SEP2X

Private Chat Client

VIA UNIVERSITY COLLEGE

Course: ICT Engineering

Created by:

Maja Petrusic, Student ID: 253899

Dimitar Ibushev, Student ID: 254056

Supervisors:

Mona Wendel Andersen

Troels Mortensen

Date:

25 August 2017

.

Table of contents

Project Description

1. Background Description

A while ago we had set ourselves a task to create a private chat room application and present it in an appropriate manner. Unfortunately, because of our disorganized work and lack of motivation, the presentation was lacking and unable to convince anyone that the team working on the creation of the program was even knowledgeable about the end product. However, we have not given up and have worked to understand and improve the program and especially on better presenting its usefulness.

The aim and the inspiration behind the application have remained mostly the same – it's inspired by the need for efficient and easy communication every company has when setting a task for its many employees. It also aims to close off outside influence from the internet it uses to bridge the gap that is distance between people using the program, making it much safer and with fewer things to distract the people from focusing on the task at hand.

This closed chat client allows, for example, employees of a company to communicate with each other freely, without being distracted by unnecessary data the common social media is overflowing with. It is also far more secure, as its administrators are the only ones who can create and remove user accounts, or to say it in a more practical manner – allow anybody to use it and be part of it.

All this ensures that the employees of the company that would be using this chat room would have an easy and secure way to communicate with each other – sharing ideas and keeping themselves informed.

2. Purpose

The project aims to give the end user the ability to securely send and receive messages through a private and closed platform, controlled by users designated as administrators.

3. Problem Formulation

Important problems that need attention and solving:

- Establishing a secure connection with the database
- Simplicity and reliability

- Ensuring the system is adaptable
- Restricting users without the required clearance

4. Deliminations

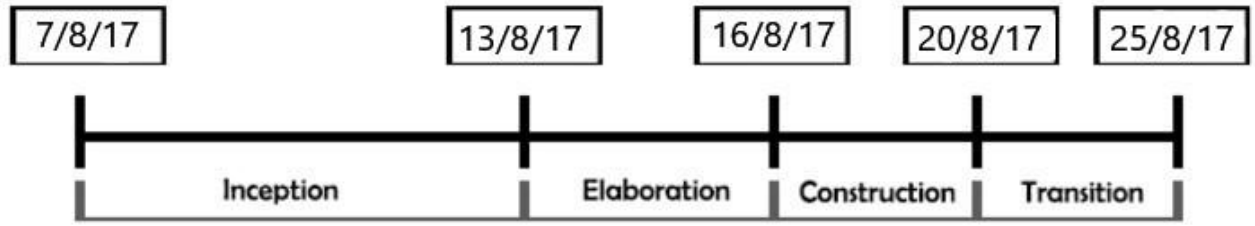
- Simplified GUI (Partially intentional as to keep things efficient)
- Communication is limited to only messaging and sending simple objects
- Security limited to only password and username

5. Choice of Model and Method

What	Why	Which
What are the most important aspects to a good performance?	Stability	GUI interface and database structure and server connection
How to make a secure connection with the database?	Main feature of the support system.	Use case modeling Requirements for database storage. Server Structure connection
How to prevent breaches in the secure connection?	Security	UML class modelling. Client/server architecture. Design patterns.
How to make the system adaptable to changes within?	For further upgrades	Use Case modelling and class diagrams for admin module or similar. Design patterns. Database design.

6.Time schedule

The time scope of the project is estimated at 3 sprints, 5 days each. Because of the small group and less time than normal, the time per day might vary but the minimum is set to 6 hours. We are still actively trying to utilize the SCRUM framework to control the project. The period for the project is between 7th and 25th of August.



7. References:

John Lewis, J. C., 2013. *Java Software Structures.* 4th ed. s.l.:Pearson.

Project Report

Private Chat Client

VIA UNIVERSITY COLLEGE

Course: ICT Engineering

Created by:

Maja Petrusic, Student ID: 253899

Dimitar Ibushev, Student ID: 254056

Supervisors:

Mona Wendel Andersen

Troels Mortensen

Date:

25 August 2017

1.Abstract

The purpose of this report is to inform the reader about the development phases of the re-designing of the Private Chat Client. The whole process took three week to accomplish. We started off with deciding what we like from the previous project and what we need to heavily alter in order to meet the necessary standard. From there we knew what needed most of our attention. First off we heavily modified the program's connection to the database, allowing us to expand in its functionalities and remove the limitations it faced before. After making sure the new things we implemented were thoroughly tested, we moved on to the next thing that needed attention. We revisited the socket connection the program uses and make sure all the new functions are properly implemented and available to the users, as well as correcting small issues with the already established features. Luckily for us, our GUI was stable enough as it is, so we did not need to modify any of it, but we made sure we go through it once again to minimize the risk of anything faulty and to test if those features we added indeed work as intended with it. While we are content with the functionality this program currently offers, it should perhaps be noted that the simplicity and efficiency of it allows for more to potentially be added without too much hassle.

2. Introduction

Private Chat Client is a Send and Receive system for instant messaging. Our goal was to make a system with a server that can handle multiple connections. The first main goal was to make a detailed yet simple and easy to use interface that would allow a client/user to log in, allow him to review his past messages to other users and save them so other users can view their past conversations.

One of our main requirements for such a system is to get a server with a database that can store and allow others to check information about the past and present users and messages. We used Java as our main tool to accomplish this task..

We anticipated difficulties along the way and made a list of possible ones. The biggest difficulty was connecting the database and the chat client(s) through a socket connection. It was to make sure the messages were instantaneous and the screens of our users updated correctly. The program also needed to be eye pleasing with a user friendly interface that was easy to use and intuitive of its purpose.

