Software Design & Implementation Group 16 Coursework

Group 16

Milosz Bednarski – Project Manager

Jonathan Archer - Software Architect

Josh Wade – Software Developer

Lab Tutor

Salisu Yahaya

SOFT20091 Software Design and Implementation

Nottingham Trent University

Department of Computer Science

United Kingdom

May 2021

Table of Contents

Table of Figures

Figure of Tables

Abstract

The report starts with an introduction to current message exchange platforms and background research into the impact of Covid-19 on businesses and educational institutions, particularly working from home. Section 1 primarily focuses on the target market through gathering requirements and deciding on an appropriate methodology whilst considering timescales and risks that could hinder the outcome of the project. Methods of mitigating risks are analysed to ensure that the project successfully delivers a functional message exchange platform that meets the previously discussed requirements. Section 2 demonstrates how messages will be transferred using sequence diagrams, how user data will be stored on a database using a class diagram and how users will be able to log into the system using communication diagrams. Additionally, a deployment diagram is used to illustrate how messages will be sent using the TCP/IP protocol. UI mock-ups are also designed to visualise how the intended outcome of the system is going to be produced.

Version History for Application

Table 1 - Version History of our Application

Version	Issue Date	Stage	Changes	Author	Approved by
V1.0.0	28/02/2021	In development	Implemented a working file system for storing user data. When users register an account, the data will be stored using this system. When users try to log in, it will search through this system to try to log them in as long as the credentials entered match the credentials in the system. MQTT client has been created, allows users on a local network to message each other without the use of a broker. Users subscribe to a topic set by their account name so that the MQTT client listens to all messages published to that topic. Users can then publish messages to the recipient's topic to allow communication between the two people. Messages sent and received by the user do not show as being sent by two different people, they show as if the user is sending all of them.	Milosz Bednarski	Josh Wade (Software Developer)
V1.0.1	04/03/2021	In development	Converted the current system to make use of a database for user data storage. The MQTT client now uses multi-level topic selection. When the user subscribes to their topic, it listens to all topics that start with the subscribed topic name, then any additional levels, this allows the topic to state the client that is using it, the user who subscribed to the topic and then the user who published the message on the topic. At this stage, the topics were not being filtered, therefore, any messages a user would send to someone were not private to that chat. Sent and received messages are now distinguishable.	Milosz Bednarski	Josh Wade (Software Developer)
V1.0.2	08/04/2021	In development	Completed manage users functionality allowing the admin to delete the user from the system or to reset their password. Messaging page now uses the database to be able to communicate with other users. Topics are now filtered to allow for private messaging and group chat functionality has been added. When the UI page loads, the chat history of two people is read before being displayed.	Milosz Bednarski	Josh Wade (Software Developer)
V1.0.3	16/04/2021	In development	Completed manage groups functionality allowing the admin to delete the entire group or to remove members from the group. Admin manage users, groups and mods functionality has been implemented.	Milosz Bednarski	Josh Wade (Software Developer)
V1.0.4	24/04/2021	In development	Group chat message history is now enabled. Users can see their profile pictures at the top of the UI.	Milosz Bednarski	Josh Wade (Software Developer)
V1.0.5	1/05/2021	In development	Dark mode feature has been added as a colour scheme preference in the settings page. Logo for log off button has been added. Group chat admins and moderators now have permissions which allow them to be able to promote, demote, and remove users as well as delete the whole group.	Milosz Bednarski	Josh Wade (Software Developer)

Table 2 - Version History of our Report

Version	Issue Date	Stage	Changes	Author
V1	1/01/2021	In development	Creation of Document for Project definition submission (Deliverable 1)	Milosz Bednarski
V2	21/02/2021	In development	Addition of all UML diagrams accompanied by explanations.	Jonathan Archer
V3	18/03/2021	In development	Feedback applied to Project definition and UML diagrams.	Milosz Bednarski/ Jonathan Archer
V4	02/05/2021	In development	Applying feedback to the report, adding missing sections	Jonathan Archer, Milosz Bednarski, Josh Wade

Section 1 - Project Definition

1.0 Introduction

“Messaging apps are now over 20% bigger than social networks” (MediaKix, 2021). In this day and era chat applications are widely used by everyday people with WhatsApp being the most widely used messaging app in the world as stated by (SPECTRM, 2020). Businesses also use the benefits of chat applications, with 91% of businesses using at least two messaging apps as stated by (Mio, 2019). Especially with the recent pandemic the need for messaging/live chat applications has risen exponentially with Microsoft teams being the main choice for most businesses and organisations.

This report aims to identify the key stages for developing a chat application, along with identifying all the requirements and resources needed to fulfil the project. All key areas of development will be addressed with the different sections of the report illustrating the groups thought processes and engagement with the project. This report will also convey the problems faced during the development process and list the strategies and work arounds used to help overcome these challenges.

The stages consisting of planning, design, implementation and testing all consisting of a number of tasks and activities that were undertaken to realise the project. Planning involved identifying the resources of the project software, external libraries etc. The design phase started with a need to understand the requirements and what the needs of the client were. For this we used the MosCow analysis to further break down the given requirements into interpreted requirements. After this a risk assessment was needed to ensure all problems our group may face for the given project were addressed and planned for. Arguably the most important part was creating the time plan (gantt chart), this would be used along with other project management tools such as Trello to ensure our group knew exactly what to do and when tasks needed to be completed by. The design phase concluded with listing out assumptions made about the project and presenting the coding standards.

The implementation phase was initiated with the creation of different types of UML diagrams, such as use case, activity, class, sequence, deployment, communication, component, finite state machine (FSM), to be exact. These would be used to define the structure of the project and give the team a good start and outlook to how the program would be formed. The implementation then goes on to show the UI design concept. The second half of the Implementation stage illustrates/ represents after the development has finished. So a detailed run down of how to use the program is provided and details about the program are given (memory, concurrency, error handling etc).

1.1 Background Research

Throughout the project certain unfamiliar technologies has to be learnt in order to develop the chat application. Several processes needed to be established and research needed to be done to understand what software and main technologies would be needed in the design and implementation phases of the project.

UML Diagrams

The first main use of software being the Gantt chart creation. Our team choose to use ……. As it provided all the necessary tool available to the team. Many other online tools were used before to try and make the chart i.e TeamGannt.

The second challenge to overcome was the design and creation of the UML diagrams. Although software's such as papyrus had been recommended for all the diagram creation, it was soon realised this might not be the best point of call. Papyrus does in fact cater for the creation of all the main UML diagrams however through research and attempts to make diagram through the software it was discovered that several key features were missed out for some of the diagrams, for instance swim lanes for activity diagrams. Adding to the fact that the group found papyrus to be a troublesome software to use, the decision was made to use a collection of software instead of solely using papyrus. Online tools were used for the creation of some diagrams. These being Cacoo and VisualParadigm along with the use of papyrus.

IDE (Integrated Development Environment)

The main IDE used to create the project was decided to be QT creator. All the necessary tools were provided to create the application, along with a space to create the UI (User interfaces) and a place to add the code to the components of the application. Other IDEs were considered to be used along side of QT, for instance eclipse, but it was decided by the group that it would an unnecessary and problematic approach to switch between IDEs when there was in Our mind, no real need to use both.

Database

Originally the idea was that we would primarily focus on a text file based system where all user accounts and information would be stored on an individual text file. One users information is stored in one text file. However, proved to be fairly problematic as there would be too many files added and it would be harder to track and make changes to the user base on the chat application. Therefore, the group shifted its original ideas and instead focused on implementing an external database.

The recommended database based on it being free and easy to set up was a MyPhP Admin. But this proved to be troublesome as getting data to and from the database was slow and tedious which they do say in the website for hosting the database. So, the database was upgraded to an Amazon Web Service hosted MySQL database which was much easier to host, setup and much faster. Making additions to tables and adding user to the application through this method was proven to be much easier in the long run, even though setting up a database was a challenge in itself. Carrying on with textfiles would have provided other challenges later on in development.

Libraries

The libraries used in the creation of the application so far have been MQTT for messaging passing, STL for use with vectors although not very much and QT for pretty much everything. QT has been used to develop the UIs for the application and then QTs library of functions have been used throughout the development. DOXYGEN has been used for the automatic generation of documentation which QT provided easy access for. And Github has been used with QT for version control and source code storage.

Other Frameworks

Originally with the first implementation of the database we used docker in order to connect up the lamp server, which we would then need to start up whenever we wanted to run the application. We felt this was problematic and would impact development, so we switch to another approach that does not require a lamp server to be used. And so in our final application docker is not used.

1.2 Requirements List

Table 3 - Requirements Analysis

No	Priority	Requirement	Implication	Tasks	Completed (Yes/No)
1	MUST	MUST provide a friendly User Interface (UI).	The friendly UI will encourage more employees to want to use the application for communication within the workplace and will cause the application to be easy to use. This should be implemented in the form of a graphical user interface (GUI) which can be manipulated by the user to suit their personal needs.	T1: Create GUIs that follow the same style and colour scheme. T2: Allow users to hover over buttons to indicate that they are interactable. T34: Implement accessibility settings such as an option for dark mode.	Yes
2	MUST	Users MUST only access their space after the login;	Each user is assigned a personal login, this is to allow personal communication to and from specific people from within the company. This stops users from having access to conversations which they are not having, allowing confidentiality.	T3: Allow user to login using their username and password. T4: Add an option for the user to update passwords through a special word. T5: Implement validations that ensure that data is successfully saved to the database. T6: Open the home page after successfully retrieving data from the database.	Yes
3	MUST	Passwords MUST be saved securely locally;	As the system will contain sensitive information about the users, users will require a password in order to use the system and so those passwords will need to be keep securely in order to prevent a breach in the system.	T6: Encrypt passwords using a cipher. T5: Save the users data to the database after passing registration validations.	Yes
4	SHOULD	Users SHOULD be logged off automatically after a specific amount of time of inactivity;	Users should be logged off automatically after a certain amount of time, this is so that the servers and connection are not taken up by unnecessary traffic and this is a security feature to make sure another person can use the user's device to look through their account when the user is not around.	T52: Implement mouse and keyboard listener events that detect if the user is actively using the app. T53: Set the users activity status to ‘Offline’ after 15 minutes of them being idle.	Yes
5	MUST	Clients MUST NOT connect directly to other clients without a server or a broker;	A server or a broker is needed to allow clients to communicate over the internet.	T13: Implement a server/broker into the application which will be used to allow client to client communication.	Yes
6	MUST	A server or a broker MUST allow multiple authorised clients to connect to it;	The server needs to allow multiple users to access it at once otherwise there would not be the ability for users to talk to each other or talk to each other very effectively.	T13: Implement a server/broker into the application which will be used to allow client to client communication.	Yes
7	SHOULD	The client SHOULD list all the personal contacts in the contacts pane;	This feature would help users as they could simply just selected one of the contacts to message directly, rather than having to go through a whole bunch of different pages to get to the contacts. This is a more ease of use feature that most users would apricate.	T1: Create an ‘Accounts’ GUI which displays a list of users along side their active status and profile picture. T15: Allow the user to click any user and start messaging them.	Yes
8	COULD	The full history of the conversation COULD be listed when a specific contact is selected;	It is important for users to be able to see contact history so that other users don’t must repeat themselves, it's easy to see what's been said and important to help maintain a professional environment.	T11: Store all messages that the user has sent and received. When the client selects a user's name, the stored messages are retrieved from the database and displayed to the user.	Yes
9	MUST	The user MUST be able to create chatrooms (rooms with more than two contacts);	The user should be able to create multiple chat rooms where the people in the rooms can be selected by the user who creates them. This is important as most messaging applications use this feature.	T1: Create a page that shows the existing group chats that the user is in T18: Create a button on the group chats page that opens a window for the user to create a group chat. T19: Allow the user to give the group chat a name, add members to the group chat, give the members roles and then create the group chat.	Yes
10	MUST	The User that creates chat room MUST be promoted to Admin of that chat room;	When a user creates a group chat, they should immediately be made the admin of the chat room because they created the group.	T1: Create a page that shows the existing group chats that the user is in	Yes

				T18: Create a button on the group chats page that opens a window for the user to create a group chat. T20: When the user creates the group chat, automatically select them to be the admin before storing the data in the database.	
11	COULD	The Admin COULD promote and demote users to moderators in channels;	This will allow users to be able to control who is in a group chat. It will also allow admins to be able to make sure channels stay on topic.	T21: Allow the group admin to have access to a list of users and moderators in the group. T22: Allow the admin to select a user or moderator from the list and promote or demote them.	Yes
12	MUST	Moderators MUST inherit all the admin permissions however Moderators cannot demote the Admin;	This is important to make sure unauthorised people don't take over the application, but permissions do need to give to more people in order to maintain the application and make users are using the application appropriately.	T23: Add roles which have different permissions. T24: Give any moderators the same permissions as the group admin apart from the ability to demote the admin or themselves.	Yes
13	MUST	Users MUST be able to see the active users in the chat room;	It is important for the users to see who is currently using the chat room especially if the application is used by business where meetings are held on the chat rooms. This allows the users to feel as though they are talking to people in the application and allows them to expect a response if they can see who is currently active.	T27: Allow users to view all the active status of their contacts. T28: Show the active status of the recipient whilst messaging them. T29: Optionally, allow the view active users in the group chat.	No for group chats. Yes, for private messaging
14	COULD	Offline messages COULD be stored in the client-side and transmitted to the target user(s) once they are online;	It is extremely important for offline messages to be stored and sent to the target user as you don't want messages to be missed.	T11: Upload the message to the database. T16: When a user sends a message to an offline user, update the user's UI to show the message. T17: When the user comes online and opens the chat, retrieve the chat history which will contain the last transmitted message from the user.	Yes
15	COULD	The application COULD allow exchanging files with contacts;	Being able to exchange files could be important for businesses who need to share documents with each other.	T45: Users will have the ability to import files and send them to a user. T46: Allow all Microsoft file types to be attached.	No
16	COULD	Files transfer COULD only start if the contact clicks in the link;	This feature could be useful as someone might share a file with you that you do not need, therefore if you don't click the link to download the file you will be saving space on your computer.	T47: Once the files is received by the user, allow them to preview the file. T48: Give the user an option to download the file onto their device.	No
17	COULD	The user COULD be able to send emoji;	Emojis are a very easy way to communicate with other people without sending long text. It means that communication between users can be done much quicker.	T49: Create a button next to the text box where emojis can be found. T50: Allow emojis to be embedded within messages. T51: Categorise the images by faces, food etc.	No
18	SHOULD	The user SHOULD be able to change his details including his picture;	Users sometimes like to personalise their accounts, and this is very helpful when it comes to group chats full of people. It's easy to see who people are before reading their username. And because of data protection and circumstances changing its important that users are able to change details like, name, address (if included), phone number, email address and password.	T7: On registration set a default profile picture for all users. T30: Allow the user to change their profile picture in the 'Settings' page. T31: Allow the user to submit JPG format of pictures.	Yes
19	COULD	The user pictures COULD be displayed in the channels;	This feature could allow for other users to easily identify who has sent a message by simply looking at a profile picture.	T32: Show images of each user next to their name in the accounts page. T33: Optionally, show the users image next to their messages in the chat room.	Yes

1.3 Additional Requirements

Table 4 - Additional Requirement Analysis

No	Priority	Requirement	Implication	Tasks	Completed (Yes/No)
1	MUST	Users MUST be able to send and receive messages;	This feature is incredibly important as without it users won't be able to communicate with each other. Communication between users is vital for a messaging application.	T8: Display all available contacts in the 'Accounts' page and available group chats in the 'Group Chats' page. T9: Allow users to select the recipient and then send messages. T10: Save sent messages to the database.	Yes
2	MUST	Users MUST be able to view their recent messages;	The main purpose of this application is to aid communication between people within a workplace. The users need access to their recent and past messages, so that they can reply to conversations and obtain new information or go back and review previous conversations that could help with the task they are doing.	T1: Create chatroom page. T11: Messages will be stored on a database, when the user wants to see their message history retrieve the messages from the database. T12: Display the users message history.	Yes
3	MUST	Storing account information;	Each account will be stored in a database which can then be updated using SQL commands. This makes managing the software easier as any changes needed to be made, can be applied to all accounts simultaneously. This could be managed using the Boost C++ library.	T5: Account information will be stored on an encrypted database using Amazon Web Services (AWS)	Yes
4	MUST	The software MUST be reliable;	There are often reports with different types of software of there being bugs which makes the program unusable. The software we will be creating should not have bugs that stop the program from being used how it should, or any bugs which stop the program from functioning properly should be fixed as soon as possible.	T57: Program the application by reusing functions to make the program run faster. T58: Implement error handling. T59: Implement additional threads for any intense tasks.	Yes for T57
5	MUST	The software MUST be accessible 24/7;	This feature is very important due to different users potentially being in different time zones. The software must be up and running during the day and at night so that any user can use it.	T14: Keep the application database and broker active 24/7 to allow the users to be able to always use the application.	Yes
6	MUST	Admins MUST have different permissions;	There will need to be users of the application who are able to manage accounts that will be on the application. This could be admin of a group that gives permissions to users within the group such as read and write.	T22: Allow the admin to manage mods by promoting or demoting them. T25: Allow the admin to manage users by resetting their passwords and remove them from the database. T26: Allow the admin to manage groups in which they can remove group members or delete the entire group.	Yes
7	MUST	The application MUST be optimized for multiple operation systems (OS);	Companies will have many different areas; some may need to have access to different operating systems to allow the use of OS specific functions. These areas still need to be able to efficiently communicate with the rest of the company, therefore will also need a version of the software that works on their OS.	T60: Develop a version of the application for Windows, Macintosh and Android operating systems.	No
8	MUST	Group owners MUST be able to add and remove members;	Group projects are often ongoing within a workplace and will be broken up into different teams that will work as one. Communication will become more efficient if the team could be added to the group for the time period they need the information for. Sometimes users could leave a group project as well, therefore they should be removed from the group to aid confidentiality within the workplace.	T22: Give group owners special permissions (add/ remove members)	Yes
9	MUST	The application MUST archive old conversations, groups, and files;	This feature will allow users to archive messages which they might find useful in the future. Old messages will automatically be archived after 6 months however, they are always accessible to the user.	T54: After a certain period of time update the message history log to "Archived" T55: Allow users to retrieve the archived messages	No

