National College of Ireland
BSc in Computing – Cyber Security Stream
2017/2018

Marken Teder
X14470572
X14470572@student.ncirl.ie

LockTalk – Secure Messaging Application

Technical Report

Table of Contents

Executive Summary

The LockTalk messaging application is being developed to respond to some of the market's unanswered needs. The usage of messaging applications has sky-rocketed in recent years however, the focus on security has only been a recent development for the major applications. The LockTalk application answers those needs by implementing E2EE encryption to secure text messages as well as techniques for securing images and documents. However, the main reason for LockTalk is the control it will give to its users over the content they share. LockTalk will provide measures to prevent recipients from saving messages without authorization and allowing users to delete messages which may have been sent while intoxicated or are in breach of privacy unlike any other application in the market today. The challenges which LockTalk had to address was finding a balance between security and performance. The difficult challenge of encrypting and decrypting images was difficult to overcome however, a method of re-arranging the pixels using efficient algorithms was found to be a solution. The other technical approaches implemented for the application include using stored environments for individual conversations so that changes from both parties could be synced to the two devices. Using local storage combined with a MySQL database running on AWS servers the different types of data such as private and public keys, user data and archived conversations can be stored easily while maintaining the confidentiality, integrity and availability of the data. The requirements have been elicited through a perfect balance of user interaction and market research.

1 Introduction

1.1 Objectives

To design and develop a messaging application for Android. This application will be targeting anyone who uses alternatives to SMS when communicating with family, friends and others. The application will provide a crisp and modern user experience while maintaining simplicity and fluidity for users while using the application. The application will have an increased focus on security and control which will help in distinguishing itself from current competitors.

Users will be able to pair their phone to the messaging app using a mobile number authenticator. Once they have entered their mobile number they will receive a text with a code that they must enter to authenticate their device. Before their device is ready they will need to enter a unique username, which will be used to identify each user.

Users will be able to communicate with others through text, voice calls and video calls. They will be able to search for other users by their usernames and initiate contact by sending a “Pair Request”. Once in a conversation they will also be able to send images, documents and 30 second video and audio clips.

The application will be assigning each participant in a conversation with a unique key which will be exchanged during the “Pair Request”. This key will be used to encrypt all the text messages, documents and multimedia data such as images. This will provide a level of End-to-End Encryption that can ensure that users won’t have external entities accessing their messages.

The application will use built-in media viewer which will open the images/video. This involves decrypting the sent data and then displaying it on the receiver's device. This will stop people from downloading the images sent to them to their personal gallery. Users will have the option of allowing the message to be "Enable to Share" which will keep a version of the decrypted message archived that can be downloaded. The application also prohibits users from taking screenshots of the texts and images.

The application will be converting each conversation into an open environment where both participants can make changes to the chats which will be reflected on both devices. This will be limited to only altering the messages that they have sent themselves.

Users will be able to manipulate entire conversations by removing them and deleting them. When users remove a conversation, it will only disappear from the list of different conversations while the Pairing remains active between the two users. When a user deletes a conversation, the application will delete all their messages up to a month in the past and then removes the conversation. This action will break the Paired Link and a new one must be made if they require to communicate again.

To gain invaluable experience in areas which I haven't been able to fully expose myself during college. This application will require high-level Android development which will teach me the principles and practices to increase my skillset in multiple areas. I will also be able to learn and gain experience in back-end coding with PHP and creating a server to host a database. I will be able to utilize my background in the cybersecurity specialization to aid me while integrating encryption and secure programming practices. The application while coded in Java which I have studied in college will require me to utilize techniques, libraries and APIs which I have

never come across before which will provide invaluable experience in Java development.

1.2 Background

I began researching areas related to my specialization which was cybersecurity. I wanted to have the chance to really build my experience in the area and since I was already very enthusiastic with the field I knew that I would not lose motivation while development when I have a project which interests me. While researching the various areas I was drawn towards encryption due to how important it had become in the recent years and my intrigue in the battle between cryptologists and code-breakers. After I had found my area of focus I proceeded to thinking of ways to implement them.

I began by researching the trends in the use of mobile technology. I began by looking at the most used applications currently and saw a huge trend of social media applications and messaging apps. It was at this stage that the spark of creating a completely secure application for instant messaging ignited. According to collected data messaging applications accounted for 3 of the Top 5 most downloaded non-gaming applications in the 1Q ’17 with WhatsApp, Messenger and Snapchat leading that charge. It was this statistic which led me to begin developing a messaging app. Since the market had already been occupied by such huge players I knew that I had to find a way to improve on the current applications and offer novel ideas to the current apps.

Since I was focusing on my specialization of cybersecurity and especially encryption I knew that one of the key features of the application was going to be security. This ideology of security will be applied to both management of conversations, security of data and general practices of development throughout the whole lifecycle. However, I knew that I needed a unique angle for my project

which would differentiate it from the current giants. I began to think on my own experience using these apps and came upon the realization that the current apps provide a lack of control. Once messages are sent there seems to be a transfer of ownership of the words or images sent. It was the lack of this feature which really began bringing together my project. I had to begin thinking about the actual need for such an application.

Following recent controversies of government agencies such as the NSA gathering communication data from the public as was uncovered by Edward Snowden a whistle-blower of the agency itself. This brought a lot of attention and awareness to people regarding the security of their private thoughts and how it must be incorporated more into the devices and applications they use. Therefore, using secure encryption to ensure that any data being sent over in any form will remain only between the parties involved in the conversation. This also includes any data being stored regarding the users themselves, which must be kept secure with data leaks becoming an increasing trend. But it is the lack of control which has become a huge topic in the recent years. With news events such as private images shared between huge groups of people who weren't the intended target or similar smaller scale breaches of trust being so prevalent these days it has become a real need for people to feel secure sharing intimate thoughts or images to their peers without having the stress that comes along with it. While the notion of sending such private messages has been debated, there should be a certain freedom in being able to express yourself without fearing these messages becoming compromised and shared.

It was at this stage I began researching the competition in more depth to analyze whether the control feature was truly unique or not. I needed to see what the standard was currently regarding messaging applications. I began researching Messenger and WhatsApp both owned by Facebook and being the Top 2 messaging applications currently used. It was only until recently that both

applications began incorporating End-to-End Encryption in the applications to secure the conversations of its users. While WhatsApp employs the technique by default Messenger has only began implementing it through a new feature called Hidden Conversations. These applications to their credit have begun implementing quite standard encryption techniques which does provide a level of security however there are applications which specialize in the security level. Signal is an example of such an application coming from Open Whisper Systems who were approved by Snowden himself also employ E2E Encryption however they also refuse to collect any meta data from the messages such as the time-stamps or user data. We can see that the level of security currently provided by messaging apps has been quite solid, so we know the bar that we must maintain or increase. However, when looking for applications which provide control over the messages being sent it is quite a different picture. Only a handful of messaging applications provide a version of control. WickR is an example of such an application where users can set a self-destruct timer on any messages sent however, this can be quite limited as people may discover the need to remove messages later which is why the self-destruct method couldn't be implemented in a more general fashion. I had conducted a small survey from peers after my research to get a sense of how the public may react to such a feature and was greeted by a lot of positive feedback.

In a broad sense the method which will be used to ensure this level of security and control is by having the application treat each conversation as an open environment where both entities can remove the message they have ownership over and it would reflect on both devices. The messages will also be stored locally so that there wouldn't be an opportunity to access archives through database attacks. The messages will be encrypted with multiple methods and the conversations can only begin once both parties have "Paired" which is when the encryption keys will be exchanged to allow for decryption. The application won't allow for any data to be downloaded or screenshotted unless the sender explicitly allows for downloads.

1.3 Technical Approach

The project task is to create a mobile messaging application called LockTalk for Android devices. This will involve research into Java and Android development and will be comprised into the following areas:

- **Defining the functional and non-functional requirements for the application. Also defining the scope of the project and to develop preliminary wireframes for the planned UI.**
- **Researching the Java language for best practices when it comes to security and overall performance.**
- **Research the Java libraries and available technologies which will aid in the development of the application.**
- **In-depth research into the functions provided by Android Studio IDE for more efficient development.**
- **The application will require research into mobile number authentication methods and implementing the most suitable one.**
- **The application will require in-depth research into the most secure encryption methods currently available for both text and multimedia such as images and video. Implementation will be based on the most balanced method in terms of security and complexity.**
- **In-depth research into using encryption keys and algorithms to create End-to-End Encryption.**
- **The application will require video and audio capture for the functionality which will require research into the methods currently used followed by the implementation.**
- **The application will require any files to be opened within the app to prevent unauthorised sharing and prohibit the user from taking screen shots. This will require research into current methods used in Android followed by creating the necessary features within the application.**
- **The application will need to handle each conversation as a separate environment so that any changes made affects both users.**
- **The application will need to store the conversation data locally in the device to ensure that messages can't be compromised while on outside servers.**
- **The application will allow users to Pair with others which allows the individual keys to be exchanged.**
- **The application will require the creation of a database to store basic user data to allow accessibility. This will only involve the Phone Number and Username.**

1.4 Special Resources Required

The project will be developed in Android Studio for Android devices. There will also be a database which will need to be hosted on a server with PHPMyAdmin acting as the UI. Due to Android using SQLite 3 natively the database will also be based on MySQL so that the syntax will overlap as much as possible. The database will be hosted through the services provided by AWS such as Amazon RDS.

- **Android Devices (Samsung Galaxy S7 Edge & Samsung Galaxy S6)**
- **Android Studio for Windows using the latest stable version**
- **Amazon Web Services Free Account**
- **Google Developer Account**

2 System

2.1 Functional Requirements

ID: FR1

TITLE: Registering the User

DESCRIPTION: Once the user has downloaded the application they should be able to register an account for their device. The user must enter their mobile phone number and a username along with an email which is optional.

RATIONALE: For the user to begin the set-up process for creating an account for the application.

DEPENDENCIES: FR1

USE CASE: Use Case 1 <Create Account>

ID: FR2

TITLE: Confirming account registration

DESCRIPTION: Once the user has entered their mobile number the application must send out a six-digit confirmation code to the user which they will enter to authenticate that they are registering the account on their device.

RATIONALE: For the user to confirm and authenticate that the device they are currently using will be the primary device.

DEPENDENCIES: FR1, FR2

USE CASE: Use Case 2 <User Authentication>

ID: FR3

TITLE: Logging the User in

DESCRIPTION: Once the confirmation code has been entered and matches, the application will bring the user to their main hub. This step will be repeated each time the user opens the application excluding the registration process.

RATIONALE: For the user to be able to log in and begin using the application.

DEPENDENCIES: FR1, FR3

USE CASE: Use Case <Log In>

ID: FR4

TITLE: Editing User Profile

DESCRIPTION: A user must have the ability to change certain information on their profile such as adding a profile image and their first and last name. They must also be able to privatize certain information such as their email, mobile number and their name so that it wouldn't show up when someone views their profile.

RATIONALE: For the user to have the ability to personalize their profile.

DEPENDENCIES: FR4

USE CASE: Use Case 4 <Edit Profile>

ID: FR5

TITLE: Search for other users

DESCRIPTION: A user must have the ability to search for other users using the username they entered during registration. The user also has an ability to search for users based on mobile number if they exist in their contacts on the device. This search will be limited to users who have their mobile number set as public.

RATIONALE: For the user to have the ability to search for their peers to begin a conversation.

DEPENDENCIES: FR4

USE CASE: Use Case 5 <Search for Users>

ID: FR6

TITLE: Send Pair Request to user

DESCRIPTION: Once the user has found the account they were searching for they can send the other user a Pair Request. This must be accepted by the other party before any messages can be sent between the two.

RATIONALE: For the users to have the control over who can send messages to them. This request will also include the sharing of encryption keys which will be used in the E2EE encryption to secure conversations.

DEPENDENCIES: FR4, FR6

USE CASE: Use Case 6 <Pair with Users>

ID: FR7

TITLE: Sending text messages

DESCRIPTION: A user must be able to send text messages to people they have paired with through the application. This will also include emojis which will have to be handled correctly by the application. There will be a certain limit set to the size of the messages however this won't usually be enforced as the limit most likely won't be reached.

