Last Minute Messenger

Client – Server Messenger Application

Software Workshop – Team Project

Team Mumbai

Nabeel Anwar (ICY) – 1333992
Ioana Avirvarei (MSc) – 1858584
Maurice Noel Bouniol (MSc) – 1854011
Roman Gaev (MSc) – 1751175
Ali Oztas (MSc) – 1851731

Contents

Team organisation

Sub – teams

To allow focused and concurrent development across all aspects of the project, the team was sub divided into smaller teams. Each sub team is therefore responsible for delivering a particular part of the project. To ensure that each section of the project can progress with minimal disruption, at least two members of the group were allocated to each sub – team. The intended effect of this organisation is to ensure high quality output and to identify and rectify issues early. Members of each team were chosen based on their strengths and weaknesses.

Team:	Client – Server
Members:	Roman, Maurice, Ali, Nabeel
Team:	Database
Members:	Maurice, Roman, Nabeel
Team:	GUI
Members:	Ioana, Ali
Team:	Software Engineering
Members:	Nabeel, Ioana
Team:	Project Management
Members:	Nabeel, Maurice

Task allocation

Each sub – team was allocated tasks relevant to their particular focus within the project. The tasks, were then further divided between members of each sub – team to ensure maximum efficiency and concurrency. This system ensured no group member was burdened with more work than they could possibly complete. Other members of the group were also able to cover and support each other where and when required as a result.

Client – Server
Tasks:

- Create a method for the user to register for the chat service. This registration method allows the user to choose a unique username and a password as well as input a name.
- Create a method for a registered user to login into the chat service once they have registered. The input username and password are checked on a database by the server before access is granted.
- Create a method for the clients to send and receive messages to and from one another, via the server.
- Create a method to allow a client to communicate with a specific client via the server.
- Create a method to allow a client to create a group chat with several other clients.
- Create a method to store the chat history of the clients temporarily on the server and permanently on the database. New message history is then appended to the chat history stored on the database.
- Design a simple protocol to allow communication between the different clients and the server.
- Implement JDBC to allow communication between the server and the database.

Database
Tasks:

- Create a table to store the client (users) login information including the username, its corresponding password and name. Each new user is also assigned a unique ID.
- Incorporate into the users table, information about the current status of a client and if they are linked to any particular group chat.
- Create a table groups which stores information regarding which user is in which group. Each group is assigned a unique ID.
- Create a table to store permanently, messages sent and received for each user/or group chat. This table also stores time stamps for the sent messages.
- Design the required SQL queries to carry out the intended functions of each table.
- Establish the connection between the Server and database using JDBC to carry execute the queries.

GUI
Tasks:

- Design and implement the GUI for the main menu screen, on which the user is presented with options to register and/or sign in.
- Within the main menu GUI, allow a registered user to enter their credentials and log in to the messaging service.
- Design and implement the GUI for the registration process, which is displayed when the user selects this option from the main menu.
- The user can input the required details to register for the messaging service in the registration GUI.
- Design and implement the main chat GUI the client is presented with, once login into the service.
- Once logged in the GUI can display any messages which are sent and received by some user as well which other clients are currently online. A user can initiate a private chat session with another client or create a group with several other clients.
- Design and implement the GUI for both private and group chats which are separate from the "public" chat room in which all users can communicate.

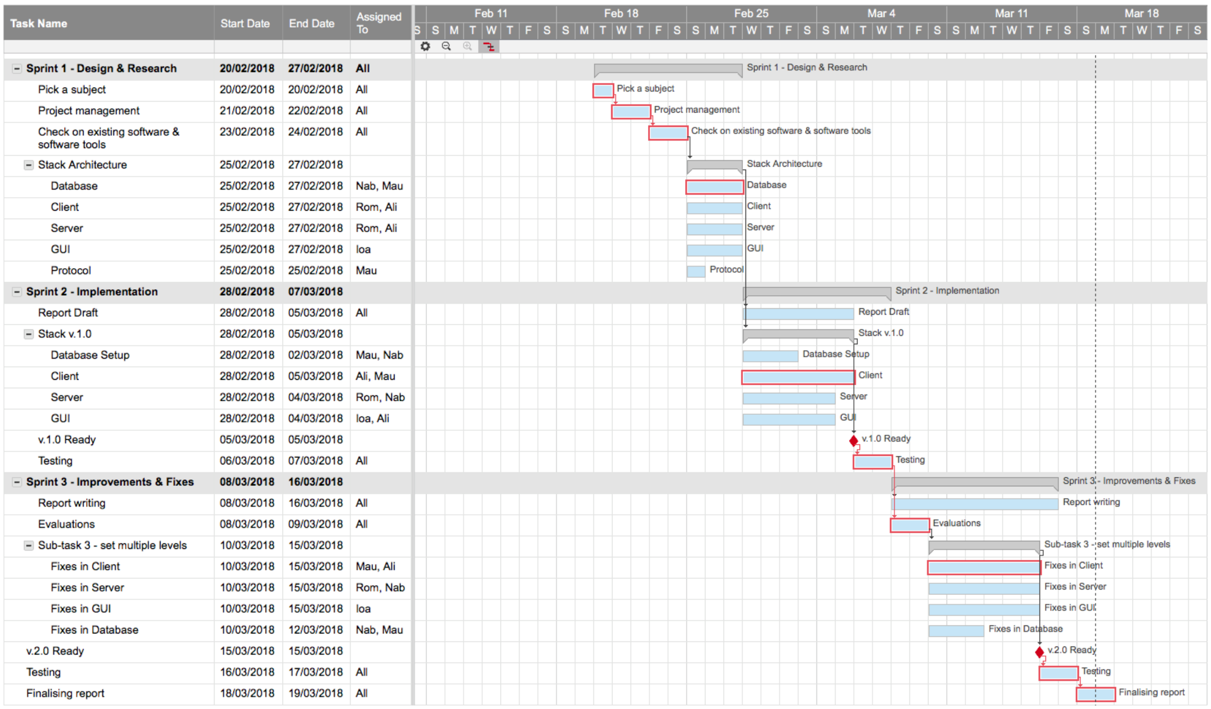
Software Engineering
Tasks:

- Document the specification for the messenger service including both functional and non – function requirements of the system.
- Design and document the system architecture for the messenger service.
- Detail information about the system architecture's main components; client, server, database and protocol.

