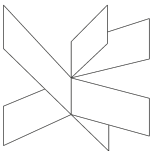
Bring ideas to life
VIA University College



PROCESS REPORT

Chat System

Ameya Mahankal (326157)

Bozhidar Manev (326391)

Joan Tammo(325753)

Radoslav Kiryazov(326155)

Zsolt Nóvé (326345)

Supervisors:

Henrik Kronborg Pedersen,

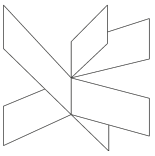
Jørn Martin Hajek

[19 171 Characters]

Software Technology

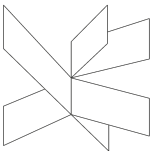
Semester:2

[11.12.2022]



Version: August, 2018
Template responsible: dans@via.dk

Table of content



1 Introduction

The documenting of the project's process has given the group a chance to trace the progress of our group's work from its formation until the handing of this document.

This report will showcase the methodology with which the work on the project was conducted, and how the group implemented the scrum framework during each of the unified proces's phases.

The report will discuss how each of the phases of the project were conducted and how these phases relate to the unified process frame of work.

At the end of the project execution section the report will discuss some identified risks, and how they were handled.

The report will also provide an insight into why some of the promised requirements were not implemented and what could have been done to prevent such shortcomings in future projects.

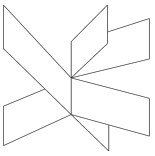
At the beginning of the project the team decided to create a chat system.

The group was working with the scrum method. In the meeting the team created a scrum group to plan sprints in order to set goals to achieve for progressing. In the first sprint the group members started to establish a client, server the connection and mvvm.

In the next sprint the group started the deeper documentation of the project.

To be able to follow up with the work done the group made a burndown chart to show our progress.

The group was under constant supervision by our teacher. The team wanted to make sure that we stay on the right track and work more efficiently.'



The following is a table of the product backlog documented in the log book:

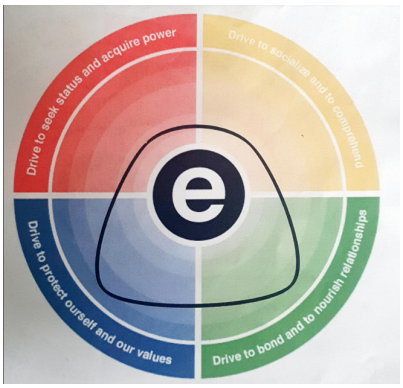
User Story	Time
As a user, I want to send a message to other users so that I can communicate with other users.	9
As a user, I want to have a personal profile, with which other users can connect.	3
As a user, I want to have an access to the history of my chats(logs), so I can read my previous conversations	3
As a user, I want to be able to manage my chats, so I can create or delete chats.	4
As a user, I want to create a group chat, so I can communicate with multiple users.	6
As a user, I want to edit my profile information.	4
As a user, I want to hide offensive and insensitive language.	3
As a user, I want to block belligerent users, so I can avoid communicating with them.	7
As a user, I want to select a theme for my chat to make it my own.	2
Download chat backup/chat logs	3

The table shows how the product owner documented the different user stories, which helped in developing the requirements for the project. The table also shows the different time estimations for the different user stories.

The table played an integral part during the duration of the project by being the basis for planning the different sprints predominantly during the establishment and conception phase of the project.

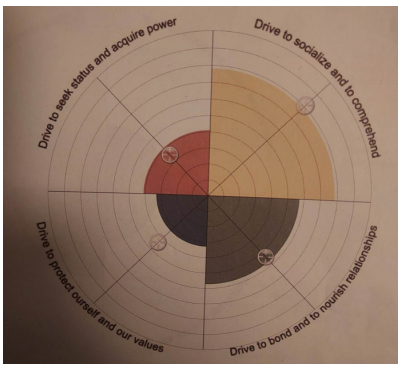
2 Group Description

The group is composed of a variety of nationalities consisting of two Bulgarians, a Hungarian, an Indian and a Syrian. The group is now considerably more experienced with the skills acquired from semester one.
The 5 members are:



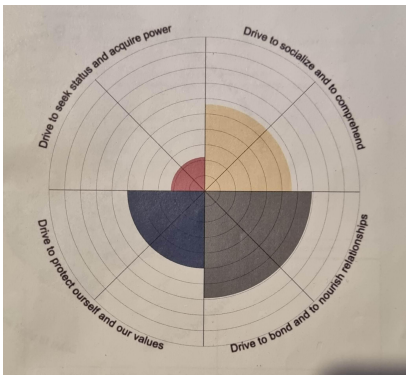
- **Ameya Sundeep Mahankal**: A student from India with basic experience in Python and MySQL.With the ongoing education, a wealth of knowledge has been gained from the past two semesters with a reinforcement of good practices.

- **Bozhidar Ganchev Manev**: A student from Bulgaria, who moved to Denmark to develop as a Software Engineer. He had a little experience in Java before his university education but after two semesters he has extended his knowledge and learned various good programming practices.

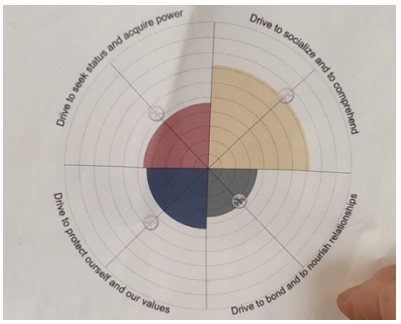


- **Joan Tammo:** A Kurdish refugee from Syria. His expertise with software in general before the start of his education is non-existent. After the first semester, he became more versed in the subject. Software engineering is his first experience with the development of software.

- **Radoslav Kostov Kiryazov**: A student from Bulgaria, who moved to Denmark to pursue education in the field of Software. Before that, he has little background in C#. Two semesters later he has gained valuable experience and is learning to implement proper programming etiquette.



- **Zsolt Nóvé**: A student from Hungary who lived in Denmark for 3 years with a small amount of knowledge coming from game programming(Unity). After studying software engineering he had found difficulties learning new things but gaining new experiences with different aspects of java and other programming languages that can be connected with it.



Cultural background:

The following tables are an overview of the 8 cultural dimensions.

The following table explains where each one of the different cultures, that compose the group, stand on the different aspects represented in the table:

🟡 - Syria 🟣 - India 🔴 - Hungary 🟢 - Bulgaria

1	2	3	4	5	6	7	8	9	10
Low-context			**Communication**						**High-context**
						🔴🟡	🟣🟢		
Direct Negative Feedback			**Evaluating**				**Indirect Negative Feedback**		
		🔴	🟢	🟡	🟣				
Egalitarian			**Leading**					**Hierarchical**	
			🟢			🔴		🟣	🟡
Consensual			**Deciding**					**Top Down**	
							🟣🟢	🔴	🟡
Task-based			**Trusting**					**Relationship-based**	
			🟢		🔴			🟣🟡	
Confrontational			**Disagreeing**					**Avoids confrontation**	
			🟡🔴	🟢		🟣			
Linear-time			**Scheduling**					**Flexible-time**	
		🔴	🟢				🟡	🟣	
Principle first			**Persuading**					**Application first**	
		🟣🟡	🟢🔴						

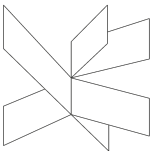
The following table explains where the members think the group stand on the different points presented in the table:

🔷The Group

1	2	3	4	5	6	7	8	9	10
Low-context			**Communication**				**High-context**		
							🔷		
Direct Negative Feedback			**Evaluating**				**Indirect Negative Feedback**		
						🔷			
Egalitarian			**Leading**				**Hierarchical**		
						🔷			
Consensual			**Deciding**				**Top Down**		
						🔷			
Task-based			**Trusting**				**Relationship-based**		
				🔷					
Confrontational			**Disagreeing**				**Avoids confrontation**		
								🔷	
Linear-time			**Scheduling**				**Flexible-time**		
						🔷			
Principle first			**Persuading**				**Application first**		
		🔷							

Even though the group comes from different cultures of varying degrees of each dimension. There were no conflicts as we had common stances on each of the eight parts of the culture map.
The group avoided conflicts by actively discussing every step of the project. When an argument arises, the group manages to resolve it by coming to a consensus.
The group did not escalate beyond stage 2 of Friedrich Glasl's conflict escalation model.

The fifth dimension is the most relevant in the group's case because of the way the tasks have been split between the members. The tasks were distributed based on the members' capabilities rather than based on personal relationships.
In the beginning, the group decided to establish a mutual understanding of the different cultures that the group is composed of, and avoid culturally sensitive subjects that



might lead to confrontations.

3 Project Initiation

With newfound knowledge gained in SWE, the group utilized the agile principle of the SCRUM framework as the primary method to work on the project.

The topic of a chat system was chosen as it seemed to cover all of the given project requirements and the group was keen on developing a system based on their preference.

The first phase of the project was group formation which was a straightforward process since the members decided to continue together with the same group from SEP-1 as there was mutual respect for each other and a good work dynamic.

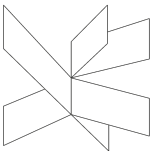
The group managed to split the project into small pieces that gradually connected to form the project.

The group started by constructing a small chat system that could only send and receive messages between all users connected to the system. Then additional features were added such as one-to-one and one-to-many communications, adding other users to chats and leaving chats, the option of chat customization, and updating profile information. By the end of the project period, the group found that the methodology with which the project was conducted was very helpful in that it allowed for an agile development process.

The division of the project into small parts is an application of the scrum method. Scrum is an application of the empirical method of research, which states that the process of development and research is appropriately achieved by researching the subject through means that guarantee transparency, provide an opportunity for inspection, and allow for adaptation.

The main focus of the scrum framework is that a project should start with the application of a small number of features, and gradually expand them. Scrum is divided into sprints, which are a specific period of time within which the group members add small components of the developing system which lasts until the system reaches a point where it is considered satisfactory by the customer.

The planning phase of the project went smoothly due to the use of Scrum, for it allowed for an agile practice of developing plans, and editing them, thus resulting in more successful plans.

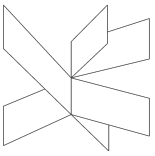


4 Project Description

During the project description phase, the group worked on the background description document after choosing the topic to explain the plan and methods used to work on this project. The group analyzed real-life examples, noting down each requirement as a user story in order to better understand and highlight real-life shortcomings that would affect how we would develop the program after writing down the project description which was helpful for the group.

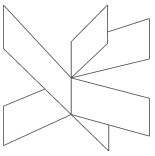
The group started by defining a set of requirements which had to be handled during the sprints of the project. During every sprint the process of implementing requirements was slightly reformed to achieve a more suitable approach which could be adapted for later sprints. For example, in the beginning the group started by issuing a set of requirements which have been excessively general. With every sprint, the group realized that it is not possible to implement multiple big requirements during the set time period for the sprint (3 days). In order to properly handle the development process, the group decided to split them into smaller, more realistic ones that could be easily implemented during a sprint duration.

The group's main goal from the project description phase was to develop a set of requirements that would be handled through the project execution phase . This was achieved after familiarizing with the scrum framework during the documentation of the project description, which led to the assigning of a product owner whose role was to assess the groups work, and evaluate if the requirements were delivered properly.



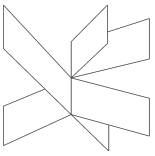
5 Project Execution

The group planned to execute the project using the SCRUM framework guidelines during the different phases of the Unified Process.
They include conducting sprints, and holding daily meetings to keep track of the project progress.
During the inception phase the group established the vision and scope of the project, and backlogged the different sprints performed during this phase, which helped in identifying the key issues that might have risen, said issues were documented in the project description, and were considered during the construction phase.
In the elaboration phase the group focused on assessing the different requirements established in the inception phase. The process of elaboration was heavily dependent on the input of the product owner with whom the designing and the analysis phase of each sprints were completed, thus giving a well structured base for the construction phase.
The elaboration phase also gave the group a chance to evaluate and conduct testings for some of the features which were implemented, thus providing an opportunity to reform the code more efficiently in the construction phase.
The construction phase was built upon the work established during the initiation and elaboration phases. The main objective of this phase was to construct the requirements established by the product owner during the inception phase. In collaboration with the scrum master’s evaluation, the group members went forth with the development of the system, and conducted tests of the work after completion. By taking advantage of the sprints concept from the scrum framework, the group was able to gradually implement the requirements in a sufficient manner that guaranteed that most of them would be implemented. By breaking the requirements into smaller components, the group was able to take some input from the supervisor, thus making the analysis and design of the requirements more manageable and applicable in a sufficient way that does not lead to time consumption.



The focus of the transition phase of the project was to test and finalize the coding of the system. The main goal of the testing was to generate feedback that would help in debugging and further refine the code.
As mentioned before, the group took advantage of the scrum framework. After every last day of a sprint the group had a review meeting to evaluate the progress.After that a new SCRUM master and product owner were elected and they planned the next sprint. Project risks were identified and monitored at every sprint review in order to ensure essential parts of the project were not obstructed. The result was approved by the product owner and made to meet their view of the final product.

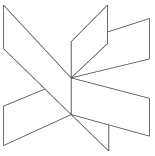
The group found the scrum framework satisfactory, since its methods helped facilitate the workflow successfully.
The group is satisfied with the work achieved, for it allowed for the implementation of the different subjects learned during the semester, as well as by applying the deep learning method in combination with project based learning, the group gained a better comprehension of the material presented through the semester.



The identification of risks was established during the inception phase of the project. The following list contains the risks which were identified and their handling by the group during the construction and transition phase:

Identified risk	The group's handling of the identified risk
Users are unable to create a chatroom with multiple users.	The group was able to implement a chat room because one of the main focuses was on creating and perfecting a one to many chat feature, thus negating the risk, and avoiding a removal of the feature.
Bad coordination due to members working/traveling .	Coordination was maintained by regular conferences to discuss statuses of each member's tasks with reviews so that no member leaves the scope of their own task or the project. The means of communication were in person meetings, and online discord meetings.
Prolonged Development Time	The group focused on avoiding spending time on specific features that might have taken too long to develop by cutting on their scope or by discussing alternatives with the supervisor
Loss of chat log data	By using properly assigned database connections, the loss of chats was negated.
Ambitious scope	The project was kept in control by first creating a barebones program with essential features so as to not lose focus on the main objectives of the project so that new and ambitious functionalities could be coded without disturbance.

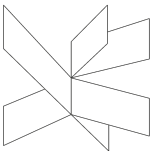
As shown by the table, the group's major approach to rising issues and risk was to involve the supervisor's input, try to better coordinate approaching tasks, and regular meetings to assess the work of the members.



The following requirements were not implemented:

1. As a user, I want to block belligerent users, so I can avoid communicating with them.
2. As a user, I want to select a theme for my chat, so that I can make it my own.
3. Download chat backup/chat logs

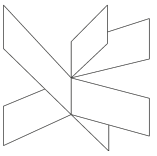
The main reason why they were not implemented was that the group chose to focus on the most important features of the chat, which were considered of higher priority by the scrum team and the scrum master.
Although there was a conflict with the product owner who emphasized the necessity of focusing on customization features, the group deemed it more appropriate to focus on the essential and critical requirements that characterize the system as a chat system. Therefore less effort was relegated toward applying them in the construction phase, which resulted in them being incomplete by the deadline.



