A smartphone application for recording and sharing moods

By

Ng Helen Hoi Ling 3035269629
Abstract

Emotional intelligence is the basis of people’s mental well-being. Recording and sharing our moods have a positive influence to us and those around us. This project creates a cross-platform mobile application which allows users to record and share their mood throughout the day. This report first discussed the objective of the project, then compared two frameworks Flutter and React Native. Flutter was chosen due to the app development experience. The backend of the project was also analysed and choose Firebase as the cloud platform. This report then discusses the implementation performed, including the user interface flow, design of cloud Firestore and features. Features incorporated are support and comment, gift store, sad warning, mood progress, emotion recommendation, Google Map integration and face emotion detection. Challenges were also encountered during the process such as server cold start and NoSQL database design. It is recommended that the Vision API and a more efficient NoSQL database design could be implemented furthermore for future works of this application.
# Table of Contents

Abstract  
Table of Contents  
List of Figures  
1. Introduction  
  1.1 Background  
  1.2 Current Status of Industry  
  1.3 Outline of Report  
2. Objective  
  2.1 Mobile Application Development  
    2.1.1 Cross Platform  
    2.1.2 Intuitive UI Design  
  2.2 Server Development  
    2.2.1 Authentication  
    2.2.2 Database  
    2.2.3 Data Processing  
    2.2.4 Endpoint Connection  
  2.3 Data Analysis  
  2.4 Attractive Features Implementation  
3. Methodology  
  3.1 Mobile App Development  
    3.1.1 React Native  
    3.1.2 Flutter  
    3.1.3 Selection & Justification  
  3.2 Server Development  
    3.2.1 Traditional Server  
    3.2.2 Cloud Services  
    3.2.3 Selection & Justification  
4. Implementation  
  4.1 Mobile Application  
    4.1.1 UI Flow
4.2 Firebase
   4.2.1 Firebase Authentication
   4.2.2 Firestore
   4.2.3 Cloud Functions
4.3 Features
   4.3.1 Friends & Posts
   4.3.2 Supports & Comments
   4.3.3 Very Unhappy Warning
   4.3.4 Mood Progress
   4.3.5 Gift Market
   4.3.6 Emotion Recommendation
   4.3.7 Friend Relationship Score
   4.3.8 Google Map Integration
   4.3.9 Photo Recognition

5. Conclusion
   5.1 Challenges
      5.1.1 Server Cold Start
      5.1.2 NoSQL Database Design
      5.1.3 Flutter’s Widget Structure
   5.2 Future Works
   5.3 Summary

References
List of Figures

FIGURE 1 Login View of the Mood App.................................................................14
FIGURE 2 Sign up view of the Mood app..............................................................15
FIGURE 3 Home View of the Mood App..............................................................15
FIGURE 4 Create new post..................................................................................16
FIGURE 5 Firebase Authentication Admin Webpage ..............................................17
FIGURE 6 Sample Email provided by the app after user registration.........................18
FIGURE 7 Database of the application................................................................19
FIGURE 8 Cloud Functions of the Mood App.........................................................20
FIGURE 9 UI design for Support and Comment buttons..........................................22
FIGURE 10 Notification for very unhappy warning................................................23
FIGURE 11 Requirement for unhappy warning notification.....................................23
FIGURE 12 Mood Chart of the app.......................................................................24
FIGURE 13 Profile page of the user which shows the gift stand..............................25
FIGURE 14 Table of gifts and points required.......................................................25
FIGURE 15 Add emotions section have mood tags for recommendation..................26
FIGURE 16 Visual Representation of the Frequency Scale of “Joy”..........................26
FIGURE 17 Visual Representation of the Frequency Scale of “Embarrassed”...........26
FIGURE 18 Map View of the Mood App..............................................................28
1. Introduction

1.1 Background

Emotional intelligence has always been an important part of everyone’s lives. It is the understanding and recognising of our own emotions. Emotional intelligence helps us in many ways, from looking after our physical and mental well-being, to the ability to inspire and lead. According to a study by Dr Travis Bradberry (2009), 58% of people’s job performance are determined by emotional intelligence. Emotional Intelligence is essential to building a balanced life.

