

TK2023

Object-Oriented Software Engineering

WhatsApp

Group Members:

- | | |
|---|---------|
| 1. Eileen Tong Hui Guan | A180693 |
| 2. Ahmad Aieman Akmal Bin Ahmad Sharul Khalil | A182997 |
| 3. Foo Jia Yung | P113856 |
| 4. Nur Khalisha binti Abdul Rahman | A179487 |

Table of Contents

CHAPTER 1	4
INTRODUCTION	4
1.1 INTRODUCTION	4
1.2 PROBLEM STATEMENT	5
1.3 PROPOSED SOLUTION	6
1.4 OBJECTIVES	6
1.5 SCOPE	6
1.6 CONSTRAINTS	7
1.7 METHODOLOGY	7
1.8 PROJECT SCHEDULE	8
CHAPTER 2	10
REQUIREMENTS SPECIFICATION	10
2.1 INTRODUCTION	10
2.2 FUNCTIONAL REQUIREMENTS	11
2.3 QUALITY REQUIREMENTS	15
2.4 CONSTRAINTS	16
2.5 DOMAIN MODEL	17
2.6 USE CASE	18
2.6.1 USE CASE DIAGRAM	19
2.6.2 USE CASE SPECIFICATIONS	21
2.6.3 SYSTEM SEQUENCE DIAGRAMS	37
3.1 INTRODUCTION	46
3.2 CLASS DIAGRAM	46
3.3 SAMPLE OUTPUT	48

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

WhatsApp is a cross-platform instant messaging application targeting all phones running on Android OS 4.1 and newer, all iPhones running on iOS 10 and newer, and all phones running on KaiOS 2.5.0 and newer. The application aims to provide free direct messaging and group messaging service between phones across different phone carriers, different phone operating systems, and across different countries and regions.

Current solutions to messaging between phones include Short Message Service (SMS) and Multimedia Messaging Service (MMS) provided by phone carriers, and exclusive instant messaging applications provided by phone manufacturers. Current solutions face limitations in terms of accessibility of the service across different phone carriers, different phone models and different regions, with messaging between phones of different carriers or platforms requiring partnerships and workarounds. Current solutions also differ in implementation for group messaging, resulting in difficulty of managing group messages. SMS and MMS in particular also faced the issue of inconsistent and costly pricing due to the nature of the service being tied to different phone carriers and one-way messaging systems.

WhatsApp would facilitate messaging between phones through internet services instead of mobile services. Users of WhatsApp would be able to send text, images, videos, audio messages, and files to other users provided they also installed WhatsApp on their phones. WhatsApp aims to be a messaging service that is richer than SMS, more reliable than MMS, has wider compatibility with more mobile devices and fewer boundaries for exchanging messages.

1.2 PROBLEM STATEMENT

1. Messaging capabilities between phones are limited by phone model, service region and the phone carriers the phones are subscribed to.

The main methods of sending messages between phones are through Short Message Service (SMS) and Multimedia Messaging Service (MMS), which are services provided by phone carriers. This has the following problems:

- The number of messages a phone can send to other phones is limited based on the phone's phone carrier plan.
- Sending a message to a phone subscribed to a different phone carrier often incurs an extra fee to the sender's phone carrier payment plan.
- Sending a message across countries requires a roaming agreement between the phone carriers, and for the sender to activate roaming service on their phones. If there is no roaming agreement between the phone carriers from respective countries, the message could not be sent. Similarly if the sender's phone does not have roaming active, the message could not be sent. Sending a message overseas incur extra fees to the sender's phone carrier payment plan.
- MMS which allows a user to send a text message greater than 160 characters in length, videos, images, and audio requires a phone that is compatible with the service. (Older phones will be unable to receive such messages).

There are Instant Messaging platforms where messaging is free of charge between phones (example: iMessage for iPhones), however it is limited to phones of the same brand.

2. Group chat capabilities are highly dependent on the phone model and phone carrier.

The most common form of group chats are sent over MMS, which has the following limitations:

- The phone must support MMS in order to participate in said group chat.
- The maximum number of recipients vary between phone model and phone carrier.

- Group MMS are not counted as single messages, but rather as repeated messages sent to different people, which can be costly towards the phone carrier payment plan.
- The phone may not process group MMS as intended, resulting in separate individual MMS threads for a single conversation, thus making the conversation hard to manage.

Similarly there are Instant Messaging platforms that have group chat features free of charge and with extra features (example: iMessage for iPhones), but they are limited to phones of the same brand.

1.3 PROPOSED SOLUTION

- Instant messaging service that works across platforms, across phone carriers (without excess fee), and across countries as long as they have the app installed.
- Minimize the fee of phone carrier subscription as it only needs mobile data to work.
- Instant messaging service that supports uniformed group chat functionality.
- Group chat feature for different users to communicate and interact with each other with different smartphones.

1.4 OBJECTIVES

The objectives of the project are as follows:

1. To implement a proper group chat that is not merely a system of simultaneous messages across multiple different users.
2. To reduce the fee of subscribed monthly plans of their phone carrier.
3. To provide inter-messaging accessibility across all phone models and carriers.

1.5 SCOPE

For this project, we shall only cover the essential functionalities of WhatsApp which are the following.

1. Log in and log out into an account,
2. Receive and send text messages to desired recipients,
3. Create a group chat and manage it, including the following: -
 - a. Adding/removing members,
 - b. Changing group name,

This is to reduce the complexity of this project and because of the limitation of our knowledge.

1.6 CONSTRAINTS

This iteration of WhatsApp is a prototype, hence most of the features shall merely be simulated, and no security measures shall be implemented. There shall be no real connectivity between different instances of WhatsApp clients. All data shall be saved locally, and will not be saved across sessions. This iteration of WhatsApp shall not have phone authentication to verify the login. Fake contacts shall be used.

1.7 METHODOLOGY

The task for this project is to choose one application that is available in the market and with at least 4 functionalities/features. As most of us are using the application Whatsapp in our everyday lives, its functional requirements are very well known and understood. Since we are going to recreate a new version of an existing application by reverse engineering, the most practical software process model to be used would be the Waterfall Model.

During the Requirements Definition phase, requirements engineering is carried out. Requirements are elicited from online document reviews about the main features of Whatsapp. During this process, UML conceptual models for instance domain model, use case, and system sequence diagram are created so that all stakeholders have the same understanding of the system to be developed. After requirements are elicited, requirements specification involving requirements documentation are done using a hybrid approach. Conceptual models approach and natural language approach are used in this process.

Design class diagram which represents a static view is constructed to represent software classes during the System and Software Design phase. This diagram is slightly different from that created during the requirement phase in terms of the classes involved. Additional attributes are identified during the design phase, and operations are included in the class diagram. Associations between classes are not required during this phase.

During Integration and System Testing phase, each functionality developed is then integrated into a complete application. Code verification and validation are carried out, and defects found are removed resulting in the application working as intended.

1.8 PROJECT SCHEDULE

[illegible]

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 INTRODUCTION

Whatsapp has been a major messaging application for people to use. It's easier for users, especially students for communication or study purposes. As an active user, it can be said that every single functionality in the application has been discovered and used. There's some quite similarity function with other messaging applications and a huge difference of function in it.

As for the functionalities itself we only focus on sending text, receiving text, creating groups and managing created groups. Users can send messages to another user for free as long as they are connected to the Internet. Users need to create an account and login to use the application. The message can be both sent whether in group chat or just simply in personal chat.

For the group chats to be functionally used, users need to add contact to a new group. So, users need to save contact details from another user first. The administrator of the group can make another user be an admin too in the same group. The admin speciality as they can remove and add members in the group chat. They also can change the group chat's name according to their likes.

2.2 FUNCTIONAL REQUIREMENTS

Table 1: Sending messages

	User Requirement
FR 1	The user should be able to send a message to another user and view it in the corresponding message log.
	Functional Requirement
FR 1.1	The system should be able to detect and select contact name/number
FR 1.2	The system should be able to receive message (input) from the user.
FR 1.3	The system should be able to save the sent message/file in the corresponding contact message log.

