Software Design and Implementation

Lab tutor: Mr. Pedro Machado

Group Number: Group 40

Group Members: Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)

Roles Listing

Id	First Name	Surname	Role
1	**Pedro**	**Machado**	**Module Leader**
2	Bogyeong	Kim	Project Manager
3	Habeebullah	Ladan	Software Developer
4	Annamalai	Selvarajan	Software Tester
5	Oluwaseun	Banjo	Software Architect

Revision History

Version	Issue Date	Stage	Changes	Author
0.1	17/05/2021	1	- Added cover, abstract, introduction	Habeebullah
0.2	26/05/2021	2	- Added background research, SWOT analyses, requirements list, risk	Habeebullah

			assessment	
Final	11/06/2021	Final	Added the remaining sections	Habeebullah

Plagiarism Declaration

This report and the software it documents are the results of my work. Any contributions to the work by third parties, other than tutors, are stated clearly below this declaration. Should this statement prove to be untrue I recognize the right and duty of the Board of Examiners to take appropriate action in line with the university's regulations on assessment.

Abstract

This document discusses the steps and procedures in designing and implementing our messaging platform. It also states the various tools used in the designing of the platform which is listed in the "List of tools" below. We also discussed the risks, the requirements, the strengths, the weaknesses and how each task was allocated to each group member, and the timeline for each task to be completed.

We also used various diagrams while designing our messaging platform to make certain tasks easy to understand and provide better insight on those tasks. Diagrams such as Use Case diagrams, Class diagrams, Gant charts, Sequence diagrams, etc. as well as various coding styles for better insight on the structure of our messaging platform. Our messaging platform has two areas of focus which are the implementation and the testing, whereby the implementation sheds more light on the approaches used in the making of the platform and the testing describes the approaches used to check that the platform is free from errors. Furthermore, the findings and approaches used by each group member in the course of the project are reflected in the conclusion.

Contents

Introduction

Since the recent outbreak of the covid-19 virus, physical interaction between people has become increasingly difficult. This may have led to an increase in the use of social media for communication. This document provides an overview of a social media application designed by group 40. The social media platform just like any other will support sending and receiving of messages between two users connected by a single server.

The project will consist of four main stages: planning, design, implementation, and testing. Every group member will be given a major role in these stages and will be expected to facilitate the activities carried out by each member during that stage to ensure every deliverable produced meets the requirement as stated in the description of the coursework. It is also highly important to note how important the planning stage is as it lays a blueprint as to what footsteps the members of the group must follow to successfully design a robust and sophisticated application.

After the conclusion of the planning stage, the design stage plans are then initiated. This stage acts as a pillar for the entire project, where all the available tools are on display. With the use of images and diagrams, a structural overview of the project is provided. The use of various diagrams such as class diagrams, use case diagrams, deployment diagrams will provide a pictorial representation of the interaction between the application and the user and a breakdown of the. The implementation stage will then be carried by all group members mainly the software tester and the software developer. This stage may develop the most complications as it involves an enormous amount of code writing which will most likely develop bugs and errors. Fortunately, a software tester will carry out tests to ensure that no bugs or errors are developed during runtime.

Finally, the risk assessment approach is carried to ensure the program can still run successfully despite the limitations of the project. Some of the major possible that can occur in this project will be documented below on an assessment form. The form is designed taking into consideration individual abilities, the scope of work, due dates, resources available, deliverables, and more. After much consideration of the risk of the project, possible solutions and workarounds will be introduced to ensure major functionalities of the program are not compromised in any way. Such workarounds and solutions are also known as mitigation plans. This approach should ensure that the project is not only handled successfully but is set on track with minimal interruptions.

Background Research

Upon planning our application, selecting what IDE to use was very essential. Therefore, the group decided to go and make findings and we decided to use Qt Creator which was used in the implementation and design functionality of our application. Qt can be used on various platforms with the same development environment where it is used as well as provide greater control, and the possibility of working with numerous libraries when coding in C++.

Subsequently, research was made on the basic features of a message exchange platform. This was very useful in giving the group members an idea of the basic functionalities of a message exchange application. Furthermore, a comparison was made between message exchange platforms that had already been deployed such as WhatsApp and Slack, to check for their limitations. This comparison made it easy to identify other possible functionalities that could be implemented in the project application.

It is also worth mentioning that another IDE was used was Qt Designer which was used to create our various diagrams like our use case diagram, sequence diagrams, etc. It can also be used to create different styles and resolutions, create menus and toolbars, etc. It is also platform and language independent which is why our group chose it as our second IDE to work with. Below are the other tools which we also made use of in the designing of our application.

List of tools

Tools	Brief Explanation
Qt Creator	Used for implementing the design and functionality of the program.
Qt Designer	Used for the 2nd deliverable design of the application.
Doxygen	Used for generating code manually.
WhatsApp	Used for communication and sharing resources.
Microsoft Teams	Used to host meetings with the group members and course leader.
Microsoft Word	Used for documentation.
Microsoft Excel	Used for Gantt chart design and documentation of roles assigned to group members.
Papyrus	Used for structural design.
Draw. io	Used for structural design.
Google Drive	Used for sharing resources.
Zoom	Used to host meetings with the group members only.
Github (Olympuss)	Used for storing source code and documentation and act as a backup in an event of file loss.

[Table 1 – List of Tools]

SWOT Analyses

Strengths

- 4 group members are assigned where each stage of the project can be successfully facilitated by one group member.
- Communication among team members is easy. All team members are available for group meetings.
- A good amount of C++ knowledge among the group members.
- The size of the team allows the group leader to identify the group members carrying out an activity.

Weakness

- little knowledge of MQTT implementation.
- Time zone difference.
- Lack of sophisticated devices used to carry out tasks by some group members.
- Clashing of submission time with other assignments and coursework.

Opportunities

- Each group member gains the chance of improving his/her coding skills.
- New software applications such as papyrus introduce group members to better and more sophisticated applications for structural design.
- Improving teamwork and corporations are necessary in the real world.
- Improves time management skills.

Threat

- Members of a group may not have the same deadlines which may lead to late submissions.
  Switching of roles on short notice can affect the members of the group

Requirement Information

The program to be designed is a message exchange platform that allows interaction between users. This mini-application will implement C++ GUI for program design and easy navigation for users. The program will allow the user to send information to another user while the receiver can view the message sent and reply to it. With the use of c++ as the major language, it will make use of the slack platform as the overall blueprint for the design of the program. Below is the requirement list for the project.

Id	Requirement	Description	Implication	Task(s)
1	Must provide a friendly User Interface (UI)	It can use either a console-based or graphics-based interface to provide user-friendliness.	A graphical user interface will be implemented with the appropriate functionality for the users	QT designer will be used. Should make UI design in advance before creating friendly UI.
2	Users must only access their space after the login	Users will not have the ability to access their space until they input their login details successfully.	Securing user information and preventing access until the user is verified successfully.	Creating a database file where created usernames and encrypted passwords will be saved.
3	Password must be saved securely locally	The user encrypted password will be saved locally in a file for easy access when it is required.	To save user details for future access.	Creating a local file to store encrypted user passwords.
4	Clients must not connect directly to other clients without a server or	To communicate with other clients, a server must be used.	It supports the transmission of messages between clients.	Make use of MQTT to create a server that allows for communication between clients.

	a broker.			
5	A server or a broker must allow multiple authorized clients to connect to it	Following the slack de-platform, the interface should allow multiple users to connect at a time.	Two or more people will be able to connect to the platform. They will also be able to Communicate with each other.	Make use of a client-server setup that will enable multiple clients to connect to it at once.
6	The user must be able to create chat rooms	the user will be allowed to create a chat room where users can communicate and perform activities together.	Create chatrooms where users can add other users and specify the maximum number of users allowed.	Add a “create chatroom” button that will automatically create an environment for message transmission and file sharing.
7	Moderators must inherit all the admin permissions (however, Moderators cannot demote the Admin)	The moderators should give admin permission to add and remove members however, added users will not be able to remove admin.	This will give an admin the ability to control a chatroom and the members of the chat room.	Will use c++ boost libraries to create moderators which will be able to give admin secondary manipulation of a chatroom.
8	Users must be able to see the active users in the chat	The user will be capable of viewing actively connected users in the chat room.	Grant users access to view chat room members.	The number of connected members will be displayed at the top of the platform and a “view members” button will be created that will allow a user to

	room			view all connected members.
9	Must use classes	To realize MVC, at least one class where private member functions for common functionality that should not be exposed in the public interface.	The use of classes will make the program more versatile and easier to understand.	Make use of classes in object-oriented programming. Different classes will provide different values and behaviors for states in the program.
10	Must use C++ library and explain	The program must make use of different C++ function libraries.	The use of C++ libraries will allow the use of different C++ syntax and codes that are supported by that library which will provide more robustness to the code.	To use CPP reference to find suitable C++ libraries that can be used to perform specific functions needed in the program code for better execution.
11	Must	Moderators must inherit all the admin permissions (however, Moderators cannot demote the Admin)	The moderators should give admin permission to add and remove members however, added users will not be able to remove admin.	This will give an admin the ability to control a chatroom and the members of the chat room.

12	Must	Users must be able to see the active users in the chat room	The user will be capable of viewing actively connected users in the chat room.	Grant users access to view chat room members.
13	Must use exception handling	The program must make use of exception handling that responds to exceptional circumstances that may arise during the execution of a program.	Exception handling will detect circumstantial errors in codes and rectify the errors to make sure the code does not generate errors during run time.	Will make use of try, catch, and throw to handle possible exceptional circumstances.
14	Must be able to access modifiers	The program must be able to determine what contents or features of the program are accessible to users.	The use of access modifiers will specify contents a user can access outside the code and the contents that are hidden from the user.	To specify the accessibility of the members of a class during the code design. This could either be a public or private access modifier.
15	Must use of multi-threads	Must be able to allow two or more parts of a program to run simultaneously.	This will reduce the processing time it takes before another part of the program can execute.	Making use of the thread library **<pthread.h>** to create executable threads in the code design
16	Must use of VCS tool	There must be a use of the VCS tool which has a repository for files that are constantly altered.	VCS will record the changes made to a file in the repository and recall a particular version.	Using git during code design to manage changes during the GUI program design

17	Testing cases and testing library	Use appropriate testing cases and testing library	Will test for all functions and process in the program and compare with expected results	Different types of test cases will be used including user interface test cases and functional test cases.
18	Could	The user pictures could be displayed in the channels	the user profile photo will be displayed in the channel.	This will allow for other users to distinguish between users and provide uniqueness in the channel.
19	Must	Must use classes	To realize MVC, at least one class where private member functions for common functionality that should not be exposed in the public interface.	The use of classes will make the program more versatile and easier to understand.
20	Must	Must use C++ library and explain	The program must make use of different C++ function libraries.	The use of C++ libraries will allow the use of different C++ syntax and codes that are supported by that library which will provide more robustness to the code.
21	Must	Must use exception handling	The program must make use of exception handling that responds to exceptional circumstances that may arise during the execution of a	Exception handling will detect circumstantial errors in codes and rectify the errors to make sure the code does not generate errors during run time.

			program.	
22	Must	Must be able to access modifiers	The program must be able to determine what contents or features of the program are accessible to users.	The use of access modifiers will specify contents a user can access outside the code and the contents that are hidden from the user.
23	Must	Must use of multi-threads	Must be able to allow two or more parts of a program to run simultaneously.	This will reduce the processing time it takes before another part of the program can execute.
24	Must	Must use of VCS tool	There must be a use of the VCS tool which has a repository for files that are constantly altered.	VCS will record the changes made to a file in the repository and recall a particular version.
25	Must	Testing cases and testing library	Use appropriate testing cases and testing library	Will test for all functions and process in the program and compare with expected results

[Table 2: Requirements List]

Risk assessment

Table Keys:

H – High

M – Medium

L – Low

Risk	Probability	Impact	Description	Impact on Project	Preventions/solution
Having unavailable member	M	M	When a team member cannot participate in the group activity for any reason.	Having an unavailable member could lead to a delay in task submission and improperly finished work.	If a member is unavailable due to external reasons, the group will collectively carry out the task of the missing person after completing their tasks.

Software loss conflicts	M	H	When the work of a group member is lost during a stage of project development.	This is a serious problem in the project development that may waste the time and resources allocated to the project.	All work must be submitted on GitHub or MS teams after every edit. This will not only backup files but will show professionalism in file handling which is required in the real world.
Internal conflicts	M	M	When there is a disagreement between the two or more of the members in a group.	Internal conflicts could interrupt the project's process and it could lead the project to unsuccessful results or failure.	To prevent internal conflicts, we should reflect every team member's opinions and try to listen carefully and respect each other, equally having a chance to speak out their opinions or express their ideas. In case, if there is a disagreement, the group will vote for the best opinion which will then be implemented.
Wrong time estimation	H	M	Certain tasks might require an increase or decrease in time to carry out an activity.	If little time is given to a member handling a large task to complete within a short amount of time, such member may not produce quality work and if	The Gantt chart will be reviewed by the project manager weekly to determine if any changes may need to be made to the time allocated to members.

				excessive time is given for a small task it would become difficult to complete the entire project on time.	
Poor tracking of resources	H	H	Resources such as group members and time may be managed poorly.	Poor management of resources may lead to overstretching or overuse of that resource. This may lead to insufficient availability of that resources in the future.	Time and members required for a particular task or process will be collectively thought of by the group members to decide how much time and manpower is required to achieve the task.
No proper background knowledge of IDE	L	H	The software developer may have little knowledge about how to navigate through an environment and use its features.	This will affect the final output of the project and many of the requirements may not be well implemented.	All the group members will be tasked with learning the IDE and its navigations and features. While the developer is designing the code, the other members will help improve the code structure.
Computational	L	L	Resources required to solve	While using applications for the	The research will be made on applications and their features. The

resources			computational problems.	project, certain apps can provide fast and easy solutions such as swift error detection. Applications without these resources can prove to be too difficult to use and may lead to wasting valuable time.	application with the most sophisticated and useful features will be used for the project development.
Lack of resources to implement the system requirements	H	H	Some of the requirements for the program may make use of external files and applications which without it the requirement may not be met.	This may not only affect the final program but will lack features that may allow it to be user friendly	All requirements will further be assessed to determine the possibility of implementation. If the task cannot be implemented an alternative feature that is more feasible will be used.
Poor contribution from team member	M	L	If a member is not able to successfully carry out the task assigned to him/her on multiple occasions.	The lack of contribution from a team member will affect the workflow of the project since some tasks may not be possible to start	Any member that intentionally disengages from any activity will be queried and if the member still cannot engage properly, such member will be replaced.

				without the completion of the previous task.	

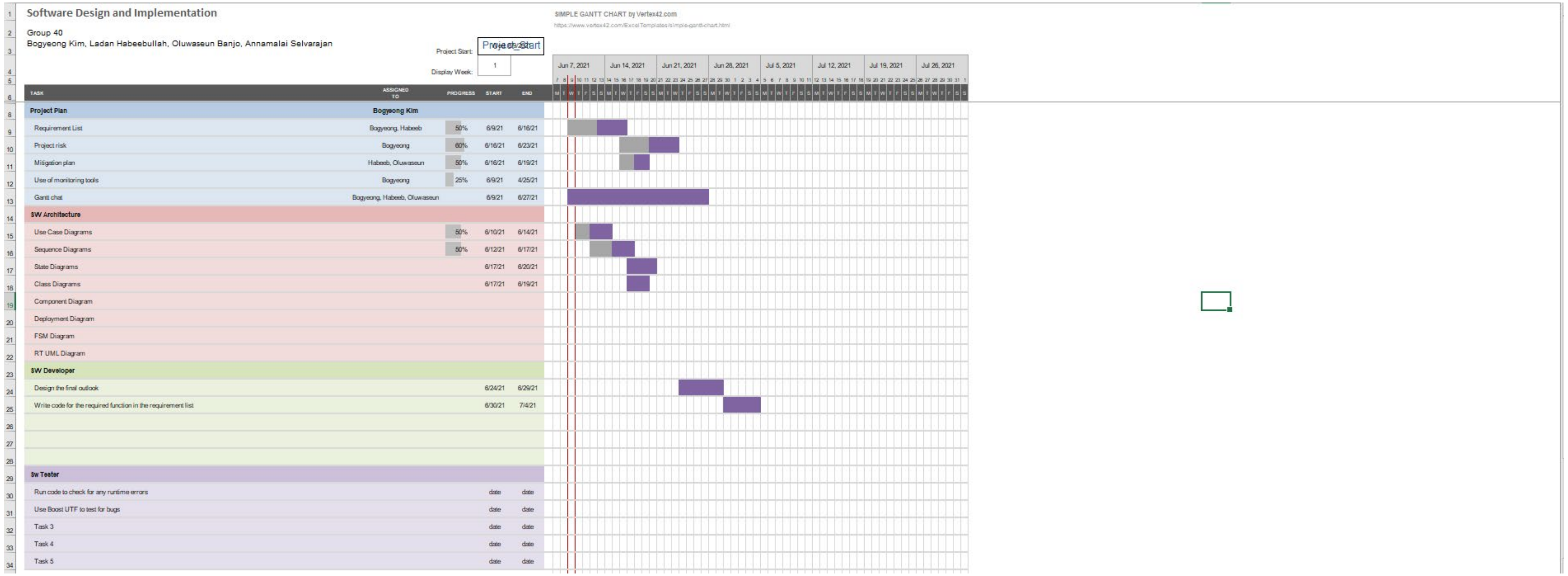
[Table 3: Risk Assessment]

Tasks list & Timeline

TASK	ASSIGNED TO	START	END
Project Plan	**Bogyeong Kim**		
Requirement List	Bogyeong, Habeeb	6/13/21	6/20/21
Project risk	Bogyeong	6/20/21	6/27/21
Mitigation plan	Habeeb, Oluwaseun	6/20/21	6/23/21
Use of monitoring tools	Bogyeong	6/13/21	4/25/21
Gantt chat	Bogyeong, Habeeb, Oluwaseun	6/13/21	7/1/21
SW Architecture	**Oluwaseun Banjo**		
Use Case Diagrams	Habeeb, Annamalai	6/14/21	6/18/21
Sequence Diagrams	Bogyeong	6/16/21	6/21/21
State Diagrams	Annamalai	6/21/21	6/24/21
Class Diagrams	Bogyeong, Annamalai	6/21/21	6/23/21
Component Diagram	Oluwaseun		
Deployment Diagram	Habeeb		

FSM Diagram	Oluwaseun		
RT UML Diagram	Oluwaseun		
SW Developer	**Habeebullah Ladan**		
Design the final outlook	Habeeb, Annamalai, Bogyeong	6/28/21	7/3/21
Write code for the required function in the requirement list	Annamalai and Bogyeong	7/4/21	7/8/21
Software Tester	**Annamalai Selvarajan**		
Run code to check for any runtime errors	Annamalai, Bogyeong	14/04/21	19/04/21
Use Boost UTF to test for bugs	Annamalai	19/04/21	3/06/21
Test documentation	Annamalai, Habeeb	3/06/21	11/06/21

Gantt Chart



Software Design and Implementation

Group 40
Bogyeong Kim, Ladan Habeebullah, Oluwaseun Banjo, Annamalai Selvarajan

Project Start: Project_Start

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Project Plan	**Bogyeong Kim**			
Requirement List	Bogyeong, Habeeb	50%	6/9/21	6/16/21
Project risk	Bogyeong	60%	6/16/21	6/23/21
Mitigation plan	Habeeb, Oluwaseun	50%	6/16/21	6/19/21
Use of monitoring tools	Bogyeong	25%	6/9/21	4/25/21
Gantt chat	Bogyeong, Habeeb, Oluwaseun		6/9/21	6/27/21
SW Architecture				
Use Case Diagrams		50%	6/10/21	6/14/21
Sequence Diagrams		50%	6/12/21	6/17/21
State Diagrams			6/17/21	6/20/21
Class Diagrams			6/17/21	6/19/21
Component Diagram				
Deployment Diagram				
FSM Diagram				
RT UML Diagram				
SW Developer				
Design the final outlook			6/24/21	6/29/21
Write code for the required function in the requirement list			6/30/21	7/4/21
Sw Tester				
Run code to check for any runtime errors			date	date
Use Boost UTF to test for bugs			date	date
Task 3			date	date
Task 4			date	date
Task 5			date	date

SIMPLE GANTT CHART by Vertex42.com

https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Figure 1 – Gantt-Chart: Part 1

This Gantt chart shows how the task was divided into segments.

	January				February				March				April				May			
Project Plan	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
T1:Requirement List	■																			
T2:Project risk		■																		
T3:Mitigation plan			■																	
T4:Use of monitoring tools			■																	
T5:Gantt chat		■																		
M1: To check if every plan is good and anythings need to add.		■																		
D1:Create deliverable1			■																	
SW Architecture																				
T6:Use Case Diagrams			■																	
T7:Class Diagrams			■																	
T8:Sequence Diagrams			■																	
T9:State Diagrams			■																	
T10:Component Diagram				■																
T11:FSM Diagram				■																
T12:RT UML Diagram				■																
M2: Make sure every diagram is done and				■																

include a description for each diagram.				🟦																
D2:Create final delivarable2					🟦															
SW Developer																				
T13:Making GUI-WelcomeMessage page/ Login/ Sign up page					🟦															
T14:MainPage- Contact List Channel					🟦	🟦														
T15:Search bar for users' name					🟦															
T16:Creating new chat rooms (include group chats)						🟦														
T17:MainPage- Message Chat Channel					🟦	🟦														
T18:display all the chatting list						🟦														
T19:Search bar for users' contents						🟦														
T20:MainPage- Chat Room page						🟦														
T21:display group admin and members						🟦														
T22:sending emoji and files while chatting							🟦													
T23:MainPage- Setting page							🟦													

T24:Deactivate account							🟦													
T25:Editing users' information (name, password, picture)							🟦													
D3:Create delivarable3								🟦												
SW Tester	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
T26:Test Plan								🟦												
T27:Results and screenshot											🟦									
T28:Workarounds								🟦												
T29:Use of Tests framework									🟦											
T30:Use of DevOp tools										🟦										
M3: Make sure every test is successful and no detecting errors on code, and write the report											🟦									
D4:Create delivarable4												🟦								

Figure 2 – Gantt-Chart: Part 2

This is a more simplified version of the Gantt chart in part 1

Coding Standards & Regulations

All code contributions to the git repository will be done by the Software Developer. Other member contributions must be submitted via Slack. Software Developer will inspect the code posted on Slack, ensure it follows the desired coding style and is bug-free.