10	SHOULD	Alternative layout for smaller devices for example, mobile phones.	Not every device has huge screens that allow for the layout design that a computer does, for example, mobile phones have much smaller screens that do not accommodate for the same application layout, an alternative layout should be made so that the functionality of the application still works the same on other devices as it would on computers.	T61: Create an alternative GUI layout to be used on small devices which allow them to still have the same level of functionality. T62: Optimise the media to be more responsive on smaller screens.	No
11	SHOULD	Ability to create group chats and assign member permissions.	Allowing users to create their own group chats and add any member they wish using their email or user ID would allow them to message in groups instead of individually messaging each person. The owner of the group is then able to assign different permissions to each group member. These permissions include, owner permission (they are able to add and delete group members and prevent others from sending messages), read-only permissions (they are only able to read messages and open files) and write permissions where they can edit documents and send messages in the group.	T22: Give group admins and moderators group permissions.	Yes
12	SHOULD	Allow users to search for other users.	This feature will allow the user to be more time efficient as they can search for any member by email or user ID, instead of browsing through their recent contact.	T8: Implement a list that allows users to see all users. T63: Optionally, Implement a search bar with archive of existing users.	Yes for T8 No for T
13	SHOULD	Software is downloadable from a website link.	Websites are an easy way of downloading software using a link posted on one, the website would contain download links to the different versions of the software making it easy for users to identify which version of the software they need with ease. However, this method would only be needed for the first download, due to the updates being prompted through the application itself once they are released.	T64: Set up a website and store the download files for each version of the software. T65: Archive previous versions of the application.	No
14	SHOULD	Users SHOULD still be able to access their systems while the software is being installed.	While the software is downloading or updating, the user should see be able to operate their computer as usual. This feature is important as it allows the user to stay productive and use their computer throughout the download/ update process.	T66: Allow the installation screen to be minimized.	N/A
15	SHOULD	Regularly scheduled maintenance.	Users will be notified if there will be an update and they can choose the time at which it happens. These updates and regular maintenance will be used to assess the performance of the software and resolve and errors the software might occur.	T67: Create a notification to be displayed when a admins issues a maintenance request.	No
16	SHOULD	Small errors SHOULD be fixed within the same working day that it is found.	Small errors typically don't affect the functionality of the program; however, they can still affect user experience, and fixed soon after being found as they could agitate the users of the program and make them not want to use it.	T56: Create a report problems form containing the title, description and allow the user to attach media as proof of the bugs.	No
17	SHOULD	The application SHOULD be easy to maintain;	More than one person will have access to the applications code during and after development. If one person is to write code in a way that is not understandable, others will struggle if they need to make changes as it will create confusion about what it does within the program.	T68: During the naming of methods and variables, one naming convention should be used that is agreed on by all developers. T69: Code blocks should be commented throughout the source code, especially blocks of code that may seem confusing. T70: Throughout the source code document, the code should be tabbed to show belonging, whether this is to methods, case statements or loops etc.	Yes
18	SHOULD	Ability to attach media to messages.	This can be useful as allowing users to attach media to messages will allow for much more complex conversations to take place between users. The handling of these files could be managed by the C++ library known as Boost.	T38: Add a button to open the user's gallery or documents. T39: Allow the user to preview the media before sending them. T40: Store sent media to the database.	No

19	COULD	Receive message alerts.	Message alerts can be important as they can make sure users don't miss messages and therefore don't miss important conversations.	T36: Send popups indicating that the user has received a message. T37: Show received messages as notifications on the home page.	No
20	COULD	COULD have auto correct option which can be toggled on or off.	Auto correct can be useless for people who tend to make spelling mistakes or typos and them being corrected automatically will save them a lot of time.	T43: Auto correct can be turned on or off through the settings page.	No
21	COULD	Display whether messages have been read.	Allowing the user to see the status of their sent messages will allow them to find out if the other user(s) is aware of their message. This feature will be particularly useful in groups in which the owner can send a message and find out if everyone has seen it.	T42: Use colours to show status of message, green for seen messages and orange for delivered messages.	No
22	COULD	Allow video calls.	Video calls will give functionality to the software and be useful to many as demonstrations will become much easier especially when giving presentations which consist of displaying physical products etc.	T41: Video calls can be started through a button at the top right of a conversation. Allow the user to accept or deny the video call.	No
23	COULD	Allow users to view other profiles.	Users will be able to view other users' profiles including their availability status, their profile picture and basic information including contact. This feature will give some extra depth to the software as users could browse through profiles and find out the office hours of a particular member etc.	T44: Profiles will display the status, profile picture (PFP), name and last active. All PFPs will be links to their profiles.	No
24	COULD	Allow users to change their account preferences.	Users will have certain preferences that they like to turn on or off within applications, this could be brightness settings, text size, or accessibility settings such as colour-blind modes.	T34: Allow the user to change the application into 'dark mode' or 'light mode' that changes the style of the application. T35: Allow the user to change the applications language.	Yes for 34 No for 35

1.3 Risk Analysis

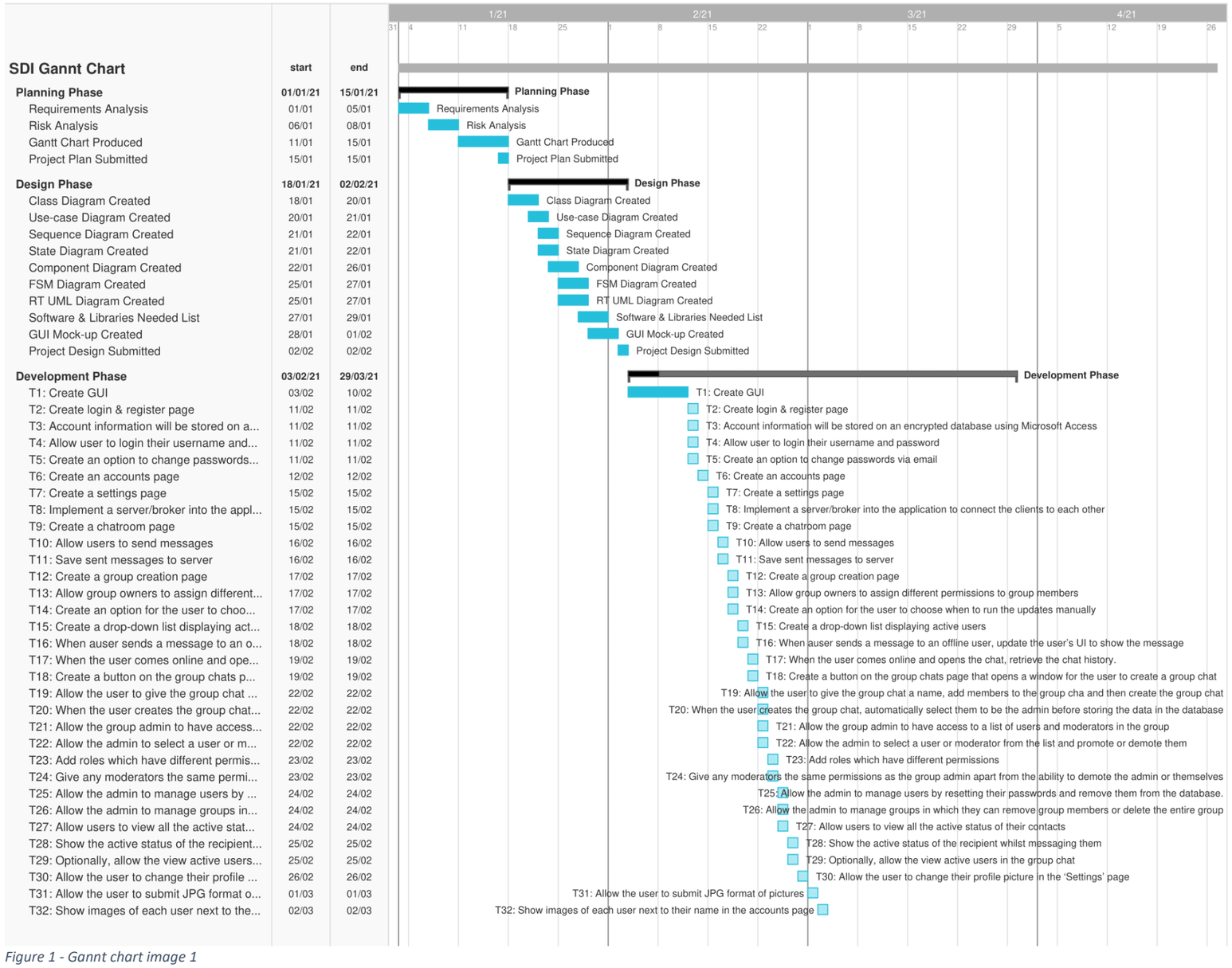
Table 5 - Risk Analysis

No	Risk	Probability	Impact	Description	Action Plan
1	Software loss.	2	5	Due to corruption or due to other external circumstances the software or progression of software may be lost.	We will be using GitHub to as an online backup when coding the application. We will also be taking offline backups at least once a week to make sure the project is not compromised.
2	Member of the team is lost.	1	5	A member of the team has left the project team for different reasons. Etc personal issues, conflicts with other team members, offered a position elsewhere.	The work will be divided between the remaining members, and more flexibility will be added to allow more time for completing the tasks as each member will have a greater workload. These out-of-control circumstances will be considered on the Gantt chart with flexibility to make tasks fit into the schedule.
3	Member/s of the team fall ill.	1	5	Team members could come down with a short or long term illness which may mean workloads cannot be completed by those members, or they cannot complete as much work.	The work will be completed by other members of the group and when the team member returns, they will have additional tasks to complete to make up for the loss of work.
4	Members conflicts.	3	4	Members disagree with the overall direction of the project, or decisions made within the project.	We will be scheduling weekly meeting on every Wednesday at 3pm. These meetings will start with a 5m reflection of the progress of the project and the direction in which its going. Any changes wanting to be made will be discussed in the presence of all members and a vote will be made if any member wishes to alter the project direction.
5	Lack of engagement from team members.	3	4	Team members may not be engaging with project the way that team needs them to, therefore not doing as much or any work for the team, putting stress and more work on other members of the team.	Progress of the project will be reviewed weekly as well as, individual contributions of each member. There will be a three strike policy in which each member will be given three chances to complete outstanding work they have yet to do. Once a member receives three strikes, the project manager can let someone superior about the members performance.
6	Lack of motivation.	3	4	Less motivation leads to less work being completed, or leads to members getting easily distracted, again leading to less work being completed.	Weekly meetings will take place where team members will be able to provide their thoughts give them their say on what is going on to keep them involved and interested in the project. This should provide the motivation members need as it will feel more like their own project.
7	Work balance isn't adjusted correctly.	2	4	This could lead to upsets and disagreements and unreasonable expectations about what work can be completed in a select time frame.	The project will be reviewed weekly to keep track of how much work individuals have put in. This will allow us to make sure that each person has a similar amount of work per week and if someone doesn't then that can be adjusted in the weekly review meetings.
8	Computational resources not being fully understood before starting the project.	2	5	Might result in the correct equipment and software being unavailable for some time. Holds up the project, wasting time and money.	We will hold meetings between people to make sure that each person understands clearly what is required. If someone doesn't understand then in these meetings it will be explained to them so that they can clearly understand the project.
9	Lack of communications.	3	4	Less direction, people will become frustrated and not want to continue with the project.	There will be weekly meetings where we find out where everyone is up to and provide them with the help they might require. We will also be splitting up the next section of work to make sure everyone knows what they are doing. We will also have a leader Who people can speak to if they need help or are unsure about something.
10	Tight schedules	4	5	This could mean that not enough time is given to certain tasks. This could lead to big delays and confusion or poor quality work as everything maybe rushed	Meetings will be held where we track individuals progress and make sure they are given the time and help they need to complete their task. If we run out of time of a certain thing then others can provide help to make sure everything gets done in the way it should.

11	Sudden growth in requirements	1	3	Most of the time, all the requirements will be found during the requirements capture, however, this is not always the case. It could be that there were a few requirements that were misunderstood, or the client could possibly want to add a few features to the end program.	During the requirements capture, be as thorough as possible with the requirements so that both the developers and the client are sure all the requirements have been found. The software will be developed to allow it to be maintainable allowing it to be easier to implement any requirements that might have been missed.
12	Unclear specification	2	4	If the specification is unclear, there is not going to be efficient communication between the developers and the client. Sometimes the developers may believe that they understand the specification, but it could be that the client had a different image of what the final product was going to be.	Make sure any confusion between the client and developers is addressed at the start, any question or queries should either be covered in the requirements or through having regular meetings with the clients to ensure that the correct product is being made for them.
13	Over complicating simple tasks	2	3	Some developers may see a specific task and have a clear step by step of how to get there in the simplest way. However, others may not have the same level of knowledge and it could cause the task to be over complicated which would decrease program efficiency.	Make sure major programming tasks are talked through with the team in order to make sure this doesn't happen.
14	Changes of government policy	1	4	Occasionally, there could be a change in government policy, such as policies built around handling data, these types of policies are likely to be made stricter in time.	Allow for contingency time in the project development to make sure if something like this were to happen then the issues can be addressed and changed to fit the new regulations
15	Not sticking to the set schedule	3	4	At the start of the development, a schedule is set to work against to know whether the team is working on track to get the project finished by the deadline, sometimes problems occur that could cause the schedule to be overrun.	Make sure the project plan contains plenty of milestones and slots for contingency time. The milestones will allow the team to see what has been done and what has yet to be done and what has not been done. The extra contingency time will allow for the project to continue on time if problems occur or allow the project to move faster than anticipated if no issues are found.
16	Poor budget estimation	2	5	If the budget were to be poorly estimated, it may lead to equipment, software and employees not properly being accounted for, this would lead to money issues later in the project which could either slow the project down until these issues were properly solved or could end up with project becoming bankrupt and therefore cancelled.	Make a proper plan of all the equipment and software that will be needed for each step of the project. This is to ensure that the correct resources can be accounted for and there for the budget can be correctly/ accurately estimated.
17	Poor code quality	2	4	Poor code quality can lead to further developments and updates on the project being slowed or near impossible to complete without completely redoing the code. This would waste a lot of time and money. Not to mention poor code would most likely be poorly optimised meaning slower loading and response times. This wouldn't sit well with users of the application.	Take regularly scheduled meetings to go over the code to see if there is a more efficient way of doing the code. This is to ensure that future code can be added easily and improve and compile and load times.