Table 2: Receiving messages

	User Requirement
FR 2	The user should be able to receive a message to another user and view it in the corresponding message log.
	Functional Requirement
FR 2.1	The system should be able receive a message from other users
FR 2.2	The system should be able to view a message from other users.
FR 2.3	The system should be able to save the received message in the corresponding contact message log.

Table 3: Create a group chat

	User Requirement
FR 3	The user should be able to create a group chat in the app
	Functional Requirement
FR 3.1	The system should able to detect and select contact name/number
FR 3.2	The system should able to add the selected contact names/numbers into a group
FR 3.3	The system should able to receive message from multiple user in the group
FR 3.4	The system should be able to display received messages from the members of the group to other group members.

Table 4: Manage a group chat

	User Requirement
FR 4	The user should be able to manage a group chat in the app by removing members and adding new members.
	Functional Requirement
FR 4.1	The system should able to detect and select contact name/number
FR 4.2	The system should be able to add the newly selected contact name/number into a group.
FR 4.3	The system should be able to remove the selected contact name/number out a group

2.3 QUALITY REQUIREMENTS

The table below are the proposed quality requirements for our application, WhatsApp.

Table 5: Quality Requirements

No.	Type	Requirement
QR1	Usability	The interface design of WhatsApp shall be similar to the interface design of pre-existing messaging services and applications.
QR2	Usability	A new user shall require no more than 1 minute to add a new contact to WhatsApp.
QR3	Usability	A new user shall require no more than 1 minute to start a new chat with an existing contact on WhatsApp.
QR4	Usability	A new user shall require no more than 8 minutes to start a new group chat with existing contacts on WhatsApp.
QR5	Performance	A WhatsApp message shall be displayed on the recipient's device within 500 milliseconds after the message was sent by the user assuming ideal internet connection.
QR6	Performance	Old WhatsApp messages should load within 10 seconds after the user scrolls up the message timeline to view old messages.
QR7	Space	The size of WhatsApp application itself shall be no more than 100MB.
QR8	Space	Length of text messages shall not exceed 65536 characters in UTF-8 encoding.
QR10	Availability	Users shall be able to send and receive messages on WhatsApp under internet speeds of at least 0.064 Mbps.
QR11	Reliability	Unexpected downtime of WhatsApp servers shall be no longer than 5 hours per week.
QR12	Reliability	Maintenance downtime of WhatsApp servers shall be no longer than 30 minutes per month.
QR13	Portability	WhatsApp shall be supported on all Android phones running on OS 4.1 and newer, all iPhones running on iOS 10 and newer, and all phones running on KaiOS 2.5.0 and newer.
QR14	Maintainability	Users will be able to use older versions of WhatsApp to communicate with other users for up to 6 months before they must update the application.

QR15	Maintainability	A crash report shall be generated and sent to the WhatsApp servers after 3 consecutive crashes occurring within an hour of usage.
QR16	Robustness	WhatsApp shall check for internet connectivity every 10 seconds upon loss of internet connection.
QR17	Robustness	WhatsApp shall attempt to resend a message every 10 seconds upon failure to send a message to the recipient.
QR18	Robustness	The frequency of crashes on WhatsApp shall not exceed 3 crashes within a day of usage.
QR19	Robustness	Probability of data corruption on application failure shall not exceed 1%.

2.4 CONSTRAINTS

Whatsapp users can upload different kinds of files but to do that the user must have access to the internet to use it for free. There's a limitation of the uploading or transferring file's size. It's due to its own term and condition of the application itself. This function may burden some users, such as if students need to upload bigger files.

This iteration of WhatsApp is a prototype, hence most of the features shall merely be simulated, and no security measures shall be implemented. There shall be no real connectivity between different instances of WhatsApp clients. All data shall be saved locally, and will not be saved across sessions. This iteration of WhatsApp shall not have phone authentication to verify the login. Fake contacts shall be used.

2.5 DOMAIN MODEL

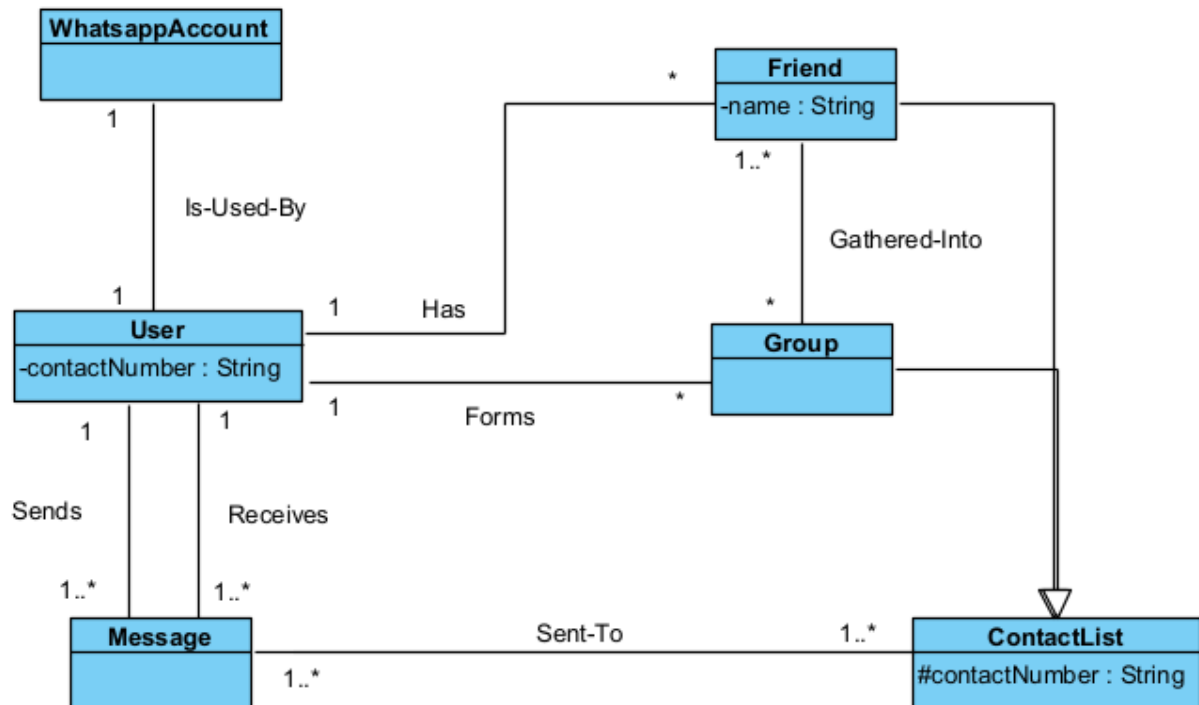


Figure 2.5 WhatsApp is an application which allows each user account to send and receive multiple messages from his/her contact list. The user is also able to form groups, add friends into groups, and send and receive multiple messages from his/her contact list.