Project Management
Tasks:

- Schedule regular meetings to track progress and resolve any issues.
- Keep a record of discussions at meetings.
- Coordinate which tasks each team is has completed, currently undertaking or have been completed.

Gantt Chart



Task Name	Start Date	End Date	Assigned To
Sprint 1 - Design & Research	**20/02/2018**	**27/02/2018**	**All**
Pick a subject	20/02/2018	20/02/2018	All
Project management	21/02/2018	22/02/2018	All
Check on existing software & software tools	23/02/2018	24/02/2018	All
Stack Architecture	25/02/2018	27/02/2018	
Database	25/02/2018	27/02/2018	Nab, Mau
Client	25/02/2018	27/02/2018	Rom, Ali
Server	25/02/2018	27/02/2018	Rom, Ali
GUI	25/02/2018	27/02/2018	Ioa
Protocol	25/02/2018	25/02/2018	Mau
Sprint 2 - Implementation	**28/02/2018**	**07/03/2018**	
Report Draft	28/02/2018	05/03/2018	All
Stack v.1.0	28/02/2018	05/03/2018	
Database Setup	28/02/2018	02/03/2018	Mau, Nab
Client	28/02/2018	05/03/2018	Ali, Mau
Server	28/02/2018	04/03/2018	Rom, Nab
GUI	28/02/2018	04/03/2018	Ioa, Ali
v.1.0 Ready	05/03/2018	05/03/2018	
Testing	06/03/2018	07/03/2018	All
Sprint 3 - Improvements & Fixes	**08/03/2018**	**16/03/2018**	
Report writing	08/03/2018	16/03/2018	All
Evaluations	08/03/2018	09/03/2018	All
Sub-task 3 - set multiple levels	10/03/2018	15/03/2018	
Fixes in Client	10/03/2018	15/03/2018	Mau, Ali
Fixes in Server	10/03/2018	15/03/2018	Rom, Nab
Fixes in GUI	10/03/2018	15/03/2018	Ioa
Fixes in Database	10/03/2018	12/03/2018	Nab, Mau
v.2.0 Ready	15/03/2018	15/03/2018	
Testing	16/03/2018	17/03/2018	All
Finalising report	18/03/2018	19/03/2018	All

This Gantt chart provides a visual timeline of how the project progressed over time. It allowed the team to keep track of progress and address any issues encountered.

Description of the system

Concept

As part of this project, team Mumbai will develop an instant messaging application, intended for use in a professional working environment. New users will be able to sign up to use the messaging service and current users will be able to log in to continue using the service. This messenger application will allow a registered user to communicate with other registered users, whom are currently online. As part of the application the client will be able to create private chat rooms with another individual or even a group. Users will also be able to access and review previous short and long – term messaging history with other users, stored in the server and database respectively.

Specification

Functional Requirements:

- The messenger system shall allow an existing client to login using a username and password.
- The username must be unique.
- The messenger system shall provide new clients with a method to create a new account.
- The messenger system shall allow the user to logout and exit.
- The messenger system shall enable the client to view a status (online/offline) of other users.
- The messenger system shall provide access to previous chat histories.
- The messenger system shall allow the client to initiate a chat with another online user.
- The messenger system shall facilitate engagement with multiple chats with different users.
- The messenger system shall provide a GUI to display both the functionality of the system and user's inputs.
- The history of every chat should be available by scrolling up, refreshing every few lines.
- The messenger should allow access to a contact list.
- The messenger should provide a timestamp for every message.

Non – Functional Requirements:

- The messenger system shall authenticate a client’s username and password within five
- seconds or less.
- The messenger system shall decline access if the username or password is entered
- incorrectly.
- The messenger system shall prompt the user to enter a correct username and the corresponding password if access is declined.
- The messenger system shall only accept usernames of length 5 to 20 characters only including letters.
- The messenger system shall only accept passwords of length 6 to 12 characters with at least one number and letter.
- The messenger system shall only accept usernames and passwords with letters and
- numbers.
- The messenger system shall check that the entered new username does not already exist in the database before accepting it.
- The messenger system shall be capable of 24 hours a day, 7 days a week availability.
- The messenger service will only display users who are currently online in a list.
- The messenger system shall only allow communication to other online users.
- The messenger system shall provide a simple GUI for the user to interact with.
- The messenger system shall allow inputs found on a standard QWERTY keyboard.
- The messenger system shall be able to send a message within 1 minute of sending it.
- The messenger system shall allow users to download files onto their computer.
- The messenger service shall store up to the current session’s chat history on the client, which later appended to the database’s long – term chat history.
- The messenger system shall limit the creation of a group chat to online users only.
- The messenger system shall only send messages which a blank containing only spaces and new lines.