18	Inadequate research into the target audience.	1	3	If the target audience is not properly researched, then features may be added into the program that are not necessary or may harm the program in terms of user reviews. Also, this may be the other way around where a desired feature is missed out which is very bad especially considering the amount of messaging applications around (competition) leading to users going elsewhere if they are not satisfied.	Use a range of research gathering techniques (both targeted and random sampling) on the desired target audience to make sure desired features are put in.
19	Not enough support and maintained after the software has been published	2	4	This would mean error/ bugs would go unchecked for a long period of time or indefinitely which would aggravate the users. If the creators of the product can't support their creation, why should users use it?	Include/ create an external plan for the after-release support for the application. This would include when updates are going to happen and when new features can be expected.
20	Lack of organisation	2	5	Lack of organisation ties into a lot of things, including mismanaging time, resources and money. Each of these can be disastrous for the project.	Take more care when making the project plan to ensure that all areas of the project are accounted for. This will help keep the project organised.
Risk Probability: 5 – Most Likely, 1 – Unlikely Impact: 5 – Severe, 1 - Minimal					

1.4 Gantt Chart



SDI Gannt Chart	start	end
Planning Phase	**01/01/21**	**15/01/21**
Requirements Analysis	01/01	05/01
Risk Analysis	06/01	08/01
Gantt Chart Produced	11/01	15/01
Project Plan Submitted	15/01	15/01
Design Phase	**18/01/21**	**02/02/21**
Class Diagram Created	18/01	20/01
Use-case Diagram Created	20/01	21/01
Sequence Diagram Created	21/01	22/01
State Diagram Created	21/01	22/01
Component Diagram Created	22/01	26/01
FSM Diagram Created	25/01	27/01
RT UML Diagram Created	25/01	27/01
Software & Libraries Needed List	27/01	29/01
GUI Mock-up Created	28/01	01/02
Project Design Submitted	02/02	02/02
Development Phase	**03/02/21**	**29/03/21**
T1: Create GUI	03/02	10/02
T2: Create login & register page	11/02	11/02
T3: Account information will be stored on a...	11/02	11/02
T4: Allow user to login their username and...	11/02	11/02
T5: Create an option to change passwords...	11/02	11/02
T6: Create an accounts page	12/02	12/02
T7: Create a settings page	15/02	15/02
T8: Implement a server/broker into the appl...	15/02	15/02
T9: Create a chatroom page	15/02	15/02
T10: Allow users to send messages	16/02	16/02
T11: Save sent messages to server	16/02	16/02
T12: Create a group creation page	17/02	17/02
T13: Allow group owners to assign different...	17/02	17/02
T14: Create an option for the user to choo...	17/02	17/02
T15: Create a drop-down list displaying act...	18/02	18/02
T16: When auser sends a message to an o...	18/02	18/02
T17: When the user comes online and ope...	19/02	19/02
T18: Create a button on the group chats p...	19/02	19/02
T19: Allow the user to give the group chat ...	22/02	22/02
T20: When the user creates the group chat...	22/02	22/02
T21: Allow the group admin to have access...	22/02	22/02
T22: Allow the admin to select a user or m...	22/02	22/02
T23: Add roles which have different permis...	23/02	23/02
T24: Give any moderators the same permi...	23/02	23/02
T25: Allow the admin to manage users by ...	24/02	24/02
T26: Allow the admin to manage groups in...	24/02	24/02
T27: Allow users to view all the active stat...	24/02	24/02
T28: Show the active status of the recipient...	25/02	25/02
T29: Optionally, allow the view active users...	25/02	25/02
T30: Allow the user to change their profile ...	26/02	26/02
T31: Allow the user to submit JPG format o...	01/03	01/03
T32: Show images of each user next to the...	02/03	02/03

Figure 1 - Gannt chart image 1

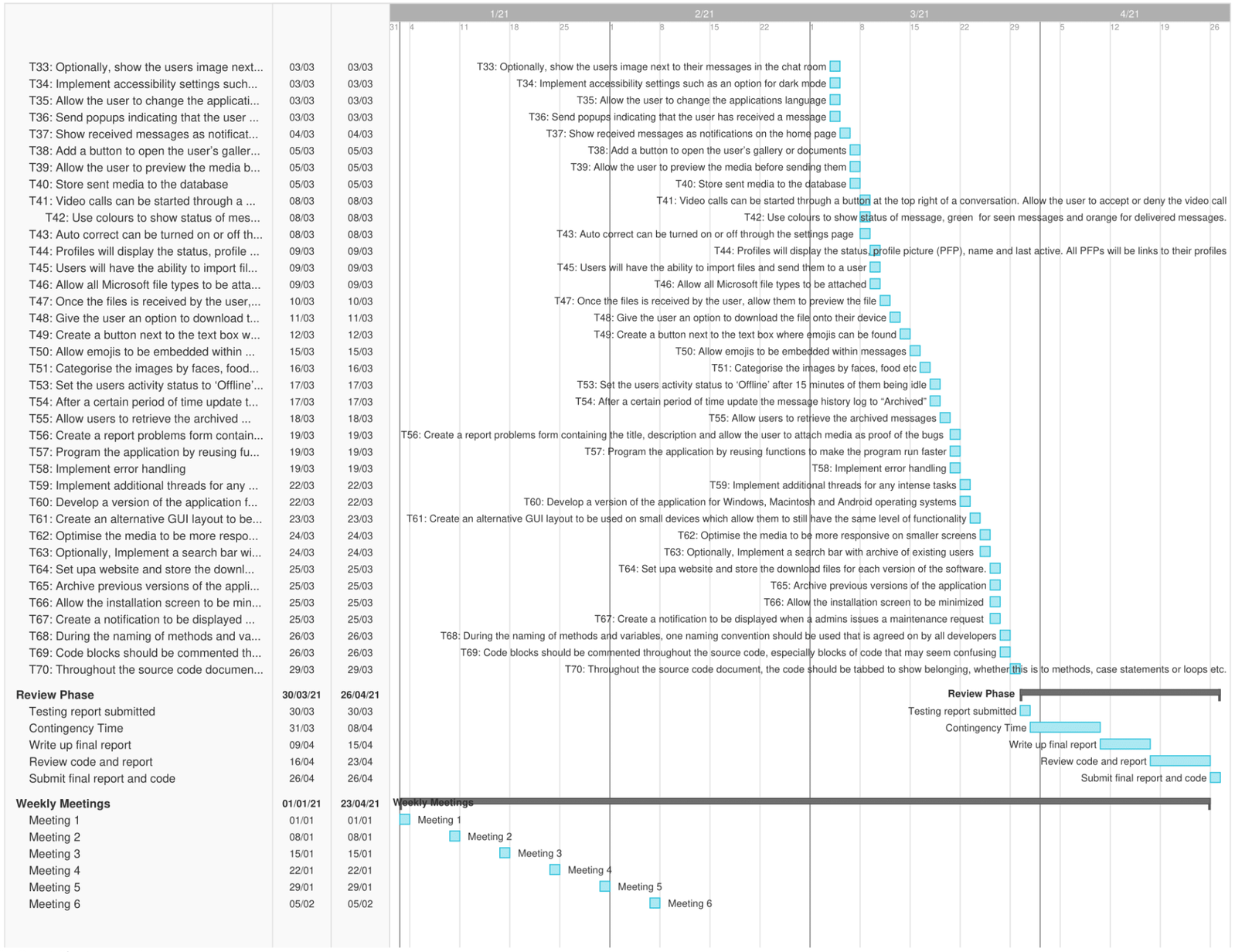


Figure 2 - Gannt Chart image 2

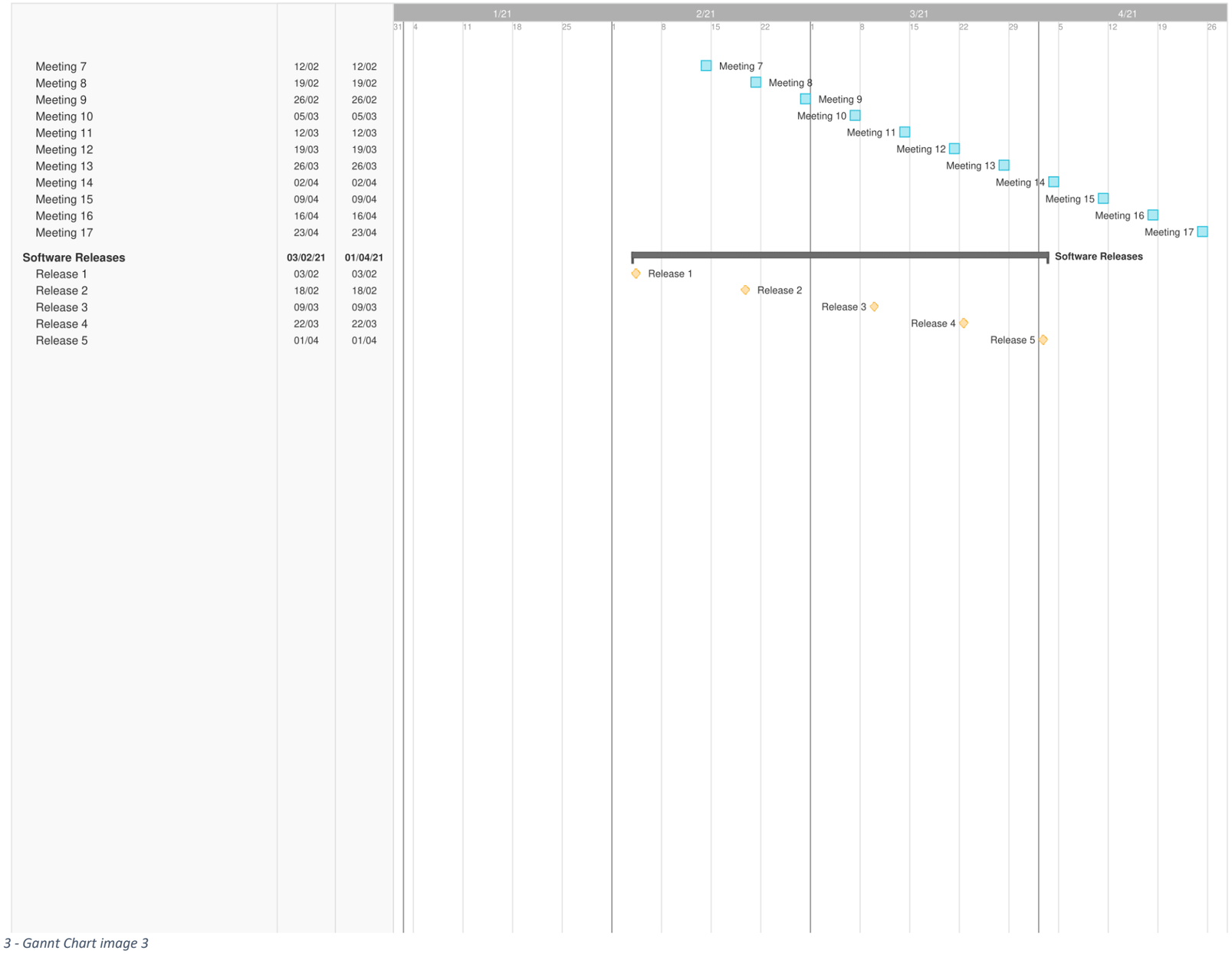


Figure 3 - Gannt Chart image 3

Assumptions Made

Many assumptions were made in the initial in the planning phase of the project. This included the dependence on using a text file based system to store all user data. It was underestimated during the planning phase the complexity and scale of the project and wasn't fully realised how much learning would need to be done in order to develop the project. One of the biggest assumptions that we've made are the projects scheduled timescales. As a group of 3, we struggled to meet some deadlines and keep up to date with the project. As a result, few features weren't implemented or were underdeveloped.

Adopted Coding Standards

C++ Version

Throughout the development of our project, we have used the QT application running on version 5.12.8 which uses C++ 11.

Header Files

A header file should be used for every class that is made. This will use the standard .h files, apart from the class used to contain global variables which will use a .hpp file.

Methods are defined in these header files instead of inside the .cpp files.

Scoping

Variables are scoped as locally as possible. Therefore, if a variable is only needed for an if statement then it should be created inside the statement. Global variables are only used when needed, such as for storing data that is to be used throughout the whole application.

Classes

Classes are used for each of the UI pages that are housed in our application. This is due to being standard practise for QT to use classes for this role.

Functions

Functions are kept as small as possible. If code can be reused, then they should have their own function. If having small functions are not possible, then they will be thoroughly commented to increase code readability.

Other C++ Features

Preincrement and predecrement should be used due to generally being more readable as this form is used in areas such as predefined for loops. Use of these may be unreadable depending on the context, therefore should be followed by a code comment.

Integer types should be used due to being explicit, meaning you are specifying the right size for the variable.

Type deduction should not be used, instead variables should be given an explicit type.

Variable initialisation should be used when the variable is declared, unless the variable has the possibility to be given a value from the user.

Naming Conventions

Using names for variables which relate to what that variable is being used for is very useful for improving readability.

Classes will use the upper camel case naming convention, for example "MessagingPage".

Methods will use underscores to separate words for example "set_active_status".

Variable names should use underscores to separate words eg password_entered. We have decided not to use a different naming convention to the methods due to method calling requiring a parenthesis which will differentiate the two.

Code Comments

Comments should be embedded within the code to make sure that each part of the code can be easily understood by reading the comments. Compatible comments should also be used on methods and important variables when using Doxygen.

Formatting

There should be spaces in-between sections of code to allow for better readability.

Spaces after commas should also be used.

Functions should be clearly separated through spaces.

Indentations should be used when appropriate (eg if statements, for loops etc).

1.0 Use Case Diagrams

Login

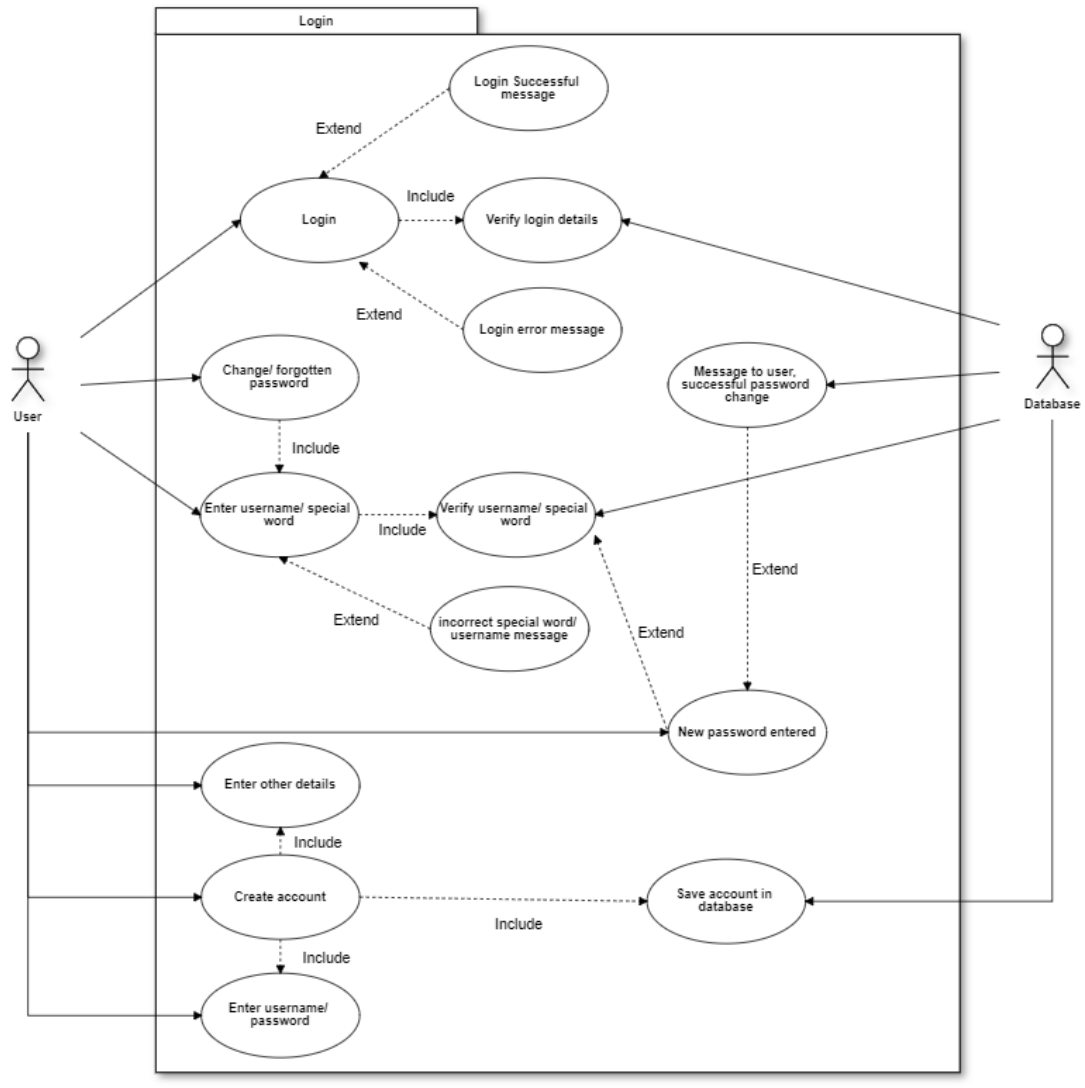


Figure 4 Login usecase diagram using Cacoo

The user has multiple actions. If the user has not created an account with the application, then they will be required to. This action will require the user to enter a password and username. As well as a number of other details. Once the details have been entered and the account has been set up all the details will be saved into a database. Once the account has been saved into the database the user will be able to login. The login will be verified using the saved database details. The user will either be met with an error message telling them either the username or password is incorrect. Or they will get a message telling them they have successfully logged in.

If the user forgets their password, then there will be a facility for this. They will be able enter their username and special word. This would be verified with the database details. If the details are wrong, then a message will be displayed telling them the details are invalid. The user can enter their new password once details have been accepted. The user will then receive a message to say the password has been updated.