The first step in building one’s emotional intelligence is to recognise his/her own emotion. In order to help people in understanding their moods, our team are building a smartphone application for users to record their own moods throughout the day. Recording one’s own moods can help users to understand the reason for their change of emotions and help them prevent future negative emotions.

1.2 Current Status of Industry

Recently, there is a popular trend on the internet which netizens post their emotion record throughout 2018. These netizens first rate their day into an emotion, then colour it onto the calendar. After a whole year, they could view how their emotions change throughout the year and how many days of a year do they feel happy or stressed as a form of statistics. The primary goal of these netizens is to build a habit that could help them identify their emotions. They would like to recognise their emotions then react to it in a healthy way.

Currently in the market, there is a mobile app called Daylio which serves a similar function. This app allows users to first choose one emotion out of five options, then choose the activity they have been doing prior to recording the emotion. This app has the function of monthly and yearly data representation. However, this app does not allow different users to interact with one another. Daylio only serves as an individual-diary type of app. Its main function is for users to record their moods only.

This is one of the reasons for our project to build an app that could allow different users to interact with each other. The current industry does not have a way for users to share their results therefore they post it onto other popular forums such as Reddit. Having a sharing function is
important as this could give users positive feedbacks from friends and this could help build a positive community in the future.

1.3 Outline of Report

This report first discusses the objective of the mobile application. The four main objectives of the project are mobile app development, server development, data analysis and implementation of attractive features. It will then compare two mobile app development frameworks and two server options for a more suitable combination. After that, implementations of this app are analysed. Lastly, the report concludes with the challenges encountered throughout the development and recommendations regarding future works are discussed.
2. Objective

This project aims to develop a cross-platform mobile application that records the user’s daily mood and provides a social platform for users to share their mood with their friends.

2.1 Mobile Application Development

A mobile application will be developed in this project. It will provide an interface for user to use our mood application. The followings are the key features or functions that we planned to implement:

2.1.1 Cross Platform

iOS and Android operating system nearly dominates the entire smartphone market. According to research, in 2018, iOS has 51.61% market share and Android has 47.93%, which they have a combination of 99.54% market share in Hong Kong. That means, if our app supports both platforms, majority smartphone users in Hong Kong can use our app.

2.1.2 Intuitive UI Design

UI design is a very important element in app development. Statistics showed that the quality of an UI design proportion to the satisfaction of user. In the other word, a good UI design would improve the user experience. It is especially important since our app is a social application, we shall design our app in a simple and intuitive manner.

2.2 Server Development

Backend is a very important part of any mobile application. For the server of this project, Firebase is chosen. The four objectives achieved are authentication, serves as a database, data processing and endpoint connection.

2.2.1 Authentication

Authentication is the part where users input their usernames and passwords to authenticate their identity. In order for a mobile application to secure users’ data and privacy, a server is needed to verify their logins. Besides, authentication can also ensure users to have the same personalised
experience across different devices. Knowing the user’s identity allows the app to feed data and interface that are personalised to that user, for example, his/her “newsfeed” or mood progress. The server implemented provides an authentication method to our app and achieves the objectives that the server shall provide, i.e. security and personalised experience.

2.2.2 Database
The second function of the server is to act as a database. The server provides database services to the mobile applications as a client-server model. Whenever the user requests data from the server, the server processes this request, such as handle tasks then send the requested data to the client device. For this project, the server handles data such as user information, friend list, mood data. When the user requests a certain information, the server chosen, Firebase, sends the requested data to the mobile device and it displays the information.

For Firebase, database for Flutter, there are two types of database provided. First, Cloud Firestore or second, Realtime Database. Realtime Database is Firebase’s original database. It is an efficient, low-latency solution for mobile apps that require synced states across clients in real time. On the other hand, cloud firestore is an improved version of Realtime Database with a more intuitive data model. Cloud Firestore is ultimately chosen, further explanation would be provided in section 4.2 when discussing Firebase in detail.

2.2.3 Data Processing
The third function of the server is data processing. Cloud Firestore allows clients to add, update, delete data and transactions or batch writes. Cloud Firestore also enable offline data. There are several ways to add or update data from Firestore. One of the solutions is to set the data to a document, the document identifier must be specified. Data can be updated by updating the document.