RATIONALE: For the user to be able to send messages to their peers. The limit is to make sure the encryption and decryption won't tax the application to affect the performance.

DEPENDENCIES: FR4, FR7

USE CASE: Use Case 7 <Send Text Message>

ID: FR8

TITLE: Securing text messages being sent

DESCRIPTION: For the system to secure each conversation the messages must be encrypted before they are sent and decrypted once they have arrived. This process will be handled by the E2EE encryption and the keys which were exchanged during the pairing.

RATIONALE: The conversations between users must be secure so that there is minimal opportunity for outside sources intercepting the messages. This will ensure privacy and security for the users.

DEPENDENCIES: FR8

USE CASE: Use Case 10 <Encrypt Messages>

ID: FR9

TITLE: User able to call

DESCRIPTION: The user must be able to call other users they have paired with. This will require an active internet connection.

RATIONALE: The user must also be able to call peers besides having the ability to text.

DEPENDENCIES: FR4, FR7

USE CASE: Use Case 13 <Voice Call>

ID: FR10

TITLE: Sending images in messages

DESCRIPTION: The user must have the ability to send images to their paired contacts. These will be either from the storage of their device or they have the option to take a picture which will be sent.

RATIONALE: The user should have the ability to send over images to their peers.

DEPENDENCIES: FR4, FR7

USE CASE: Use Case 8 <Sending Images>

ID: FR11

TITLE: Securing images sent

DESCRIPTION: The system will encrypt the data of the image and will only be decrypted if the recipient opens the image from inside the app. The application will by default won't let images to be downloaded or screenshotted from within the application.

RATIONALE: The users will be able to assume that the images they have sent will only stay with the intended target and not spread. This will ensure security and control of content.

DEPENDENCIES: FR11

USE CASE: Use Case 10 <Encrypt Messages>

ID: FR12

TITLE: Sending document attachments

DESCRIPTION: The user must be able to send attachments of documents on their device. This document will be limited by size to maintain performance and limit data usage.

RATIONALE: The user should have the ability to send over any documents to peers.

DEPENDENCIES: FR4, FR7

USE CASE: Use Case 9 <Sending Documents>

ID: FR13

TITLE: Securing documents being sent

DESCRIPTION: The system by default will be using encryption algorithms to encrypt the document and will only be accessible once the recipient has decrypted the message. The saving of documents will not be permitted for the recipient.

RATIONALE: The users should be secure sending important documents if necessary without the worry of it being intercepted ensuring security and control of content.

DEPENDENCIES: FR13

USE CASE: Use Case 10 <Encrypt Messages>

ID: FR14

TITLE: User setting message access levels

DESCRIPTION: The user must be able to control manually if they want a message with images or documents to be available for saving by choosing the option before sending. This will ensure the system to create a copy of the decrypted version on the recipient's local storage in a folder created for the application.

RATIONALE: This will give the user the control over which messages they send are sensitive over ones which aren't.

DEPENDENCIES: FR11, FR13

USE CASE: Use Case 11 <Altering Message Access Level>

ID: FR15

TITLE: Storing a history of conversation

DESCRIPTION: The system will keep a history of the conversation for up to a month in the past. This will be the limit so that the conversation environments won't be using too large amounts of data.

RATIONALE: The users should be able to read back messages that have been sent in the past.

DEPENDENCIES: FR8

USE CASE: Use Case 12 <Store Conversation State>

ID: FR16

TITLE: Users deleting messages

DESCRIPTION: The users must be able to delete messages they have sent in all available formats such as text and media/documents. This will be limited to only messages they have sent in the conversation.

RATIONALE: This will give the user the ability to retract images or texts which they don't wish to be seen anymore. This will ensure control of content.

DEPENDENCIES: FR8, FR16

USE CASE: Use Case 14 <Delete Message>, Use Case 15 <Authenticate Sender>

ID: FR17

TITLE: Updating the conversation

DESCRIPTION: Once a user has deleted a message from the conversation the application will update the environment and send it to the recipient with the updated content.

RATIONALE: This will ensure that if a message is deleted by one party it will reflect on both users view of the conversation.

DEPENDENCIES: FR16, FR17

USE CASE: Use Case 16 <Update Conversation State>

ID: FR18

TITLE: Archiving a Conversation

DESCRIPTION: The user must have the option to hide conversations from the main hub of the application. This will only remove the conversation from the rest but keep the conversation archived. This conversation can be accessed again if retrieved or if the other users sends a message.

RATIONALE: The user must have the ability to control their conversation list to keep it organized. This may mean archiving conversations which have ended.

DEPENDENCIES: FR7, FR16

USE CASE: Use Case 17 <Archive Conversation>

ID: FR19

TITLE: Deleting a conversation

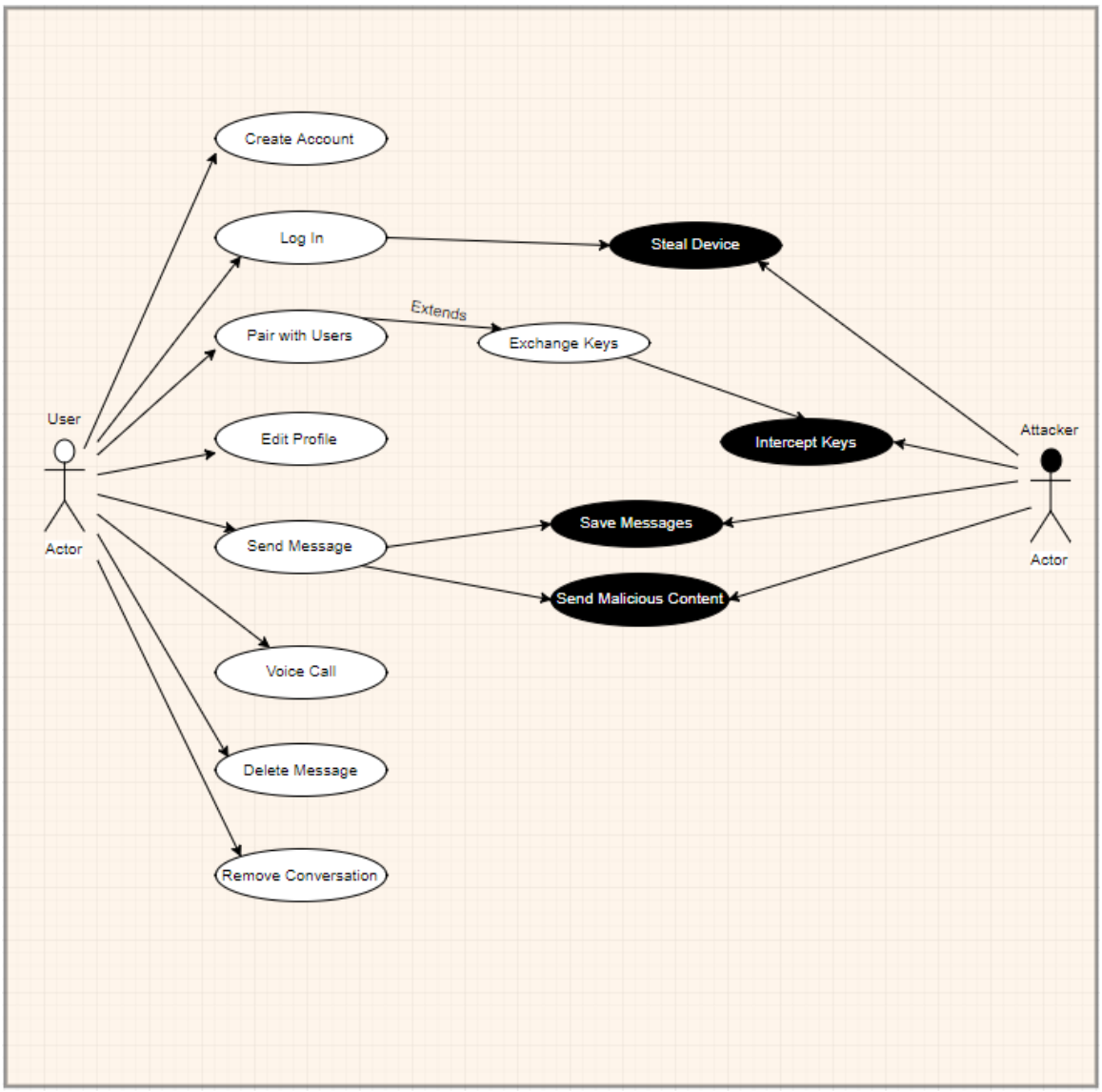
DESCRIPTION: The user must have the ability to complete delete a conversation they have had with other users. Once a conversation is deleted the application must delete all messages which the user had sent to the other party and remove it from the user's application completely. The conversation will be then updated on the other user's side to contain no messages from the deleting user.

RATIONALE: This will give the user the ability to be in complete control if they want to remove their communications completely with all history of their messages also removed.

DEPENDENCIES: FR7, FR16, FR18

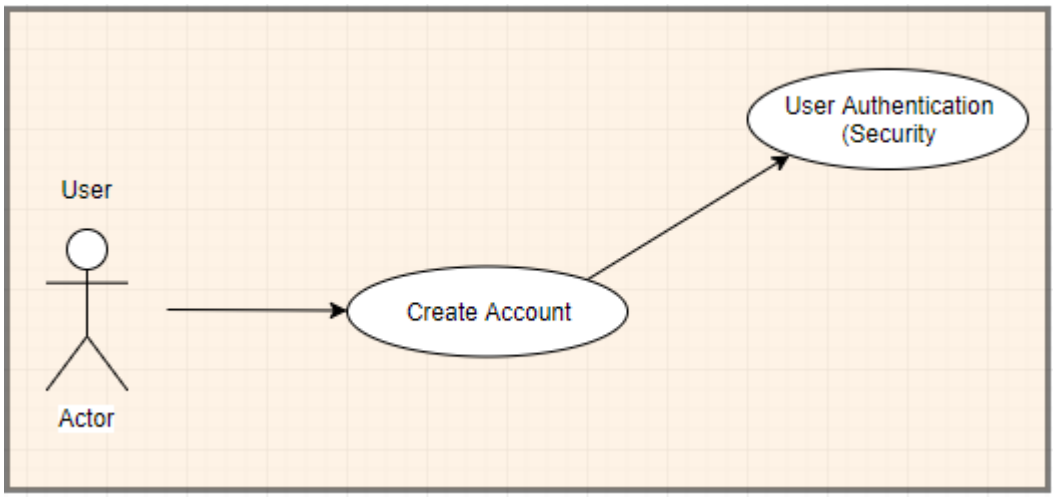
USE CASE: Use Case 18 <Delete Conversation>

2.2 Use Case Diagram



2.3 Use Case Descriptions

2.3.1 Use Case 1 <Create Account>



2.3.1.1 Brief Description

This use case describes how the LockTalk user creates their account on the application

2.3.1.2 Actors

1. User

2.3.1.3 Preconditions

- The device being used has an active connection to the internet.
- The device has successfully installed the application from App Store.

2.3.1.4 Basic Flow of Events

1. The use case begins when the user opens the application for the first time.
2. The application will display an option to create an account which the user will select.
3. The application will now display the user with certain fields which need to be filled. These include a username, mobile number and an optional field for email.
4. Once the user has filled the fields they will proceed to select the confirm account button.
5. The user will be prompted to enter a six-digit confirmation.
6. Use Case: User Authentication is performed.
7. The application will complete the creation of the account and the user will be directed to the main hub of the application.
8. The use case ends successfully.

2.3.1.5 Alternate Flow

2.3.1.5.1 Already existing Username

If in step 4 of the basic flow the user has input a username which already exists, then

1) The user will be prompted to enter another username until the username is unique.
2) The use case will then resume at step 5.

2.3.1.5.2 Invalid Mobile Number

If in step 4 of the basic flow the user has input an invalid mobile number, then

1) The user will be prompted to enter their correct mobile number.
2) The use case will then resume at step 5.