Logoff

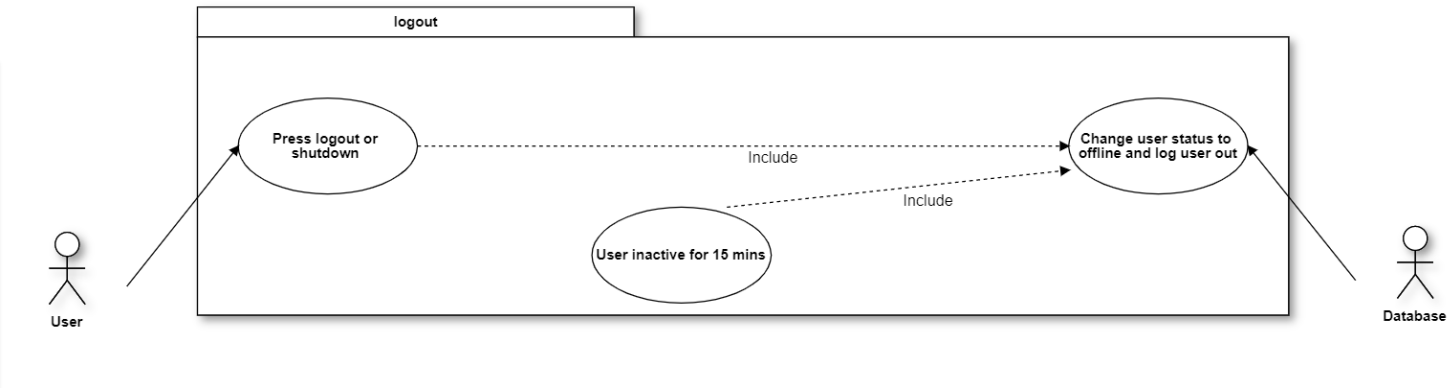


Figure 5 - Logoff usecase diagram using Cacao

The user has the ability to press the logout or shutdown button on any page of the application. Once this has been done the user status is changed to offline in the database.

Also, the user status can be changed based on inactivity. For instance, for 15 minutes the user status is changed to offline within the database and the login page is displayed.

Send message

Figure 6 Send Message usecase diagram using Cacoo

During the send message phase, the user will be able to do a number of things. For instance, the user must pick from a list of existing users in the application. This list will be provided by the external database. The user and broker will need to be connected in order to send or receive any messages. Once this is done the user will need to subscribe to the conversation as this uses MQTT. The user can then send and receive messages. When a user sends a message, the message is saved to the message history in the database. The sent message is sent to the MQTT broker. After this it is passed onto the user.

1.1 Activity Diagrams

login

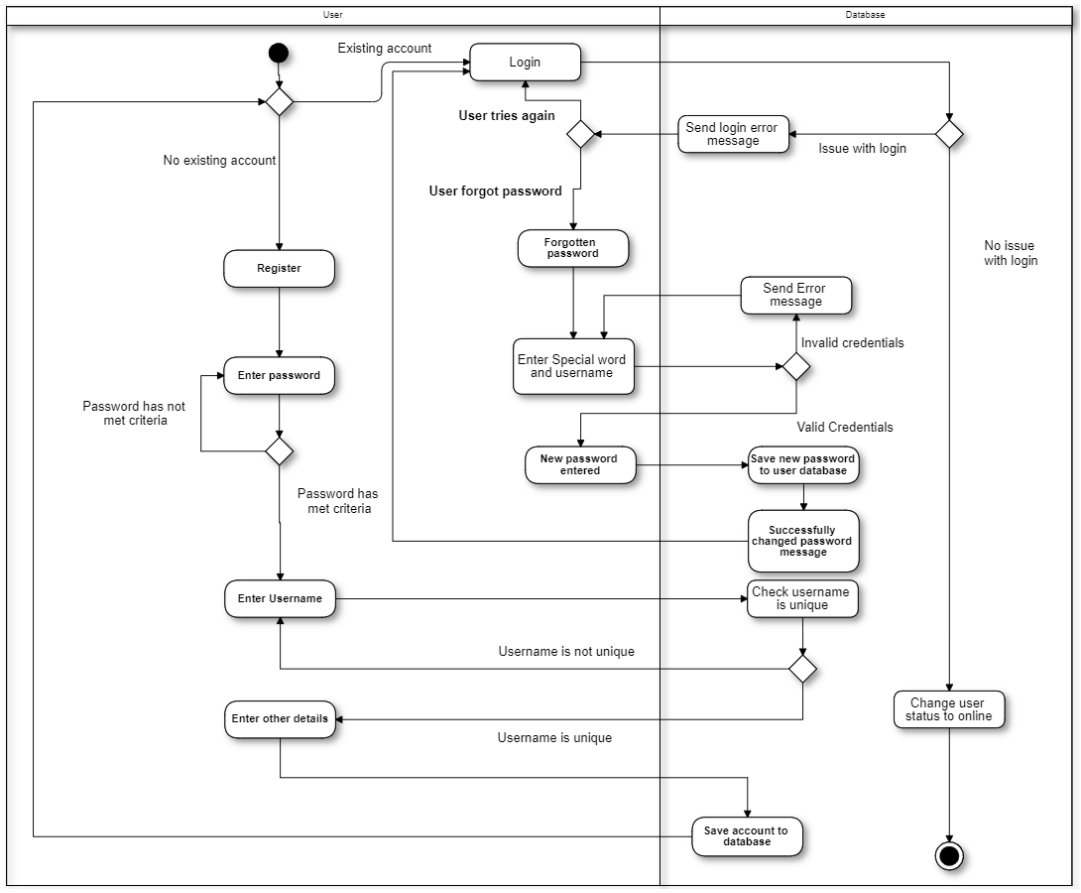


Figure 7 Login Activity diagram using Cacoo

At the start the user can either have an account or they won't have an account. If the user doesn't have an account then they will be asked to create one. So the user will register. The user will enter many details including a password which will be check to see if it meets the criteria of the application. For example, the password must have more than 8 characters, must have numbers and special characters etc. if the password does not meet the criteria then the user will be required to keep putting passwords in until all of the criteria has been met. The username that is entered will be checked against the database to check to see if its unique or not. Like with the password the username will need to keep being entered in till it is unique. Once this is done all the other details can be entered. The details are saved into the database and then the user would be expected to login. The diagram there is taken back to the start. If the user has an existing account, then they will be prompted to login. If there is an issue with the login an error message will be shown to the user. The user can keep trying to login and get the right login details. Or they can opt to change their password/ forgot their password. They will be required to enter their special word and username which were set up when the account was created. The database then checks these details to see if they are valid. Once the details are valid they can enter in there new password. The new password is then saved to the database. Once the password has been saved a message is displayed to the user. The user will be required to login. If all goes well and there is no issue with the login the user status is updated to online and the user will be logged in.

Logoff

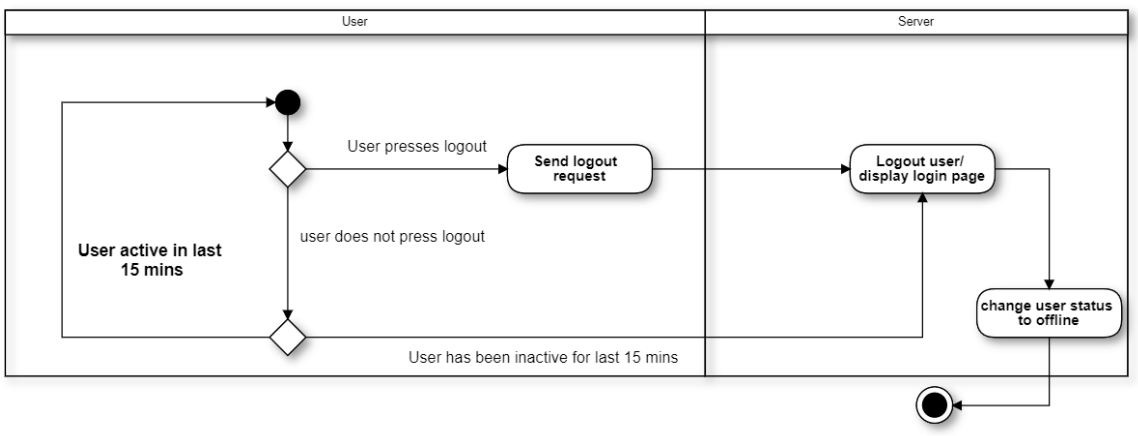


Figure 8 Logoff Activity diagram using Cacoo

The first decision node checks whether the user has opted to log off or not. If the user has opted to log out for themselves then the log out request is sent to the server. The database then takes this request and logs the user out. The user status is changed to offline. If the user does not opt to log out for themselves then the next decision node checks for 15 mins of inactivity. If the user does not have 15 mins of inactivity, then the process is started again (activity diagram starts from the top again). If the user has been inactive for 15 mins, then the system will log the user off. The login page will be displayed, and the user status will be changed to offline.

Send message

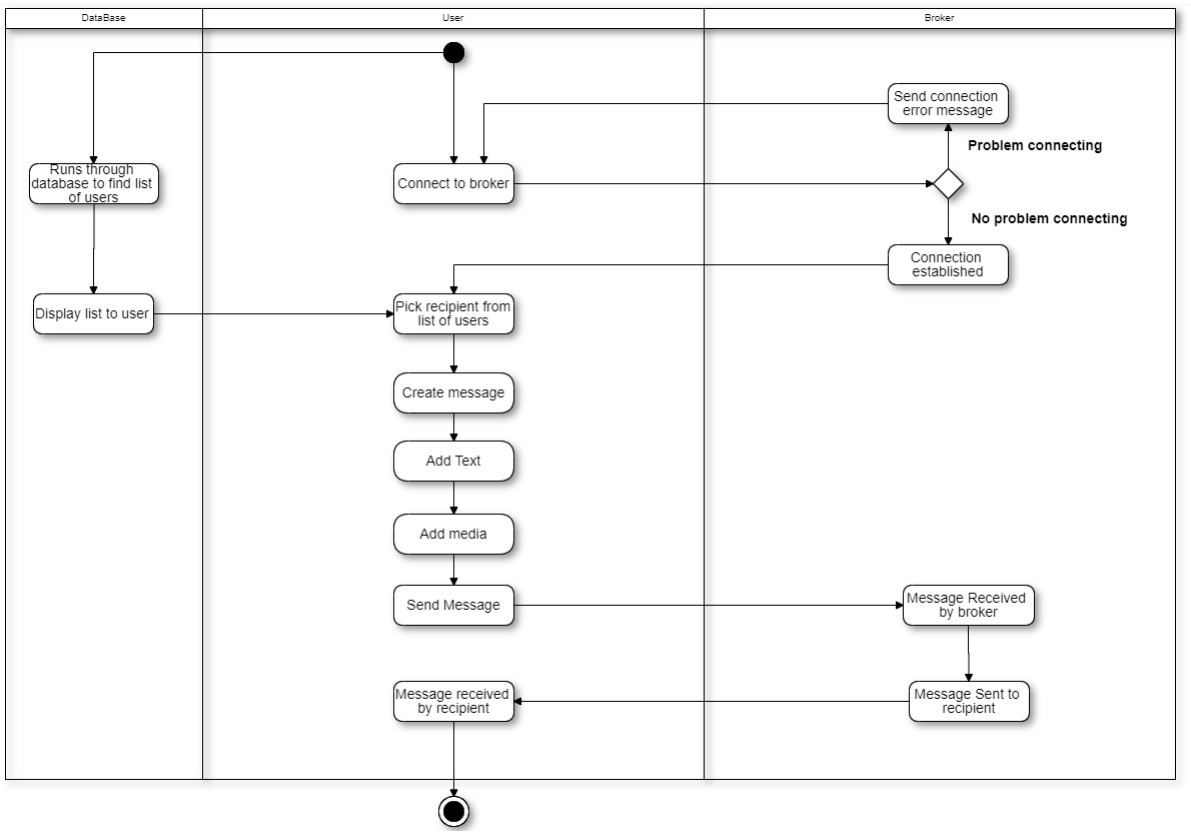


Figure 9 Send Message Activity diagram using Cacoo

The start of the diagram shows the user connecting to the broker. On the broker side the user can either be connected or no connected. If there is a problem connecting then an error message will be displayed and the process will repeat till the broker has successfully been connected to. Ince the connection has been established the user will get to pick a recipient from a list of users. Going back to the start of the diagram, the database is also tasked with creating a list of users and displaying those users to the user. Once the user has pick the recipient, then they can create a message. Add text to the message, add media and then send the message. The message is received by the broker. The message is then sent onto the recipient. The message is finally received by the recipient.

1.2 Class Diagram

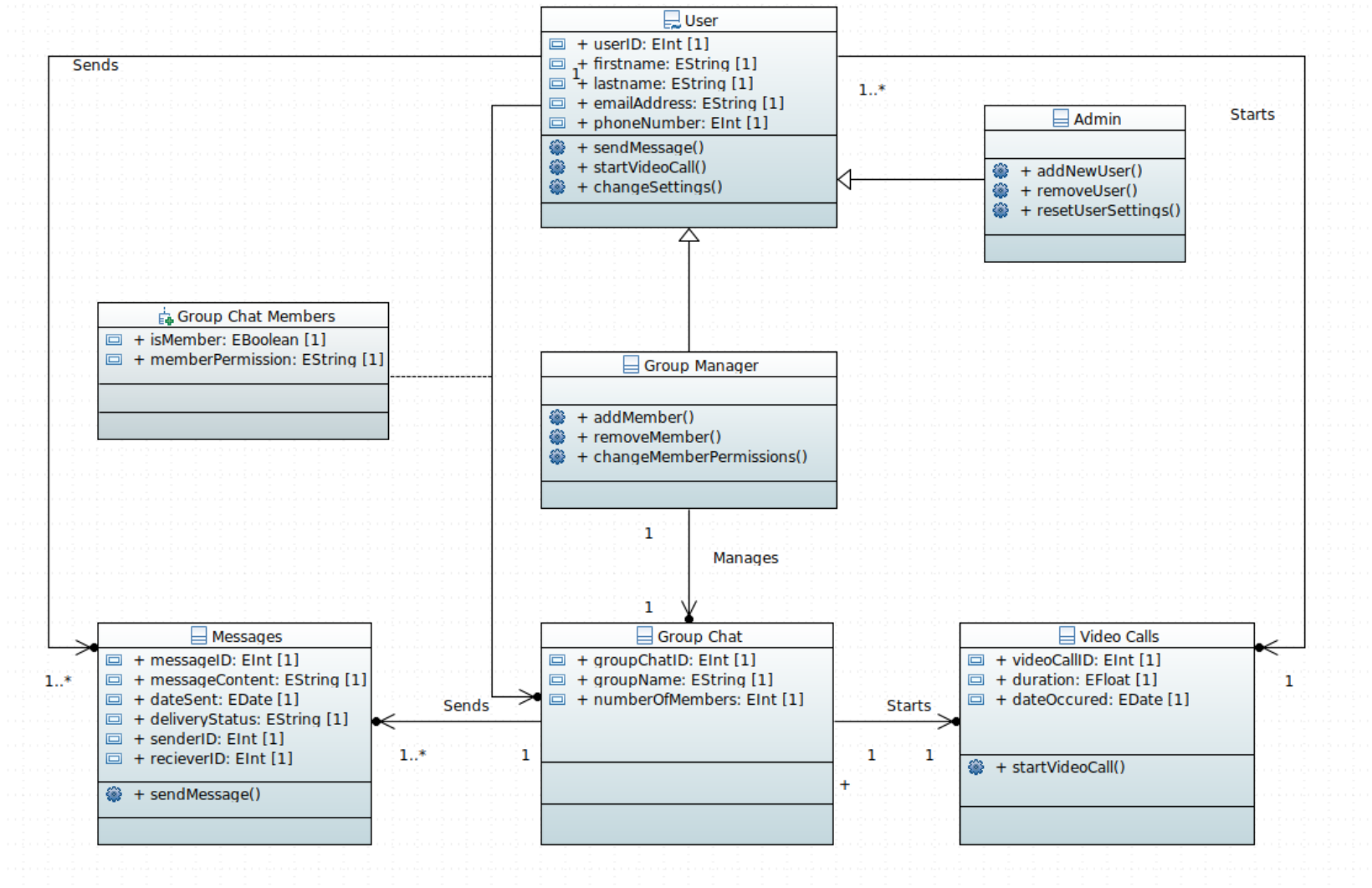


Figure 10 Class Diagram using Papyrus

The User class consists of 5 attributes that contain variable names such as the userID, first name, last name etc that represents the user's details. The User class also contains 3 methods which allow the user to send messages, start video calls and change settings. Two associations are used to show the relationship of the User class with other classes. The first association is a 1 to many relationship between the User and Messages as the each message can only be sent by one sender however, 1 sender can send multiple messages to different users (receivers). There is also another association between the User class and the Video Calls class which shows a many to 1 relationship. This is because the user can take part in 1 video call at a time however, 1 video call is between many users.

The next class is the Admin class that contains just methods that allow them to add new users, remove users and reset user settings. The Admin class does not contain any attributes as it is a generalisation of the User Class, meaning that the admin inherits all the attributes from the Users class. Similarly, the Group Manager class does not contain any attributes as it's a generalisation of the User class as each group manager is also a user. However, the Group Manager class contains additional methods to add members to the group, remove members and change their member permissions e.g. write-only or read-only.

