|  |
| --- |
| **Scrum – Up! Project Report** |

**Florin Bordei, 280593**

**Dave Le, 280071**

**Jaume Lopez, 282231**

**Supervisors:**

**Joseph Chukwudi Okika**

**Jakob Knob Rasmussen**

**34.920 number of characters**

**ICT Engineering**

**3rd Semester**

**19/12/2019**

**Table of content**

[Abstract iv](#_Toc501021071)

[1 Introduction 1](#_Toc501021072)

[2 Requirements 2](#_Toc501021073)

[2.1 Functional Requirements 2](#_Toc501021074)

[2.2 Non-Functional Requirements 2](#_Toc501021075)

[3 Analysis 3](#_Toc501021076)

[4 Design 4](#_Toc501021077)

[5 Implementation 5](#_Toc501021078)

[6 Test 6](#_Toc501021079)

[6.1 Test Specifications 6](#_Toc501021080)

[7 Results and Discussion 7](#_Toc501021081)

[8 Conclusions 8](#_Toc501021082)

[9 Project future 9](#_Toc501021083)

[10 Sources of information 10](#_Toc501021084)

[11 Appendices 1](#_Toc501021085)

**List of figures and tables**

Optional

# Introduction

The Danish educational system is well known worldwide for its efficiency and for the qualitative results it produces over the students engaged in it. As many European educational systems as theoretical based, the Danish one focuses mostly on the practicality of the studies, in the sense that theory should be learnt how to be implemented and not just memorized. Opposite many other European educational systems, it also embraces teamwork and collaboration so that traits similar to a regular workplace can be comprehended in the early life of the students. Here comes into play the PBL approach to learning, as it helps students to make a deeper integration of theoretical and practical learnings. At the same time, it supports collaborative learning and helps students to become more motivated and independent learners with a deeper subject understanding.

VIA University College holds such a practice inside of their educational program for Software Engineering, called “Semester Project”. The course takes place each semester and it involves unique methods and requirements to be fulfilled by the students. Therefore, the students must identify a problem on which they are required to work collaboratively and find a solution using certain methods and frameworks.

Starting with the first semester, students apply the Waterfall model, that comes with certain limitations, most significant being ones being the high amount of risks and lack of flexibility.

As with their second semester, students are introduced to new methodologies and frameworks more complex that they can apply inside of their Semester Project. As an alternative to Waterfall model, the students are required to use Unified Process, that is a popular iterative development process for building object – oriented systems. As mentioned before, UP overcomes the limitations of the Waterfall model by being able to deliver fast core parts of the system through its iterations. Each phases compromises of different stages to be completed before delivering the running parts of the system: analysis, design, implementation and testing. Here intervenes one of the challenges for the students are they have a tendency to skip some important phases as they “rush into coding”. Such challenges affect the learning process of the new methods and frameworks as being not implementing them, students are not able to fully understand their advantages and scope behind it.

Alongside Unified Process, students are also required to use SCRUM framework, which addresses complex adaptive problem. The framework consists of Scrum Teams and their associate roles, events, artifacts and rules. Each component of the frameworks serves a specific purpose and it is essential to Scrum’s success and usage. The rules of Scrum bind together the roles, events and artifacts governing the relationships and interaction between them.

The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master. The Product Owner is responsible for maximizing the value of the product resulting from work of the Development Team. The Development Team consists of professionals who do the work of delivering a potentially releasable increment of “Done” product at the end of each Sprint. The Scrum Master facilitates the implementation of Scrum framework and is a servant-leader for the Scrum Team. Scrum Teams are self-organizing and cross-functional. The teams deliver products iteratively and incrementally, maximizing opportunities for feedback, incremental deliveries of “Done” product ensure a potentially useful version of working product is always available.

The heart of Scrum is a Sprint, a time-box of one month or less during which a “Done”, useable, and potentially releasable product Increment is created. Sprints have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint. Sprints contain and consist of the Sprint Planning, Daily Scrums, the development work, the Sprint Review, and the Sprint Retrospective.

Artefacts defined by Scrum are specifically designed to maximize transparency of key information so that everybody has the same understanding of the artifact. The Product Backlog is an ordered list of everything that is known to be needed in the product. It is the single source of requirements for any changes to be made to the product. The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal. The Scrum Burndown Chart is a visual measurement tool that shows the completed work per day against the projected rate of completion for the current project release. Its purpose is to enable that the project is on the track to deliver the expected solution within the desired schedule.

Even though the Scrum and UP are simple to understand, it is difficult to master and fully implement them as it has been observed from previous second semester students.

Regarding Scrum, initiating and managing the artefacts causes misunderstandings among the students as there are not many templates available. Additionally, there are issues in storing artefacts in a logical manner. For example, in the Sprint Backlog the students omit the assigning the responsibility of a user story to one of the member’s group and breaking down the user story into multiple tasks with their specific story points.

Consequently, if the user story was not broken down into tasks, the Product Backlogs’ user story points will not be successfully updated, this situation includes also Burndown chart. Another issue the students are facing while using the Scrum framework is that they are not properly following the order of events and a lack of a clear overview regarding the issues that have been reported in each meeting. Because of a badly stored and managed data, it is difficult to extract required documentations for the Project and Process Reports. In some cases, it has been noticed a lack of communication from the student’s side towards their supervisors and this has created situations, where the supervisors were not aware of the lack of progress.

Regarding UP, the biggest challenge faced by the students is a rush to code instead of following strictly UP model. That is a consequence of misunderstanding UP model, but also a lack of task management. To help the reader to understand better, in each iteration (Sprint) the selected user stories are being considered as “Done” once the following have been completed: Analysis, Design, Implementation, Testing, Deployment.

Within such a dynamic environment, Group 3 has decided to try to address to all the issues above and develop a system that will assist the group of students in their efforts to apply SCRUM and Unified Process. The system presented in this Project Report serves the purpose of storing information on the status of the development of the project and what phases have been completed or overlooked. Also, it will give the possibility of the user to have a direct overview on the project but also on the group status and information.

The general aim is to provide an “one space storage” instead of a work methodology where information is being kept in different files and sources.

The next section will give insights into the requirements of the system. The mentioned requirements have been extracted from the user stories gathered by the team members during their investigation.

# Requirements

In the first part of this section we will present the collected user stories, followed by the requirements.

In the collected user stories, we have identified different roles and different actors but we had to generalize some roles as they had the potential to substitute each other.

As the Scrum contains 3 different roles (Scrum Master, Product Owner and Developer) we have decided to generalize the roles of Scrum Master and Product Owner under the terminology of “Manager”. Such decision has been taken as we’ve considered that there is a high potential to substitute each other in case of sickness or leave. Therefore, the Product Owner and Scrum Master will be assigned under their roles in the system but as acting behavior, they will have access to same tools and functionalities.

User is the general actor that accesses the system, creates an account or log in into the system. Developer is a version of this User account as he has limited access, mostly view functions and limited abilities to input data.