The following set of rules depict the do's and attitudes each member should adopt during this development. These are the basis for ethical decision-making skills in the conduct of Software Engineering.

Standards.

- Respect each member and their privacy.
- Take responsibility for your actions & work.
- Take criticism as a learning opportunity.
- Deliver work on time.

Unacceptable.

- Do not take credit for somebody else's work.
- Fail to meet and take responsibility for a piece of the task requirements assigned to you.

Commits

Programmers should clone Project Directory from the git repository and function within it. However, the Software Developer alone must commit to the git repository. Rather, members' contributions should be added to Slack for Software Developer to inspect, make sure it meets

Coding Style standards, then commit to the git repository. Commits may occur upon achieving any milestone, each time the system passes a new test and at least every couple of days.

Members are urged not to commit any code that does not build. Code with bugs may be added to Github but if it cannot build, do not add it to Github. please first ensure the code compiles with zero errors or warnings before contributing.

Every code posted on Slack must follow a description of the code changes via the WhatsApp channel. At all times, avoid posting code without any description outlining the different functionalities added to the code. Software Developer will then append the code changes description to the 'feature Documentation – logging' folder on a git repository.

Bug Detection

The Following set of guidelines outline the procedures programmers are to observe to lower the probability of bugs in code.

- Build & run the project, confirm build issues zero errors or warnings.
- Ensure the updated version of the code still passes all tests.
- Beware of memory leaks! Use as few new/delete commands as possible. Ideally, none.

If any bug is detected in code; please either add a git issue or inform the Software Developer via one of our communication methods. Information on the bug should be descriptive and outline how to recreate the bug. The software Developer will debug the code.

Language – C++17

Integrated Development Environment (IDE) – Qt Creator 4.11.1

Graphical User Interface (GUI) – Qt Designer Form

Compiler - MinGW 7.3.0 32-bit

Coding Style

The following set of guidelines outline the desired programming style of the C++ code. All code must maintain this style for code consistency and readability.

Header Files

This development will have three main files. One for the header files (.h), one for the source files (.cpp), and one for the resources of the project (target folder).

- All headers must be kept in one file. E.g. "project/headers/allHeaderFiles.h."
- .h files should have a .cpp file associated with them (exempt main). All .cpp files may also be kept in a separate unique file. E.g. "project/src/allCppFiles.CPP"
- All header files should compile on their own. Hence, include #define guards to prevent multiple inclusion.
- Programmers are encouraged to Inline some shorts (accessors, mutators) or make use of lambda functions for more efficient and elegant code.

Header files must be listed as preceded with their project directory. E.g. #include "headers/header1.h"

In "project/src/src1.cpp", whose purpose is to test or manipulate members defined in "headers/header1.h", Include headers should be implemented in the following order;

i. ***Related Project headers, e.g. "headers/header1.h"***
ii. ***C system headers, e.g. <stdio.h>, <stddef.h>***
iii. ***C++ standard library headers or QT libraries, e.g. <QFileSystemModel>, <QFile>***
iv. ***Skip a line***

Source Files

Named & Unnamed Namespaces

Where appropriate, code should be placed in a Namespace as a safeguard to prevent collisions in the global scope. However, avoid Inline Namespaces and using derivatives. Keep Namespace relative to the project name and as first letter Upper, second letter Lower. E.g Ui.

Pointers

Whenever possible avoid multiple new/delete calls in code, as this increases the probability of memory leak. Rather, programmers are encouraged to make use of Smart Pointers.

Variables & Functions naming

- Variable names may be mixed cases. Begin the first letter with a capital letter followed by small letters (e.g. Line, Shapes, Roof).
- Method names must include at least one verb (e.g. displayImage() , modifyShapeSize()).
- Variable names with more than one word should be written as first letter capital followed by small letters till the end of that word. This should be continued with an underscore and starting with a capital letter and following small letters and so (e.g. Roof_Of_Shape, Name_Of_User, Current_Password).

Local variables*: Programmers are encouraged to initialize variables upon declaration, and have them close to first use, and within* as local a scope as possible. Whenever possible, the initialization should be used rather than declarations or assignments. Unless compelled to do otherwise, variables needed for if, while, and for statements should be declared within those statements.

Global & Static variables: Global or Static string constants should be avoided. If otherwise, use char array. However, consider global variables to avert the use of magic numbers. Global variable names may all be capital.

Functions: Write small and focussed functions as they are easier to test and maintain. Parameters passed by reference must be const. Functions overload is allowed where appropriate but there must be no semantic difference between the variants.

Classes

Keep everything in a class. Do **not** make use of struct unless having conversed with a Software Developer.

Constructors

- Virtual functions are forbidden within constructors.

Inheritance

- All Inheritance must be public.
- Multiple Inheritance is encouraged.
- Multiple Implementation Inheritance is discouraged.

Access Control

The access control sections may start in any order so long as the spacing is relatively constant and definitions neat to the eye. That being said, strive to begin with public members, then protected, and lastly private. Programmers are encouraged to group similar definitions within these sections.

Commenting

- Do not comment on the completely obvious.
- Comments should be a single-lined comment.
- Comments should be simple and precise.

Structural Diagrams

Use Case:

Sign-in:

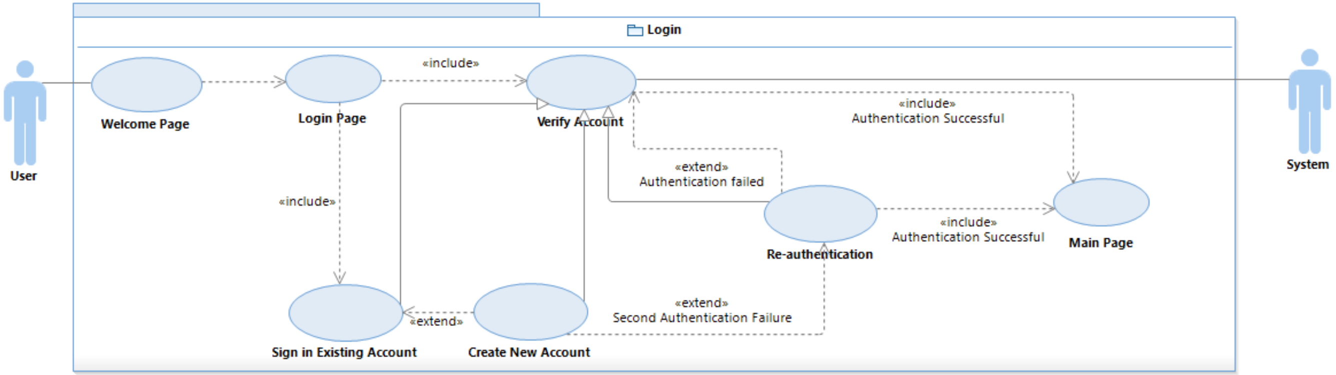


Figure 1 –Use case diagram 1

The Sign-in page starts with a welcome message. It requests the user input to continue to either the Log-in page or for Creating an account. Once the user entered his details, the system will authenticate the account validity. On account of authentication success, the system grants permission to proceed to the main screen. Contrarily, when authentication is failed, the user is asked to re-enter his/her details for verification. When authentication is failed multiple times, the system gives you a suggestion to create a new account. When the user tries to create a new account from the start, the system will verify some details and creates a new account for the user and automatically goes to the main page.

Create Chat Room:

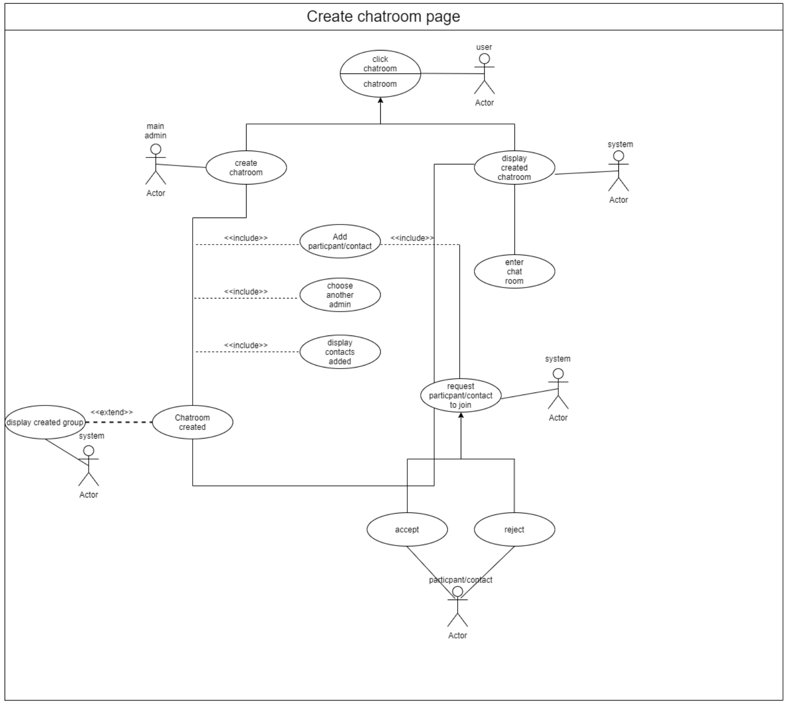


Figure 2 –Use case diagram 2

To create a chatroom, a user will need to click on the chat room selection on the platform. Once the user clicks on the chatroom, the user can either create a chat room or go to an already created chatroom which will be displayed as “display created chatroom”. If a user creates a chatroom, the user automatically becomes an admin who will need to add participants from a contact list. When a user tries to add a contact, the system will send a request to the contact to join the group. The contact can either accept or reject this request. The system will display any chartroom created by the main admin. The newly created chatroom will then join the list of a created chatrooms that a user can enter.

NOTE: When the main admin creates a chatroom they can still give the administration authority to other participants in the chatroom. Furthermore, any user can access a created chatroom so long as they are participants of that chatroom.

Add contacts:

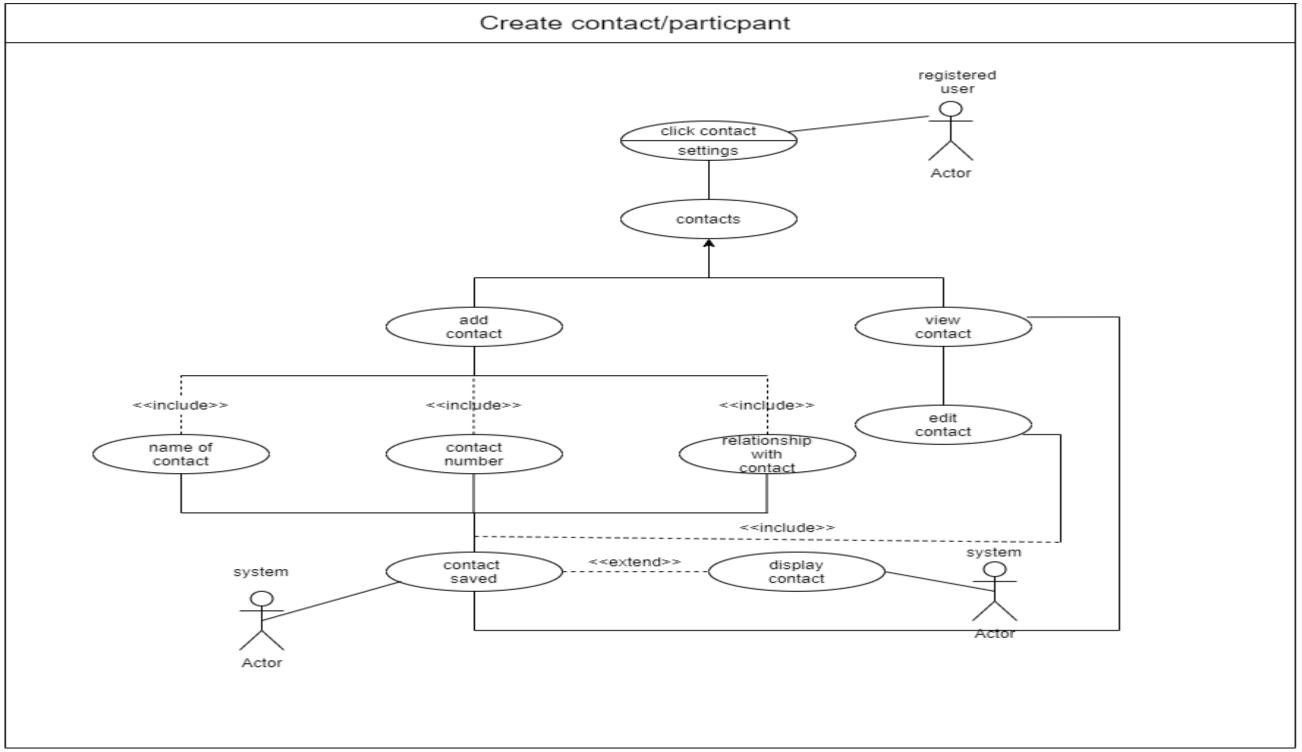


Figure 3 –Use case diagram 3

On the main chat page, a user can edit a contact or add a new contact. If a user wants to add a new contact, the user must name the new contact, add a contact number, and describe the relationship with the contact. Once the user is done creating the contact, the contact is then saved by the system and can be later viewed by the registered user.

NOTE: in the diagram, edit contact copies with the same action as add contact. This is because editing the contact takes the user back to the actions associated with adding a contact. The “relationship with contact” action describes the relationship with contacts such as a co-worker, friend, or family.

Main page:

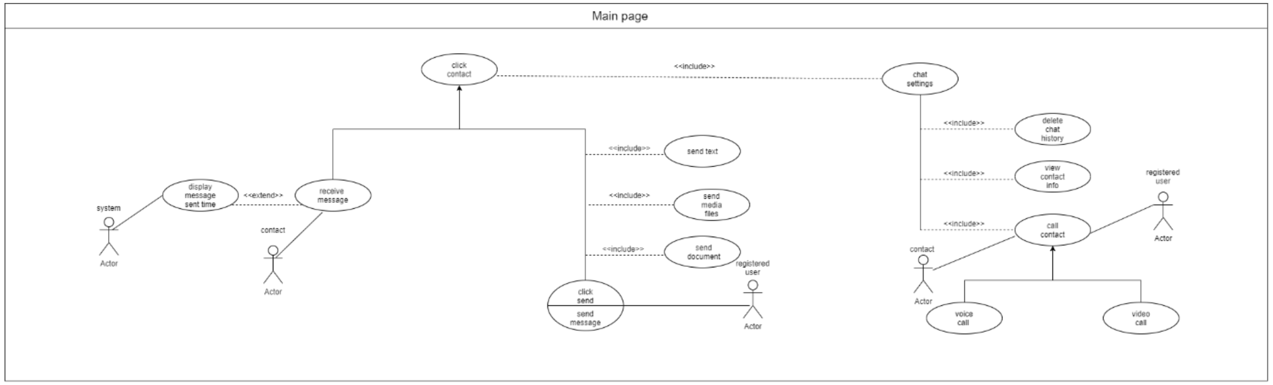


Figure 4 –Use case diagram 4

On the main page of the platform, the user mainly communicates with the contacts added. To communicate with a contact a user must click the contact the user wishes to communicate with. The user can either send or receive messages. The user can send normal text messages, media files such as audio and video files, and send documents such as word documents and text files. To send a message a user only needs to click the send button. Once a message is sent contact can receive the message and reply to it. The message sent and the time it was sent will be displayed by the system. In every personal chat, there will a “settings” action that will allow a user to delete chat history with contact, view contact info such as the date contact was added and relationship with a contact and user will be allowed to call contact although this feature may not be very feasible as it requires resources that are not available.

Settings:

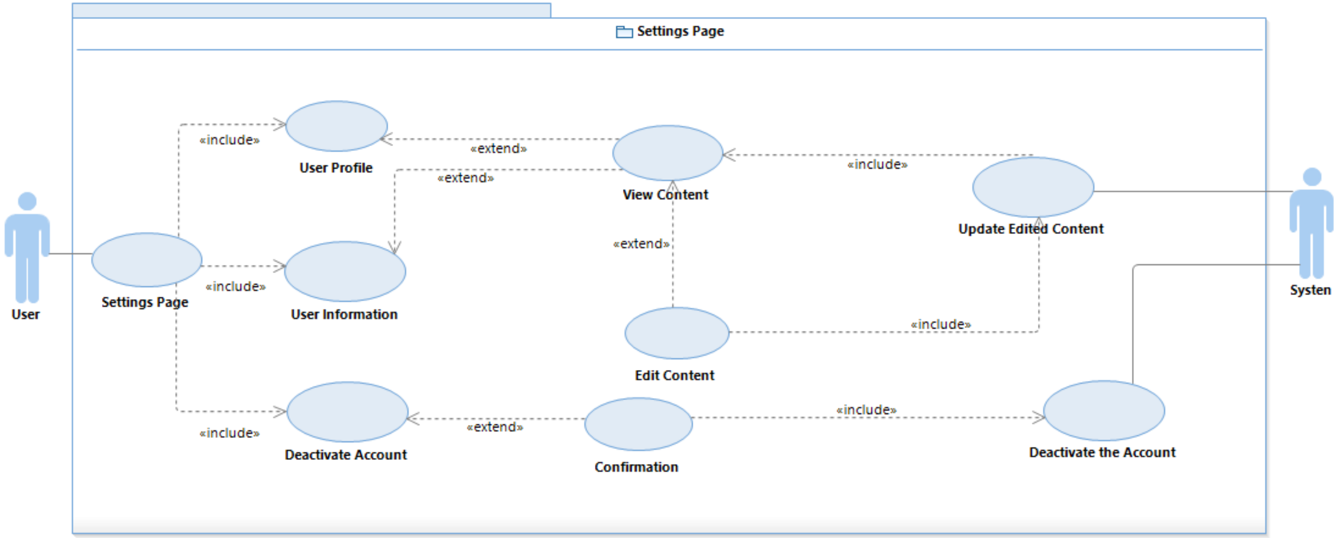


Figure 5 –Use case diagram 5

The settings page has three important features. They are,

- User profile.
- User details.
- Log-out/deactivate account.

The user has the liberty to view and edit their profile and other details. The user can log out or deactivate their account if they wish to do so.

Activity Diagram:

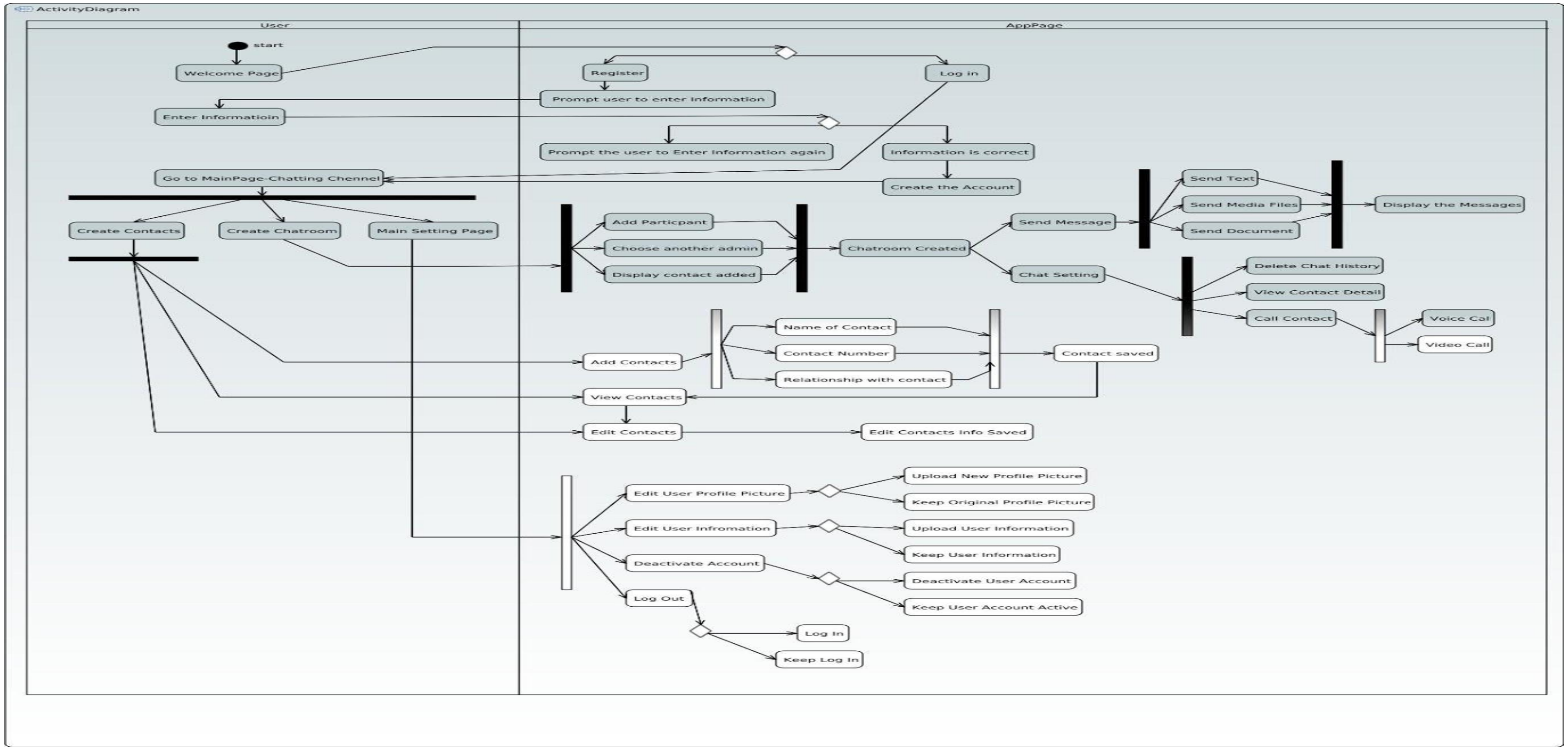


Figure 6 –Activity diagram 1

The activity diagram is a diagram that showcases the basic working functionality of our project. It starts by welcoming the user and proceeds to them to either logging in into their account or registering an account with us, hence where they will provide the necessary details and once they are done, their account is thereby created.