The server should also provide a way for users to retrieve data from the database for this project. In order to do that, queries were written to achieve that. There are two ways to retrieve data from Cloud Firestore, first by calling a method to get the data, or second, by setting a listener to receive data-change events.

The ideal server for this project should also have secure methods to manage the data. Security rules for Cloud Firestore for the mobile client libraries were written. Security Rules has an assumption that not all requests from the client are friendly. It means that the Security Rules will check and authenticate each requests before forwarding to the database. This can prevent any hackers to give out any unfriendly requests to the database in order to try to get data from the
database. The Security Rules provide access control and data validation to keep the users’ data safe.

2.2.4 Endpoint Connection
The fourth function of the server is to allow efficient and quick endpoint connection. Traditionally when a programmer updates the app’s function, the programmer needs to deploy a new version of the app. For this project, we would like a server that could allow real time connections for all device so when a new function is deployed, the function is updated automatically.

Cloud Functions for Firebase allows an automatic backend run whenever there are event triggers. These triggers are specified by the developer. Whenever the developer would like to write a new function, first they define the conditions for the function to be triggered in the code. Then when the developer deploys the function, Firebase connects to the selected event provider to invoke the function. The Cloud Function for Firebase has the benefit of zero maintenance and keeping the logic private and secure. The mobile app’s code is stored in the Google’s Cloud so there is no need to manage and scale the own server. On the other hand, as the cloud function is fully insulated from the client, the codes and logic is private so it is secure and would not be reverse engineered.

2.3 Data Analysis
The Mood App collects various data from the users, including user information, location and mood data. With these data, the mobile app can analyse the user’s behaviour and possibly predict it.

One of the objectives of this project is to implement functions that could utilize the data points. Some of the data points are collected through the user’s input when writing their posts. For every post, user’s current mood, mood intensity, location, mood keyword are collected. With the current mood and mood intensity, there is a mood chart in every user’s profile. The mood chart analyzes the user’s mood through the week or month. The user or their friends could view this mood chart to know how the user’s mood change. For location, there is a map in the app which indicates the locations of the data entry point. The user can view how he/she felt in a certain location and analyze whether certain locations are more prone to make him/her feel a certain way. For example, the user is always happy when he is doing exercise at the park. These data allows users to understand more about their mood and help them achieve a healthier, happier life.
2.4 Attractive Features Implementation

Another objective of this project is to implement attractive features to attract users to use our mobile app. Aside from the basic functions of a social media app, such as posts and comments, there are other special features implemented. It is important for the app to have attractive and special features other than basic ones because the special features give users a unique mobile experience that other apps in the market could not provide. According to the article on Forbes website (Shaoolian, 2017), the top one feature smartphone users are looking for is rich experience. The key is to offer an experience that is not offered elsewhere. Duplicative experience gives less incentive for users to download the app. The most important function is provide utility or value to the users. This project would like to enhance the efficiency that other similar similar mood app do not.

Attractive features for this project include depressed warning, gift shop, mood progress and friends score. Depressed warning is a warning sent to friends when a user is recorded to be sad for more than two days so their friends should check in on that user. The gift shop is a place for users to buy virtual gifts in the app that could be sent to his/her friends. The mood progress is chart which displays the change of mood of a user. And finally, the friend score is a feature that indicates which friend of the user is closest. This helps the priority and display of the friend’s posts in the feed. The details of each feature will be discussed in section 4.3.
3. Methodology

This chapter discusses the methodology adopted in this project to achieve the objectives listed in Section 2. Methodology used includes the platform for mobile app development and server development.

3.1 Mobile App Development

There are many choices for mobile app development framework. Each framework have its own characteristic and nature. Below is a comparison between React Native and Flutter.

3.1.1 React Native

React Native is a JavaScript framework for writing mobile applications for iOS and Android. It is based on React, a JavaScript library for building user interfaces. React-Native requires a packager to maintain the application. Libraries of React-Native are mature and it does have a large variety (since React and javascript libraries are also supported). There are also many choices of UI frameworks, including Material UI, that can be work with.

3.1.2 Flutter

Flutter is another cross platform app development framework which is developed by Google. It was released at 9th March 2018 and aimed to provide a organised and comprehensive app development experience. It uses Dart programming language and it has a build-in UI framework (Google’s Material UI).