2.3.1.5.3 User exits application

If at any point prior to step 7 in the basic flow the user exits the application, then

1. The application will disregard all input data up to that point.
2. The use case terminates.

2.3.1.6 Key Scenarios

1. No Active Internet Connection

2.3.1.7 Post-conditions

2.3.1.7.1 Successful Completion

The user has authenticated their device and created an account successfully. They will be directed to the main hub of the application.

2.3.1.7.2 Failure Condition

The application has disregarded all input data and will close.

2.3.2 Use Case 2 <User Authentication>

2.3.2.1 Brief Description

This use case describes the process of authenticating the user who is creating an account.

2.3.2.2 Actors

1) User

2.3.2.3 Preconditions

1) The user has an active Internet connection.
2) The user has provided a valid mobile number during the registration.
3) The user has provided a unique username.

2.3.2.4 Basic Flow of Events

1) The use case begins when the user has input their details and has pressed the ‘Confirm Account’ button.
2) The application will receive the user’s mobile number.
3) The application will send a message containing an OTP to the user’s device through SMS.
4) The user will receive a message containing a six-digit confirmation code.
5) The user will copy the confirmation code and return to the application.
6) The user will input the code into the required field and push the ‘Confirm’ button.
7) The application will compare the input code to the stored confirmation code.
8) The application confirms that the codes match.
9) The use case ends successfully.

2.3.2.5 Alternate Flow

2.3.2.5.1 Invalid Confirmation Code

In step 6 of the basic flow if the user inputs the wrong confirmation code, then

1) The application will compare the input code to the confirmation code.
2) The application will return a false result.
3) The user will be notified that the confirmation code is incorrect.
4) The application will accept 3 subsequent attempts.
5) The application will request the user if they would like a new code sent.
6) The use case will fail.

2.3.2.6 Post-Conditions

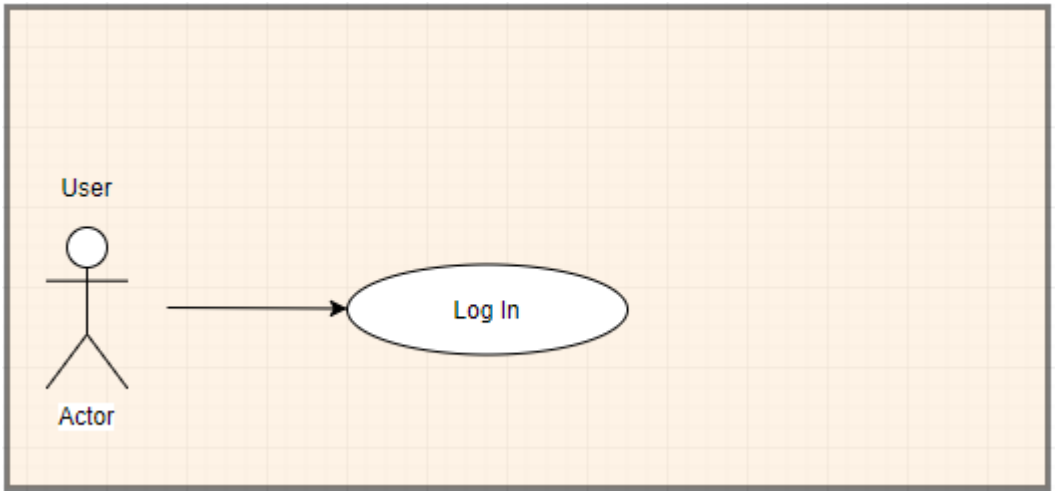
2.3.2.6.1 Successful Completion

The user has input the correct confirmation code and will be logged in to the main hub of the application.

2.3.2.6.2 Failure Condition

The user failed to enter the correct confirmation code and will be asked to request a new code.

2.3.3 Use Case 3 <Log In>



2.3.3.1 Brief Description

This use case describes how the application will log the user into their account.

2.3.3.2 Actors

1. User

2.3.3.3 Preconditions

- The user has an active Internet connection on their device.
- The user has completed the account creation process prior.
- The application has stored log-in feature.

2.3.3.4 Basic Flow of Events

1. The use case begins when the user opens the application from their phone.
2. The application will perform a log-in sequence where all the conversations will be loaded up.
3. The user will be brought to their main hub.
4. The use case ends successfully.

2.3.3.5 Alternative Flows

2.3.3.5.1 Opening Minimized Application

If the user has already had the application opened before and they have minimized it, then

1. The application will open normally without the log-in sequence.

2.3.3.5.2 Loss of Connection

If the user loses their connection to the internet during the log-in process, then

1. The application will state that no connection is detected.
2. The application will terminate.

2.3.3.6 Key Scenarios

1. No Active Internet Connection

2.3.3.7 Post-conditions

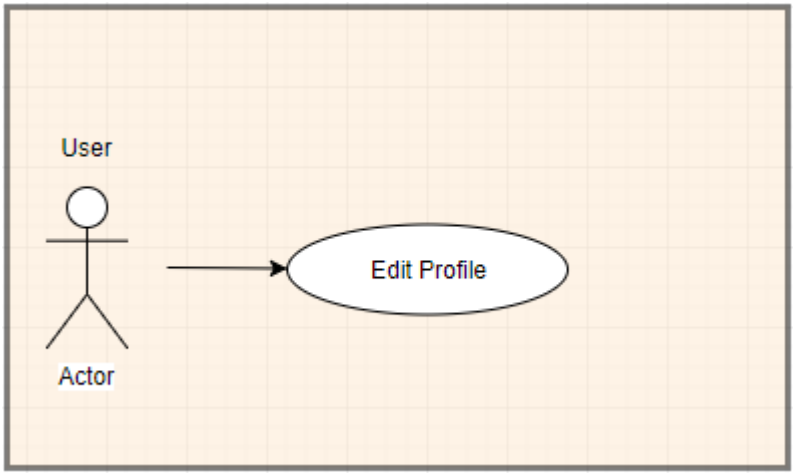
2.3.3.7.1 Successful Completion

The user has been logged in to their account and have access to the functionality of the application.

2.3.3.7.2 Failure Condition

The application will close and will require to be opened again.

2.3.4 Use Case 4 <Edit Profile>



2.3.4.1 Brief Description

This use case describes how the LockTalk user can edit the information on their profile to personalize their account.

2.3.4.2 Actors

1. User

2.3.4.3 Preconditions

- There is an active Internet connection on the device.
- The user has previously created an account and has logged in.

2.3.4.4 Basic Flow of Events

1. The use case begins when the user is in the main hub.
2. The user presses the navigation button on the screen and chooses the ‘Profile’ section.
3. The application will display the current information regarding the user including a section for a profile image.
4. These sections will have an option to edit by pressing the change option.
5. The fields which the user can edit are the email, first/last name and the image section.
6. Once the user selects one of the fields they will be prompted to input their changed version and save.
7. The user will then be able to click on the + icon to add a profile image for their account.
8. Once the user has pressed the + icon they will be able to choose from an image saved on device or to take a picture.
9. The user can press the Padlock symbol besides certain fields to privatize the information from other users.
10. The Padlock will turn blue if it has been activated or remain red if it is public.
11. The user can save the changes by pressing the Save button at the bottom of the screen.
12. The application will return the user to the main hub.
13. The use case ends successfully.

2.3.4.5 Alternative Flows

2.3.4.5.1 Choosing Image from Gallery

If in step 7 of the basic flow the user has chosen to select an image from their device, then

1. The user will be brought to their Gallery on their phone.
2. The user can navigate between different folders in their Gallery.
3. Once the user selects an image they will be directed to crop the image to fit the required dimensions.
4. The user will use a re-sizeable square to select the dimensions.
5. Once the user is satisfied they will press the ‘Select Image’ button.
6. The user will be brought back to the Profile page with the image as the profile image.
7. The use case resumes at step 8.

2.3.4.5.2 Taking a picture

If in step 7 of the basic flow the user has chosen to take a picture which will be used, then

1. The application will direct the user to their camera.
2. The user can then switch between the front-facing and back-facing camera.
3. Once the user presses the button to take a picture it will direct them to the cropping section.
4. The user will use a re-sizeable square to select the dimensions for the image.
5. Once the user has cropped the image they can press the ‘Select Image’ button.
6. The user will be brought back to the Profile page with the image now being used as the profile image.

The use case resumes at step 8.

2.3.4.6 Post-conditions

2.3.4.6.1 Successful Completion

The user has successfully updated their profile and has been directed back to main hub.

2.3.4.6.2 Failure Condition

The system failed to update the profile and notifies the user with error message.

2.3.5 Use Case 5 <Search for Users>

2.3.5.1 Brief Description

This use case describes how a user can search for other users on the application.

2.3.5.2 Actors

1) User

2.3.5.3 Preconditions

1) The user has an active Internet connection.
2) The user has previously created an account.
3) There are other users who have created accounts.

2.3.5.4 Basic Flow of Events

1. The use case begins when the user is on the main hub of the application.

2. The user pushes the navigation button on the screen and selects the ‘Connect’ section.
3. The user will be directed to a search function where they will be required to insert the username of the user they are searching for.
4. The user will be shown the first 20 entries available with the username they have entered. This will include usernames which append the input to other usernames.
5. The user will click on the profile of the username they searched.
6. The system will display the information which the other user has kept public on their profile.
7. The use case ends successfully.

2.3.5.5 Alternate Flow

2.3.5.5.1 Username does not exist.

In step 3 of the basic flow if the user has inserted a username which does not exist, then

1. The system will display message that username does not exist.
2. The system will prompt the user to change the search input.

The use case resumes at step 4.

2.3.5.6 Post-Conditions

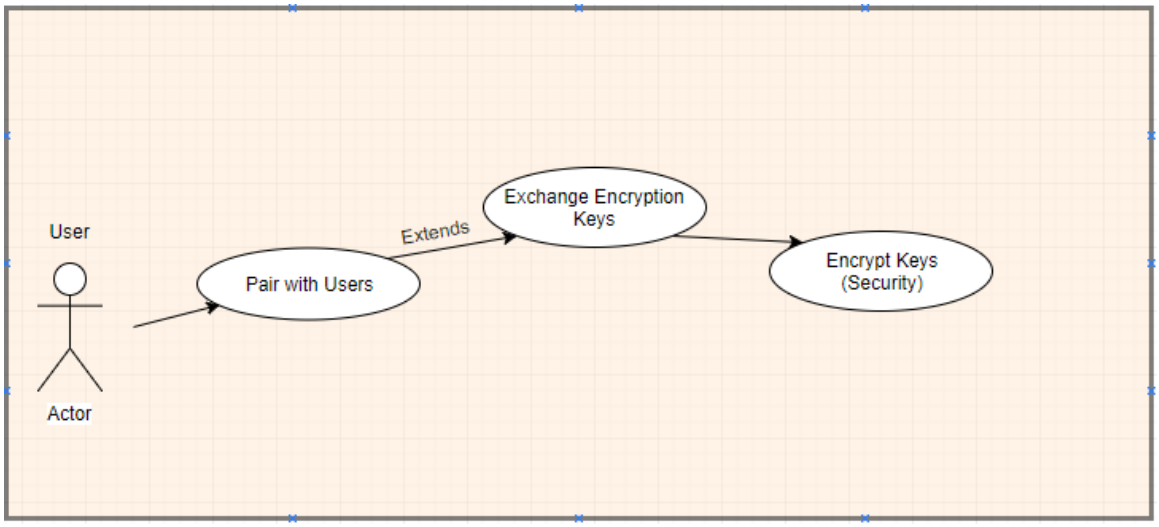
2.3.5.6.1 Successful Completion

The user has successfully searched for other users on the application, and viewed their profile.

2.3.5.6.2 Failure Condition

The application was unable to find a user who matched the username input by the user and displayed an appropriate message.

2.3.6 Use Case 6 <Pair with Users>



2.3.6.1 Brief Description

This use case describes how a user can request to pair with other users to begin communication.