Once their account is created, the user is then taken to the main chatting channel where they can create new contacts, create chatrooms and view existing chatrooms, create groups, and their settings page. In the create contacts section, users can add, view and edit existing contacts and their information. In the chatroom section, they can create a new chatroom by selecting a contact from their contact list and then adding the participant to the chatroom. This sequence can also be done for groups as well by selecting multiple contacts and then adding them to the chatroom. In the chatrooms, users can send files, texts, images, documents, etc. to other users as well as delete chat history, view the contact information of the person they are in contact with, and make calls either voice or video depending on their choices.

On the main settings page, where the user can edit their profile picture either with a new picture from their device or take a new picture with their camera and upload it as their new profile picture, edit their contact information, log out or deactivate their account.

Class Diagram:

Sign-in:

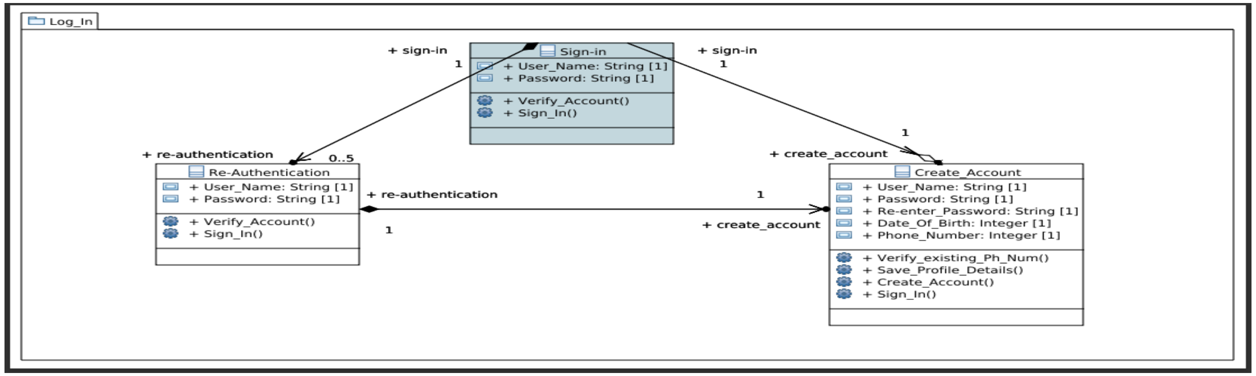


Figure 7 –Class diagram 1

The welcome page has only two key functions.

- Sign-in account.
- Create account.

When the user needs to sign in to an existing account, a user must enter their username and password which has a string value. The system must verify the details entered and re-direct to the subsequent pages accordingly. When the authentication is failed, the re-authentication should collect similar details and verify the account before signing in. When creating an account, the user enters their username, password twice, date of birth, and phone number. Since every user must have a unique phone number, the system verifies their phone number. Unique phone numbers will be approved by the system. Contrarily, existing phone numbers will be verified and delivers a message to type a unique phone number. When all details are verified the system saves the details and creates the account for the user.

Main Page:

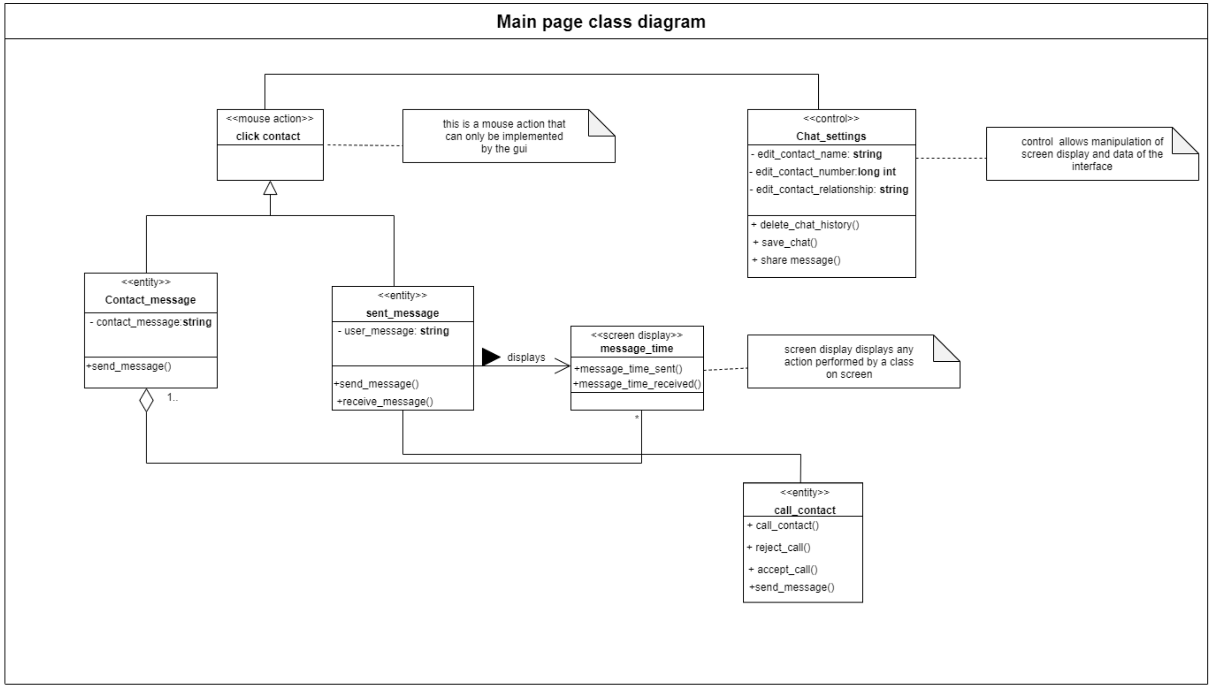


Figure 8 –Class diagram 2

Click contact: the click contact is a mouse action class. It does not contain any function or declaration, however, it is a superclass.

Sent_message: this is a class that involves the sending and receiving of messages between two users communicating with each other. The **send_message()**, will allow a user to send a message to the chatting contact, while the **receive_message(),** will allow a user to view a message sent by contact even when one of the two actors are not in the chat at that moment in time.

Contact_message: this is both a string variable and a class. The class is represented with a capital "C", while the variable is represented with a small "c" that shows a message sent by a contact. As shown in the diagram it is a private class. This is because until the **send_message()**, is applied no member of any other class can access the string inputted by the contact. As shown in the diagram shares a relationship with the **message_time** class.

message_time: message_time shares a relationship with both the contact_message and sent_message class. It is a class that essentially contains functions that display on the screen the time a message was sent and a time a message was received.

Call_contact: this is a class that allows a user to make calls, receive calls, accept calls, and reject calls. It is a feature that is related to the **sent_message** class. It uses the **send_message()**, to request a contact to join a call.

Chat_settings: the chat settings class is essentially a control class that edits contact data and chat data. The **edit_contact_name** is a string variable that allows the user to edit a contact's name while the **edit_contact_number** is a long int variable where the user can edit user number which is similar to how numbers can be edited in the chatroom. The edit_contact_relationship is also a string variable.

Three major functions are implemented in this class to ensure control. The first is the **delete_chat_history()** which allows the user to delete all the messages that have occurred between the user and the contact. The **save_chat()** permits the user to save a specific chat that is sent by a contact. Saving the chat is essential as all chats may auto-delete if not save to save RAM. Although this is the case the main reason for the use of **delete_chat_history(),** is to delete all the chats that were saved since the beginning of user and contact interaction. Finally, the **share_message()** allows the user to send messages to any other contact outside the private one on one chat between the user and contact.

Contacts:

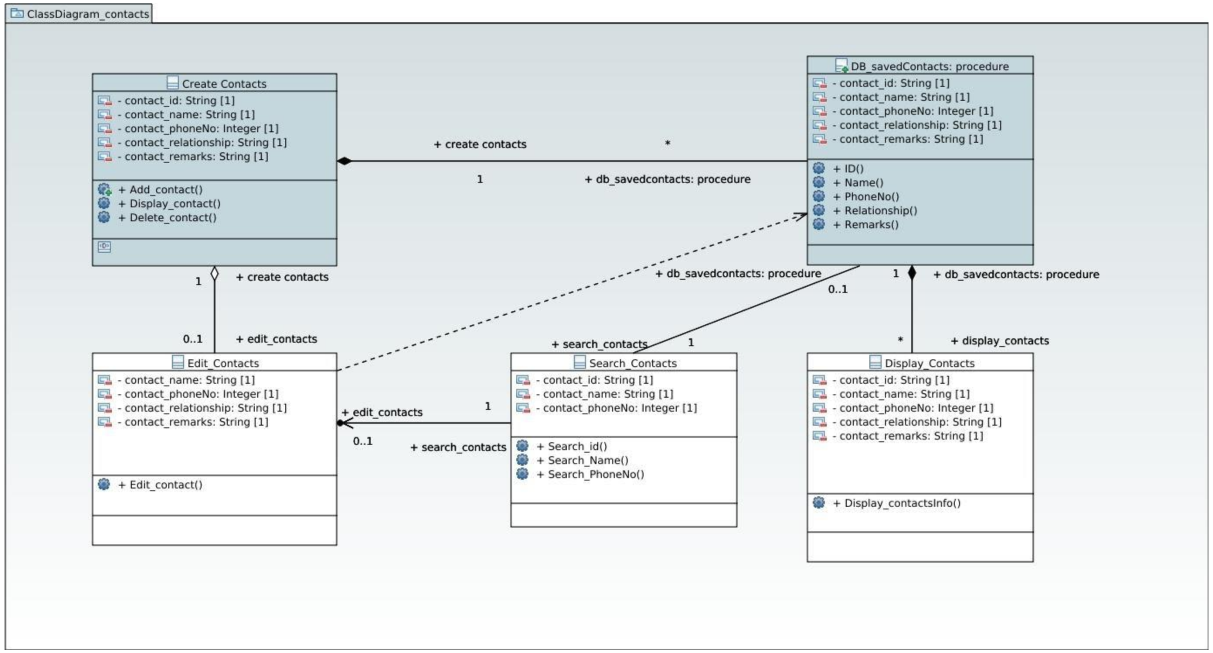

ClassDiagram_contacts
Create Contacts
- contact_id: String [1]
- contact_name: String [1]
- contact_phoneNo: Integer [1]
- contact_relationship: String [1]
- contact_remarks: String [1]
+ Add_contact()
+ Display_contact()
+ Delete_contact()
DB_savedContacts: procedure
- contact_id: String [1]
- contact_name: String [1]
- contact_phoneNo: Integer [1]
- contact_relationship: String [1]
- contact_remarks: String [1]
+ ID()
+ Name()
+ PhoneNo()
+ Relationship()
+ Remarks()
+ create contacts
*
1
+ db_savedcontacts: procedure
1
+ create contacts
0..1
+ edit_contacts
+ db_savedcontacts: procedure
0..1
1
+ db_savedcontacts: procedure
+ search_contacts
1
*
+ display_contacts
Edit_Contacts
- contact_name: String [1]
- contact_phoneNo: Integer [1]
- contact_relationship: String [1]
- contact_remarks: String [1]
+ Edit_contact()
+ edit_contacts
1
0..1
+ search_contacts
Search_Contacts
- contact_id: String [1]
- contact_name: String [1]
- contact_phoneNo: Integer [1]
+ Search_id()
+ Search_Name()
+ Search_PhoneNo()
Display_Contacts
- contact_id: String [1]
- contact_name: String [1]
- contact_phoneNo: Integer [1]
- contact_relationship: String [1]
- contact_remarks: String [1]
+ Display_contactsInfo()

Figure 9–Class diagram 3

This class diagram contains the function of creating and editing contacts. For creating contacts, it has class attributes which are followed by customer_id: string, customer_name: string, contact_phoneNo: int, contact_relationship: string and contact_remarks:string. Also, it has class operations which are add_contact(), display_contact(), and delete_contact(). Also, in Edit_contacts, except for the contact_id, every other attribute that is stored in creating contacts, users can edit contacts. For all stored attributes will be saved in DB_savedContacts. When the system searches contacts details from DB_savedContacts, it will display their contact details. This is what Display_contacts class does.

Group Chat:

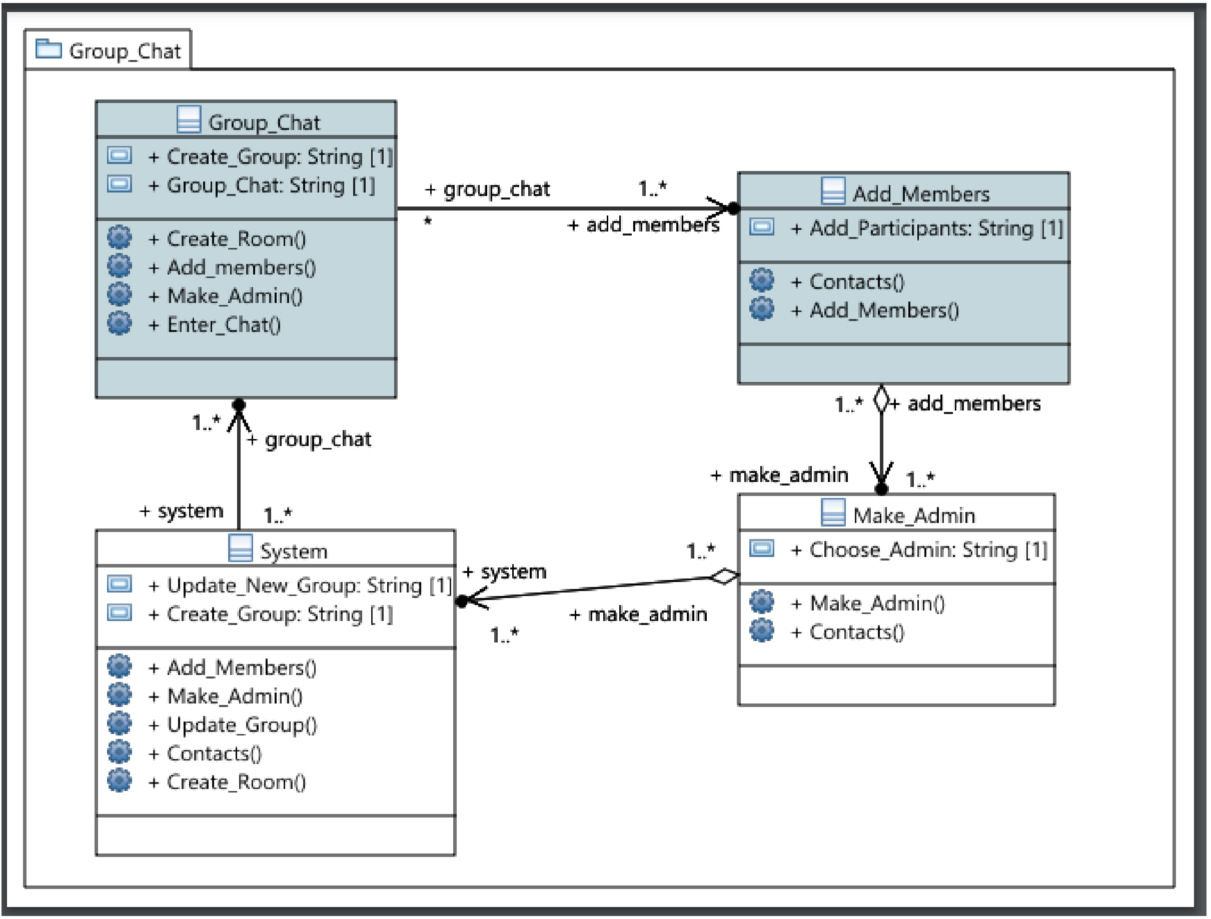


Figure 10–Class diagram 4

This shows how the creation of a group chat within the app should take place. It first creates the group, then creates the room and proceeds to prompt the user to add the members they would like in the group, then it allows the user to select any of the members to be an admin in the group. Once that's done the group chat is created.

To add a user to the group, the user would be prompted to go into their contacts and add the contact to the group to be a member.

To make a contact with an admin in the group, the user will go to their contacts and select the desired contact and make them an admin in the group. So, therefore, the system updates the group based on the user's inputs e.g adding new members, changing the group name, creating more admins in the group, etc.

Settings:

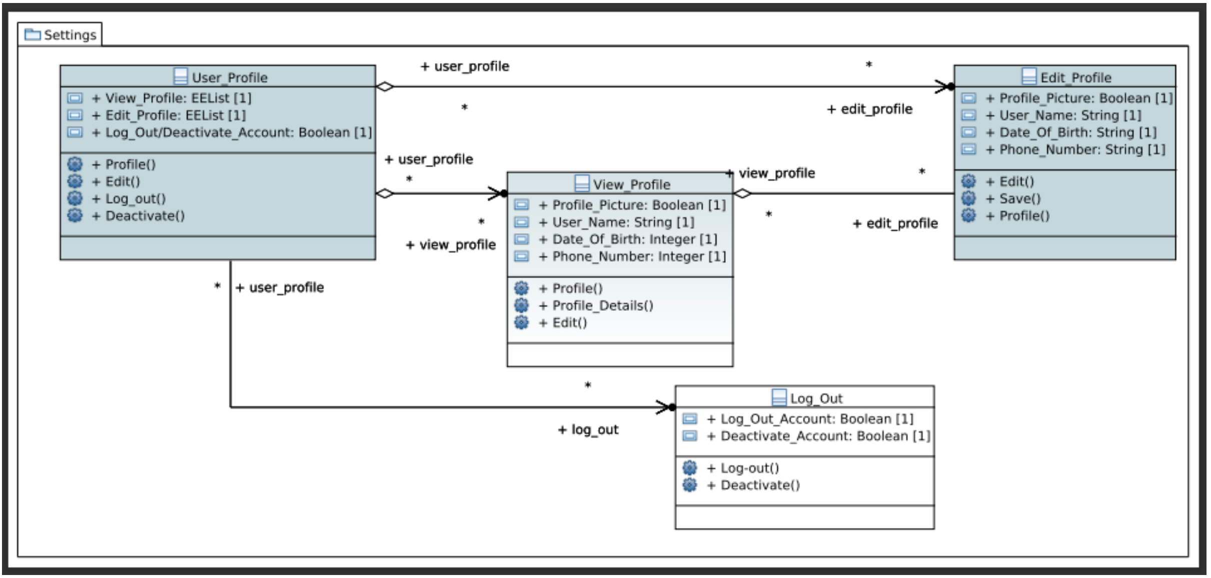


Figure 11–Class diagram 5

Settings page has the option to view and edit user profile picture and details. By doing so, the system must save the edited profile and update them to the user. The user can log out their account after receiving a confirmation from the system. Same procedure follows for deactivating the account.

Sequence Diagram:

Log-in:

As mention is the use case diagram, the user has the option to choose to sign-in or create an account in this page. In the process of sign-in, the user is asked to enter their username and password. After authentication success, user can go directly to the main/home page. On authentication failure, user enters the username and password again. On success it proceeds to the main page. Contrarily on multiple failure, user is suggested to create a new account. While creating a new account, user must enter their username, password, date of birth and phone number. They would also be asked to re-enter their password. On all the details have been entered, the system verifies their phone number with existing phone numbers. If the phone number exists, user will be asked to enter a valid phone number. If it does not exist, system saves the details and creates the account and enters main page.

