MangaKo

A Manga-Style Graphical Chat Application

Noppavit Kaewlin

Tinnapop Songprachakul

Bachelor of Engineering in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Academic Year 2017

KMITL-2018-IC-B-003-011

Thesis – Academic Year 2017

Bachelor of Engineering in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Title: MangaKo: A Manga-Style Graphical Chat Application

Authors:

1. Mr. Noppavit Kaewlin Student ID 57090014
2. Mr. Tinnapop Songprachakul Student ID 57090040

Approved for submission

Natthapong J.

(Dr. Natthapong Jungteerapanich)

Advisor

Date 19 / 7 / 2018

Abstract

Text chatting is a communication tool that we use every day. However, text chatting has a flaw in not being able to convey facial expressions and emotions directly. As such, to counter this flaw, we have developed an Android app that displays chat messages in the form of manga panels.

Our project aims to create an Android application that lets users create avatars and let them use those avatars to communication their emotions and facial expressions. This application also employs the usage of sentiment analysis so that our application can determine emotions automatically from their text messages. The server side of the application uses EjabberD which is an open-source chat server that implements the XMPP Protocol. The client side is developed using Unity and Live2D SDK.

Acknowledgements

We would like to express our gratitude to our project advisor Dr. Natthapong Jungteerapanich who accepted our proposal in the first place. We are the only group of two out of the entire year and we are not the ones with the greatest of records but he went onboard with our idea and continues to be a driving force in what we want to accomplish. Without the various books and funding and experiences that he has shared with us, we wouldn't have the solid knowledge of what we wish to do.

We would like to also acknowledge and thank the various artists that came onboard with our project even if it is on a commissioning basis. Without these valiant drawers this project would not be what it is. It wouldn't be a stretch to say a big part of the quality of our work came from their prolific art styles.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Communication across all boundaries and distance has always been something humans have strived for. With the help of many new gadgets and technology, a world of distance free communication has been realized.

In the past, people used snail mail to communicate across great distances. Snail mail was taken over by telecommunication. Phones were created, and this was when people were introduced to the concept of instantaneous distance free communication. If they had phones, they can send their words and thoughts in matters of minutes and seconds, compared to weeks and days.

A few years after the introduction of house phones, came the age of portable gadgets. Mobile phones flooded the market and the world of communication got turned over its head as house phones grew obsolete and communication went as far as signals could reach.

Shortly after the advent of mobile phones, came the age of the Internet. From the most common peeps to the most influential people, everyone can, not just send their thoughts to people across the globe, but also send pictures, files and documents that, looking back to a few hundred years before, was impossible. The physical distance between people have decreased to nearly zero. The only thing that is left to be digitalized is only parcels which still rely on snail mails.

From these advancements, some people may ask, "what now?". We're able to send all kinds of data between each other. What else can we do to improve our communication?

The answer to that lies within new ways to communicate our thoughts and specifically feelings to others. Microsoft, Apple and the chatters themselves came up with the idea of Emoji and Emoticons. Line came up with the idea of sending "stickers". These inventions were created to fulfill the purpose of creating better ways to send emotions, something textual communication has never been able to do, something that's only available in the realms of video calls and face-to-face conversations. This project, MangaKo, aims to create a new platform that can effectively and creatively relay the user's emotions and feelings to their recipients.

1.2 Problem statement

The main factor that continues to burden textual communication is its inability to carry on subtle contexts that build up to 70% of successful communication. Texts that have been written to others do not have the ability to carry intonations and other features such as facial expressions and body language and this can lead to misunderstandings between people.

Line application created stickers that allow people to communicate with others via creative drawings and pictures. These things serve to not just lighten other people's day but also allow people to communicate emotions to a certain degree.

However, stickers and image macros that people use to communicate with rely heavily on artists, and there are no ways for people who cannot draw to express themselves in those forms. Instead of sitting behind images created by others, to be able to create something that you can use, that is uniquely you, would prove to be more fulfilling.

As such, project MangaKo aims to create a mobile application that you can literally insert your avatar and make them chat for you, giving life to conversations in a way that is unique to each user.

1.3 Objectives

The objective of this project is to develop a chat application prototype that can do the following:

1. Display chat conversations into manga form
2. Allow the user to express their emotion through the facial expressions of their avatar in manga form.
3. Create an immersive and new experience in text chatting.

Chapter 2

Literature Review

This chapter is a comparison of concepts and applications that are related to our project.

2.1 Emoticons

Emoticons also known as Emojis, are a composition of symbols used to represent a certain emotion or facial expression. This subchapter will go through the impacts of emoticons on chatting, and how crucial it was to the growth of the concept of adding emotions to text chatting.

An engaging conversation consists of not only words but also many different components such as facial expressions, body language, pacing and timing, all of which serve to get the correct emotion across between senders and receivers.

However, chatting is primarily text based, meaning that there is a lot of data being left out from a transmission. You cannot see the intent of the sender behind a single text bubble. Emotion is lost, and this leads to people misinterpreting texts in a chat room.

A simple joke might be taken the wrong way because the recipient doesn't know that the sender was trying to make a joke or trying to make a statement. Take for example, this conversation:

"Hey, have you seen this image yet?"

"This is the 5000th time someone had sent me that"

"Why are you angry?"

Figure 1 Example of mundane text without emoticons

This simple three-line exchange can be interpreted in two very distinct ways:

"Is the man angry and annoyed because he saw the image a couple of times already?"

Or

"Is he simply overexaggerating for comedic effects?"

With emoticons, this issue becomes less of a problem. Because we can see the icons that represents an emotion that is meant to connotate the text. For example:

" Hey, have you seen this image yet?"

"This the 5000th time someone had sent me that 😂"

"Ahaha, still pretty good the 5000th time isn't it?"

Figure 2 Example of giving emotion to text with emoticons

We now know that the recipient of the image is enjoying the image despite seeing it many times. The creation of emojis/emoticons changed the world of texting and created a new dimension for expression in a text-based conversation.

However, there are still problems with emojis, such as emojis being used for sarcasm, or how meanings of an emoji tarnishes over time.

The reason that emojis/emoticons tie directly into our project is because our project intends on providing a better solution to sending emotions across to others in a textual chat room.

2.2 Line application

Without doubt, when the general Asian populace mentions a portable chat program, most would recall "Line", a chat application created by the Line Corporation. This is due to the application's ability to be fast, reliable and enjoyable to users.

However, one of the most prolific aspects of Line is their "Line Stickers". The concept of Line stickers is that users can show their thoughts not with just faces like emojis, but with colorful and creative image macros.

These "Line stickers" took the world by storm, with creative stickers flooding the market all at once. Users do not need to write any text nor Emojis because the sticker encompasses what the user had intended to convey. From the most abstract concepts and emotions to the most general ones, artists were able to draw everything out into a tangible form that is comprehensible to anyone in and out of the conversation.

This is one solution to the problem with texting. It allows people to see the intended emotion through different methods of artistic representation.

Our project aims to take this notion even further. Line stickers are limited because everything is premade. It lacks the ability for people without the ability to draw to express their emotion in a way that is unique to them.

Instead of just letting the users choose from something premade, our project aims to create a program where the user feels immersed with what they're typing into the chat. By allowing people to use their own avatar, and having those avatars act out facial expressions, this makes the chat more personal to each user. And instead of random stickers being thrown about, users can see their conversation unfold in front of them in the form of a manga, breaking the barrier that text messaging has created, its inability to convey intent.