Teacher is the version of actor that is able to access the system, have an overview of the groups work and able to communicate with the groups.

**User**

# As a user I need to be able to create an account so I can be added to a group.

# As a user I need to be able to change my credentials so I can keep my account secure.

# As a user I need to be able to log in to the system so I can access the system.

# As a user I need to be able to create a group so that our sprints can be organized.

# As a user I need to be able to create and remove comments on (sprint) backlog items to effectively communicate ideas between the group members.

# As a user I need to be able to react to any comment to effectively communicate between group members

**Developer**

1. As a developer I need to be able to complete tasks and stories of any sprint to the data can be updated

**Supervisor**

# The supervisor needs to be able to see the sprint information of any group related to it to effectively assist them

# As a supervisor I need to able to communicate with the group to able to announce them on the new meeting times.

**Manager**

1. As a manager I need to be able to add and remove members from a group to keep the correct members in the group
2. As a manager I need to be able to remove group members from a group to keep the correct members in the group
3. As a manager I need to able to view my group members and their assigned roles so I can have a complete overview of my group
4. As a manager I need to be able to change the access of the members so that everyone has the correct authorization
5. As a manager I need to be able to create a product backlog to store the user stories of the system
6. As a manager I need to be able to add user stories to the product backlog to keep track of the current user stories
7. As a manager I need to be able to remove user stories from the product backlog to keep track of the current user stories
8. As a manager I need to be able to assign story points to a user story to accurately depict the progress
9. As a manager I need to be able to view the backlog items from Product Backlog so I can be aware of the current status of the project
10. As a manager I need to be able to change the Product Backlog items to keep them accurate
11. As a manager I need to be able to set the number of sprints so that we can have an overview of the total number of sprints
12. As a manager I need to be able to see the current Sprint Backlog so I can have an overview of the current progress.
13. As a manager I need to be able to edit the number of Sprints in case the initial number was a mistake.
14. As a manager I need to be able to remove user stories to a sprint to keep the accounts accurate
15. As a manager I need to be able to set the duration of each sprints so that they accurately depict the time spent
16. As a manager I need to be able remove user stories to a sprint to keep the accounts accurate
17. As a manager I need to be able to add tasks to user stories to break down tasks and make them more approachable
18. As a manager I need to be able to see the status of the current tasks so I can be aware of the progress of my team.
19. As a manager I need to be able to complete tasks and stories of any sprint to the data can be updated
20. As a manager I need to be able to create a log for the day to keep accurate documentation of the project
21. As a manager I need to be able to input attendance and hours worked and issues of the day to the daily log to keep accurate documentation on the progress
22. As a manager I need to be able to create a sprint daily meeting note to keep accurate documentation of the project
23. As a manager I need to be able to create a retrospective meeting note to keep accurate documentation of the project
24. As a manager I need to be able to create a Review meeting to keep accurate documentation of the project
25. As a manager I need to be able to create a planning meeting note to keep accurate documentation of the project
26. As a manager I need to be able to view the meeting notes so I can have an overview on the situation
27. As a manager I need to be able to edit the meetings so I can avoid human errors.

Based on the selected user stories, the following functional and non-functional requirements have been formulated:

## Functional Requirements

1. User must be able to create an account.
2. User must be able to log in into the system.
3. User must be able to change his credentials.
4. User must be able to create a group
5. User must be able to create comments.
6. User must be able to reply to comments.
7. Developer must be able to complete his tasks in the systems.
8. System must be able to update the status of the tasks.
9. Manager must be able to add new members to the group.
10. Manager must be able to remove members from the group.
11. Manager must be able to view the group formation.
12. Manager must be able to change roles of team members .
13. Manager must be able to create a product backlog.
14. Manager must be able to add user stories to product backlog.
15. Manager must be able to remove user stories from product backlog.
16. Manager must be able to assign story points to the user stories.
17. Manager must be able to view the product backlog.
18. Manager must be able to edit the product backlog.
19. Manager must be able to set the number of Sprints.
20. Manager must be able to set the duration of each Sprint.
21. Manager must be able to see the current Sprint.
22. Manage must be able to add user stories to Sprint Backlog.
23. Manager must be able to remove user stories to Sprint Backlog.
24. Manager must be able to assign tasks to user stories.
25. Manager must be able to set complete on tasks.
26. Manager must be able to create a log.
27. Manager must be able to input data in log.
28. Manager must be able to input data in Sprint Daily Meeting.
29. Manager must be able to input data in Review Meeting.
30. Manager must be able to input data in Retrospective Meeting.
31. Manager must be able to view meeting notes.
32. Manager must be able to edit meeting notes.
33. Manager must be able to input data in Planning Meeting.
34. Supervisor must be able to see the sprint status.
35. Supervisor must be able to communicate with the groups.
36. System must re-calculate the number of story points.
37. System must distinguish between completed and incomplete tasks.
38. System must distinguish between completed and incomplete user stories.
39. System must be able to generate a burndown chart.

## Non-Functional Requirements

1. The system must on a 3tier Arhitecture (Presentation – Business – Database)
2. The system must be using a database for persistance
3. The system should make use of the database
4. The system must contain Client/Server infrastructure

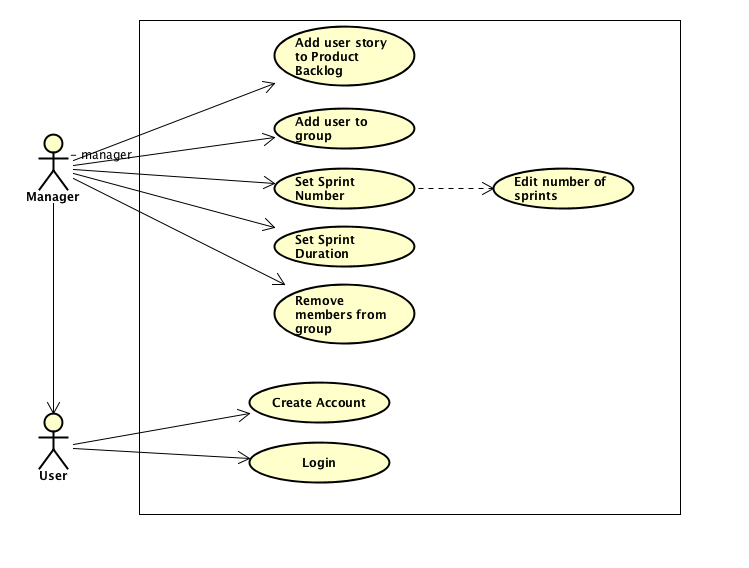
# Analysis

## Analysis

According to the collected user stories and resulted requirements, the objective is to set up a system that would enable users to input data, observe data and extract data from the system. We have identified the following main actors that will have a direct impact in the system:

* User – the inception phase of each user before being part of a group where it takes a different role.
* Developer – limited access to actions
* Manager (known also as Scrum Master or Product Owner) – has superior access than the Developer as he is able to change data in the system; and he is able to communicate with the Supervisor in the name of the group
* Supervisor (known also as Teacher) – that has similar access like the Developer, but he is also able to communicate with the group through the manager

As being able to identify the main actors and having the required data, it is possible to start building the use case diagrams that is a dynamic behavior diagram in UML. The use case diagrams model the functionality of the system using actors and use cases.