3. Requirements

List of requirements:

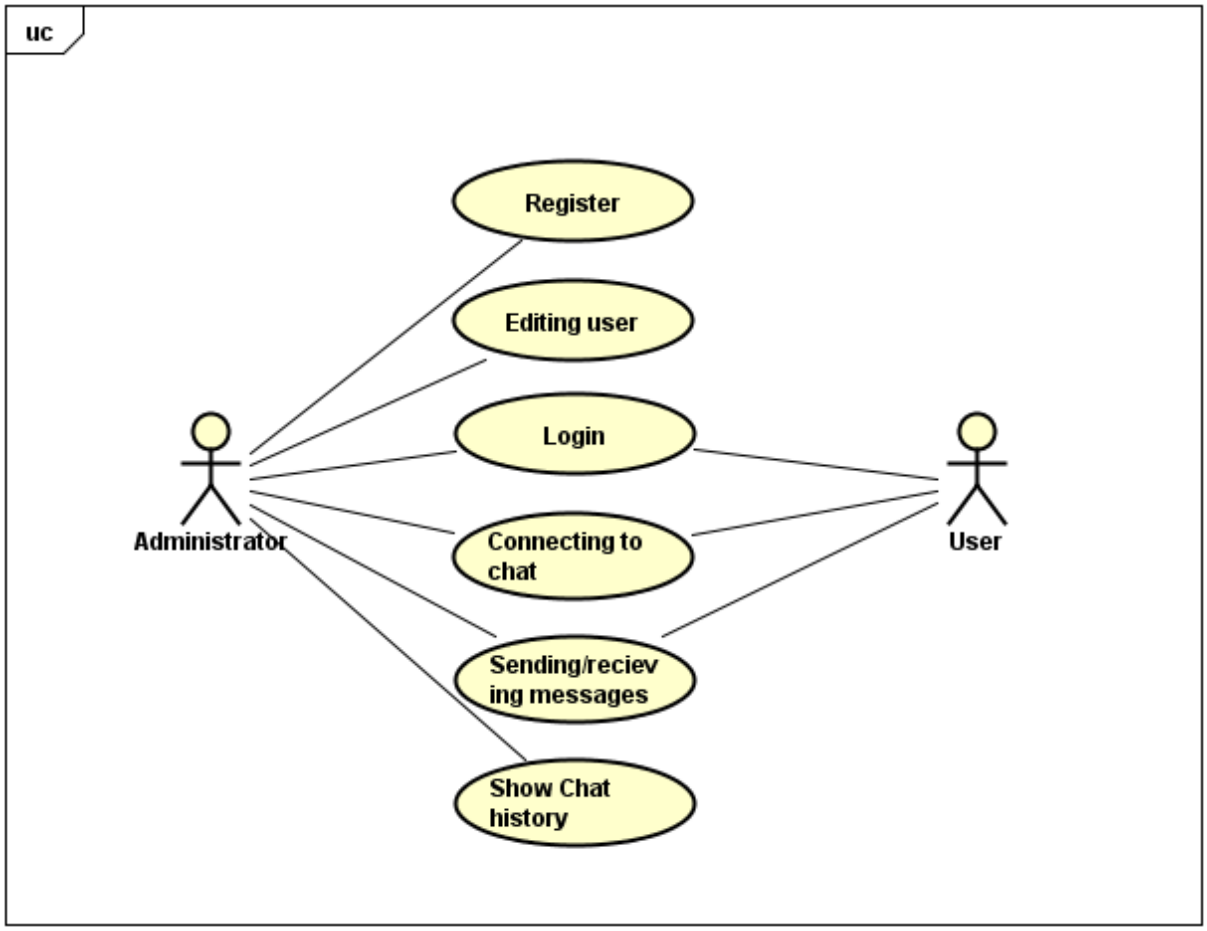
FUNCTIONAL:

1. Client-Server system: the user and administrator must be able to connect to the chatroom
2. Sending/Recieving messages: the user and administrator must be able to send messages into the system and mesages should be received by others
3. Graphical interface: the program must have grafical a interface
4. Create an account: an administrator must be able to create an account
5. An administrator must be able to edit users
6. An administrator must be able to see chat history

NON-FUNCTIONAL:

1. Instant messaging system
2. System must be written in Java
3. An administrator must be able to see chat history
4. User or Administrator has to be able to see who else is present in a chatroom

4. Use-Case Diagram



Register - Only an administrator is able to create a new accaunt for user

Editing user - Only anadministrator is able to modify any other user's information.

Login - A user or an administrator is able to log in into the server using his username and password

Connecting to chat – A user is able to join a chatroom and is able to send IM's (instant messages) to other users.

Sending/receiving messages – Users and administrators are able to send messages to other users in the system and also receiving messages.

Show Chat History - Only an administrator is able to see chat logs and history of other users

5.Use Case Description

-Desription for: “ Sending/receiving messages” use case.

UseCase :	Sending/receiving messages
Summary:	The users and administrators are able to send messages to other users and administrators as well as receiving messages
Person:	User/Administrator
Precondition:	There are users and administrators connected to the server
Postcondition:	The other users and administrators receive the user’s or administrator’s messages
Base Sequence:	1. Type a message. 2. Server receives the message and transfers to the other users and administrators. 3. The user/Administrator receives the message.
Exception Sequence:	Server crashes: 1. Type a message. 2. User/Administrator receives error message.

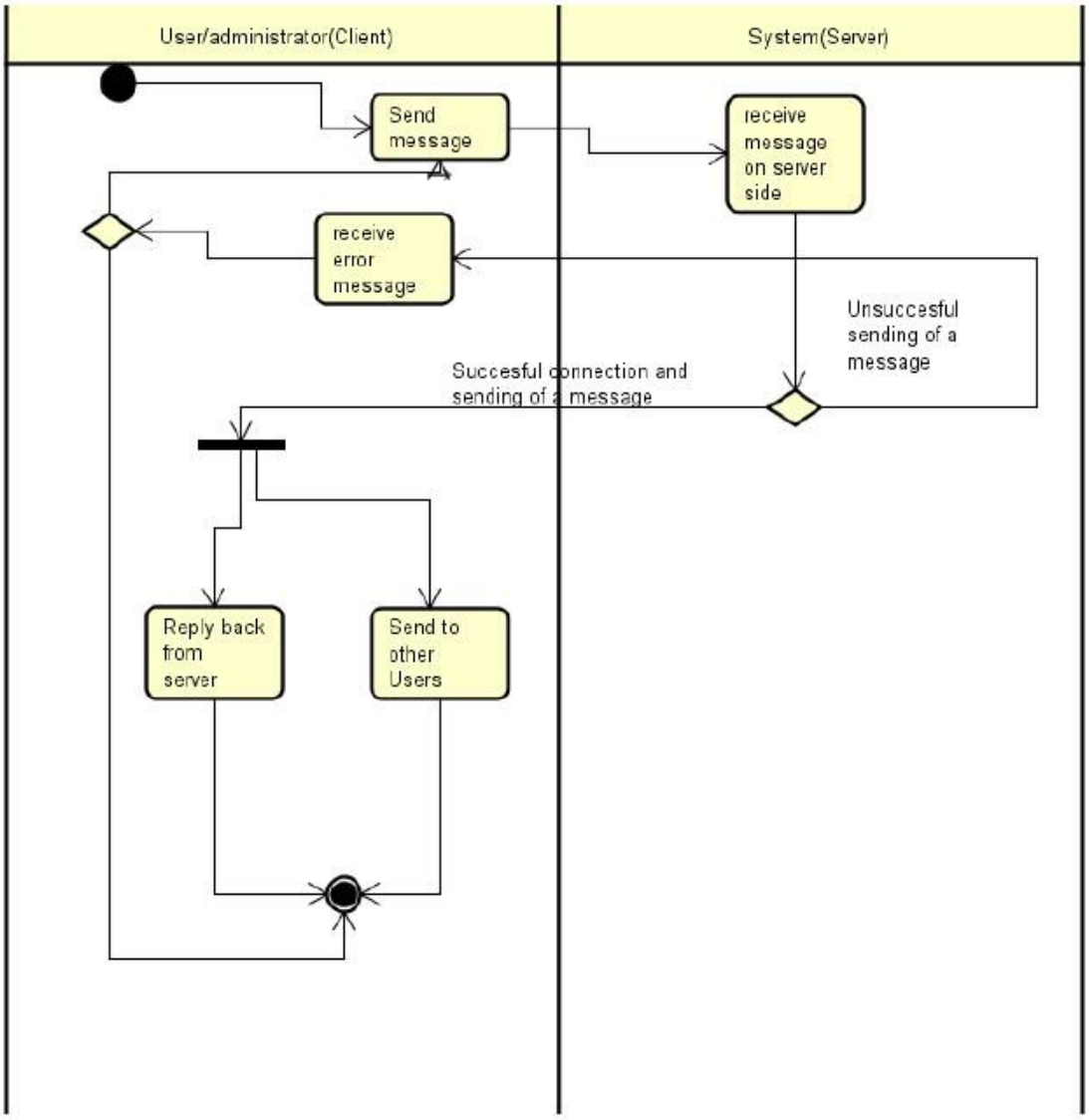
Desription for: “Connecting to chat” use case.