3.1.3 Selection & Justification

After considering all of the requirements, we plan to use Flutter as our development framework. Our justification to choose Flutter over React-Native is we prefer a better development experience.

During development, React-Native requires a computer to run a NodeJs packager to maintain the app’s state, so when we test the app with the real machine, our machine need to connect, through network socket, to the hosting computer. Whenever there is update or state change, the packager will push the update to the testing device. It means that if we take our device away from the hosting computer network, the app cannot run.
Flutter uses another approach, the app packager is running in the device, same for both development and production environment. When there are hot updates, we need to connect the device to the computer (physically) to update the packager in the device. Flutter will compare and only do the necessary update to the packager. Once it is done, we can disconnect the device from the computer and test our app wherever we want.

As our app will be distributed during the development stage for testing purpose, so we prefer Flutter over React-Native.

3.2 Server Development

There are lot of choices of backend solution. Traditional servers that require lots of setup and configuration in a machine, cloud service appears to be a better approach as it has a lower maintenance and set up time.

3.2.1 Traditional Server

Traditional hosting has two forms, dedicated and shared. Dedicated hosting is where a company pays the service provider for the complete resources of one or more servers. The client has full control over the server resources. As for shared hosting, the server’s resources are shared by a number of websites. It is more cost efficient and low maintenance than dedicated hosting.

On the development point of view, a developer needs to set up their own server. The developer would then deploy the app to the backend server. In order to access codes and data from the server, the mobile app uses an API or library to communicate with that server.

3.2.2 Cloud Services

Cloud service is a service made available to users on demand via the internet from a cloud computing provider’s servers instead of a traditional server. Cloud services are more dynamic and scalable than the traditional servers. Besides, the load of cloud services are balanced among several server. The data are mirrored across different servers.

On the development point of view, a developer does not need to set up its own dedicated server. Instead, the developer could user cloud services provided by companies such as Google’s Firestore.
3.2.3 Selection & Justification

Cloud services are chosen to be our server. Specifically, Firebase as our backend. Firebase is a mobile and web development server powered by Google. It is a cloud platform and it have some general backend tools, for instance, authentication module, real-time database, hosting, and data processor.

The main reason that a cloud platform is chosen over a tradition server is the development time. Compare to traditional server, cloud platform does not require the set up on hosting, network configuration and security. This save lots of time and the team can focus more onto the development of features.

Another reason is the pricing. Cloud platform usually operate in a ‘pay as you go’ format. Google integrated the payment of Firebase to the Google Cloud Platform payment, and the Google Cloud Platform has 300 USD free credits for the users in first year. This fits our need and hence we can use this industrial standard service in this project.

The third reason is that Firebase has a good integration with Flutter. In their documents, there are instruction of integrating Firebase function to Flutter app. The community are also supportive and there are thousands of developer are currently contributing to it. By considering the community support is very important since our team is not a expert in app development. We need a supportive community to solve the questions we faced during the development process.
4. Implementation

4.1 Mobile Application

4.1.1 UI Flow

Figure 1. Login View of the Mood App

**Layout:** Figure 1 shows the design of *Login view.* This view appears when users launch our app. There are intuitive text input fields for users to sign up with their email.

**Flow:** When ‘sign up now!’ button is pressed, our app will allow users to sign up with their email.
Figure 2. Sign up view of the Mood app.

**Layout:** Figure 2 shows the design of *sign up view*. This view allows users to create an account by filling in the sign up form. Similar to *login view*, we implemented intuitive text input field for the sign up form.

**Flow:** When ‘Sign Up’ button is pressed, sign up form will be sent to our server. If the form is valid, the user is registered and will redirect to *home view*. 
Figure 3. Home View of the Mood App.

**Layout:** Figure 3 shows the design of home view. This is the mood feed. Users will see their friends’ mood posts here. Users can ‘support’ or comment on their friend’s post.

**Flow:** When the ‘Support’ button is pressed, the user will send a like to the person who posted. When ‘Comment’ button is pressed, the user can write a comment and interact with the person who posted.

Figure 4 shows the interface for users to create new posts. Users input their current mood, the intensity of the mood and add mood tags and descriptions.