2.3.6.2 Actors

1. User

2.3.6.3 Preconditions

1. The user has an active Internet connection.
2. The user has created an account prior.
3. There are other users who have also created accounts.
4. The user has searched for another user and clicked on their profile.

2.3.6.4 Basic Flow of Events

1. The use case begins when the user has visited the profile of another user.
2. The user can proceed to push the Connect button to send a Pair Request to the recipient.
3. The user will have to wait until the recipient has accepted the Pair Request to message them.
4. The user will be able to browse the Pair Requests sent to their account by navigating to ‘Connections’ section.
5. The application will display the usernames of other users who have sent a Pair Request to the user.
6. The user can click on the usernames to view the profile of the request sender and choose Accept or Decline.
7. The application will send a notification to the user when a connection has been completed.
8. Use Case: Exchange Keys is performed.

9. The user can navigate back to the main hub to begin a conversation with the other party.
10. The use case ends successfully.

2.3.6.5 Alternative Flows

2.3.6.5.1 User loses Internet connection

In step 6 of the basic flow if the user has lost Internet connection while pressing the Connect button, then

1. The system will attempt to re-send the Pair Request in 3 seconds.
2. If the Pair Request fails to send, the application will notify the user to reconnect to the Internet.
3. The use case will fail.
4. The use case will resume from step 4.

2.3.6.5.2 Accepting a Pair Request

In step 11 of the basic flow if the user chooses to Accept the request sent by another user, then

1. The application will send a notification to the user that a connection has been made.
2. The application will add the connected user to the conversation list on the main hub.
3. The use case will resume from step 14.

2.3.6.5.3 Declining a Pair Request

In step 11 of the basic flow if the user chooses to Decline the request sent by another user, then

1. The application will ask for a confirmation whether the user wishes to decline.
2. The user will press on the Yes option.
3. The application will remove the username from the ‘Connections’ page.
4. The use case resumes from step 11.

2.3.6.6 Post-conditions

2.3.6.6.1 Successful Completion

The user has successfully paired with another user and they can begin communicating from the main hub.

2.3.6.6.2 Failure Condition

The user will be notified if they have lost Internet connection to the application and be asked to reconnect.

2.3.7 Use Case 7 <Exchange Keys>

2.3.7.1 Brief Description

This use case describes the process of exchanging encryption keys from the users during a pairing.

2.3.7.2 Actors

1) User

2.3.7.3 Preconditions

1) The user has an active Internet connection.
2) The user has initiated a Pair Request with another User

2.3.7.4 Basic Flow of Events

1) The use case begins when the user has pushed on the 'Connect' button on the profile of another user.
2) Use Case: Encrypt Keys will be executed.
3) The application will receive the encrypted key from the user and send it to the server.
4) The server will receive the encrypted key from the other user.
5) The keys will be temporarily stored in a queue until they are sent to their recipients.
6) The user will receive the encrypted key and use the public key stored on the server for the other user to decrypt it.
7) The application will store the other user's encryption key into local storage.
8) The end case will end successfully.

2.3.7.5 Alternate Flow

2.3.7.5.1 Use Case <Exchange Keys> fails

In step 13 of the basic flow if the system fails to exchange the encryption keys, then

1. The application will re-attempt to transfer the encryption keys within 10 seconds of the failure.
2. The application will have a maximum of 3 attempts to re-send the encryption keys.

3. The system will notify the user that the Connection has failed.
4. The user can attempt to send another Pair Request.
5. The Use Case: Exchange Keys will retry until the connection has been successful.

2.3.7.6 Post-Conditions

2.3.7.6.1 Successful Completion

The application has successfully exchanged the encryption keys from both users.

2.3.7.6.2 Failure Condition

The application has failed during the exchange and will return an error message to the user with instructions to attempt the process again.

2.3.8 Use Case 7 <Encrypt Keys>

2.3.8.1 Brief Description

This use case describes the process of adding a second level of encryption to the user's encryption key during the exchange. This use case is to combat the Abuse Case <Intercept Keys>.

2.3.8.2 Actors

1) User

2.3.8.3 Preconditions

1) The user has an active Internet connection.
2) The user has initiated a Pair Request with another User
3) The application is in the process of exchanging the encryption key stored on the device.

2.3.8.4 Basic Flow of Events

1) The use case begins when the application has initiated the Use Case <Exchange Keys>.
2) The application will receive the encryption key from the local storage of the user's device.
3) The application will encrypt the user key using a private key known only to the application.
4) The encrypted key will then be prepared to be sent to the server.
5) The use case ends successfully.

2.3.8.5 Alternate Flow

2.3.8.5.1 Use Case <Encrypt Key> fails

In step 3 of the basic flow if the system fails to encrypt the user's key, then

1. The application will immediately halt the exchange process.
2. The user will be notified of a problem with the exchange.
3. The user will be prompted to attempt the process of exchanging keys.
4. The use case will terminate with a failure.

2.3.8.6 Post-Conditions

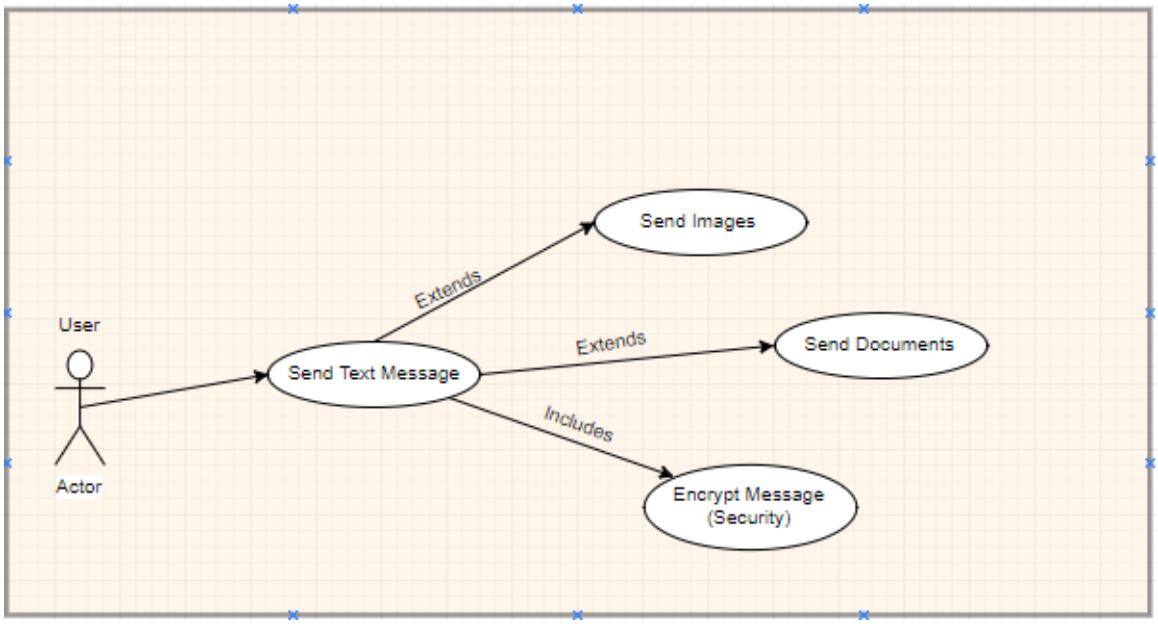
2.3.8.6.1 Successful Completion

The application has successfully added a second level of encryption to the user's key and prepared it to be sent to the server.

2.3.8.6.2 Failure Condition

The application has failed to encrypt the device key and will immediately terminate the use case before the key is sent to the server. The application will notify the user that a problem has occurred and requests that the Pair Request must be sent again.

2.3.9 Use Case 7 <Send Text Message>



2.3.9.1 Brief Description

This use case describes how a user can send text messages to others.

2.3.9.2 Actors

1) User

2.3.9.3 Preconditions

1) The user has an active Internet connection on their device.
2) The user has previously paired successfully with another user.

2.3.9.4 Basic Flow of Events

1) The use case begins when the user is on the main hub of the application.
2) The user will choose a user they have paired with from the list of conversations and press on them.
3) The application will direct the user to a screen which has a textbox on the bottom with various options. The rest of the screen will be used for the conversation speech bubbles.
4) The user will press on the textbox and their device will bring out the keyboard to allow typing.
5) Once the user has finished writing their message they will press send.
6) Use Case: Encrypt Messages will be executed.
7) Once the message is successfully encrypted the message will be sent to the recipient.
8) Once the recipient has successfully received the message the application will decrypt the message and display in clear text.
9) The use case ends successfully.

2.3.9.5 Alternate Flows

2.3.9.5.1 Loss of Connection

In step 7 of the basic flow if the user loses Internet connection while the message is being sent, then

1) The application will highlight the sent message with a red outline.
2) The user will be notified that the message has failed to send.
3) The user will be requested to re-connect to an active connection.
4) The use case will terminate with a failure.

2.3.9.6 Post-conditions

2.3.9.6.1 Successful Completion

The user has successfully sent a secured text message to a paired peer.

2.3.9.6.2 Failure Condition

The application will display to the user that the message has failed to be sent with an appropriate error message.

2.3.10 Use Case 8 <Sending Images>

2.3.10.1 Brief Description

This use case describes how the user can send an image to a peer.

2.3.10.2 Actors

1) User

2.3.10.3 Preconditions

1) The user has an active Internet connection.
2) The user has previously paired successfully with another user.

2.3.10.4 Basic Flow of Events

1) The use case begins when the user has opened a conversation from the main hub.
2) The user presses the Camera button beside the text box.
3) The application will open the device's Gallery section.
4) The user will be able to browse naturally through the various folders they have available.

5) Once the user has selected an image they can choose to Select Image by pressing the Select button.
6) The application will bring the user back to the conversation screen with the image filling the textbox.
7) The user will press send and the image will be sent to the recipient.
8) Use Case: Encrypt Message will be executed.
9) Once the encryption is complete the image will be sent to the recipient.
10) When the recipient has received the image, it will be decrypted only when the user clicks on the image to open it within the application.
11) The user will be disallowed from taking screenshots and saving the image to their device.

2.3.10.5 Alternate Flow

2.3.10.5.1 Loss of Connection

In step 9 of the basic flow if the user loses Internet connection while the message is being sent, then

1) The application will highlight the sent message with a red outline.
2) The user will be notified that the message has failed to send.
3) The user will be requested to re-connect to an active connection.
4) The use case will terminate with a failure.

2.3.10.6 Post-Conditions

2.3.10.6.1 Successful Completion

The user has successfully chosen an image from their device and sent an encrypted version to a peer.

2.3.10.6.2 Failure Condition

The application has failed to send the image to the recipient and will display an appropriate error message.

2.3.11 Use Case 9 <Sending Documents>

2.3.11.1 Brief Description

This use case describes how the user can send a document to their peers.

2.3.11.2 Actors

1) User

2.3.11.3 Preconditions

1) The user has an active Internet connection.
2) The user has previously paired successfully with another user.

2.3.11.4 Basic Flow of Events

1) The use case begins when the user has opened a conversation from the main hub.
2) The user presses the Paperclip button besides the Camera symbol.
3) The application will open the Documents folder on the user's device.
4) The user can browse through their documents which they have saved on their device.
5) Once the user has selected the document they wish to send they can press the Select button.
6) The application will bring the user back to the conversation screen with the document name and extension displayed in the textbox.
7) The user will press the send button and the document will be sent to the recipient.
8) Use Case: Encrypt Message will be executed.
9) Once the encryption is complete the message will be sent by the application to the recipient.
10) Once the recipient receives the document it will be decrypted only when they open it within the application.

2.3.11.5 Alternate Flow

2.3.11.5.1 Loss of Connection

In step 9 of the basic flow if the user loses Internet connection while the message is being sent, then

1) The application will highlight the sent message with a red outline.
2) The user will be notified that the message has failed to send.
3) The user will be requested to re-connect to an active connection.
4) The use case will terminate with a failure.

2.3.11.6 Post-Conditions

2.3.11.6.1 Successful Completion

The user has successfully selected a document from their device and sent an encrypted version to their peer.