As we can see in this specific Use Case Diagram, we have two actors that have different actions and interactions in the system:

* User:
* is able to create an account
* is able to log in into the system
* Manager:
* Incorporates all the functionalities of a User
* Is able to add a User Story to Product Backlog
* Is able to add a User to a group where it will take a new role
* Is able to remove a group member from group
* Is able to set the number of Sprints which extends the action of Editing the number of Sprints
* Is able to set the duration of each Sprint

Identifying the main actors and their main actions and interactions with the system we can proceed in formulating the Use Case Descriptions which incorporates the following elements:

* A short textual description of the main actor
* A short textual description of the actions and interactions that the main actor will perform in the system
* Sequences of steps of the actions and interactions
* Post-Conditions and Pre-Conditions of these actions
* Exception sequences, if there is the case

For an example, one of the core parts of the system will be used for example reference:

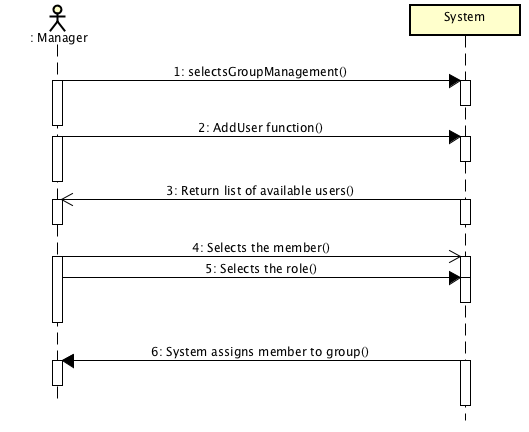
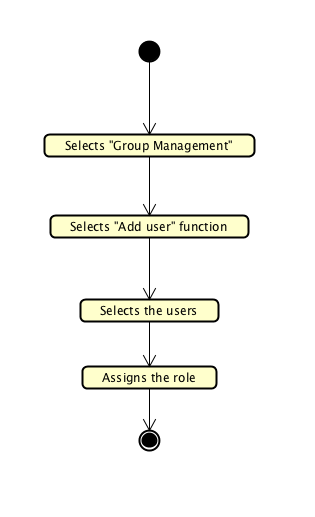
*As a manager I need to be able to add and remove members from a group to keep the correct members in the group*

Scrum Master is the servant-leader of the group. If the group decides to expand the group, to let additional members of the group have access to the specific Scrum Team project and all of its content in the tool, the Scrum Master needs to be able to add them to the group.

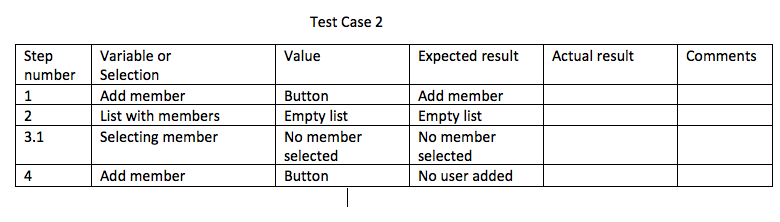
|  |  |
| --- | --- |
| Use case | **Add users to a group** |
| Summary | The Manager is able to add users to a group |
| Actor | Manager |
| Precondition | The Manager is logged in the system  The group is already created |
| Postcondition | Members have access to group’s project content  Added members |
| Base sequence | 1. Manager selects “Group Management” menu. 2. Manager selects “Add user” function. 3. System returns the list with existing users.   3.a Manager selects member to add.  3.b Manager assigns role to member.  4. System saves the selected member in the group. |
| Exception sequence | 2. No existing users  2.1 Cancel  4. Assigns Product Owner/Scrum Master, the role is already assigned  4.1 Another role to assign |
| Sub use case |  |
| Notes | User can cancel at any time. |

To be observed that this Use Case Description incorporates all the elements mentioned earlier. We have the textual description where it provides context on the main actor and the desired action to have it performed in the system. The preconditions of this use case is that the Manager should be already logged in, the group exists. As a postcondition, the members will be added, and they will have access to the system. The Base sequence describes the steps that need to be followed in order to perform and complete the action while the Exception sequence sets the context of what issues there may be involved in this process.

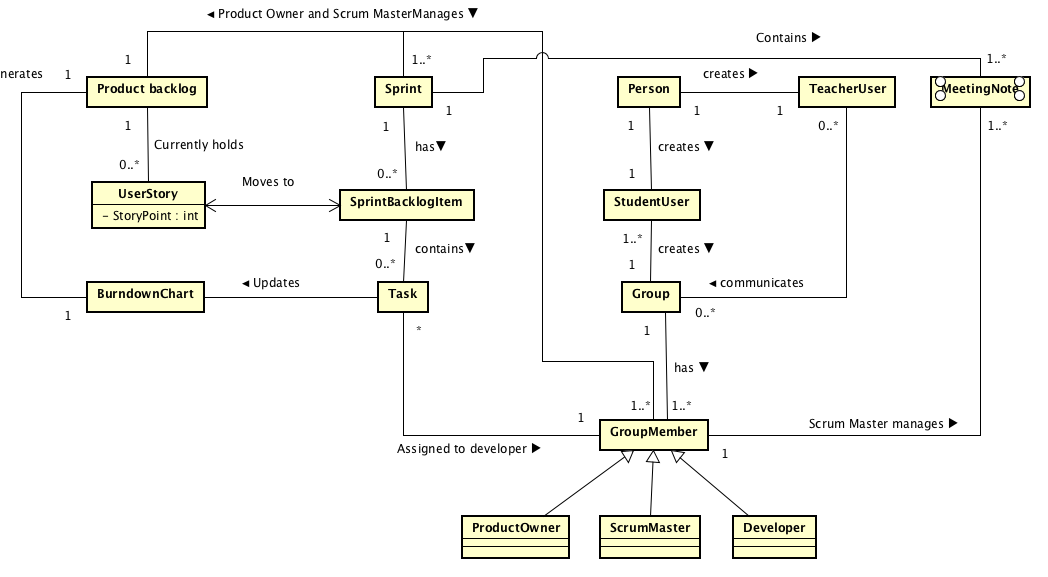
For further scenario clarifications, the team has used an Activity Diagram and a System Sequence Diagram:



And as for a Scenario verification we can use Test Case:



Once these steps have been completed, the domain model it can be drawn. Domain model is the last sequence of the Analysis as it is a conceptual modelling, priori to Class Diagram from Design phase. It incorporates behavior and data. The rest of files on Analysis can be viewed in APPENDIX B.