UoB

LMM

mumbai

System design

Architecture

The architecture of this messaging application and service, follows a client – server model (figure 1). A single computer will be used to host a server to which other machines are able to connect to access the messaging service. The clients wishing to connect to the server to access the messaging service, will need to interact with it via a GUI. The GUI will allow the user to carry out all the actions facilitated by the service such as signing up, logging in and send or receiving messages and files. All requests by the clients will be sent to the computer acting as the host server, which will also connects to the database.

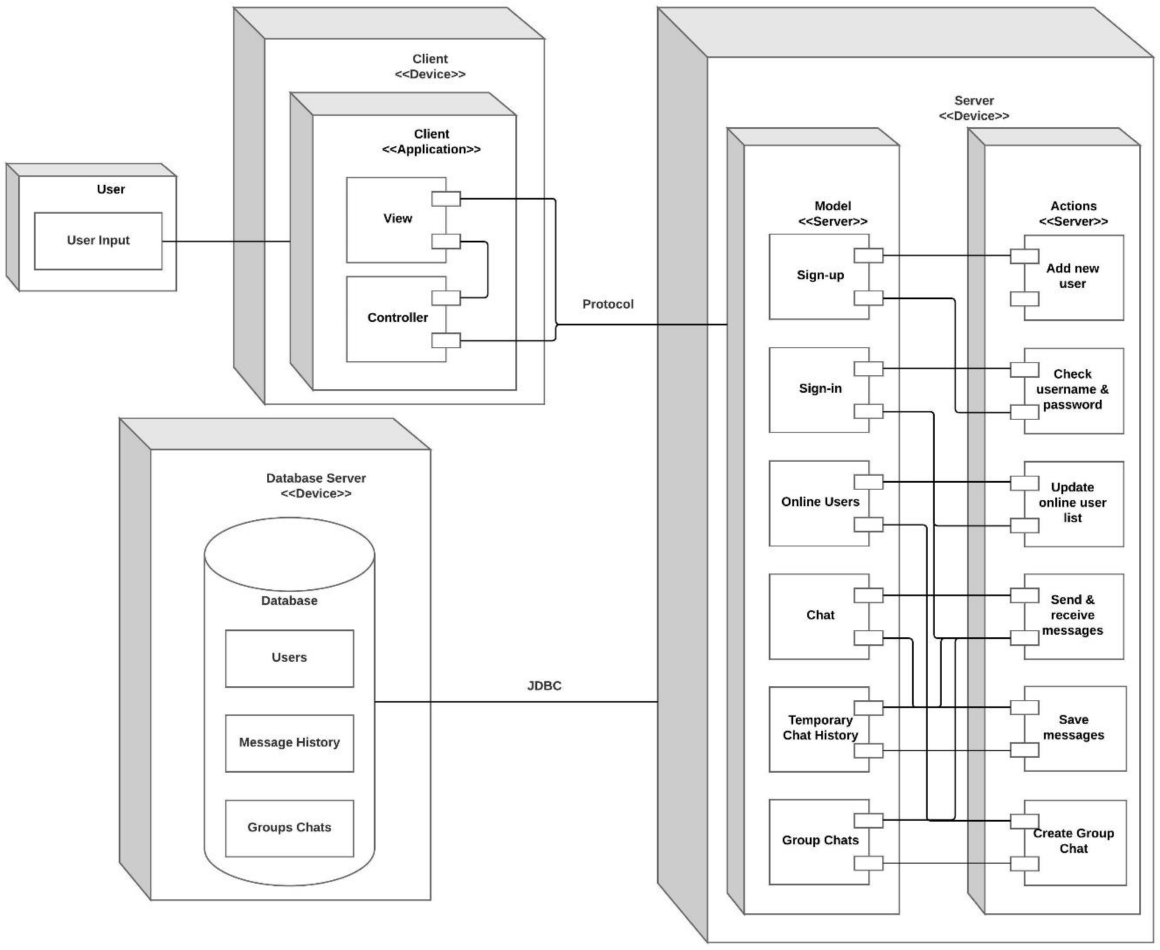


Figure 1. Overall system architecture for the client – server messaging service.

The database stores all the required information relating to the clients, such as login credentials and long – term chat history. The database will also facilitate storage of group chat information. Whereas the server is also able to store a temporary chat history, which is later appended to the permanent chat history in the database. The server performs all the actions available for request from the user, which may also be relayed to the database. The server utilises both a proprietary protocol designed for communication between the clients and the server and JDBC for communication the database.

Database

The database is designed with the intention of serving two specific functions. The first function of the database is to store login credentials for each client using the service. To achieve this key function, the database contains a table handling data regarding the clients. The said table, referred to as 'Users', stores the names, usernames and their corresponding password (figure 2). The user inputs these fields when they register to use the service for the first time, which are stored on the database. Each user is assigned a unique ID as an integer, which is used as a primary key. This table also stores a value for each user indicating the they are online or offline.

When a user subsequently chooses to login again to the service, the input username and its corresponding password are checked against this database. If the user input matches the stored credentials, the client is consequently given access to the chat service to communicate with other clients. Other information not critical to login, is also stored in the 'Users' table. Such information is only required to be input upon registration of the new client. Information about the user's current status is dynamically updated by the server through SQL queries, as and when the user logs in or out.