2.3.11.6.2 Failure Condition

The application has failed to successfully send the document to the recipient and will notify the user with an appropriate error response.

2.3.12 Use Case 10 <Encrypt Messages>

2.3.12.1 Brief Description

This use case describes the process in which the application encrypts the various types of content sent by the user.

2.3.12.2 Actors

1) User

2.3.12.3 Preconditions

1) The user has an active Internet connection.
2) The user has attempted to send a message to a peer.

2.3.12.4 Basic Flow of Events

1) This use case begins when the user has pushed on the 'Send' button beside the textbox.
2) The application will receive the content which is in the textbox.
3) The application will determine the nature of the content.
4) The application will apply content specific encryption methods to the message.
5) The application will return the encrypted version of the message to the textbox.
6) The application will proceed to send the message to the recipient.
7) The use case ends successfully.

2.3.12.5 Alternate Flow

2.3.12.5.1 Encrypting text messages

In step 3 of the basic flow if the application determines the content to be a text message, then

1) The application will extract the content from the textbox.
2) The application will use E2EE encryption utilizing the keys stored in the local storage of the device.
3) The use case will resume from step 5 of the basic flow.

2.3.12.5.2 Encrypting Images

In step 3 of the basic flow if the application determines the content to be an image, then

1) The application will extract the image from the textbox.
2) The application will break down the image down to pixels.
3) The application will apply an encryption algorithm to change the position of the pixels within the width and height parameters.
4) The use case will resume from step 5 of the basic flow.

2.3.12.5.3 Encrypting Documents

In step 3 of the basic flow if determines the content to be a document, then

1) The application will extract the document from the textbox.
2) The application will open the document and scan the content.
3) It will then apply a symmetric encryption on the contents of the document based on the encryption key stored on the device.
4) The use case will resume from step 5 of the basic flow.

2.3.12.6 Post-Conditions

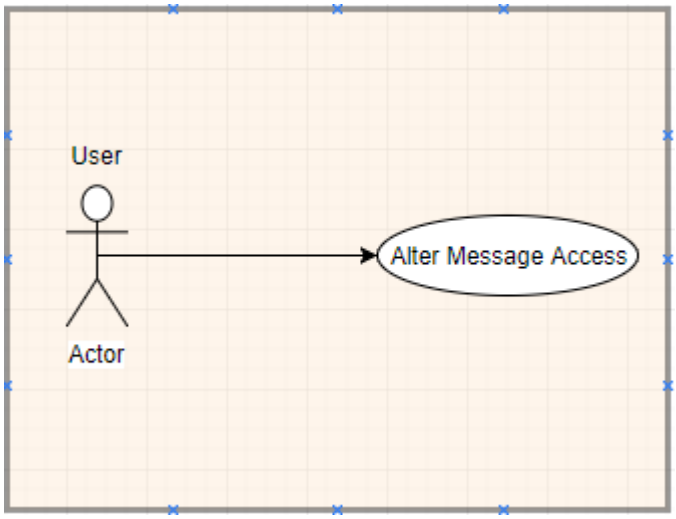
2.3.12.6.1 Successful Completion

The application has successfully determined the type of content that is present and has applied an encryption algorithm specifically based on the content.

2.3.12.6.2 Failure Condition

The application has failed to determine the type of the content and has not completed the encryption process. The use case will fail.

2.3.13 Use Case 11 <Altering Message Access Level>



2.3.13.1 Brief Description

This use case describes the user being able to control the level of access granted to the messages being sent. This will apply to messages containing images or documents.

2.3.13.2 Actors

1) User

2.3.13.3 Preconditions

1) The user has an active Internet connection.
2) The user has compiled a message to be sent to a recipient.

2.3.13.4 Basic Flow of Events

1) The user can press the Padlock symbol beside the textbox before sending the message.
2) The application will send the message normally like before.
3) Once the recipient has received the message whether image or document, the application will decrypt the content.
4) The application will save a copy of the decrypted version in a media folder in the application.
5) This data will be stored on the device and a folder will be created in the Gallery section of the device.

2.3.13.5 Alternate Flow

2.3.13.6 Post-Conditions

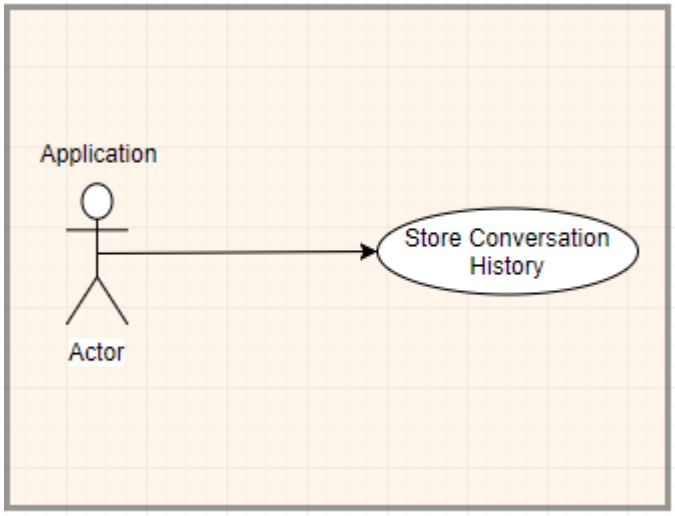
2.3.13.6.1 Successful Completion

The user has successfully altered the access rights of the image and the recipient can now save the content to their device.

2.3.13.6.2 Failure Condition

The application has failed to enforce the changes made by the user and the recipient will fail to store the content on their device.

2.3.14 Use Case 12 <Store Conversation History>



2.3.14.1 Brief Description

This use case describes the system storing the messages which have been sent by both users for a period of one month.

2.3.14.2 Actors

1) Application

2.3.14.3 Preconditions

1) The user has an active Internet connection.
2) The user has sent and received messages from a Paired peer.

2.3.14.4 Basic Flow of Events

1) This use case begins when the user has started a conversation with another user.
2) The application will create an object for each conversation which will be stored in the user's local storage.
3) This object will be updated whenever a new message has been successfully received or sent.
4) The user can scroll through the history of the conversation.
5) The application will only load chunks of the object to lower computational load.
6) The user can choose 'Show More' when they have reached the end of the chunk of messages.
7) The use case ends successfully.

2.3.14.5 Alternate Flow

2.3.14.5.1 Periodic Updates

In step 3 of the basic flow if the application has failed to successfully update the conversation state, then

1) The application will have a pre-set time interval where it will scan the whole conversation.
2) The application will create a new object for the conversation.
3) The application will access the local storage and find the specific conversation.
4) The application will proceed to overwrite the old conversation state with the new object.
5) The use case will resume from step 4.

2.3.14.6 Post-Conditions

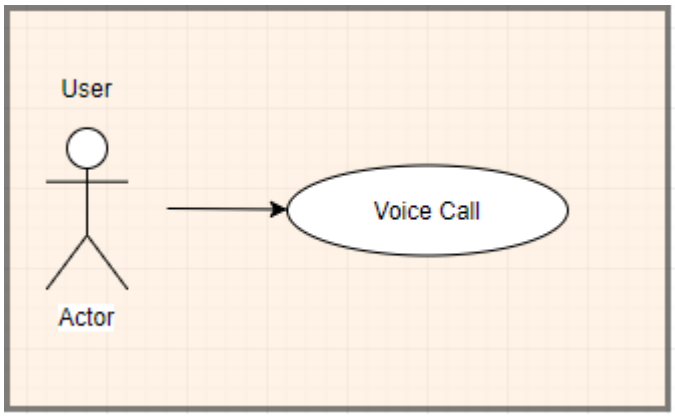
2.3.14.6.1 Successful Completion

The application has successfully kept a log of conversations through maintaining the conversation object. The automatic scans will ensure data integrity if there was an error in saving a message.

2.3.14.6.2 Failure Condition

The application has failed in storing the message and the user will only have a partial record of the conversation.

2.3.15 Use Case 13 <Voice Call>



2.3.15.1 Brief Description

This use case describes how a user can voice call another user they have paired with.

2.3.15.2 Actors

1) User

2.3.15.3 Preconditions

1) The user has an active Internet connection.
2) The user has previously paired with other users.

2.3.15.4 Basic Flow of Events

1) The use case begins when the user is in the main hub of the application.
2) The user will find a user who they wish to call and press their username.
3) The application will open the conversation area where the users can send messages between each other.
4) The user can press Phone symbol in the top right corner of the screen.
5) The application will use an internet connection to call the other user.
6) The application will continue to ring until the recipient answers the phone call.

7) The application will open a call screen which provides the user with several options available.
    a. Mute call
    b. End call
    c. Loudspeaker
8) The user can choose to end the call at any time by pressing the End Call button.
9) The use case ends successfully

2.3.15.5 Alternate Flows

2.3.15.5.1 Recipient Fails to Answer

In step 6 of the basic flow if the recipient does not answer, then

1) The application will continue to ring for 15 seconds.
2) If the recipient has failed to answer within that time the application will display a message saying the User is Busy.
3) The use case will fail and revert to step 3 of the basic flow.

2.3.15.6 Post-conditions

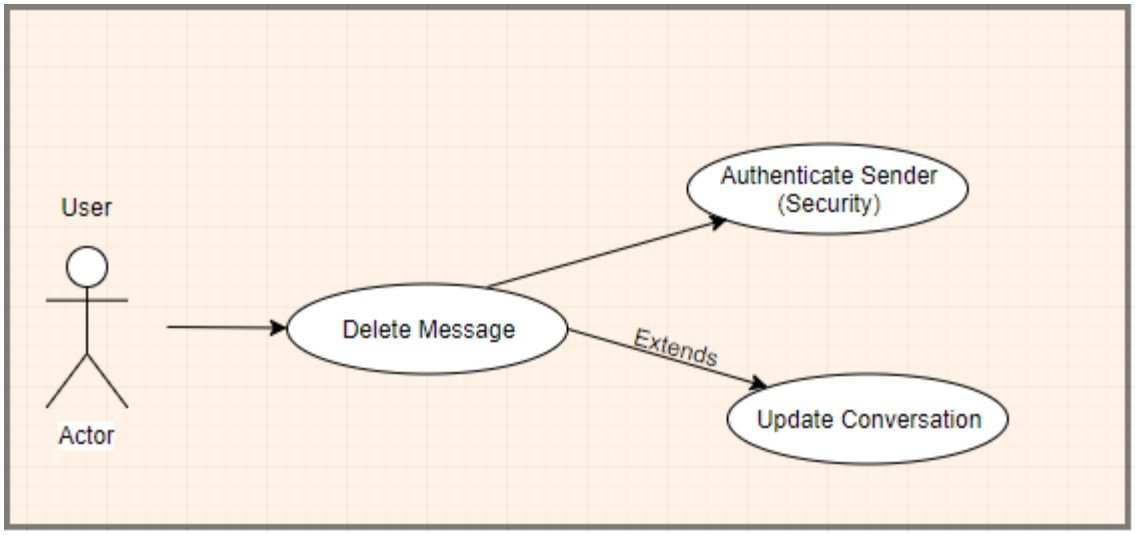
2.3.15.6.1 Successful Completion

The user has successfully called a user.

2.3.15.6.2 Failure Condition

The call failed to be established and the application will revert to the conversation screen.

2.3.16 Use Case 14 <Delete Message>



2.3.16.1 Brief Description

This use case describes the process of a user choosing to delete a message from a conversation.

2.3.16.2 Actors

1) User

2.3.16.3 Preconditions

1) The user has an active Internet connection
2) The user has an active conversation with another user.

2.3.16.4 Basic Flow of Events

1) The use case begins when the user is the main hub of the application.
2) The user chooses a conversation from which they wish to delete a message from.
3) The user will be able to scroll through the conversation for up to a month in the past.
4) Once the user has found a message they can press on the message.
5) The application will display a Trash icon beside the chat bubble.
6) The user presses the Trash icon.
7) Use Case: Authenticate Sender will be executed.
8) The application will delete the message from the conversation.
9) Use Case: Update Conversation will be executed
10) The use case ends successfully.