6 Personal Reflections

Ameya Mahakal :

With valuable experience gained from semester one as well as feedback received from SEP-1 exam,I and our group decided to start a different approach to give equal efforts to documentation and coding which was helped with the highly restrictive waterfall approach being scrapped in favor of the detailed SCRUM framework and Unified Process.The group all were motivated by the project topic which felt unique and would truly test the group where everyone contributed their hundred percent be it coding or documenting although documentation is a weakness of the point which will be my main focus of improvement for myself and hope I will be able to program further while hoping to to neglect programming.The group complements each other well even if are cultural backgrounds may differ, I have learnt much from my fellow members and also guided my teammates.We all agreed in principle with each part of the cultural map.I truly thrived as a team member and did not enjoy being SCRUM master.I was satisfied with my work on documentation but felt overwhelmed by taking in charge of documenting which left me no room to code as much I would like to but still am satisfied with my profanity filter.The group often did argue but did not escalate beyond stage 2 of Friedrich Glasl’s conflict stairs and used the supervisors help when arguments could not be settled.Overall I am satisfied with the project which met all essential features .



Bozhidar Manev:

Coming from a weak first-semester project(SEP1), the group was motivated to achieve more. We knew our weaknesses and we were aiming at overcoming them in this semester project(SEP2). Big changes were made to improve the group's performance. The first one of them has been to ditch the aged Waterfall method of development and start working with a newer and more efficient SCRUM framework, combined with the Agile method - Unified Process. They helped us to better structure our workflow and allowed us to work in an iterative manner. The project topic was chosen and liked by all members of the group, which facilitated our discussions and development. Even thought we might have different visions, based on the different experience we have with such systems, we came up with a system that covers each of ours demands from the system. For me personally, with this project I wanted to improve my project developing skills and take part in as many aspects as possible. Quality code and well structured diagrams were a goal which i nearly achieved. Even thought the code wasn't able to follow all principles I wanted, it's more important for me that I know what should be changed and how. I am satisfied with my work for the time period we had and the results we achieved

Joan Tamo :

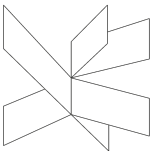
By the end of the second semester we decided to use our experiences from last semester's sep1 exam as base for how to proceed with sep2. We mainly tried to not repeat the same mistakes. In Sep2 we involved the supervisor more often, and we started documenting more often.

Working with scrum helped us a lot in organizing the project. The coding and documenting process during each sprint were the highlights, because the sprints were like mini projects within the main sep2 project.

The project period helped us apply more critical analysis, which is an aspect of deep learning, that leads to developing a more personalized knowledge of the subjects, and this makes them easier and more appealing when this knowledge is used in the future.

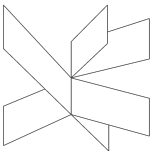
Radoslav Kiryazov :

Approaching the end of our second semester project, I can’t lie and say this has been a smooth sailing. This time we tried to approach things differently, look at things from different angles. We tried to avoid repeating the same mistakes as last time, and in my opinion we managed to avoid some of them, but not all. We collaborated, we tried to account for everyone’s strong points and even give tasks that put members out of their comfort zone, hopefully giving them a chance to improve themselves in the process. I hated working with Waterfall last year, it felt so innovative and robust, with its philosophy of barely ever going back, meaning that if you fail to recognize your mistake at the early stages, it might just be too late. Scrum has been much kinder to me this time around with its way of working and organizing stuff with its philosophy of being agile, allowing much more room for mistakes and time for their resolutions. I am happy with the way we work and the interactions we have as a group, but I can’t say that the quality of work matches the level of tiredness I'm feeling currently.



Zsolt Nóvé :

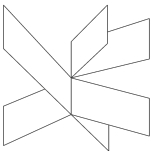
In the group project each group member had their fair share of work. I have always been a talkative person, but this year was not without conflict. I have learnt to cooperate with the team more in order to make sure everyone is doing a specific task that has to be documented and agreed on by the group in order to not repeat the same process and to work more efficiently with task based work.
If we have project work again using Scrum and taking a leader to the group that would be a benefit for all of us. Our group has very talented people and they all have a great knowledge but all of us are different, therefore we could divide the programing and documenting parts. Personally I wanted to improve my overall understanding of the whole project and every part of it. As last year I was understanding more in the code this year I had to rely on the help of my groupmates. Unfortunately , due to my accident I tried to participate in person as much as possible, but the team has agreed to work more online. I personally am not pleased with my results in the coding part. Last year I had made more working code therefore, I don't find it very successful.



7 Supervision

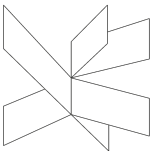
The group cooperated with our supervisors and was beneficial for the project work due to the participative supervisor approach who offered guidance while offering the group the chance to take initiative in the creation of all aspects of the project.

The group would communicate with the supervisors at the end of every sprint to summarize and check if the work is up to standard, and fix any mistakes and misunderstandings.



8 Conclusions

Overall, the project was invaluable in increasing the group's expertise and work ethic where the group gave it their level best to achieve the essential features of the project Some tactics to enforce next thereafter maybe is to focus on refining the groups documentation skills and to have a strict adherence to a time schedule to prevent any less refined work.



9 Logbook

Product Backlog.[1]

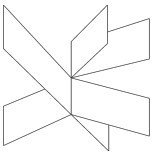
User Story	Time
As a user, I want to send a message to other users so that I can communicate with other users.	9
As a user, I want to have a personal profile, with which other users can connect.	3
As a user, I want to have an access to the history of my chats(logs), so I can read my previous conversations	3
As a user, I want to be able to manage my chats, so I can create or delete chats.	4
As a user, I want to create a group chat, so I can communicate with multiple users.	6
As a user, I want to edit my profile information.	4
As a user, I want to hide offensive and insensitive language.	3
As a user, I want to block belligerent users, so I can avoid communicating with them.	7
As a user, I want to select a theme for my chat to make it my own.	2
Download chat backup/chat logs	3

Sprints Information.

Length: Each sprint is 3 days long:

Meetings: One big meeting on the last day of the sprint covering sprint retrospective, review, the supervisors input, and planning the next sprint.

Scrum master Rotates each sprint.



Sprint 1:

Start Date: 23 November 2022

End Date: 26 November 2022

Scrum Master: Bozhidar Manev

Sprint Review + Sprint Retrospective + Sprint Planning meeting: 26 November 2022

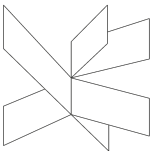
Sprint Backlog:

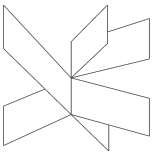
Working On

1	As a user, I want to send a message to other users so that I can communicate with other users.	9
2	As a user, I want to have a personal profile, with which other users can connect.	3

Task	Time	Status	Done by:
Create an RMI Server	2	Done.	Radoslav
Create an RMI Client	2	Done.	Zsolt
Establish a connection to the server.	1	Done.	Joan
Create a test chat view	3	Done.	Ameya
Implement Observer Pattern and MVVM	2	Done.	Bozhidar

Results: 1: Moved to next 2: Moved to next





Sprint 2:

Start Date: 27 November 2022

End Date: 30 November 2022

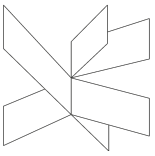
Scrum Master: Bozhidar Manev

Sprint Review + Sprint Retrospective + Sprint Planning meeting: 30 November 2022

Sprint Backlog:

Working On

1	As a user, I want to send a message to other users so that I can communicate with other users.	9
2	As a user, I want to have a personal profile, with which other users can connect.	3

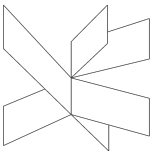


Tasks	Time	Status	Done by:
UseCase Diagram	½	Done	Joan
UseCase Descriptions	1	Moved to next.	Joan
Activity Diagrams	1	Moved to next.	Joan
DataBase requirements	1	Done.	Bozhidar
Conceptual ER diagram/EER/GRD(Normalization)	2	Moved to next.	Everyone/Group Meeting.
Document Process Report	6	Moved to next.	Ameya
Document Cultural Background	1	Done	Everyone

Results:

1: Fully implemented

2: Moved to next



Sprint 3:

Start Date: 1 December 2022

End Date: 3 December 2022

Scrum Master: Radoslav Kiryazov

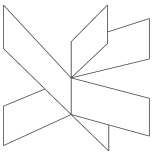
Sprint Review + Sprint Retrospective + Sprint Planning meeting: 3 December 2022

Sprint Backlog:

Working On

1	As a user, I want to have a personal profile, with which other users can connect.	**3**
2	As a user, I want to have a personal profile, with which other users can connect.	**3**

Tasks	Time	Status	Done by:
Database Requirement		Done.	Everyone/Group Meeting.
Conceptual ER diagram/EER/GRD(Normalization)	3	Done	Bozhidar
Create Database	4	Done	Bozhidar
Connect Database.	3	Done	Radoslav
Sign-in GUI.	5	Done	Radoslav



Sign-up GUI with methods.	2	Done	Radoslav
Document Process Report	6	Moved to next.	Ameya
Use case description	3	Done.	Joan
System Sequence Diagram	½	Done.	Ameya
Activity Diagram	1	Done.	Joan

Results:

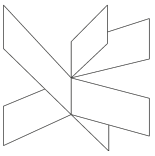
1: Fully Implemented. 2: Moved to next.

Sprint 4:

Start Date: 4 December 2022

End Date: 7 December 2022

Scrum Master: Ameya Mahakal



Sprint Review + Sprint Retrospective + Sprint Planning meeting: 7 December 2022

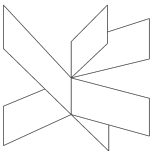
Sprint Backlog:

1	As a user, I want to hide offensive and insensitive language.	9
2	As a user, I want to have an access to the history of my chats(logs), so I can read my previous conversations	3
3	As a user, I want to select a theme for my chat to make it my own.	2
4	As a user, I want to be able to manage my chats, so I can create or delete chats.	4

Working on

Tasks	Time	Note	Done by:
Access previous chats	5	Done	Joan
Document Process Report	6	Moved to next.	Ameya
Document Project Report	6	Moved to next.	Ameya
Sign in functionality.	3	Done	Radoslav
Connect sign-up to the database.	4	Done	Radoslav
Chat GUI	6	Done.	Bozhidar
Store the messages in the database.	4	Done	Joan
Censoring expletives/swears	2	Done	Ameya
Chat customization	7	Moved to next.	Zsolt

Result: Everything except 3 has been implemented



Sprint 5:

Start Date: 8 December 2022

End Date: 11 December 2022

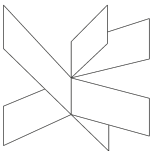
Scrum Master: Joan Tammo

Sprint Review + Sprint Retrospective + Sprint Planning meeting: 10 December 2022

Sprint Backlog:

1	As a user, I want to select a theme for my chat to make it my own.	**2**
2	As a user, I want to create a group chat, so I can communicate with multiple users.	**6**
3	As a user, I want to edit my profile information.	**4**

Task	Time	Note	Done by
Redesign Sign-up and Log-in Views	3	Done	Radoslav
User Manual	3	Moved to next.	Radoslav
Document Process Report	6	Moved to next.	Ameya/Joan
Document Project Report	6	Moved to next.	Ameya/Joan
Actor Description	1	Done	Zsolt



Domain Model Description	1	Done	Zsolt
Adding more people to a chat	2	Done.	Bozhidar
View and methods for updating profile credentials.	4	Done.	Radoslav
Chat customization	7	Moved to next.	Zsolt

Result:

Everything except 1 has been implemented

Sprint 6:

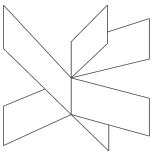
Start Date: 12 December 2022

End Date: 16 December 2022

Scrum Master: Joan Tammo

Sprint Review + Sprint Retrospective + Sprint Planning meeting:

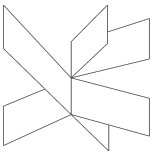
Sprint Backlog:



1	As a user, I want to select a theme for my chat to make it my own.	2
2	As a user, I want to block belligerent users, so I can avoid communicating with them.	7
3	Download chat backup/chat logs	3

Tasks	Time	Status	Done By:
Class diagram	5	Done	Radoslav/Bozhidar
Sequence diagram	4	Done	Joan
Document Process Report	3	Done	Everyone
Document Project Report	3	Done	Everyone
Test cases	2	Done	Ameya
Junit testing	5	Done	Ameya
System sequence diagrams	3	Done	Joan
User Manual	3	Done	Radoslav
Chat Customization	7	Failed	Zsolt

Bring ideas to life
VIA University College



Project Report

Chat System

Ameya Mahankal(326157)

Bozhidar Manev (326391)

Joan Tammo (325753)

Radoslav Kiryazov (326155)

Zsolt Nóvé (326345)

Supervisors:

Henrik Kronborg Pedersen,

Jørn Martin Hajek

VIA University College

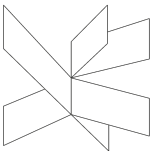
[30452 characters]

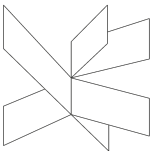
[Software Technology]

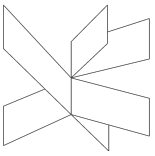
[Second Semester]

[11.12.2022]

Table of content







Abstract

The goal of this project is to program and document the process of creating a chat system by devising requirements and with the help of different diagrams while using the SCRUM framework, while taking into account agile principles, and working within the unified process frame of work.

The program is a client/server system using RMI with applied relevant design patterns such as MVVM, Factory Pattern, Observer pattern, Singleton and Adapter and following the SOLID principles resulting in a maintainable and expandable software.

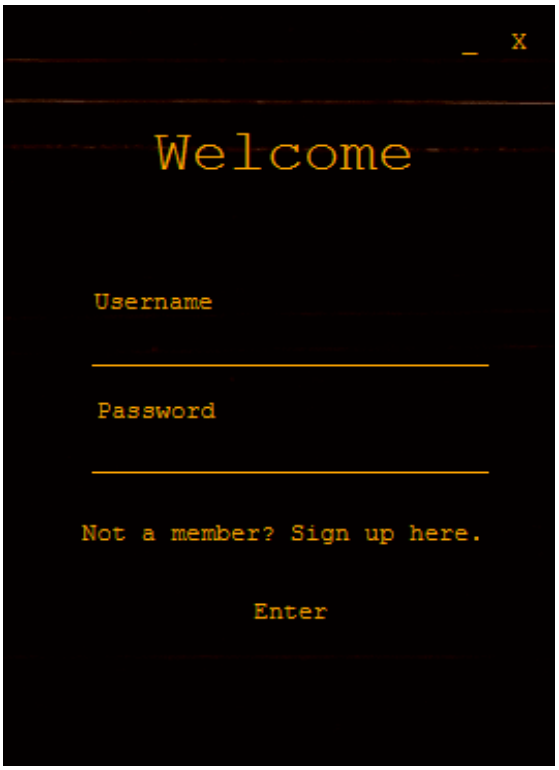
The GUI was made in Scene builder and coded in JavaFX.

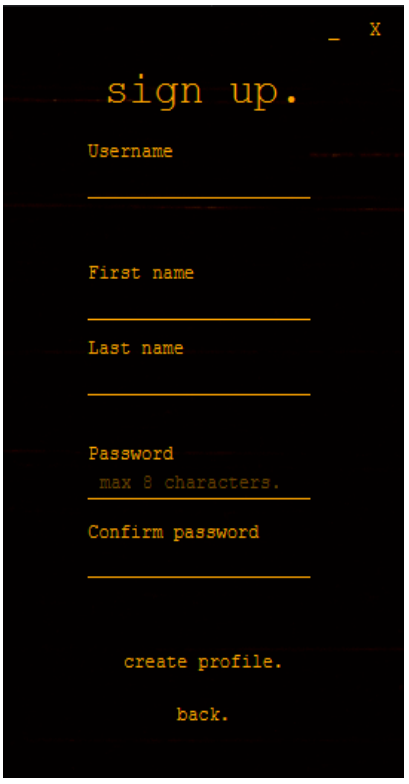
The project was analyzed and designed with the help of UML diagrams, was written in Java, graphically designed using FXML, further visually enhanced by CSS and connected to our own designed relational database using postgreSQL.