2.3 Microsoft comic chat

Microsoft comic chat is an interesting relic of the past. It is a program that displays chats as comic panels. It has a similar concept to our project, which is to display chats in a form of a consistent story with characters that can show their emotions and some degree of action.

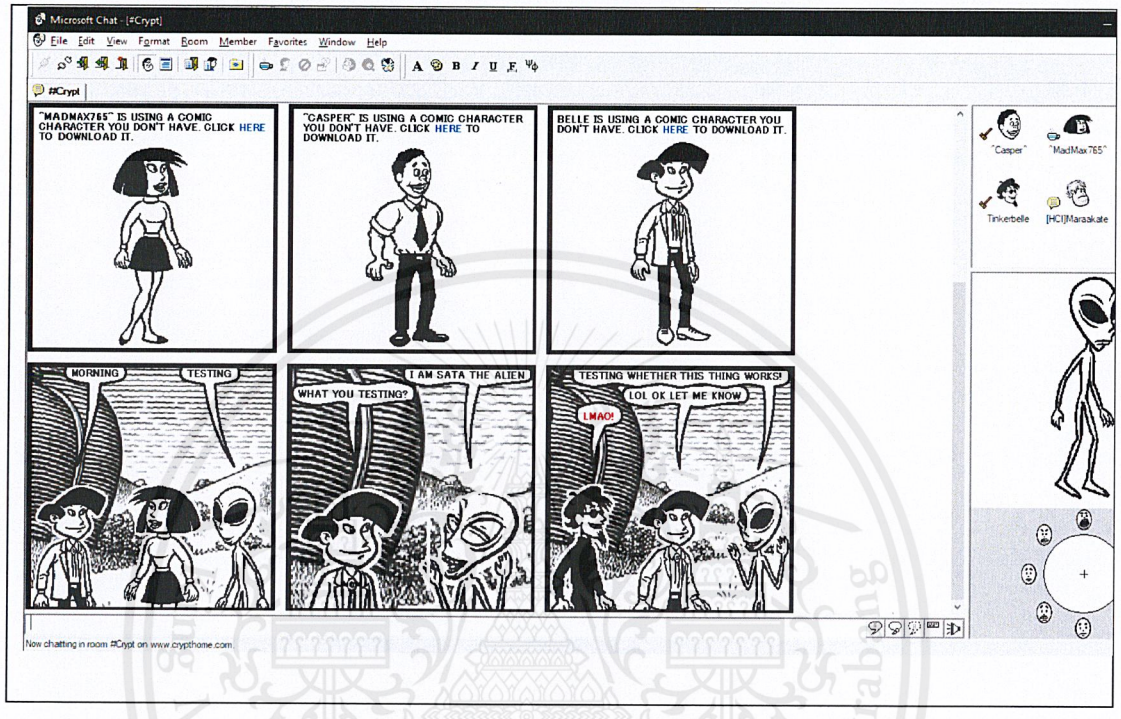

Figure 3 What Microsoft comic chat looks like

This program has many limitations. One such limitation is the difficulty of adding custom characters. To add a character, users must draw everything from scratch. They need to draw a total of 8 different postures, tied with facial expressions to be able to display their own personal avatar. Furthermore, because the postures are tied to the reactions, users have very limited choices of expressing themselves.

Project MangaKo aims to fill in these gaps by adding in a character creation system where the user does not need to draw anything. They simply need to choose what they want their character to look like. Our program will generate the facial expressions on its own.

2.4 MangaChat

MangaChat is a Chinese indie mobile application created by a team of developers called MangaDev. Their aim is to create something akin to a chat application that is image based like Project MangaKo, where the user can customize what kind of images are to be shown with their text, drawn in manga form.

Although this application wants to explore the concept of image-based chatting, it is still severely limited. The way that MangaChat came up with its core functionality is more like "Customizable Line stickers" than Project MangaKo is to Comic Chat. Users type their text then choose from a preset pool of user-submitted avatars with fixed expressions and poses. After that the program will merge the text with the user chosen avatar.

Unlike MangaKo and Microsoft Comic Chat, whose panels are automatically generated instead of manually chosen and in which unique characters can be created without the need to go through filtering, MangaChat is highly manual and offers little to no personal customization to users. All drawings must be approved and downloaded from their character template shop. MangaKo will be filling in a much different niche compared to MangaChat.

Chapter 3

Requirements Analysis

3.1 Functional software requirements

System allows users to create their own avatars	Functional
System allows users to choose their emotions and modify their avatars' facial expressions	Functional
System allows users to chat with another user by sending text messages	Functional
System allows chatting with more than two people in one room	Functional
System can automatically detect emotions from user's text	Functional
System can generate manga panels with each text sent	Functional
System allows users to create their own chat room and add friends to the chat room	Functional
System allows users to create their own account	Functional
System allows users to add friends to their friend list	Functional
System allows users to scroll through their history	Functional

Table 1 Functional software requirements

3.2 Non-functional software requirements

The client can run on an Android mobile device with at least 2 GB of RAM and 500 MB of storage space, and on an Android OS of version 7.0 or higher.	Usability
Server has the ability to keep at least 100 messages from users for a duration of at most 3 months	Usability
Server response time under normal conditions of regular internet connection, will take no longer than 3 seconds to response	Usability
Server response time will not exceed 5 seconds under normal conditions in the case where sentiment analysis is called	Usability
Server used must be compatible with a protocol that allows for usage across many platforms and can send pictures	Usability
The client must have the ability to retain user data such as auto logins.	Usability
Usage of application for an extended period of time will not slow down application and will retain response time of less than 3 seconds.	Usability
System character creation is extensible	Supportability
Sentiment analysis system must be interchangeable with newer technology	Supportability

