
8	Not implemented
9	Not implemented
10	Implemented
11	Implemented
12	Implemented
13	Implemented
14	Implemented
15	Implemented
16	Implemented
17	Implemented
18	Implemented
19	Implemented
20	Implemented

21	Implemented
22	Implemented
23	Implemented
24	Implemented
25	Implemented
26	Implemented
27	Not implemented

The fifth requirement has not been implemented at all. There is a start date which is assigned with date when project or requirement is created.

Seventh requirement is not implemented. Time was used to add and see how much time members spent on a task. There is another time spent for requirements. Registering time spent for a task, will increase, with the same amount typed, time spent on requirement chosen.

The eighth requirement is not implemented at all. On a project there is not a button for assign members. Responsible team member should be added in every requirement that is created and rest of the other members should be added on every task of selected requirement. Task has more members. After adding every member, in "member" column from project list table the number will be increased with one, but there is no feature to see member details from project list.

The nineth requirement is not implemented at all too. The start date is assigned with the date when project and requirement is created.

The twenty seventh requirement is not implemented. There is a problem with implementation of writing to file. It could not be done with just one file. Because of three windows of listing projects, requirements and tasks, it required to make more files than one.

This system is working fine, although it does not have all functionalities. Every button, its functionality is doing what it supposed to do and are working well. However, there are some flaws and ways to improve this system. First of all, it does not read or write information about objects into files, because writing and reading class files were unable to find files path. Furthermore, if our writing and reading part was working, we did not optimize it. Instead of rewriting all files, it could have changed or added certain parts of information into the file. This would not create any problems in this program, but in a bigger program with more data this would create a huge period of time just to overwrite the file. The search feature could also include more search options, not just by project, requirement ID. This would help end user to easier search for the requirement or project they want.

7 Conclusions

The system is developed for small businesses with simple and easy to understand design. The system will help companies to manage time and workload. This is new and innovative solution to manage projects. The customer dictated the requirements and we analyzed them. From analyses we found that the key is simplicity and reliability. Uncomplicated design would help with the basic understanding of the system fundamentals. The requirements were very basic. The customer wants to add, remove, customize projects and content in it. It is also requested that JavaFX is used as GUI tool and the system is coded with Java. The structure is very basic. The goal was to make an easy to develop software that would still work as intended. In general, the system is working without errors and has working functions like: managing projects, requirements, tasks, members, deadlines and etc. However, few requirements were not implemented into the system, so it does not have a full functionality, of what was expected. In the end, our system is working properly and after a few changes and added writing to file and reading from it, it would be good enough to be used in an actual working environment even though a few requirements were not implemented.

8 Sources of information

CareerAddict Team, 2018. The 9 Key Benefits of Project Management Software. Available at: https://www.careeraddict.com/project-management-software-benefits
[Accessed September 23, 2020]
Vartika Kashyap, 2020. 10 Project Management Tools The World's Top Companies Actually Use. Available at: https://www.projecttimes.com/articles/10-project-management-tools-the-worlds-top-companies-actually-use.html
[Accessed September 23, 2020]

Courtney Cavey, May 29, 2020: A Guide to an Effective Team Workload Management. Available at: https://blog.hubstaff.com/effective-workload-management/ [Accessed September 23, 2020]

9 Appendices

All appendices were uploaded in “Appendices” folder from project document.