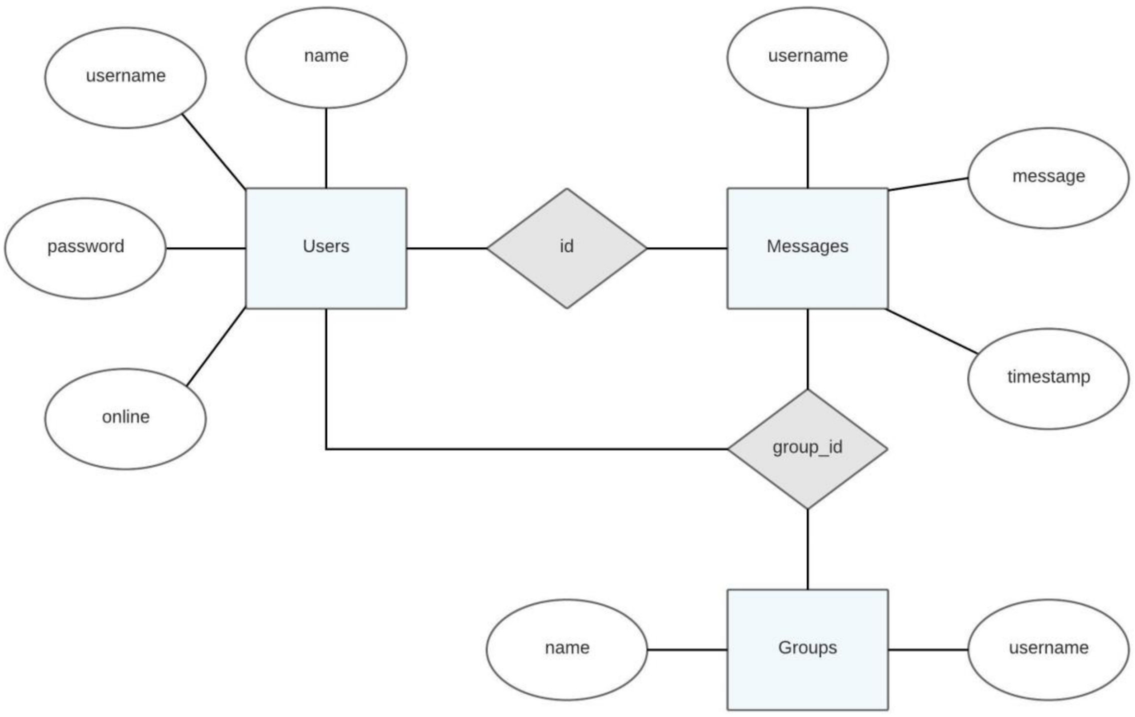


Figure 2. Database ER diagram.

The other key function of the database is concerned with the long – term storage of the chat history between clients. The table 'Messages' will store the chat history itself as well as it's associated data, such as date and time of the chat, stored as a timestamp (figure 2). The messages themselves are stored as text. The primary key for this table is 'id' which is just an integer number assigned to each entry in the table. This database structure allows efficient method to locate the most recent chat history for a user as the ID is associated with the 'Users' table. The chat history is first temporarily stored by the server for more efficient access. This chat history is then appended to the existing chat history for each user on the database.

The final table is responsible for holding information about group chats (figure 2). This table stores the usernames of each user and the group chat to which they associated with. Each group is assigned a unique group ID as an integer, to allow distinct identification of each group chat. This table is linked to the messages table via the same key, to allow the association of the messages sent between clients within the same group.

Server

The server is designed to perform all the of actions required of the messenger application, thus housing the business logic associated with the chat services. The overall function of the server is to allow new users to register for the service and allow existing clients to communicate with one another. The server can communicate with multiple clients through a defined protocol as well as to the database which stores the client and group chat information as well as the chat history. The connection with the database is made possible by implementation of the JDBC Java API. The server can communicate simultaneously and seamlessly with both the connected clients and the database. All the communication between the clients and the database take place via the server which relays the relevant information to the correct destination.

When a new client attempts to sign up to the service, the server first checks that the username doesn't already exist in the database. If this condition is satisfied, the user is then able to register with the inputted username and a password of their choice. The server will update the database with a new entry for the new client with an SQL query via JDBC connection. Equally the server will query the database to check that a client wishing to log into the service has entered the correct username and password. If the entered credentials an entry in the database, the server will grant access to the client.

Upon logging in to the chat service, the server will both update the database to show that the client is now online and retrieve a list of other clients which are also online. The list of online clients is then relayed to the client who is then able to see this in the GUI as a list of contacts. When a message is then sent to other online clients, it is first sent to the server using the protocol. The server then relays the message to the other client or client and stores the message temporarily in its memory. The messages are then later appended to the database for each user. This approach was chosen to improve the efficiency of retrieving the most recent chat history.

Should the user to send a message to another individual client or group of clients, the server will initiate a new private chat room for the clients involved. Such requests as before, are sent to the server which will create the group using information obtained from the database about the other clients. The server will query the database to assign a new group ID and the users associated with that group chat. These features are only accessible if the other clients in question are online during the process. The messages sent to and from clients within the groups are also saved first temporarily in the server, then later appended to the database.

The activity diagram (figure 3) demonstrates how the server plays a vital role in communication between clients and the database. All communications go via the server and which handles the appropriate requests and actions that need to take place.