Table 2 Non-functional software requirements

3.3 Use-case diagram

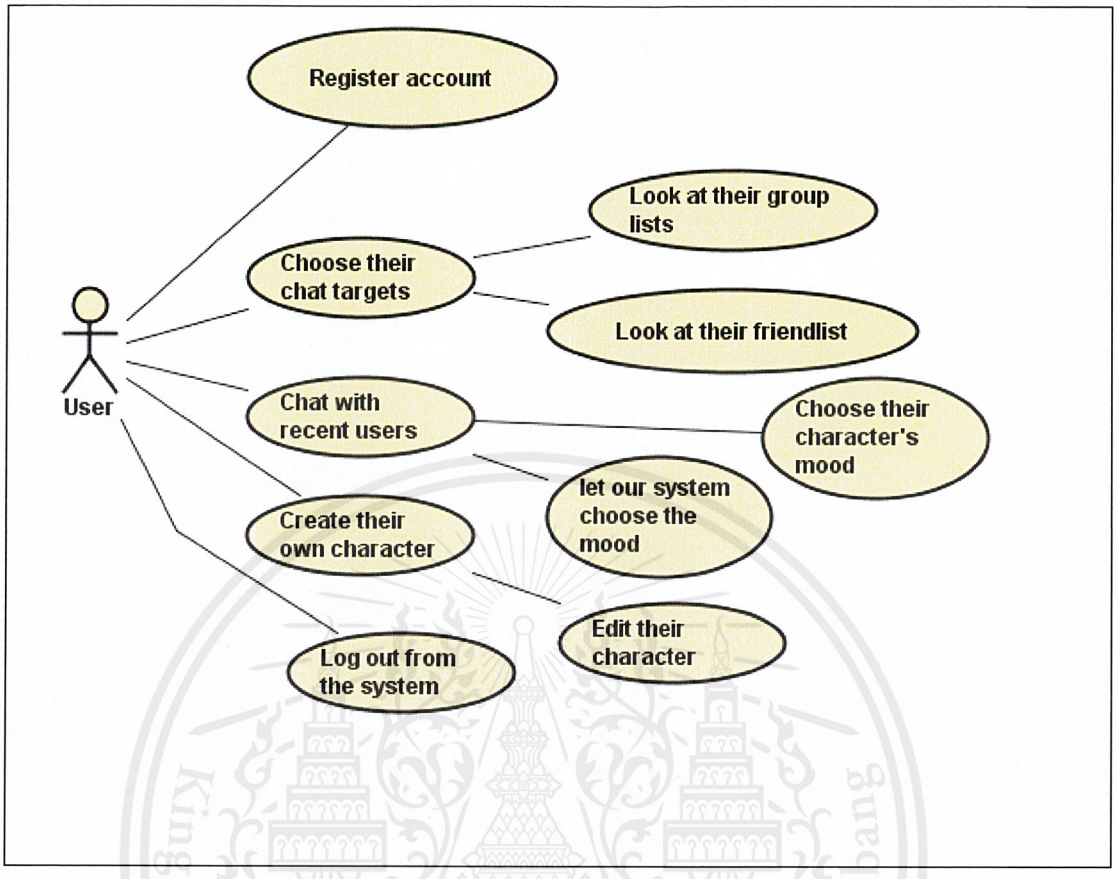

Figure 4 Use-case diagram

3.4 Character creation

This feature is one of the most important parts of our application as we want to allow users to be able to customize their experience as much as possible.

In this feature, users will be able to create their own avatars by choosing from an assortment of facial features and accessories. Once assembled they will be able to see their characters make different facial expressions despite being created from just one original base form.

3.5 Standard features of a chat program

Although MangaKo is meant to be an image-based chat application, it still needs to have all of the basic features that are expected to be in a typical chat application. Features such as group chats, individual chats, retrieving chat histories and friend management will be present in this application.

3.6 Manga chatting and emotion selection

This is the defining feature of our program, the ability for users to see their text portrayed as manga in real time. Users will be able to see the avatars they have created show up in manga panels for every text they type into the chat. These avatars will be able to make facial expressions that match the tone and overall feeling of the text sent. To this end, users are given two choices of whether they want to allow the system to decide the emotions for them or whether they want to decide the emotions on their own. This gives users more freedom over how they want their character to be portrayed and a very good way to allow things such as sarcasm and satire to be conveyed.

Chapter 4

Design and Planning

In this chapter we will be talking about our design of the software and how we plan on implementing the features

4.1 Background knowledge on tools used

4.1.1 Live2D

Live2D is an application created by Live2D incorporated. It was originally created for animators to help them animate characters with bones instead of drawing animations using the onion skin technique. Onion skin is a technique where the animators must draw their character frame by frame. The paper used to draw on top of the prior frame is usually thin to allow animators to see the frames before it as reference, hence the name "Onion skin". This technique can be very herculean and so, this application was born.

Live2D went on to become an application for changing 2D art into 3D/2.5D models. From the success of Live2D, the company went on to create other applications that specialize in converting 2D to 3D. From this came the Live2D API which allows Live2D models to be exported in a way that allows them to be used with other software such as Unity and Cocos2Dx, both of which are used to create games.

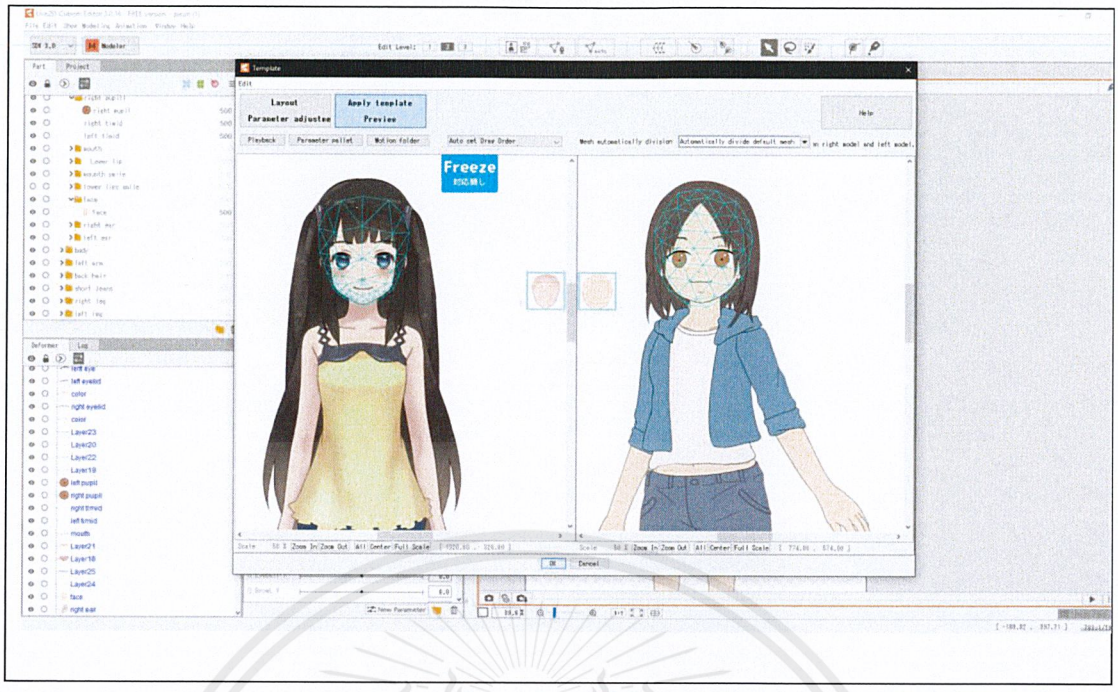

Figure 5 A screenshot of Live2D work space

As seen in the screenshot in Figure 5, the picture on the right is a plain PSD file with many different layers for each part of the character's body. By running it through Live2D's modeler, you can directly give these layers polygonal meshes. This allows users to create mesh based animations from nothing but the single picture that they draw.

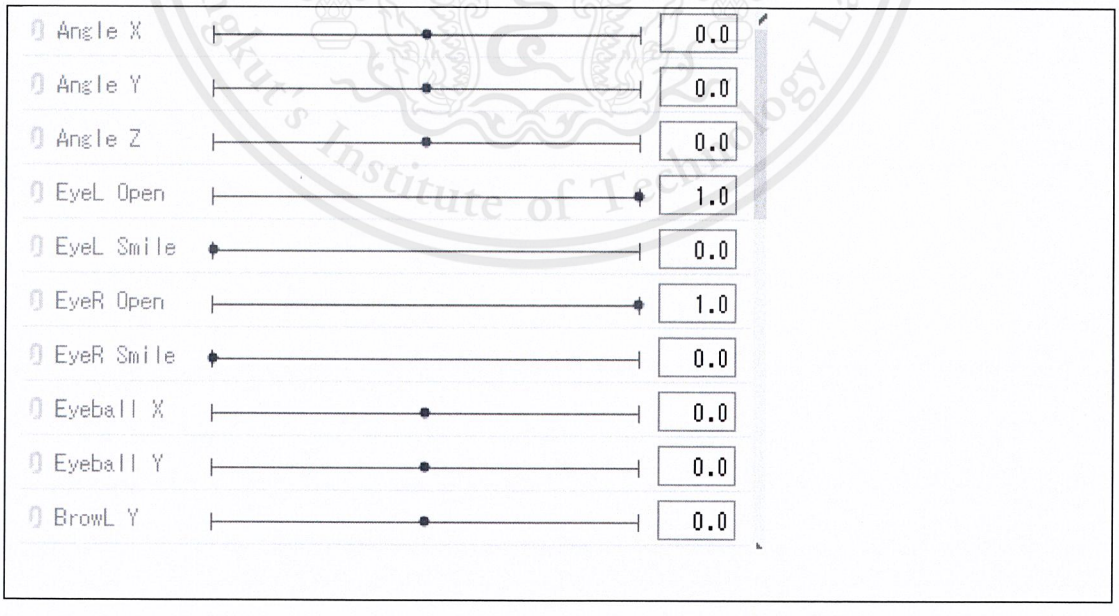

Figure 6 Live2D parameters

Live2D can apply parameters onto each Live2D parts, allowing these parts to be manipulated by a tangible value. You can assign a number to a state that the parts are in and whenever you desire to recreate that state, you simply need to change the parameter to that number, instead of having to recreate the state from scratch. By using this tool, you will be able to swap between different facial expressions and/or actions by calling on assigned numbers instead of recreating the scene and manually manipulating each pieces in the model. To put simply, a more competent variation of the keyframe animation method.