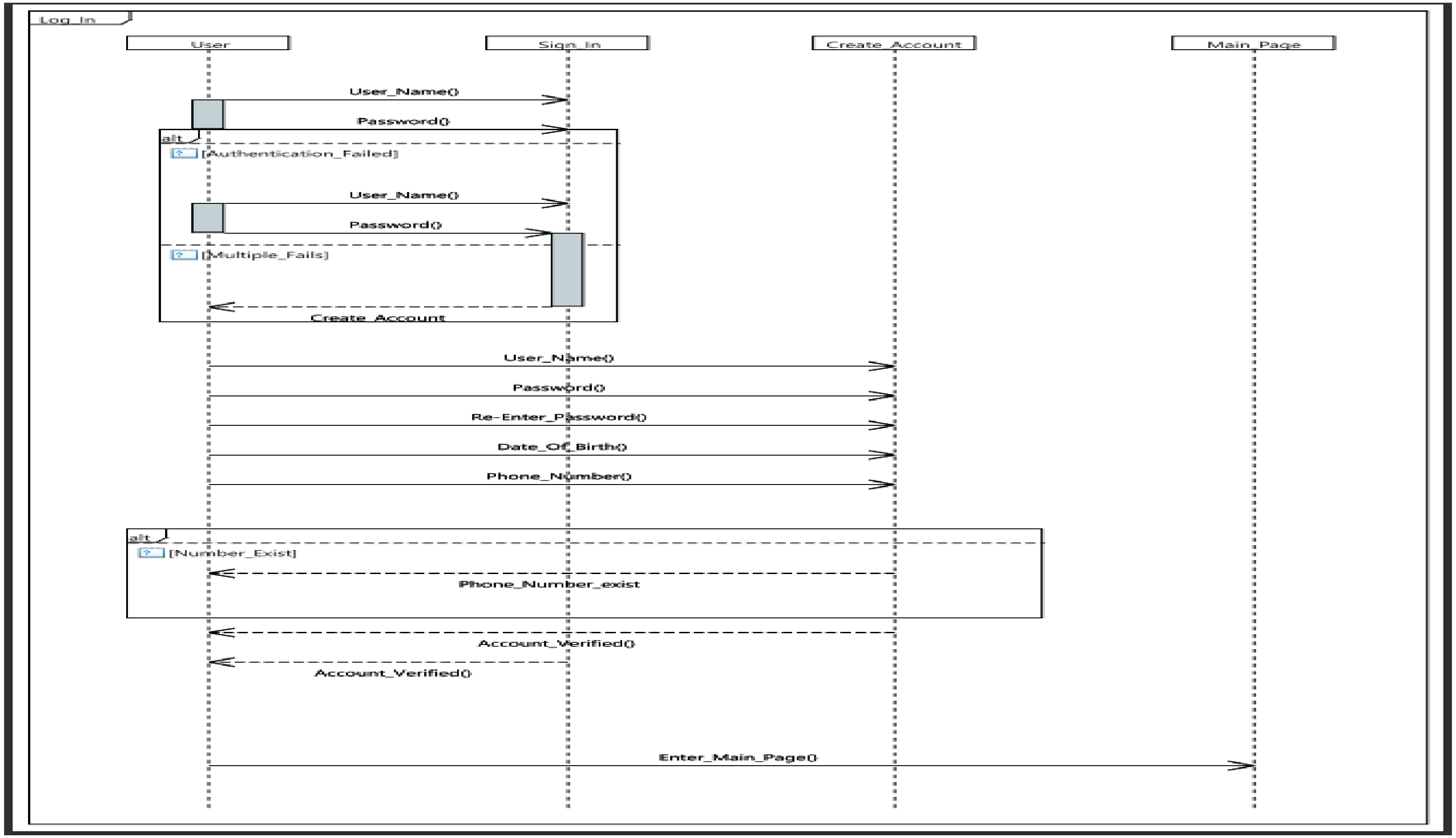

Log_In
User
Sign_In
Create_Account
Main_Page
User_Name()
Password()
alt
[Authentication_Failed]
User_Name()
Password()
[Multiple_Fails]
Create_Account
User_Name()
Password()
Re-Enter_Password()
Date_Of_Birth()
Phone_Number()
alt
[Number_Exist]
Phone_Number_exist
Account_Verified()
Account_Verified()
Enter_Main_Page()

Figure 12–Sequence diagram 1

Home page:

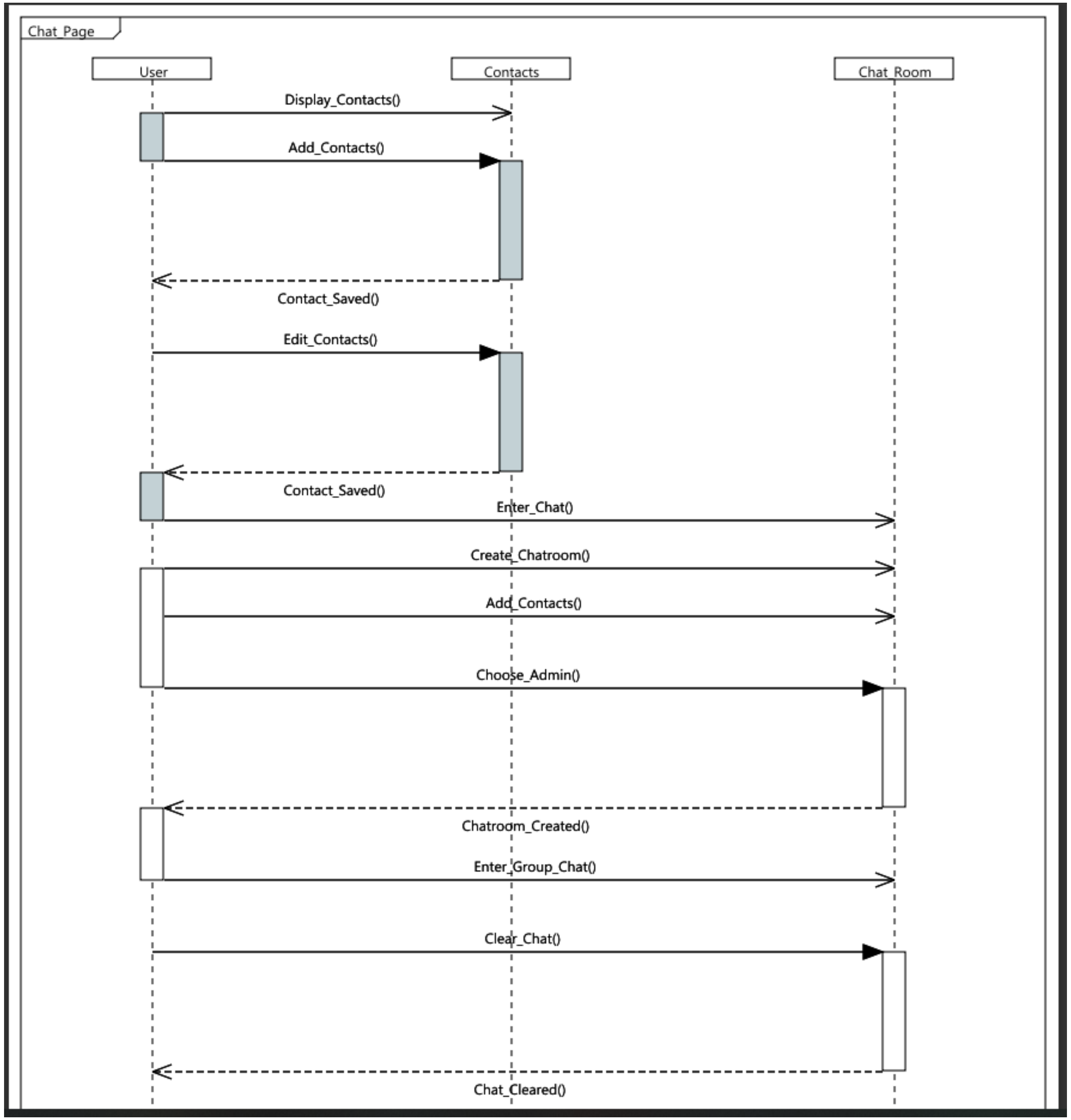

Chat_Page
User
Contacts
Chat_Room
Display_Contacts()
Add_Contacts()
Contact_Saved()
Edit_Contacts()
Contact_Saved()
Enter_Chat()
Create_Chatroom()
Add_Contacts()
Choose_Admin()
Chatroom_Created()
Enter_Group_Chat()
Clear_Chat()
Chat_Cleared()

Figure 13–Sequence diagram 2

This page is mainly for chatting with user’s contacts and creating a group with them. The user has the provision to add new contacts and edit existing contacts. The system updates the contact as the user edit or create a contact. Additionally, user could also create a group chat room to chat in groups. This could be done in three easy steps.

1. Selecting create chat room.
2. Add participants.
3. Choose admin.

Chat room will be created and ready to go.

The user also has the liberty to clear their chat history.

Settings Page:

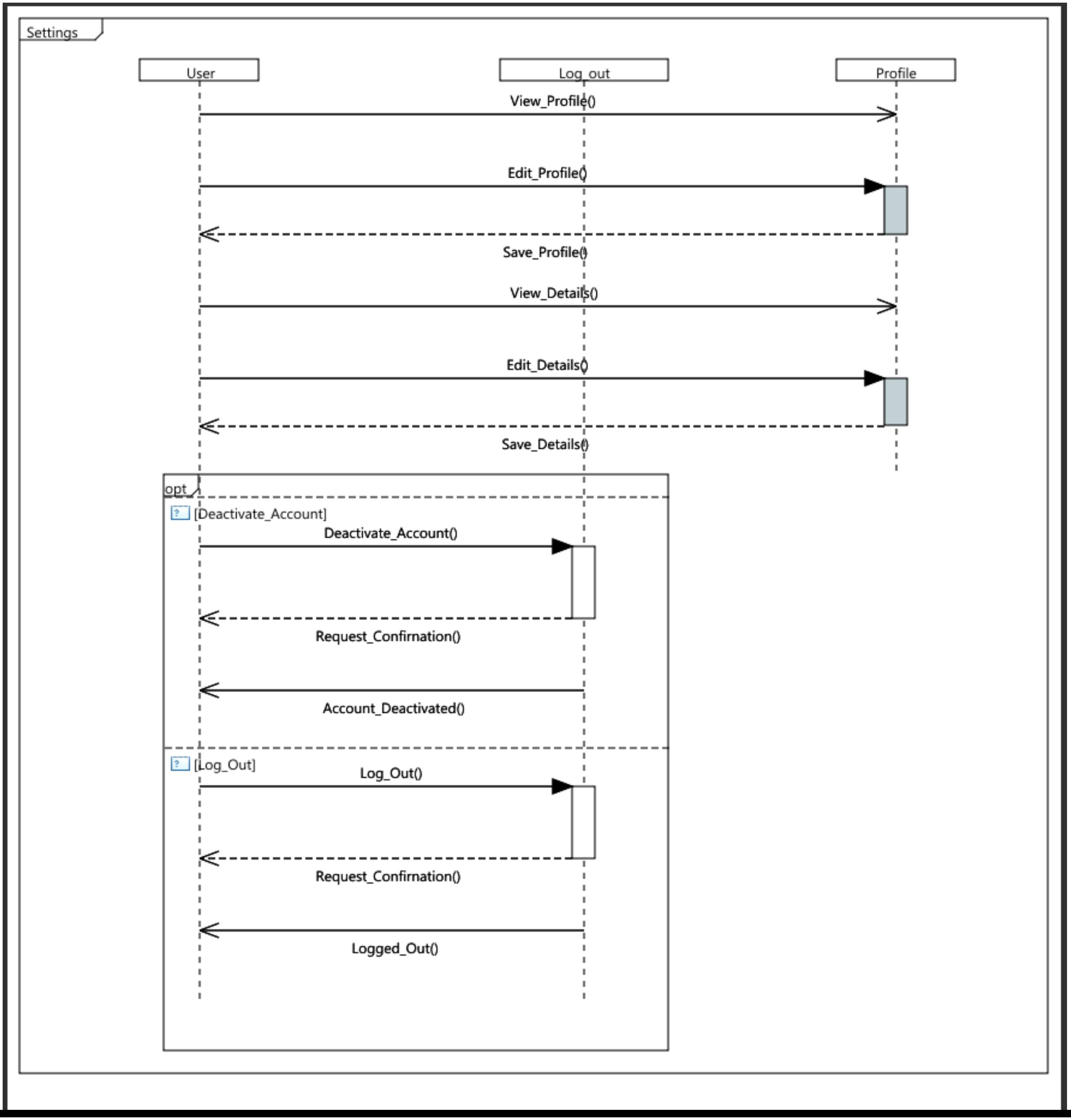
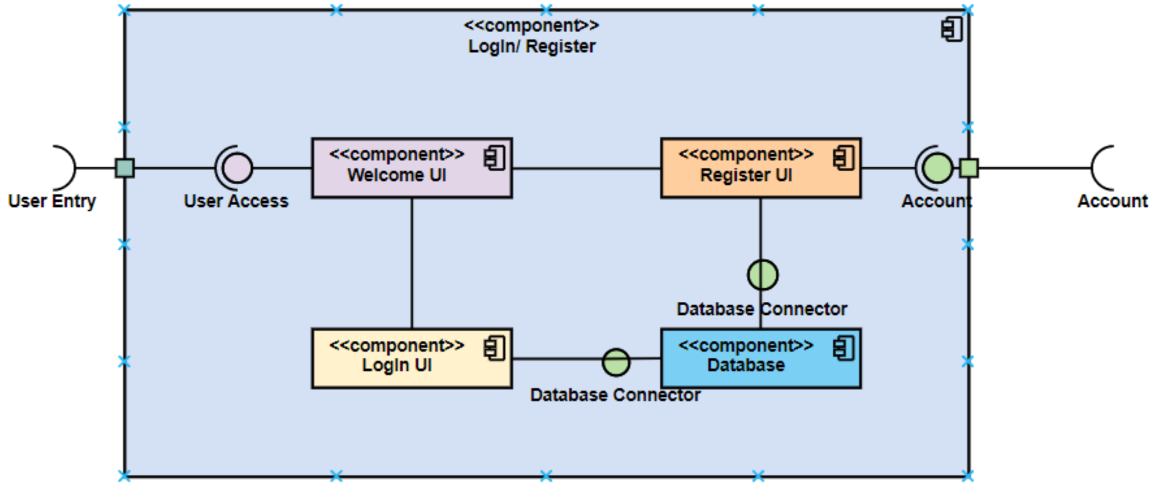

Settings
User
Log_out
Profile
View_Profile()
Edit_Profile()
Save_Profile()
View_Details()
Edit_Details()
Save_Details()
opt
[Deactivate_Account]
Deactivate_Account()
Request_Confirmation()
Account_Deactivated()
[Log_Out]
Log_Out()
Request_Confirmation()
Logged_Out()

Figure 14–Sequence diagram 3

The settings page is the user to edit the user profile, details and log out/deactivate the user account. The user could view and edit not only his profile picture but also his details. When the user wished to log out of deactivating their account, the system asks for a confirmation and makes that action.

Component Diagram:



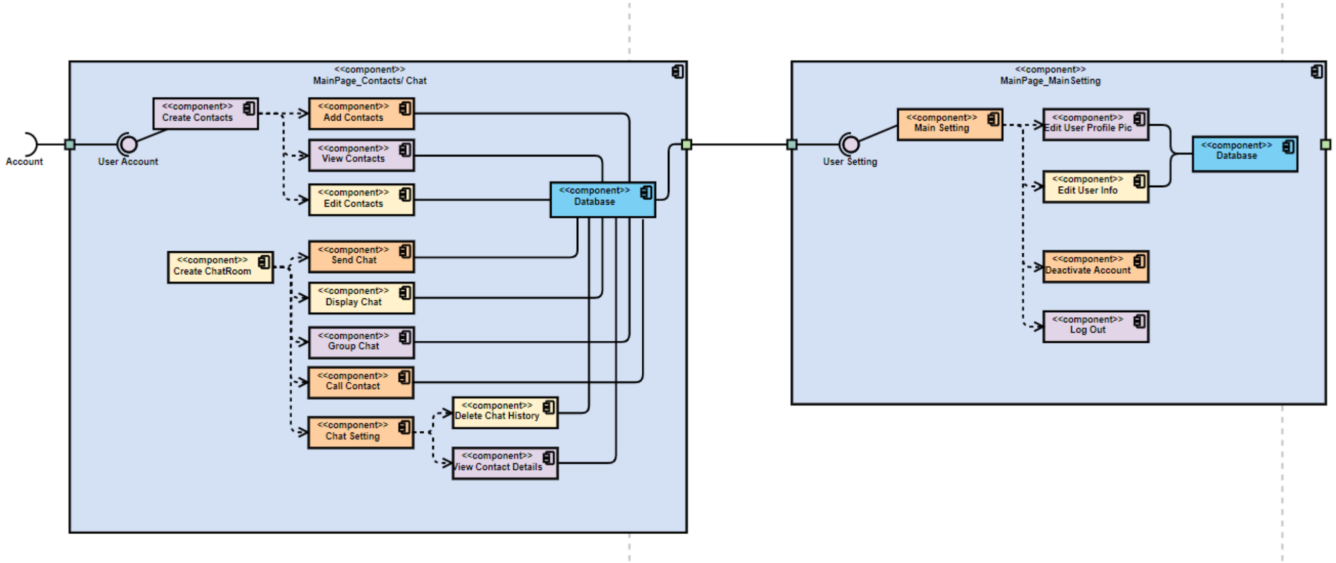


Figure 15–Component diagram 1

In this system, there are three main components. These are Login/Register, Mainpage-Contacts/Chat, and Mainpage-Main Setting. For the Login/Register components part, it gets the input for User Entry. It connects to the welcome UI and it goes to Login UI or Register UI. If a user has signed up for the account, it goes to Login UI. If a user hasn't, it goes to Register UI. Also, those UI is all connected to Database. This Database manages the Login information and registration information. After when the account has been created, it goes to the create contacts. This creates contacts component contains: add contacts, view contacts, and edit contacts. When created contacts have finished, the user can allow creating a chat room. This creates a chat room component that includes sending chat, display chat, group chat, call contact, and chat setting. In this chat setting, the user can delete chat history and view contact details. Everything in this Mainpage- contacts/ chat component stores the information and data in the Database. This system also has the Main Setting component. This Main Setting component allows user to edit their profile picture, edit user information.

This component will edit the information and data that is already updated in the Database. Once user edits their picture and user information, and this newly changed information will be updated to Database. Also, in the main setting, the user can do deactivate the account and log out.

FSM Diagram:

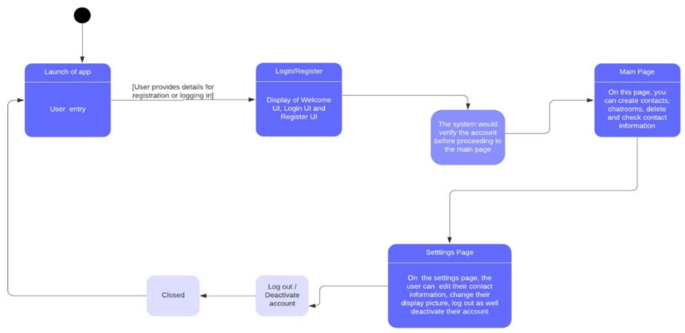

Launch of app
User entry
[User provides details for registration or logging in]
Login/Register
Display of Welcome UI, Login UI and Register UI
The system would verify the account before proceeding to the main page
Main Page
On this page, you can create contacts, chatrooms, delete and check contact information
Setttings Page
On the settings page, the user can edit their contact information, change their display picture, log out as well deactivate their account
Log out / Deactivate account
Closed

Figure 16–Fsm diagram 1

This diagram shows how the application will operate as a user makes use of it. It will be idle at the start, then the user will launch the application, then be prompted to enter their details for either registration or for logging in as an existing user. The prompt for the login or registration is then displayed which the user must pick one to allow access to the application. Beforehand, the system will prompt the user to verify their account before they are proceeding to the main page once verified, they are proceeding to the main page. Once on the main page, the user can therefore perform operations such as create contacts, chatrooms, delete and check contact information. The user can also visit the settings page where they can edit their contact information, change their display picture, log out and deactivate their account. They are also prompting which allows the user to choose between logging out and deactivating their accounts, once they have chosen the action of their choosing, the application is then closed.

Communication Diagram:

Sign-in:

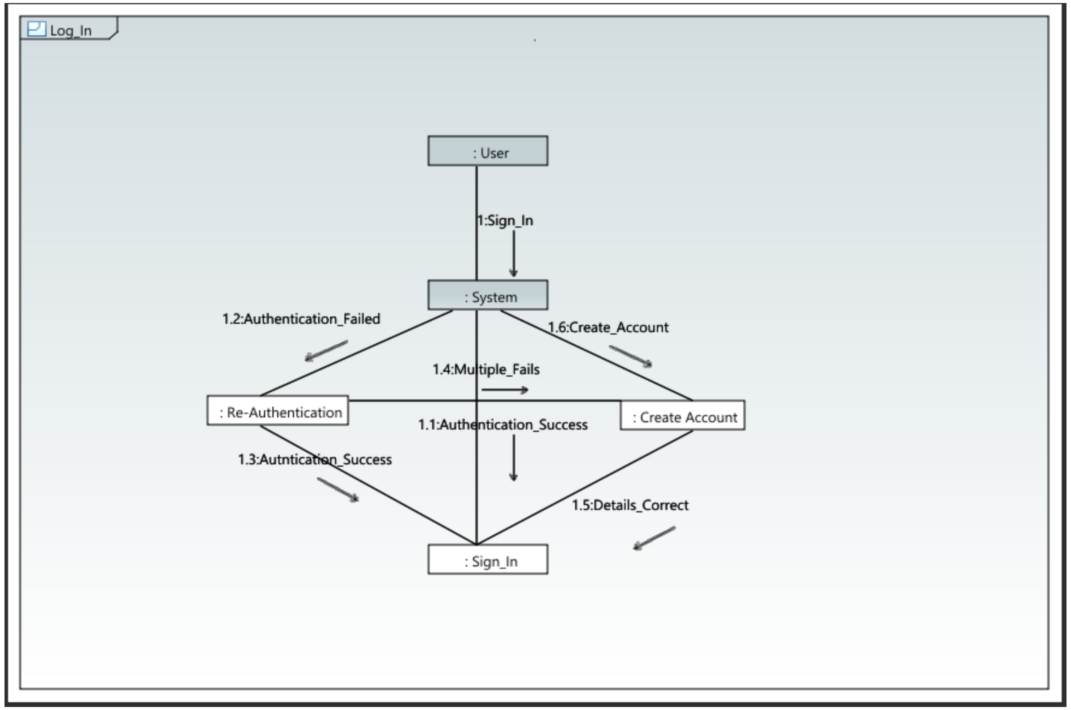


Figure 17–Communication diagram 1

On the sign-in page, the user asks the system to sign in. When authentication is a success, the system gets signed in. when the authentication failed, the system asks for reauthentication and gets signed in. On account of multiple authentication failures, the system suggests the user create a new account. When the user wants to create an account, the system creates an account when the entered details are correct.

Home page:

In the first sequence, the user tries to view his contact, then adds contacts, and the contacts are updated. Or the user could edit created contacts and it gets updated. In the second sequence, the user tries to create a chat room. To do so, the user must ask the system to create chatroom form settings, add members, choose admin and the group chat is now created. In the third sequence, The user asks the system to clear the chat history of their contacts.