The Group Chat class contains three attributes (groupChatID, groupName and numberOfMembers) which represent the group chat details. Two associations are used to show the relationship between the Group Chat and the other 2 Classes. The first association represents a 1 to many relationship between the Group Chat and Messages. Each message can only be sent within 1 group chat however, 1 group chat can contain many messages. A 1 to 1 association is used to represent the relationship between the Group Chat and Video Calls. 1 group chat can have 1 video call going on at once and 1 video call can only be between members in 1 group chat.

The Messages class contains 6 attributes that represent each messages detail, these include message ID, message content, date sent, delivery status, sender ID and received ID. The senderID is the usersID, and the receiverID is the groupChatID. It also contains a method to send messages. The Video Calls class similarly contains attributes such as videoCallID, duration and dateOccured. A method is used to start the video call.

Lastly, an association class is used to display the group chat members. There is a 1 to many association relationship between the User and Group Chat Members classes. This is because 1 user can be in many Group Chats. Another association is used to show that 1 Group Chat can have many Group Chat Members.

1.3 Sequence Diagrams

Login

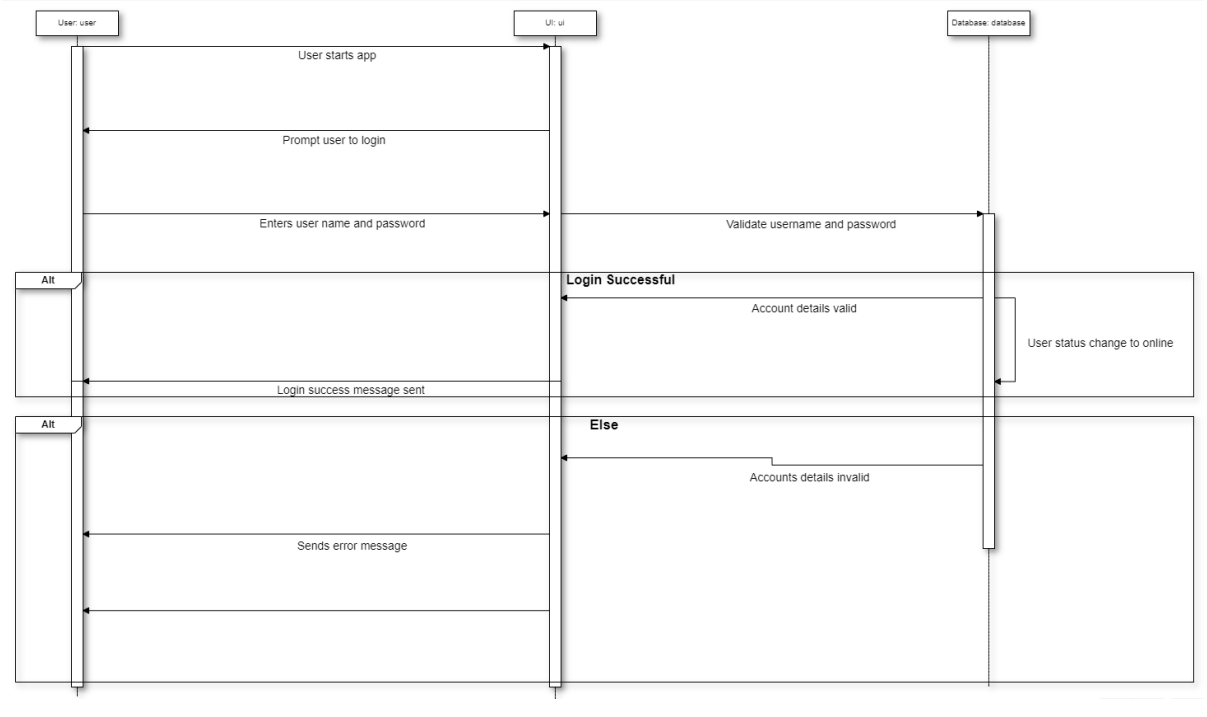


Figure 11 Login Sequence Diagram using Cacoo

The user initially starts the app. They are then prompted by the UI to enter their credentials. The user will then enter there username. The UI then gets the database to validate the credentials entered. This can cause 2 events. The first one is that the login is successful and so the database will return saying credentials are valid. The database will then change the user status to online. Then the UI displays a successful login message to the user.

The other option is that the details are invalid. In this instance the database will return saying the account details are invalid. Then the UI will present an error message to the user.

Register

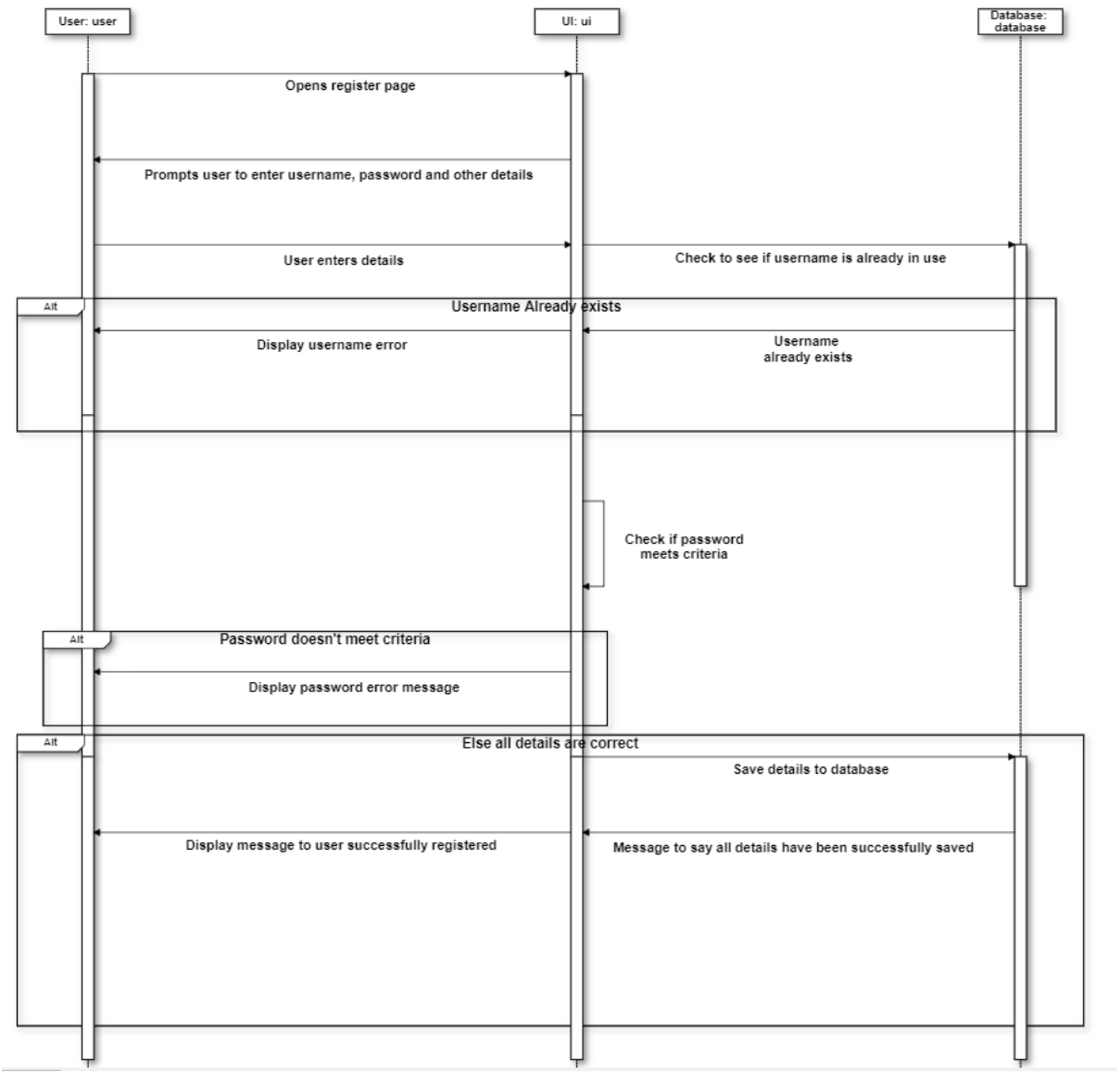


Figure 12 Register Sequence Diagram using Cacoo

The user will open the register page. The UI will then prompt the user to enter their details. User will enter their details. The UI then prompts the database to check that the username isn't already in use. If the username already exists, then the database will inform the application and the UI will display an error message to the user. After this the UI/ application will check the password meets the criteria. If it does not, then an error message will be displayed to the user. If everything is ok and no error are generated, then the details are saved to the database. The database sends a message to say details have been saved and the Ui displays it to the user.

Logout

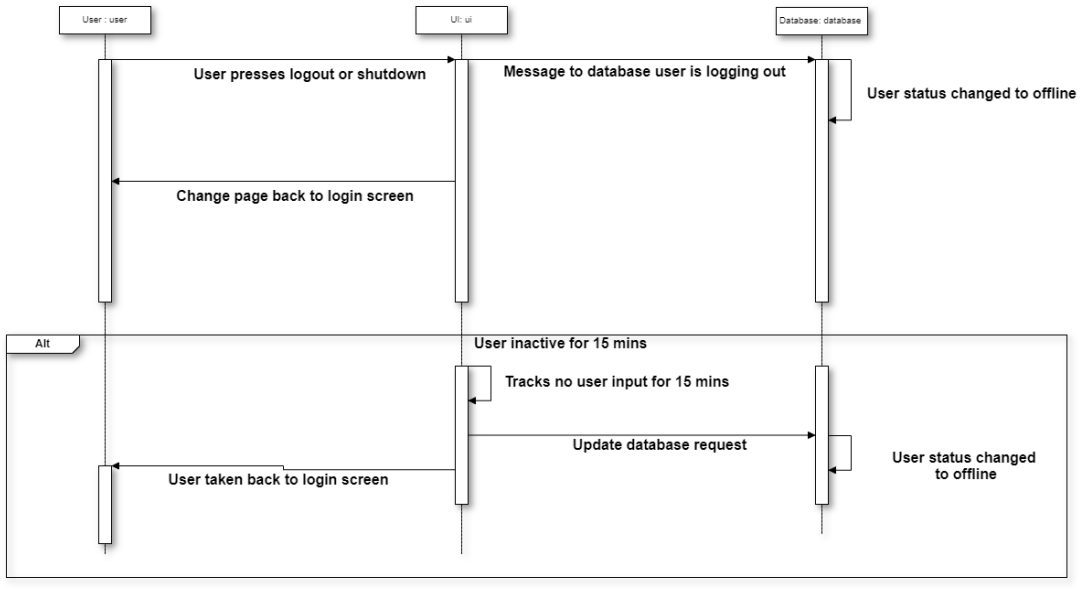


Figure 13 Logoff Sequence Diagram using Cacoo

The user prompted log off is done when the user presses either the shutdown or logoff button. The database will be informed of the user's logout and the user status will be changed to offline. The login screen will be displayed if the user has logged off. Then if the user activity is not picked up for 15 mins the application will request that the user status is changed to offline, and the user is logged out.

Send Message

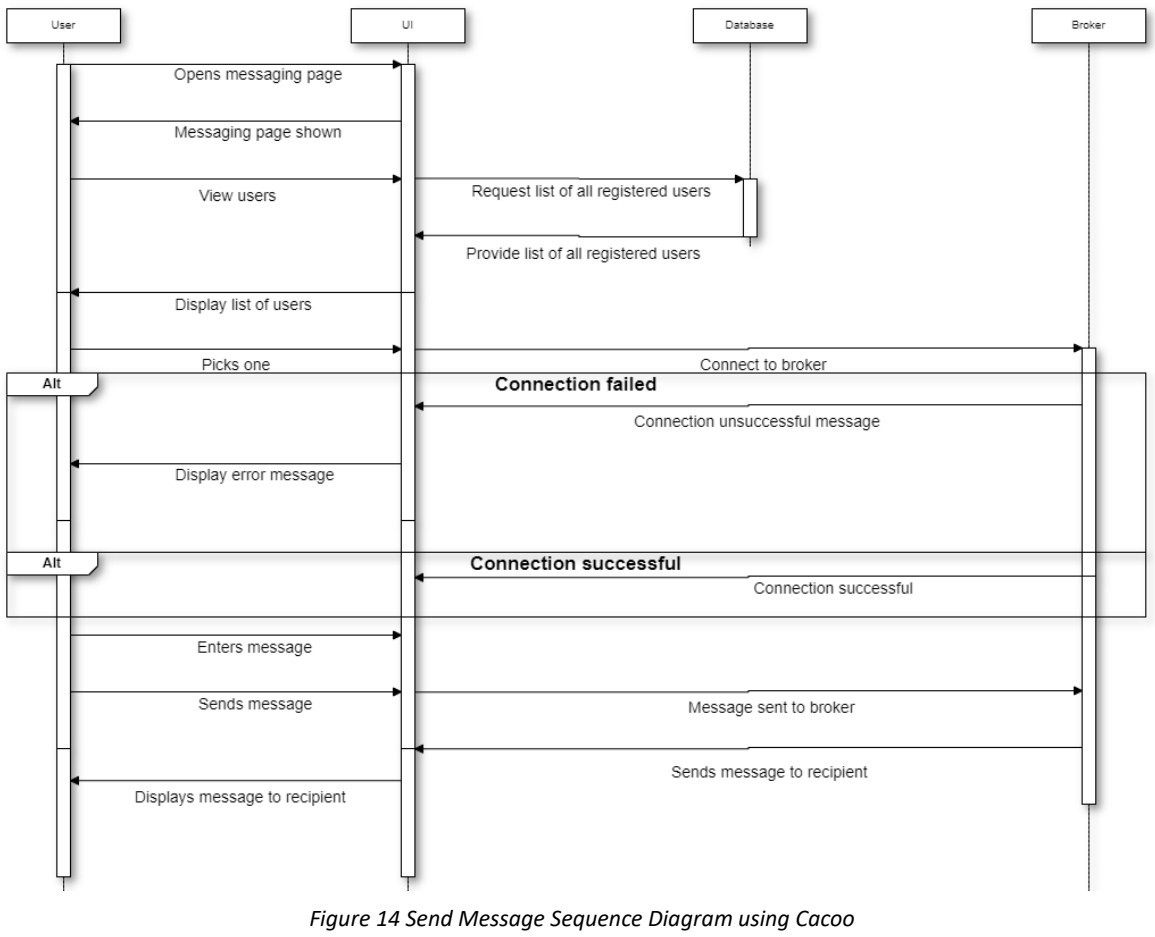


Figure 14 Send Message Sequence Diagram using Cacoo

The user opens the messaging page. The Ui will then open the messaging page. The user will request to view other users on the application. The UI will take this request and prompt the database to send a list of all current users. The database will then respond and provide a list of users. The UI displays this list to the user. The user will pick one to message. The user will then connect to the broker. 2 events can happen as a result of this. The connection can be unsuccessful and so the broker will send a message to the application and the message will be displayed to the user. If the broker is connected, then the application will be informed but eh user expects the connection to be established so they will not be informed. The user can enter their message and then send then message. The UI/application then sends the message to the broker. The broker will then send the message to the recipient. The UI displays this message to the recipient.

1.4 Deployment Diagram

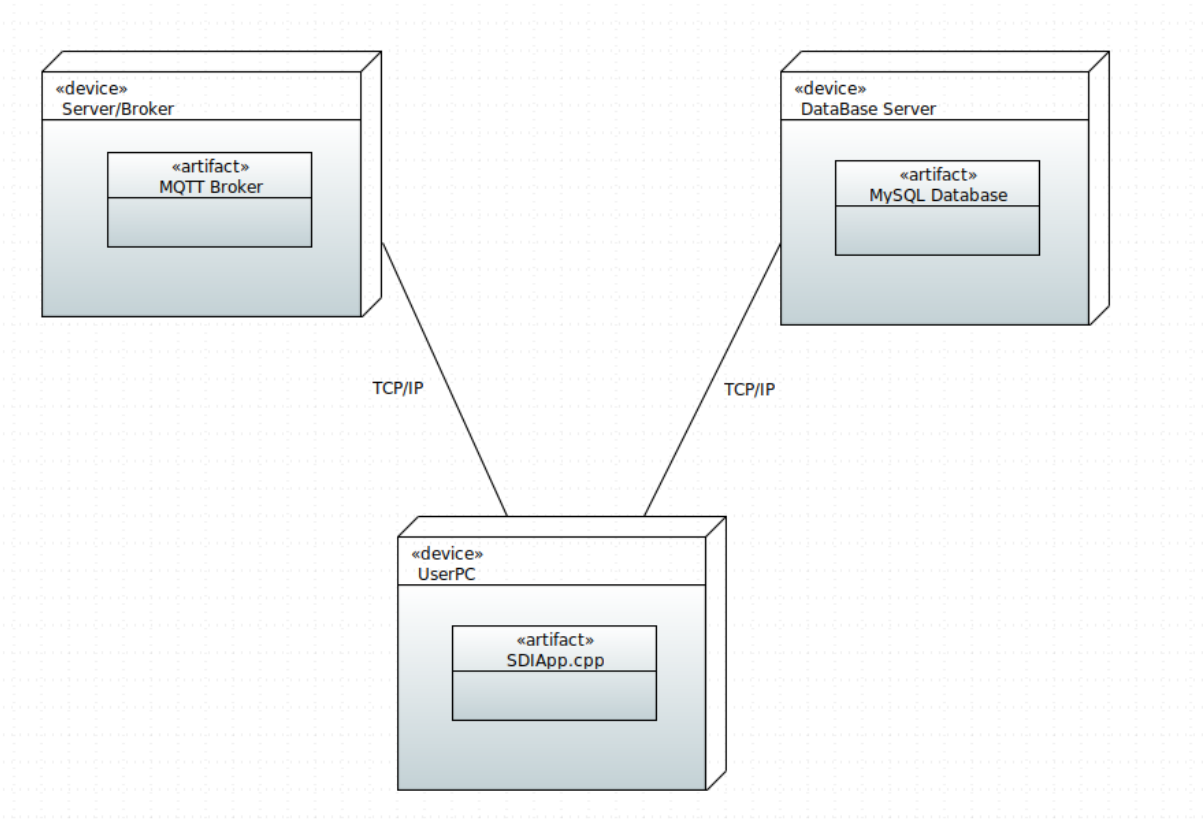


Figure 15 Deployment Diagram using Papyrus

The way the application will work is show here. The User PC will house the application, so the application will need to be downloaded on each user PC and cannot be accessed remotely. The system will use an external MySQL database. This will be used by the application for nearly every task within. For instance, login, logout, send messages, update settings etc. This will be using a TCP/IP connection and will connect directly to the user. This will not connect directly to the broker. The last part of the application is the external MQTT broker. This will handle all the message passing by the users. This will connect once again directly to the user using a TCP/IP connection. The broker and database do not connect to each other at all and any information sharing that maybe required will be done through the application on the user pc.