2.3.16.5 Alternate Flows

2.3.16.6 Post-conditions

2.3.16.6.1 Successful Completion

The user was successful in deleting a message and the application successfully updates the conversation sides for both parties.

2.3.16.6.2 Failure Condition

The user failed to delete a message which was not sent by them and the application sent a valid error message.

2.3.17 Use Case 15 <Authenticate Sender>

2.3.17.1 Brief Description

This use case describes the process of authenticating the sender of the message before allowing deletion by user.

2.3.17.2 Actors

1) User

2.3.17.3 Preconditions

1) The user has an active Internet connection.
2) The user has an active conversation with another user.
3) The user has proceeded to attempt a deletion of a message.

2.3.17.4 Basic Flow of Events

1) The use case begins when the user has pushed the Trash symbol beside a message.
2) The application will check the encryption key of the message and match it to the user's encryption key.
3) The application will confirm that the message has been sent by the user and not the other party.
4) The use case ends successfully.

2.3.17.5 Alternate Flow

2.3.17.5.1 Deleting Other's Messages

In step 1 of the basic flow if the user has selected to delete a message not sent by them, then

1) The user will press the Trash icon.
2) The application will execute Use Case: Authenticate Sender
3) The encryption key of the User will not match with the encryption key used for the message sent.
4) The application will notify the user the message is not sent by them.

The use case will fail and resume from step 3 of the basic flow.

2.3.17.6 Post-Conditions

2.3.17.6.1 Successful Completion

The application has successfully confirmed the sender of the message to be the user preserving the integrity of the data.

2.3.17.6.2 Failure Condition

The application failed in the authentication process and allowed the user to delete messages sent by their peers.

2.3.18 Use Case 16 <Update Conversation>

2.3.18.1 Brief Description

This use case describes the process taken by the application of updating the conversation for both parties when a deletion has occurred.

2.3.18.2 Actors

1) User

2.3.18.3 Preconditions

1) The user has an active Internet connection.
2) The user has an active conversation with another user.
3) The user has deleted a message of theirs.

2.3.18.4 Basic Flow of Events

1) The use case begins when the user has deleted a message of theirs.
2) The application will immediately scan the state of the conversation object.
3) The application will overwrite the old state from the user's device with the newly scanned object.
4) The application will send the new environment to the server, which will be directed to the recipient.
5) The application will update the state of the environment on the recipient's device by overwriting the old environment with the new one.
6) The use case ends successfully.

2.3.18.5 Alternate Flow

2.3.18.5.1 Failure to Send Environment

In step 4 of the basic flow if the application fails to send the new state to the server then,

1) The application will receive a message from the server indicating a no response.
2) The application will attempt to send the state again.
3) This process will repeat 3 times.
4) The application will initiate a new scan of the conversation state.
5) The application will collect the new state and attempt to send it to the server.

2.3.18.6 Post-Conditions

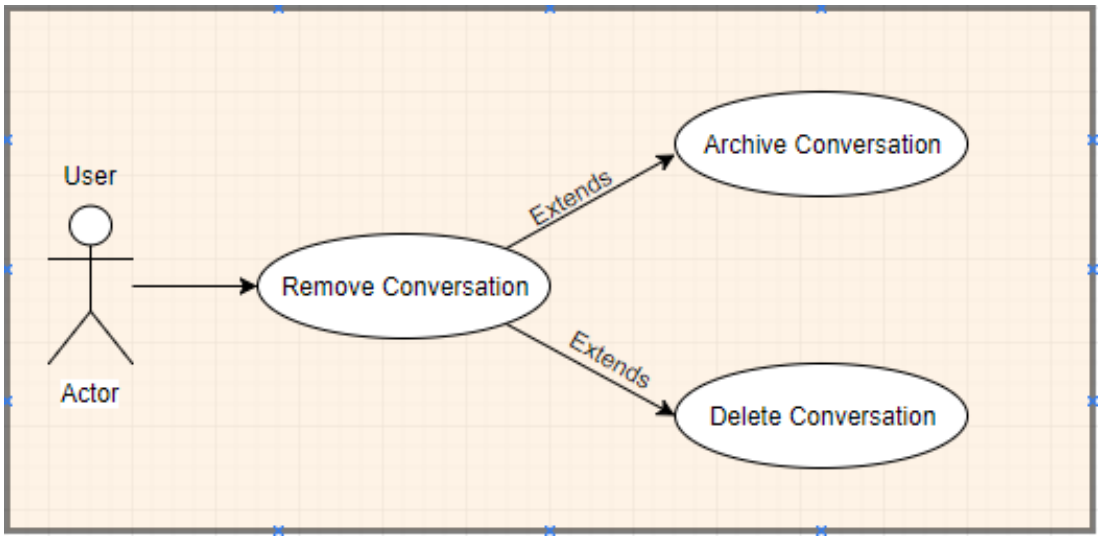
2.3.18.6.1 Successful Completion

The application has successfully allowed the user to delete a message and updated the state of the conversation on the devices of both parties.

2.3.18.6.2 Failure Condition

The application has failed to send the conversation state to the recipient and an entry will be logged into the System Log.

2.3.19 Use Case 17 <Archive Conversation>



2.3.19.1 Brief Description

This use case describes the ability of a user to archive a conversation from their main hub.

2.3.19.2 Actors

1) User

2.3.19.3 Preconditions

1) The user has an active Internet connection on their device.
2) The user has pre-existing conversations in their main hub.

2.3.19.4 Basic Flow of Events

1) The use case begins when the user is in the main hub of the application.
2) The user chooses a conversation which they wish to remove from their hub.
3) The user will hold down the conversation for 2 seconds.
4) The application will display the user with two options:
    a. Archive Conversation
    b. Delete Conversation
5) The user chooses the 'Archive Conversation' option and the application will ask for a confirmation.
6) Once the user has confirmed the request the application will remove the conversation from the main hub of the application.
7) The application will move the conversation to an archived section.
8) The use case will resume from step 1 of the basic flow.
9) The use case ends successfully.

2.3.19.5 Alternate Flow

2.3.19.5.1 Removing Conversation from Archives

In step 7 of the basic flow if the user has archived a conversation which they would like to bring back to their main hub, then

1) The user can choose to access the archived conversations to view the current conversations existing there.
2) The user can hold the conversation for 2 seconds.
3) The system will ask if the user would like to remove the conversation from the archive.
4) If the user confirms the request the application will move the conversation back to the main hub.

2.3.19.6 Post-conditions

2.3.19.6.1 Successful Completion

The user was able to successfully remove a conversation from their main hub.

2.3.19.6.2 Failure Condition

The system failed to remove the conversation and will notify the user with an appropriate error message. The application will request the user to re-attempt.

2.3.20 Use Case 18 <Delete Conversation>

2.3.20.1 Brief Description

This use case describes the process of the user deleting a conversation.

2.3.20.2 Actors

1) User

2.3.20.3 Preconditions

1) The user has an active Internet connection on their device.
2) The user has pre-existing conversations in their main hub.

2.3.20.4 Basic Flow of Events

1) The use case begins when the user is in the main hub of the application.
2) The user chooses a conversation which they wish to remove from their hub.
3) The user will hold down the conversation for 2 seconds.

4) The application will display the user with two options:
    a. Archive Conversation
    b. Delete Conversation
5) The user chooses the ‘Archive Conversation’ option and the application will ask for a confirmation.
6) Once the user has confirmed the request the application will remove the conversation from the main hub of the application.
7) The application will proceed to delete all the messages sent from the user in the conversation.
8) The application will update the conversation for the other party.
9) The application will then remove the conversation from the main hub of the application.
10) The use case will resume from step 1 of the basic flow.

2.3.20.5 Alternate Flow

2.3.20.6 Post-Conditions

2.3.20.6.1 Successful Completion

The user was successful in deleting a conversation with the messages also being removed from the recipient’s side.

2.3.20.6.2 Failure Condition

The application failed to remove the messages from the recipient’s device.

2.3.21 Abuse Case 1 <Steal Device>

2.3.21.1 Brief Description

This use case describes how a malicious user or attacker can gain privileges to the user’s account.

2.3.21.2 Actors

1) Attacker

2.3.21.3 Preconditions

1) The device has the application installed and an account registered.
2) The device has an active Internet account.

2.3.21.4 Basic Flow of Events

1) The use case begins when the attacker can gain access to the user's device through theft.
2) The attacker will access the device and open the application.
3) The attacker will have access to all the conversations of the user.
4) The attacker can view the profile of the user and view details which the user has input.
5) The attacker can message people pretending to be the victim.
6) The attacker can delete conversations from the victim's main hub.
7) The use case ends successfully.

2.3.21.5 Alternate Flows

2.3.21.5.1 Device has authentication

In step 2 of the basic flow if the victim's device has been set up with authentication such as a pattern lock or a passcode, then

1) The attacker will attempt to break through method of authentication.
2) If the attacker is unsuccessful the use case will end in a fail state.

2.3.21.6 Post-conditions

2.3.21.6.1 Successful Completion

The attacker was successful in stealing the victim's device and gaining access to privileges in the account they didn't have authorization for.

2.3.21.6.2 Failure Condition

The attacker was unable to bypass the devices authentication method and failed to gain access to the application.

2.3.22 Abuse Case 2 <Intercept Keys>

2.3.22.1 Brief Description

This use case describes the process an attacker can use to intercept the encryption keys of the communicating parties.

2.3.22.2 Actors

1) Attacker

2.3.22.3 Preconditions

1) The attacker has software which allows for eavesdropping network traffic.

2.3.22.4 Basic Flow of Events

1) The use case begins when the victim is starting to Pair with a User.
2) The application is in the process of exchanging encryption keys during Pairing.
3) The attacker uses software to eavesdrop on the network activity of the victim.
4) The attacker isolates the traffic generated by the LockTalk application.
5) The attacker manages to access the encryption keys being exchanged.
6) The application has failed to encrypt the encryption successfully and are available in its original state.
7) The attacker can now use the encryption keys to intercept messages being sent and received and decrypt them without being in the conversation.
8) The attacker gains access to unauthorized privileges and breaches the victim's security and privacy.
9) The use case ends successfully

2.3.22.5 Alternate Flows

2.3.22.5.1 Successful encryption of keys

In step 5 of the basic flow if the application has successfully encrypted the keys, then

1) The attacker will have access to the encrypted keys
2) The attacker will use the encryption keys to decrypt messages.
3) The messages will stay encrypted as the key doesn't match.
4) The attacker will fail to intercept messages in their decrypted state.
5) The use case will fail.

2.3.22.6 Post-conditions

2.3.22.6.1 Successful Completion

The attacker has successfully gained access to unencrypted keys during the Pair Exchange. The attacker will breach the confidentiality of the data.

2.3.22.6.2 Failure Condition

The application has successfully encrypted the keys during the exchange and the attacker will receive data which will be unable to decrypt the messages.

2.3.23 Abuse Case 3 <Save Messages>

2.3.23.1 Brief Description

This use case describes how an attacker can breach the privilege of saving user messages.

2.3.23.2 Actors

1) Attacker

2.3.23.3 Preconditions

1) The attacker has paired successfully with the victim.
2) The attacker has access to a separate device with a functional camera.

2.3.23.4 Basic Flow of Events

1) The use case begins when the attacker is having a conversation with a potential victim.
2) The victim has send personal data whether it is text or an image.
3) The attacker will be unable to screenshot the messages due to the restriction applied by the application.
4) The attacker will view the message as normal.
5) The attacker will use their secondary device to take a picture of the screen on their primary device.
6) The application will not be able to detect any breaches from a secondary device.
7) The attacker has gained access to data which the victim has privatized and has breached the confidentiality of the data.
8) The attacker can use the data for numerous malicious purposes such as blackmail or to breach privacy.

2.3.23.5 Alternate Flows

2.3.23.6 Post-conditions

2.3.23.6.1 Successful Completion

The attacker has successfully captured data from the application bypassing the built-in measures to counter unauthorized saving of messages.