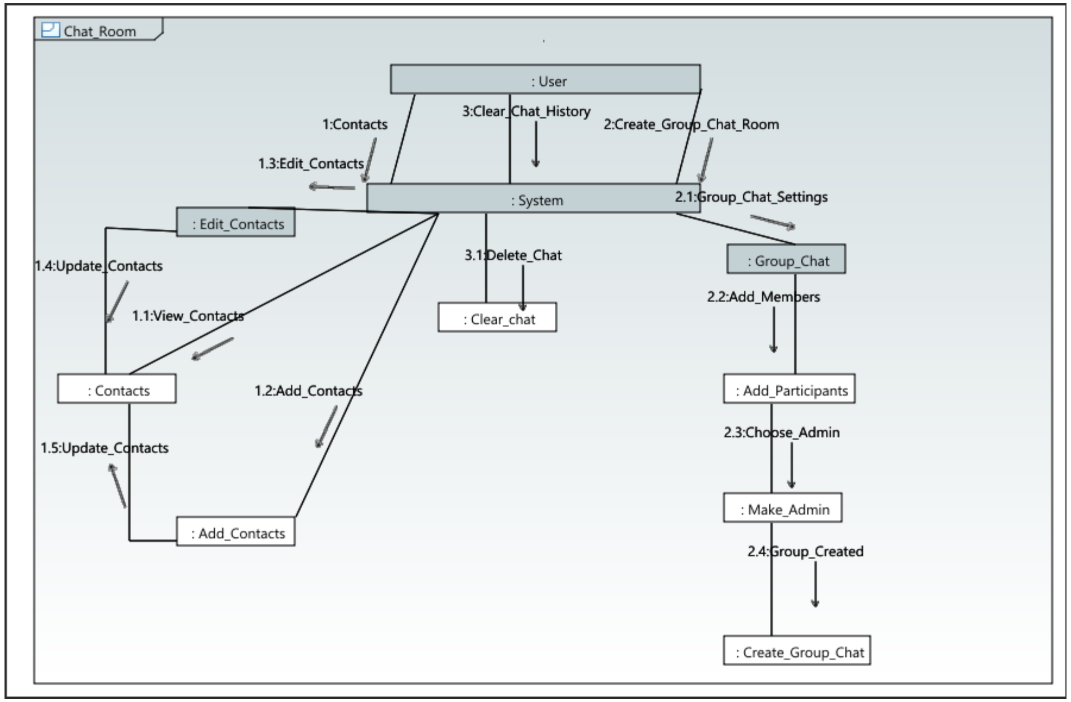


Figure 18–Communication diagram 2

Settings:

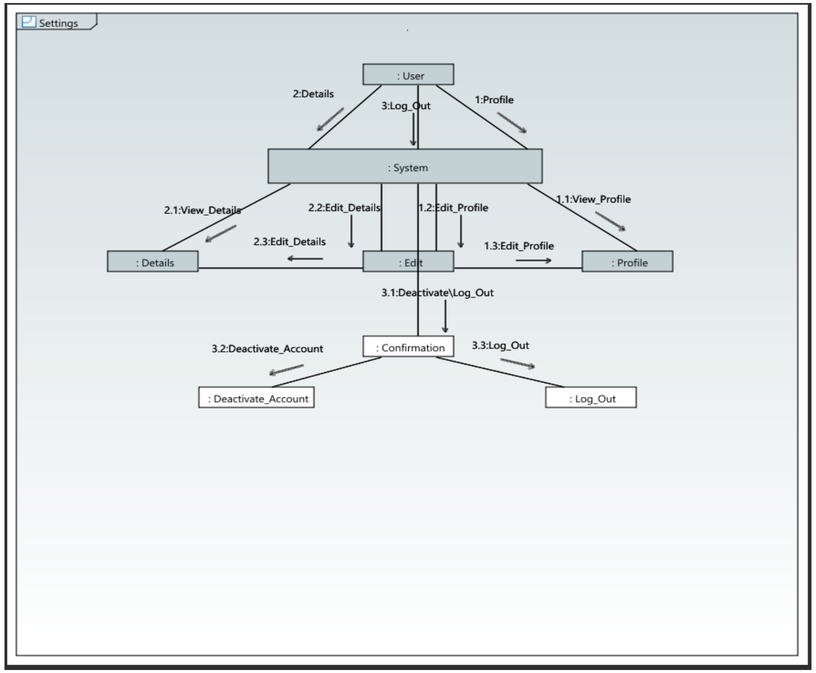


Figure 19–Communication diagram 3

The user can view and edit their profile and details by communicating with the system. In the first sequence, the user tries to view and edit his profile. In the second sequence, the user tries to view and edit his details. In the third and final sequence, a user asks the system to log out, which is executed by asking for confirmation from the user. The same procedure is followed when the user wants to deactivate their account.

Deployment Diagram:

Network topology

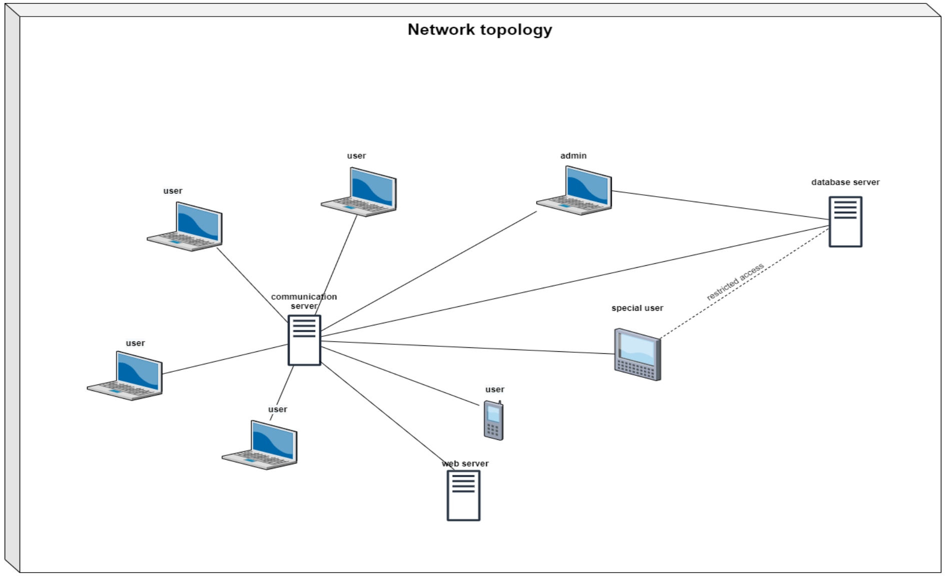


Figure 20–Deployment diagram 1

The network will make use of a star topology. All devices will be connected to a central server where the majority of the network activity is carried out. The central server also is connected to a web server to support an internet connection to retrieve internet resources. The central server will also connect to a database to store information. The database will also connect to specific users such as admin who will have access to the database.

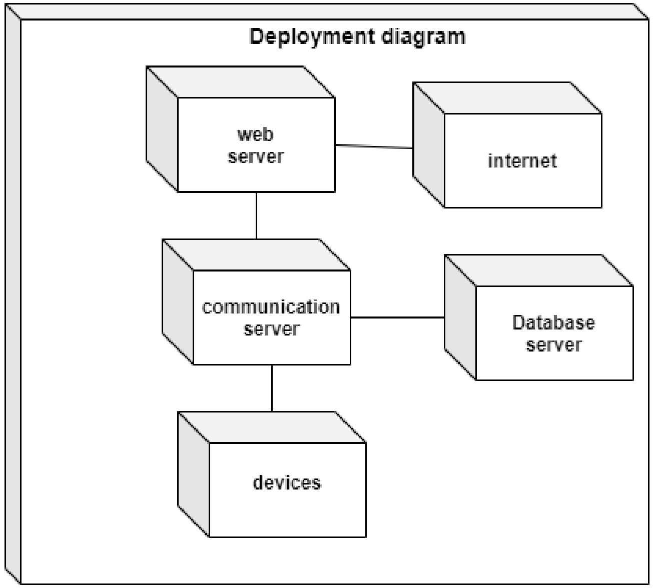


Figure 21–Deployment diagram 2

The web server is a server that will be used to provide web references and internet resources. Essentially it provides a connection to the internet. The communication server is responsible for the organization of data between user and contact. It is linked to the database server which collects vital information about a user and a contact and securely stores it in a database. Furthermore, there is a connection of devices that are directly connected to the communication server. The devices are the medium where the platform will be tested and implemented examples of these platforms are, computers, tablets, phones, and mobile devices.

Database server

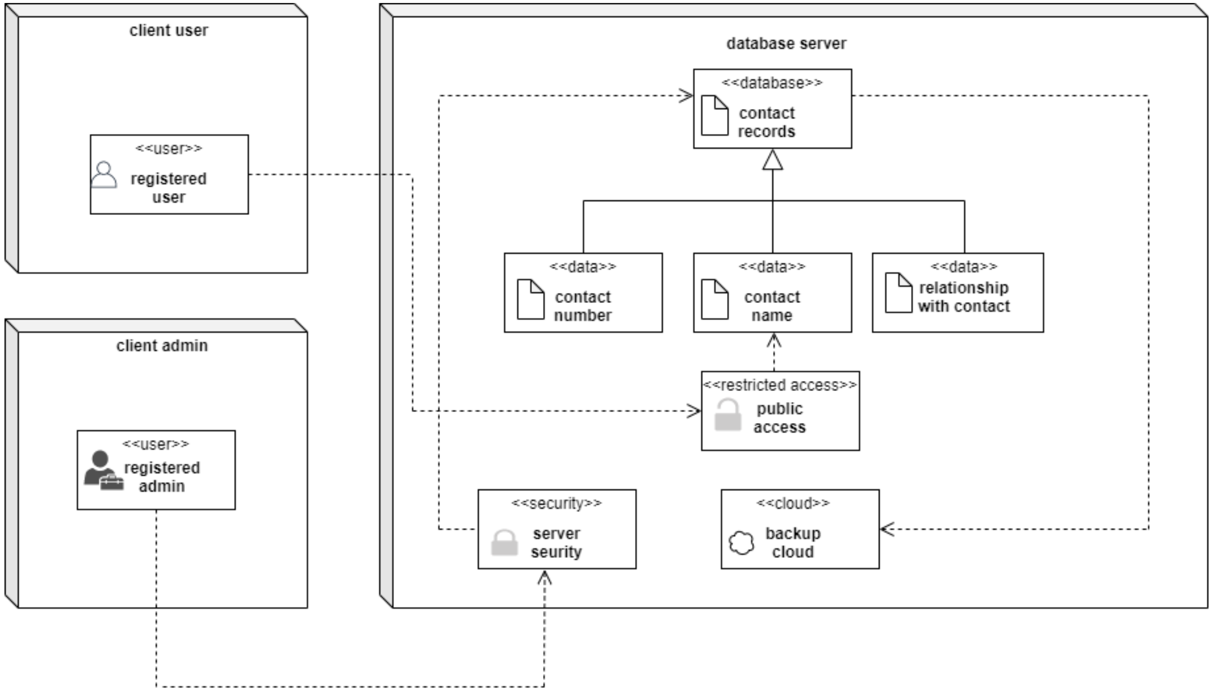


Figure 22–Database server diagram

The database manages all the data retrieved from the communication server. Specifically, the contact's record is derived from the communication server. Once the data is stored in the database, server security is implemented to secure information and data for user confidentiality. This security prevents any random external user except an admin to have access to it. However, specific users can have restricted access to the database. This is made in case of a situation where an admin is not available to view the database. The information provided to these users is highly restricted. In this scenario, only the contact name is accessible from the database. The database will be backed up in a cloud in a situation where the server crashes.

Communication server

The communication server involves the communication between a user and a contact and collects user information which is transferred to the database server for security and recovery. A user will need to register to the platform. This is where user data will be collected. Once a user is registered user will need to verify details to have access to contacts and personal information. Every saved chat will be immediately backed up in a situation where the server crashes.

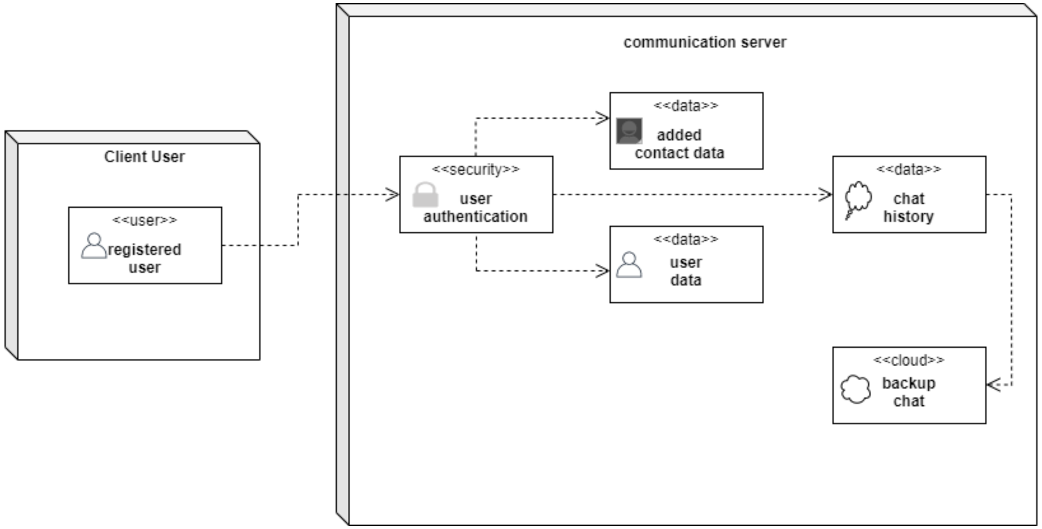


Figure 23–Communication server diagram

GUI design

First design

Welcome:

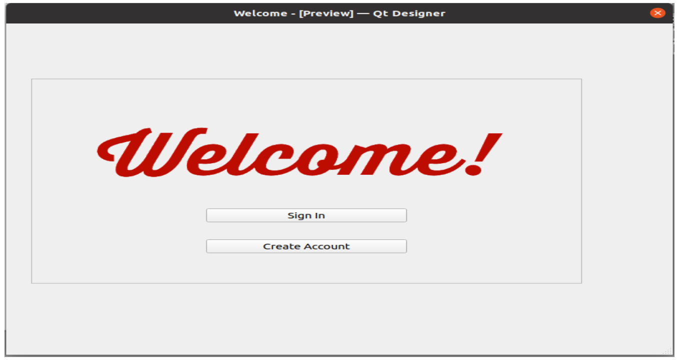


Figure 24–GUI first design

As the user installs the software, the first screen that will be popping up will be the welcome page. It has two Push buttons that link to a sign-in account and create an account.

Sign-in:

The Sign-in page has two text inputs that ask for a username and password. It also has to push buttons to continue or cancel the sign-in process.

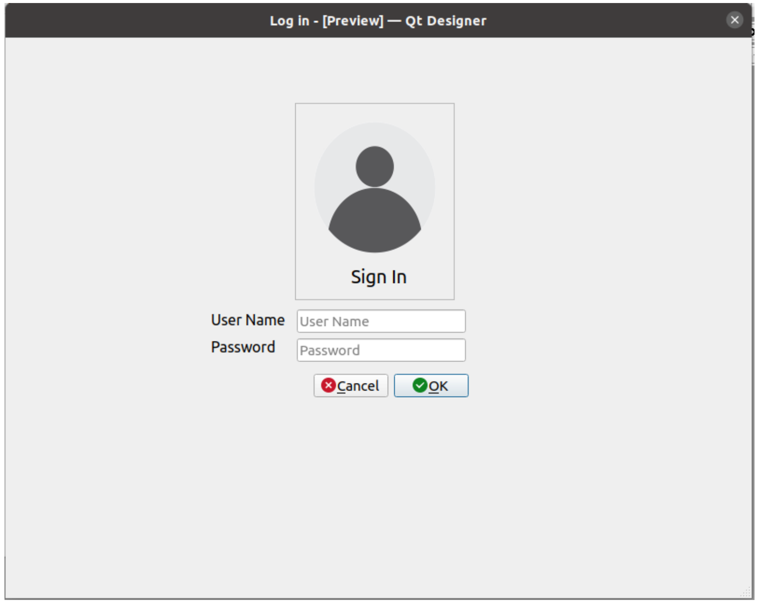


Figure 25–GUI first design sign in

Create account:

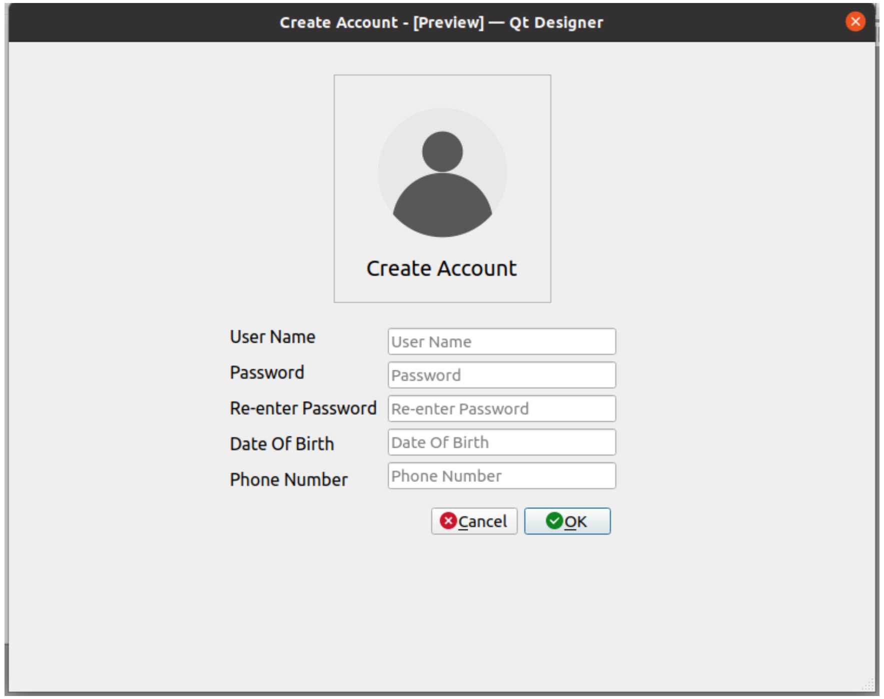


Figure 26–GUI first design create account

Create account page is used to create an account for the user if they do not have one. The user must fill in all the required details and create an account. They could also cancel the process if they have an account.

Home page:

The main page/home page is the heart of the software. It has a search bar at the top to search contacts and groups that exist. The settings dropbox has the provision to create a group chat room, add contacts, and settings page. The left corner box displays the contact list of the user and the right corner box displays the groups where the user is a participant. In the main chatbox, the user contact's name and profile. They could view their full details, send documents, and clear chat history by clicking the settings near the contact profile. The user can not only messages send but also emojis to make the chat lively.

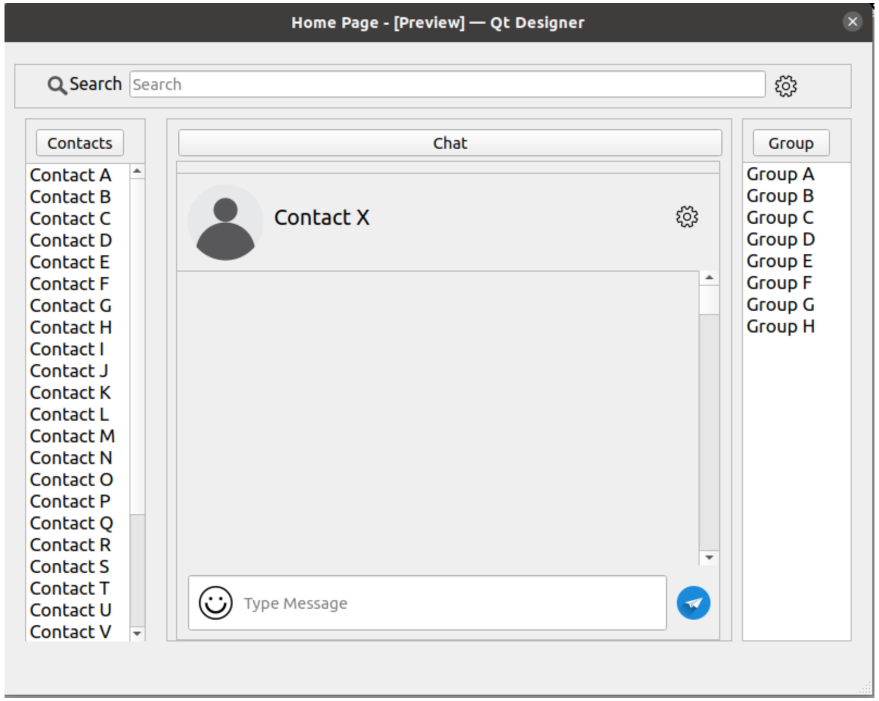


Figure 27–GUI first design chat page

Settings:

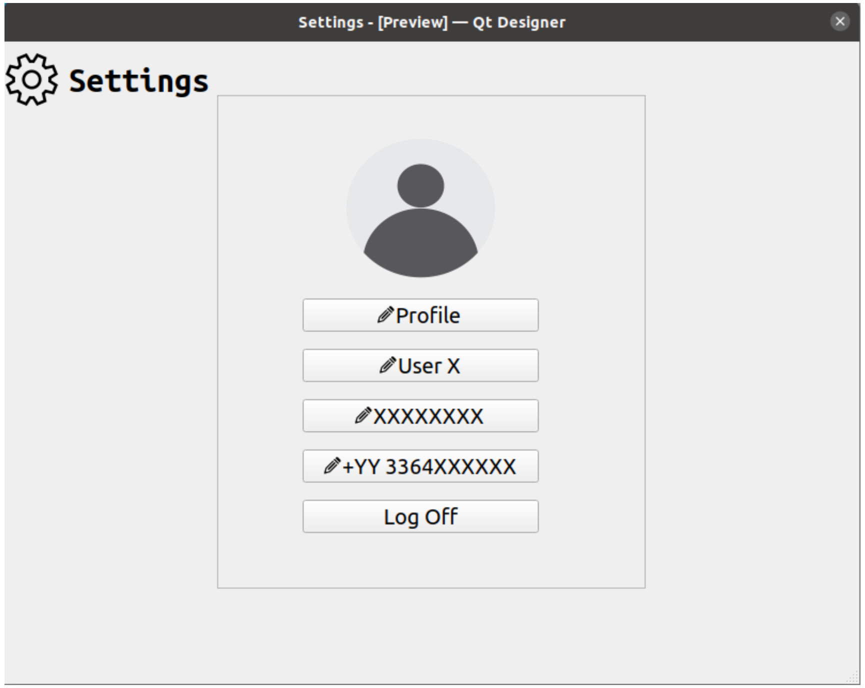


Figure 28–GUI first design settings

The settings page has the option to view and edit user details. They could log out or deactivate their account from here.