4.1.2 Unity

Unity is a game engine created by Unity Technologies. It is a free to use application when creating non-commercial applications but paid when used commercially.

Unity is a multipurpose and multiplatform game engine that supports both 2D game creation and 3D game creation. Unity's most amazing feature is in its simplicity. It is one of the most simple yet very powerful game engine in the market. Users can create games using its asset placement system without the need for knowledge of coding, while empowering users that can code to be able to make their creations even more complex.

Unity allows users to drag and drop their assets directly into their creation and takes care of linking many things for the users. Its flexibility and ease of keeping track makes its one of the best competitors to other engines such as Unreal engine, although the purpose and the processing power of these two engines are vastly different. Unity games use c# as a programming language. Everything in unity in put into scenes. A can have many scene in which in swaps out and load into the world space of unity's editor.

4.1.3 XMPP and EJabberd

XMPP protocol also known as the Extensible Messaging and Presence Protocol has the following architecture:

- Application Layer
- A client with a unique name that can communicate with another client with another unique name through an associate server

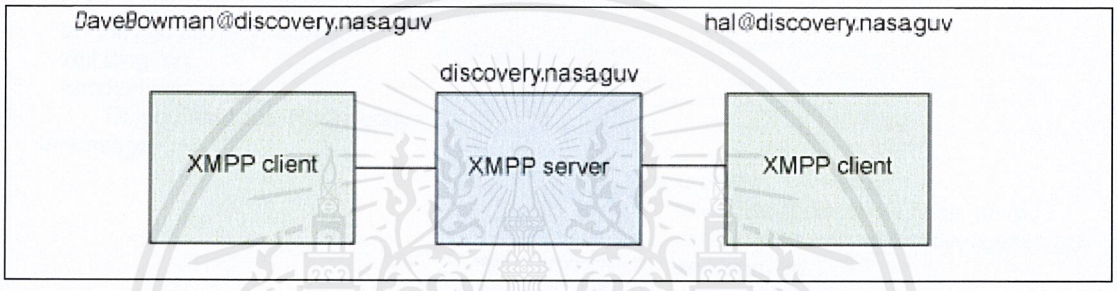

Figure 7 XMPP server architecture

XMPP is multiplatform and can be used to communicate between all sorts of devices. Most importantly it allows users to send text, images and files through the server.

4.2 Software architecture

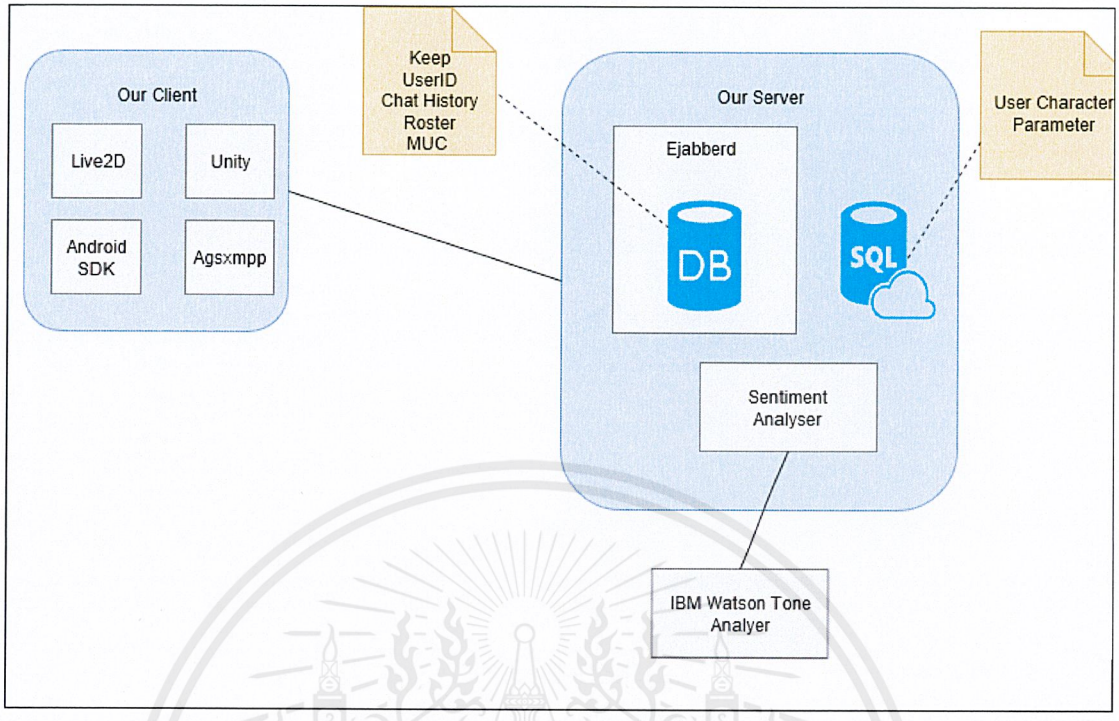

Figure 9 System architecture

For our project we will be creating 3D models for use with Unity using Live2D to facilitate and preserve the art style of the artists that create body parts for our character customization. Live2D will serve as the main method in our project for creating variations for the faces.

The chat program is created from Unity's UI element. This element runs at the front in the unity world space at all times once set to active. Instead of swapping scenes, we chose to go with swapping canvases instead in order to save both time and space of the application. Live2D models are loaded directly into the scene.

For our back end, we use a server called EbajjerD which is a ready to use server that uses the xmpp protocol. The back-end is connected to an webservice that employs the IBM Watson tone analyzer allowing us to send text to the API and allow us to get the emotions of the text.