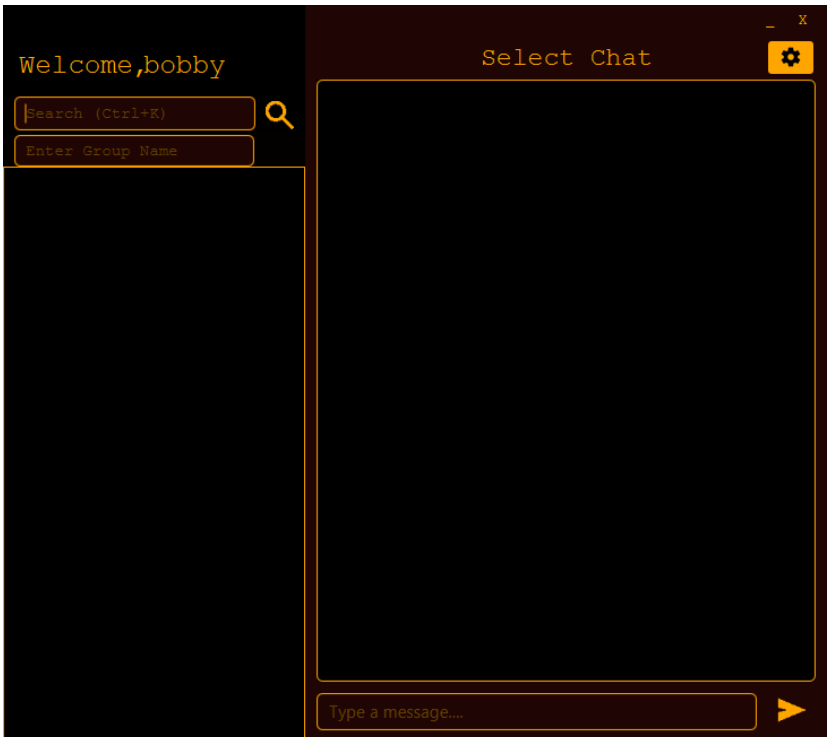
The group has successfully implemented all critical and high priority requirements. Resulting in a working program capable of creating a one-to-one as well as one-to-many chat groups. The database was implemented successfully and has been linked to the server using elephantSQL.

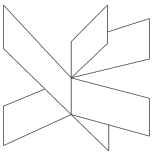
1. Introduction

The main purpose of the project was to create a client/server chat system. A program that isn't bloated by unnecessary features,is free of distractions and is easy to use. We're pleased to state that the group believes that we have managed to deliver what we promised.

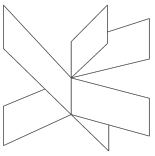








There are no voice messages, audio nor video chats, and any type of communication media due to the group's inexperience in the field. Looking back on our newly gained knowledge of design patterns, scrum, and much more, we believe to have managed to implement them to the best of our abilities during the creation of this project.
This report expands on the difficulties that have arisen along the way, and their solutions.



2. Analysis

2.1 Requirements:

The requirements are the demands which are made by the product owner, and assessed by the development team. The priorities were assessed based on the group's definition of what makes a chat system a chat system, which is that it sends and receives messages, and that a user must have a personal profile which is unique from other user profiles.

Critical Requirement(s).

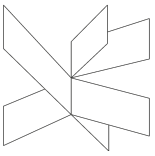
1. As a user, I want to send a message to other users
2. As a user, I want to have a personal profile, by which other users can connect with.

High Priority.

3. As a user, I want to have access to the history of my chats(logs),so I can read my previous chats.
4. As a user, I want to be able to manage my chats, so I can create or delete chats.
5. As a user, I want to create a group chat, so I can communicate with multiple users.

Low Priority.

6. As a user, I want to edit my profile information.
7. As a user, I want to hide offensive and insensitive language.
8. As a user, I want to block belligerent users, so I can avoid communicating with them.
9. As a user, I want to select a theme for my chat, so that I can make it my own.
10. Download chat backup/chat logs



2.2 Actors Description

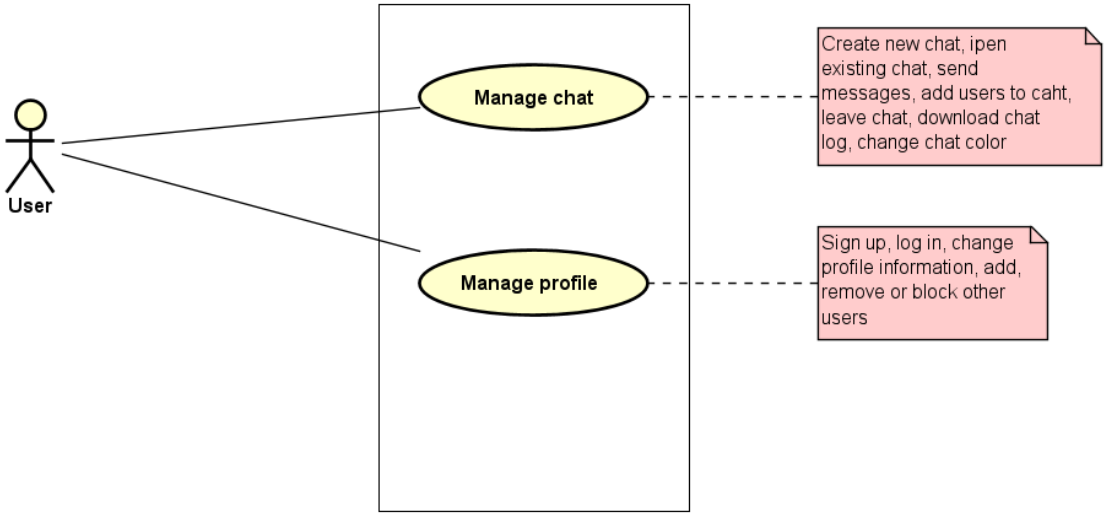
User : The user registers him/herself at the start page of the app.The database registers the new user and all its information.The new user will be visible in the search menu due to its registration in the database.

After the user is able to search for other users in the search bar. When the user finds another they write or upload message types and press the send button on the user's interface.

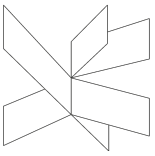
The user sends and receives text messages that are being saved in the database.

2.3 Use Case Diagram:

The use case diagram shows the main actor(the user), and the interactions they can perform with the system.



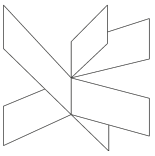
The user can create new chats with other users or open existing ones. There they can send messages, adding more users and leaving unwanted chats.
The user also has the option to customize the background of their chats and download a chat log for any chat they have access to.
The user can log-in into the system or if needed they can sign-up to the system.
The user can change their profile information. Furthermore they could be able to manage their connections by adding, removing or blocking other users.



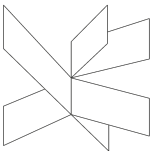
2.4 Use case description:

Use case: **Management of chat**:

The focus of this use case is to facilitate the different functionalities related to the management of chat. Due to the development team's definition of what a chat system is, the main focus of Chat management is to send a message, thus this functionality was chosen as the main success scenario.

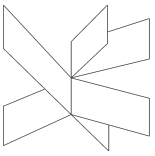


Use case description	
Use case name	Manage chat
Scope	Chat System
Level	User Goal
Primary actor	User
StakeHolders and interests	User(s):wants to send, and receive messages to one or many users, as well as also wanting the ability to customize the chat window.
Preconditions	User logged in the system
Success Guarantee	The message is sent and delivered once the receiver checks their side of the chat catalog, and the message is saved in the database.
Main Success Scenario	1- User chooses to open an existing chat 2- User sends messages to other users. The user can repeat step 2 as much as they desire.
Extensions	1a-Establish a new chat with one more user. 2a -The user can leave a chat. 2b - Add more users one at a time to a chat 2c - Change chat color. 2d - Download Chat log. *a- Exit the application.
Frequency of Occurrence	Could be nearly continuous.



Use case: **Management of profile**

In this use case the development team chose to focus on logging in as the main definition of what a chat system should revolve around in regards to the management of profile.

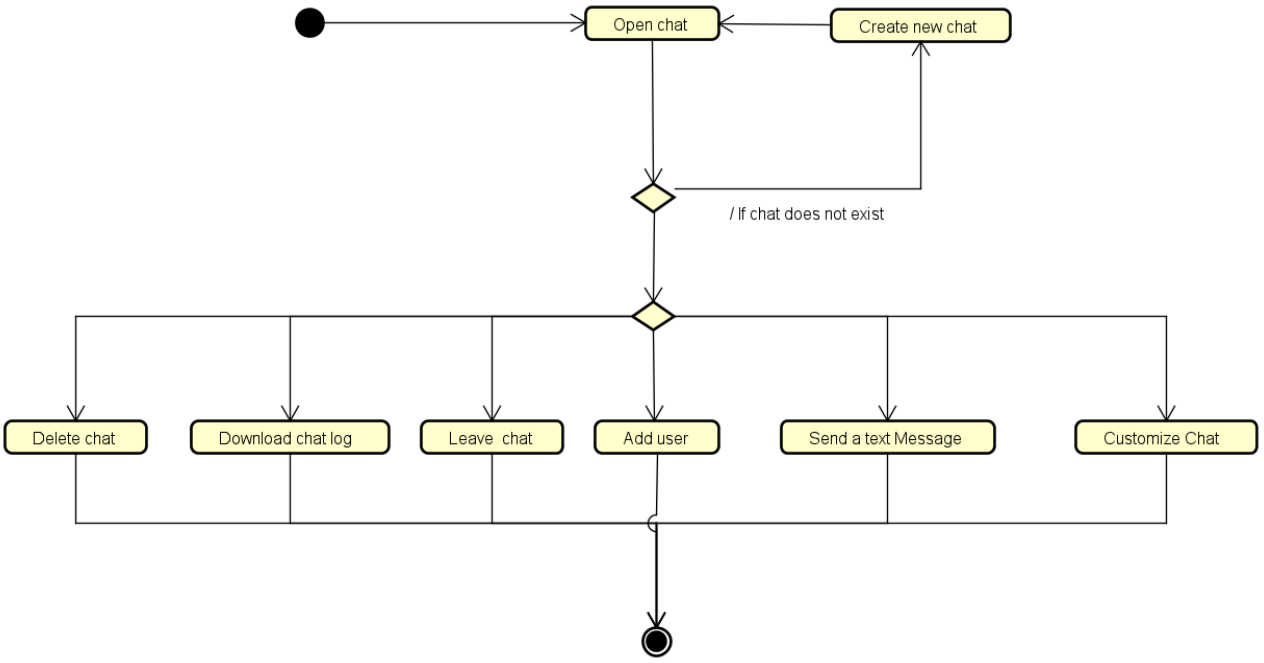


Use case description	
Use case name	Manage profile
Scope	Chat system
Level	User Goal
Primary actor	User
StakeHolders and interests	User: Wants to log in and create a profile which will allow them to make use of the system's features. The user also wants the ability to customize their profile and the themes of the system.
Preconditions	User exists in the system
Success Guarantee	The user is able to log in with a username and a password, and can use the system's features.
Main Success Scenario	1 - The user can log in to the system using their unique username and password. 2 - The user can change their profile information
Extensions	1a - If the user doesn't have an already existing account then they sign up to the system: 1a.1 - The user enters a first name, last name, username and a password 1a.1.1 If the username already exists in the database , the user will be asked to enter a different one. 2.a - Manage connections 2.a.1 - add users 2.a.2 - remove users 2.a.3 - block users *a- Exit the application.
Frequency of Occurrence	Depending on user
Miscellaneous	

2.5 Activity diagrams

Manage Chat Activity Diagram

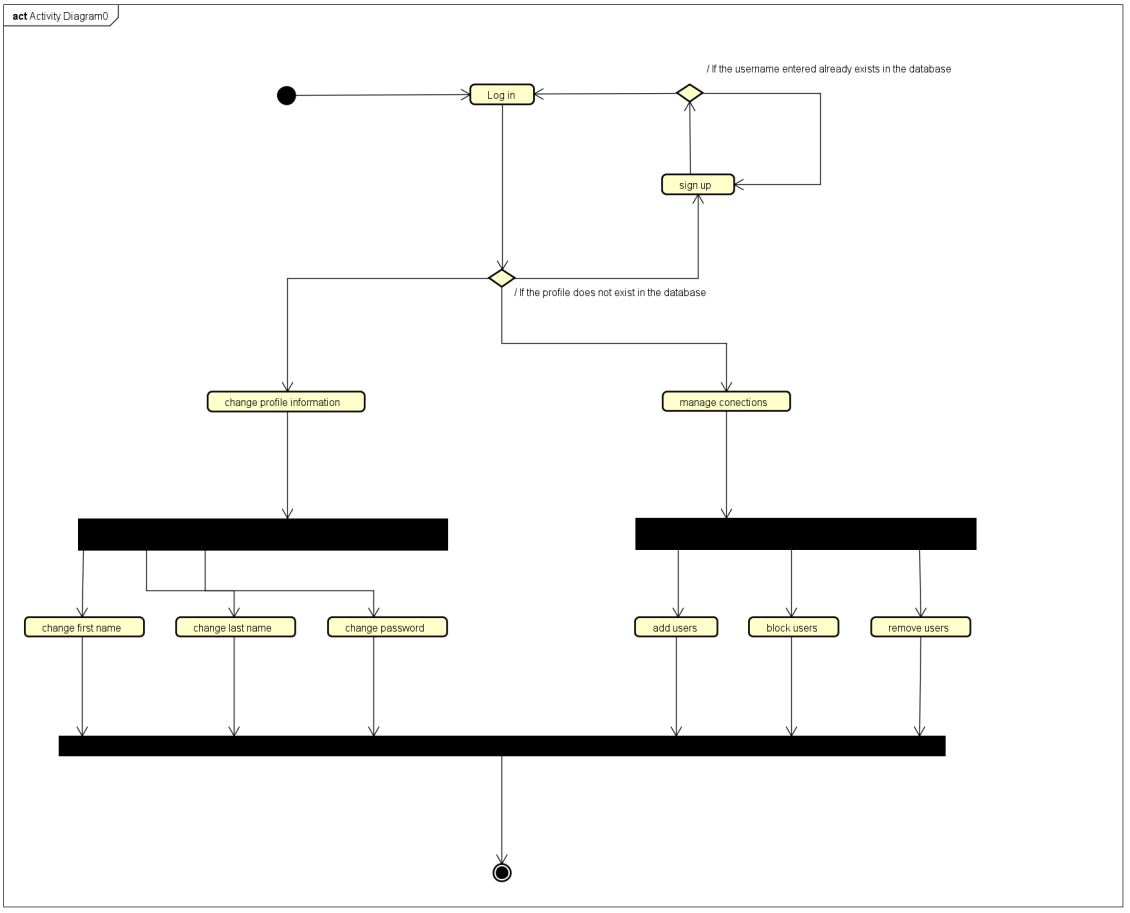
The following activity diagram shows the different activities that the system provides for the user to interact with.

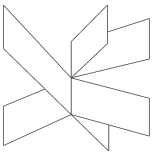


After creating a chat, the user can access said chat and perform multiple operations, such as sending text messages, customizing their chat, adding multiple users, deleting specific messages, or deleting entire chat catalogs.
The main objective of this activity diagram is to showcase the different options a user has in regards to the managing of the chats use case.