**3.2 Security**

It is needed to assess the security of the system to assure the success of it and also possibility of the user to make use of it. It is essential for the users to be assured that their access to the system is always available and unique access and also the safety and the integrity of data that exists in the system.

As it is needed to analyze possible threats and vulnerabilities, Threat model has been formulated to list all the possible threats that the system can face. Also, the goal and means of the attacker are mentioned for each threat by also including the place and the person being able to perform such an attack.

Threat model

1. DoS/ DDoS attack

The attacker sends multiple packets of data (such as a large number of user stories to be added to Product Backlogs/ large number of groups to be created) and the system will run out of resources and it will not be able to perform all the requests. As a consequence, the system will crash, and it will not be available to work.

1. Goal of the attacker

The goal of the attacker is Denial of Service and the objective is the Availability of the system – prevent

1. Means of the Attacker

Active – Blocking the application

1. EINOO
2. WHO – Internal as the system requires creation of an account and it is not available in guest mode
3. WHERE – online
4. Unauthorize access to the Database

The attack is able to access the mainframe of the system where he is able to make significant changes in the database where all the users’ information is stored or even delete the data.

1. Goal of the attacker

Tampering – the attacker will be able to modify data stored for each specific user (example: change group, change user stories to make damage, beneficial changes)

Elevation of privileges – the attacker can create from inside a super account with much more superior access than the rest of the users

Inform

Spoofing Identity – the attacker will be able to impersonate other users (Authentication)

Denial of Service – the attacker can delete from the system user accounts that will not be able to access the system

Information Disclosure – the attacker will have access to private data (Confidentiality)

1. Means of the Attacker

Active – Modification

Active – Blocking

1. EINOO
2. WHO – External
3. WHERE – Online
4. Access to manager account

In this scenario the attacker can use a Brute Force attack where he will be able to obtain the passwords of the managers and he will be able to make significant changes in the data stored in the system for each Scrum Group (can delete Product Backlog, modify user story points etc.)

1. Goal of the attacker

Spoofing Identity – he impersonates other users (Authentication)

Tampering: manipulates data (Integrity)

Information Disclosure – access to private data (Confidentiality)

Repudiation: attacker will be able to deny his actions as he acted on behalf of another account

1. Means of the Attacker

Active – Modification

Passive – Information disclosure

1. EINOO
2. WHO – Internal
3. WHERE – Online
4. Access to developer account

In this scenario the attacker can use a Brute Force attack where he will be able to obtain the passwords of the developer and he will be able to make significant changes in the data stored in the system for each Scrum Group (can make changes in the Sprint Backlog and mark all tasks as completed)

1. Goal of the attacker

Spoofing Identity – he impersonates other users (Authentication)

Tampering: manipulates data (Integrity)

Information Disclosure – access to private data (Confidentiality)

Repudiation: attacker will be able to deny his actions as he acted on behalf of another account

1. Means of the Attacker

Active – Modification

Passive - Information disclosure (reading messages with other group members)

1. EINOO
2. WHO – Internal
3. WHERE – Online
4. Modification attack on communication between Group – Supervisor

In this scenario the attacker intercepts the messages that are being sent from the Group or from the Supervisor to the Group and the messages are being modified before reaching their final destination. It can be a danger for when the students ask for an input on an obstacle they face in their project.

1. Goal of the attacker

Tampering – manipulates data (Integrity)

1. Means of the Attacker

Active – Modification

1. EINOO
2. WHO – Internal
3. WHERE – Online
4. Manually shutting down the server

If the server is not located in a secure location, the attacker can access it and manually shut it down

1. Goal of the attacker

Denial of Service – the system will not be available (Availability)

1. Means of the attacker

Active – Blocking

1. EINOO
2. WHO – External
3. WHERE – Offline

Risk Assessment model

In the previous section we have mentioned all the possible threats that are able to happen. As now for Risk Assessment we will look into the frequency, what preventive measures can be taken, how we can prevent it, what possible threat effects we might have, the corrective measures that should be applied and the risk if the attack will happen. Threat frequency has been measured with the indicators of Low, Medium, High.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attack/Threat** | **Threat frequency** | **Preventive Measures /**  **Vulnerabilities** | **Incident Prevention** | **Threat effect** | **Corrective Measures** | **Risk** |
| DoS/ DDoS attack | Medium | Secure configuration of the server | Replace current server/ Third party allocation | System is not able to perform | Contact the server owner | System will not be available for a period of time. |
| Unauthorized access to database | Medium | Public password | Store safely the password/ Back-up server | Loss of data, alterations or stealing identities | Frequent back-ups and change of passwords | Inconsistencies in data, stolen accounts |
| Unauthorized access to manager account | High | Encrypted passwords | Digital signatures, back-up, changes of passwords | Loss of data, alterations | Change passwords and back-up | Potential loss of data, accounts |
| Unauthorized access to developer account | Low | Encrypted passwords | Digital signatures, back-up, changes of password | Loss of data, alterations | Change passwords and back-ups | Potential loss of data, accounts |
| Modification attack | Medium | Secure configuration on the server | Digital signature | Messages has been changed | Restore backup | Inconsistency in communications |
| Manually shutting down the server | Low | Server located in a third-party company | Have a back-up server active | System is not able to perform | Contact the server owner/ Activate back-up server | System will be unavailable for a period of time |