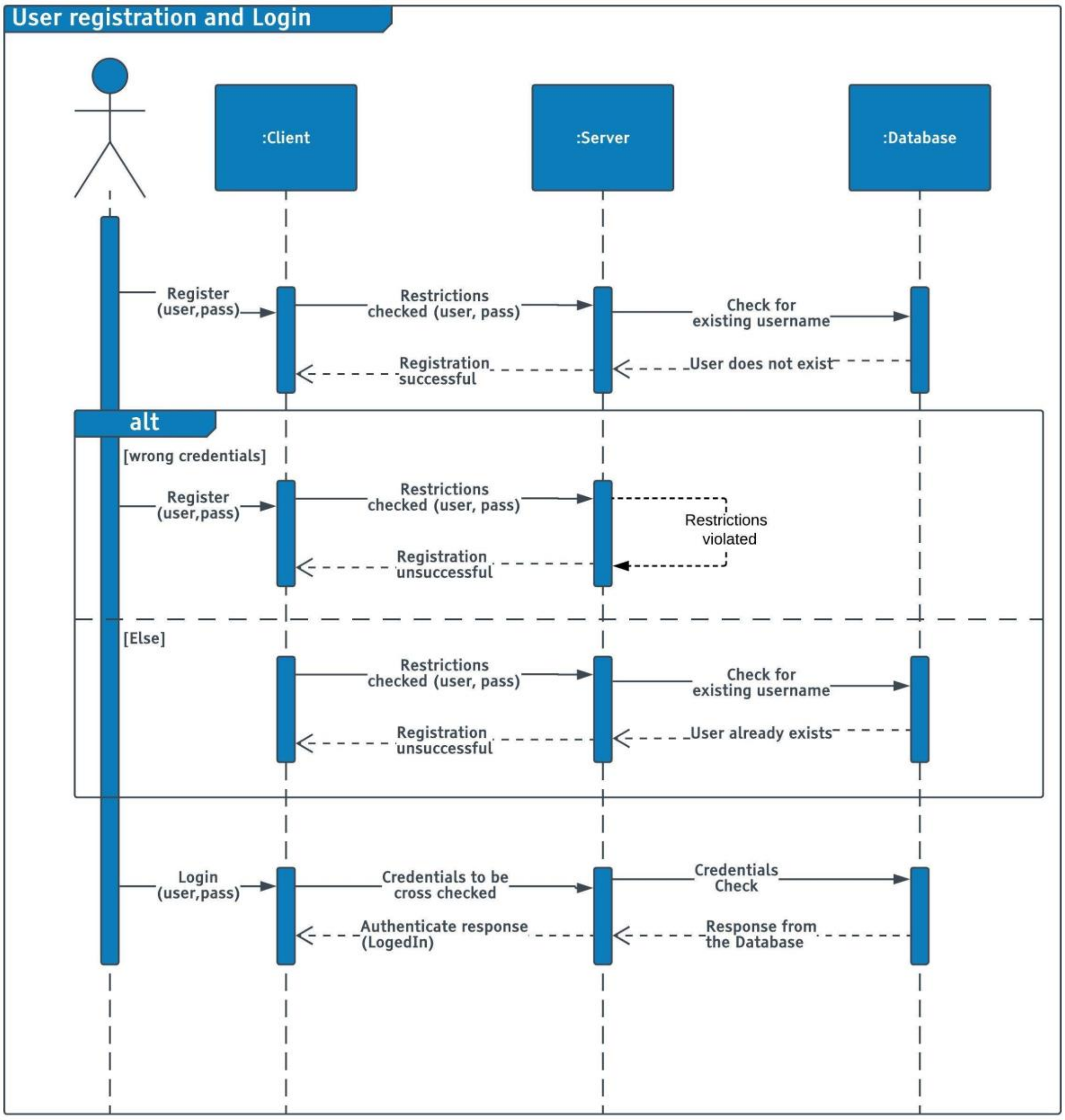
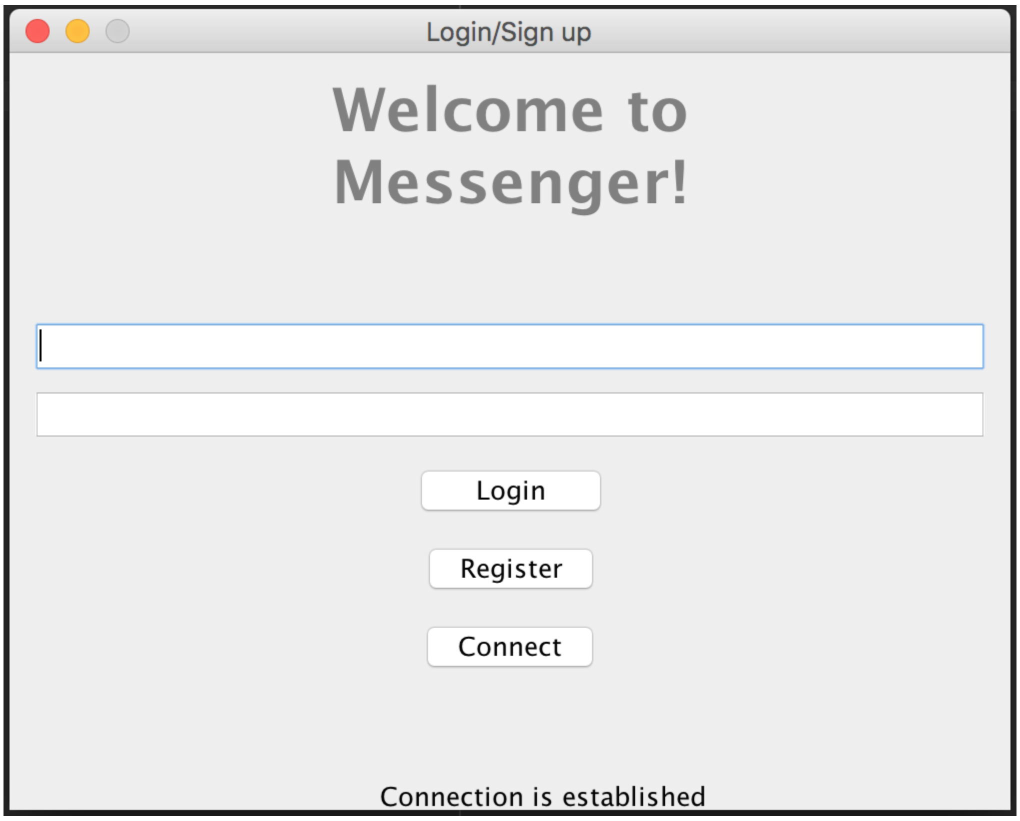