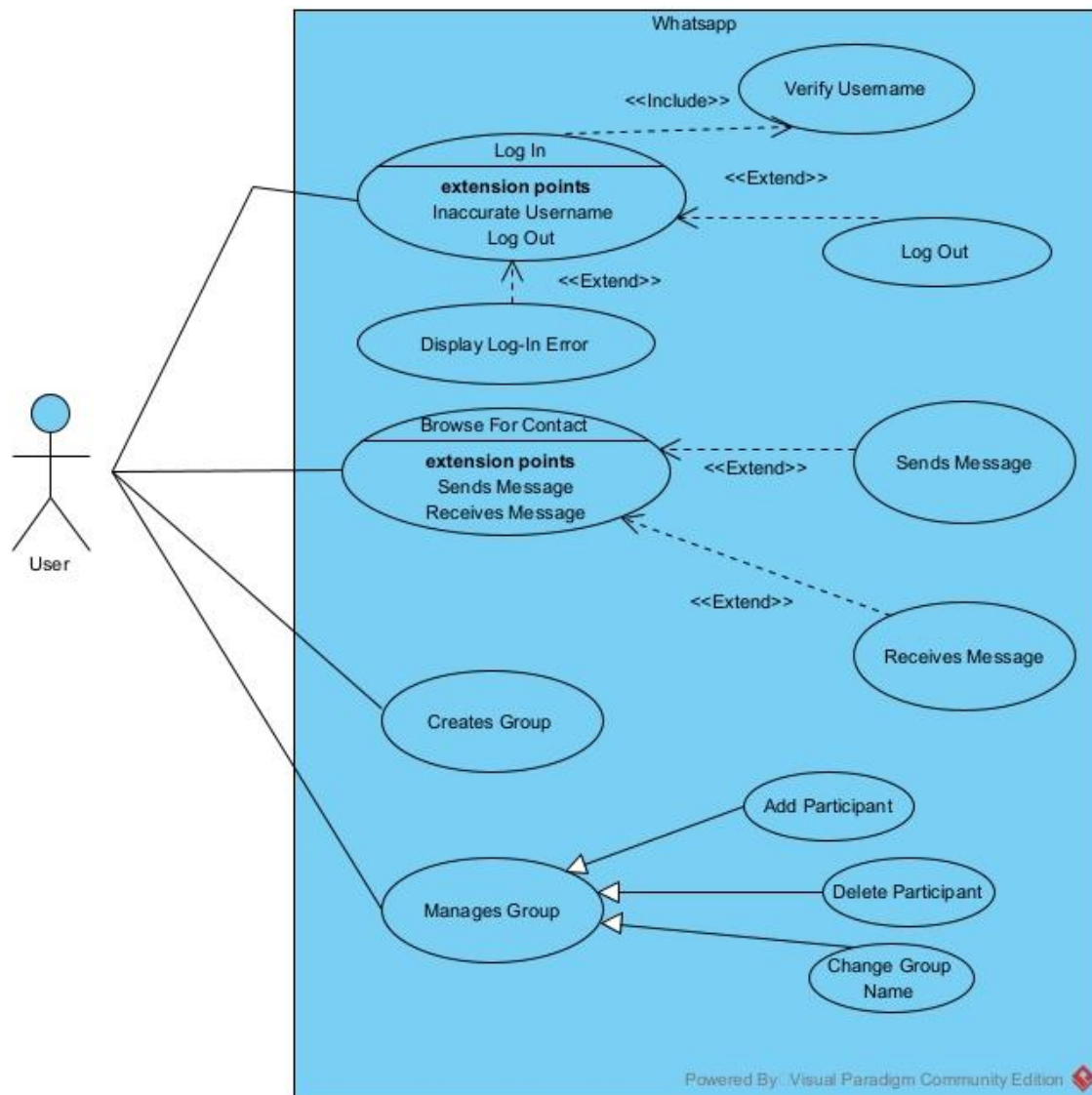
2.6 USE CASE

The table below lists the user goals for our WhatsApp system.

Table 6: User Goals

Primary actor	User goal
WhatsApp User	<ul style="list-style-type: none">- Create a group chat- Add contact to group chat- Remove contact from group chat- Change group chat name- Send a message to either a contact or group- View a message from either a contact or group

2.6.1 USE CASE DIAGRAM



[Figure 2.6.1] Users of Whatsapp shall be able to log in to the system, upon which the system will verify their username. If the login details are wrong, the system shall display a log-in error to the user. Users of Whatsapp shall be able to browse for a contact in order to send messages or read messages. Users can create groups and manage them by adding participants, removing participants, or changing the group name. Users shall be able to log out of the system.

2.6.2 USE CASE SPECIFICATIONS

ID:	US001	
Title:	Login to Whatsapp	
Description:	The user can log in to their respective account by entering the correct password and username.	
Primary Actor:	User	
Precondition:	The user already has register the account	
Postcondition:	The user is logged in to their account	
Main Success Scenario:	Actor	System
	1. The user enter username and password	2. The system verifies that the username exist and the password received is correct. 3. The system gives the user the access to login and display the main page.

Alternative Scenarios:	Actor	System
	1a. The user enters incorrect username/password or unregistered username.	2a. The system displays a login error.

ID:	US002	
Title:	Log out of Whatsapp	
Description:	The user can log out from their respective account.	
Primary Actor:	User	
Precondition:	The user already has logged in to their account	
Postcondition:	The user is logged out from their account	
Main Success Scenario:	Actor	System
	1. The user presses the log out button.	2. The system logs out the user from the system. 3. The system displays the log in page.

ID:	US003	
Title:	Browse for contact	
Description:	The user can search for contact from the contact list	
Primary Actor:	User	
Precondition:	The user already has logged in the account	
Postcondition:	The user selects a contact from the contact list	
Main Success Scenario:	Actor	System
	<ol style="list-style-type: none"> 1. The user selects the “contact” button. 3. The user selects a contact 	<ol style="list-style-type: none"> 2. The System display contact from the contact list 4. The system returns the selected contact to be used in an operation.

ID:	US004	
Title:	View message log of the selected contact	
Description:	The user views message log of the selected contact	
Primary Actor:	User	
Precondition:	The user already has logged in the account	
Postcondition:	The user views the message log of the selected contact.	
Main Success Scenario:	Actor	System
	<ol style="list-style-type: none"> 1. The user browses for contact. (See "Browse for Contact" specification) 2. The user selects a contact from the list of contacts. 	<ol style="list-style-type: none"> 3. The system displays the message log between the user and contact.

ID:	US005	
Title:	Send a message to the selected contact	
Description:	The user can send a message to the selected contact	
Primary Actor:	User	
Precondition:	The user already has logged in the account.	
Postcondition:	The user sends a message to the selected contact.	
Main Success Scenario:	Actor	System
	<p>1. The user views the message of the contact list. (See "View Message" specification)</p> <p>2. The user enters a text message to be sent to the contact.</p>	<p>3. The system successfully sends the message to the contact.</p> <p>4. The system updates the message log between the user and contact and displays it.</p>

Alternative Scenarios:	Actor	System
		<p data-bbox="995 309 1394 405">1a. The system fails to send the message to the contact.</p> <p data-bbox="995 456 1394 611">2a. The system displays "Message failed to be sent. Retrying."</p> <p data-bbox="995 663 1394 817">3a. The system tries to send the message to the contact again.</p>

ID:	US006	
Title:	Create new group chat	
Description:	The user can create a new group chat by specifying the name of the group chat and the friends that shall be members of the group chat.	
Primary Actor:	User	
Precondition:	The user already has logged in the account	
Postcondition:	The user creates a new group chat.	
Main Success Scenario:	Actor	System
	1. The user selects “Create group” button 3. The user enters the name of the new group. 5. The user selects the new group members from a list of contacts. (Refer to "Browse For Contact" specification)	2. The system asks for the name of the new group. 4. The system displays a list of friends.

	6. The user confirms the selection of the new group members.	7. The system creates the new group chat.
--	---	--

ID:	US007	
Title:	View message log of the selected group chat	
Description:	The user view message log of the selected group chat	
Primary Actor:	User	
Precondition:	The user already has logged in the account	
Postcondition:	The user views the message log of the selected group chat.	
Main Success Scenario:	Actor	System
	1. The user selects the group chat from a list of contacts.	2. The system displays the message log for the selected group chat.

ID:	US008	
Title:	Add new contacts to group chat	
Description:	The user adds new group chat members to an existing group chat.	
Primary Actor:	User	
Precondition:	The user already has logged in the account and is a member of the group chat.	
Postcondition:	The user adds new friends as members of the group chat.	
Main Success Scenario:	Actor	System
	1. User select "Group Details" 3. User selects "Add Group Member".	2. System displays the group name and the list of group members. 4. System displays a list of friends that are not yet group members.