UseCase :	Connecting to chat
Summary:	The users and administrators are able to connect to the chat system
Person:	User and Administrator
Precondition:	Running client-server connection
Postcondition:	User/Administrator logged to the system
Base Sequence:	1. User/Administrator tries to connect with his account and password 2. Server receives the request

	3. loggin is successfull if the account and password are correct
Exception Sequence:	Connection user/administrator and server fails: 1. User/Administrator recieves an error and has to try to connect again. 2. If it is successful User/Administrator are connected

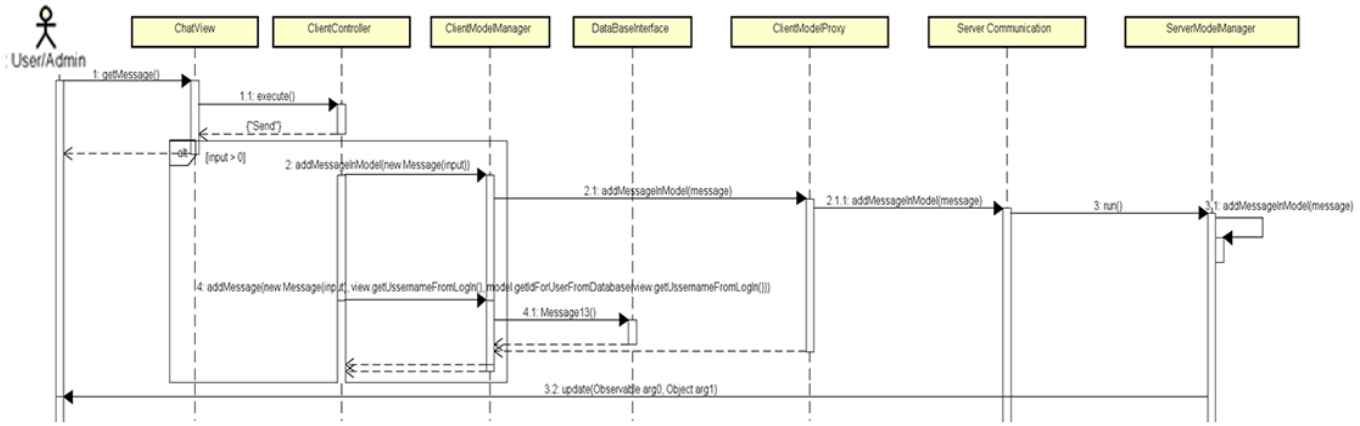
6. Activity Diagram

The activity diagram below shows detailed information regarding the possible actions of a user or administrator when sending a message

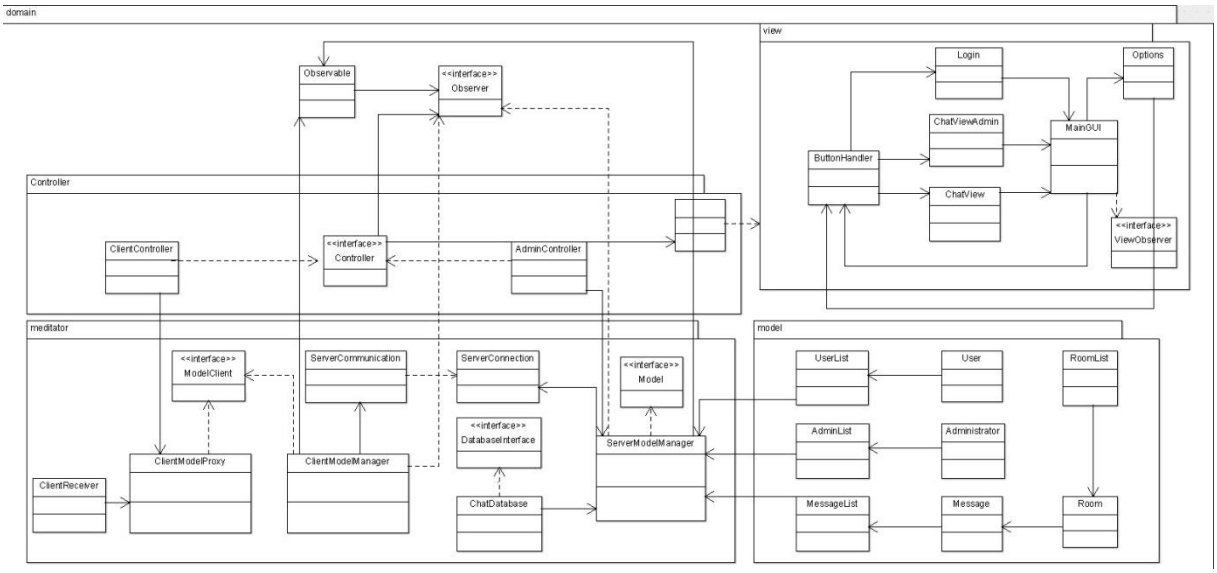


The process is simple – a message is transmitted client-side, which is in turn received by the server. If the server receives the message and successfully connects to the client it gives out a reply and broadcasts the message for all the users to see. If the connection is unsuccessful the user receives an error and has the option to send a new message.