# Design

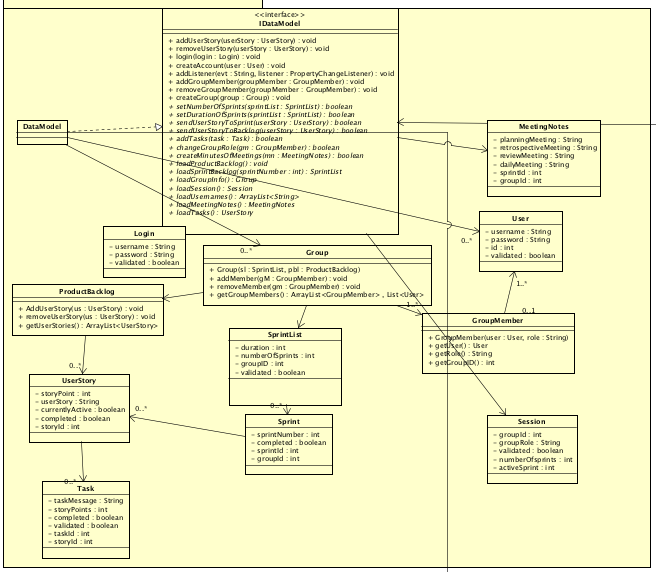
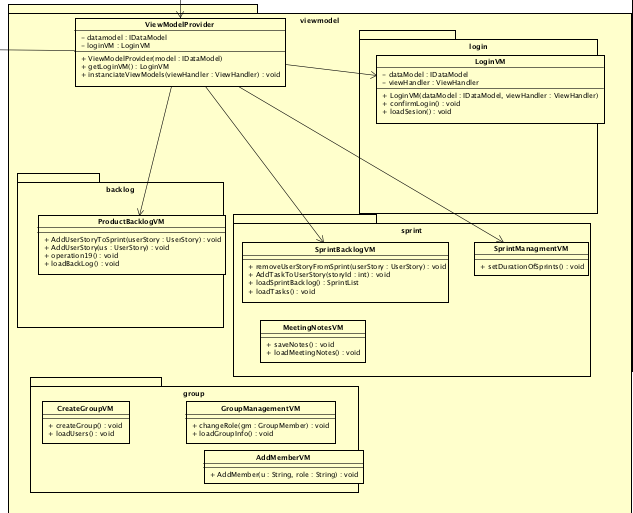
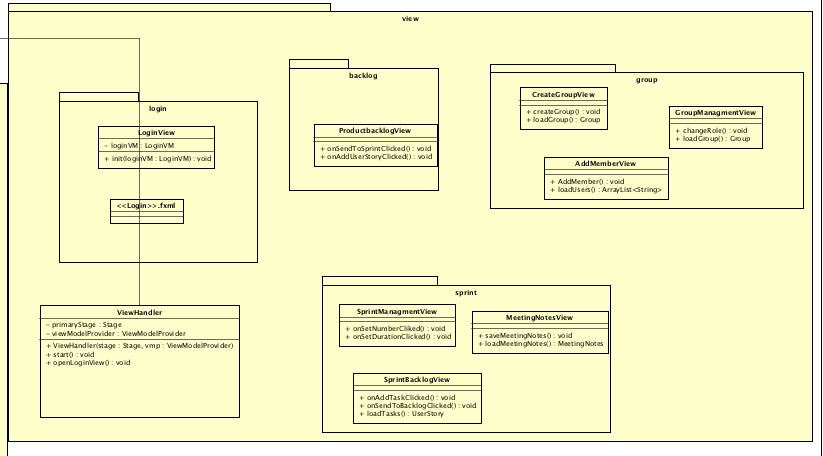
Design is derived from the Analysis and this is the section of the project where the classes and the relations between classes can be classified. Design is prior to coding implementation. To be noted that small parts of Design will be presented while the rest of the Design can be accessed through Appendix C.

4.1. Arhitecture

The 3 tier architecture pattern has been chosen as it is a distributed system. This pattern gives different responsibilities to each tier. Another advantage of this pattern is also the scalability and flexibility as we can grow the system or add different components without any interruptions on the current active ones.

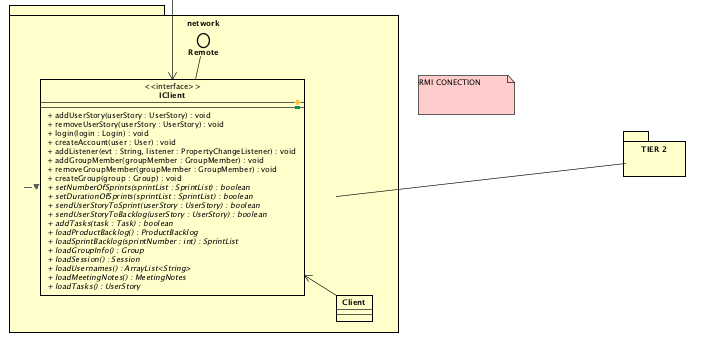
4.2 Presentation Layer

The basic principle of the Presentation Layer is that it will not provide any logical or storing operations. The only purpose of it is to have used for displaying data or to facilitate the input of data. The chosen GUI’s have been built in Scene Builder and behind them lies the MVVM design pattern. The MVVM pattern that separates the Model from the View Model and the View. To be noted that the View is aware of the View Model and the View Model is aware of the Model but not vice versa. This makes it uncoupled for future developments or changes in the View.



All the operations or data to be sent or to be retrieved they follow the path of View -> View Model -> Model -> Client.

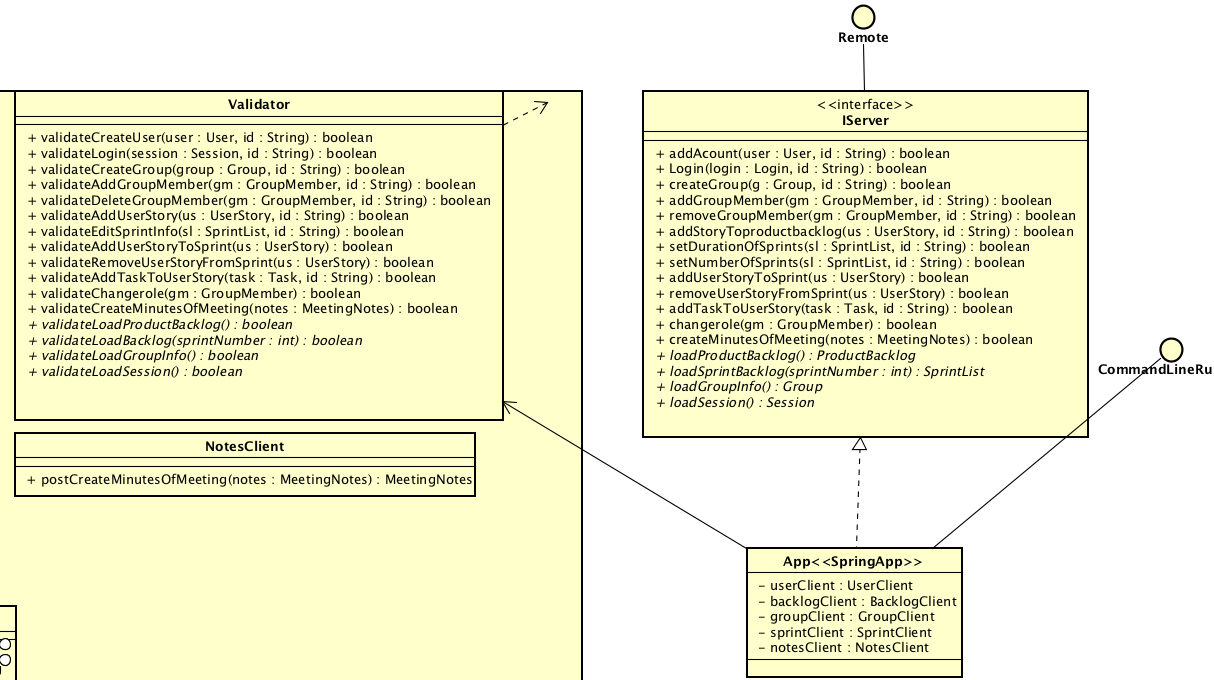
Client is using RMI design pattern that allows the Client to invoke methods on an IServer that exists in Tier 2, that allows to send or receive data through the Tiers. Objects are serialized when sent and desterilized when retrieved.