**Flow:** When the user press or drag their finger on the mood intensity bar, the background colour of the page changes. The example shown on Figure 4 represents a very happy and high intensity mood.
4.2 Firebase

4.2.1 Firebase Authentication

Firebase makes authentication easy for end users. For this project, the user’s identity is required so the app can provide a personal experience and it keeps their data secure. Firebase allows different third-party providers to authenticate. The developer can build their own interface or use Google’s open source UI which is also fully customizable.

Once the user login, the user’s information is returned to the device via callbacks. This allows developer to personalize their app experience. Among the data returned, there is a unique user ID (see Figure 5) which is unique for every user but same across all providers such as Facebook or Twitter. This unique user ID identifies what part of the backend system that user is authorised to access. Firebase also record user sessions so the user can return to their session after they login back to the app.

For this project, the users sign in using the Firebase Authentication SDK. After selecting the authorization method, email, to provide to the users, the UI is set up to allow users to input their login credentials. After receiving the user input, the user’s email address and password is passed to the Firebase Authentication SDK to verify their credentials. If the login is successful, the user will be directed to their own profile page to start using the app.

Although Firebase supports third-party authentication, we did not use any because it creates unique accounts per login method. For example, a user uses his Facebook account as login, this creates a user ID that indicates this account. Then, if he uses the same email address of the Facebook login, but register another account through the email login method, there will be a second account even though the email address is the same in both cases. Therefore, although Firebase provides an easy authentication for third-party access, our group decided not to include it.
Figure 5. Firebase Authentication Admin Webpage.

Figure 6. Sample Email provided by the app after user registration.
4.2.2 Firestore

Firestore is a cloud database for mobile, web and server development from Firebase. It allows synchronization of data across different devices. It also allows offline support for the app. There are several benefits of using Firestore, including flexibility, expressive querying and realtime updates.

Cloud Firestore is cloud-hosted and NoSQL database for both iOS and Android and it can be accessed via native SDKs. With the NoSQL data model, the data are stored in documents that contains field mapping to values. The documents are then stored in collections for organization and building queries. Common data types are supported such as strings, numbers to nested objects. Firestore is flexible as it supports different data structures. For this project, the data are structured according to several categories, comments, posts, subscriptions, relationships etc (See Figure 7). The data are first stored in a document, then in a collection.

![Figure 7. Database of the application.](image)

Several NoSQL database design are considered for a faster and more efficient querying. For this project, the data are stored in several trees according to their functionality instead of a deep tree...
which contains everything. It is discovered that a more shallow tree with more children is much faster and costs less than a tree with many layers. For example, the posts are stored as a separate tree instead of a subtree under users. It is quicker to search for a post in thousands of posts in the post document than 3 posts in the users collection.

Another reason for this design is cost. Firestore charges developers for each query in the database. Cloud Firestore charges primarily according to the number of document read, writes and deletes performed instead of the amount of data stored. Although there is 50,000 free quota for document reads per day, it is better to consider the pricing for future scalable uses.

4.2.3 Cloud Functions

Cloud Functions is a feature provided by the Cloud Firestore for developers to run backend code in the mobile app. Cloud functions responds to events triggered by Firebase features and HTTPS requests. The code of the app are stored in Google’s cloud. After deploying a function, Google’s servers manage the function and respond to the load by scaling the number of virtual servers needed to run the function.

The lifecycle of a cloud function begins with the developer writing the code and defining the conditions which the function should be triggered. The function is then deployed and Firebase connects it to the selected event trigger, Cloud Firebase. If the function’s condition is met, the code will be evoked and instances are created to handle the events. After the function is finished and idle, the instances are deleted.

For this project, several cloud functions were written to provide functions to the app, including onNewPostCreated, loadUserPosts, loadUserMoods etc. The cloud functions were written in Node.js and most of them are triggered by cloud HTTPS. Function for new user created are triggered by Firebase Authentication while function for new relationship notification is triggered by cloud Firestore to update the document. Others are triggered and retrieved via cloud HTTPS requests (see Figure 8). An advantage of using cloud function is that it minimizes the actions needed to retrieve data. With Stream Builder, a single action can retrieve username, posts, comments, mood data etc which is very efficient.
4.3 Features

4.3.1 Friends & Posts

One of the basic features of any social media app is to add friends and to create posts. For this project, these two features were implemented. After login, the user can add friends by first searching for their friend’s username in the search bar. The user can then click on the add friend button which sends a friend request to the other user. The other user receives the add friend notification and decides whether to accept or decline this friend request. This add friend feature is supported by cloud functions. A cloud HTTP request is triggered and sent after clicking on the add friend button. When the other user accepts the friend request, the cloud Firestore is triggered to update and write the relationship document to record a new relationship between two users.