	<p>4. The user selects the new group members from a list of contacts. (Refer to "Browse For Contact" specification)</p> <p>5. The user confirms the selection of the new group members.</p>	<p>6. The system adds new members to the group chat and updates the list of group members.</p>
--	---	---

ID:	US006	
Title:	Remove friends from group chat	
Description:	The user removes existing group chat members from the group chat.	
Primary Actor:	User	
Precondition:	The user already has logged in the account and is a member of the group chat	
Postcondition:	The members selected by the user are removed from the group chat.	
Main Success Scenario:	Actor	System
	1. User select "Group Details" 3. User selects a group member from the list of group members. 4. User selects the button "Remove Group Member"	2. System displays the group name and the list of group members. 5. The system removes the selected group member of

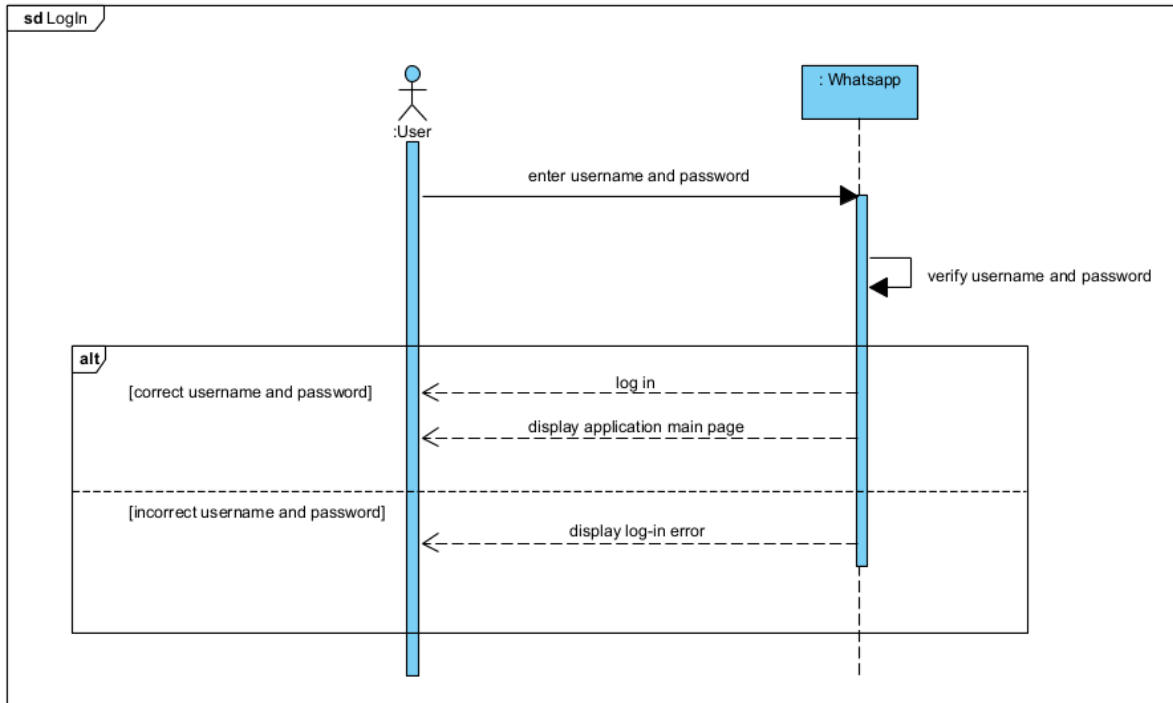
		the group chat and updates the list of group members.
--	--	--

ID:	US006	
Title:	Change name of group chat	
Description:	The user can change the group name of their group chat.	
Primary Actor:	User	
Precondition:	The user already has logged in the account and is a member of the group chat.	
Postcondition:	The name of the group chat is changed to what the user specified.	
Main Success Scenario:	Actor	System
	1. User select "Group Details" 3. User selects "Change Name".	2. System displays the group name and the list of group members. 4. System displays "Enter group name" and asks for a new group name.

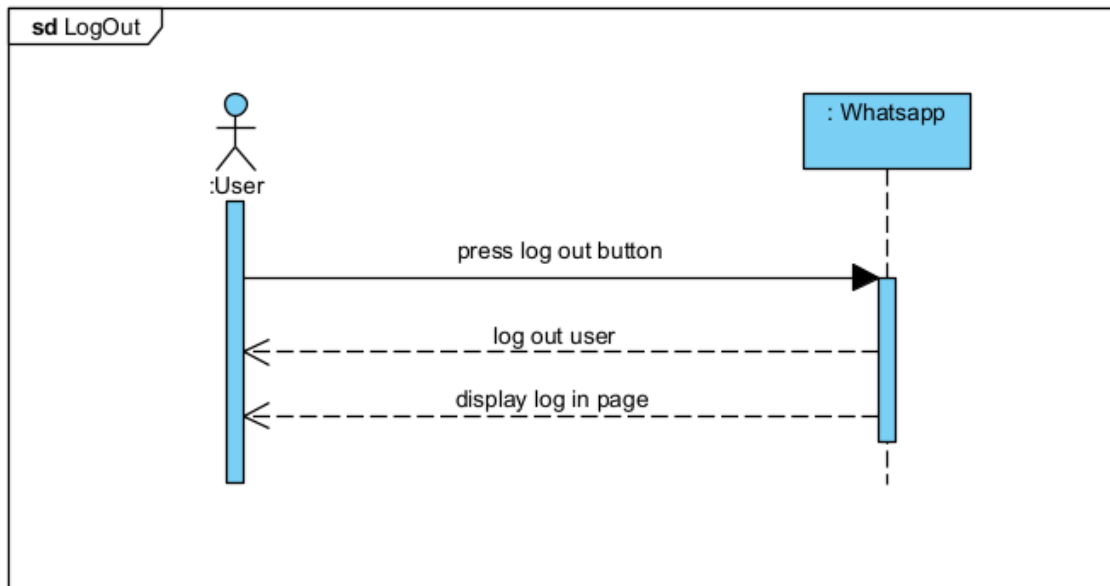
	4. The user enters a new name for the group.	5. The system updates the name of the group chat to the new group name.
--	---	--