2.3.23.6.2 Failure Condition

2.3.24 Abuse Case 4 <Send Malicious Content>

2.3.24.1 Brief Description

This use case describes the method in which an attacker can send malicious content to victim.

2.3.24.2 Actors

1) Attacker

2.3.24.3 Preconditions

1) The attacker has paired successfully with the victim.
2) The attacker has been having a conversation with the victim.
3) The attacker has an active Internet connection.

2.3.24.4 Basic Flow of Events

1) The use case begins when the attacker is having a conversation with the victim.
2) The attacker has created a document which embeds itself into the device when opened.
3) The attacker proceeds to send the document the victim claiming innocence of the document.
4) The victim opens the document and the malware gains access to the device.
5) The nature of the malware can be varied and therefore the impact can also vary from access to features of the application and device.
6) The attacker can use blackmail or extortion to gain something of value from victim.
7) The use case ends successfully.

2.3.24.5 Alternate Flows

2.3.24.5.1 Anti-Malware software

In step 4 of the basic flow if the victim's device has pre-installed anti-malware software, then

1) The software will recognize that the document has introduced a threat to the device.
2) The software will notify the user that a threat has been identified.
3) The software will eradicate the malware from the victim's device.
4) The victim will recognize the malicious intent of the attacker.
5) The victim will proceed to delete their conversation and not contact the attacker.
6) The use case terminates with a failure.

2.3.24.5.2 User refuses to open document

In step 4 of the basic flow if the victim refuses to open the sent document, then

1) The attacker will fail to infect the victim's device with malware.
2) The use case terminates with failure.

2.3.24.6 Post-conditions

2.3.24.6.1 Successful Completion

The attacker has successfully infiltrated the victim's device through an infected document. The user has gained access to unauthorized data and functionality.

2.3.24.6.2 Failure Condition

The attacker fails to gain access to the victim's device.

2.4 Non-Functional Requirements

2.4.1 Performance/Response time requirement

Since LockTalk is a messaging application and therefore does not require near-instantaneous response time the following metrics will be applied to measure the performance and response time:

1) The system shall keep a steady rate of less than one second when transitioning between the various sections of the application.
2) The system shall not allow more than five seconds to pass when saving any changes made to the profile.
3) The system shall allow twenty seconds for messages to be delivered from the sender to the recipient.
4) The system shall return a response to user search queries within 2 seconds.

2.4.2 Availability requirement

The LockTalk messaging application will require an active Internet connection to be available for use however, there are some availability metrics from the application side which must also be considered:

1) The system shall aspire to maintain an availability percentage of 97% throughout the year.
2) The system shall aspire to resolve any issues regarding availability within the first day of the issue being discovered.

2.4.3 Security requirement

The LockTalk messaging application is heavily focused on security therefore there is large focus on establishing metrics to gauge the security of the application:

1) The application shall authenticate the user of the application before allowing any interaction with the functionality provided.
2) The application shall not allow unauthorized access to the database and any of the data within.
3) The application shall ensure the integrity of the data stored by the application.
4) The data stored on the database must be encrypted in the case of a leak.
5) The application shall protect the data transfer between two communicating parties from outside malicious sources.
6) The application shall implement secure up-to-date encryption methods to avoid attacks on the confidentiality of the data.
7) The application shall adhere to the best practices available during the development process to avoid possible vulnerabilities.
8) The system shall ensure that any files sent over to a recipient are free from known malware.
9) The application shall ensure that users can't access functionality which they do not have privilege to.

2.4.4 Extendibility requirement

Since LockTalk is subject to future improvements made there must be certain extensibility requirements in place:

1) The system shall be able to adapt new functionality or modifications to existing functionality without having to modify the whole system.
2) The system shall implement an open-box extensibility protocol to allow for extending or modifying the source code.

2.4.5 Resource utilization requirement

Since LockTalk is a mobile application there must be certain metrics put in place to control the resource utilization of the application:

1) The system shall ensure that the overall storage of data will not exceed more than 100MB.
2) The system shall not use more than 0.7MB of 3G data per minute while a user voice calls another party.

2.5 Data requirements

The LockTalk messaging application focuses heavily on user security and therefore, must apply those principles to the way it handles the user data. There are certain key points which the application must address when it comes to the data requirements:

1) The application will require two separate components for storing data.
    a. The local storage will be handled by a SQLite 3 database which is native to Android devices. This database will be responsible for storing the private keys for the user to ensure the confidentiality of the data which is integral to the security aspects of the application. The SQLite database will also store states of each of the conversations which the user has on their main hub. These states will be constantly updated and maintained for data integrity purposes. The conversation states will only story messages for a period of one month after which older messages will begin to be discarded.
    b. The AWS server which will be running the MySQL server is where the application will store the user data provided during account creation. This data will be stored in encrypted forms so that any breaches in the security of the database will cause the damages to be mitigated. The AWS server will also store any archived conversations from users which they don't store on their device. These conversations may contain sensitive information therefore, they must stay in their encrypted forms. The user must have the ability to request conversations to be removed from Archives and receive them without delay. The database must be able to allow fast access and retrieval of such records to ensure data availability.
2) The types of data which will be stored by both mediums are as follows:
    a. Username
    b. Mobile Number
    c. Email (Optional)
    d. First & Last Name
    e. Public Keys
    f. Private Keys
    g. Conversation States
    h. Users
    i. Images
    j. Documents

2.6 User requirements

During the requirement's elicitation a survey was carried out through questionnaires being asked from potential users in Dublin City Centre. This included stopping people from various demographics and asking them a pre-set list of questions. The first half of the questions was to determine the demographic and their interest in the market. The second half dealt with specific requirements which the users were requesting. The results of the elicitation were the following:

1) There was a total of 30 people question with the demographics split as follows:
    a. Men constituted 63.3% and women 36.7% of the questioned people.
    b. The age groups of the questioned people can be broken down to:
        i. 14 – 18 constituted 16.7% with 5 people questioned
        ii. 19 – 25 constituted 70% with 15 people questioned
        iii. 26 – 40 constituted 20% with 6 people questioned
        iv. 40+ constituted 13.3% with 4 people questioned.
2) Amongst the age groups of 14 – 25, 100% of the questioned demographic used some version of a messaging application. Amongst the older 26 – 40+ range the figure dropped to 60%.
3) The age group with the highest percentage of concern for security aspects of messaging apps was 19 – 25 with 75%. They were followed by 14 – 18 with 60%. Both the 26 – 40 and 40+ age groups shared an average of 50% in concern with security in messaging applications.
4) Amongst the 19 – 25 demographics 86% of the people questioned replied that they had experienced moments when they wanted to take back a message whether it was sent while inebriated or it was private. The age groups below and above shared a smaller percentage which averaged to 38% who had shared the same experience.
5) The final section of the questionnaire was to ask the people for input on what 3 features they would like to see in the application if they were to use it. These were 10 most popular requirements according to the questioned people:
    a. The application should not be too slow in response and performance.
    b. The application should disallow screenshotting of content.
    c. The application shouldn't use confusing transitions between pages.
    d. The application should allow for images/documents to be sent.
    e. The application should give the users an option on how private they want to be.
    f. The application should follow a fluid and modern design.
    g. The application shouldn't allow outside sources to eavesdrop conversations. This should be achieved by E2EE.
    h. The application should allow both single and group chats.
    i. The application should allow people to call others.

j. The application should ensure that all data is securely stored.

A feature which was quite surprising was to send documents which were mostly suggested by the older age groups who wished that sending work documents could be safer and more secure.

2.7 Environmental requirements

2.8 Usability requirements

Since the intended demographic for LockTalk ranges widely from all ages the application must provide certain usability metrics to gauge the level of difficulty for users to access the provided functionality.

1) The application shall be designed with clear representations of the current area the user is on.
2) The system shall clearly state where the user can navigate to following the press of a specific button.
3) The system shall be designed with focus on certain disabilities such as colour-blindness by excluding certain colour schemes.
4) Users shall be able to operate the application comfortably after 15 minutes of use.
5) Over 90% of the users will be able to execute all functionality provided by the application within their first time using the application without assistance.

2.9 System Evolution

The evolution of the system can be comprised of various categories in which improvements can be made over the course of the application life cycle. These categories are as follows:

2.9.1 Compatibility

The current version of the LockTalk messaging application will be available exclusively on the Android platform, however this is an aspect which can be improved upon in the future. The application can be made cross-platform so that it can be available on iOS devices as well as being accessible through a web application.

2.9.2 Functionality

There are numerous functionality improvements which can be implemented on the LockTalk application such as:

2.9.2.1 Group Messaging

The current version of the application limits the users to have conversations only with one person however, a future update could be implemented where users can create group chats where multiple parties can participate in while still maintaining the security and control of content features.

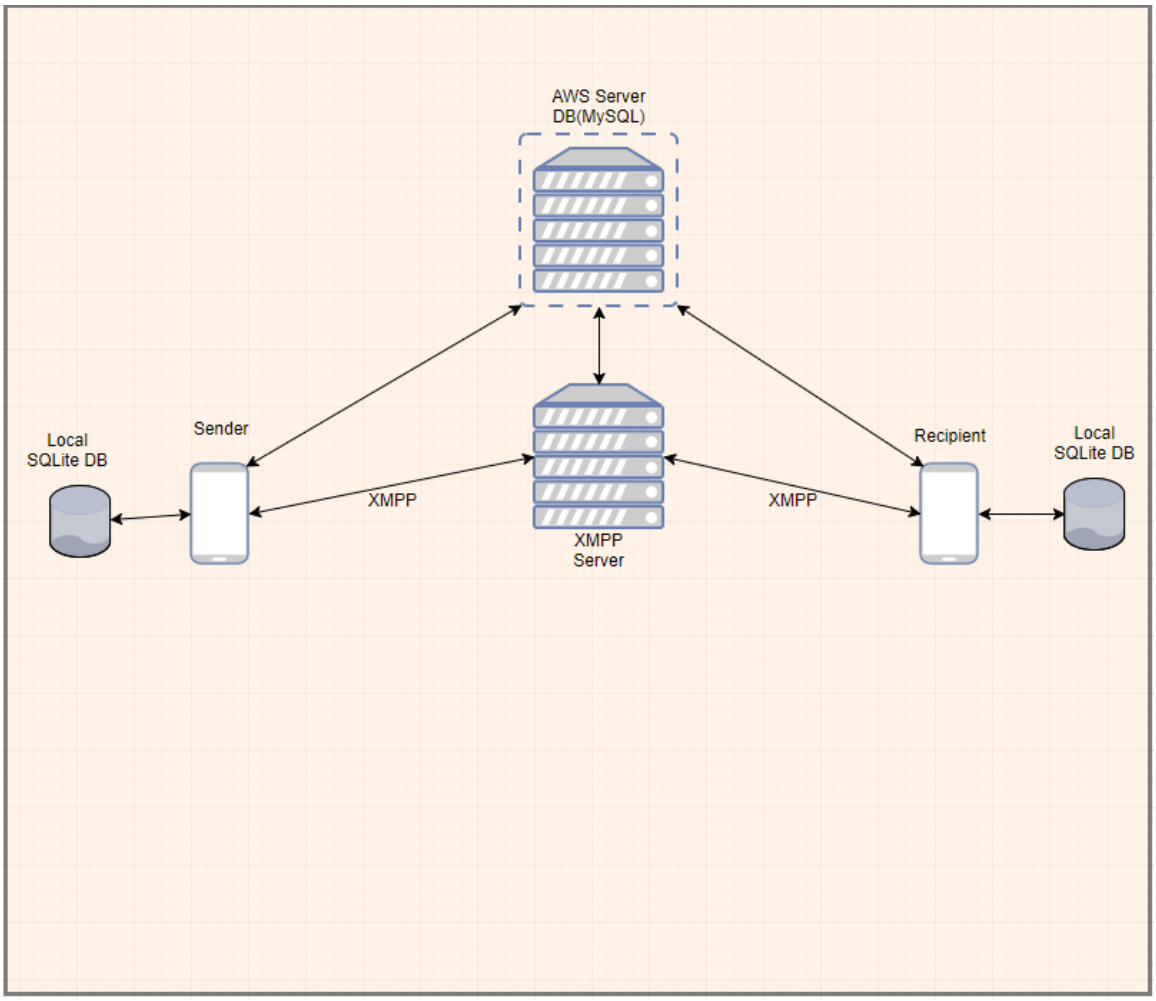
2.9.2.2 Message Content

The first version of LockTalk will support messages including text, images and documents. This can be further improved in the future to allow users to send animated GIFs using a variety of libraries, as well as audio and video clips.