1.5 Communication Diagram

Login

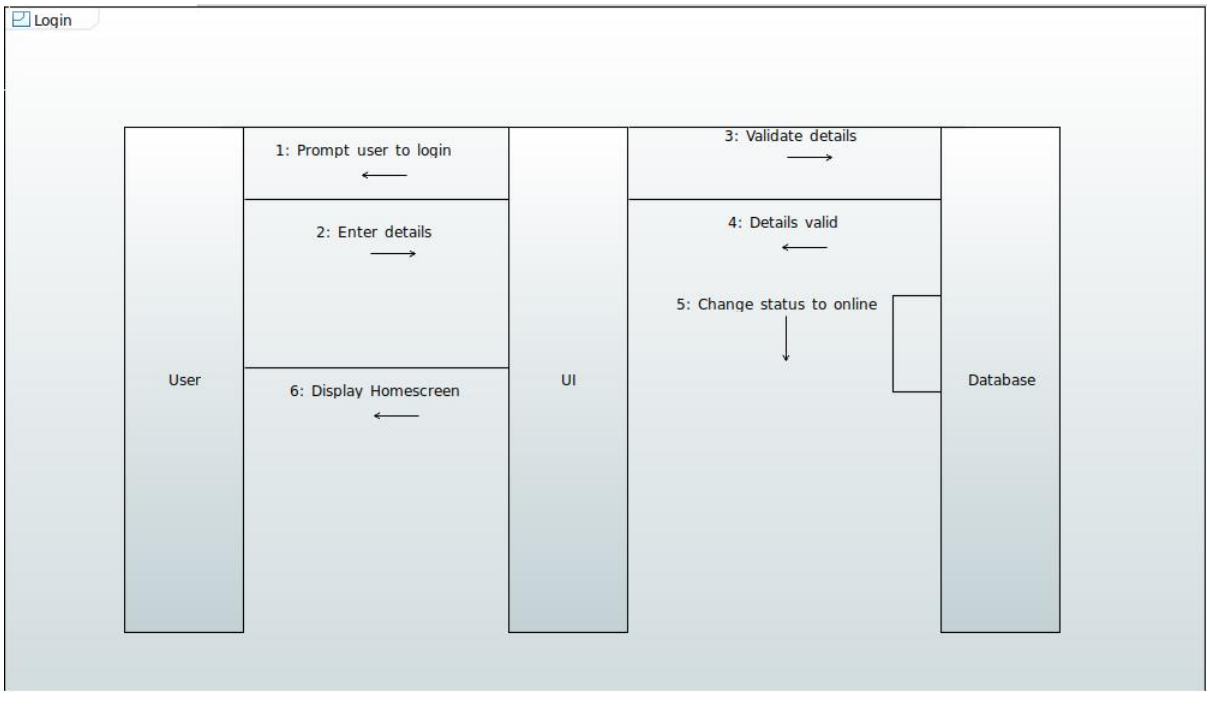


Figure 16 Login Communication Diagram using Papyrus

The UI will prompt the user to login. The user will enter their details into the UI. The database will validate these details and if it is found that the details are valid then the user status will be changed to online, on the database. Once this has been done the home screen will displayed.

Register

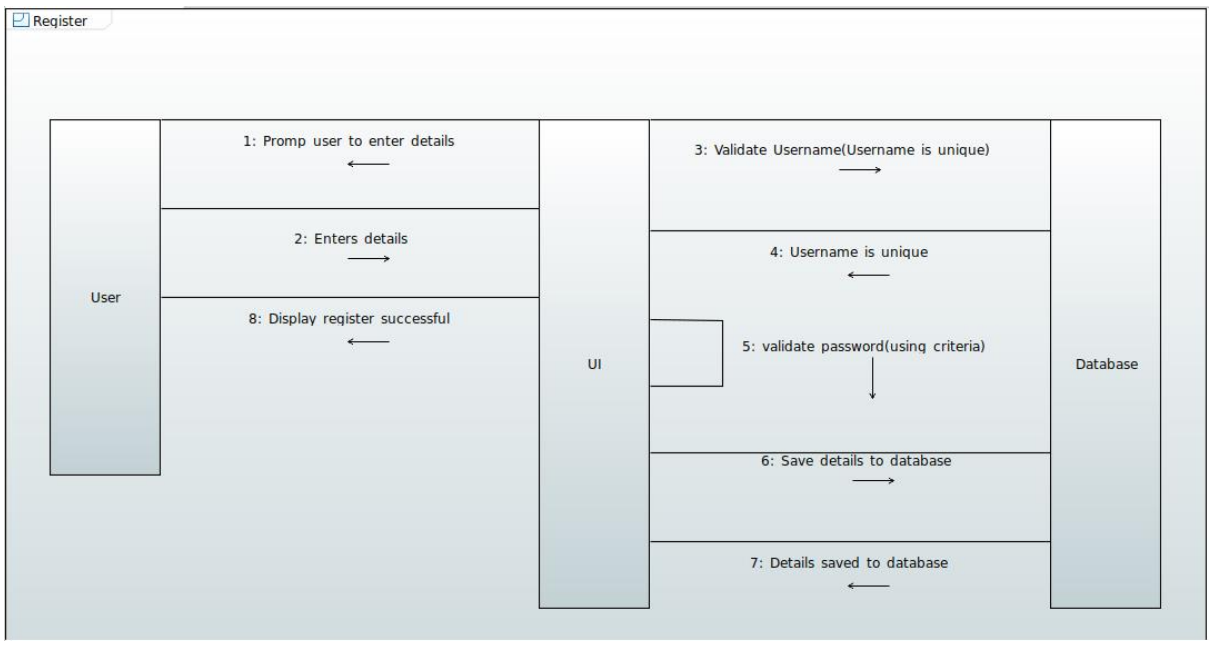


Figure 17 Register Communication Diagram using Papyrus

The UI will prompt the user to enter their details. The user will then enter their details into the UI. The UI will then send off the username to the database to check if it is a unique username. If the username Is found to be unique then the application will be notified and continue. The application will then validate the password against the given criteria. If the password is validated, then the details will be saved to the database and the UI will display a message to the user to say register successful.

Send message (private message)

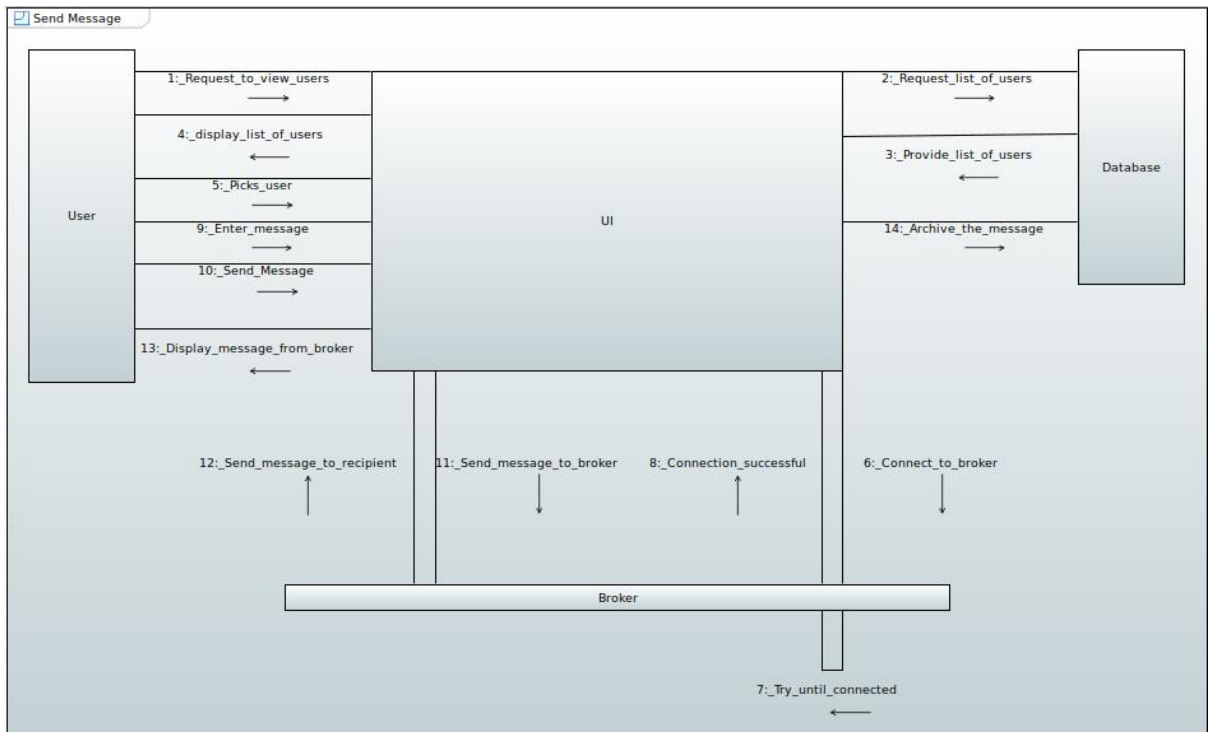


Figure 18 Send message Communication Diagram using Papyrus

The user will need to see a list of users in order to pick a recipient, so they will request to view the users which will be passed through the UI to the database. The database will then provide a list of users. The UI will take this list of user and display it to the user. The user will then pick a recipient for the message. Once this has been accomplished the UI will request to connect to the broker. The UI will continue to try the connection until successful. If the connection is successful, the application will be notified. The user can then enter and send a message. This message will be sent to the broker first. Once the message has been received by the broker it will be passed on to the recipient. The message will then be displayed to the recipient and the user. Finally, the message will be archived so that the users can see their message history.

Logoff

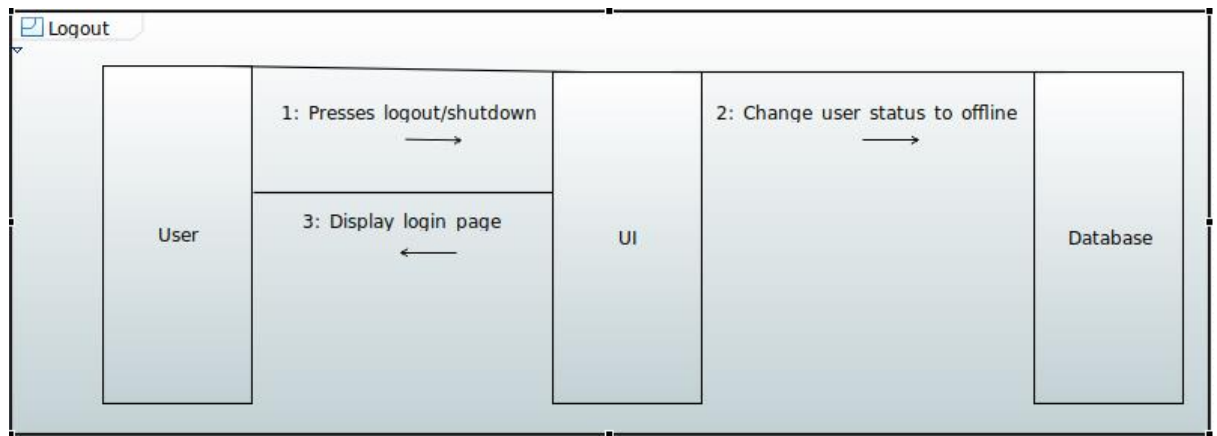


Figure 19 Logoff Communication Diagram using Papyrus

The user presses logout or shutdown on the UI. The user will then tell the database to change the status to offline. The database will change the status to offline. After this if the user has logged off the login page will be displayed.

1.6 Component Diagram

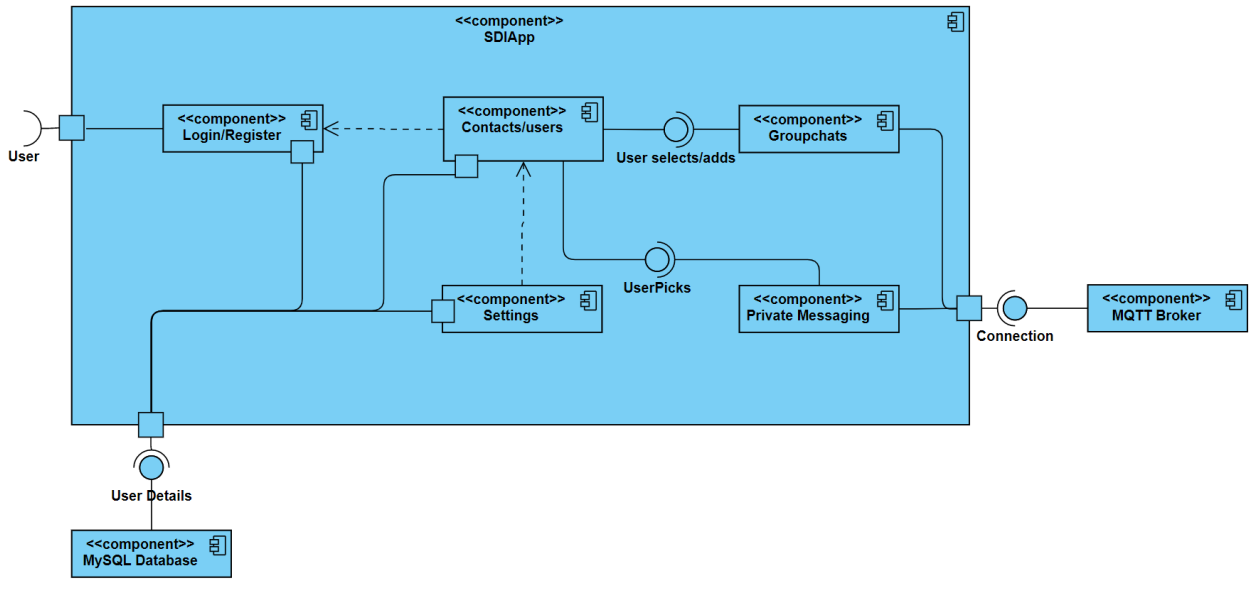


Figure 20 Component Diagram using VisualParadigm

The SDI app is controlled initially by the user. The user is the driving force off app. The login/ register system needs the user to enter their details in. the contacts and users is dependent on the login/ register system as without the users registering there are no users on the app. Settings is the same, as there would be nothing to change if the users don't exist. These three components all connect to the external database. The database provides the user details and makes them available to these three components. Login uses it to validate details, register to assign details, settings to update the details and contacts/ users to store the details for other components. The user can then either select a group chat/ add to a group chat using the details from the contacts/ user's component. Private messaging works in the same way only 1 user is picked. Both of these components use an MQTT broker which supplies them with a connection.