User registration and Login
:Client
:Server
:Database
Register (user,pass)
Restrictions checked (user, pass)
Check for existing username
Registration successful
User does not exist
alt
[wrong credentials]
Register (user,pass)
Restrictions checked (user, pass)
Restrictions violated
Registration unsuccessful
[Else]
Restrictions checked (user, pass)
Check for existing username
Registration unsuccessful
User already exists
Login (user,pass)
Credentials to be cross checked
Credentials Check
Authenticate response (LogedIn)
Response from the Database

Figure 3. Activity Diagram

Client

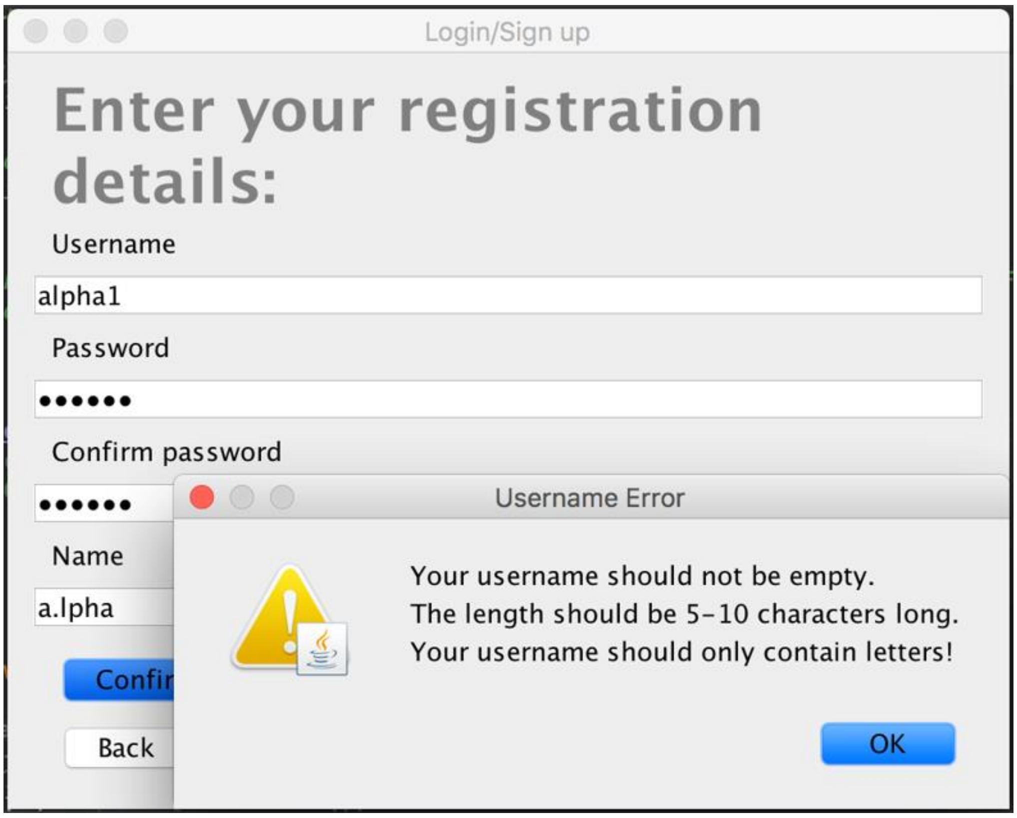
The client consists of several different views which perform the requests or actions of the user. When the user opens the application on their system, they are presented with a main start up screen (figure 4). This screen presents the client with several different options including registering, logging in and checking the connection. The user can input their credentials to sign into the service on this view and attempt to connect to the service. For a new client, a new view is presented when register is requested (figure 5). The buttons on these views send the relevant requests to the server for processing. The views will then display the appropriate feedback received from the server. As with the connect button which will give feedback as to whether the client application is able to communicate with the server or not.



Connection is established

Figure 4. Main view to register and/or login to the service.

When choosing the option to register, the new user then be able to input the required credentials to register such as a username and a password into the fields on the view. Pressing register will then initiate a request to the server which will process the provided information. Once the server has processed the request by the client, the result will be feedback to the client's view (figure 4). The user will be able to see a message on screen about the success or failure of their attempt to register. Once successfully registered, the user can return them to previous main view and attempt to log in.


Login/Sign up

Enter your registration details:

Username: alpha1

Password: ••••••

Confirm password: ••••••

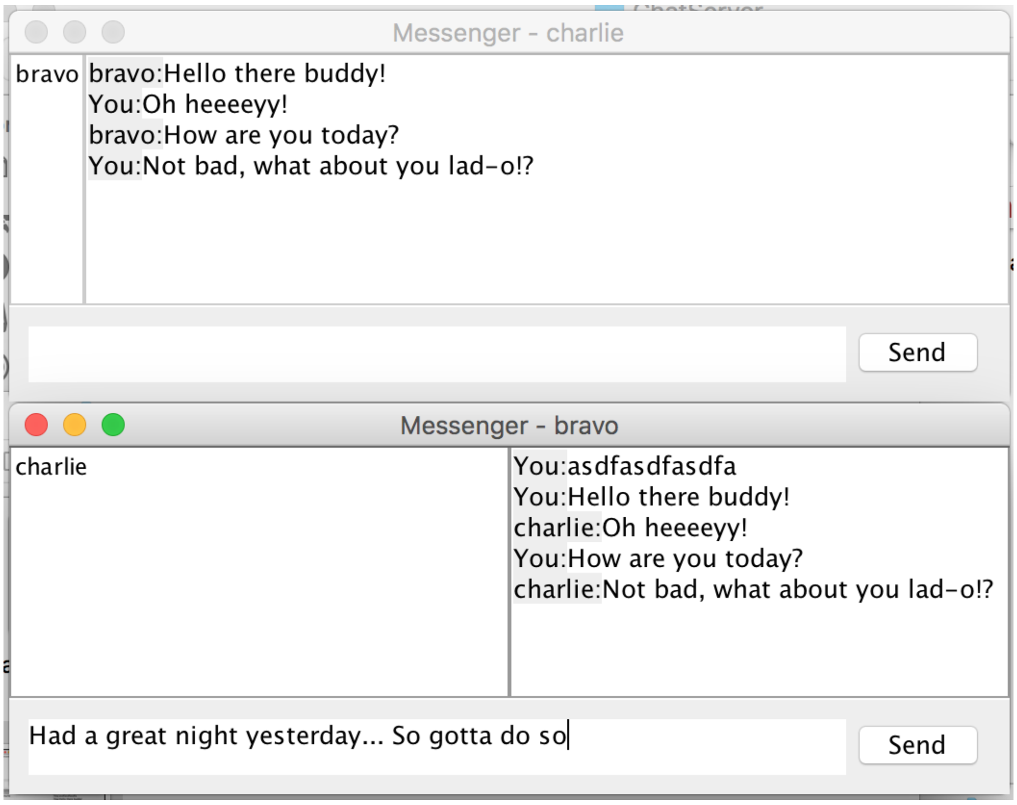
Name: a.lpha

Username Error

Your username should not be empty.
The length should be 5-10 characters long.
Your username should only contain letters!

Figure 5. View for new client registration.

Having successfully logged into the messenger service, the client is presented with another main view in which they can see all available online clients (figure 6). In this view the user can chat with all other online users or start a private conversation with one or more clients. If private conversation is initiated, the user will be presented with a new chat view (figure 7). This view will also be displayed to all other clients who are involved with this private chat room. As with both types of chat room, the client can input text to submit to the other clients via the server. The received messaged are displayed above, with information relating the message to a particular client and the time. The displayed messages can be scrolled through to view the history of the chat over time.


Messenger - charlie

bravo

```
bravo:Hello there buddy!
You:Oh heeeeyy!
bravo:How are you today?
You:Not bad, what about you lad-o!?
```

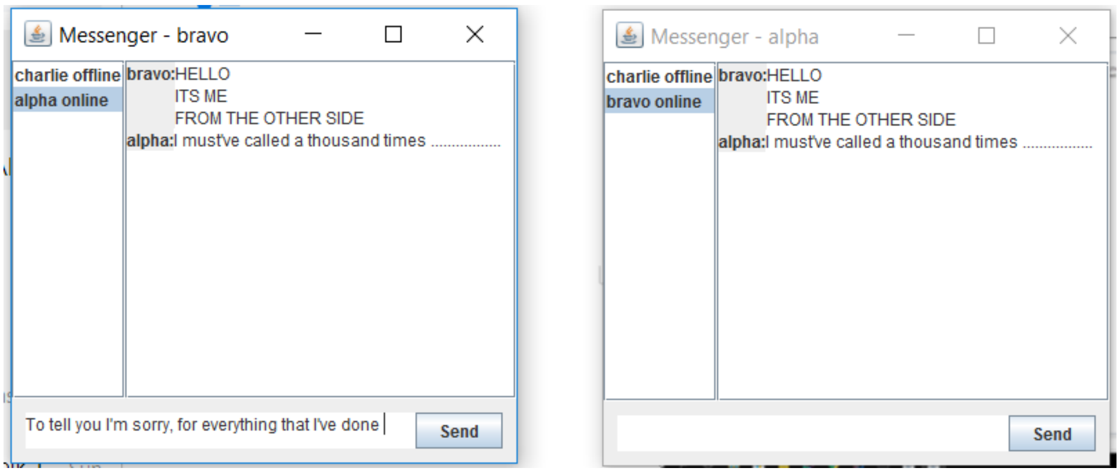
Messenger - bravo

charlie

```
You:asdfasdfasdfa
You:Hello there buddy!
charlie:Oh heeeeyy!
You:How are you today?
charlie:Not bad, what about you lad-o!?
```

Had a great night yesterday... So gotta do so

Figure 6. Main chat view once the client has logged in.


Messenger - bravo

charlie offline
alpha online

```
bravo:HELLO
      ITS ME
      FROM THE OTHER SIDE
alpha:I must've called a thousand times ................
```

To tell you I'm sorry, for everything that I've done

Messenger - alpha

charlie offline
bravo online

```
bravo:HELLO
      ITS ME
      FROM THE OTHER SIDE
alpha:I must've called a thousand times ................
```

Figure 7. private chat/group chat

Protocol

The protocol is designed to allow both efficient and effective communication between the server and clients connected to it. The protocol covers all the key communications the client and server could possibly want to make to each other. All communications between the clients and the database must occur via the server. The protocol must therefore be “understood” by both the client and the server.

The protocol enables the user to send a request to register to the user, which must then make the relevant SQL query to the database by utilising JDBC driver to connect to the database. Once this is achieved a client can their credentials for to login. This process sends another separate request to the server to check these credentials against the database. Which in turn, the server sends SQL query to the database to check for the existence of an entry with the entered credentials. Once the user has access to the messenger service, they can send and receive messages to and from other clients. The messages are sent to other clients via the server using a dedicated protocol.