For the documentation and record of the relationships in Firebase, two relationship data will be created for any two users to become friends. For example, Alice and Betty became friends on the
app. One relationship document for Alice and one for Betty are created. The reason to create two documents for one relationship is because of the friend score. Friend score is a score that indicates how close a user is with another user. When a user interacts with another user on the app, the score increases. We record this interaction as a one-way relationship because there may be situations where interactions are one-way. For instance, Alice is very supportive of Betty. Alice keeps commenting and supporting Betty’s posts, but Betty do not support nor comment of Alice’s posts. More details about the score system is discussed in Section 4.3.7.

For new posts, after login, the user clicks on the “What’s in your mind?” bar of the newsfeed page for writing a new post. Data include current mood, mood intensity, description and mood tags are inputted. Mood is required and other descriptions are optional. After finalising the post, the user publishes the post and a cloud function is triggered to create this new post. The post will then appear on the user’s friends’ feeds which they could support and comment.

For the priority of the newsfeed, the posts would be sorted by time and friend score (Details in Section 4.3.7). For posts that are created in less than an hour, the posts will be at the top. Then, the posts are sorted in 1-3 hours, 3-12 hours, 12-24 hours and others from the highest to lowest priority. Then in each category, the posts are sorted by the friend score.

4.3.2 Supports & Comments

Another basic features for any social media is Like and Comment. For this project, Like is named as Support because it is not appropriate for a user to another user’s unhappy posts. In the newsfeed page, there are many posts are the user’s friends. The UI design is intuitive including buttons for Support and Comments and we take reference from Facebook (see Figure 9). Cloud functions were implemented for support and comments, which writes and updates the documents in the cloud Firestore. The original user who posts the post receives notifications for Supports and Comments and can view them.

Figure 9. UI design for Support and Comment buttons
4.3.3 Very Unhappy Warning

Mood data are collected through the new posts created by the user. One of the goal of this project is to create a supportive environment for the users and a place for users to release their negative emotions. When a user is recorded to be sad continuously for a while, a sad warning will be triggered and sent to the user’s friends (Figure 10). This feature helps users to better cope with their emotions and create a support system for friend groups.

The requirement for getting a sad warning is to achieve 25 points according to the table in Figure 11. The reason for the large difference in the first and second unhappy activity is to avoid spamming of posts. Our group would like to make sure that this user is sad for a period of time instead of just a few hours. Some users may spam the newsfeed within an hour because they are frustrated in a short period of time. This score will be reset after a period of time so it does not accumulate overtime.

![Figure 10. Notification for very unhappy warning.](image)

<table>
<thead>
<tr>
<th>Activity</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>First very unhappy activity</td>
<td>+10</td>
</tr>
<tr>
<td>Second very unhappy activity</td>
<td>+3</td>
</tr>
<tr>
<td>Starting from third very unhappy activity</td>
<td>+1</td>
</tr>
<tr>
<td>Unhappy / Neutral activity</td>
<td>0</td>
</tr>
<tr>
<td>Happy / Very Happy activity</td>
<td>-5</td>
</tr>
<tr>
<td>Previous day have no unhappy or very unhappy activity</td>
<td>-3</td>
</tr>
</tbody>
</table>

![Figure 11. Requirement for unhappy warning notification.](image)
4.3.4 Mood Progress

After collecting mood data from the user, the mood data are represented in a way so users can better understand their mood throughout time. The general representation of mood can be presentation in an x-axis, where the level of mood represents the intensity.

The collected mood data are then combined into a mood chart (Figure 12). The x-axis of the chart is time and y-axis is the mood. Currently, the mood is represented in a respective colour and number on the chart. Value of 0 and red colour represents very sad while value of 4 and yellow colour represents very happy. The mood chart is still under construction and the scale of time in the y-axis should be calibrated. The data on the recent few days are smashed together. This mood chart tracks the user’s mood throughout the dates and helps user to identify and control their moods.