7. Sequence Diagram



8.Class Diagram



This is the complete class diagram of the program. It includes Model, Viewer, Controller and Meditator segments. The first three will be explained in the respective design pattern explanation, while the last one (Meditator) is a design pattern on its own, serving the purpose of facilitating the connection between the database and the client. It contains two

nearly identical socket connection classes for the purpose of distinguishing between administrator and normal user.

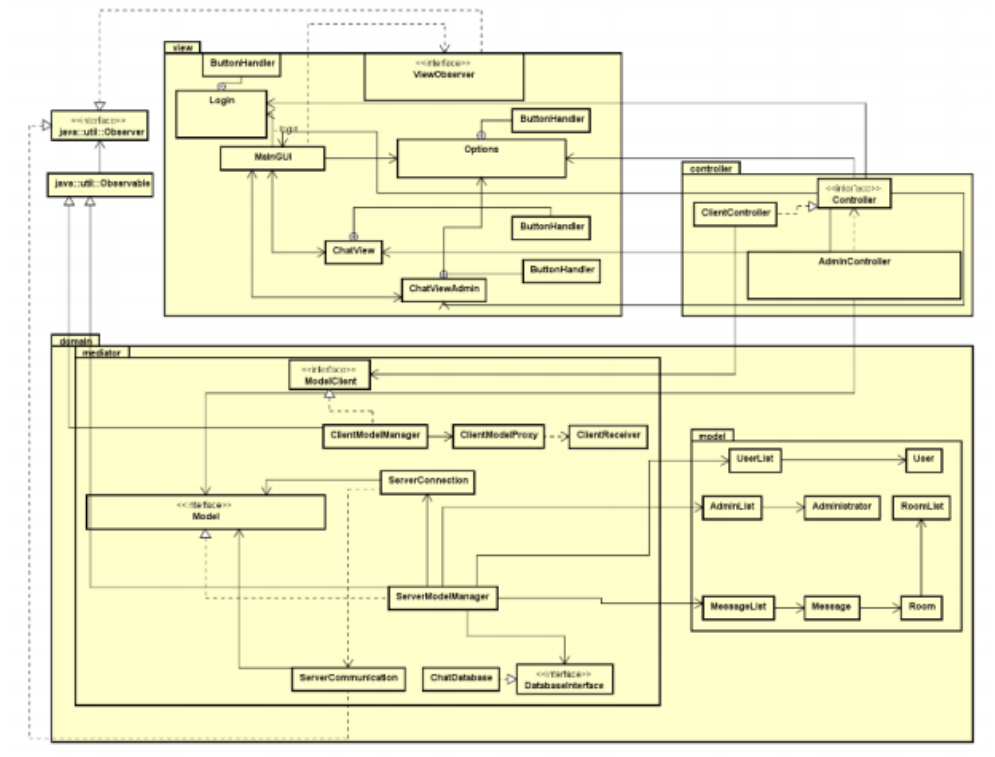
9. Model Viewer Controler

The system was built around MVC (Model-Viewer-Controller) execution pattern.

The program is based on a standard model package used for storing and processing. A Client and Admin Controller, acting as a model and view, handle the data stream towards the objects and updating the view when there are changes. Access to the model is achieved through the ClientModelManger and ServerModelManager classes.

For updating the information shown on the screens that the user sees an Observer pattern was added. Both ClientModelManger and ServerModelManager use it.

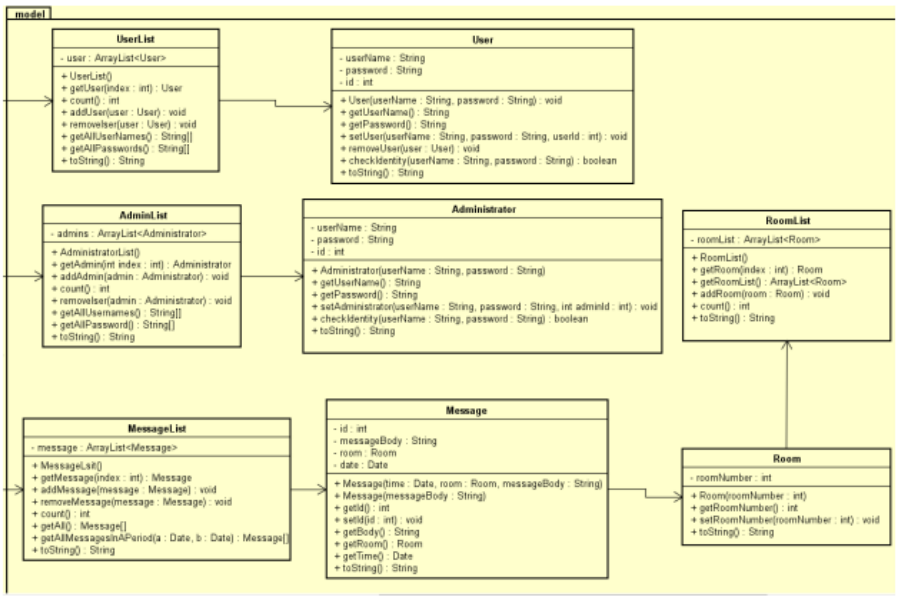
The relation between ServerModelManager and the databases interface is a vital part of the database, allowing communication between the two via the use of SQL statements.



.

10. Class Diagram for Model

The class diagram shows the model of the MVC pattern used. It includes the object classes Message, Administrator, Room, User alongside the list classes that contain the appropriate methods for their management, creating, deleting etc.