1.7 FSM Diagram

Represents whole SDIAPP

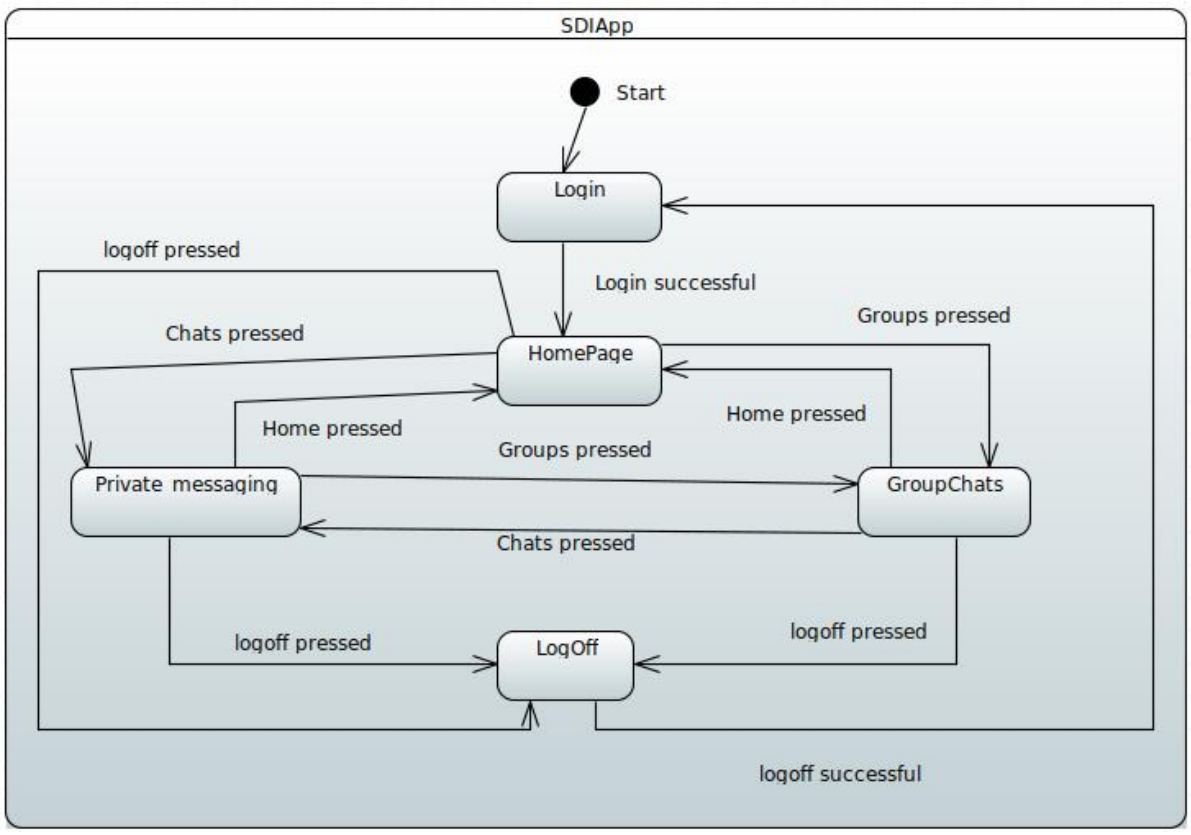


Figure 21 SDIApp FSM Diagram using Papyrus

This is a diagram to represent an overview of most of the application. The diagrams after will focus and expand on the key states represented in this diagram.

When the application starts the user will be required to login. If the login is successful, then the user will be moved to the home page of the application. From here the user can access all the other states. The user can either go to the group chats page or the private messaging page depending on which navigation button the user presses. This is a 2-way connection so the pages can then navigate back to the home page by pressing the home button. And all other pages can access each other as well. So, groups can be accessed by chats and vice versa. All pages on the application will have access to the logout feature. If the logout button is pressed on any page in the application, they will be in the logout state and if logout is successful taken back to the logout page.

Login

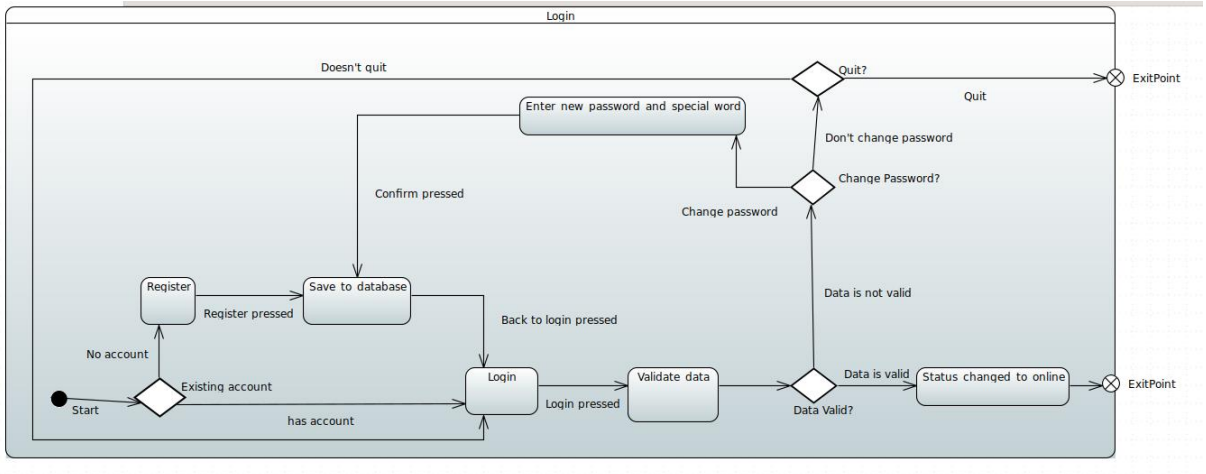


Figure 22 Login FSM Diagram using Papyrus

When the user goes through the login state they will need to check if they have an existing account. If they do not, then they will need to register an account. This will require details to be entered and when the user presses register the details will be saved to the database. Whether the user had an account or not they will then need to login. When details have been entered and login is pressed the data will be validated by the database. If the data is not valid then the user will need to decide if they want to change their password. If they do then they will be prompted to enter their special word and password. Once the password has been confirmed the data is saved to the database and they will need to login again. If they don't want to change their password, then they can either quit or not quit. If they don't quit, it would assume that they would want to try and their details again. Once the data is valid the user status will be changed to online, and they will move to the next state.

Send private message

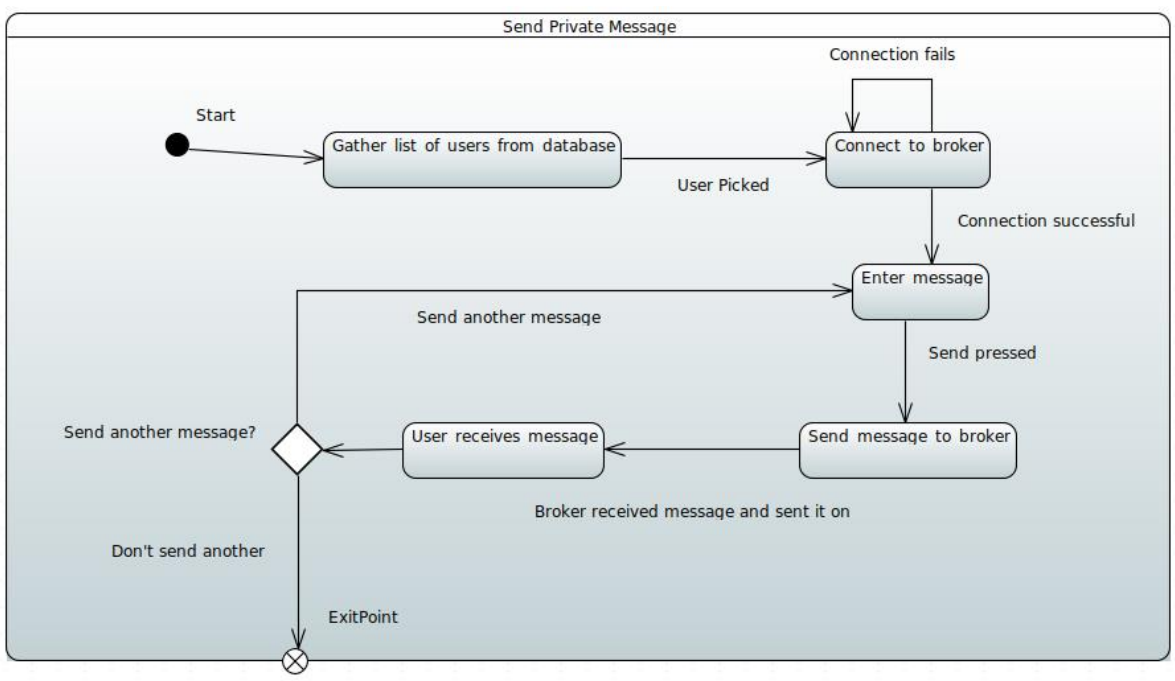


Figure 23 Send Private message FSM Diagram using Papyrus

At the start of the state a list of users on the application is gathered from the database. When the user picks one the connection to broker is started. If the connection fails, then the connection to the broker state will repeat till successful. When the connection successful, the user enters the enter message state. This ends when send message is pressed. The message is sent to the broker. When the broker has received to the message, the state transitions to the user receiving the message as the broker has sent the message to the user. The user can either send another message or choose not to. If the send another message they will be taken back to the enter message state. Else they will exit.

Group chats

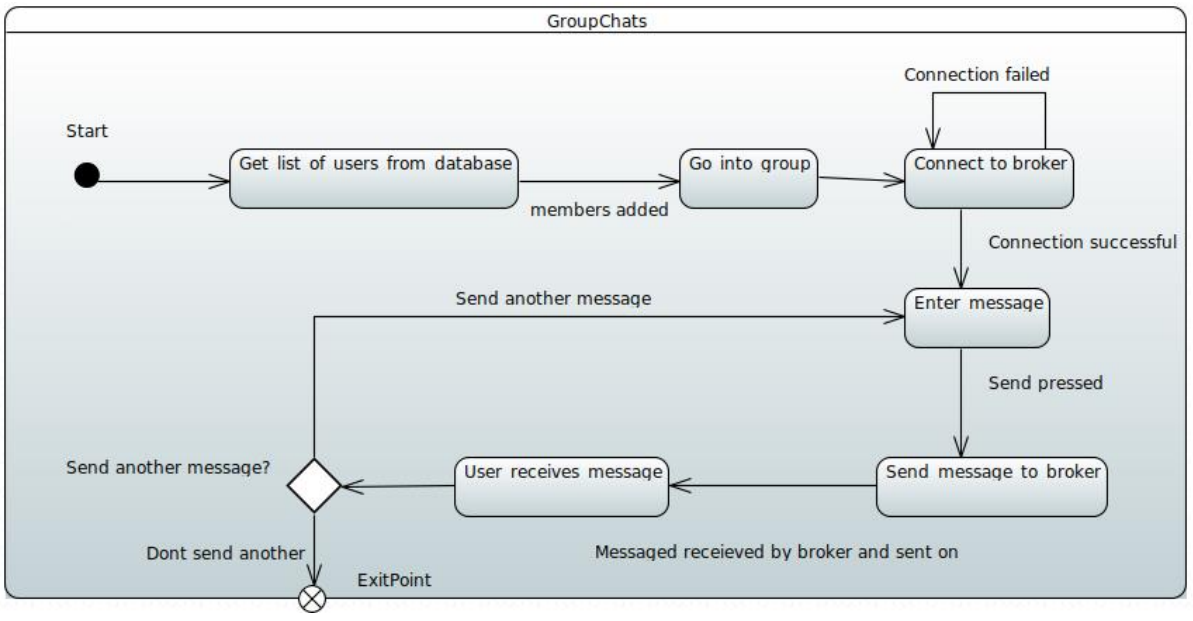


Figure 24 Groupchats FSM Diagram using Papyrus

The group chats work nearly in the exact same way as the private messaging, the only noticeable difference is that when the list comes back from the database, the user can add multiple users to the conversation.

Logout

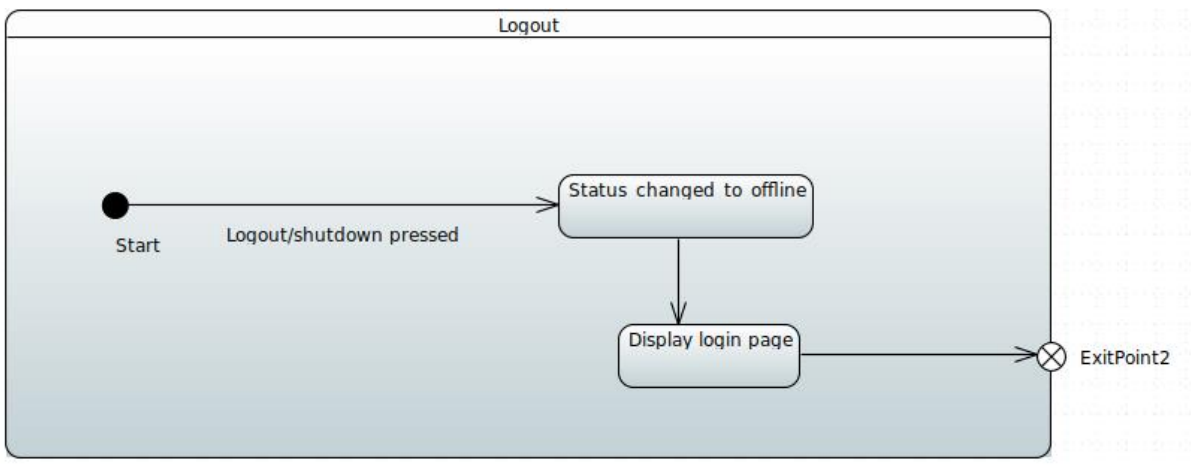


Figure 25 Logout FSM Diagram using Papyrus

When the logout state is started, the user will have pressed with shutdown or logout. Either way the user status is changed to offline, and the login page will be displayed.

1.8 UI mock-up

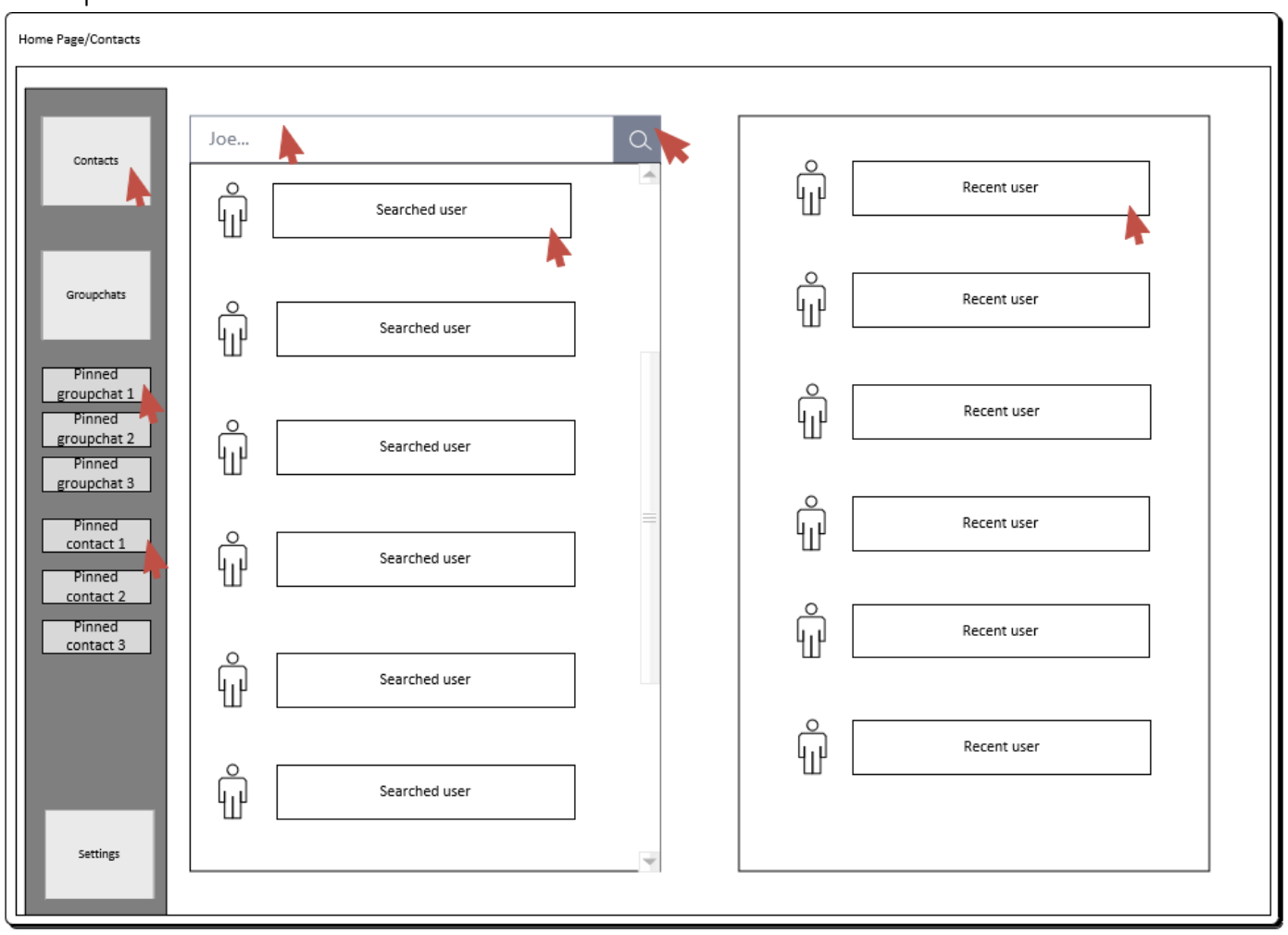


Figure 26 - UI Mock Up

This user interface shows the different aspects of the main graphical user interface (GUI). The main GUI is based around the contacts page, which allows the user to quickly access the different users within the application. On the left-hand side the navigation bar is found which contains the multiple buttons which will take you to different pages, the contacts page, the group chats page and the settings page. Apart from those three buttons, there are other buttons which can be pinned to the navigation bar as a favourites section for different group chats or contacts, these can be added or removed by the user by making changes to their favourite contacts. Next to the navigation bar, there is a search bar which allows the user to search for another user using names, this will display a list of users that are found matching the name searched, a scroll bar is included to the side to allow the user to scroll through the list if the search brings back a lot of users. On the right hand side of the GUI, a recent user list is displayed which will show the most recent users that the app user has been in contact with as they are the users that are most likely to be contacted regularly.