4.3 Class design

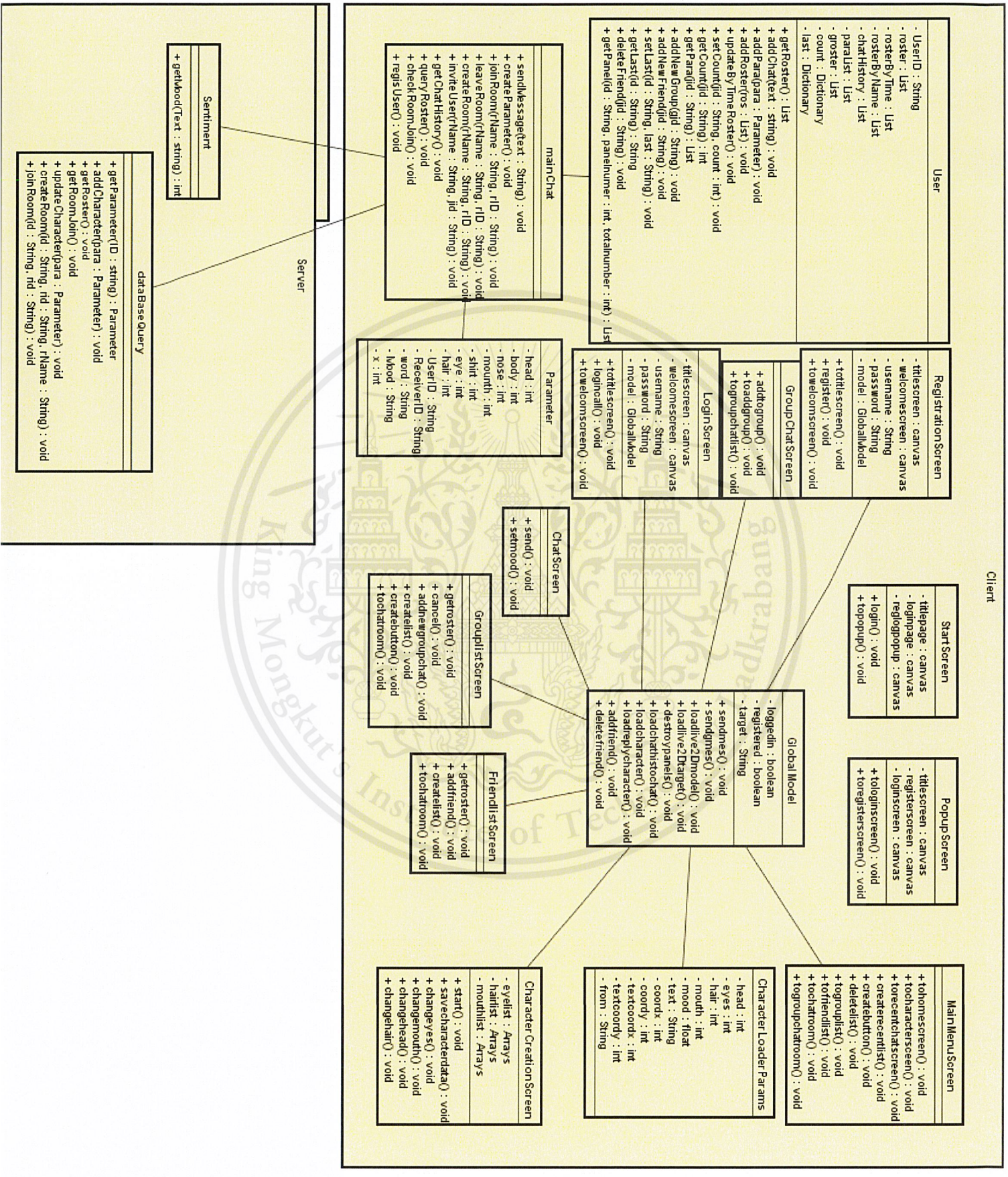

Figure 10 Class diagram

4.4 Database design

This is the database diagram of our project:

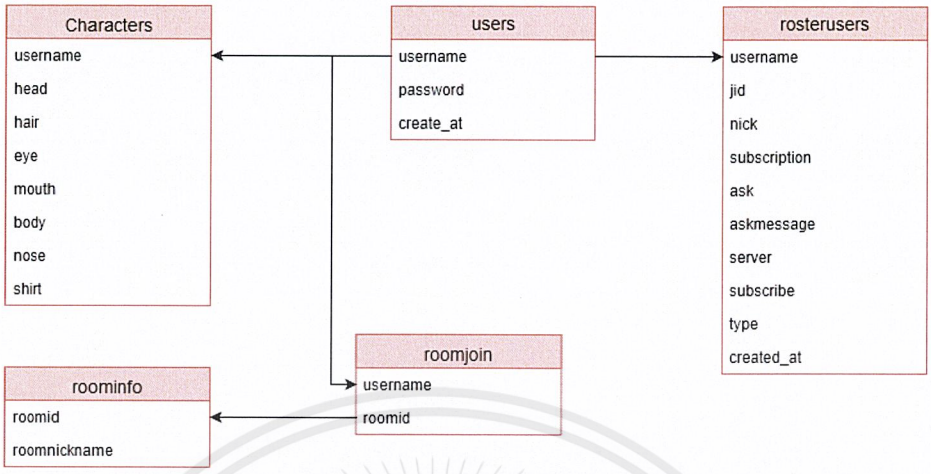

Figure 11 Database diagram

For our database we use a total of two different databases to keep our data. The `Users` table and `RosterUsers` table are in the database that is built-in on Ejabberd while the rest are in our own custom database. The `Users` table is used to keep the data of our users. `username` is user's username. `password` is user's password. `created_at` is the user's timestamp when the user is created. `RosterUsers` is used to keep each user's subscription (or friendlist). `Username` is user's username. `Jid` is contacted username. `Nick` is contact nickname. `Subscription` is status of user and contact user, if they are subscribed to each other, they will become friend. `Ask` is subscription request, 'S' for ask for subscription, 'U' for ask for unsubscription and 'N' for no request. `Askmessage` is a message that appear when user send subscription request. `Created_at` is timestamp for the first time two users contact each other. `Characters is` used to keep track of the user's character models. `Roomjoin` is the list of MUC that user has joined. `Username` is user's username. `Roomid` is the id of the MUC. `RoomInfo` is the info for each MUC. `Roomid` is MUC id. `Roomnickname` is the name of MUC that appear for user to see.

4.5 Development process

In our project there are a total of two team members. Due to this, the project has been split into two parts. One member will be taking care of the front-end components, while the other member will take care of the back-end components. Specifically, one member will be creating the client side application that will be used by the user, and the other member will be creating web services, servers and databases that will be used by the client side.

For this project we have planned for our development to be in three phases. The first phase is the pre-prototyping, the second phase is prototyping, and the final phase is finalization. The aim of the pre-prototyping phase is to create different features of the program as standalones as a proof of concept.

In this pre-prototyping phase, we clean up a lot of irrelevant or unobtainable goals from our project and partition our features into two separate categories, the first category being "Basic features" and the second being "Advanced features" which can be built on as an extension of our prototype. This is done by researching the extents of what our tools can allow us to achieve, in this case, by testing what Live2D can do on its own, what Unity can achieve and what tools such as Ejabberd can be used to achieve our specification.

After the completion of this pre-prototyping phase where we sort out what we want to put in our prototype, we enter our prototyping phase. The purpose of this phase is to create a working prototype of the program that contains the basic features that we have listed from our pre-prototyping phase. This part of the process involves finding the suitable methods of implementing each of our features and integrating the client with the back end.

After this phase is over and the features are implemented and integrated together into a working prototype, we would then enter the final phase, which is finalization. This phase involves the cleaning up of the messy code from the prototype and applying software design principles, proper renaming and refactoring of code. In this phase, the finalization of the UI occurs, and the application becomes more user friendly and presentable.

These are the three different phases of our program that we have planned and went with throughout our project life cycle.

Chapter 5

Software Development and Implementation

5.1 Features present in development

From the earlier mentioned development process, we have trimmed and selected these following features to appear in our final build.

1. Basic chat application functionalities
    a. Chatting in groups
    b. Chat histories
    c. Friend management
2. Character creation
3. Sentiment analysis
4. Displaying chat as manga panels

This segment of this document will be explaining how we accomplished the listed features.