Client Send/Response:	**Object type:**	**Server Send/Response:**	**Object type:**
“Register”, username, password	String[]	“Register”, true/false, message	String[]
“Login”, username, password	String[]	“Login”, true/false, message	String[]
“Message”, command, content	String[]		

Test Plan

Types of Testing

To test all aspects of the client – server application, several different methods were employed:

- GUI – Tested by running the application and using each function.
- Methods – Tested through Junit testing.
- Networking – Tested by creating several client machines to access the server.
- Database – Tested by querying the database.

Test Items

Items to be tested:

- Chat application running on different client devices
- Client registration
- Client login
- Sending and receiving messages
- Private or Group chats
- Chat history
- Database queries and entries

Features to be Tested

Feature:	**Description:**
Client registration	Create a new user to access the messenger service
Client log in	Log in as authenticated user
Client log out	Log out from the system
Send message	Send a message in public chat room
Group chat	Add selected online users to the group chat and send messages
Private chat	Send message to one other user
Appropriate feedback messages	Look for the right message appears when a given action is performed
Database	Create new user and send messages both privately and publicly
Chat history	View previously sent messages between clients
I/O	Press buttons and inputting text to carry our actions

Software Risks and Issues

There are several risks and issues which have been recognised, that could have an impact on the quality of the application:

- No database backup in place.
- Security of information being transferred and stored.
- Reliability of the client to host the server for sustained periods of time.
- Clients may lose connection to each other or the server and/or database.
- The server may lose connection to its clients and/or database.
- Difficulty testing some aspects such as GUI may lead to some unresolved issues.
- Unknown maximal limit of the number of clients capable of connecting at once.
- Clients may not know if they have been added to a private/group chat.

Tests

Test:	**Description:**	**Expected:**	**Actual:**
Close the client	Click the close button	Program closed	Program closed
Enter right address and port	Enter the same address and port as the server	Successfully establish connection to the server	Successfully establish connection to the server
Enter wrong address and port	Enter different address and port to the server	Error message displayed	Error message displayers
Create account with username that already exists	Click create account and enter existing username and password	Error message displayed	Error message displayed
Create account with unique username	Click create account and enter unique username and password	Message of successful account creation displayed	Message of successful account creation displayed
Enter invalid username	Enter a username of wrong length and with a number	Message of invalid username displayed	Message of invalid username displayed
Enter valid username	Enter a username of correct length and without a number	No error message displayed	No error message displayed
Enter invalid password	Enter a username of wrong length and/or without at least one letter and number	Message of invalid password displayed	Message of invalid password displayed
Enter valid password	Enter a username of right length and/or with at least one letter and number	No error message displayed	No error message displayed
Login with registered account	Enter username and password and click log in	Go to main chat view	Go to main chat view
Send message in public chat	Type message and click send	Sent message displayed in above panel	Sent message displayed in above panel
Create new private chat	Add new users to chat	New group chat view displayed with selected clients only	New group chat view displayed with selected clients only
View previous chat history	Scroll up on chat to see previous messages	Previous messages displayed	Previous messages displayed

Evaluation

User Feedback

Having let a group of users (peers) test the application, the general feedback received was generally positive. The application proved to serve multiple clients, without any major issues. Messages between multiple clients were sent and received reliably and efficiently. Users felt that the application was simple and easy to use, with a very clear layout. Registration was found to be simple to complete although, users felt that there should be more information displayed about restrictions before completing each section. This is something which can be fixed in a future iteration of the application.

Users also found creating private sessions, simple and intuitive, but felt that that public chat became somewhat redundant as a result. Additionally, only being able to send text and no other means of communication such as emoticons was flagged as a place for improvement. Given more time, this is a feature which could be integrated into the application. Users also suggested integration of small file transfers between users. This feature given the limited time and required understanding was not implemented in the final product. However, a file transfer feature was considered as a possibility should time permit.

Testing

Testing proved to provided invaluable information about the current state of the application. Although it was difficult to test all aspects of the application, many bugs and inconsistencies where detected through Junit testing. The issues with specific code were then addressed accordingly to improve the robustness of the overall system. Testing of GUIs was achieved by simply running the application and ensuring the desired actions occurred and that correct items were displayed. Although not as thorough a testing method as Junit testing, it still provided quality feedback about user interface's quality. To ensure the system could cope with multiple clients, several clients were set up and connected to the server. Here we learnt that the system was both sustainable and reliable, communicating with all clients and the database effectively. The system rarely if at all experienced networking issues of its own, which gave us confidence about its reliability to function without constant supervision and maintenance.

Self-Evaluation

Overall the project was successful in delivering a client – server application which served its basics functions reliably and consistently well. Each group member performed to their maximum capacity and ability to ensure that every aspect of the project was delivered to the highest quality. There was strong communication between individuals and sub – teams throughout the project. Consequently, issues were resolved relatively easily and efficiently, limiting impact on the progress of the project. Regular meetings ensured that the project was on kept on track and that all tasks were completed successfully. Given more time however, the consensus was that other small features could be implemented to enhance the overall completeness and functionality of the system.

Statement of contribution

I can confirm that I have made an equal contribution to this software workshop group project. I have produced contributed a good proportion of non – trivial Java code to the project. I have taken part fully in all aspects of the of the group – including attending meetings, planning, research, programming, report and presentation preparation.

Name:	Nabeel Anwar
Signature:	Nabeel Anwar
Name:	Ioana Avirvarei
Signature:	Ioana Avirvarei
Name:	Maurice Noel Bouniol
Signature:	Maurice Noel Bouniol
Name:	Roman Gaev
Signature:	Roman Gaev
Name:	Ali Oztas
Signature:	Ali Oztas