Manage profile activity diagram

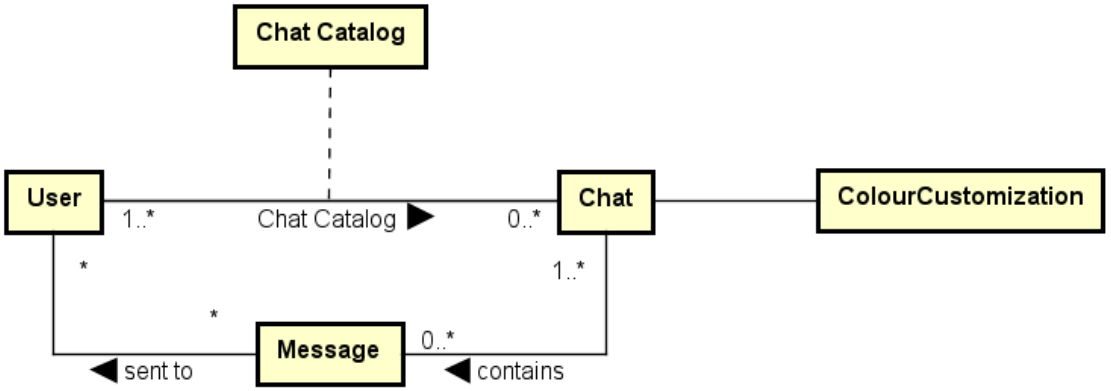
The manage profile case is illustrated below which shows the different changes the user can make to his profile





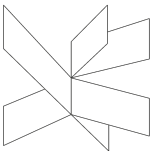
2.6 Domain model

The domain model diagram below shows the overall basic structure and relationships of the program on which is based the design.



2.6 Domain Model Description

The user can access different chats with users. Users can send text messages. The chat always creates a chat log that saves the sent messages. The user can send multiple messages into a chat.Chat can be customized by changing the background color. Messages are only in text format.



3. Design

The architectural pattern used was MVVM, displayed in the class diagram below,to structure different parts of the program

namely:

Client

- **core** - There are kept the factories of the viewHandler, viewModel,Model,Client to create and maintain a single instance of each class of each subject respectively.
- **model** - Consists of the project's client-side business logic.
- **networking** - Contains the class that is responsible for the connection to the server.
- **style** - Contains additional visual settings for the GUI, into CSS files.
- **views** - Contains a package for each of the different views, with their respective FXML file, Controller and View Model.

Server

- **model** - Contains the Server-side logic which the client requests
- **networking** - Contains Server implementation that has methods to start the server and return info or changes to/for the user

Database

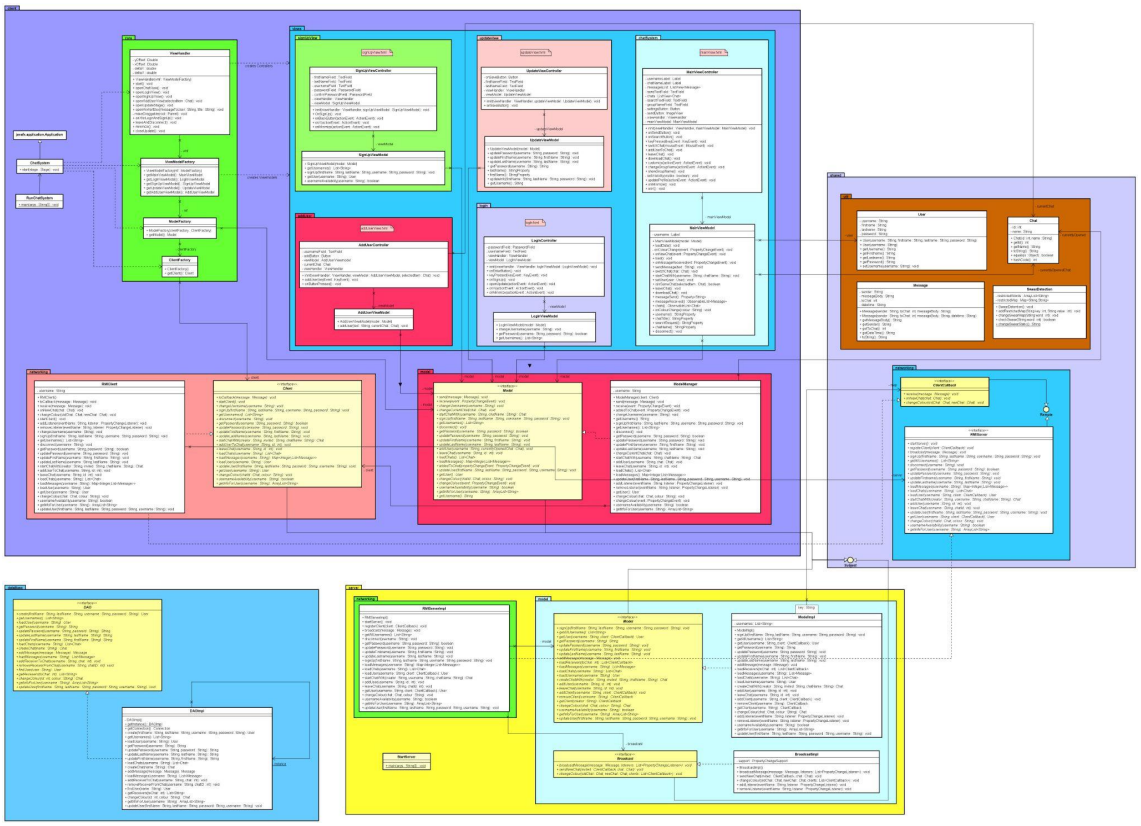
Contains the DAO singleton class, responsible for communicating with the database, which is stored in the cloud using ElephantSQL.

Shared

- **networking** - Contains the interfaces which are used in both Client-side and Server-side.
- **util** - Contains serializable class Message along with profanity filter.

3.1 The class diagram:

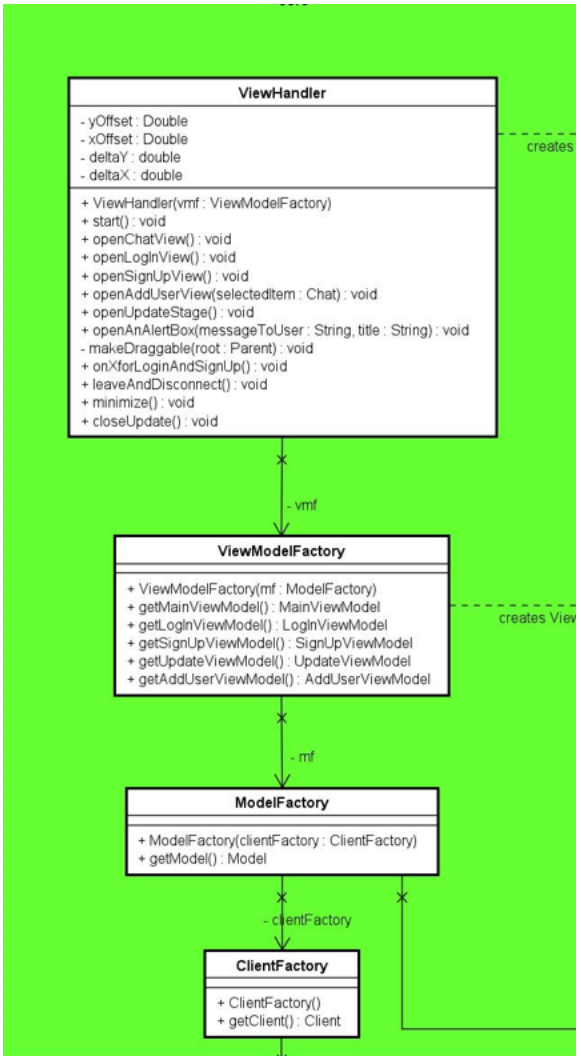
The following is the class diagram for the project, it will be used through the report to elaborate and the design choices for the system:



3.2 Use of Design Patterns:

Factory: The pattern was used to create client, model, viewmodel. Shown below are examples of the ClientFactory, ModelFactory and the ViewModelFactory.

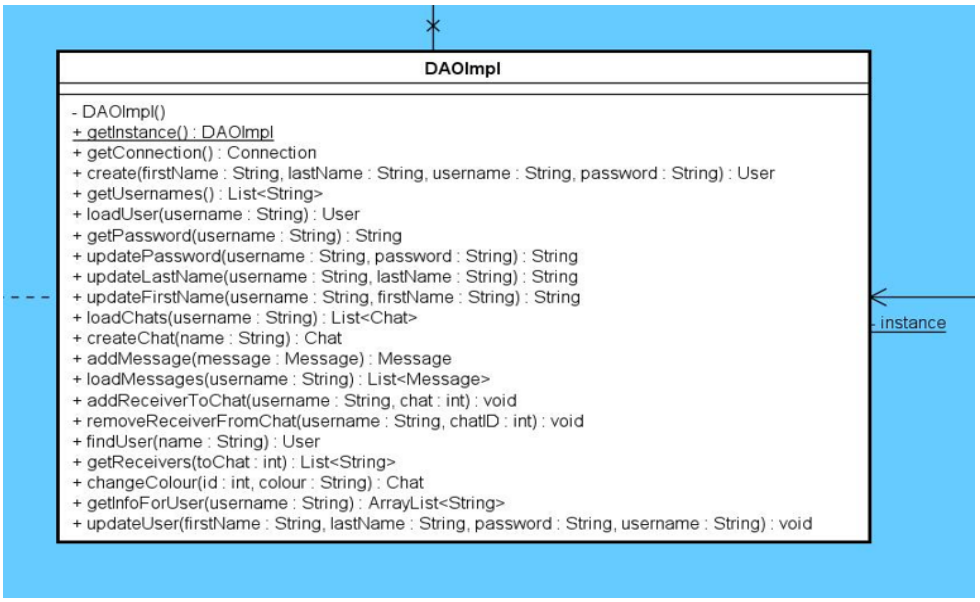
```
public MainViewModel getMainViewModel() {
    if (mainViewModel == null) {
        mainViewModel = new MainViewModel(mf.getModel());
    }
    return mainViewModel;
}

1 usage  radoslavKiryazov
public LogInViewModel getLogInViewModel() {
    if (logInViewModel == null) {
        logInViewModel = new LogInViewModel(mf.getModel());
    }
    return logInViewModel;
}

1 usage  radoslavKiryazov
public SignUpViewModel getSignUpViewModel() {
    if (signUpViewModel == null) {
        signUpViewModel = new SignUpViewModel(mf.getModel());
    }
    return signUpViewModel;
}

1 usage  radoslavKiryazov
public UpdateViewModel getUpdateViewModel() {
    if (updateViewModel == null) {
        updateViewModel = new UpdateViewModel(mf.getModel());
    }
    return updateViewModel;
}

1 usage  Bozhidar Manev
public AddUserViewModel getAddUserViewModel() {
    if (addUserViewModel == null) {
        addUserViewModel = new AddUserViewModel(mf.getModel());
    }
    return addUserViewModel;
}
```




```
public class ModelFactory {
    private ClientFactory clientFactory;
    private Model model;

    public ModelFactory(ClientFactory clientFactory) {
        this.clientFactory = clientFactory;
        model = getModel();
    }

    public Model getModel() {
        if (model == null) {
            model = new ModelManager(clientFactory.getClient());
        }
        return model;
    }
```

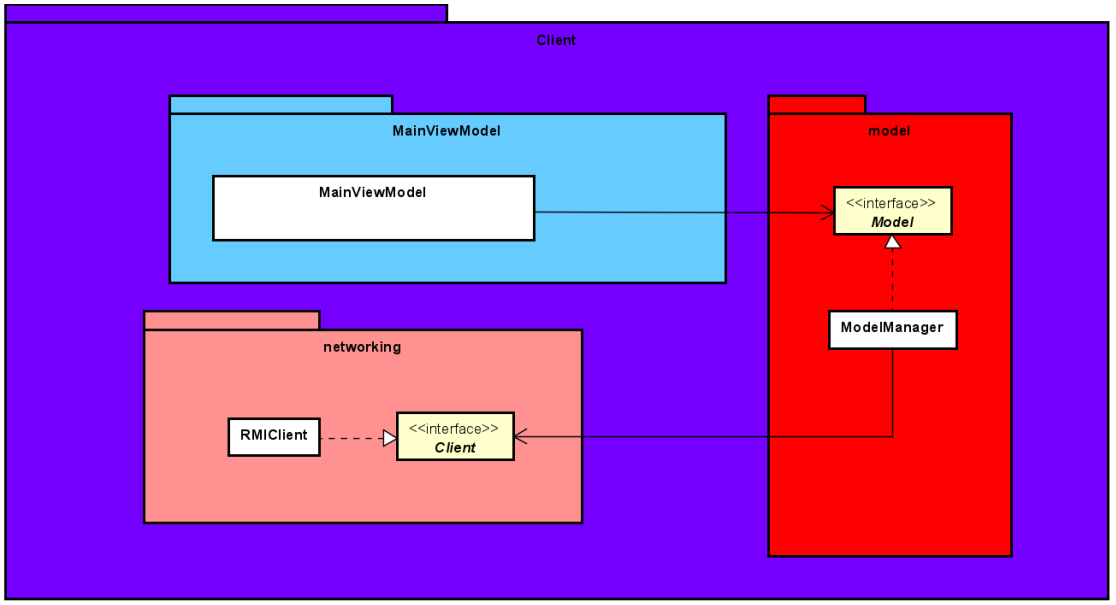
```
public class ClientFactory {

    private Client client;

    public ClientFactory() { this.client = getClient(); }

    public Client getClient() {
        if (client == null) {
            client = new RMIClient();
        }
        return client;
    }
}
```

Singleton: The singleton pattern was used in the DAOImpl class, to ensure a single tunnel of connection to the database.




```
private static DAOImpl instance;

private DAOImpl() throws SQLException
{
  DriverManager.registerDriver(new org.postgresql.Driver());
}

public static synchronized DAOImpl getInstance() throws SQLException
{
  if (instance == null)
  {
    instance = new DAOImpl();
  }
  return instance;
}
```

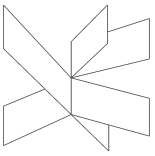
Adapter: The adapter pattern can be identified in the class diagram where MainViewModel can be considered as the client, Model as the target, ModelManager as the adapter and the (RMI)Client as an adapter.(Added another diagram with method and operation visibility off below)



Observer pattern: Observer pattern was used multiple times in the program, with its main usage being moving information from the Model in Client to any ViewModel, shown below is the observer pattern used as a way of communication between the Model and the MainViewModel. The MainViewModel is added as a listener to the model, and depending on the eventName ex.”MessageReceived” is going to call the appropriate method, “this::onMessageReceived”.