5.2 Implementation of basic chat features

To implement the basic chat features that should exist in every chat application such as Facebook messenger or Line messenger, we chose to use Ejabberd as our server. We have chosen this server due to it being a ready-made server which allows us the spend lesser time on setting up a server and spend more time on each of the features that are available in our project. It also has its own database management system which is built-in and easy to use.

What it doesn't have however is the ability to retain chat history and group chat. It only allows for the usage of single chat rooms. To connect the client to the Ejabberd server, a Library called the Agsxmpp is employed to bridge the gap between the client side and the server side.

However, Agsxmpp too has a problem it which it doesn't allow the client to retrieve the chat history from the server. Therefore, we did the following the implement these features.

5.2.1 Implementation of the group chat rooms

By default, the Xmpp server will not keep Multi User Chatroom(MUC) as permanents. As such we need to configure our server to make these MUC permanent.

According the Ejabberd documentation, to enable the MUC and make it permanent, we need to enable and modify the mod_muc Module in the following way:

```
mod_muc:
      host: "conference.@HOST@"
      access:
           - allow
     access_admin:
          - allow: admin
     access_create: muc_create
     access_persistent: muc_create
     default_room_options:
           persistent: true
           allow_user_invites: true
           mam: true
           maxuser: 10
```

By adding this configuration to our server, we are able to make the Multiple User Chatrooms permanent and therefore implement our group chat room function. A fundamental function that needs to exist in all chat programs.

5.2.2 Chat histories

The Xmpp server does not keep a user's chat history after they have finished their session and closed their client. However, this is one of the most important part of a chat application and we have to configure our server to allow the usage of this function.

In order to configure our server to make the chat histories permanent, we need to enable the `mod_mam` module in the Ejabberd config file. The modification in the file is as follows:

```
mod_mam:
            default: roster
            assume_mam_usage: true
```

By adding the aforementioned to the config file, we are now able to save the user history for other sessions.

5.2.3 Retrieving the chat history to the client side

Now that our server allows for making chat history permanent, the problem that we had was using Agsxmpp to retrieve said chat history. Agsxmpp itself is simply a library used to connect between the client side and the server, it never had the functionality to retrieve chat histories.

We implemented this function by making an extension to the Agsxmpp library, we did so by referencing the xmpp protocol and writing out own request.

According to xmpp protocol to request we need to send XML to server as follow

```
<iq type= 'set' id='form1'>
  <query xmlns='urn:xmpp:mam:0'>
    <x xmlns='jabber:x:data' type='form'>
      <field type='hidden' var='FORM_TYPE'>
        <value>urn:xmpp:mam:0</value>
      </field>
      <field type='jid-single' var='with'/>
        <value>example@Jabberserver</value>
      <field type='text-single' var='start'/>
        <value>2018-01-01T00:00:00Z</value>
      <field type='text-single' var='end'/>
        <value>2018-12-01T00:00:00Z</value>
```

```
    </x>
  </query>
</iq>
```

From example above show the requesting of chat history from server that have example@jabber.server as sender or receiver, starting from 1 January 2018 at midnight until 1 December 2018 at midnight.

And server will answer back as:

```
<message id='aeb213' to='example@Jabberserver'>
  <result xmlns='urn:xmpp:mam:0' queryid='f27' id='28482-98726-
73623'>
    <forwarded xmlns='urn:xmpp:forward:0'>
      <message to=' example@Jabberserver'
         from='example2@Jabberserver'
         type='chat'>
         <body>Message<\body>
         <stanza-id xmlns='urn:xmpp:sid:0' by='example@Jabberserver'
id='28482-98726-73623' />
      </message>
    </forwarded>
  </result>
</message>
```

This is how we implement the basic functionalities of a chat application.

5.3 Character creation using Live2D

For the next feature in our project, the ability for users to create their own character, the following was done.

To make our character creation system, we used a tool called the Cubism SDK. This tool serves to allow Live2D models to appear onto Unity. The reason that we have chosen Live2D for this project is because of its ability to distort models according to set parameters instead of creating variations in images manually.

The very gist of our project is that we want to have users be able to create a single character and then use that character to make different facial expressions as they talk to other

users, without the need to make multitudes of face variation artwork, but a single model that can be articulated to reflect the mood of the user. Live2D allows use to control and create these distortions from a piece of artwork.

To achieve this effect of allowing users to choose any combination of the available pieces of their character, we create each parts of the characters as a separate Live2D model. Every Live2D model has a set of variables attached to them. Like strings to a puppet. We can manipulate the model to a certain state and assign a value to that state of the model. Then whenever we want to replicate that state, we simply need to manipulate the variables attached to the Live2D model to a certain value, and the state would be recreated, regardless of how each of the parts are shown on the screen. So instead of creating multiple Live2D models of whole characters with different combinations, we simply need to create multiple Live2D files of the different parts of a character instead, and let the users combine them, themselves.

Once the user is done creating their preferred character, a Unity object will check which Live2D models are active and then save them as a set.

For example:

A user wants to create a character with

1. Long hair
2. Lean face
3. Small nose
4. Small mouth

When a user chooses their parts from the list of choices, that part would be marked as active. These features have assigned values, long hair would correspond to the number 2, while lean face small nose and small mouth would correspond to the number 3. The Unity object will then check for which models are active and write the corresponding number of these models to an array. In this case the array would look like such: [2,3,3,3]. This is how we save the user's

We chose this API due to its ability to be accurate in this way as well as giving us numerical results that we can map to our Live2D model and make those models change according to the mood value. This is how we implement our sentiment analysis system.

5.5 Displaying chat as manga panels

The final part of this chapter will be dedicated in explaining the implementation of our "Manga chatting" system. The very core and the biggest appeal of our application.

This feature is the magnum opus of our application, the goal of it is to allow users to be able to chat together with the character that they have crafted from their reference. Once a user type something into the chat room, the system would make a call to the character database, and query for the user's character specifications. This data would come in the form of arrays that we have written about earlier. These numbers will then be sent to a character loader which will load the individual models according to the numbers retrieved from the database. The character loader will then put the Live2D models on to the Unity screen and assemble the user's character piece by piece at a coordinate that corresponds to whoever typed first in the chat. The person that typed first will have their character shown on the left side of the screen while the second one will be shown on the right side of the screen. A manga panel will then be loaded onto the screen to create the Manga theme that we have been aiming for. Speech bubbles will be loaded onto the vacant space, the user's inputted text would then be written onto to these bubbles finally completing the Manga look that we want to accomplish.

This is the default loading method of our system, this method however changes when we involve the mood system aka the sentiment analysis.

When the user types something into our chat box and sends it to another user, before the model is even loaded. The user input is immediately forwarded to the sentiment analysis webservice that we have created. The service will then give us a numerical result as to which mood was carried by the user's text. This number will then be searched through our system's dictionary which holds the definition of each of the mood, in the terms of Live2D model values.

For example, if the user's text carries the mood of being happy, the sentiment analysis would give out a number such as a 15, which means happy. The number 15 will then be used to search for which combination of Live2D values will transform the character's face to a happy face.

The number 15 could mean The eye model has to change its value to 20, the hair model has to change its value to 50 and the mouth model has to change its value to 300. This would result in the character's model being manipulated to form a happy face, instead of loading an art piece that is specific to that character, like a sprite sheet, which is inefficient and takes a lot more time and space.

Once the numbers are loaded, the character loader would then go through the aforementioned process and load the user's model onto the chat page as a manga panel.