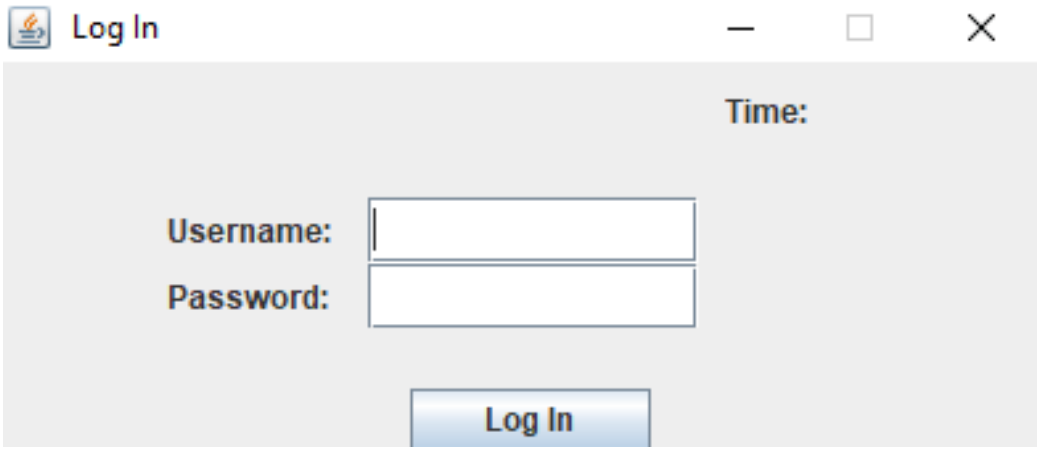
11. Design

User Interface

The following pages portray the user-friendly graphical user interface

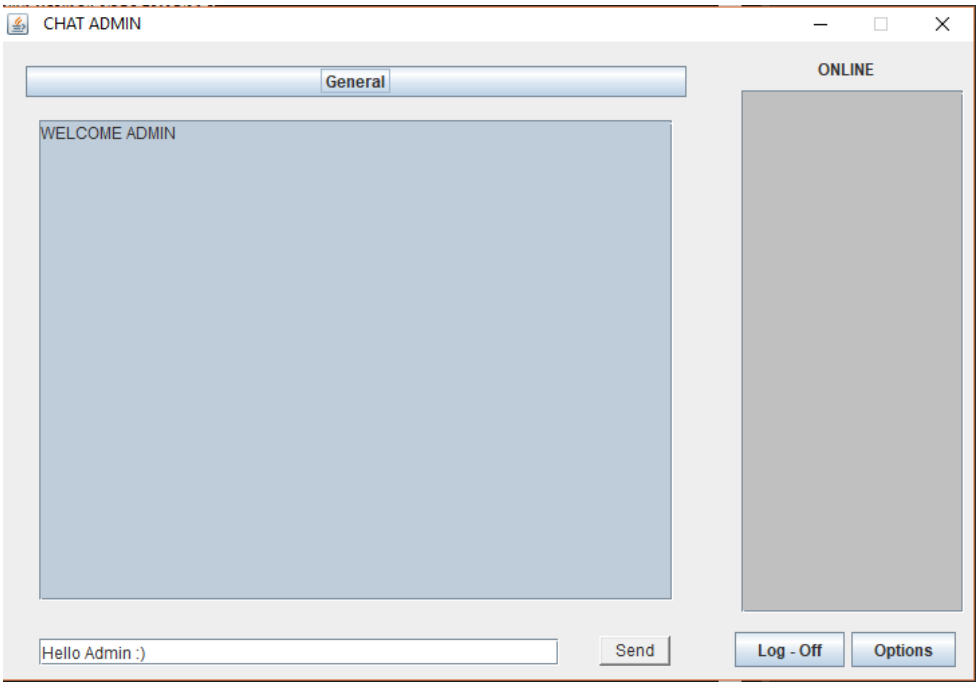
Our program is GUI based and it has a simple design which is very intuitive to use.

This is the login screen. It allows the user or administrator to log in the program



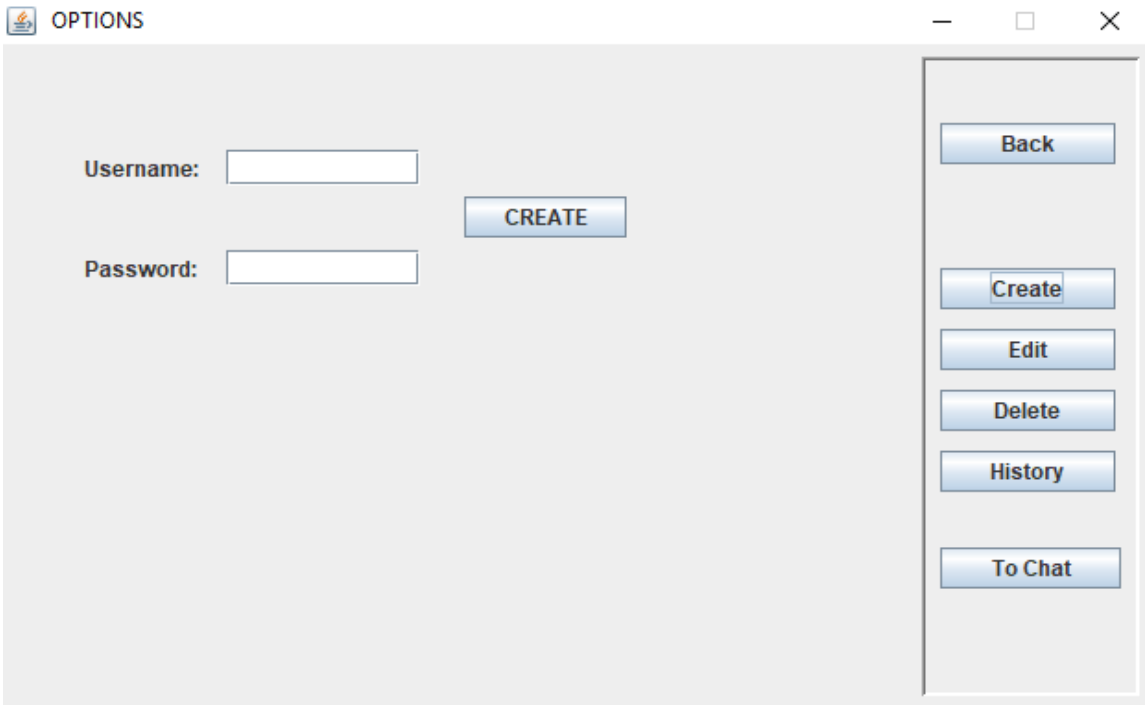
When starting the program you have to enter your Username and Password and if you want to login press the button Log in

Admin



This is the panel for administrator.

The text bar functions which allows the admin to send text messages to other people in the chat system.The Log-Off button works and it returns the admin to the login screen.The Option button opens option exclusive to admins only.