The interface implemented by every listener.

```
public interface Subject {
    4 implementations  radoslavKiryazov
    void addListener(String eventName, PropertyChangeListener listener);

    4 implementations  radoslavKiryazov
    void removeListener(String eventName, PropertyChangeListener listener);
}
```

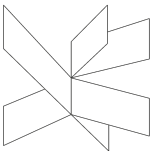


MainViewModel

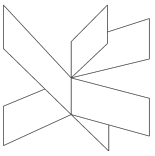
```
public MainViewModel(Model model) {
    this.model = model;
    messageSend = new SimpleStringProperty();
    username = new SimpleStringProperty();
    chatTitle = new SimpleStringProperty();
    searchRequest = new SimpleStringProperty();
    writeChatName = new SimpleStringProperty();
    currentConversation = FXCollections.observableArrayList();
    currentlyOpenedChat = new Chat( id: -1,  name: "Select Chat");
    chatCatalog = FXCollections.observableArrayList();


    chatMap = new HashMap<>();
    user = null;
    username.setValue("");
    chatTitle.setValue(currentlyOpenedChat.getName());
    model.addListener( eventName: "MessageReceived", this::onMessageReceived);
    model.addListener( eventName: "AddedToChat", this::onNewChat);
    model.addListener( eventName: "ColourChanged", this::onColourChange);

    loadData();
```

Methods to be called on the corresponding eventName.

```
public void onMessageReceived(PropertyChangeEvent evt) {
    Platform.runLater(() -> {
        Message msg = (Message) evt.getNewValue();
        int toChat = msg.getToChat();
        chatMap.putIfAbsent(toChat, new ArrayList<>() {
        });
        chatMap.get(toChat).add(msg);
        if (toChat == currentlyOpenedChat.getId()) {
            currentConversation.add(msg);
        }
        /*New messages will always be added to chatMap
         * and in case the received message is for the opened chat,
         * it will be added to the opened chat as well */

    });

}
```



```java
private void onColourChange(PropertyChangeEvent event) {
    Chat newChat = (Chat) event.getNewValue();
    Chat oldChat = (Chat) event.getOldValue();

    chatCatalog.set(chatCatalog.indexOf(oldChat), newChat);
    if (currentlyOpenedChat.equals(oldChat)) {
        currentlyOpenedChat = newChat;
    }
}

public void onNewChat(PropertyChangeEvent event) {
    Platform.runLater(() -> {
        Chat chat = (Chat) event.getNewValue();
        chatCatalog.add(chat);
        chatMap.put(chat.getId(), new ArrayList<>());
    });

}
```



```
public void receive(PropertyChangeEvent event) {
    Message msg = (Message) event.getNewValue();
    support.firePropertyChange( propertyName: "MessageReceived",  oldValue: null, msg);
    System.out.println("Model Manager::receive " + msg);
}

1 usage  Bozhidar Manev
@Override
public void addedToChat(PropertyChangeEvent event) {
    Chat chat = (Chat) event.getNewValue();
    System.out.println("ModelManager2");
    support.firePropertyChange( propertyName: "AddedToChat",  oldValue: null, chat);
    System.out.println("Model Manager::added to " + chat);

}
```

Model

Commmunication between server and client

RMI was selected over Sockets because the group found it simpler and more comprehensive thanks to its handling of the objects transferred between client and server in the background, removing clutter.

Shown bellow are the RMIServer and ClientCallback both extending Remote.

```
public interface RMIServer extends Remote {
    1 usage  1 implementation  Bozhidar Manev
    void startServer() throws RemoteException;

    1 usage  1 implementation  radoslavKiryazov
    void registerClient(ClientCallback client) throws RemoteException;

    1 implementation  radoslavKiryazov
    void broadcast(Message message) throws RemoteException;

    1 implementation  radoslavKiryazov
    void signUp(String firstName, String lastName, String username, String password) throws RemoteException;

    1 usage  1 implementation  radoslavKiryazov
    List<String> getAllUsernames() throws RemoteException;
```

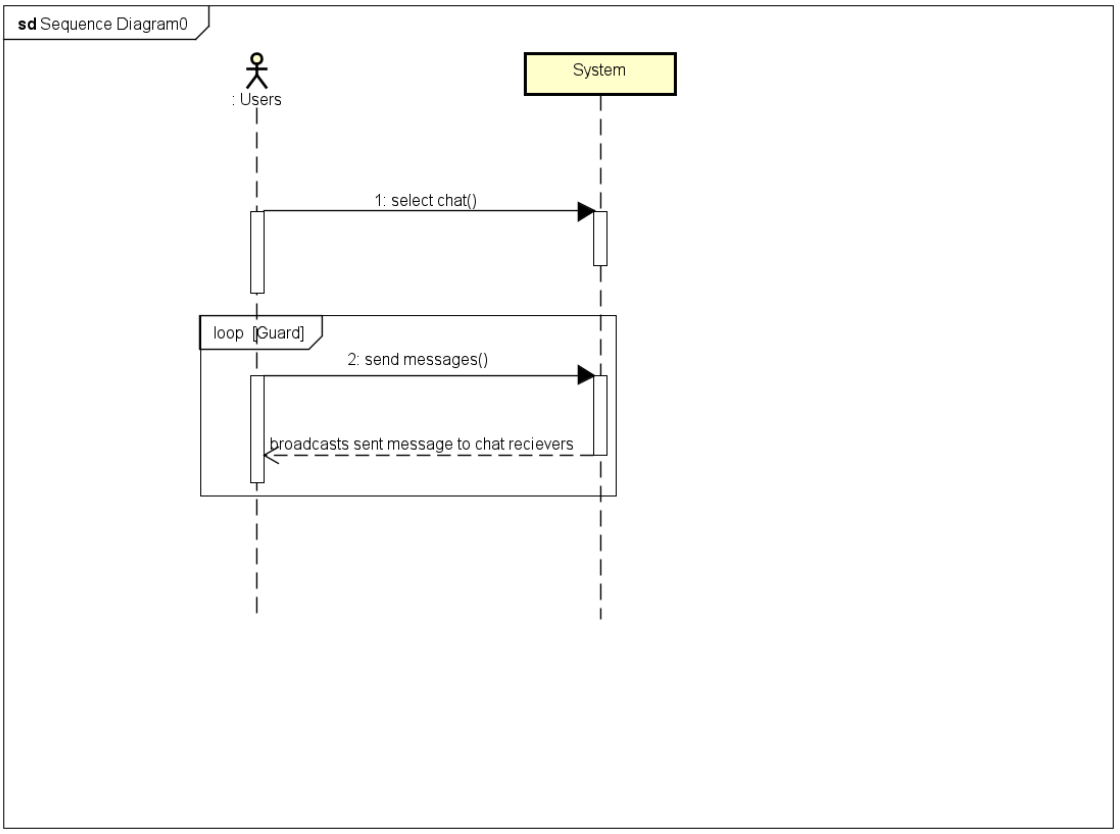
```
public interface ClientCallback extends Remote {

    1 implementation  radoslavKiryazov
    void receive(Message message) throws RemoteException;

    1 usage  1 implementation  Bozhidar Manev
    void onNewChat(Chat chat) throws RemoteException;

    1 implementation  Bozhidar Manev
    void changeColour(Chat oldChat, Chat newChat) throws RemoteException;
}
```

3.3 System sequence diagrams:

Manage chat Main System Sequence Diagram

This sequence diagram shows the main objective of the manage chat use case.

The main objective is to send messages and this is done through the following sequence:

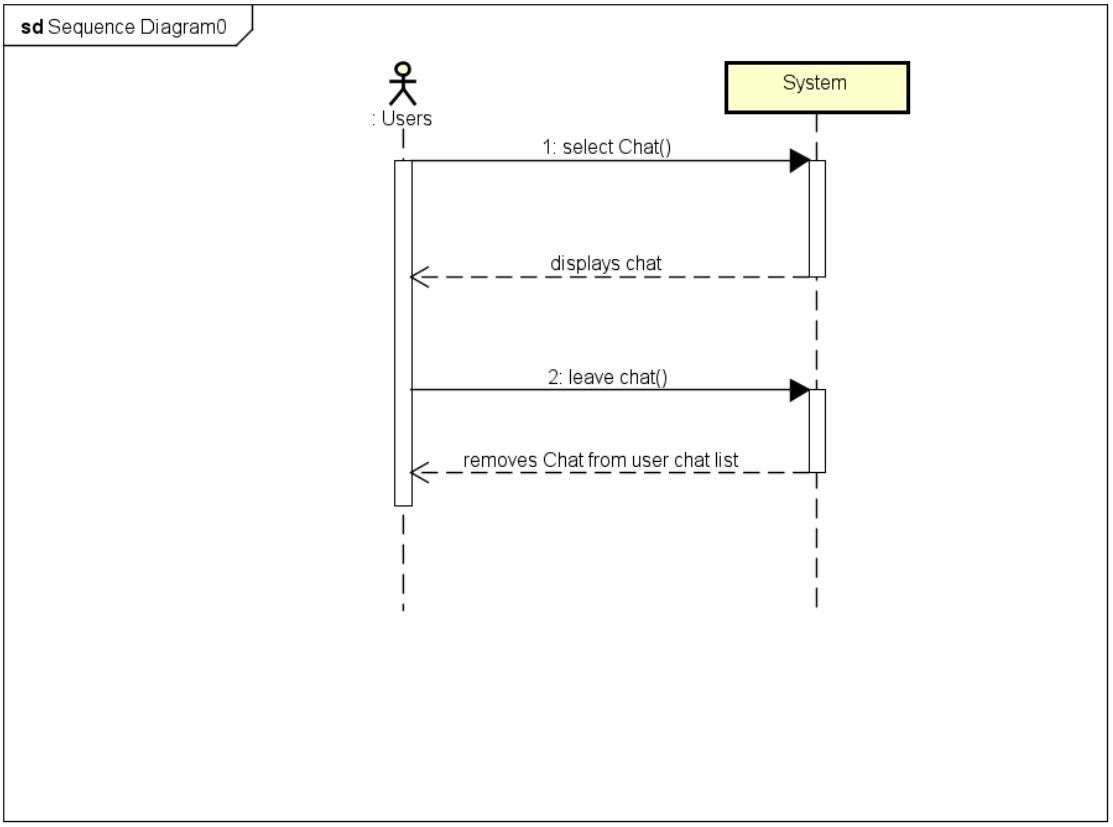


The user selects a chat. afterward they can send as many messages as they want to the selected user, and the system will broadcast the sent message to both users.

Manage Chat Delete Chat System Sequence Diagram

This sequence diagram shows an extension of the Manage chat use case.

The main objective of this sequence is to show how a user is able to leave a chat

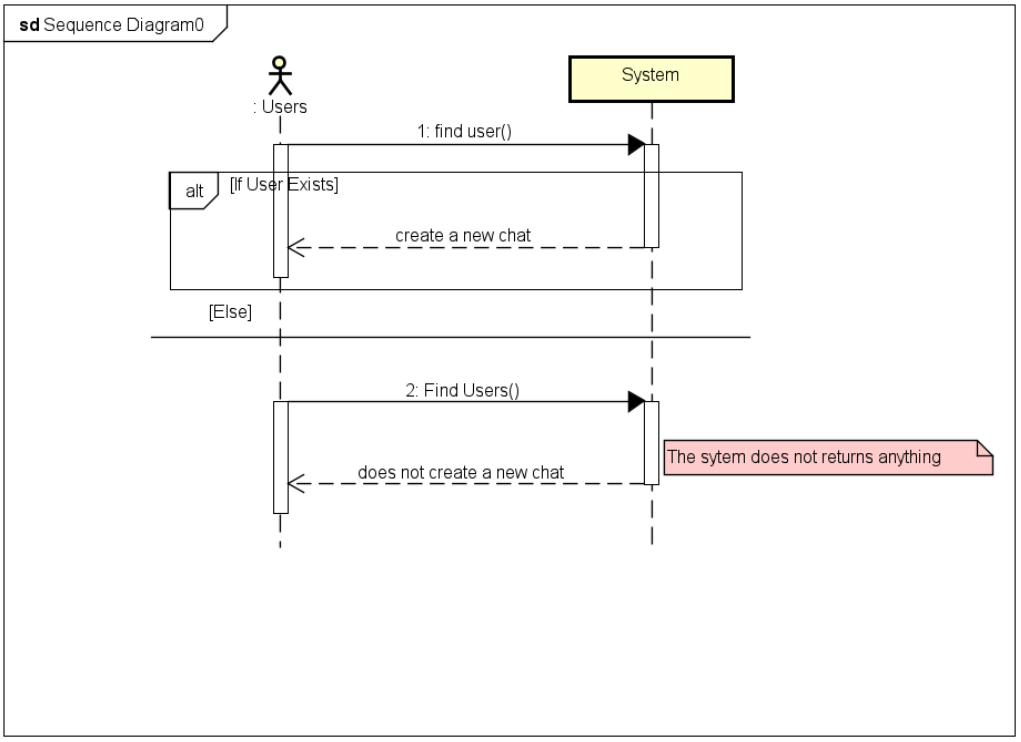


The user first selects a chat, and the system displays their previously sent messages with the selected chat.

where then they can choose to leave the chat which removes them from the chosen chat

Manage Chat Find Users System Sequence Diagram:

The following sequence shows how the searching for other users is accomplished in the system:

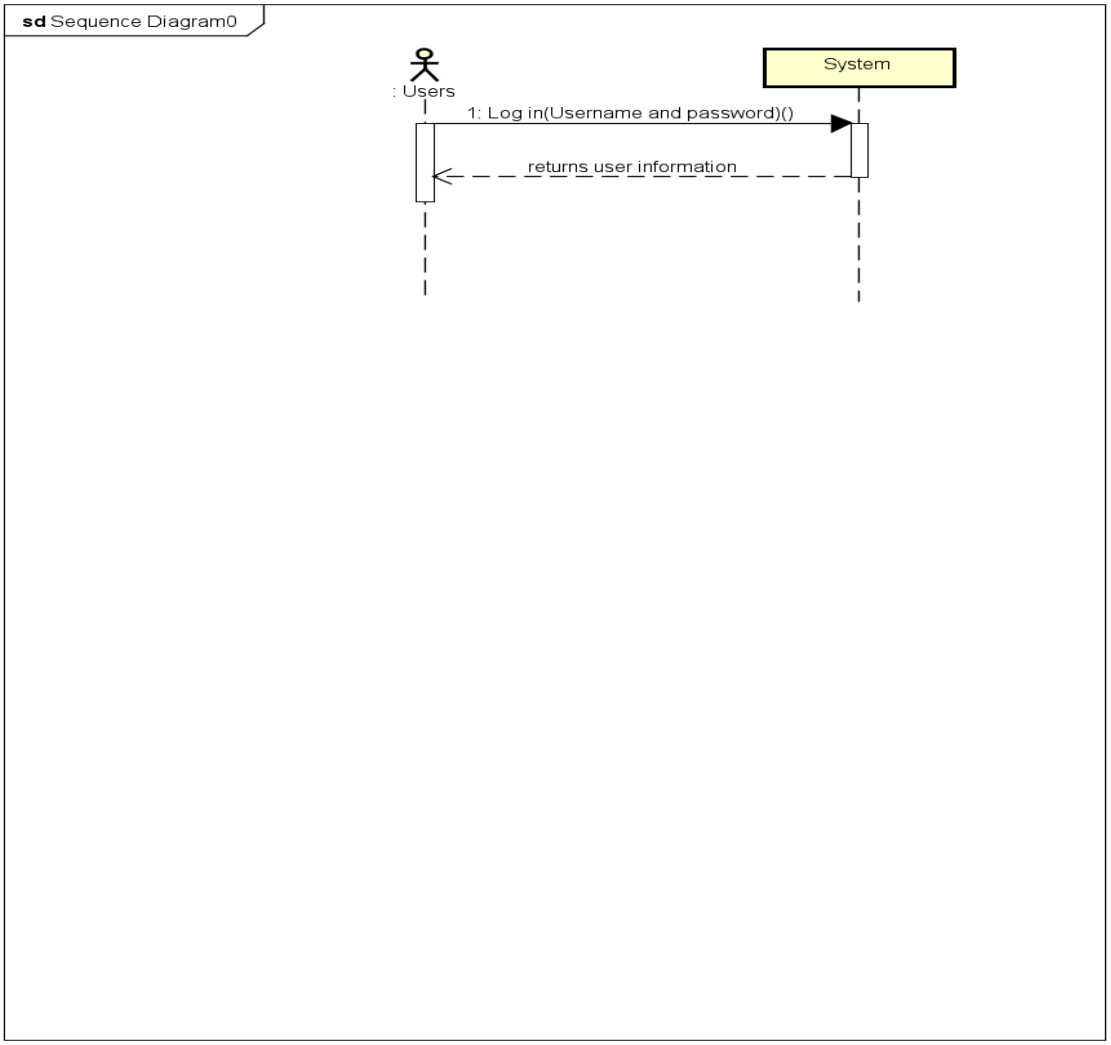


The user types in the name of the username they want to contact, and the system creates a chat between the user and their contact.