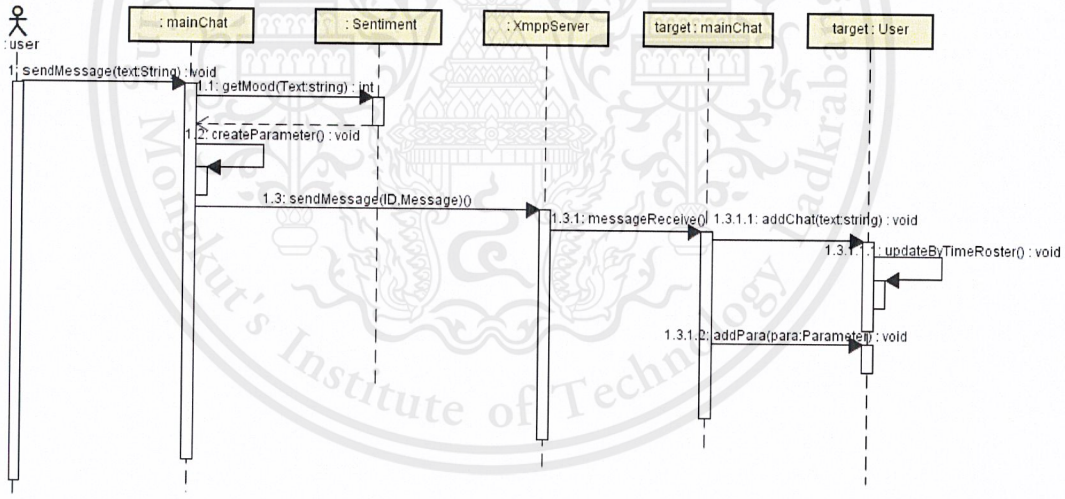

Figure 12 Sequence diagram of the MangaKo's chat feature

The end result of all of this would be a manga panel with the user's character speaking through a speech bubble, replicating the style of a manga.

Chapter 6

User Interface and Experience

The chapter will be explaining some of the things that gave us the idea of what kind of system interface to use as well as the experience that we want our user to have when using our system.

6.1 System interface inspirations

The UI of our system was inspired by three different applications that are currently available on the market. What we were looking for is “an interface that the user would want to press”, the kind of colorful interface that exists in mobile games would be adapted into our project.

6.1.1 Princess connect Re:dive

This game heavily uses the Criware system which is a substitute for Live2D but with more active components and at a much higher price. The UI of this game gave us pointers as to what we can do with the limited space that is the base of the Menu.

Figure 13 UI of the game Princess connect Re:dive

While it is hard to see with the static image, this game uses it animation component right at the very home page of the game with a moving art, which in our case will be a moving character. The buttons also jump up and down which creates a feeling of wanting to press it.

6.1.2 Mithra Sphere

The mobile game Mithra sphere gave us pointers as to how to create a comprehensive character edit/character creation system.

Figure 14 Mithra sphere's character edit UI

The game Mithra sphere is also a game where users have the ability to create their own character and bring them into combat, however they do not have the ability the change the facial expression of their characters. Their character edit system is extreme useful and easy to use, which is why we decided to draw inspirations from it.

6.1.3 Last period

The final game that we used as inspiration in our project is the Last period, also known as LasPeri. This game features a heavily cluttered UI which however, doesn't detract the user's attention but in fact makes the user more attentive to the game due to the amount of detail there are. The colors and the Icon buttons are all very effective at making the user stay with the application.

Figure 15 Last period's User interface

6.2 Our system's UI

This following section will be a show case of the designs of our UI, the UIs shown here will be a tad bit different from the final product due to the fact that there are still some improvements that can be made to improve the quality of life of the user. This is the Mock up of our program.

Figure 16 Mangako's start screen

The idea for this logo comes from the fact that Mangako's name is a play on the work Ko, which means mouth. The names holds the overall meaning as "A Manga that speaks" Or a "Manga chat".