2.6.3 SYSTEM SEQUENCE DIAGRAMS

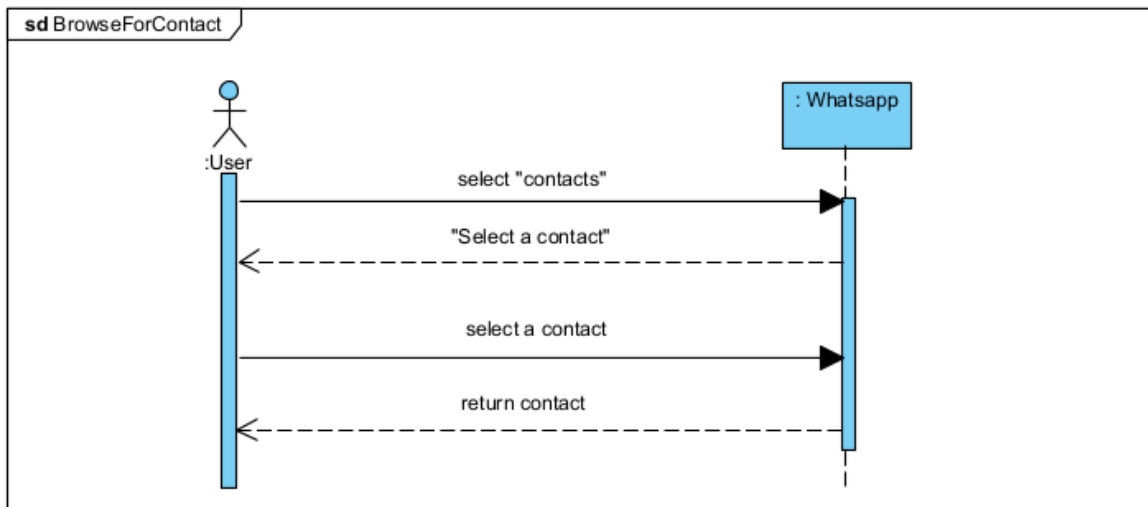
Log in



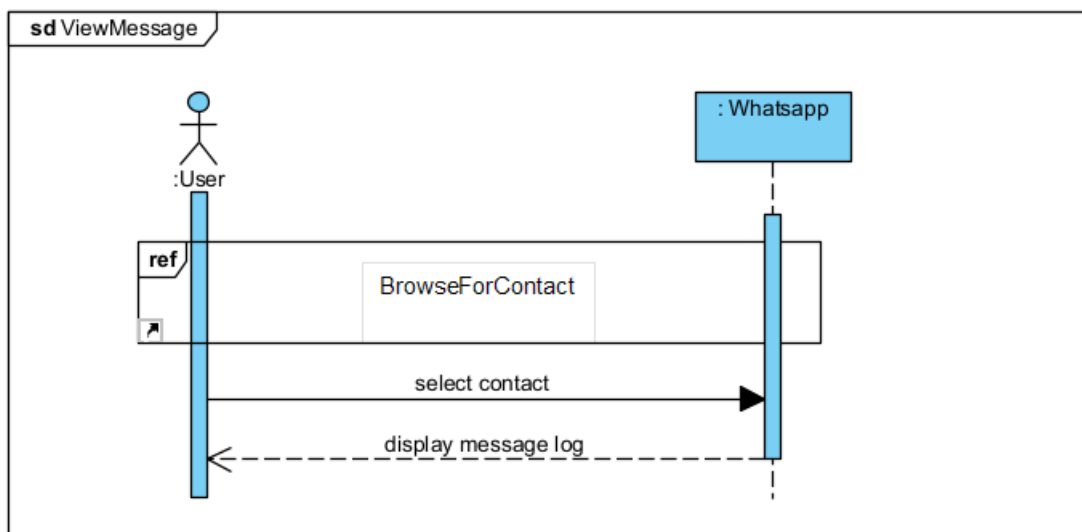
Log out



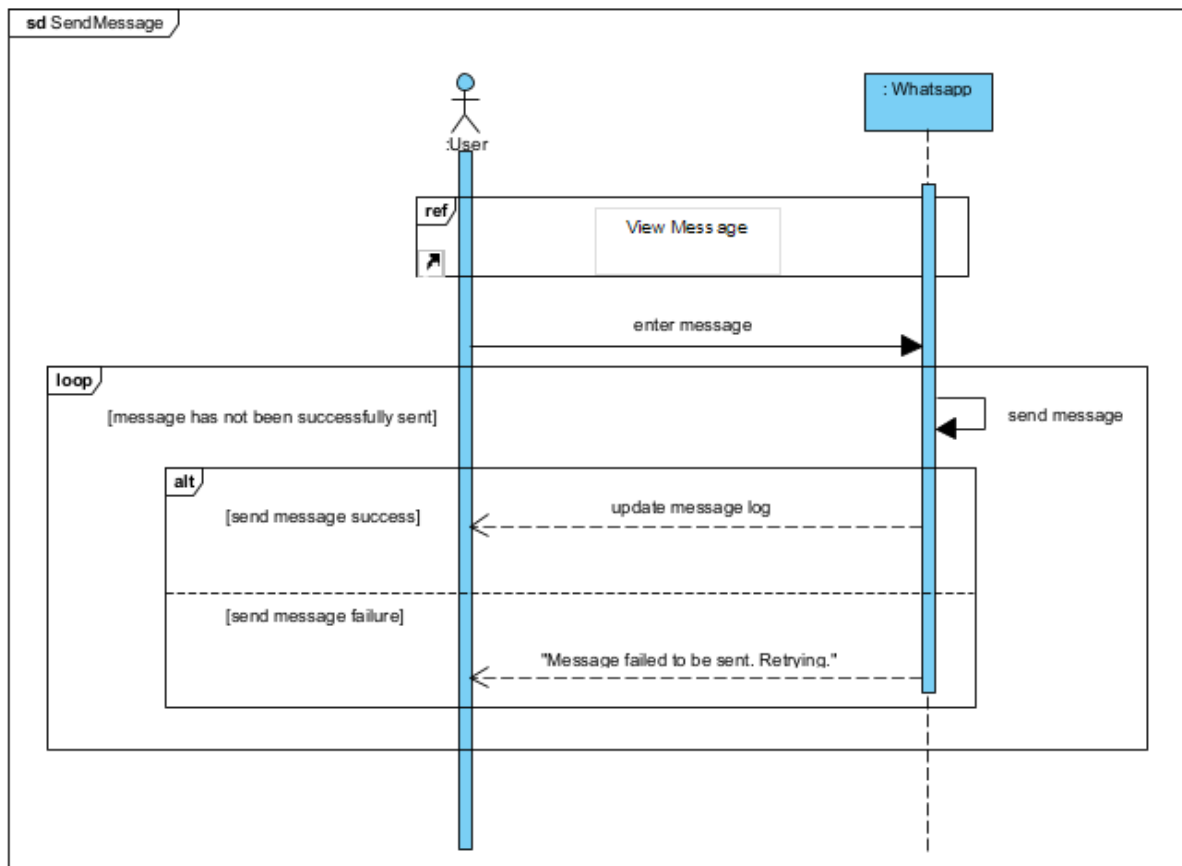
Browse for contact



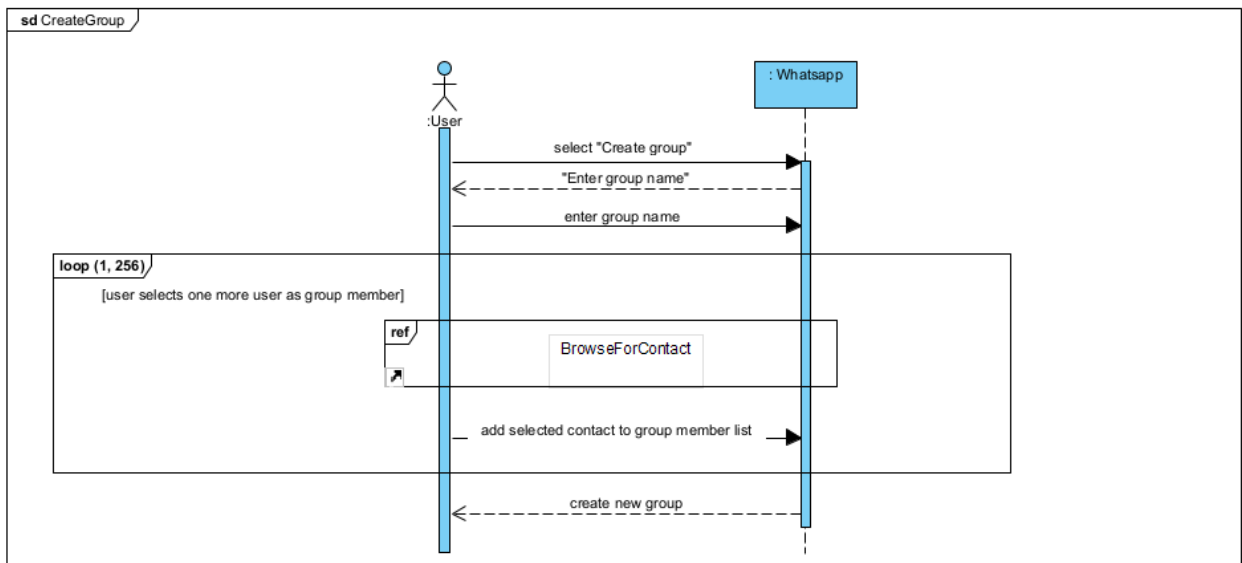
View Message



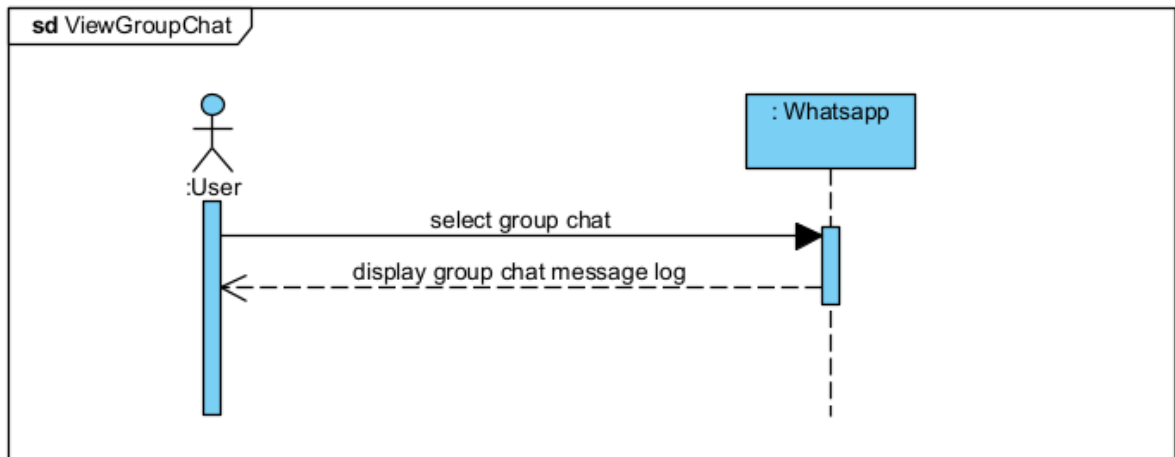
Send message



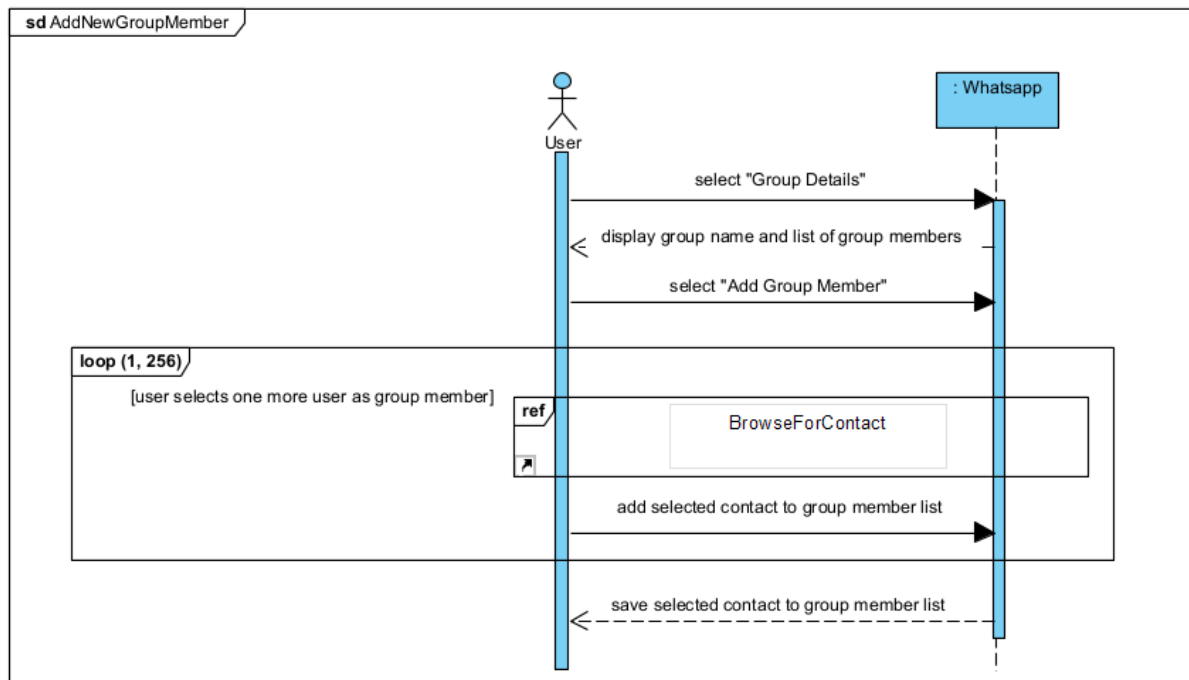
Create group



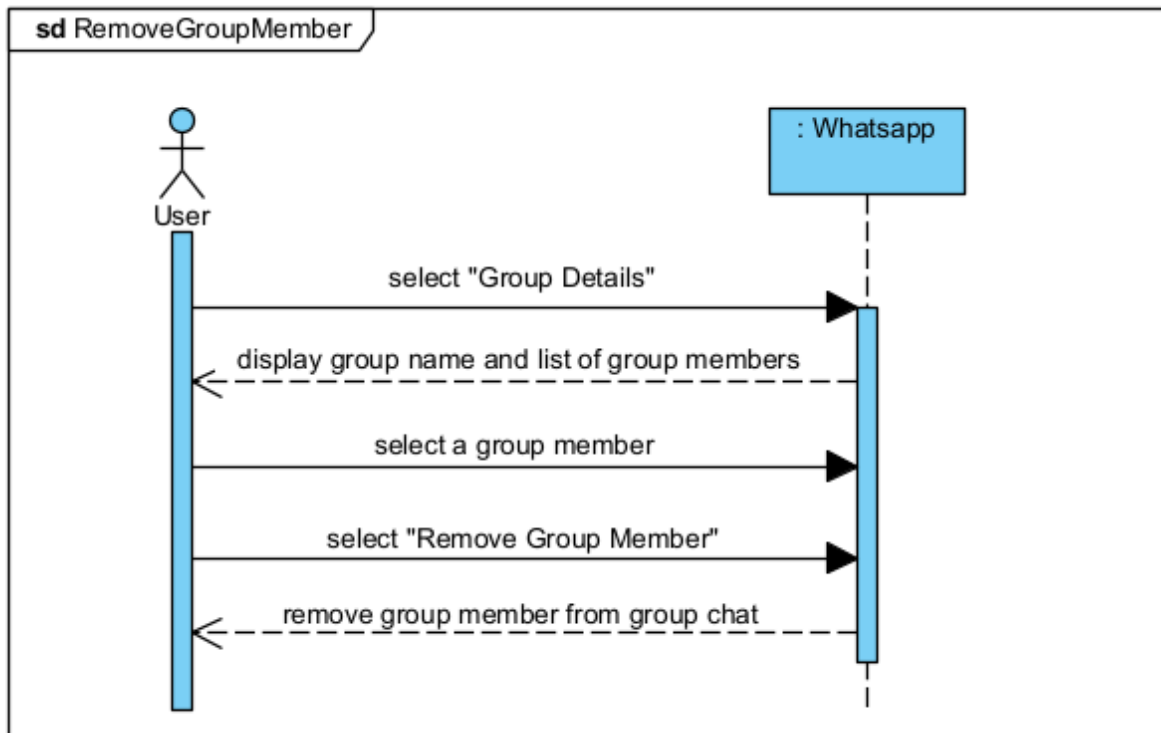
View Group Chat



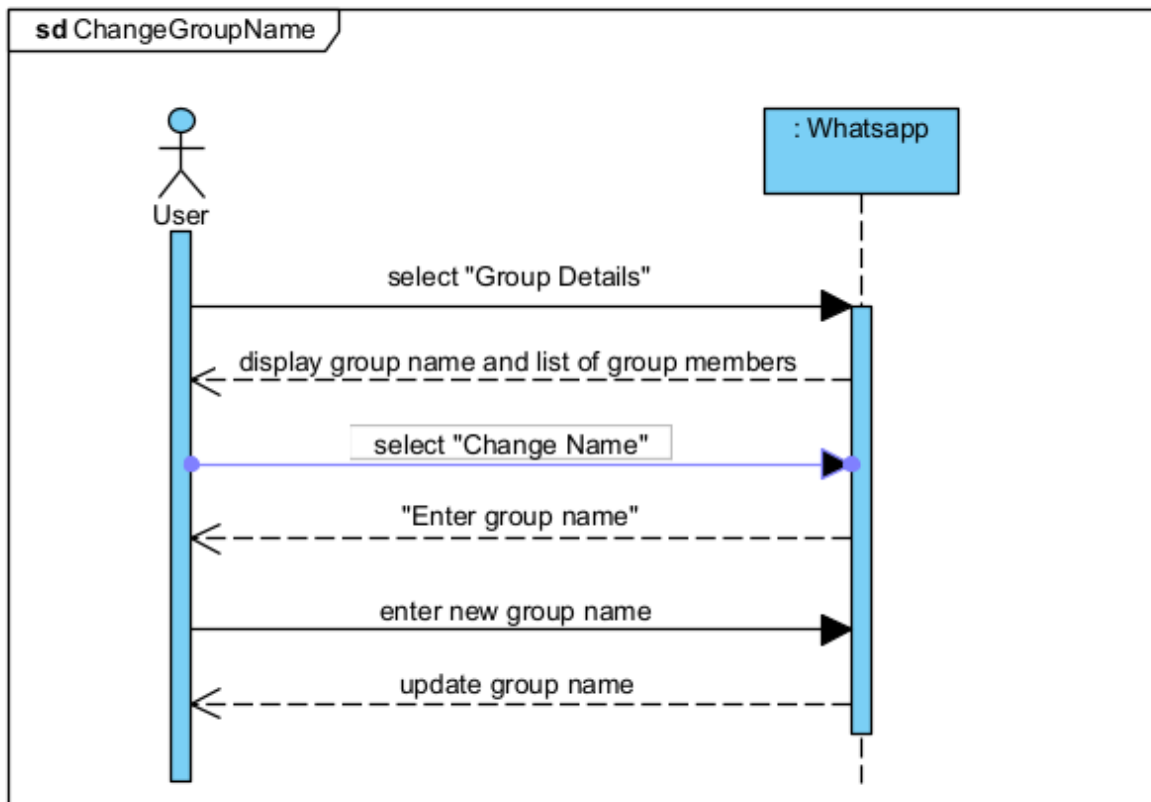
Add New Group Chat Member



Remove Group Chat Member



Change group name



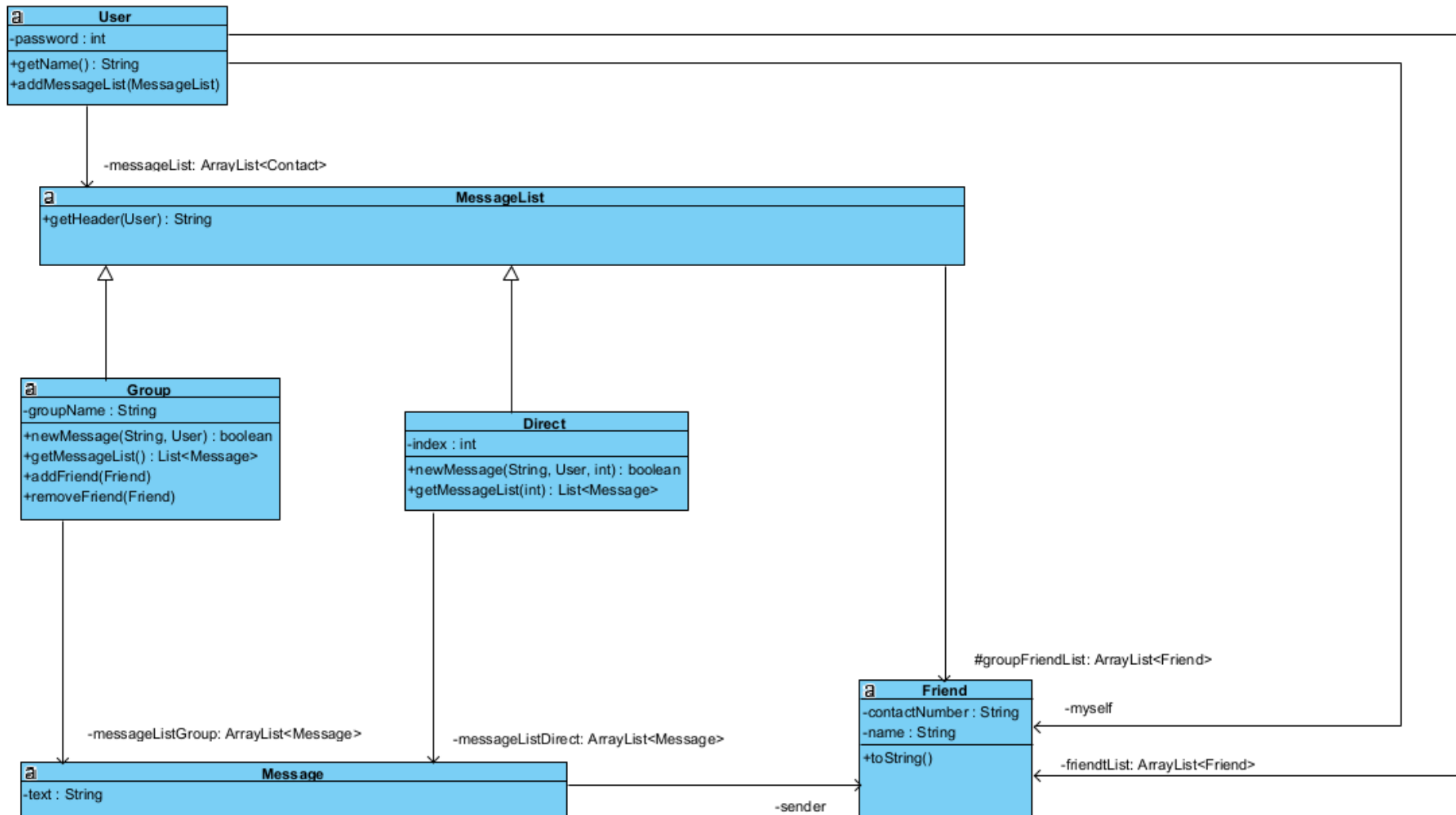
CHAPTER 3

DESIGN AND IMPLEMENTATION

3.1 INTRODUCTION

This section describes the design and implementation for the bare bones version of WhatsApp. The requirements, domain models, and system models outlined in chapter 2 are used as the basis for the program design. Class diagram is built based on the domain model. The program design uses the Model-View-Controller framework. The entire program is coded in Java with data stored in ArrayLists. The Java GUI implements Swing and Abstract Window Toolkit (AWT). The final program created is a prototype where a user can send messages to placeholder contacts, create groups, and manage the groups. Data is not saved upon session close.