Testing and Implementation phase

Lemon App Test

Test Plan:

First, I would like to test whether my login screen buttons are clickable and could manage to jump between pages. They should be able to return to the same page if the cancel button is clicked on corresponding pages.

Then, I would like to test whether the sign-in page works correctly with response to the data entered. It should validate the registered user and verify them successfully. Once logged in Chat GUI should show up.

On the chat page, the message should be delivered and displayed in the chat window. Users should be able to change between different channels (i.e., chats/groups). Users should be able to search the contacts and send media files. The user should be able to navigate to the settings page.

On the settings page, user details should be displayed. It should be editable. Provisions for navigating back to the chat page, log out, and deactivate the account should be provided.

On the Register account page, a user should be able to enter their details. The system should validate the entered data with the existing data and look for errors. If there is an error, an error message should pop up. If not, the system should proceed to the chat page.

1. Provide images after each test result to show how the pages work.

ID	01_Login
Test Type	Qualitative
Description	The system should be able to navigate to different windows once the corresponding buttons are clicked.
Success Criteria	Successful navigation between pages.
Equipment(s) Requires	A system with Ubuntu Linux OS Qt Creator
Instructions	1. Click on the “Sign in” button. 2. GUI should navigate to a sign-in page. 3. Click the “Cancel” button to return to the welcome page. 4. Click the “Register” button. 5. GUI Should navigate to the register account page. 6. Click the “Cancel” button to return to the welcome page.
No. of attempts	
Failure/corrections	

Engineer/technician	
Result	
Date	

ID	02_ Sign in
Test Type	Qualitative
Description	The system should be able to verify user account details if created and allow the registered user to log in successfully
Success Criteria	Successful validation of user data with the existing data.
Equipment(s) Requires	A system with Ubuntu Linux OS Qt Creator
Instructions	1. Enter “Username” and “Password”. 2. Click the “Sign in” button. 3. The system should validate and check for registered users. 4. If a user is verified, System should navigate to

	the chat page. 5. If a user is not registered or entered incorrect details, the system should respond with an error pop-up message.
No. of attempts	
Failure/corrections	
Engineer/technician	
Result	
Date	

ID	03_ Chat Page
Test Type	Qualitative
Description	After successful login, the chat window should open. The message typed in the “Type” text box should be displayed in the label placed in the chat window. When the “settings” button is clicked, the system should navigate to the settings page.

Success Criteria	1. Successful message displayed in the chat window. 2. Should be able to create a new contact and delete existing contacts. 3. Should be able to use the search bar. 4. Successful navigation to the settings page.
Equipment(s) Requires	A system with Ubuntu Linux OS Qt Creator
Instructions	1. Type a message in the “type” text box. 2. Given message should be displayed in the chat window. 3. Click the “Settings” button on the top right corner. 4. The system should navigate to the settings page. 5. Type on the “Search” text box to search your available contacts.
No. of attempts	

Failure/corrections	
Engineer/technician	
Result	
Date	

ID	04_ Settings Page
Test Type	Qualitative
Description	When the settings page is opened, the system should get user details and display them on the corresponding boxes. The user should be able to navigate back to the chat window. They should have the provision to log out or deactivate their account.
Success Criteria	1. The system should display user details and profile pictures. 2. Navigation from the settings page to the chat page. 3. Users log out and deactivate account provision. 4. User information is editable.

Equipment(s) Requires	A system with Ubuntu Linux OS Qt Creator
Instructions	1. Edit Details in and press the “Confirm” button to change details. 2. Enter current user ID and Password to Deactivate the account. 3. “Log out” and “back” should respond accordingly.
No. of attempts	
Failure/corrections	
Engineer/technician	
Result	
Date	

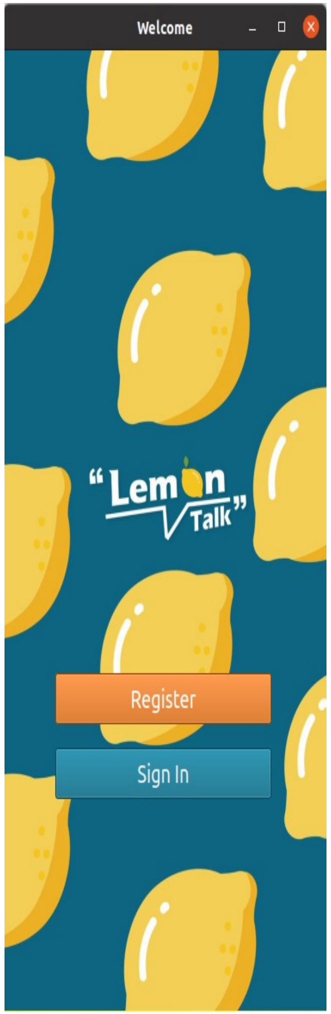
ID	05_ Register
Test Type	Qualitative
Description	When the user clicks the “Register” button, the system should navigate to the register page. The user information should be stored. The system should validate passwords entered and existing phone numbers. The system should display a pop-up error message if the phone number exists in another account and if the password entered is not as same as confirm password.
Success Criteria	1. Successful navigation form login to register page. 2. Successful validation of existing phone number with entered phone number. 3. Successful verification of password and confirm password.

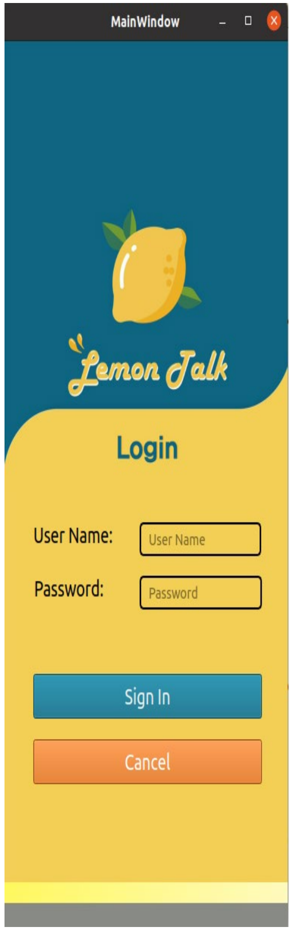
Equipment(s) Requires	A system with Ubuntu Linux OS Qt Creator
Instructions	1. Click the” Register” button from the login page. 2. Register window opens. 3. Enter required details. 4. Click create an account.
No. of attempts	
Failure/corrections	
Engineer/technician	
Result	
Date	

Test result

ID	01_Login
Test Type	Qualitative
Description	The system should be able to navigate to different windows once the corresponding buttons are clicked.
Success Criteria	Successful navigation between pages.
Equipment(s) Requires	A system with Ubuntu Linux OS Qt Creator
Instructions	1. Click on the "Sign in" button. 2. GUI should navigate to a sign-in page. 3. Click the "Cancel" button to return to

	the welcome page. 4. Click the "Register" button. 5. GUI Should navigate to the register account page. 6. Click the "Cancel" button to return to the welcome page.
No. of attempts	5
Failure/corrections	Nil
Engineer/technician	Annamalai Selvarajan
Result	Pass
Date	11/06/2021



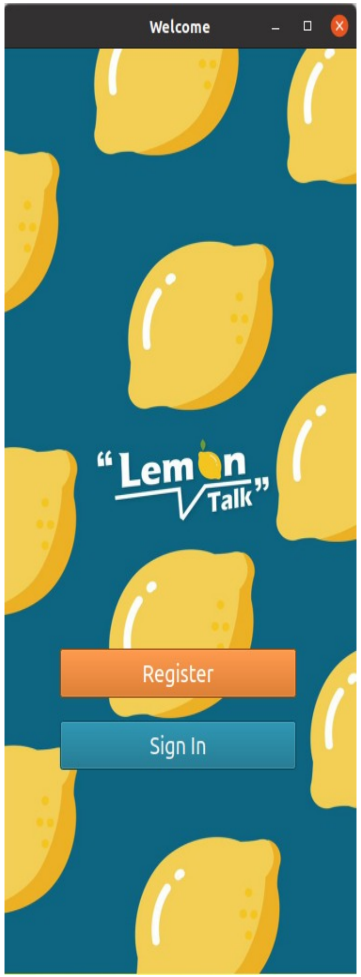


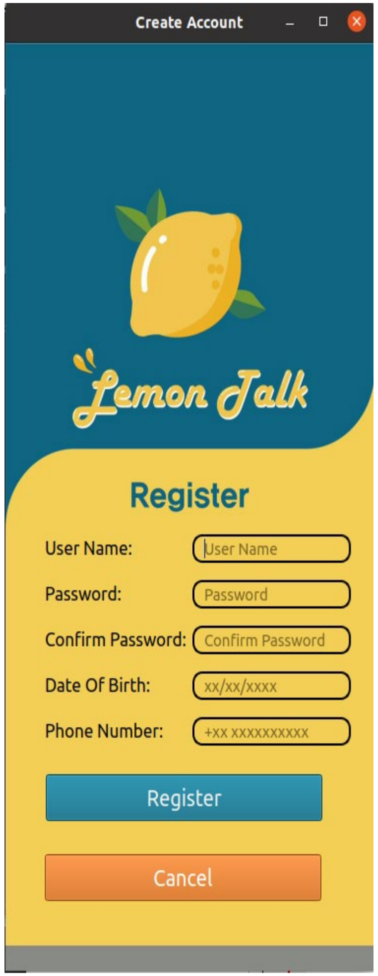
Login

User Name: User Name

Password: Password

Navigation between sign-in and welcome page





Register

User Name: User Name

Password: Password

Confirm Password: Confirm Password

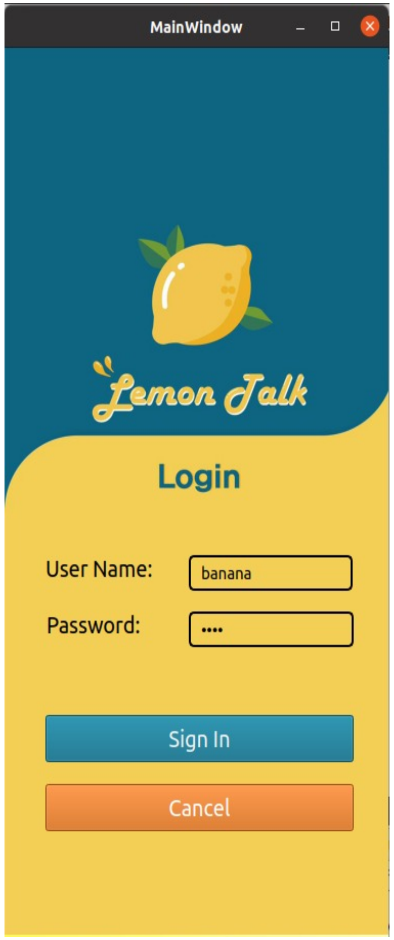
Date Of Birth: xx/xx/xxxx

Phone Number: +xx xxxxxxxxxx

Navigation between register and welcome page

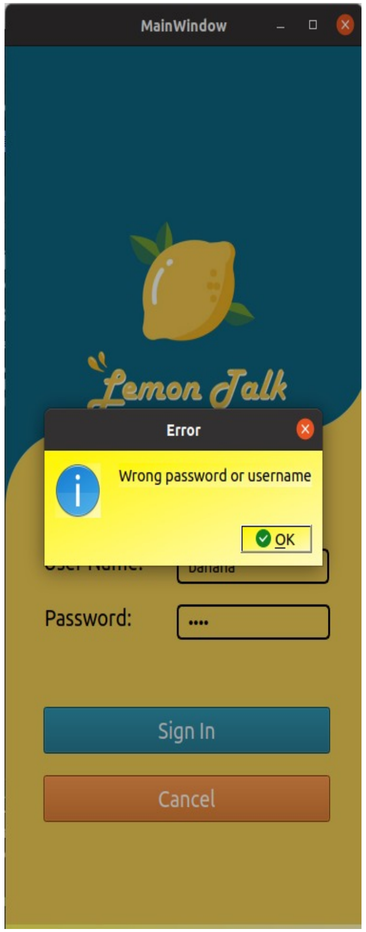
ID	02_ Sign in
Test Type	Qualitative
Description	The system should be able to verify user account details if created and allow the registered user to log in successfully
Success Criteria	Successful validation of user data with the existing data.
Equipment(s) Requires	A system with Ubuntu Linux OS Qt Creator
Instructions	1. Enter “Username” and “Password”. 2. Click the “Sign in” button. 3. The system should validate and check for registered users. 4. If a user is verified, System should navigate to the chat page.

	5. If a user is not registered or entered incorrect details, the system should respond with an error pop-up message.
No. of attempts	10
Failure/corrections	Nil
Engineer/technician	Annamalai Selvarajan
Result	Pass
Date	11/06/2021



User Name: banana

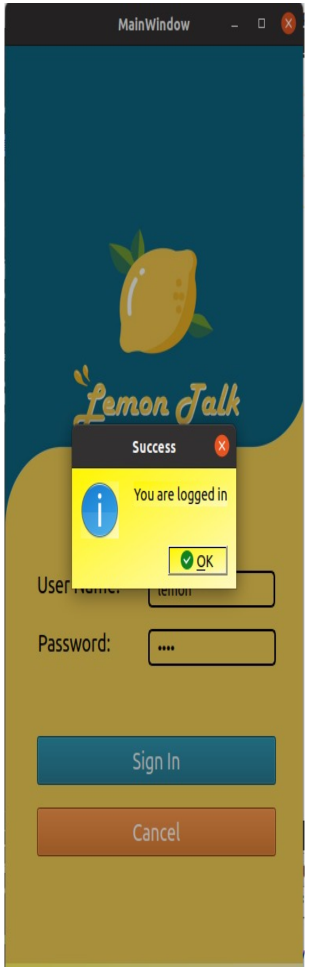
Password: ••••



Wrong password or username

Password: ••••

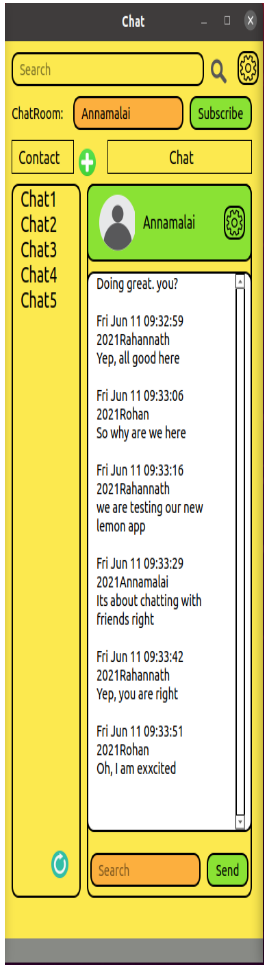
If wrong details are entered



You are logged in

User Name: lemon

Password: ••••



ChatRoom: Annamalai

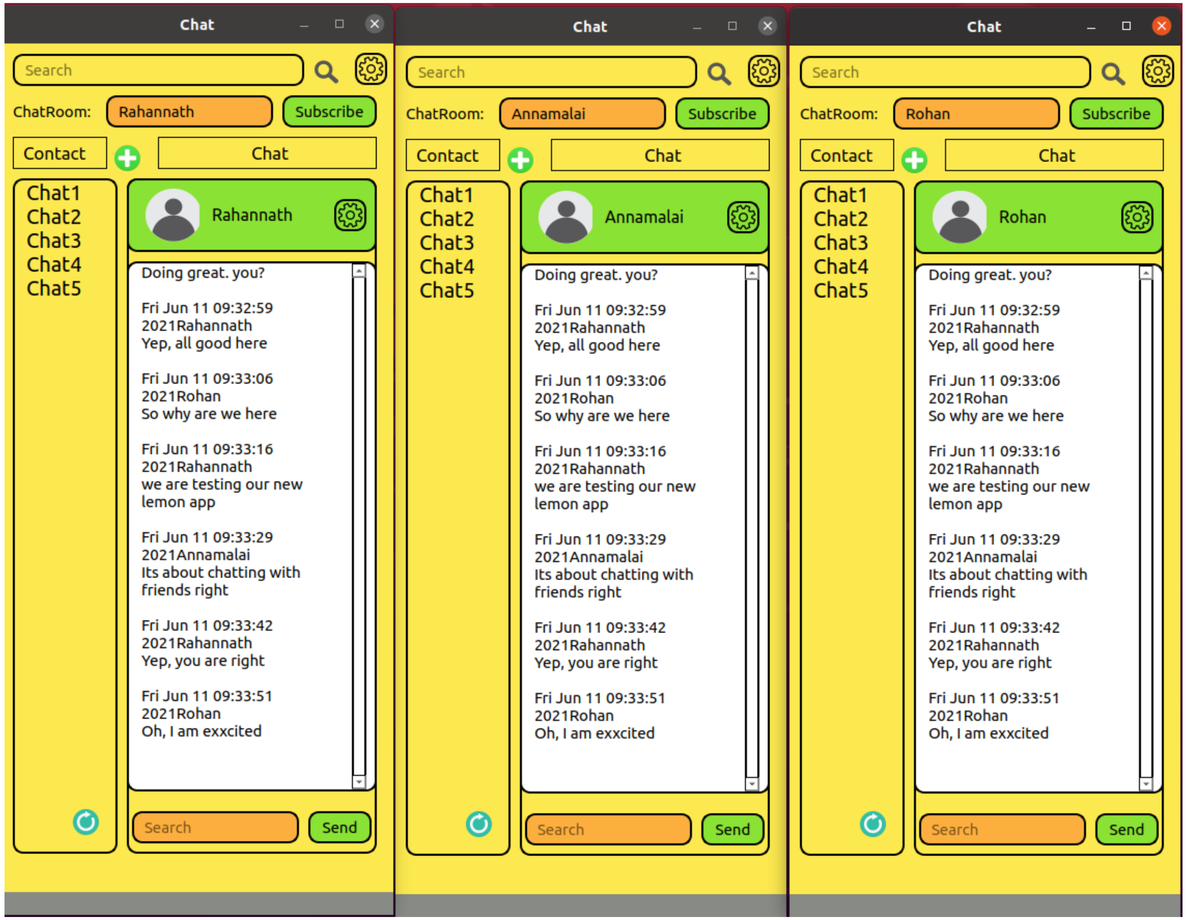
```
Doing great. you?

Fri Jun 11 09:32:59
2021Rahannath
Yep, all good here

Fri Jun 11 09:33:06
2021Rohan
So why are we here

Fri Jun 11 09:33:16
2021Rahannath
we are testing our new
lemon app

Fri Jun 11 09:33:29
2021Annamalai
Its about chatting with
friends right

Fri Jun 11 09:33:42
2021Rahannath
Yep, you are right

Fri Jun 11 09:33:51
2021Rohan
Oh, I am exxcited
```

If Correct details are entered

ID	03_ Chat Page
Test Type	Qualitative
Description	After successful login, the chat window should open. The message typed in the “Type” text box should be displayed in the label placed in the chat window. When the “settings” button is clicked, the system should navigate to the settings page.
Success Criteria	5. Successful message displayed in the chat window. 6. Should be able to create a new contact and delete existing contacts. 7. Should be able to use the search bar. 8. Successful navigation to the settings page.
Equipment(s) Requires	A system with Ubuntu Linux OS Qt Creator
Instructions	1. Type a message in the “type” text box. 2. Given message should be displayed in

	the chat window. 3. Click the "Settings" button on the top right corner. 4. The system should navigate to the settings page. 5. Type on the "Search" text box to search your available contacts.
No. of attempts	20
Failure/corrections	1. Messages are being delivered successfully without any issues. But the user must subscribe to each other to begin a chat. It is not auto subscribed as we create a new contact. 2. Could not choose individual chat members. 3. Could not add or remove contacts. 4. Could not add or delete groups. 5. Auto subscription between users should be implemented. 6. Could not use the search bar to find

	contacts.
Engineer/technician	Annamalai Selvarajan
Result	Correction required.
Date	11/06/2021



ChatRoom: Rahannath

Rahannath

Doing great. you?

Fri Jun 11 09:32:59 2021Rahannath
Yep, all good here

Fri Jun 11 09:33:06 2021Rohan
So why are we here

Fri Jun 11 09:33:16 2021Rahannath
we are testing our new lemon app

Fri Jun 11 09:33:29 2021Annamalai
Its about chatting with friends right

Fri Jun 11 09:33:42 2021Rahannath
Yep, you are right

Fri Jun 11 09:33:51 2021Rohan
Oh, I am exxcited

ChatRoom: Annamalai

Annamalai

Doing great. you?

Fri Jun 11 09:32:59 2021Rahannath
Yep, all good here

Fri Jun 11 09:33:06 2021Rohan
So why are we here

Fri Jun 11 09:33:16 2021Rahannath
we are testing our new lemon app

Fri Jun 11 09:33:29 2021Annamalai
Its about chatting with friends right

Fri Jun 11 09:33:42 2021Rahannath
Yep, you are right

Fri Jun 11 09:33:51 2021Rohan
Oh, I am exxcited

ChatRoom: Rohan

Rohan

Doing great. you?