However, if the username does not exist in the database, then nothing will occur.

Manage Profile Main System Sequence Diagram:

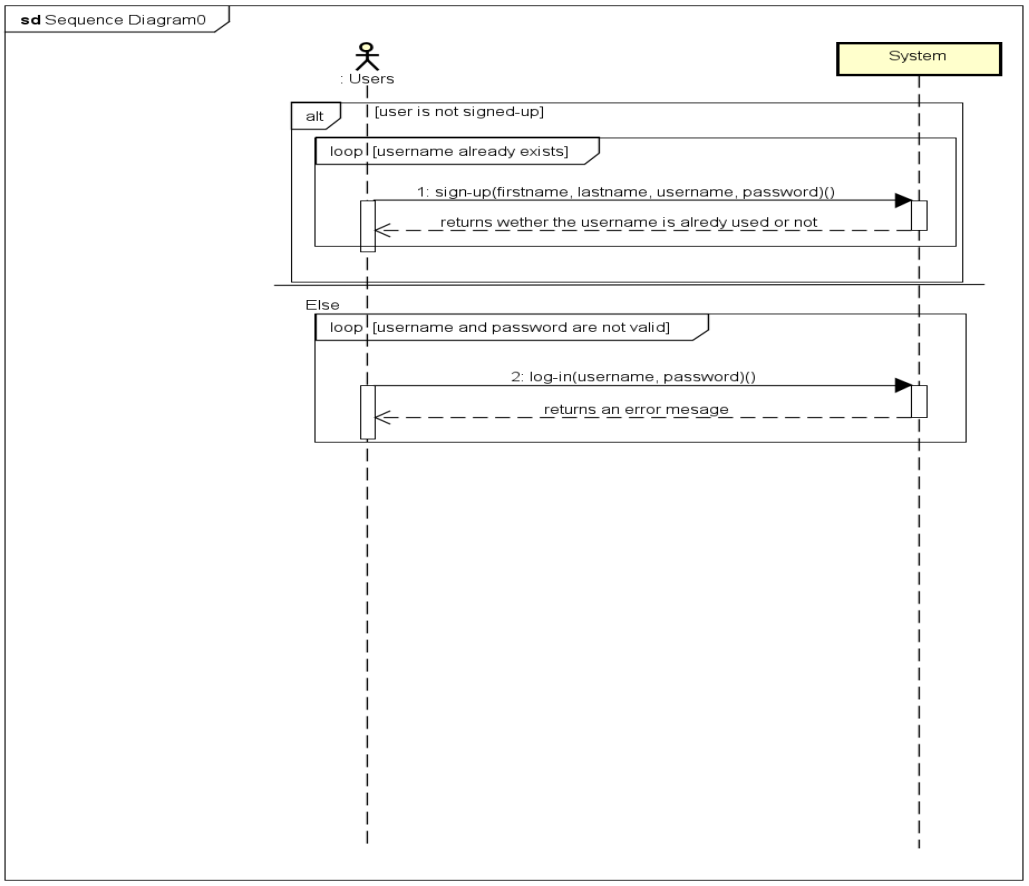
The following sequence shows the main purpose of the manage profile use case:



The sequence of this operation starts with the user entering their username and password, and the system will connect them to their profile.

Manage Profile registration System Sequence Diagram:

The following sequence is an extension of the Manage Profile use case. The focus of this sequence is to show the process of signing up to the system:



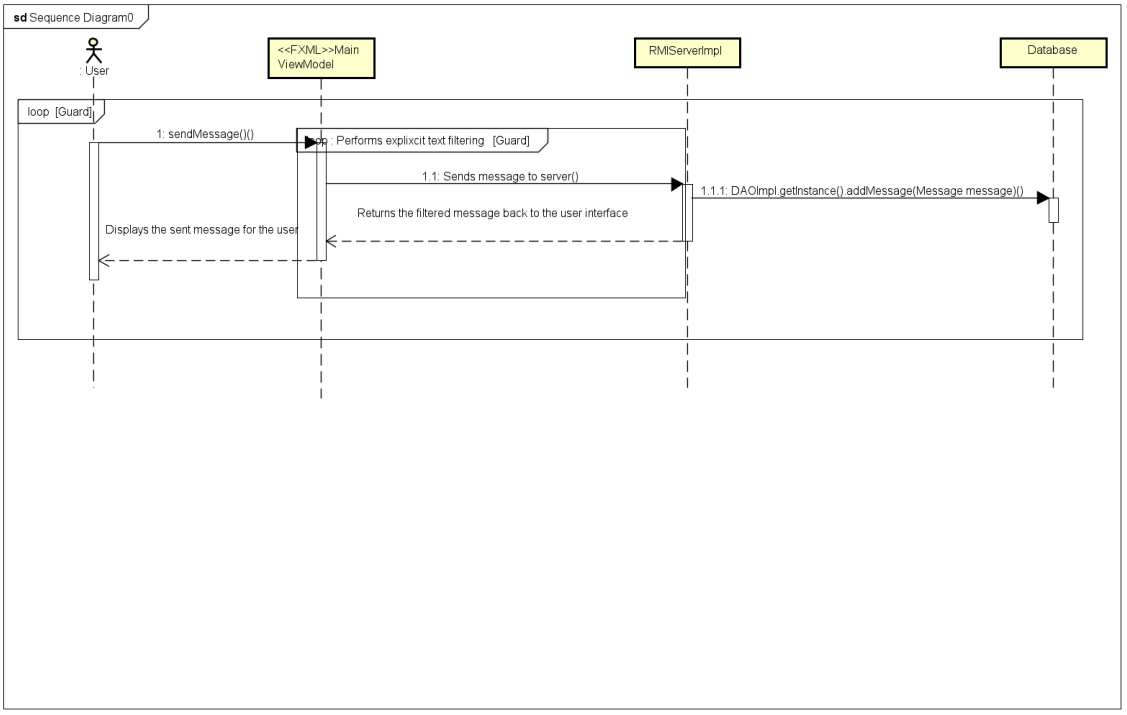
The user will sign up by entering a first name,last name, username and password and the system will create a profile for them.

If the username entered already exists in the database, then the user will be asked to enter a different one.

Otherwise, the user may log-in with their username and password, and if the entered username or password were incorrect, the system will display an error message.

3.4 Sequence diagram:

The sequence diagram below shows how one of the main scenarios for manage chat use case functions:



The sequence starts the user writing a message in the text field section of the MainViewModel. Afterward the user interface sends the message to the RMIServer.

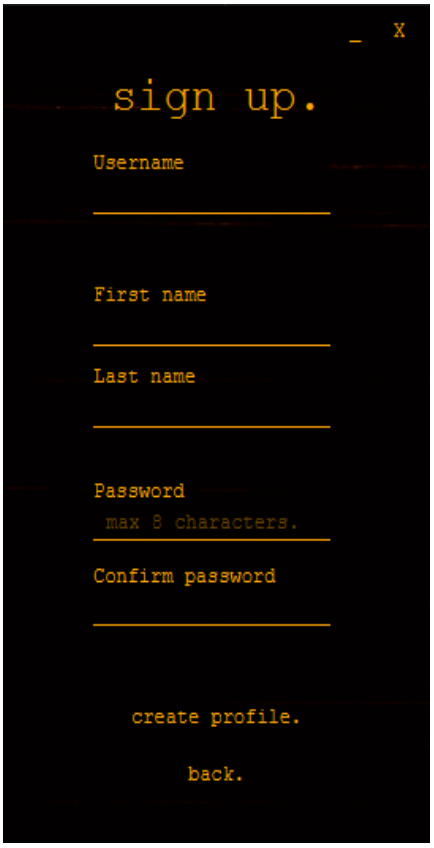
the RMIServer has two functions:

1. It stores the message in the Database
2. It checks for explicit in every sent message after replacing the explicits with asterisks, it returns the message to the user interface.

The MainViewModel then displays the message for both the users(the sender, and the receiver).

This sequence is done in a loop so the user can write and send as many messages as they want.

4. Implementation



4.1 Creating a profile

The sign - up GUI is fairly simple. It has text fields for every piece of information necessary to create a profile in our system.

Upon trying to click ‘create profile.’ the program will execute several checks. If it encounters any invalid username, empty fields, unmatchin passwords or a password bigger than 8 characters is going to create a custom popup and display it to the user.

By far the most important check is the presence of the username.

```java
public boolean usernameAvailability(String username) {
    boolean isAvailable = false;
    try {
        if (DAOImpl.getInstance().getUsernames().contains(username)) {
            isAvailable = true;
        }
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
    return isAvailable;
}
```

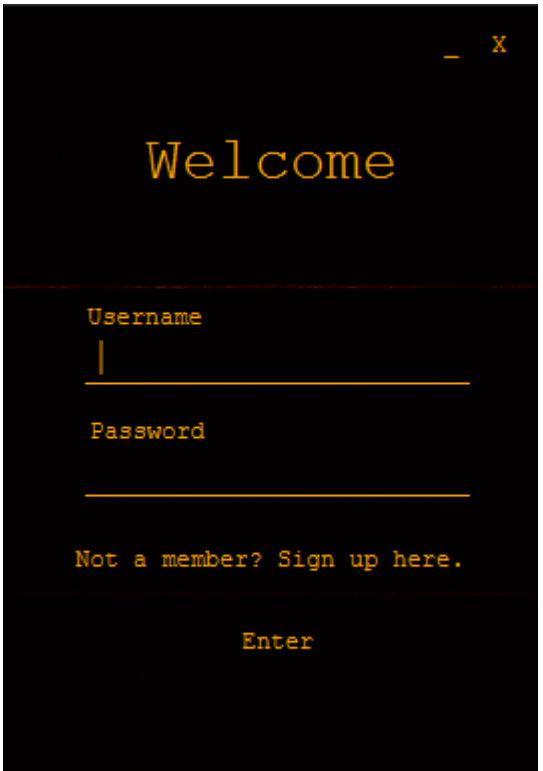
This happens by chaining methods transferring the text from the username field to the Model of the Server.

This will execute a query which will return a list of all of all the usernames and is going to check if it is part of the list, if it is not is going to allow the profile creation.

```java
@Override public List<String> getUsernames() throws SQLException
{
  ArrayList<String> userNames = new ArrayList<>();
  try (Connection connection = getConnection())
  {
    PreparedStatement statement = connection.prepareStatement(
        sql: "SELECT * FROM user_table");
    ResultSet resultSet = statement.executeQuery();
    while (resultSet.next())
    {
      userNames.add(resultSet.getString( columnLabel: "username"));
      System.out.println(userNames);
    }

  }
  return userNames;
}
```

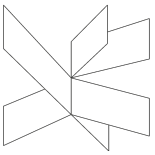
Signing In

The Sign-Up GUI has a TextField for the username and a PasswordField for the password, sign-in button and a button leading you to the Sign-up Stage.



When trying to sign-in, checks for the fields are executed again, denying entry if it finds any fields.

We've opted out of checking for the existence of the username given to us by our user. Instead the username and the password are collected to the GUI and sent down to the server to be processed.



```
public void onEnterButton() {
    if (usernameField.getText().isEmpty() || passwordField.getText().isEmpty()) {
        viewHandler.openAnAlertBox( messageToUser: "Empty Fields", title: " ");
    } else if (viewModel.getPassword(usernameField.getText(), passwordField.getText())) {
        viewModel.changeUsername(usernameField.getText());
        viewHandler.openChatView();
    } else {
        viewHandler.openAnAlertBox( messageToUser: "Invalid Credentials", title: "Login Denied");
    }

}
```

The Model in the server is going to contact the DAO to execute a query fetching the password for a certain username and send that password back to the server.

The server is going to perform a check between the two passwords and return a

```
@Override
public String getPassword(String username) {
    try {
        return DAOImpl.getInstance().getPassword(username);
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

```
@Override public String getPassword(String username) throws SQLException
{
  String password = " ";
  try (Connection connection = getConnection())
  {
    PreparedStatement statement = connection.prepareStatement(
        sql: "SELECT * FROM  user_table where username = ?");
    statement.setString( parameterIndex: 1, username);
    ResultSet resultSet = statement.executeQuery();
    connection.close();
    if (resultSet.next())
    {
      password = resultSet.getString( columnLabel: "password");
    }
    statement.close();
  }
  return password;
}
```

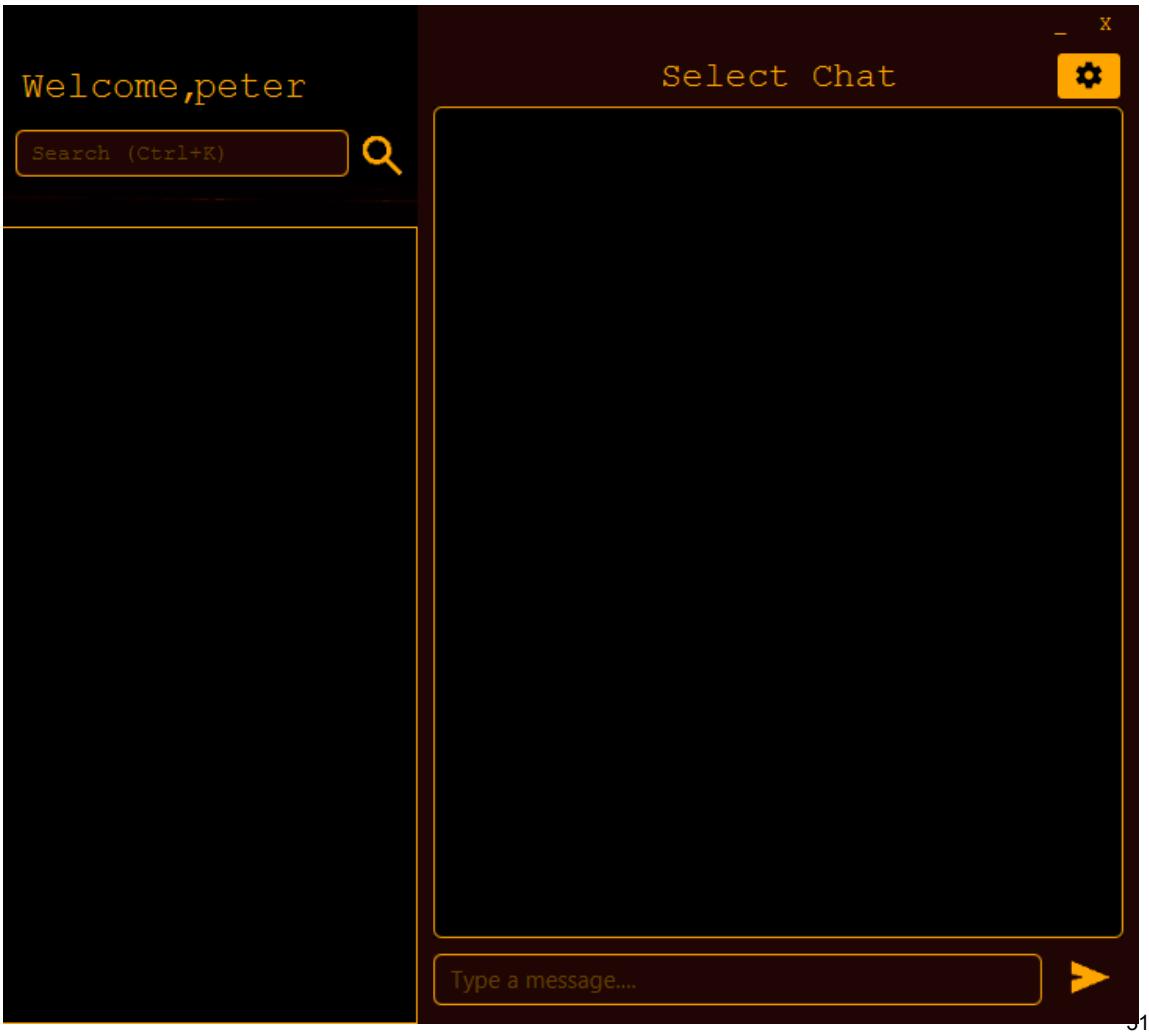
boolean to the user.

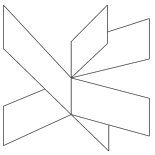
```java
public boolean getPassword(String username, String password) {
    return model.getPassword(username).equals(password);
}
```

We return a boolean, to prevent the user from getting the password of a random username he/she may decide to input to the system, sent to them locally.

4.3 Chat System

The Chat GUI is a more complex story. It has two ListViews. Chats on the left side on the screen and messages on the right. Both of them are connected to Observable Lists in the Controller of the View.