3.2 CLASS DIAGRAM



User has an attribute myself (class Friend), list of Friends, and list of MessageList, and password. The attribute myself links to a Friend object which serves as the identity of the user in the system. The list of Friends are the friends the user has in the system (ie. contacts). Whereas the MessageList are the chats that the user has participated with in the system. The getName() operation retrieves the name of the user from their Friend object. The addMessageList() operation will save a new message list if the user joined a new chat, whether direct or group.

MessageList is the parent class of Group and Direct. MessageList has an attribute groupFriendList which stores the Friends that participated in the chat. MessageList has the operation getHeader() which is abstract in the parent class, and should return the header of the chat to be displayed in the app. Direct MessageList will return the recipient's name, whereas Group MessageList will return the group name.

A Group object has a group name, which can be retrieved from the getHeader() operation implemented in the Group class. It has messageListGroup which stores the messages in the group chat. It has a newMessage() operation which accepts a String and a User to save a new message sent by the user, while returning a boolean to indicate if the operation is a success or a failure. getMessageList() returns an immutable List of Messages to be displayed in the app. addFriend() accepts a Friend to be added into the group. removeFriend() removes an existing group member from the group by specifying the Friend to remove.

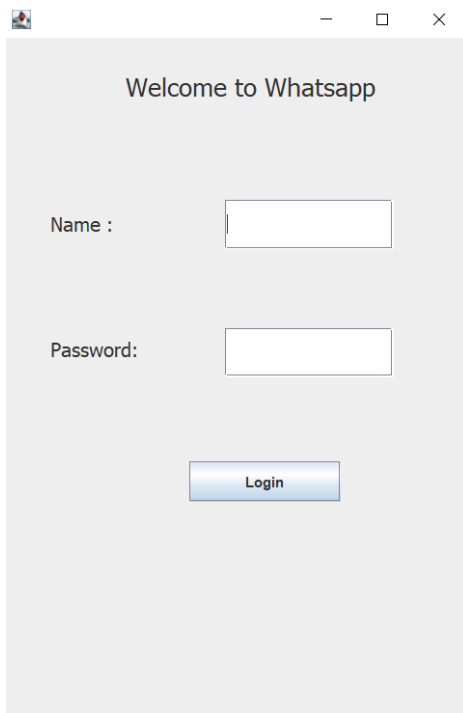
A Direct object has an index which is used to retrieve the chat log by the system. It has messageListDirect which stores the messages in the direct chat. It has a newMessage() operation which accepts a String, a User and an integer (index) to add a new message to the chat with the appropriate header. It has a getMessage() operation which accepts an integer (index) to retrieve a chat, while using the index to retrieve the header of the chat.