1.9 C++ Library's

GUI

Qt

Message Passing

Mqtt

Data Structures

STL

Generation of documentation from source code

DOXYGEN

Register An Account

Step 1: Open Application

Step 2: Press register button on login page

Step 3: Fill in all details ensuring your username is unique and your password is at least 8 characters long, contains a number and contains a special character

Step 4: Once happy press register

Step 5: If all is well then, a message saying “data saved should be displayed”. Else follow the error message

Forgotten Password

Step 1: Press Forgotten password on login page

Step 2: Enter your username and special word

Step 3: Enter your new password and then confirm it

Step 4: Press confirm

Step 5: If all is well then, your password will have changed, else follow the error message

Login

Step 1: Enter correct username and password

Step 2: Press login

Step 3: If successful message will be displayed “login successful”, else follow the error message

Change user details.

Step 1: When logged in click on settings page.

Step 2: On the left side of the page enter any detail you want to change and press update details

Step 3: If you want to update the user profile pic, press the button, select an image (JPEG work best) and press open. Once done you should see the image at the top of the page.

Send Private message

Step 1: If no previous chats exist, go to the “Accounts” page, this is found on the left hand side navigation bar. All users on the app are displayed here.

Step 2: Select a user from the table, and a chats page will be displayed, write the message you would like to send to the user you selected, then press the send button.

Step 3: Once the message has been sent you can find all your contacts (people who have either sent you a message or who you have sent a message to) on the chats page.

Create Group Chats

Step 1: Go to “Groups” page, this is found on the left-hand side navigation bar.

Step 2: Click “Create Group” button in left corner of page, a new page will be displayed.

Step 3: Select the users on the page you wish to be in the group chat and select if you want them to be a moderator or not.

Step 4: Press “Create Group” and the group will be set up, you will then be returned to the “Groups” page.

Find a Group Chat

Step 1: Go to “Groups” page and all group chats you are a part of will be displayed there.

Step 2: Select the group chat you wish to message from the table shown.

Manage your group chat (only accessible if you are a moderator or an admin of the chat)

Step 1: Go to “Groups” page, either go to a group you know you are a moderator of or create a new group (follow the create groups tutorial for the second option)

Step 2: Once you are on the group press the “Manage” button, located at the bottom of the screen by the message input.

Step 3: The manage groups page will be displayed. All members of the group will be displayed apart the admin of the group and yourself (whether you are a admin or a moderator. This stops you from removing yourself or an admin from the group chat).

Step 4: You can press either demote or promote buttons for selected users, to either promote a user to a moderator or demote a moderator to a user.

Step 5: Once done press “Main Menu” on the side bar. This will take you back to the main user area.

Logout

Step 1: On any page look at top bar of application, press the button closest to the right (the one with the logoff symbol).

Step 2: Your status will have been changed to “offline” and the login page will be displayed.

Shutdown

Step 1: On any page go to the top bar of the application, click on the button closest to the left (the one with the shutdown logo).

Step 2: the application should then be shut down

Admin/ Moderator part of application

Admin

Step 1: Log into admin using **username:** sysAdmin and **password:** admin123

Moderator (if you have moderator permissions)

Step 1: You will have access to all of these pages via the bottom of the navigation bar at the side on your user accounts. If you are a moderator, you will see buttons for “manage users”, “manage groups” and “manage mods”. If you are a regular user, you will not see these buttons.

Manage Users

Step 1: Click on the “manage users” page from the navigation bar either from the admin account or user account that has been given admin permissions.

Step 2: Once on the page select a user.

Step 3: You can either delete the user or reset their password to password using the buttons on the bottom. The buttons at the bottom will only be active when a user is selected on the table. (Just press on a user on the table to select)

Manage Groups

Step 1: Click on the “manage groups” page from the navigation bar either from the admin account or user account that has been given admin permissions.

Step 2: Once on the page select a user (each row represents a user not a group, the group ID column shows what group each user is in).

Step 3: Using the buttons at the bottom to delete the group or remove the user from that groups. The buttons only work when a user is selected.

Manage Mods

Step 1: Click on the “manage mods” page from the navigation bar either from the admin account or user account that has been given admin permissions.

Step 2: Once on the page select a user

Step 3: Using the buttons at the bottom to promote or demote users to moderators or users.

Details on how data is saved and loaded into the memory

As we are using a database, whenever we need data from the database, we need to first establish a connection to the database. From this we do not need to use vectors or lists, we can simply extract certain parts of data and put them straight into a variable. This maybe more Hassell to set up but is easier to use than an array, vector or list, simply because there is no need to sort or organise anything. We get values straight from the database and can assign any individual detail to a variable and then we can simply append the value if needed and then send it straight back to the database.

For creation of user accounts and other details, the data is taken from line-edits within qt and once again stored into variables. These are then used throughout the program or they are stored into the database.

Loading images works in a similar way except a blob file type is used on the database to store the image. And when the image is taken out of the database and store in the memory a QImage type is used. The image is stored in a global variable.

Concurrent Programming within our project

QT is a very thread safe program with lots of safety measures and tools put in place to easily produce a threads QT C++ program. But QT also uses threading with general applications as well. For instance, the main class in QT runs on its own thread when the program is launched as stated by (QT, 2021).

Source Code management

For source code management we used GitHub throughout the project. Everything was stored on the GitHub aside from the report.

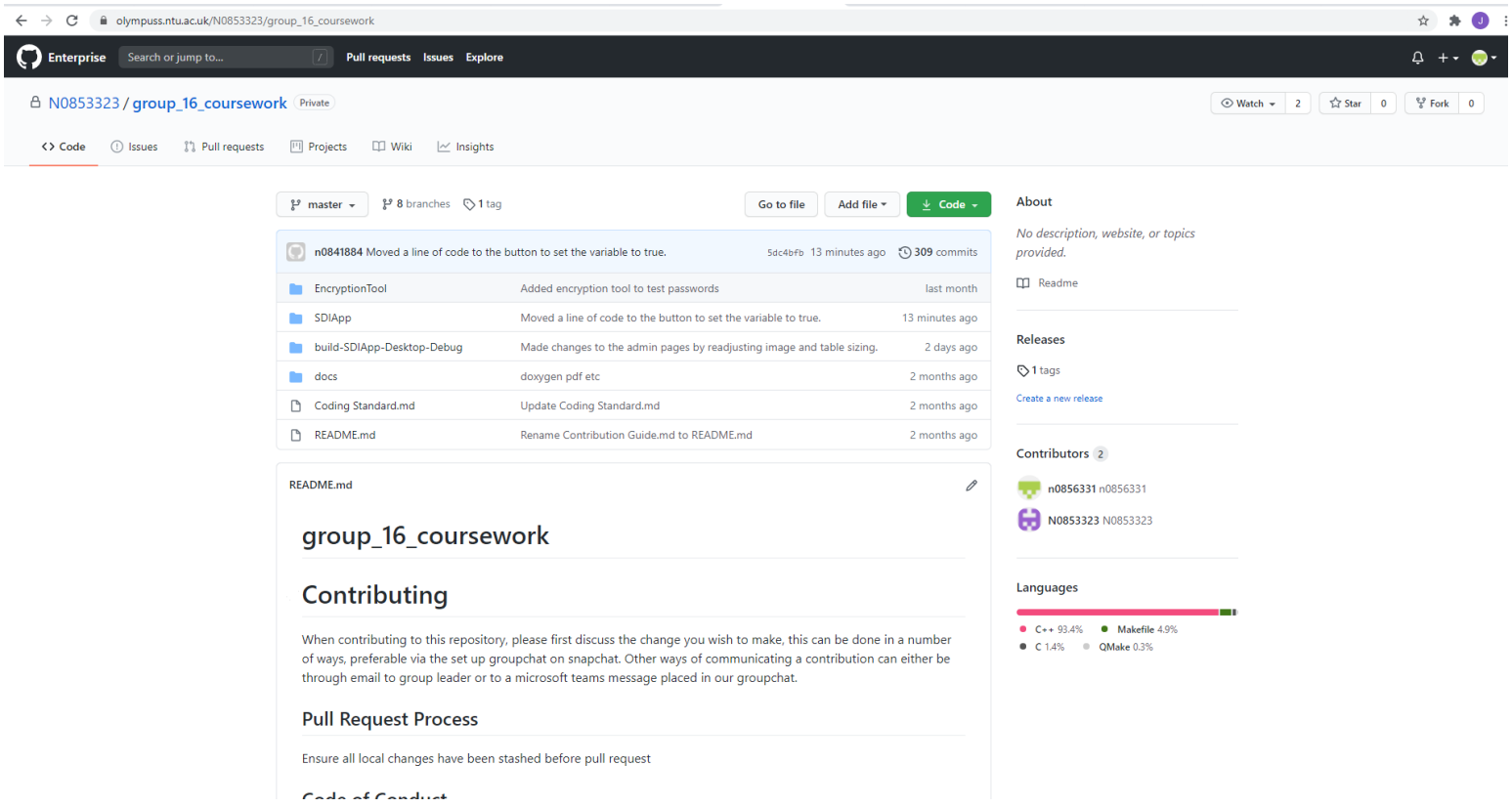


Figure 27 - Our GitHub

Conclusion and Future Work

The main requirements for the system in summary were that the system should look user friendly and should operate with a client connection to the broker, not clients connecting directly together. User details were to be stored locally and securely and can be changed at the user's discretion. The next requirements were to make sure users could create chat rooms and create roles based on who created the chat room and people who were chosen to monitor the chat. And the final requirements state that users statuses should be known and offline messaging should be considered. The majority of these requirements have been addressed and accounted for, and have been implemented into the application. With more time and maybe some better planning the full list of requirements could have been realised.

Although lots of thought and planning did go into the development of the application. The diagrams produced show a clear inspiration and guide for the structure of the program. Plenty of thought went into how the system should work, for example the login system, the sequence of events that would lead to a user being able register their account and login to the application. The diagrams do document the sequence of events and the thought processes behind the implementation of those features very well. There were some alterations that had to be made, not because the diagrams were wrong but in terms of planning out a project and then executing that project certain expectations can lead to miss conceptions about what and in what way you will be able to achieve these goals.

The main tools and framework remain consistent with what was stated at the start of the report. The main tools for diagram creation were Cacoo … and Papyrus. This again being due to complications with Papyrus meaning diagrams couldn't be made as efficiently as required and without being able to show all the necessary details. So, for the diagrams that were affected by this, the decision was made to use alternative software which were more suitable for our needs, not to mention easier to use for those particular diagrams. In terms of software tools and libraries used for the implementation, that remains consistent as well. QT was used to general creation of the UIs and addition of the source code to each specific component. STL was used but not very much as there wasn't as much need for the features it offers due to the use of a database instead of a textfile based system. MQTT was used as the communication tool between users and DOXYGEN was used to automatically generate the documentation. QT's internal libraries were heavily used for the development of the program.

There are several ways we would like to improve the application. One way to improve the application is giving the ability for users to make video calls to other users. This has become a popular feature in the business world with 78% of corporate companies using video calling software as stated by (GetVoip, 2020). Another feature which would greatly improve the application is the ability for users to transfer files to another user or a group of users. This feature is very helpful for collaboration within companies and therefor can greatly increase productivity. A further feature we could add which would improve the ease of communication and make the app a better experience for the user is emojis. "Emojis have an important role in interpersonal communication" (Chairunnisa & Benedictus, 2017).

Group Experience

The group believes that gaining more practice and experience with projects like this would be an important steppingstone for making better use of source code and libraries. Being one of the first project like this for the majority of the team came as a surprise on how much had to be learnt in order to implement the project. It is important to address this was a good teaching experience for us and that given a project like this again we would be more equipped to implement it.

All in all, the project was completed to the best standard we could possible do, given our lack of experience. The group worked well together and were ready and eager to help each other out. To summarise what went well, all of the implementation we believe to have gone well. We are very pleased with the chooses that were made and very happy with way it turned out.

In terms of what could be improved we believe that although implementation was most definitely a highlight for us, as all members except one were involved with implementation, we believe there were still areas to improve in. For instance, refining our classes and making more us of the STL library. With more experience no we have done this project we would know more about where to start for another project and know what we would most likely try out in that case.

Aside from the obvious skill being more confidence with programming abilities, skills like working within a coordinated team have been strengthened with use of this project. A more solid understanding had been gained by the group on the techniques and strategies involved with the creation of software which was an interesting experience. Lessons from this project will most certainly be built upon for 3rd year, being time management, teamwork and honestly the ability to research exactly what you need has been an import part of this project.

For the learning experience, we all feel as though more time should have been dedicated to QT creator especially during term 1. Obviously, we understand that the module has other priorities in the first term but giving students tasks to do in QT creator even if they were more STL based would be a more gradual and better way to introduce QT creator to the students. Making the point where students start the coursework less confusing and stressful. Giving students a more solid foundation to work from rather than them having to go away and figure everything out from scratch would have been a better approach we feel. Especially when it comes to MQTT to which finding resources for was very difficult.

Individual Contributions

1) Abstract: Milosz
2) Plagiarism Declaration: Milosz, Jonathan, Josh
3) Version History of Application: Milosz, Josh
4) Version History of Report: Milosz, Jonathan
5) Introduction: Milosz, Jonathan
6) Background Research: Jonathan
7) Requirement List: Milosz, Jonathan, Josh
8) Additional Requirements List:
9) Risk Analysis: Milosz, Jonathan, Josh
10) Gantt Chart: Milosz, Jonathan, Josh
11) Use Case Diagrams & Description: Jonathan
12) Activity Diagrams & Description: Jonathan
13) Class Diagram & Description: Milosz
14) Sequence Diagram & Description: Jonathan
15) Deployment Diagram & Description: Jonathan
16) Communication Diagram & Description: Jonathan
17) Component Diagram & Description: Jonathan
18) Finite State Machine Diagram & Description: Jonathan
19) UI Mock-up Diagram & Description: Josh
20) C++ library: Jonathan
21) Application Development:
    a. Josh: all messaging functionality
    b. Jonathan: UI design, extra features
    c. Milosz: Admin, database functionality
22) User Manual: Jonathan
23) Conclusion and future work: Milosz, Jonathan, Josh
24) Group experience: Milosz, Jonathan, Josh

References

Chairunnisa, S. & Benedictus, A., 2017. *Analysis of Emoji and Emoticon Usage in Interpersonal Communication of Blackberry Messenger and WhatsApp Application User.* [Online]
Available at: https://www.nepjol.info/index.php/IJSSM/article/view/17173
[Accessed 02 May 2021].

GetVoip, 2020. *The State of Video Conferencing in 2020 [50 Statistics].* [Online]
Available at: https://getvoip.com/blog/2020/07/07/video-conferencing-stats/#:~:text=45%25%20of%20teams%20use%20video,replace%20audio%2Donly%20conference%20calls
[Accessed 02 May 2021].

MediaKix, 2021. *The 8 Biggest Messaging App Statistics Advertisers Must See.* [Online]
Available at: https://mediakix.com/blog/messaging-app-statistics-advertisers-need-to-consider/
[Accessed 02 May 2021].

Mio, 2019. *Workplace Messaging Report - Slack, Microsoft Teams; Webex Teams.* [Online]
Available at: https://dispatch.m.io/mio-workplace-messaging-report/
[Accessed 02 May 2021].

QT, 2021. *Threading Basics | QT 5.15.* [Online]
Available at: https://doc.qt.io/qt-5/thread-basics.html
[Accessed 02 May 2021].

SPECTRM, 2020. *Messaging Apps Have Taken Over | Usage & Growth Statistics.* [Online]
Available at: https://spectrm.io/insights/blog/messaging-app-statistics-most-popular-communication-method-2020/#:~:text=Businesses%20and%20users%20exchange%2020,is%20active%20in%20messaging%20apps
[Accessed 02 May 2021].

Plagiarism Declaration

This report and the software it documents is the result of my own 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 recognise the right and duty of the Board of Examiners to take appropriate action in line with the university's regulations on assessment.

Signed by:

Name: Milosz Bednarski Student ID: N0853323

Name: Jonathan Archer Student ID: N0856331

Name: Josh Wade Student ID: N0841884

Third party resources used:

- https://doc.qt.io/QtMQTT/qtmqtt-simpleclient-example.html - used on messagingpage.cpp
- https://www.youtube.com/watch?v=oaP20lrd6i8 – used in all encryption and decrypt functions on the mainwindow.cpp file, register.cpp, forgotten_password.cpp and settings.cpp.