4.3 Application Layer

The basic principle of the Application Layer is to perform the required operations and methods for the whole system. It can retrieve data from the Presentation Layer to process it and to send in the Third Tier if it complies with the requirements of the Application Layer. In the same, it performs the vice-versa route as in retrieving data from the Third Layer to the Presentation Layer.

As a simple example, a login object received from the Presentation Layer will be sent directly to the Third Tier, the database will perform the necessary operations (comparing with object already stored in) and it will return back to the Application Layer with the login object that will set the validate (boolean) to true and then it goes through the Validator class which will check if the Boolean is true and other basic operations.

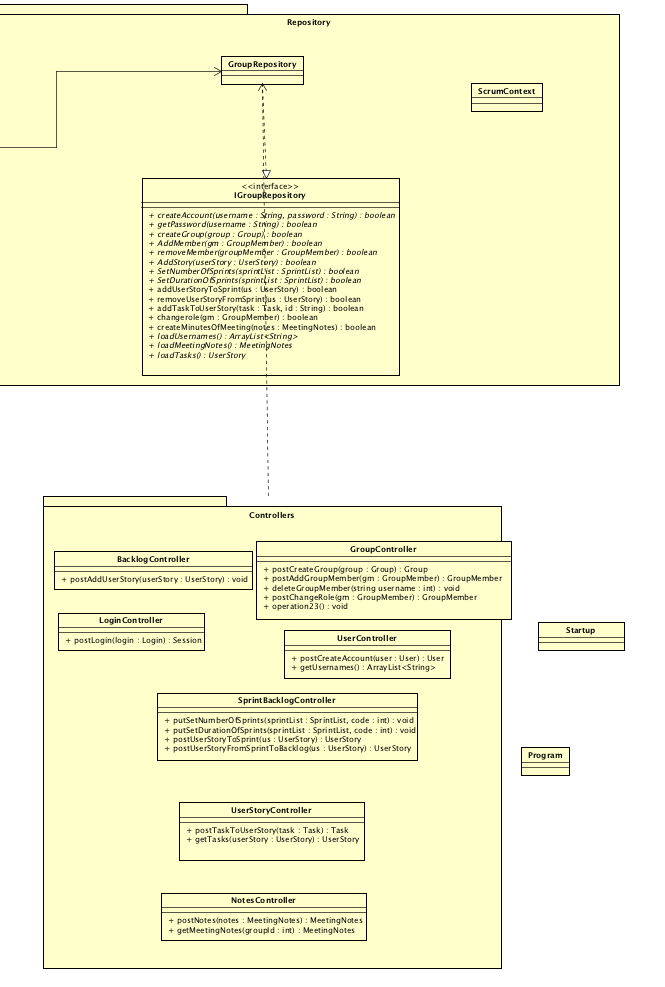


As for connection between the Tier 2 and Tier 3 we have used REST API. Using the basic CRUD operations which are get, delete, put and post.

4.3 Database Layer

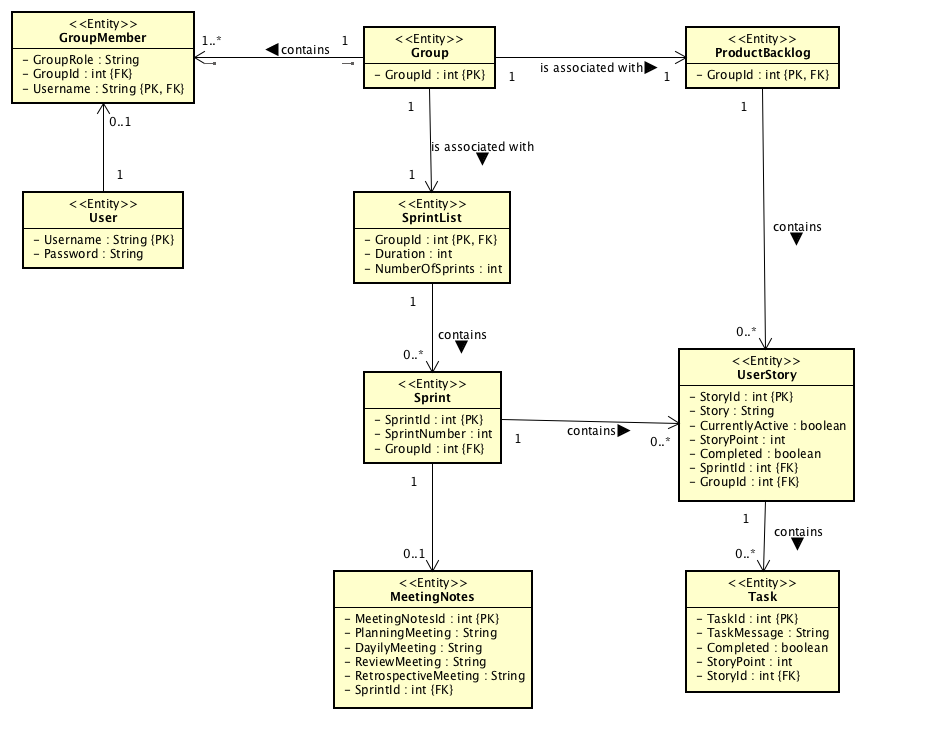
To make the database, the team used Entity Framework Core, the technology that is suggested to access data and is intended to be used with .NET Core applications To achieve the database, the system was developed with code-first approach. That means, that the classes for domain entities are created firstly, followed by making the database based on the code using migrations. The reason of choosing the approach is, that the team had no clear image of how the database should look.

Now in the third layer of the system when the HTTP request is received, it is processed by the Controller classes and sent to the Group Repository.



Further, from the Group Repository is going to the SCRUM context (database) and returns the Boolean or object back to the Controller and from the Controller to the Application Layer.

For the database we have identified different entities in the ER diagram with their assigned relationships.



4.3 Security Design

The below measures have been designed in order to prevent or protect the system against attacks.

Proactive mechanisms.

1. Authentication protocol

The information that is being used at the log in form (username, password) are being sent to the database to be checked if they are correct and the system should return a confirmation, otherwise an error. The security component should issue a session key that is limited in lifetime.

1. Password encryption

When the user is creating a new account, the system should use an encryption type SHA-256, that will be stored in the database. At each attempt of login, the hash keys (one from login, one stored in database) are being compared. If the attacker has no access to the system, the method of attack will be of Brute Force. If there are any data leaks from the system, the attacker may use Ciphertext Attack as the attacker may get in the possession of the ciphertext.

1. Back-up

In case of attacks whereas a consequence there are inconsistencies in the system it would be required to create unique daily back-ups of the system and stored in remote and safe locations.

1. Digital signature

Digital signature it can be useful to prevent Identity Spoofing, a proof of authorization. It could be used when creating a group, removing or adding members from groups.