Fri Jun 11 09:32:59 2021Rahannath
Yep, all good here

Fri Jun 11 09:33:06 2021Rohan
So why are we here

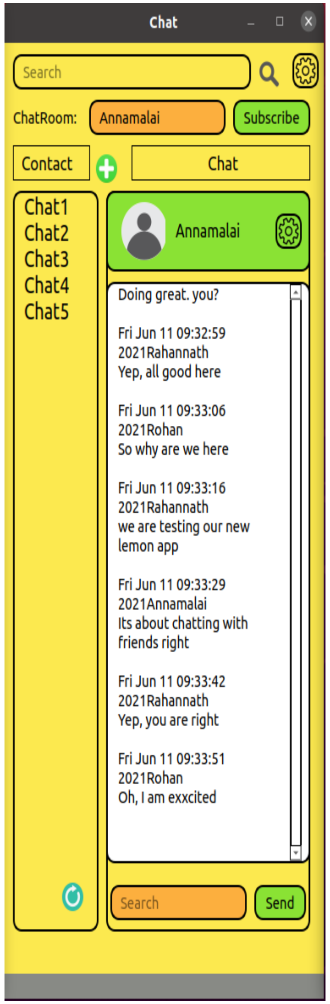
Fri Jun 11 09:33:16 2021Rahannath
we are testing our new lemon app

Fri Jun 11 09:33:29 2021Annamalai
Its about chatting with friends right

Fri Jun 11 09:33:42 2021Rahannath
Yep, you are right

Fri Jun 11 09:33:51 2021Rohan
Oh, I am exxcited

Some of the test results for messaging



ChatRoom: Annamalai

Annamalai

Doing great. you?

Fri Jun 11 09:32:59 2021Rahannath
Yep, all good here

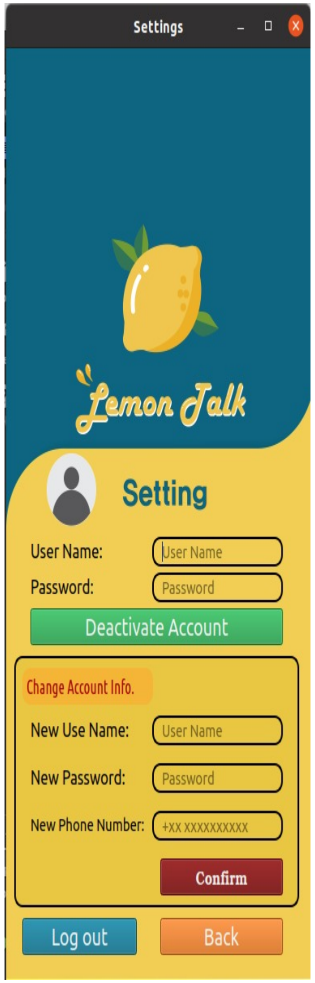
Fri Jun 11 09:33:06 2021Rohan
So why are we here

Fri Jun 11 09:33:16 2021Rahannath
we are testing our new lemon app

Fri Jun 11 09:33:29 2021Annamalai
Its about chatting with friends right

Fri Jun 11 09:33:42 2021Rahannath
Yep, you are right

Fri Jun 11 09:33:51 2021Rohan
Oh, I am exxcited

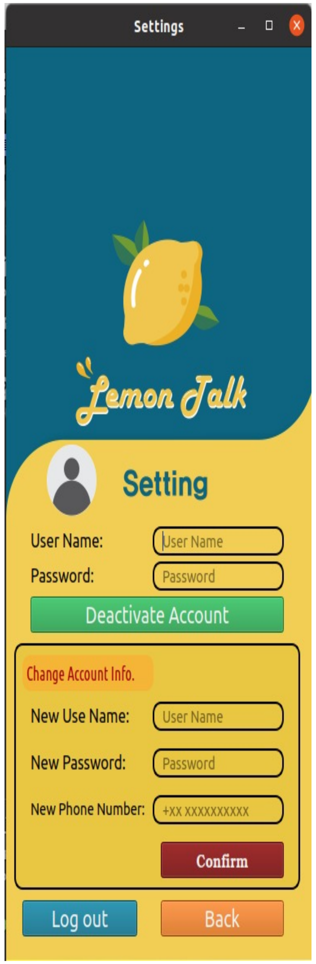


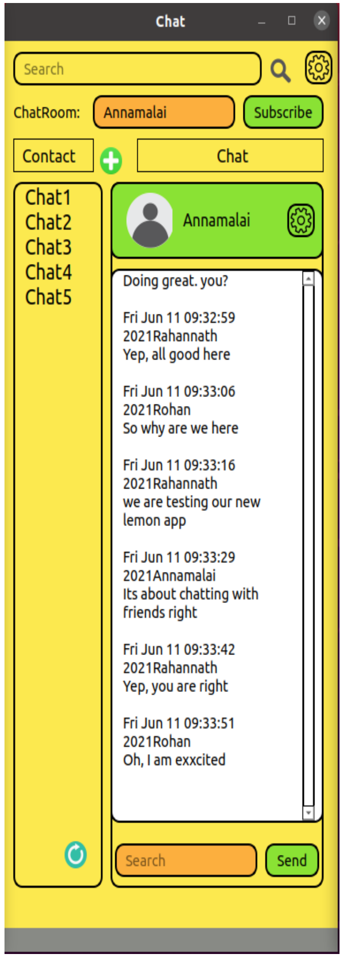
Navigation between settings and chat page

ID	04_ Settings Page
Test Type	Qualitative
Description	When the settings page is opened, the system should get user details and display them on the corresponding boxes. The user should be able to navigate back to the chat window. They should have the provision to log out or deactivate their account.
Success Criteria	1. The system should display user details and profile pictures. 2. Navigation from the settings page to chat page. 3. Users log out and deactivate account

	provision. 4. User information is editable.
Equipment(s) Requires	A system with Ubuntu Linux OS Qt Creator
Instructions	1. Edit Details in and press the “Confirm” button to change details. 2. Enter current user ID and Password to Deactivate the account. 3. “Log out” and “back” should respond accordingly.
No. of attempts	5
Failure/corrections	1. When wrong details are entered, an error message should be shown. 2. When changing the user password, his confirm password column also should be changed accordingly in the database.
Engineer/technician	Annamalai Selvarajan

Result	Correction required.
Date	11/06/2021





Doing great. you?

Fri Jun 11 09:32:59 2021Rahannath
Yep, all good here

Fri Jun 11 09:33:06 2021Rohan
So why are we here

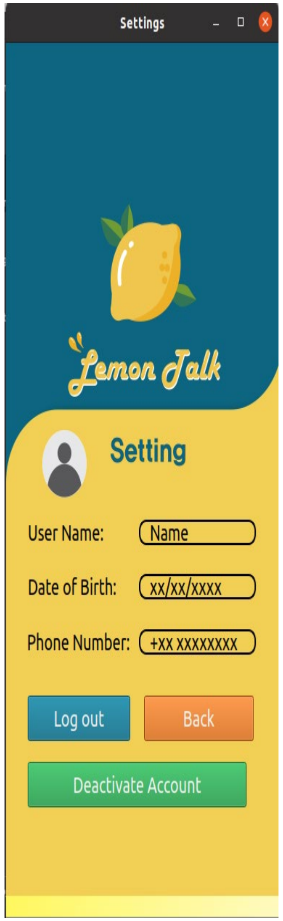
Fri Jun 11 09:33:16 2021Rahannath
we are testing our new lemon app

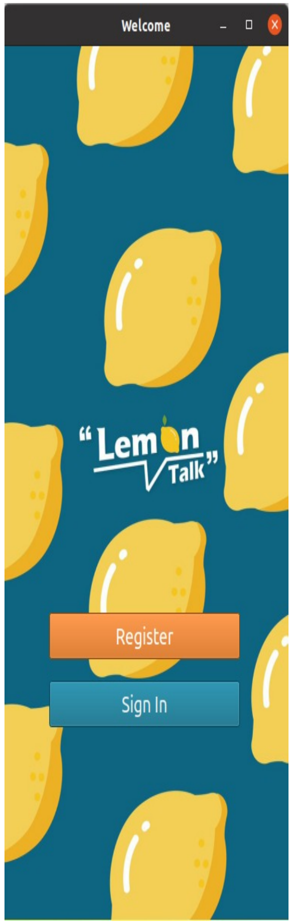
Fri Jun 11 09:33:29 2021Annamalai
Its about chatting with friends right

Fri Jun 11 09:33:42 2021Rahannath
Yep, you are right

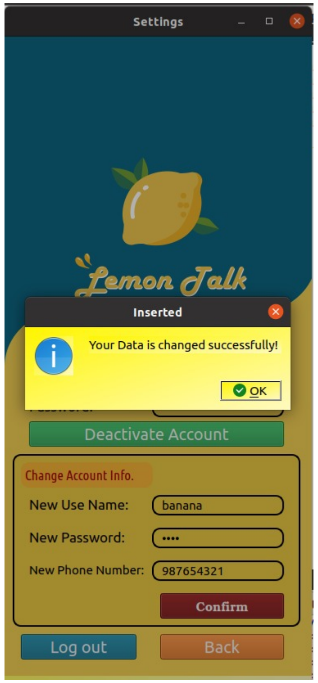
Fri Jun 11 09:33:51 2021Rohan
Oh, I am exxcited

Navigation between chat and settings page





Navigation between settings and login page



Settings

Inserted

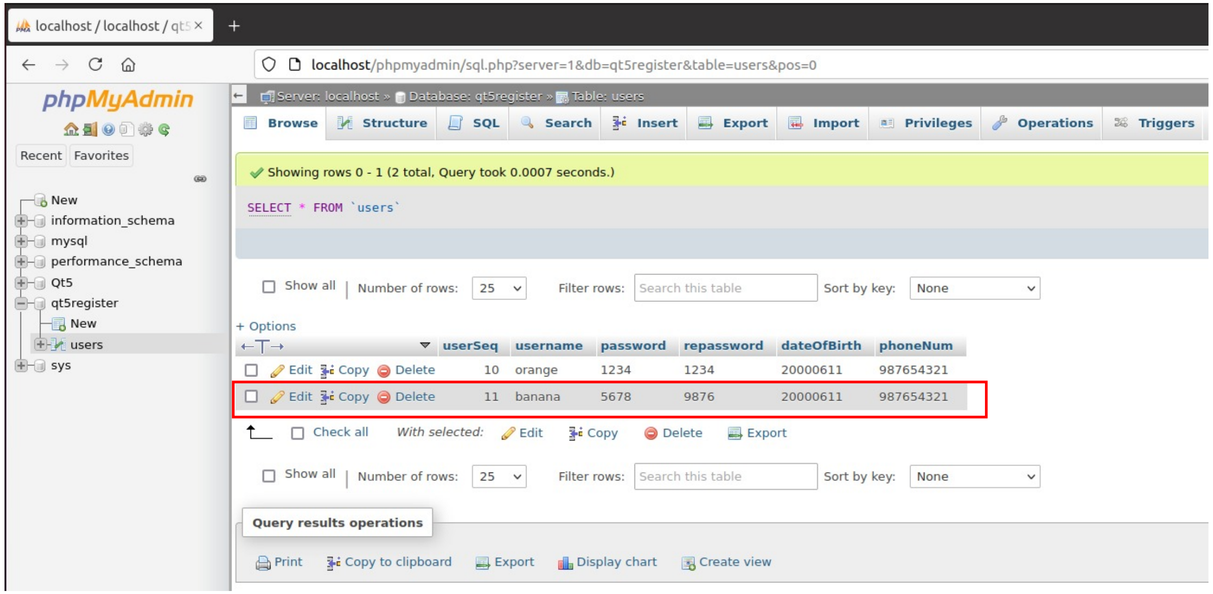
Your Data is changed successfully!

Change Account Info.

New Use Name: banana

New Password: ••••

New Phone Number: 987654321



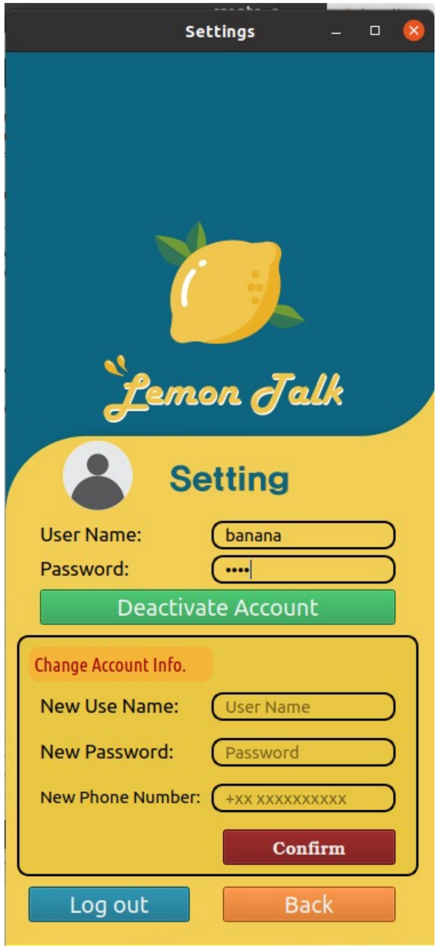
✓ Showing rows 0 - 1 (2 total, Query took 0.0007 seconds.)

```
SELECT * FROM `users`
```

□ Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

	userSeq	username	password	repassword	dateOfBirth	phoneNum
□ Edit Copy Delete	10	orange	1234	1234	20000611	987654321
□ Edit Copy Delete	11	banana	5678	9876	20000611	987654321

After changing data successfully



Settings

Setting

User Name: banana

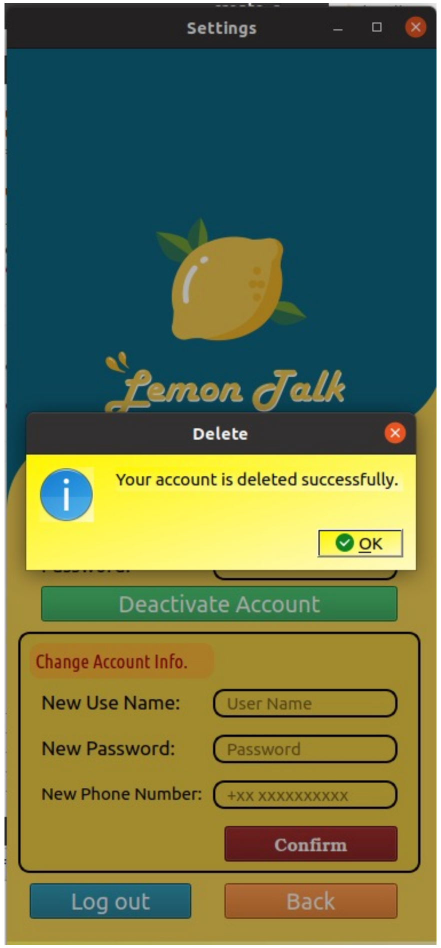
Password: ••••

Change Account Info.

New Use Name: User Name

New Password: Password

New Phone Number: +xx xxxxxxxxxx



Settings

Delete

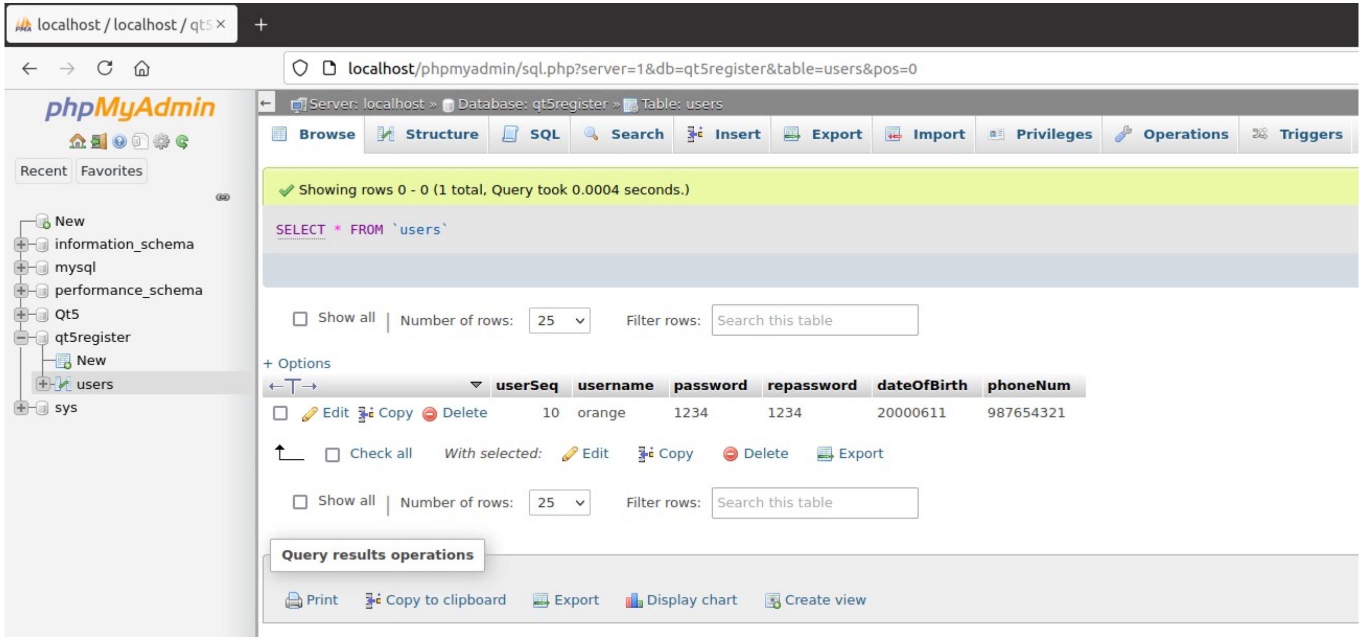
Your account is deleted successfully.

Change Account Info.

New Use Name: User Name

New Password: Password

New Phone Number: +xx xxxxxxxxxx



✓ Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

```
SELECT * FROM `users`
```

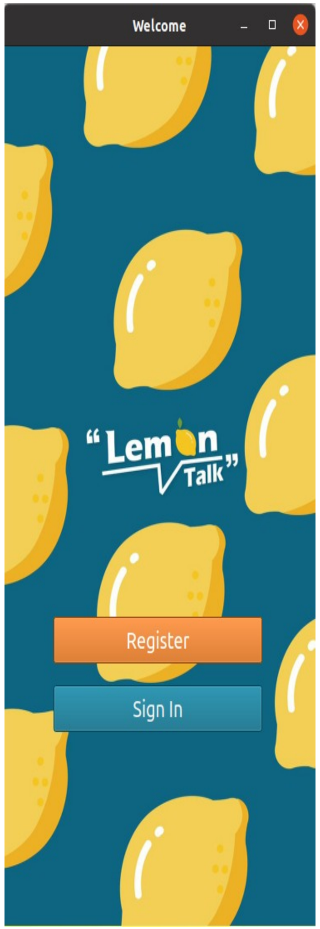
	userSeq	username	password	repassword	dateOfBirth	phoneNum
□ Edit Copy Delete	10	orange	1234	1234	20000611	987654321

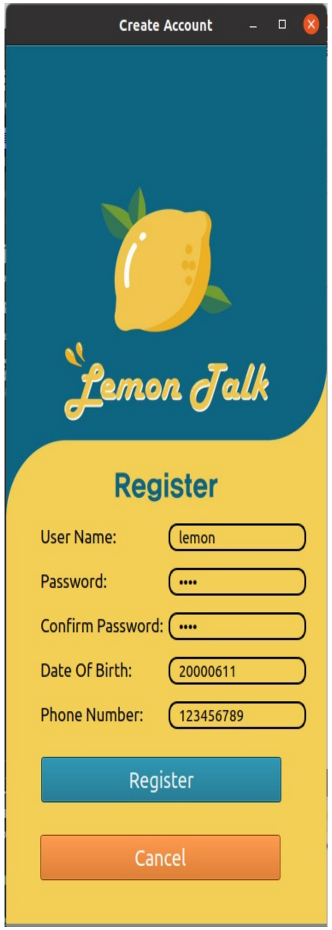
After deactivating account successfully

ID	05_ Register
Test Type	Qualitative
Description	When the user clicks the “Register” button, the system should navigate to the register page. The user information should be stored. The

	system should validate passwords entered and existing phone numbers. The system should display a pop-up error message if the phone number exists in another account and if the password entered is not as same as confirm password.
Success Criteria	1. Successful navigation form login to register page. 2. Successful validation of existing phone number with entered phone number. 3. Successful verification of password and confirm password.
Equipment(s) Requires	A system with Ubuntu Linux OS Qt Creator
Instructions	1. Click the” Register” button from the login page. 2. Register window opens. 3. Enter required details. 4. Click create an account.
No. of attempts	5

Failure/corrections	Nil
Engineer/technician	Annamalai Selvarajan
Result	Pass
Date	11/06/2021