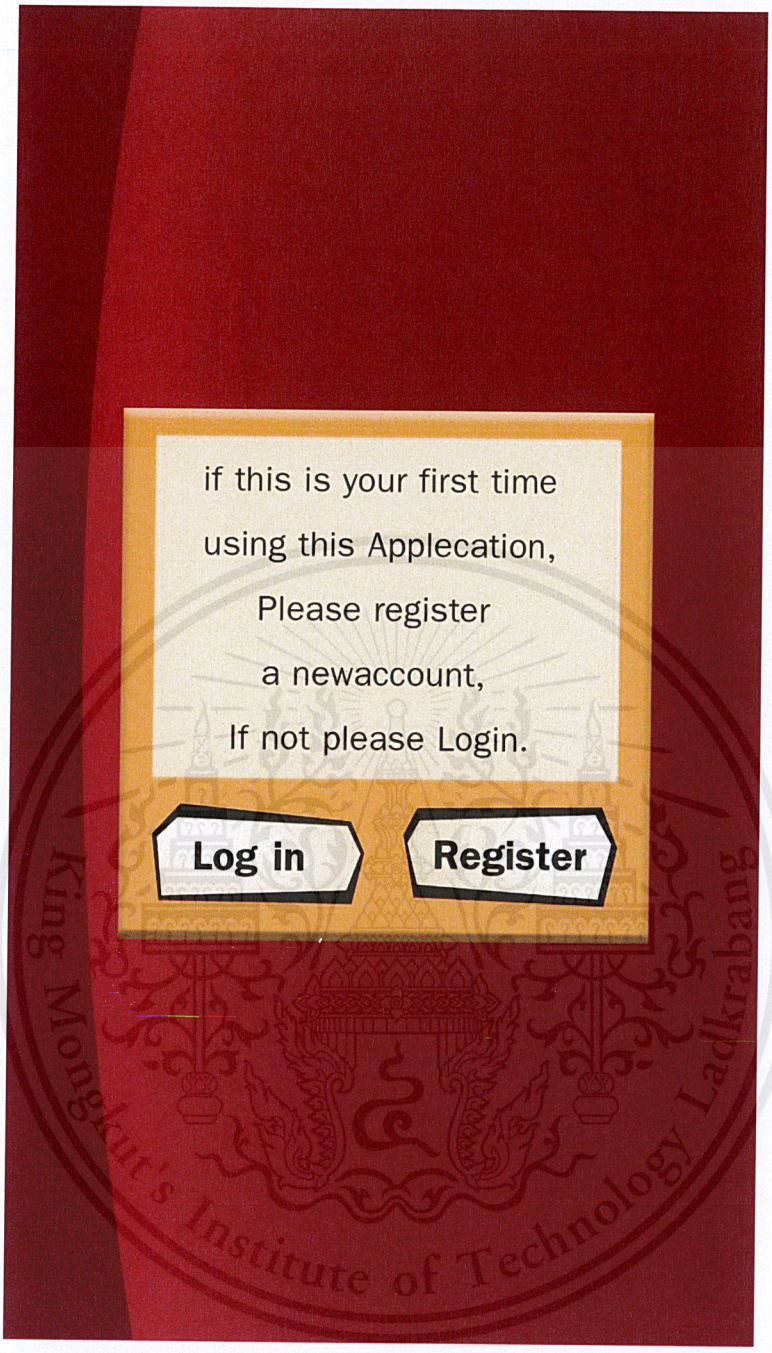

Figure 17 Generic modal dialogue box for user decisions

Figure 18 Mangako's Registration and Login screen

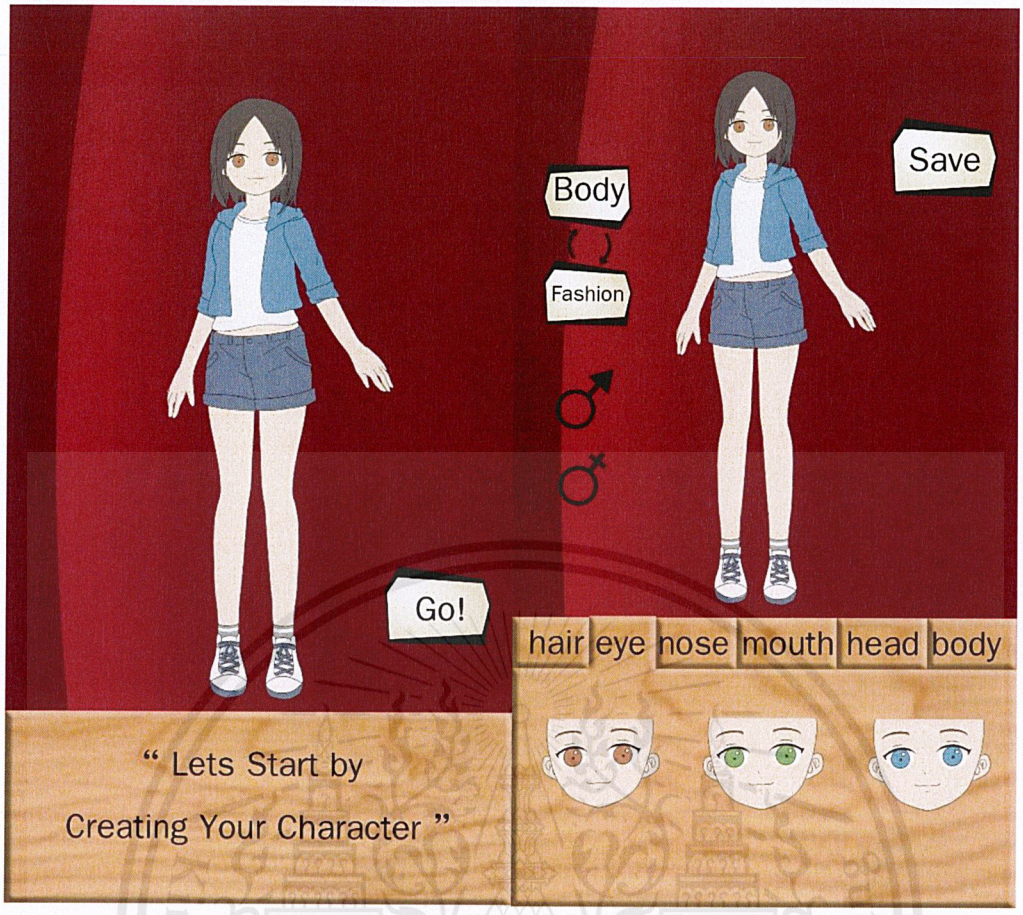

Figure 19 Welcome screen and Character creation

The influences of the game Mithra sphere can be spotted here in the character creation where the character's parts are displayed on a scrollable panel in which the user can easily switch between different kinds of eyes or in this case, the colors of the eyes.

Figure 20 Mangako's Main menu

This main menu is inspired by Princess Connect Re:dive's usage of the character mascot icon as well as Last period's heavily detailed background, which while a picture in this mock up will look different in the final product as it is a part that is still being drawn by the third party artist.

Figure 21 Mangako's recent chat list and chat room

The final part here shows the application's most important part which is the manga chatting. As one can see here, on the picture on the right, the chat room, the speech bubbles are a remnant of a certain manga style. The character's expressions change according to the text of the user, which is controlled through the sentiment analysis system.

Chapter 7

Project Result

7.1 Result

The result of this project is an application where the users are able to chat with their friends, and in groups, while having their characters show up in manga panels. Complete with an overhead sentiment analysis system which will automatically detect the mood of the user's text, and a system that uses the result from the sentiment analysis system to augment the user's Live2D model characters to create the desired facial expression.

We started off our project with these following goal points:

1. Create a core chat program with a single chat system and a group chat system where the users are able to read their chat history.
2. Have basic Manga panels be generated automatically when the user sends their text.
3. Allow users to choose their facial expression as well as allowing our system to pick their character's facial expression using sentiment analysis.

By the end of this project we were able to accomplish all our basic goal points, the goal points that signifies the purpose of our project. All of these features have been designed to be as lean as possible for we have checked

We were able to test what Unity can do when it is being used to design an application that I wasn't designed to make AKA chat programs instead of games and saw the usefulness of Live2D to create characters with multiple actions from just a single artwork.

When we started out with our project, we had no prior knowledge of Unity, back end development and Live2D. By the end of this project we were able to research and learn how to use each of these tools in our project and gained the ability to create an application that is practical and has the scalability to grow further.

Chapter 8

Conclusion and future work

8.1 Conclusion

In conclusion, this project turned out to be much more that we had anticipated. When we first started out, we had the simple goal of wanting to try out development using Live2D as well as Unity as these were the areas that we have never worked on before. This project gave us the opportunity to research into the system while trying to apply it to a real-world problem not being able to communicate effectively due to the inability of chats not being able to convey all of the missing nuances of a proper conversation.

We believe that we have managed to create an application that can act as a trailblazer for chat applications that carry more in a message than a simple text or image macro that cannot be changed by anyone unless they are able to draw and be in a creative position to create art work. Despite the problems with time manage and our inexperience on the subject at hand we were able to pull through to create a practical application that has most of the things we have envision when we first started out with our project.

8.2 Future works

For our future works we would like to be able to implement the missing advanced goal that we didn't have the time to accomplish which is:

1. Sending files and pictures to the other recipient which the characters acting accordingly to the sending action
2. Exporting the manga as books/chapters which can be shared accordingly.

There are also many other UI elements that we want to incorporate into our project.

We have created the application to allow easy addition of more parts that the user can choose to create their character with, and we certainly would love to add more parts to the pool and allow for even more unique experience for the users, as well as quality of life improvements.

References

[1] Wild, Matthew, and Kevin Smith. “Message Archive Management.” XMPP | About XMPP, XMPP Standards Foundation, 22 Feb. 2017, xmpp.org/extensions/xep-0313.html.

[2] ラストピリオド – 終わりなき螺旋の物語 –, lastperiod.happyelements.co.jp/.

[3] Paterson, Ian, et al. “Message Archiving.” XMPP | About XMPP, XMPP Standards Foundation, 15 Nov. 2017, xmpp.org/extensions/xep-0136.html.

[4] “SDK Tutorials Live2D Documentation.” TOP | Live2D Manuals & Tutorials, Live2D Creative Studio, 2 May 2018, docs.Live2D.com/en/cubism-sdk-tutorials/top/.

[5] Sifium. “Top Five Emotion / Sentiment Analysis APIs for Understanding User Sentiment Trends.” Medium, Augmenting Humanity, 20 May 2017, medium.com/@sifium/top-five-emotional-sentiment-analysis-apis-116cd8d42055.

[6] “Unity User Manual (2018.1).” Unity - Scripting API: Physics.Raycast, Unity Technologies, 30 Apr. 2018, docs.Unity3d.com/Manual/index.html.

[7] “プリンセスコネクト！Re:Dive (プリコネR) 公式サイト | Cygames.” プリンセスコネクト！Re:Dive (プリコネR), Cygames, priconne-redive.jp/.

[8] “ミトラスフィア -MITRASPHERE-.” ミトラスフィア -MITRASPHERE-, mitrasphere.jp/.