2.9.3 Security

One of the most important aspects of the messaging application is the security it provides to the users when they communicate to peers through LockTalk. The current E2EE methods may become obsolete in the future and it is therefore, the duty of the application to continue improving the encryption methods as time goes on to ensure the users of its security capabilities. There is also a possibility of allowing each conversation to begin the use of VPN and Proxy servers to provide users with much more anonymity while using the LockTalk application.

3 Design and Architecture

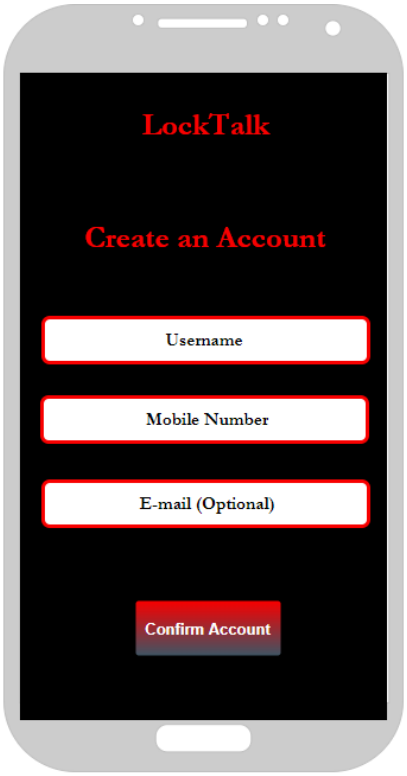


The system architecture will consist of numerous components which will communicate between each other to maintain the functionality of the application. The first components are the Android applications named ‘Sender’ and Recipient’. These represent a communication between two users who have registered an account. Each of these devices have a link to a local SQLite database which will be used to store data regarding the encryption keys for the device as well as the encryption keys for the other party. The application on the devices will be connected to the AWS Server which will be running the MySQL database containing user data which was provided during account creation as well as other information generated by the application. The XMPP server which will be linked to

both the AWS server as well as the devices running the application will be handling the Queue management for the messages which are being sent. The communication between the application and the XMPP server will be handled by the XMPP protocol. The encryption of messages will be done before the messages have been sent to the XMPP server.

4 Implementation

5 Graphical User Interface (GUI) Layout

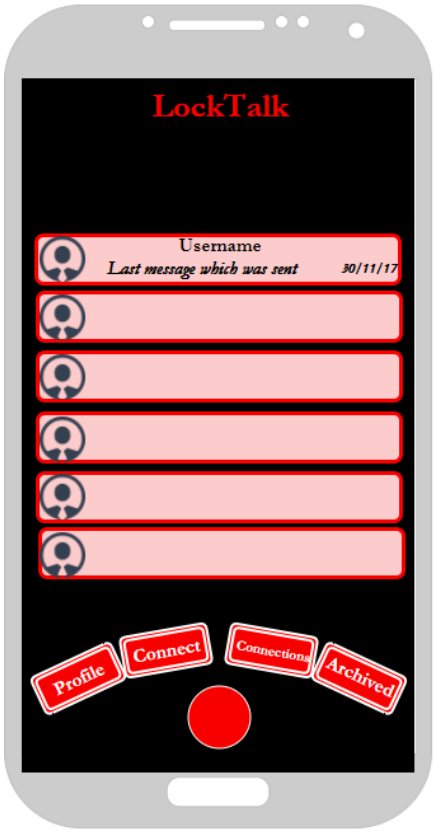


1) Create Account page

This page shows an early mock-up of the feel the Create account page will have with the fields which need to be filled as well as the button to continue into the Authentication.

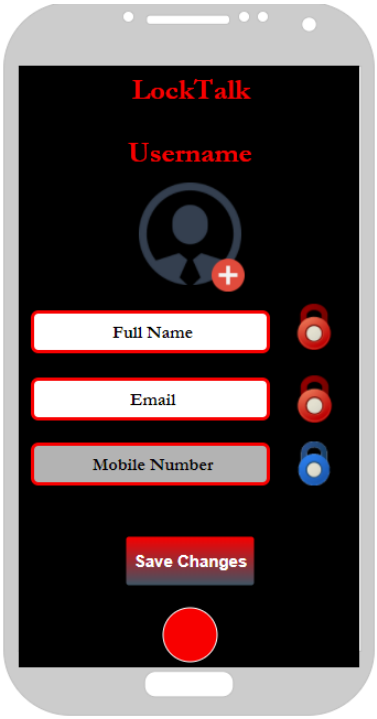
2) Authentication page

This page shows the mock-up of the look for the User Authentication page where the user must provide the 6-digit confirmation code.



3) Main Hub with Menu

This page shows the early look and feel of the main hub where the user can view their conversations and navigate to other sections.

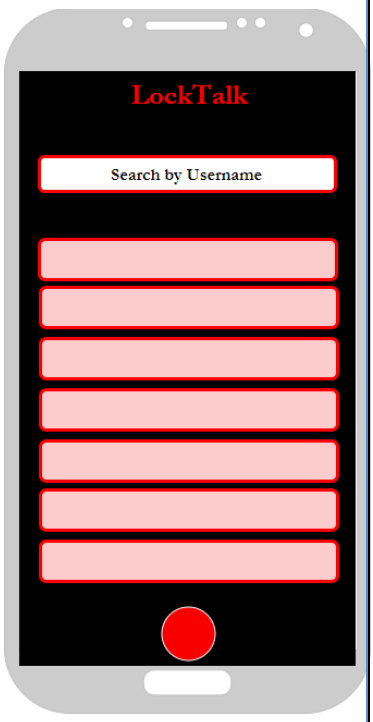


4)Personal Profile

This page shows a mock-up of the profile page of the user. They can change the fields in white and add a profile image. They can also control which fields are visible to the public during a search, using toggleable locks. Once they are happy the can save changes by pushing the button.

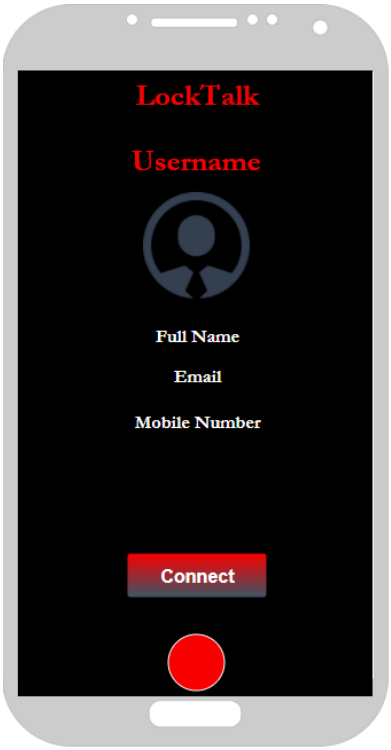
5) Search for Users

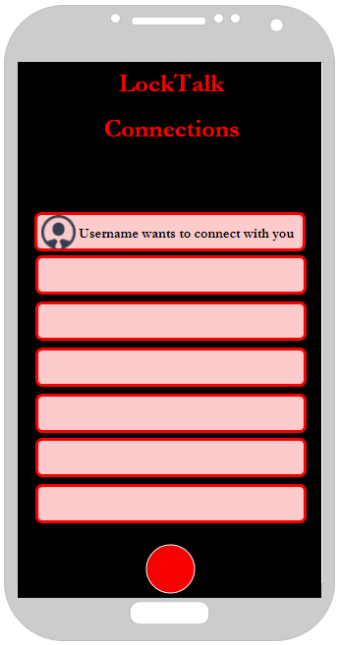
This page shows an early mock-up of the page where users can search for another user. The results will be limited to the top 20 most similar, in the case of appending.



6) Sending Pair Request

This page shows the mock-up for how a user can initiate a Pairing by visiting a user's profile showing the info they have kept public. The user can then press on the Connect button to initiate the Pairing.



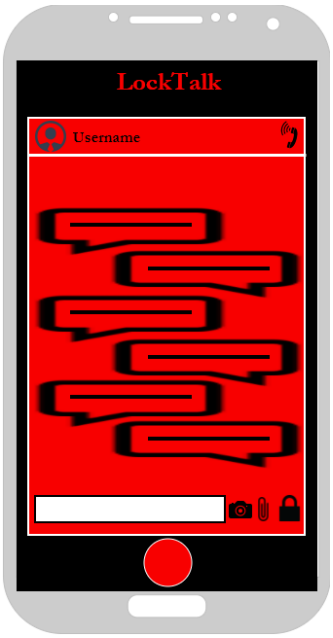


7)Connections page

This page shows a mock-up of the Connections page. This is where users can browse through the Pair Requests they have received from other users. They can choose to Accept or Decline them by clicking on a single request.

8) Conversation page

This page shows a mock-up of what the conversation page would look like for a user. It has the textbox at the bottom accompanied by the buttons for sending images, documents and controlling the access of messages. The top shows the username of the person who is communicating with the user as well as a Call button for initiating a call.



6 Testing

7 Customer testing

8 Evaluation

9 Conclusions

Towards the end of the development cycle of the application there was a lot of time to reflect on some of the different aspects regarding the project.

1) Having underestimated the complexity involved in a secure messaging application it became evident that the resource which was lacking the most was manpower. Undertaking a project which could potentially lead to launching a competitive application into the messenger market with a single developer made itself evident. The applications which dominate the market for Messaging applications have teams with dedicated developers who have immense experience in the field. This allows the developmental progress to move much quicker as tasks can be split between different groups and solving issues much more efficient when multiple people can spend time on implementing a solution.
2) During the development of the application it became clear that using a Relational Database was not the correct choice for a real-time messaging application. While the structured storage can be easier to query linked data, it became unfeasible to query data when the user expects near-instantaneous response times. Therefore, a migration was made from AWS services to using Google Firebase which employs a NoSQL database.

10 Further development or research

11 References

11.1.1.1 Square.github.io. (2018). *Picasso*. [online] Available at: http://square.github.io/picasso/ [Accessed 13 May 2018].

11.1.1.2 issue, o. (2018). *onOptionsItemSelected issue*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/10038093/onoptionsitemselected-issue [Accessed 13 May 2018].

11.1.1.3 android.support.design.widget.CoordinatorLayout$Behavior, E. (2018). *Error : Program type already present: android.support.design.widget.CoordinatorLayout$Behavior*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/49112190/error-program-type-already-present-android-support-design-widget-coordinatorl [Accessed 13 May 2018].

11.1.1.4 GitHub. (2018). *firebase/FirebaseUI-Android*. [online] Available at: https://github.com/firebase/FirebaseUI-Android [Accessed 13 May 2018].

12 Appendix

12.1 Project Proposal

12.2 Project Plan

12.3 Monthly Journals

12.4 Other Material Used

12.4.1 Questionnaire Used for User Requirements Elicitation.

The following questions were asked from a sample of 30 people in Dublin City Centre. The results were recorded on paper and then transferred to this document.

Q1) What is your age?

Q2) Do you currently use any form of a messaging application? E.g. WhatsApp, Messenger, Snap Chat.

Q3) Are you aware of any security issues regarding messaging applications? Do you have a personal worry for some of them?

Q4) Have you ever experienced the regret of sending a message which you couldn't take back?

Q5) Would you be willing to try a new messaging application which brings security as well as control over what you send?

Q6) If you were to suggest 3 features for such an application which you would consider of high importance what would they be?