User Name: lemon

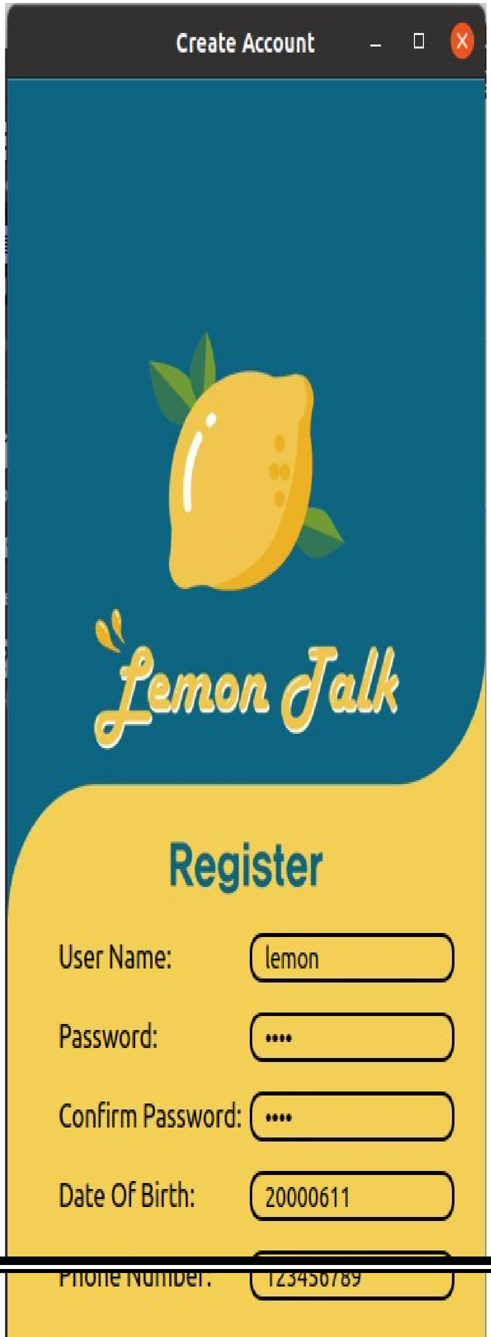
Password: ••••

Confirm Password: ••••

Date Of Birth: 20000611

Phone Number: 123456789

Navigation between login and register page



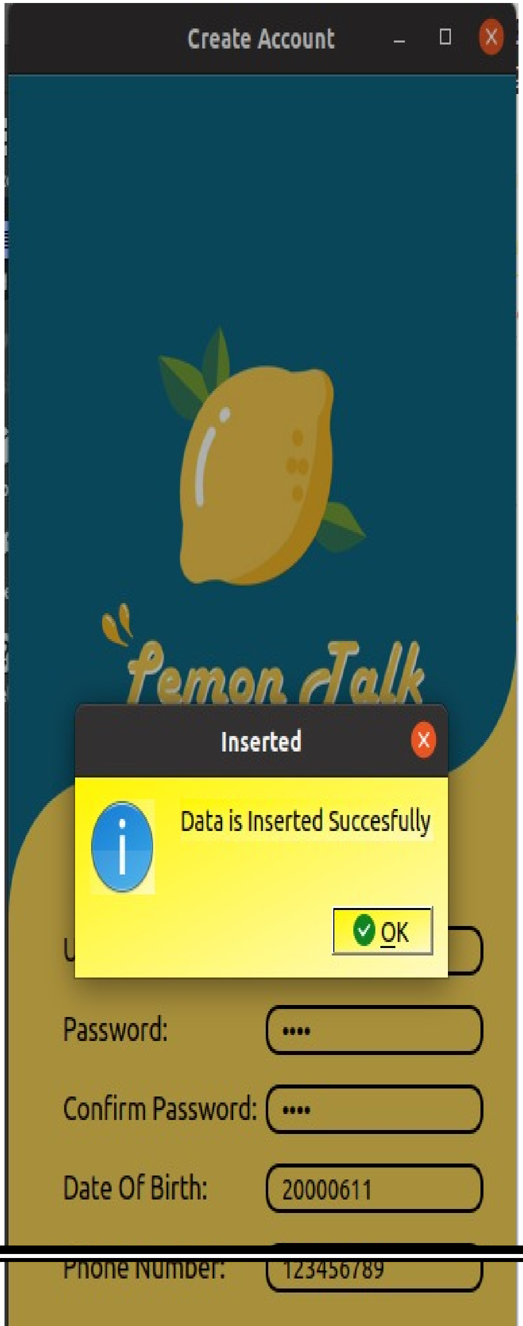
User Name: lemon

Password: ••••

Confirm Password: ••••

Date Of Birth: 20000611

Phone Number: 123456789



Data is Inserted Succesfully

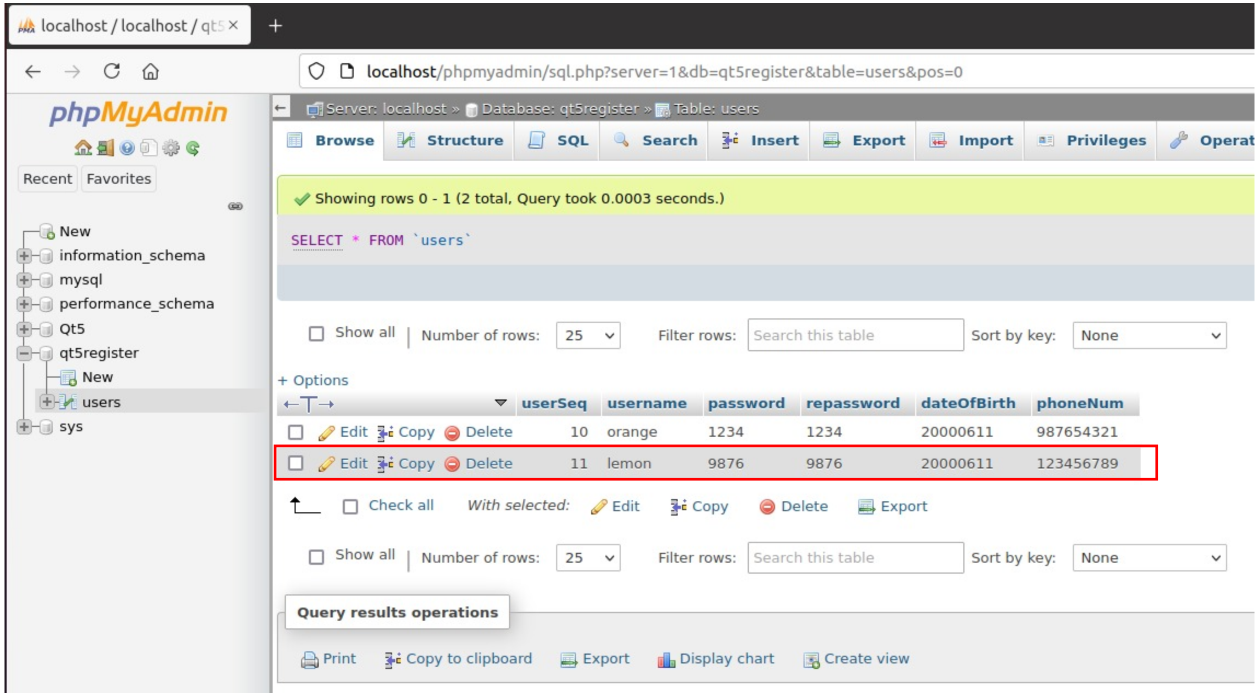
Password: ••••

Confirm Password: ••••

Date Of Birth: 20000611

Phone Number: 123456789

If all entered details are validated successfully

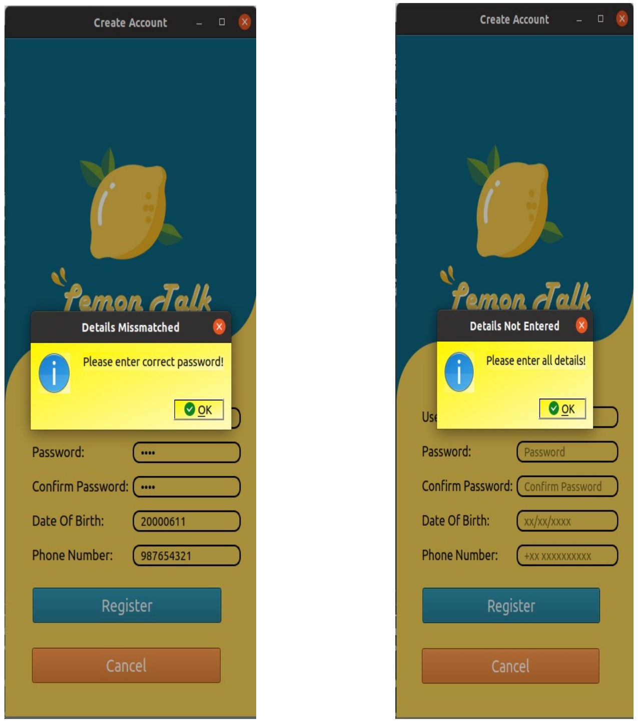


✓ Showing rows 0 - 1 (2 total, Query took 0.0003 seconds.)

```
SELECT * FROM `users`
```

	userSeq	username	password	repassword	dateOfBirth	phoneNum
□ Edit Copy Delete	10	orange	1234	1234	20000611	987654321
□ Edit Copy Delete	11	lemon	9876	9876	20000611	123456789

Database after creating an account



Details Missmatched

Please enter correct password!

Password: ••••

Confirm Password: ••••

Date Of Birth: 20000611

Phone Number: 987654321

Details Not Entered

Please enter all details!

Password: Password

Confirm Password: Confirm Password

Date Of Birth: xx/xx/xxxx

Phone Number: +xx xxxxxxxxxx

System validation before creating an account.

Links

Git hub repo:

https://olympuss.ntu.ac.uk/t0116478/Group40_SDI_Chat_App.git.

Conclusion and future works

In these trying types of Covid 19, where everything we do has slightly changed to make the lives of everyone easier, the importance of communication with one another cannot be overshadowed and that is why we created our messaging platform, to be a medium where people can communicate with each other.

From the planning to the design, then the coding and the implementation which produced our application had various risks involved which the team considered and minimized those risks.

Our biggest strength with our application is our coding and implementation which was handled carefully but other features could be added but could not be due to time. Our application is very easy to use and understand and makes sure it is not complicated for the user to operate on.

References

- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=EkjaiDsiM-Q&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA.
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=eS7ank-qFjg&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=2
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=5JVLO8yBMXA&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=3
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=Y1c-ieVO-UY&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=4
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=F56fSKoNCtk&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=5
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=xJdxE_7IBsU&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=6
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=gWa2rqe8l6E&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=7
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=y9Zx_FJBC1U&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=8
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=VigUMAfE2q4&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=9
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=6_elY8O20I8&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=10
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=CTbpS0PN-JQ&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=13
- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=2YRAJt-LbkM&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA&index=20

- ProgrammingKnowledge. (2021). Retrieved 13 June 2021, from https://www.youtube.com/watch?v=tY6nW3Wm3NE&list=PLS1QulWo1RIZiBcTr5urECberTlTj7gjA&index=22
- Nottingham Trent University (2021). Retrieved 13 June 2021, from https://now.ntu.ac.uk/d2l/le/content/711579/viewContent/4572016/View

Individual Reflection

Project Manager: As a project manager, I enjoy making processes to work on our project. I have a responsibility to make a schedule for our meeting and assign the work to our team members. Before having our meeting, also we planned for our project and made a Gantt chart. I think to make good progress, it is important to have well planned and every team member's contribution. While doing the meeting, we divided our tasks for each team member so that all team members can contribute equally to our project. Also, we discuss together our chatting app name, design, concept, and functionality. For making our project, I give contributions: a design for our app background images (lemon talk's login, register, setting page background image), implement the code for connecting with database (users can save their database information, edit their information, delete their account information), Setting page, implement the code for chatting page using MQTT. Overall, I think our chatting app project is well finished. Even though we could not finish them on time as the first submission because we had some situation. However, I felt like I had enjoyed making progress to see our app getting better. Our application is not perfect and needs some improvement in many aspects such as adding members on contact and sending emoji and change the user picture, etc. However, I think we have implemented for main functionality for the chatting app enough. Also, I think additionally we work hard to design for our app to look pretty and connecting with the database. While doing our project, I learned lots of things like teamwork is important. Also, it was a great chance to learn how I can manage team members in better ways and what is good leadership.

Software tester: I joined this group a bit late. I was assigned to some other group. Due to some reasons, I was asked to be a Software Tester of SDI Group 40. When I joined the group, almost 80% work for the first submission was done. However, I gave my input as much as I can. After that, the real fun begins. We were asked to do all the diagrams required for this project. We had a great time and a lot of fun in completing those work and were able to submit it. Though our work was not perfect, we could learn something from our mistakes. Moving on, the biggest challenge on our way. It is none other than coding the app. Since I designed the GUI for this app, I had better knowledge about its navigations. After completing lots of tutorials and gathering all required support from my teammates, I could finish our final GUI output with actual code running on the application. I could achieve this with the help of my teammates. They gave me enough ideas, confidence, and motivation to complete the process. Once the programming is done, I am supposed to test the program on every possible aspect. I have conducted several tests and prepare a document about it. With the help of that, we stepped into the final phase of our project. Unfortunately, we were not lucky enough to complete the project in time. I got affected by covid-19 and could not contribute well with the team.I am feeling guilty about that. However, we got a second chance. We took our time and researched many things. We could complete the login server setup completely. However, with little time left to submit, we could not complete our chat functions. However, we can still chat with people by just typing their names and subscribing to them. However, it is not linked with the database. So, there could be "n" number of subscribers chatting in the same window. Anyways, our project is a great success in our mind because it is a gift to have a team like this. I completely enjoy working and learning with them. We learned a lot together and succeeded together. I am proud and honored to be a member of this team. Thank you for all your support event when I was affected by Covid-19. Thank you so much.

Software Developer: In attempting this coursework, I was overcome by the fear of implementing such a huge project despite having little knowledge of a c++ GUI. I made it a goal to get ahead of my fears for the sake of my hard-working group members and attempt to produce the best results. All the group members came from different parts of the globe hence it made all the members shy at first. Realizing this I made sure I made myself as approachable as possible and tried to get the other members to participate as much as I could in my way. Over time we started cooperating better as a team which made it easy to carry out the coursework.

Software Architect: As the software architect, I have enjoyed deciding how our application should turn out. I had the responsibility of using diagrams to give my group members an idea of how our application should turn out. Diagrams such as the Use case diagram, Sequence diagram etc were used in the making of our application. My team members responded well and understood the concepts of the diagrams and well as give contributions to the sign in and register pages of our application. We also had meetings which talked about each team member's contribution which made us make a lot of progress, decide and act on those various contributions such as our application name, designs, concepts, and its functionality. Overall, I believe our group did very well in these trying times especially with how our application turned out. It is not perfect but through hard work, determination and perseverance we made it work. The group worked immensely hard on our design and connection with our database. This group taught me the benefits of teamwork, communication and good leadership skills.

Appendix i

Meetings

Meeting 1		
Platform: zoom	**Date:** 12/02/2021	**Duration:** 2 Hours
Attendees		
Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)		
Achievements		
Discussion and submission for each role the project plan and the responsibilities of each person. Research and discussion about possible tools to use.		

[Table1: Meeting 1]

Meeting 2		
Location: zoom	**Date:** 19/02/2021	**Duration:** 1 Hours
Attendees		
Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)		
Achievements		
Group Feedback for the project manager deliverable (project plan) and brief explanation of each team member task. More disccusion about the tools and libraries to be used for the software development.		

[Table2: Meeting 2]

Meeting 3		
Location: zoom	**Date:** 19/02/2021	**Duration:** 2 Hours
Attendees		
Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)		
Achievements		
Group Feedback for the project manager deliverable (project plan) and brief explanation of each team member task. More disccusion about the tools and libraries to be used for the software development.		

[Table3: Meeting 3]

Meeting 4		
Location: zoom	**Date:** 19/02/2021	**Duration:** 2 Hours
Attendees		
Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)		
Achievements		
Team leader discussed the future milestones that needed to be completed and took group members feedback in regard to the overall work process so far. Further discussions were then made in regards of code implementation and dividing the work.		

[Table4: Meeting 4]

Meeting 5		
Location: zoom	**Date:** 26/02/2021	**Duration:** 1 Hours
Attendees		
Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)		
Achievements		
Team reviewed the software developer's draft work and gave feedback where appropriate. Further discussions were then made in regard to the Beta release version and the final report.		

[Table5: Meeting 5]

Meeting 6		
Location: zoom	**Date:** 21/03/2021	**Duration:** 1:30 Hours
Attendees		

Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)
Achievements
Team reviewed the beta release version submitted in regard to the software developer's deliverable. Discussion on the final stages of the application were addressed and milestones were set.

[Table6: Meeting 6]

Meeting 7		
Location: zoom	**Date:** 15/04/2021	**Duration:** 1 Hours
Attendees		
Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)		
Achievements		
Casual team review of progress update.		

[Table7: Meeting 7]

Meeting 8		
Location: zoom	**Date:** 26/04/2021	**Duration:** 1 Hours
Attendees		
Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)		
Achievements		
Progress update was addressed. Additionally, further discussions regarding the final report and enhancing deliverables were discussed.		

[Table8: Meeting 8]

Meeting 9		
Location: zoom	**Date:** 26/04/2021	**Duration:** 2 Hours
Attendees		
Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)		
Achievements		
Testing assigned to software developer as software tester is unresponsive. Additionally, the group then discussed the release of the final version of the application.		

[Table9: Meeting 9]

Meeting 10		
Location: zoom	**Date:** 26/04/2021	**Duration:** 1:15 Hours
Attendees		
Bogyeong Kim(T0116478), Habeebullah Ladan(T0258313), Annamalai Selvarajan(N0881865), Oluwaseun Banjo(N0878755)		
Achievements		
Final review of the application and the report took place. Room for improvements (report) were addressed.		

[Table10: Meeting 10]

Class List

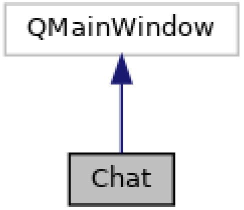
Here are the classes, structs, unions and interfaces with brief descriptions:

- C **Chat**
- C **Create_Account**
- C **MainWindow**
- C **Settings**
- C **Sign_In**

Chat

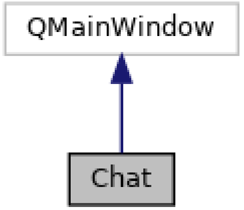
Chat Class Reference

Inheritance diagram for Chat:



[legend]

Collaboration diagram for Chat:



[legend]

Public Member Functions

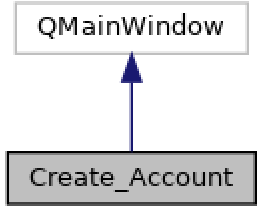
	Chat (QWidget *parent=nullptr)

The documentation for this class was generated from the following files:

- **chat.h**
- chat.cpp

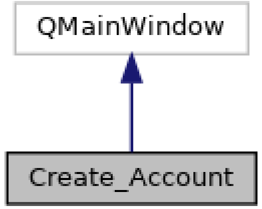
Create_Account Class Reference

Inheritance diagram for Create_Account:



[legend]

Collaboration diagram for Create_Account:



[legend]

Public Member Functions

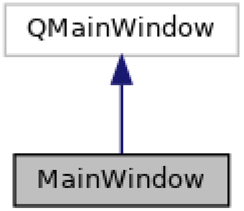
Create_Account (QWidget *parent=nullptr)

The documentation for this class was generated from the following files:

- create_account.h
- create_account.cpp

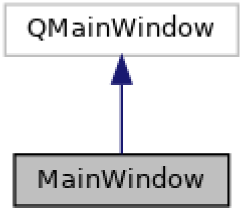
MainWindow Class Reference

Inheritance diagram for MainWindow:



[legend]

Collaboration diagram for MainWindow:



[legend]

Public Member Functions

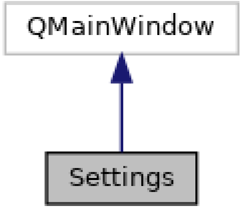
MainWindow (QWidget *parent=nullptr)

The documentation for this class was generated from the following files:

- mainwindow. h
- mainwindow.cpp

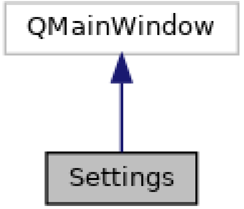
Settings Class Reference

Inheritance diagram for Settings:



[legend]

Collaboration diagram for Settings:



[legend]

Public Member Functions

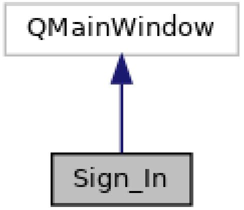
Settings (QWidget *parent=nullptr)

The documentation for this class was generated from the following files:

- **settings.h**
- settings.cpp

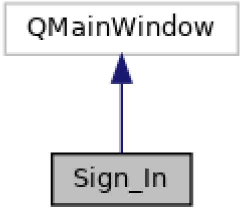
Sign_In Class Reference

Inheritance diagram for Sign_In:



[legend]

Collaboration diagram for Sign_In:



[legend]

Public Member Functions

Sign_In (QWidget *parent=nullptr)

The documentation for this class was generated from the following files:

- **sign_in.h**
- sign_in.cpp

Class Index

c | m | s

c

Chat

Create_Account

m

MainWindow

s

Settings

Sign_In

c | m | s

Class Hierarchy

Go to the textual class hierarchy

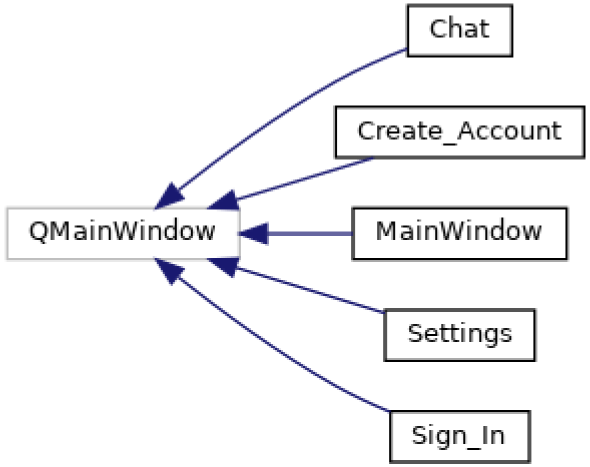


Figure 29–Class Hierarchy

Github Repository Link

https://olympuss.ntu.ac.uk/t0116478/Group40_SDI_finalCode_chatApp.