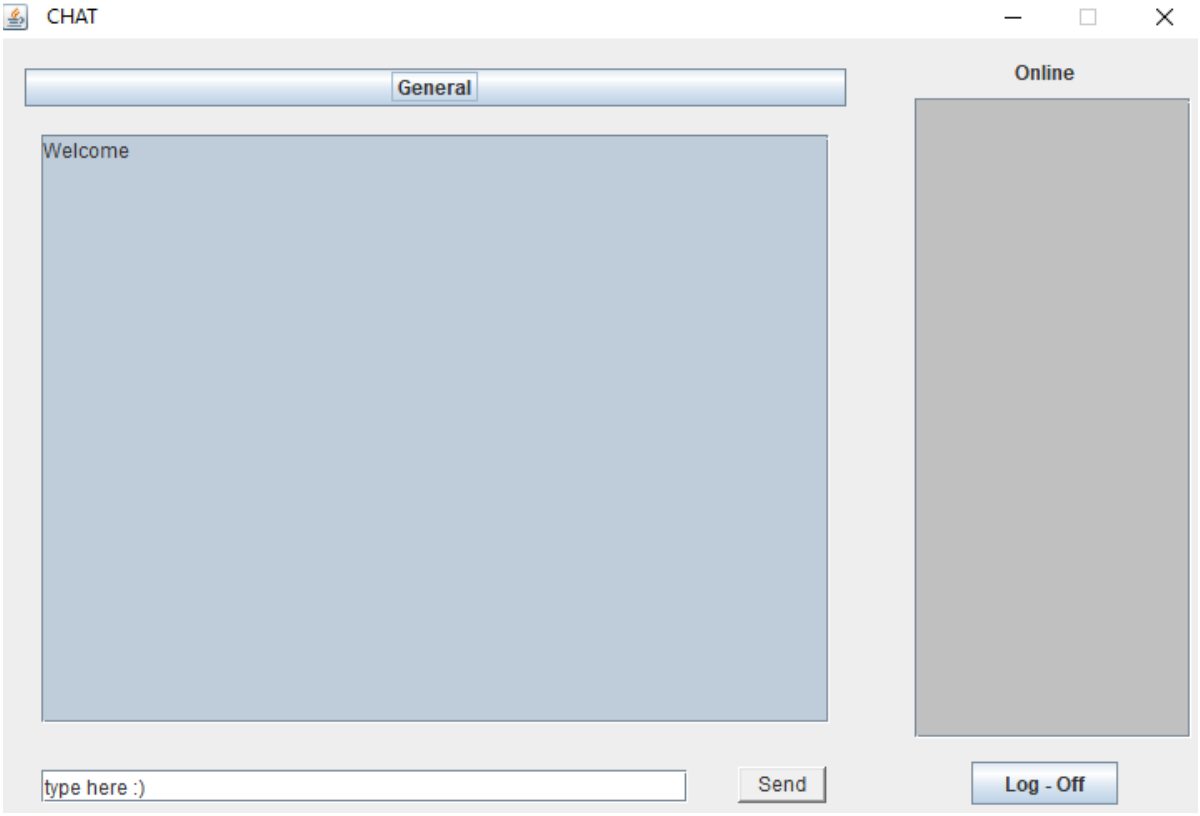


This is the Option's menu which is available to administrators ***only***

It allows creating, editing and deleting User accounts from the system. Also there is a **To chat**

button which sends the administrator back to the chat screen. The history button is not functional but if it was it would show the history of the chat room.

User



This is our User panel.It has a functional text bar which allows typing and a functional send button which allows the user to send messages to the other people in the chat.Also there is the log-off button which logs the user out of the char.

12. Implementation

The MainGUI class controls the display of GUI.

```
1  package view;
2
3  import java.util.ArrayList;
7
8  public class MainGUI implements ViewObserver {
9
10     private static Controller controller;
11     private LogIn logIn;
12     private ChatView chatView;
13     private ChatViewAdmin chatViewAdmin;
14     private Options option;
15
16     public MainGUI() {
17         this.controller = controller;
18     }
19
20     public int activeGUI() {
21         if (chatView == null)
22             return -1;
23         if (chatView.isActive()) {
24             return 1;
25         } else if (chatViewAdmin.isActive())
26             return 2;
27         return -1;
28     }
29
30     @Override
31     public void update(Observable o, Object arg) {
32
33         if (activeGUI() == 1) {
34             System.out.println("userss");
35             chatView.show(arg);
36             chatViewAdmin.show(arg);
37         }
38         if (activeGUI() == 2) {
39             System.out.println("adminn");
40             chatViewAdmin.show(arg);
41         }
42
43     }
44
45     public void LogInUser() {
46
47         logIn.dispose();// To close the current window
48         chatView.setVisible(true);// Open the new window
49     }
50
51     public void LogInAdmin() {
52         logIn.dispose();// To close the current window
53         chatViewAdmin.setVisible(true);// Open the new windows
54     }
55
```

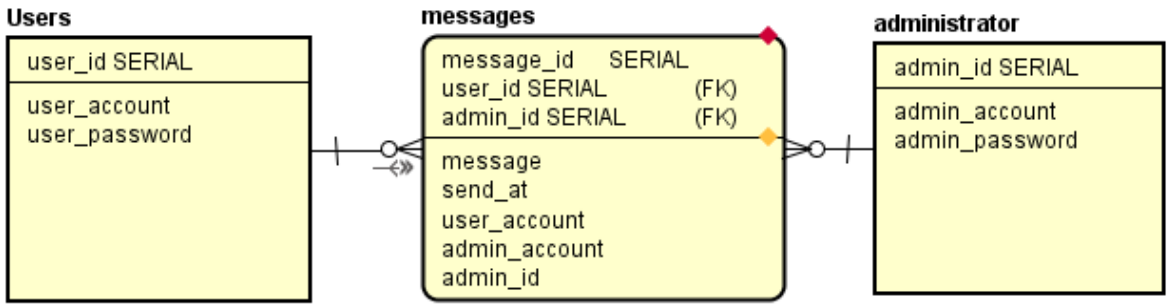
```
    }

    public String getMessage() {
        return chatView.getMessage();
    }

    public String getMessageAdmin() {
        return chatViewAdmin.getMessage();
    }

    public String getUssernameFromLogIn() {
        return logIn.getUssername();
    }

    public String getPasswordFromLogIn() {
        return logIn.getPassword();
    }

    public void start(Controller controller) {
        logIn = new LogIn(controller);
        chatView = new ChatView(controller);
        chatViewAdmin = new ChatViewAdmin(controller);
        option = new Options(controller);
    }

    public void LogOut() {

        if (chatView == null ||chatViewAdmin.isActive()) {
            chatViewAdmin.LogOut();
        }
        if (chatView.isActive()) {
            chatView.LogOut();
        }

        logIn.setVisible(true);
    }

    public String getNewUssernameCreate() {
        return option.getNewUssername();
    }

    public String getNewPasswordCreate() {
        return option.getNewPassword();
    }

}
```