```
chats.setItems(mainViewModel.chats());
messagesList.setItems(mainViewModel.messageReceived());
```

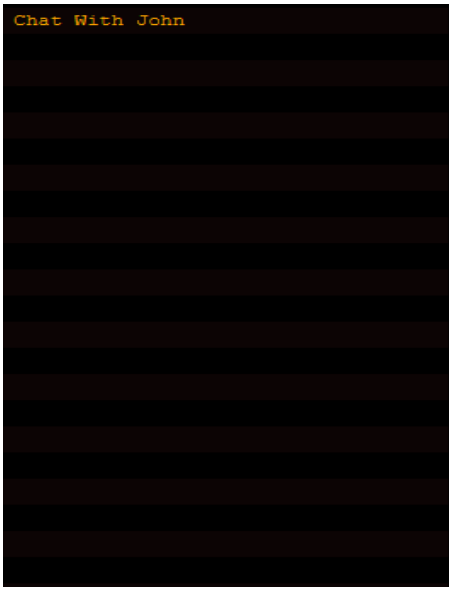
```
currentConversation = FXCollections.observableArrayList();
chatCatalog = FXCollections.observableArrayList();
```

Upon adding a user a method startChatWith is executed.

```
public void startChatWith(String username, String chatName) {
    System.out.println("MainViewModel: " + username);
    Chat chat = model.startChatWith(username, chatName);
    chatCatalog.add(chat);
    chatMap.put(chat.getId(), new ArrayList<>());
}
```

It creates a new instance of a Chat passing the username and the chat group as parameters to the server to execute a query, after that it adds it to the observableArrayList updating the GUI in the process. In the end it adds the chat to our map which has the chat id as the key pointing to an ArrayList.

```
public Chat createChat(String name) throws SQLException {
    Chat chat = null;
    try (Connection connection = getConnection()) {
        PreparedStatement statement = connection.prepareStatement(
                sql: "INSERT INTO chat_table(name) VALUES (?) returning *;");
        statement.setString( parameterIndex: 1, name);
        ResultSet set = statement.executeQuery();

        if (set.next()) {
            int id = set.getInt( columnIndex: 1);
            addMessage(new Message( sender: "@server@", id, messageBody: "Chat started"));
            chat = new Chat(id, set.getString( columnIndex: 2));
        }
        statement.close();
    }
    return chat;
}
```



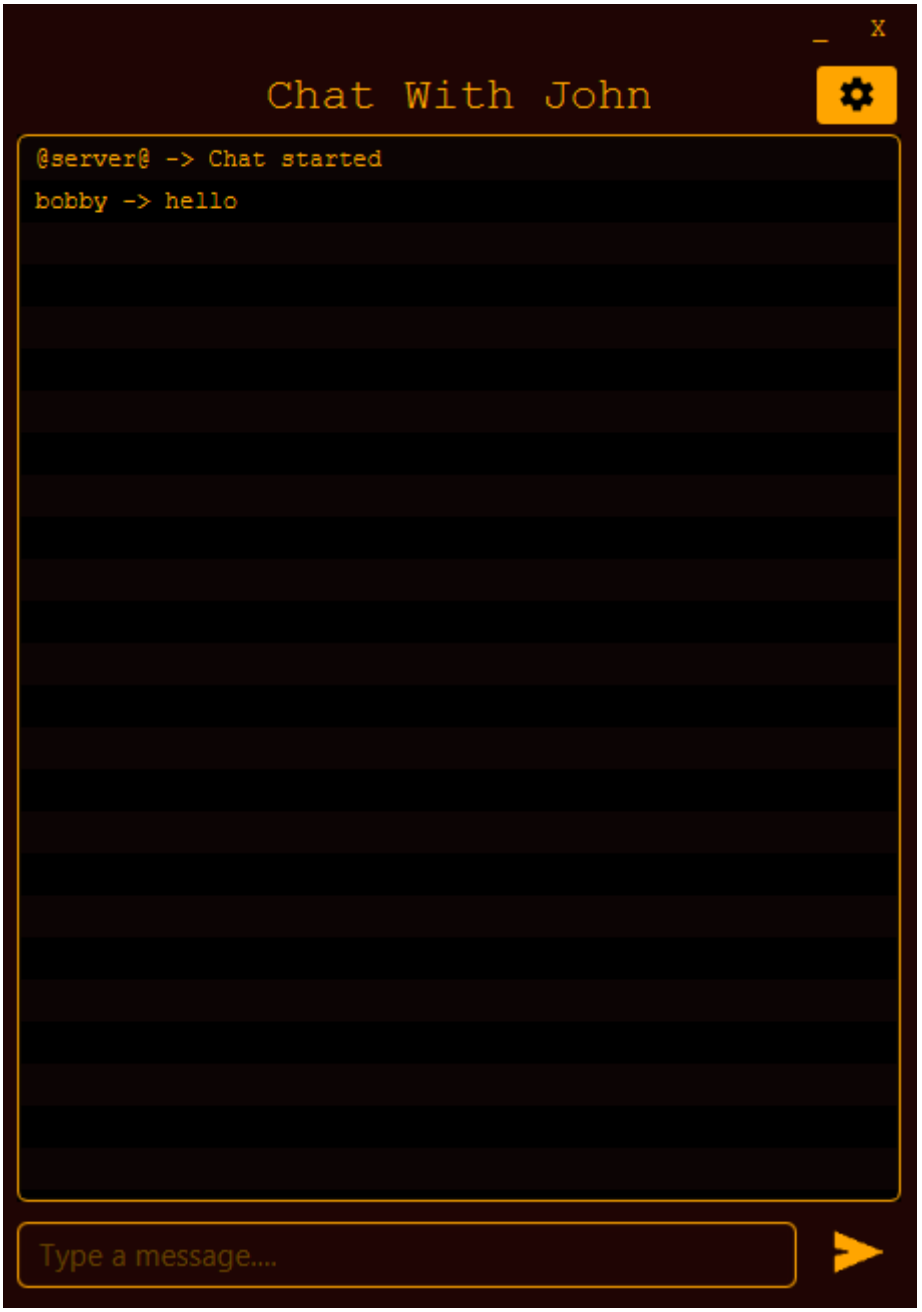
The messages will be re-loaded into the second ListView everytime a different item from the ListView is selected.

```
public void switchChat(MouseEvent mouseEvent) {
    if (!messagesList.isVisible()) {
        setVisibility(true);
    }

    Chat selectedItem = chats.getSelectionModel().getSelectedItem();
    if (!mainViewModel.isInSameChat(selectedItem)) {
        mainViewModel.switchChat(selectedItem);
        chatNameLabel.setText(selectedItem.getName());
        System.out.println("id " + selectedItem.getId());
    }
}
```

This will execute a query which will fetch all of the messages from the database related to the current chat id.

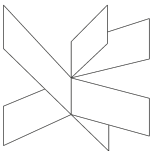
```
public void switchChat(Chat chat) {
    currentConversation.clear();
    currentConversation.addAll(chatMap.get(chat.getId()));
    currentlyOpenedChat = chat;
    chatTitle.setValue(currentlyOpenedChat.getName());
}
```

```
@Override public List<Message> loadMessages(String username)
{
  List<Message> list = new ArrayList<>();
  try (Connection connection = getConnection())
  {
    PreparedStatement statement = connection.prepareStatement(
        sql: "SELECT * from message_table where toChat in (select chat from receiver_table where username = ?)");
    statement.setString( parameterIndex: 1, username);
    ResultSet resultSet = statement.executeQuery();
    while (resultSet.next())
    {
      String sender = resultSet.getString( columnLabel: "sender");
      int toChat = resultSet.getInt( columnLabel: "toChat");
      String text = resultSet.getString( columnLabel: "text");
      String date = resultSet.getString( columnLabel: "datetime");
      list.add(new Message(sender, toChat, text, date));
    }
  }
  catch (SQLException e)
  {
    throw new RuntimeException(e);
  }
  return list;
}
```



When sending a message, a method will be called in the MainViewModel.

```
public void onSendButton() {
    mainViewModel.sendMessage(sendTextField.getText());
    sendTextField.clear();
}
```



```
public void sendMessage(String text) {
    if (text != null && !text.equals("")) {
        model.send(new Message(user.getUsername(), currentlyOpenedChat.getId(), text));
    }
}
```

As long as the message body is not empty, this method will create a new Message and pass the username of the sender, the chatid and the body of the message. This will flow down to the server and eventually to the DAO.

```
@Override
public Message addMessage(Message message) throws SQLException {
    try (Connection connection = getConnection()) {
        PreparedStatement statement = connection.prepareStatement(
                sql: "INSERT INTO message_table(sender,toChat,text,datetime) VALUES (?,?,?,?);");
        statement.setString( parameterIndex: 1, message.getSender());
        statement.setInt( parameterIndex: 2, message.getToChat());
        statement.setString( parameterIndex: 3, message.getMessageBody());
        statement.setString( parameterIndex: 4, message.getDateTime());
        statement.executeUpdate();
        statement.close();
    }
    return message;
}
```

4.4 Solid principles

S-(Single Responsibility Principle)

O-(Open-Close Principle)

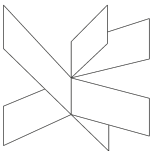
L-(Liskov Substitution Principle)

I-(Interface Segregation Principle)

D-(Dependency Inversion Principle)

Single Responsibility Principle - During the span of our project, the group tried our best to avoid any violation of the principles. Unfortunately, the group violated the Single Responsibility Principle by saving the information in the GUI. The group managed to come up with a solution utilizing the Proxy Pattern, but unfortunately time wasn't in the group's corner and that was left unimplemented.

Open-Close Principle - Violation was avoided by using interfaces, creating an expandable system.



Liskov Substitution Principle - Violation was avoided.

Interface Segregation Principle - Violation was avoided.

Dependency Inversion Principle - Violation was avoided.

5. Test

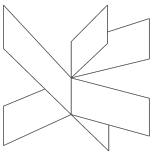
5.1- Testing use cases:

below are the testings of all the use cases, they include all the possible scenarios and their outcome:

Use case: Manage chat.

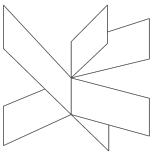
Manage chat: Sending messages

Step	Action	Reaction	Result
1	Open a chat	The GUI displays the content of the chat	Passed
2	Write a message and send it	The System sends a text to the user on the other end of the opened chat	Passed
3	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



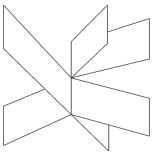
Manage chat: Establishing a new chat

Step	Action	Reaction	Result
1	Search for user	Connects to the user	Passed
2	Create a new chat	A connection is created between the user and the searched user.	Passed
3	Write a message and send it	A message is sent and added to the connection.	Passed
4	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



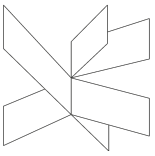
Manage chat: leaving a chat

Step	Action	Reaction	Result
1	Select a chat	The GUI displays the content of the chat	Passed
2	User removes themself from chat	The user is removed from the chat	Passed
3	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



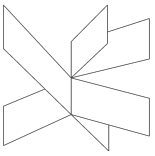
Manage chat: Customization in chat

Step	Action	Reaction	Result
1	Select a chat	The GUI displays the previous chats	Passed
2	User changes the color of the chat window	The chat background can be changed to certain preset colors.	Failed to implement a contact list in the system
3	User changes the font of the text in the sending messages text field.	The GUI chat increases or decreases its font size according to the user's request.	Failed to implement a contact list in the system
4	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



Manage chat: Request Chat log

Step	Action	Reaction	Result
1	Open chat log	The GUI displays the previous chats	Passed
2	Request Chat log	The system displays the requested chat log	Passed
3	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



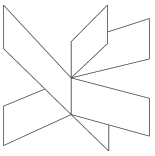
Manage chat: Create a group chat

Step	Action	Reaction	Result
1	Create a new chat	A new empty chat is created	Passed
2	Add multiple users to the chat	Multiple Users can now access the newly created chat	Passed
3	Leave the group chat	The User is properly detached from the server avoiding exceptions.	Passed

Use case: Manage profile.

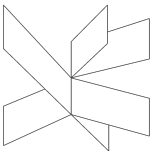
Manage Profile: Log in to the system.

Step	Action	Reaction	Result
1	Enter username, and password		Passed
2	User logs in the system	The user is allowed to enter into the system	Passed
2	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



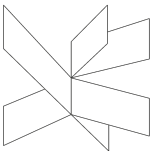
Manage Profile: sign up in the system.

Step	Action	Reaction	Result
1	User enters first name, last name, username and password		Passed
3	The user registers in the system.	The user is added into in the system	Passed
3	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



Manage Profile: sign up in the system alternative flow:

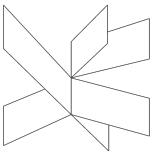
Step	Action	Reaction	Result
1	User enters first name, last name, username and password		Passed
2	Username already exists in the system.	The system denies the sign up request.	Passed
3	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



Manage Profile: Adding, deleting and blocking users from contact list.

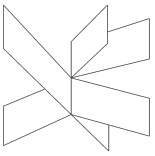
1. Adding

Step	Action	Reaction	Result
1	User searches for other users		Passed
2	User adds other users to their contact list	GUI is updated with the user added into the connections list.	The system does not add other users to the contact list. This feature was not implemented because the focus was on creating a system that serves more prioritized features that better serves the user.
3	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



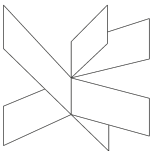
2. Deleting

Step	Action	Reaction	Result
1	User selects an added contact in their contact list		Failed to implement a contact list in the system, for it was deemed unnecessary.
2	User deletes other users from their contact list		Failed to implement a contact list in the system, for it was deemed unnecessary. However, the system can delete chat logs that were established between two users.
3	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



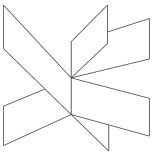
3. Blocking

Step	Action	Reaction	Result
1	User selects an added contact in their contact list		Failed
2	User blocks other users from their contact list		Failed
3	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



Manage Profile: Customizing profiles.

Step	Action	Reaction	Result
1	Users change their first and last name, username, and password.		Passed
2	User changes the theme of their profile	The changes are registered and saved in the system	Failed
3	User leaves the system	The User is properly detached from the server avoiding exceptions.	Passed



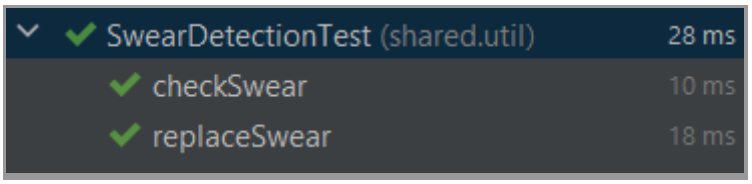
5.2. Junit testing:

JUnit was used to experiment with the methods created in the project, usually to test for both ideal and non-ideal conditions to see how a certain method can be modified to handle exceptions in a way so that the errors may be avoided.

Below, we check the Profanity filter feature which is the SwearDetection class for various possible ways that users could potentially evade the filter by testing using an assertTrue such as a large String or maybe even changing the case of each character in the profanity.
Also testing if the class is successfully able to change inputs from the user to censor crude language and return a more appropriate message.