Message has attribute text and sender of object friend, which stores the text of the message and the sender of the message.

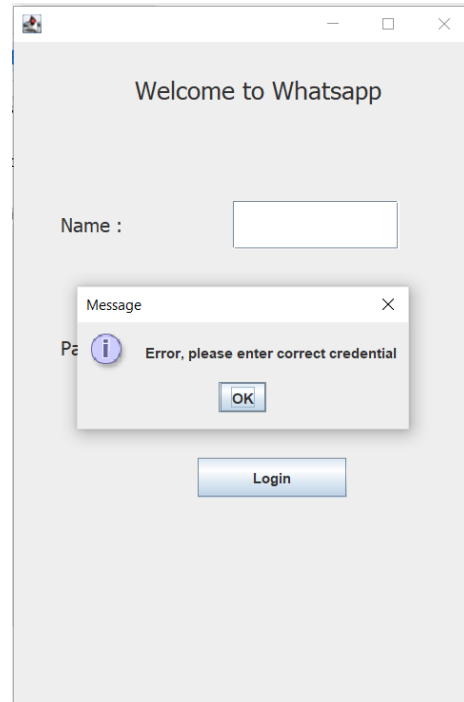
A Friend object has the contact number of a user and the name of a user, both stored in strings. It serves as the identity card for the users, and it is how users identify other users without exposing private details. The toString() method returns the name of the Friend.

3.3 SAMPLE OUTPUT

Login Screen



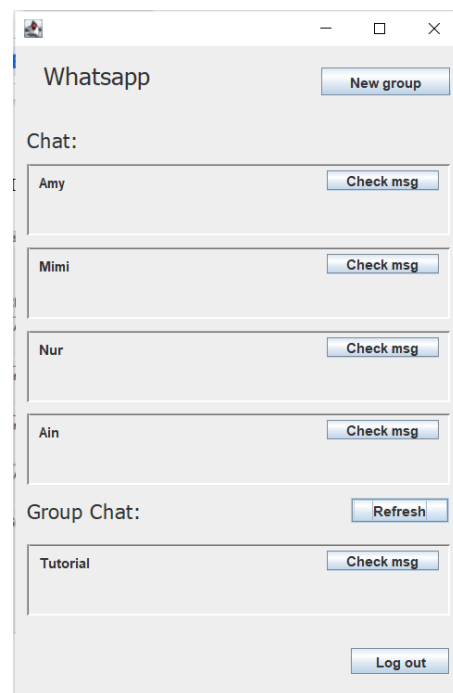
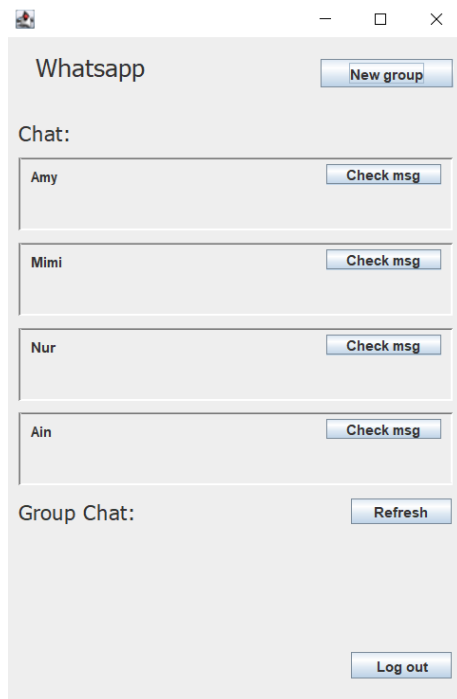
A screenshot of a login window titled "Welcome to Whatsapp". It features two input fields: "Name :" and "Password:". Below the password field is a blue "Login" button. The window has a standard Windows-style title bar with minimize, maximize, and close buttons.



A screenshot of the same login window, but with an error message displayed. A small dialog box titled "Message" is overlaid on the login screen. The dialog box contains an information icon (i) and the text "Error, please enter correct credential". Below the text is an "OK" button. The login screen itself is partially obscured by the dialog box.

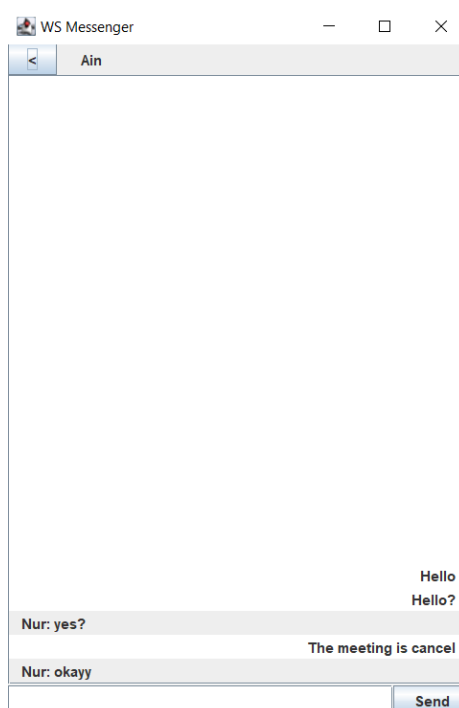
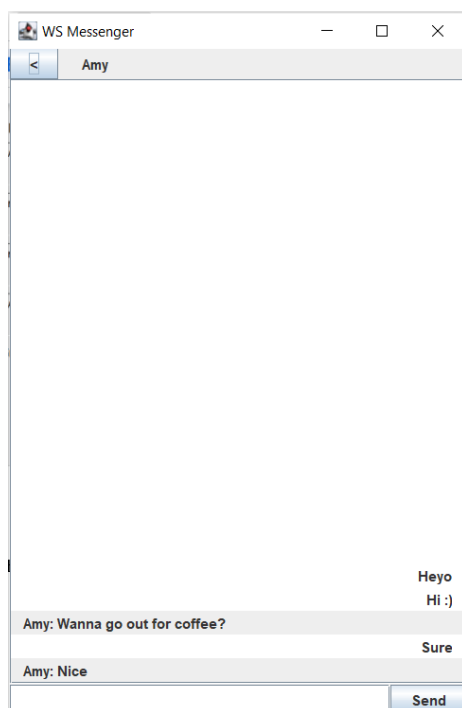
On the left side is the sample of our login page. The id and password set is “Kate” and “1234” respectively. On the right side is the sample output if the password or id given is incorrect.

Main menu

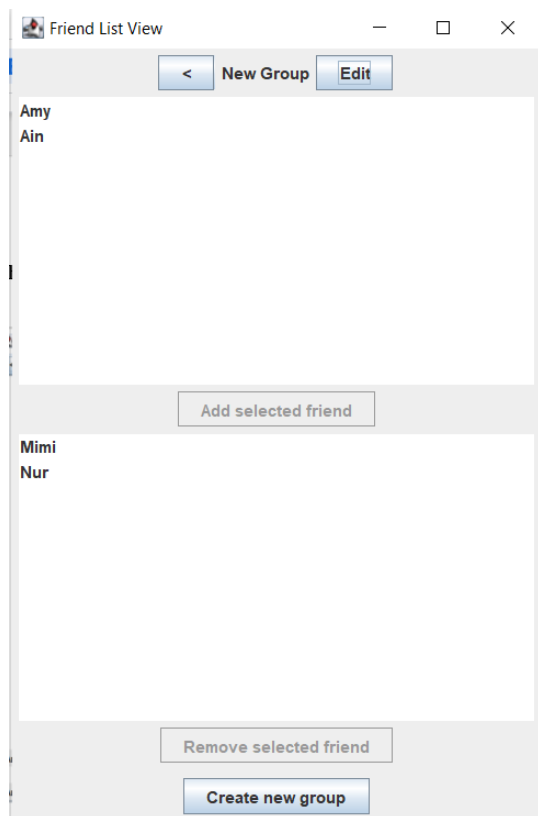


On the left side is the main menu for our output. On the right side is the same output with a group box. In this main menu, there is log out button to log out.

Chat log view



Creating new group



On the left side is the creating group output. On the right side is the message log for the created group.

Managing group (add/delete participant and rename group name)