The Update method utilises two external methods from GUI's with the goal of displaying the appropriate information onto the active panel :

```
    public void show(Object arg1)
      {
         if (arg1 == null || arg1.toString().length() < 1)
         {
            return;
         }
         String old = messlist.getText();
         if (old.length() > 1)
         {
            old += "\n";
            messlist.setText(old + arg1);
         }

      }



   public String getMessage() {
           if (text == null || text.toString().length() < 1)
                 {
                    return null;
                 }
              return text.getText().trim();
           }


   public boolean isActive()
   {
      java.awt.Frame win[] = java.awt.Frame.getFrames();
      boolean isOpen = false;
      for (int i = 0; i < win.length; i++)
      {
         if (win[i].getName().equals("  CHAT ADMIN  "))
         {
            isOpen = true;
         }
      }
      return isOpen;
   }
public void LogOut()
```

13.Database

The Database is as seen below



Outside of message_id, messages use a foreign key from user and administrator. For the purposes of the project this simple arrangement proves sufficient. We limit the rights of uses by not giving them access to the forms inside the client.

14. Conclusion

We started out with an already established project, that however needed a lot of improving. Our excellent teamwork ensured that we successfully fielded code refinements, fixing errors and adding missing features, while preserving the simplicity the GUI offers and the adaptability of the project.

We also completely re-wrote the documentation in order to keep up with the application and ensure it has the needed quality.

We are very content with how things unfolded and with the end product we have. It certainly shows improvement in our teamwork since the last project and it also provided experience on how to work in a focused group.

Process Report

Private Chat Client

VIA UNIVERSITY COLLEGE

Course: ICT Engineering

Created by:

Maja Petrusic, Student ID: 253899

Dimitar Ibushev, Student ID: 254056

Supervisors:

Mona Wendel Andersen

Troels Mortensen

Date:

25 August 2017

1.Project plan

Our main objective was creating a private chat room. It was to be based on a clienet/server architecture with features such as:

- Sending and receiving messages between users
- Logging in and out of the system
- Presence of administrator with the ability to create, modify and delete users

2.Group policy:

Roles:

- Dimitar: Scrum role: SCRUM master and developer.
  Belbin role: Plant (Specialist, Complete Finisher)
- Maja: Scrum role: Product owner and developer
  Belbin role: Shaper (Monitor Evaluator, Resource Investigator)

Contract:

In our last project we skipped the group contract and that affected us negatively. This time we knew better and make sure we are both clear on our individual and team expectations and priorities. We were, of course, to follow SCRUM's theory to ensure our project is finished on time and delivers a functional product. We also agreed on the following points:

- Meeting at least 5 times each week
- Notifying each other if we can't make it to a meeting
- We were to both upload everything or work directly on One Drive to make sure we followed the same material
- Working together as a team would be a must
- We'd try our best to help each other

Consequences:

We did not have to set any practical consequences as we both remembered how and why we failed our last project. We certainly didn't want to repeat.

All rules were of course made with agreement from both and were as such object to change in case of same approval.

3.SCRUM

SCRUM roles assessment

Note: Because our group consists of only 2 people, the SCRUM model had to be adapted and slightly compromised to fit in

SCRUM Master - Dimitar Ibushev

As a SCRUM master, I was making sure we are sticking to to following the plans and the time schedules, as well as scheduling meetings and setting up deadlines.

Due to only two people working on the project, the role of meditator the master usually has was not much applicable for this project.

I did however wrote big parts of the project report - such as use-case, activity etc diagrams, documenting the implementations, putting out examples, making conclusions and so forth.

Product Owner - Maja Petrusic

The Product Owner represents users, customers and others in the process. My role was to designate changes to be made on the code and to assert the goals we need to meet to maximize the value of the product.

Because of the size of the group, the application of my role was limited as we both more or less knew what needed doing and we worked on it together. As such, there was not much need of pointing stuff out, other than suggesting what to improve and what new documentation to write.

Of course, I took my own part in writing said documentation, mostly engaging in project description and process report.

Product Developers - Maja and Dimitar

Because of the team size, we had to be both leaders and developers in this project. We worked together following the pair programming pattern, since it's most efficient for a small team.

4. Product backlog

Product Backlog is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product. The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering.

Product backlog contains the items that the Project owner would like his project to contain and also estimated hours for each of item from list.

Number	Product backlog	Estimated Hours	Actual Hours	Status
1.	As a user I would like to have multiple clients connected to one server	6	10	Done
2.	As a user I want to be able to send a message to other Users	5	5	Done
3.	As a user I want to be able to receive messages from other users	5	5	Done

4.	As a user I would like to be able to connect to the chat	5	4	Done
5.	As a user I would like to have user friendly interface	6	4	Done
5.	As a administrator I would like to be able to access database with all information about users	3	4	Done
6.	As a administrator I would like to be able to access database about chat history	3	2	Not Implemented
7.	As a product owner I would like to have project and processs report	20	18	Done

5.Sprint retrospective and daily log

Sprint 1

Sprint Summary: A sprint dedicated to reflecting upon what we have at hand and what we intend to change. Also the sprint in which we made the general schedule and group contract.

Meetings:

-7th of August:

- Attending the presentation

- Confirming group
- Deciding on goal - refining our previous project

-8th of August:

- "What we did wrong last time" discussion
- Planning ahead - designating sprints and their purpose

-9th of August:

- Identified first code segment in need of change (database connectivity)

-10th of August:

- Identified second code segment in need of change (Client/Server manager)

-11th of August:

- Wrap-up meeting
- Evaluation and preparation for next Sprint

Sprint reflection: little to no problems with this sprint. We easily agreed on plan of action, schedule and what exactly we'll be changing. We were also both happy with our assigned roles.

Sprint 2

Sprint Summary: A sprint dedicated to fixing and expanding the code we have, as well as thorough testing.

14th of August:

- Work on first code segment
- Keeping a log on changes

15th of August:

- Testing of the first code segment
- Corrections and refinements

16th of August:

- Work on the second code segment
- Keeping a log on changes

17th of August:

- Testing of the second code segment
- Corrections and refinements

18th of August:

- Minor code improvements
- Full code review
- Full code testing

Sprint reflection: this sprint had some difficulties, mostly stemming from problems with solving technical issues and understanding the code we had at hand. We ultimately managed to complete what we planned through extensive use of pair-programming.

Sprint 3

Sprint Summary:A sprint dedicated to documentation. Perhaps the most important one since this was lacking with the last project. Also a chance to look back one final time and

make sure everything works and is documented as intended.