```
@Test public void checkSwear()
{
  SwearDetection swearDetection = new SwearDetection();

  assertTrue(swearDetection.checkSwear( word: "crap")); //Checking single String

  assertTrue(swearDetection.checkSwear(
       word: "thats a bunch of crap")); //Checking String with multiple words separated by spaces

  assertTrue(swearDetection.checkSwear(
       word: "thats a bunch of cRaP")); //Checking string with different case letters


}

@Test public void replaceSwear()
{
  SwearDetection swearDetection = new SwearDetection();

  assertEquals( expected: "c***", swearDetection.changeSwearMap(
       word: "crap")); //Checking that method replaces profanity

  assertEquals( expected: "thats a bunch of c***", swearDetection.changeSwearMap(
       word: "thats a bunch of crap")); //Checking that profanity filter filters String separated by spaces

  assertEquals( expected: "thats a f*** bunch of c***", swearDetection.changeSwearMap(
       word: "thats a fucking bunch of crap"));//Checking that multiple profanities are filtered in a single String

  assertEquals( expected: "thats a bunch of c***", swearDetection.changeSwearMap(
       word: "thats a bunch of CrAp")); //Checking that due to different case per character filter still detects swears

}
```

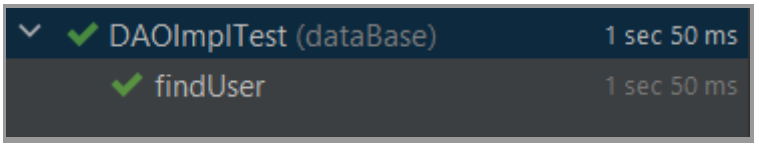
All tests were successful as shown below

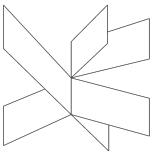


Below we test the Data Access Object class for finding a user which is essential for sending messages to the right user with 2 examples to check for a certain user

```
@Test public void findUser() throws SQLException
{

  User user = new User( username: "test1",  firstname: "test1",  lastname: "test1",  password: "test1");
  User user1 = new User( username: "kro",  firstname: "test",  lastname: "test",  password: "test");

  assertEquals(user.toString(), DAOImpl.getInstance().findUser( name: "test1")
      .toString()); //Checking that data access object class finds required user

  assertEquals(user1.toString(),
      DAOImpl.getInstance().findUser( name: "kro").toString());

  assertThrows(java.lang.NullPointerException.class,
      () -> DAOImpl.getInstance()
          .create( firstName: null,  lastName: null,  username: null,  password: null)); //Cannot add null user

}
```

Here we get a success scenario



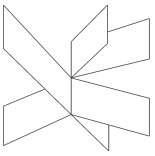


Results and Discussion

Overall, the group delivered a good program, by our standards. Everyone tried their best at their job to apply the knowledge gained all throughout the semester. Each sprint was carefully planned according to the SCRUM methodology, and with each sprint review and retrospective meeting new ways of working were taught of, some worked, some didn't, all in the spirit of the agile principles.

Unfortunately, as mentioned in the design part of the document, there was one violated SOLID principle in the face of the single responsibility principle, this could have been avoided if given enough time.

In the end, every critical and high level requirement was successfully implemented and tested,although leaving a few low level requirements unfulfilled.



Conclusions

In conclusion, in the Analysis segment of the report, the functional requirements were listed by priorities as well as containing the analysis diagrams with brief descriptions for each diagram showing the group's examination of the topic and based analysis on.

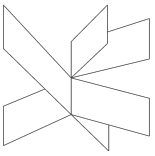
The design part talks about the different types of design and architectural patterns along with system sequence diagrams that have been used in order to create the chat system.

The Implementation segment mentions interesting code snippets that visualize the main Success scenario such as Creating a profile, Signing in and sending and receiving messages.

Tests segment consists of different test use cases and their final results also with snippets of JUnit testing.

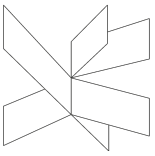
And finally the results and discussion segment is the group's final thoughts on the project with frameworks used and design principles.

In conclusion the group has not implemented all the features but fulfilled the essential requirements.



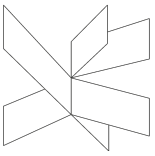
Project future

In hindsight, the group would have liked to implement a solution to the readers-writers problem faced when accessing the methods relating to the database to avoid any errors. Implementing the proxy patter to successfully Also the group could have implemented a state pattern in the project by maybe having a user which has different states such as being able both send and receive messages or only able to read messages.In addition more Unit testing could have been helpful instead of the basic testing done in the console/psvm.



Sources of information

- Craig Larman, 2004 : Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and Iterative Development,Third Edition
- Ken Schwaber, Jeff Sutherland, 2011:The Definitive Guide to Scrum:The Rules of the Game
- Samuel Oloruntoba, 2020: SOLID: The First 5 Principles of Object Oriented Design: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
- Santhakumar Raja, 2021: The Unified Process in Software Engineering: https://theteche.com/the-unified-process-in-software-engineering/



Appendices

The purpose of your appendices is to provide extra information to the expert reader.

List the appendices in order of mention.

Examples of appendices

Appendix 1: Project Description

Appendix 2:User Guide

Appendix 3:Source code – source documentation

Appendix 4: Domain Model

Appendix 5:ClassDiagram

Appendix 6: Manage Chat (Main System Sequence Diagram)

Appendix 7: Manage Chat (Find Users System Sequence Diagram)

Appendix 8: Manage Chat (Activity Diagram)

Appendix 9: Manage Chat (Delete Chat System Sequence Diagram)

Appendix 10: Use Case Diagram

Appendix 11: Manage Profile (Registration Sequence Diagram)

Appendix 12: Manage Chat Activity Diagram

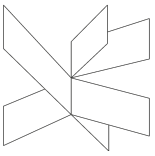
Appendix 13: JavaDocs

Appendix 14: Manage Profile (System Sequence Diagram)

Appendix 15:Manage Profile (Main System Sequence Diagram)

Appendix 16: Database Diagrams

Appendix 17: Database Logic



Appendix A Project Description

SEP2's Project Description

- **Ameya Mahankal. (326157)**
- **Bozhidar Manev. (326391)**
- **Joan Tammo. (325753)**
- **Radoslav Kiryazov. (325155)**
- **Zsolt Nóvé. (326345)**

Software Technology Engineering

SECOND SEMESTER

07/09/2022

Table of content

1. Background Description

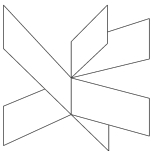
In today's world we find ourselves communicating more and more through online channels such as messaging, social media, and video calls (Dave,C.,2022.Global social media statistics research summary 2022). It often comes at the expense of supplying personal information to third-party advertising companies.

There are many advantages to online communication that deserve to be outlined. It opens up the possibility of immediate communication with people in different places around the globe. It provides the chance for families and friends to keep in touch on a daily basis. Furthermore, communication channels provide considerable benefits in the workplace as communication can take place via chat systems, which saves time, provides simplicity, and prevents long-haul business trips between cities across the world.

One of the most significant disadvantages of chat systems is the dramatic shift of the companies behind them towards a much greater level of intrusiveness in an effort to increase advertising effectiveness (A.Guttmann, 2022,Growth of advertising spending worldwide 2000-2024). Since it often is the primary source of revenue for many companies, it results in dishonest practices. Through these actions, the social giants are violating their users' internet privacy by tracking their online footprint and in-app actions (Harris.S.,2022.What is internet privacy and why does it matter so much in 2022?). Therefore, treating themselves to more user information than necessary for their system to function.

The addition of distracting features in chat systems is the main goal of the social giants to compel the users to spend more time in-app. Although more features are always better, companies are trying to have and eat their cake, by having a feature-packed product that is easy to use, and that is a balance that is hard to strike (S.Perez, 2022, Mobile users are now spending 4-5 hours per day in apps)
Although more features are always better, companies are trying to strike a hard balance between a product that is feature-packed and easy to use (S.Perez, 2022, Mobile users are now spending 4-5 hours per day in apps)

By focusing on monetizing, social giants loosen their grip on the safe environment of their products. This leads to some users exercising malicious activities, which can prove harmful to other users' well-being. At least users in most cases are able to block others from exercising such activities.



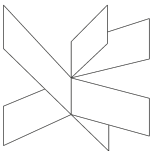
2. Problem Statement

Problem statement:

Proprietors of chat systems often focus completely on monetizing their products by exploiting users' private data, which is a controversial topic in today's day and age.

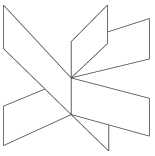
Sub Problems:

1. What is the minimum amount of information that can be collected without intruding on the user's privacy?
2. How to find the perfect balance between a system that is feature-packed, but easy to use?
3. How can people protect themselves from malicious activities?



3. Definition of purpose

The purpose is to help facilitate communication between people around the world, with a communication tool that collects only the necessary information for it to function, while keeping it easy to navigate.



4. Delimitation

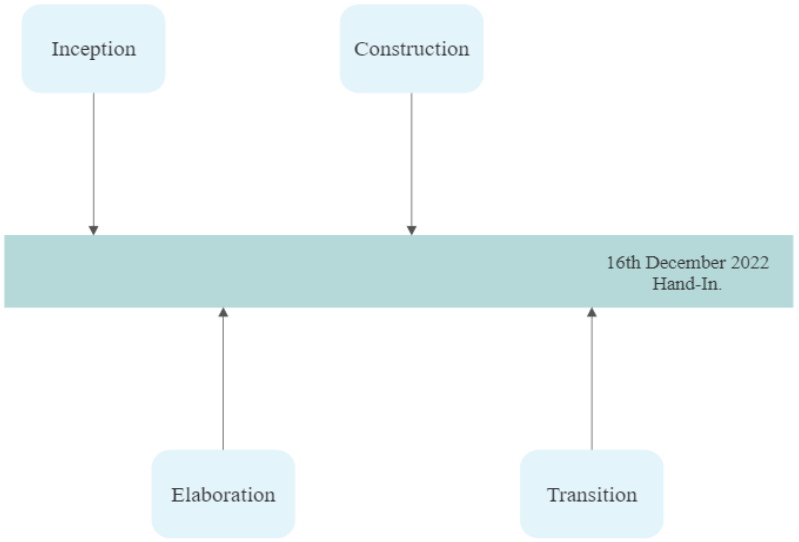
1. We will not include features that deviate away from the main product purpose.
2. We will not include advertisements.
3. We will not require more information from the users than the one necessary for log-in identification.

5. Methodology

The project will implement Scrum - an Agile methodology established as one of the most successful and productive (Clair,D..Scrum.Learn how to scrum with the best of them). In addition, The Unified Process framework will be used because of its iterative and incremental software development process. Furthermore, daily meetings will be used to keep track of members and their work. Part of Scrum is the two roles - Product Owner, who determines whether a goal is complete, and a Scrum Master who selects user stories for the next sprint. The Rational Unified Process defines nine disciplines: Business Modelling, Requirements, Analysis and Design, Implementation, Test, Deployment, Configuration and Change Management, Project Management, and Environment. The project is divided into phases: Inception, Elaboration, Construction, and Transition.
Each sprint, performed during the project will feature a rotation of the Product Owner and Scrum Master roles. This way each member of the team can experience the responsibilities of each role, and understand Scrum and Unified Process on a deeper level. A Burndown chart is going to be created at the end of the first sprint session and is going to be updated at the end of each sprint to accurately represent the work that has been done, and what is still left to be implemented

6. Time schedule

Each sprint will last 3 days, where a sprint review and planning of the next sprint session will be held at the end of the last sprint day.



Inception.

Approximate hours spent: 125h

Elaboration.

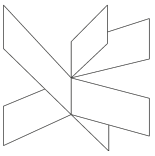
Approximate hours spent: 325h

Construction.

Approximate hours spent. 450h

Transition.

Approximate hours spent: 230h



7. Risk assessment

Risks	Likelihood Scale: 1-5 5 = high-risk	Severity Scale: 1-5 5 = high-risk	Product of likelihood and severity	Risk mitigation e.g. Preventive- & Responsive actions	Identifiers	Responsible
Users are unable to create a chatroom with multiple users.	3	4	12	Having a one hundred percent working 1on1 chat before continuing development.	Program crashes or displays errors when creation is attempted.	Group Member
Bad coordination due to members working/traveling.	3	3	9	Consistent flow of communication between team members.	Missing deadlines,	Group Member
Prolonged Development Time	2	5	10	Team concentrates on developing the product, leading to major neglect of the documentation and vice versa	Majority of documentation is missing or unfinished, close to the deadline.	Group Member
Loss of chat log data	2	4	8	Creating multiple backups of chat logs	Failing to find/connect database	
Ambitious scope	3	5	15	Meeting with instructors to ensure scope is maintained	Incomplete features,behind schedule	

8. Sources of Information

Dave,C.,2022.Global social media statistics research summary 2022.[blog]22 August, Available at:

https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/

A.Guttmann, 2022,Growth of advertising spending worldwide 2000-2024, Available at:

https://www.statista.com/statistics/272443/growth-of-advertising-spending-worldwide

Harris.S.,2022.What is internet privacy and why does it matter so much in 2022?[blog]24 August, Available at:

https://www.purevpn.com/blog/what-is-internet-privacy-scty

S.Perez, 2022, Mobile users are now spending 4-5 hours per day in apps[article] 3 August, Available at

https://techcrunch.com/2022/08/03/mobile-users-now-spend-4-5-hours-per-day-in-apps-report-says/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAIwD1Cz0qLHF2ktj-fcyuikGpOoOab7QuS65CVL_O2IwpbyzB2AJQsIB8POA-UWfKJYtyEsQvihIV1Z5uHNonGAJYPpT4RghU48PfddSirg4lLmHgVAmP8EYKjcA1hoSujMgW-OiMBZofrEWAVUuaYXnQnynuYOXzzC4Ie-hQRWF

Clair,D..Scrum.Learn how to scrum with the best of them, Available at:

https://www.atlassian.com/agile/scrum

Craig Larman, 2004 : Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and Iterative Development,Third Edition

Jeremy, H.,2018,Russian company had access to Facebook user data through apps,Available at:

https://money.cnn.com/2018/07/10/technology/mailru-facebook-russia/index.html

Donie O'Sullivan, Drew Griffin and Curt Devine: 2018: CNN Business.

Anonymous, 2018, What are the benefits of chat and messaging?

Available at:

https://www.digitalcitizenship.nsw.edu.au/articles/what-are-the-benefits-of-chat-and-messaging#:~:text=Key%20message,and%20help%20each%20other%20out.

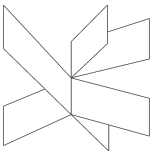
Lauri A. Jensen-Campbell,Rui Zhang, Samantha Heintzelman ,2021,Friendship Importance Around the World: Links to Cultural Factors, Health, and Well-Being

Available at:

https://www.frontiersin.org/articles/10.3389/fpsyg.2020.570839/full

Archna, O.,2019,Multilingual Chat: Overcome Language Barriers for a Hassle Free Customer Support,Available at:

https://insights.daffodilsw.com/blog/multilingual-chat-overcome-language-barriers-for-a-hassle-free-customer-support



Appendices