1. Frequent changes of passwords

There could a policy in place where it will be required to have the password changed every 3-4 weeks and it would be required for the password to a minimum of 8 characters long (including containing numbers, uppercase, lowercase, symbols).

Reactive mechanisms

1. Restore backup

This would bring the original data present in the system before the attack and the state of affairs can be continued.

1. Change of passwords

If any account has been attacked, it can be restored back to the original user by reset and changing the password with enforcing the policies of regular changes of the passwords.

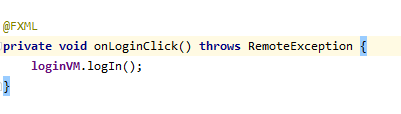
1. Change to a third-party server hosting party

If the own server does not cover many security mechanisms, it would be recommended to buy hosting services of servers from a third party specialized in the field with a clean records of attacks and successful preventions history.

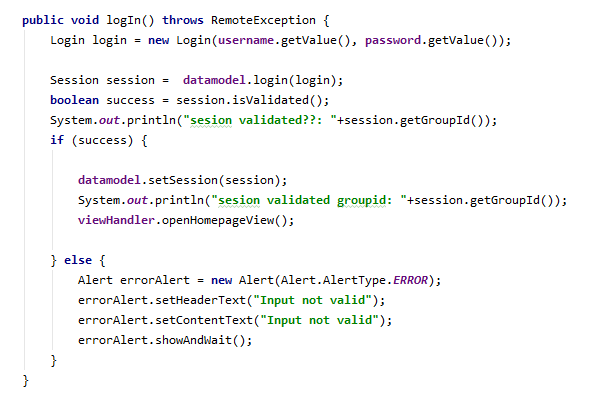
# Implementation

For the section of Implementation, the group will use as for example the implementation of the log in requirement.

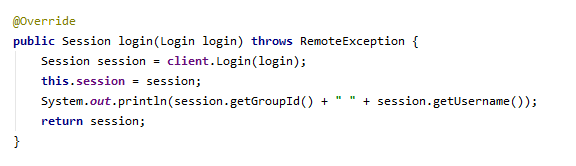
When the user fills in the log in fields and press Login, it will call this method which calls another method in the VM.



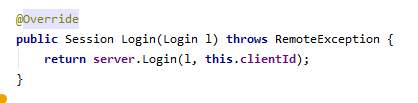
In the View model it creates a Login object with the username and password input by the user. Then it will create a Session object based on the return from the call in data model login method. We will create a Boolean based on the field and validated in the object we receive in the return method. If the Boolean is true, it will set the session in the data model and open the home page view, if not it will pop a error message.



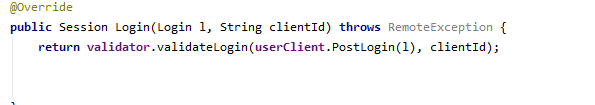
This will method will create a session based on the return from the Client Login method and return it to the View Model



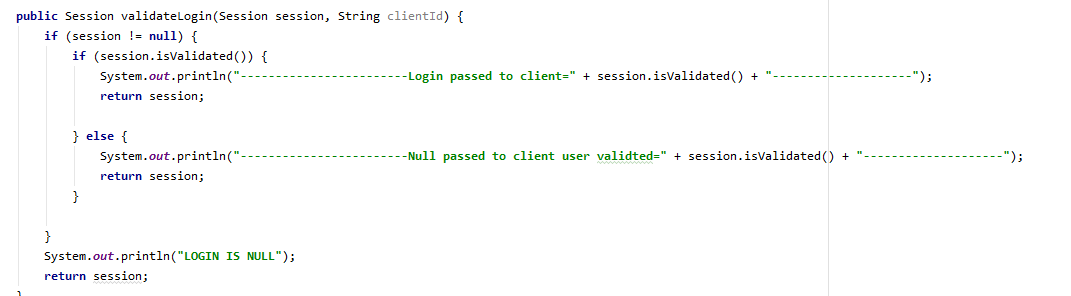
Then this method will basically return the object from the server Login method



In the server login method, it calls the validator to check if all parameters are correct based on the database response



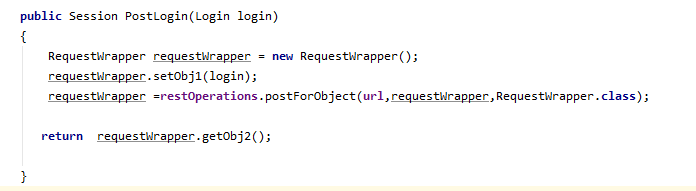
If the Session is not null and its validated it will pass the login object to the call else it will return a empty object with no data



In the User Client it creates a Request Wrapper object and set the login object inside then it will make a post operation to Tier 3 and post the request wrapper with the login object inside

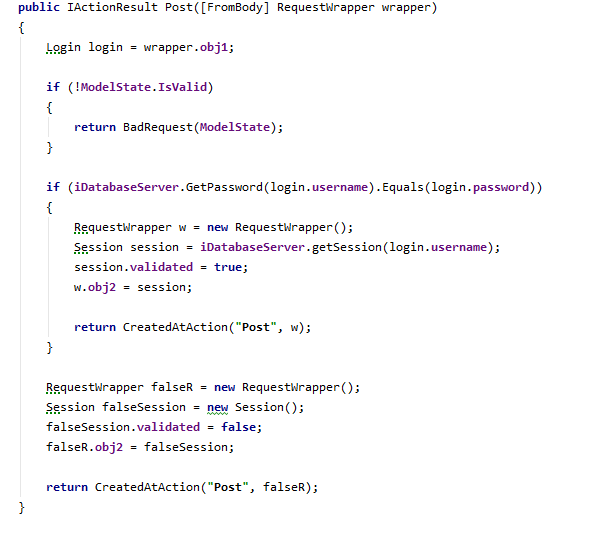
The method will return a Request Wrapper object with a Session object stored inside

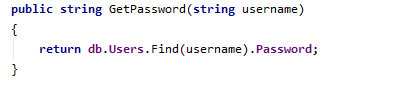
And return it.

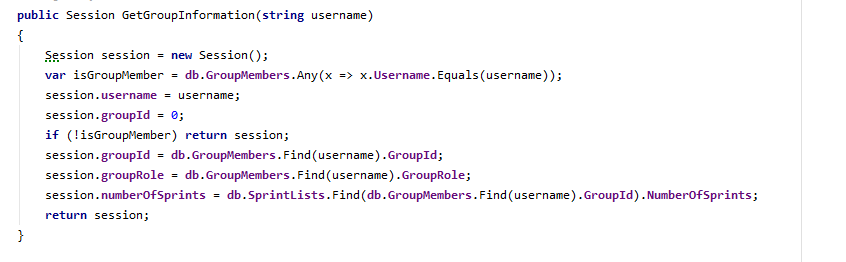


In the Login Controller at tier 3 It will call a method in the group repository getPassword()

If the password matches with the login password is will call another method get session and set validated to true and return it to the calling method else, it will return a Session object with the validated field = false.