21st of August:

- Writing a new project description
- Writing an introduction to project report

22nd of August:

- Tables and diagrams for project report
- Descriptions for said tables and diagrams

23rd of August:

- Putting together all the logs into a process report and giving it overall shape

24th of August:

- Reivew of documentation
- Small errors fixing

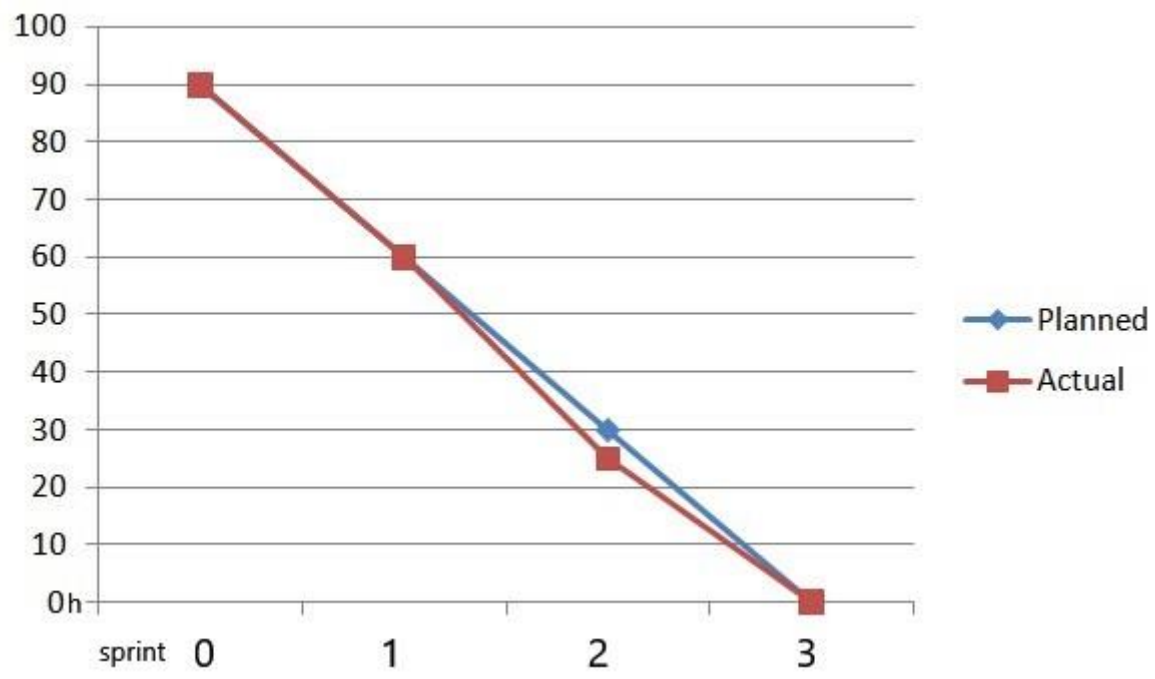
25th of August

- Final re-reading of everything before turning in

Sprint reflection - we had some disagreements and some lack of inspiration here and there regarding the documentation. We did however eventually come to an agreement and are satisfied with how things turned out and the quality of our work.

6.Burndown chart

The Burndown Chart displays the remaining effort for a given period of time. It shows an ideal work curve that will be needed for finishing project on time according to the estimations made by team and the curve which presents how many hours we actually worked. It is seeable that we spent roughly the same amount of time as we planned to spend.



=Burndown Chart for the entire project.

Sprint 2:

Backlog Item	Activity	Estimated Time/Time taken
Multiple clients	Threaded socket connection	4h/7h
	Keeping a log	1h/1h
	Testing	1h/2h
Sending message to users	Creating designated classes and methods	3h/3h

	Keeping a log	1h/1h
	Testing	1h/1h
Receiving message from users	Creating methods. Adding an Observer class and making classes Observable	3h/3h
	Keeping a log	1h/1h
	Testing	1h/1h
Connecting to chat	Setting up the meditator pannel and its classes	3h/2h
	Keeping a log	1h/1h
	Testing	1h/1h
User-friendly GUI	Creating GUI classes	2h/1h
	Creating methods	2h/2h
	Keeping a log	1h/1h
	Testing	1h/1h
Database accces for admins	SQL statements	0.5h/1h
	ServerModelManager methods	0.5h/1h
	Keepin a log	1h/1h
	Testing	1h/1h
Chat History	SQL statements	0.5h/1.5h
	ServerModelManager methods	0.5h/0h
	Keeping a log	1h/0.5h
	Testing	1h/0h

Sprint 3:

Backlog Item	Activity	Estimated time/time taken
Project Report	Writing core paragraphs	4h/4h
	Creating diagrams	2h/1.5h
	Writing diagram descriptions	2h/1.5h
	Taking screenshot/code sniplets for showcasing	1h/1h
	Gramatical changes and formatting	1h/1h
Process Report	Writing core paragraphs	4h/4h
	Creating charts and diagrams	1.5h/1h
	Writing chart and diagram descriptions	1.5h/1h
	Belbin and contract	2h/2h
	Grammatical changes and formatting	1h/1h

7.Team and self-assessment

Maja Petrusic

Team assessment:

I was personally not very happy with how our first project went. We were unable to really work as a team and there was an obvious lack of communication and I was afraid it would be the same case now. I was also uncertain if the small group could handle the task we had. However, it seems my concerns were unfounded as we managed to come together as a team and solve most if not all of the issues we had with our project and deliver a quality end product.

Personal assessment:

While we knew what our goals were as a team, it so happened that there were side things that prevented the proper focus on the task at hand. And whilst managing to finish our project on time without making many compromises shows we handled that superbly, I can't help but feel that perhaps it's possible to better plan for those somewhat unlucky cases. They are an inevitability and this project reminded me that. Still, I am content with how things went and my own performance.

Dimitar Ibushev

Team assesment:

I was left with a bitter taste in my mouth from the last semester examination. I felt left out and unfairly treated. Our team had little to no teamwork and very poor communication. I was determined to shrug that off and do better this time and luckily it was just what happened. While the group was small, which doesn't make anything easier, I am glad we managed to work together and succeed in our task.

Personal assessment:

Because of how small the group was this project was somewhat straining for me. Still, I actually enjoying seeing myself involved and making a difference, unlike the previous project

where I was forced to stand by and do nothing. There were times when I couldn't count on help from Maja and perhaps that is where I mostly feel I need improving. I felt like my hands were tied when in fact I could have done more to help the project. In that regard I definitely learned from this project and will make sure to respond better to such cases.