In Group Repository it will make a query in the data base and return the Session object.

The object returned will go back until the Login view model so it can open next window or throw a error message.

# Test

The tests performed on the system were Junit testing on the model and scenario testing for wider functionality. The system has passed all the tests given to it.

## Test Specifications

The tests have been performed in Java 8 using Junit 4.

# Results and Discussion

The outcome of the project can be considered positive as the core parts of the system have been implemented. The system is still partly functional, but it can be made functional in a short period of time and it can be delivered for real customers.

In Appendix D, the manual of the system can be found.

# Conclusions

The project scope has been to assist students in their SEP projects where they are required to use SCRUM framework and UP models.

The first step has been to make the analysis on the background of the problem. In this case we looked also beyond the user needs, as we looked also in what the system should perform. In this way, the analysis part has covered the formulation of requirements, user case diagrams, use case descriptions, activity diagrams, system sequence diagram and test scenarios. In the analysis, it has also been included the analysis of the security to understand to what kind of dangers the system can be exposed to.

Second step has been to proceed forward, has been the design phase where the classes diagrams has been distinguished and the relationships between the classes have been classified. As this is a distributed system, attention has been given to what kind of responsibilities each class should have according to the tier level that it exists.

Secondly, it has been given attention to what kind of communication protocols to have in place between the tiers so we can ensure that the communication is efficient and reliable. At the end of this process, the security design has been discussed with its final design.

Having the Design phase completed, it has been proceeded with the implementation. The system has been implemented according to its design which covered all the aspects.

Final action before the delivery of the system has been the testing which proves the success of the analysis, design and implementation. Testing has ensured that the system is performing according to the requirements.

# Project future

Teams reflection on the current of the and it can be said that there could have been more functionalities added if the team would had avoided working on the load and working more on specifics, like:

* Communication between team members
* Communication between group and supervisor
* Communication between groups and supervisor
* Encryption or more security implemented as the system lacks it
* The chance to export all documentation from system into files

# Sources of information

Banger, D., 2014. A Basic Non-Functional Requirements Checklist « Thoughts from the Systems front line.... Available at: https://dalbanger.wordpress.com/2014/01/08/a-basic-non-functional-requirements-checklist/ [Accessed January 31, 2017].

Business Analyst Learnings, 2013. MoSCoW : Requirements Prioritization Technique — Business Analyst Learnings. , pp.1–5. Available at: https://businessanalystlearnings.com/ba-techniques/2013/3/5/moscow-technique-requirements-prioritization [Accessed January 31, 2017].

Dawson, C.W., 2009. *Projects in Computing and Information Systems*, Available at: http://www.sentimentaltoday.net/National\_Academy\_Press/0321263553.Addison.Wesley.Publishing.Company.Projects.in.Computing.and.Information.Systems.A.Students.Guide.Jun.2005.pdf.

Gamma, E. et al., 2002. *Design Patterns – Elements of Reusable Object-Oriented Software*, Available at: http://books.google.com/books?id=JPOaP7cyk6wC&pg=PA78&dq=intitle:Design+Patterns+Elements+of+Reusable+Object+Oriented+Software&hl=&cd=3&source=gbs\_api%5Cnpapers2://publication/uuid/944613AA-7124-44A4-B86F-C7B2123344F3.

IEEE Computer Society, 2008. *IEEE Std 829-2008, IEEE Standard for Software and System Test Documentation*,

Larman, C., 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*,

Mendeley.com, 2016. Homepage | Mendeley. Available at: https://www.mendeley.com/ [Accessed February 2, 2017].

YourCoach, S.M.A.R.T. goal setting | SMART | Coaching tools | YourCoach Gent. Available at: http://www.yourcoach.be/en/coaching-tools/smart-goal-setting.php [Accessed August 19, 2017].

MSP Guide. 2012. Tuckman model. [ONLINE] Available at: <http://www.mspguide.org/tool/tuckman-forming-norming-storming-performing>. [Accessed 2 June 2019].

E-Stimate. 2015. E-Stimate model. [ONLINE] Available at: <https://www.e-stimate.com/wp-content/uploads/sites/16/Doris-Muster-.pdf>. [Accessed 3 June 2019].

Scrum Guide. 2018. Scrum Guide. [ONLINE] Available at: <https://www.scrumguides.org/scrum-guide.html>. [Accessed 4 June 2019].

Conolly, T. and Begg, C., 2010. *Database Systems*. 5th ed. US: Pearson Education.

Larman, C., 2004. *Applying UML And Patterns*. 3rd ed. US: Pearson Education.

jrebel.com. 2014. Object-oriented design principles and the 5 ways of creating SOLID applications. [ONLINE] Available at: <https://jrebel.com/rebellabs/object-oriented-design-principles-and-the-5-ways-of-creating-solid-applications/>. [Accessed 15 May 2019].

docs.oracle. 2017. Lesson: All About Sockets. [ONLINE] Available at: <https://docs.oracle.com/javase/tutorial/networking/sockets/index.html>. [Accessed 15 May 2019].

https://www.ibm.com. 2006. Traceability from Use Cases to Test Cases. [ONLINE] Available at: <https://www.ibm.com/developerworks/rational/library/04/r-3217/index.html>. [Accessed 20 May 2019].

docs.oracle. 2019. Guarded Blocks. [ONLINE] Available at: <https://docs.oracle.com/javase/tutorial/essential/concurrency/guardmeth.html>. [Accessed 13 May 2019].

sourcemaking.com. 2019. Observer Design Pattern. [ONLINE] Available at: <https://sourcemaking.com/design_patterns/observer%E2%80%8B%E2%80%8B>. [Accessed 13 May 2019].

via.itslearning.com. 2019. The Observer Design Pattern (SVA, February 2019).pdf. [ONLINE] Available at: <https://via.itslearning.com/LearningToolElement/ViewLearningToolElement.aspx?LearningToolElementId=364658>. [Accessed 13 May 2019].

en.wikipedia.org. 2019. State pattern. [ONLINE] Available at: <https://en.wikipedia.org/wiki/State_pattern>. [Accessed 13 May 2019].

sourcemaking.com. 2019. Visitor Design Pattern. [ONLINE] Available at: <https://sourcemaking.com/design_patterns/visitor>. [Accessed 13 May 2019].

geeksforgeeks.org. 2019. Java Singleton Design Pattern Practices with Examples. [ONLINE] Available at: <https://www.geeksforgeeks.org/java-singleton-design-pattern-practices-examples/>. [Accessed 13 May 2019].

# Appendices

The purpose of your appendices is to provide extra information to the expert reader. List the appendices in order of mention.

* Appendix A – Project Description
* Appendix B – Analysis documents
* Appendix C – Design
* Appendix D – Manual

**Appendix A Project Description**

Insert the original Project Description here