![Mood Chart](image)

Figure 12. Mood Chart of the app.

4.3.5 Gift Market

Sending gifts to friends is a gesture of gratitude. People should appreciate their friends more. The gift market is for users to buy virtual gifts and send it to their friends. A point system is set up as below for users to save points then use those points to purchase virtual gifts (Figure 14). It is similar to the experience points a user gains when he performs some actions in the app. The actions are login into the account, create new posts, comment on friends’ post and add a new friend. When a user receives the gift, the present will appear on his/her present stand in the profile (Figure 13). The user can also choose to create a post about this gift and share it with their friends.
4.3.6 Emotion Recommendation

When the users are creating a new post, in the “Add emotions” section, there are some mood tags for them to choose from (Figure 15). The mood tags are emotion recommendation generated according to the current mood and mood intensity.

A library of mood tags were first created. For each mood tag, it has its own scale from unhappy to happy. This scale has values which represent the frequency of this mood tag being used in this mood. The more frequent this mood tag is used, the higher the value. For example, for “joy” this mood tag, it is more often used with a happy emotion. The value for happy in this frequency scale is higher (See Figure 16). Another example for a more neutral emotion is embarrassed (See Figure 17). This emotion is neither happy nor sad. The scale of this emotion would have higher frequency scattered across instead of concentrating on a certain point.
Figure 15. Add emotions section have mood tags for recommendation

![New Mood Post](image)

Figure 16. Visual Representation of the Frequency Scale of “Joy”.

![Frequency of "Joy" used in different moods](image)

Figure 17. Visual Representation of the Frequency Scale of “Embarrassed”.

![Frequency of "Embarrassed" used in different moods](image)
4.3.7 Friend Relationship Score

The friend score system is a hidden score calculated between two users. When the score is higher, this indicates they have a closer relationship and their posts will have a higher priority to appear first in the newsfeed.

The friendship score is calculated by monitoring the actions a user acts against his/her friends. When a user clicks into other user’s profile page, support or comment on a friend’s post, this increased the relationship score temporarily. After a certain period of time, this score will be decreased back to the original score until some actions trigger the increase of the score again. Furthermore, an emoji will appear next to the username when the users have the highest relationship score among their friends. This indicates a close relationship which the users can view as well.

Another function of the hidden score system is to prioritize the newsfeed so the most related user’s posts will appear further on top. This hidden score system is similar to the Instagram’s algorithm. Since 2016, Instagram ditched the reverse chronological feed and developed an algorithm to prioritize the feed so users will be more engaged in the app. According to Constine(2018), the three main criteria for their algorithm are interests, recency and relationship. Interest is the prediction made by Instagram estimating how much the user care about a post, a higher interest gives a higher ranking to the post. Recency is the time that the post is posted, a more recent post gives a higher ranking. As for relationship, the relationship score is calculated in proportion to the interactions such as like and comment which our project adopts as the second main factor. The first factor for prioritizing the feed is recency which is reverse chronology.

4.3.8 Google Map Integration

Google Map is integrated in the app to view the locations of the different moods. It is integrated by importing the google_map.dart package. After the user input their moods, their locations are recorded by the app and a marker is created on the location of the map view (Figure 18).

The Map View allows users to add a floating button on the map with a description and mood intensity. For instance, in Figure 18, a happy mood was recorded and the description is indicated in a window text container over the floating button. This map view helps users to record their moods in different locations around Hong Kong.
4.3.9 Photo Recognition

Vision API by Google is used to analyze photos uploaded by users. Their face is first detected then emotion expressed by the face is analyzed. The Natural Language API is also developed by Google Cloud Platform. It analyses the text description and find the emotion represented by it.

The main implementation of these two technologies is by comparing the two emotions detected by the Vision API and Natural Language API. After comparing the two, a match score is shown to the user. A higher match score indicates that the user’s face emotion matches the description of the post.

Normally, a machine learning model requires a number of data to train in order to have a reliable result. In our situation, we are not having a reliable data source (i.e. brunch of human faces with different emotion or text phrase that representing any kinds of mood) for us to train a model, so we prefer to use a pre-train machine learning model, which are the Vision API and natural Language API. In fact, both of the API can customize or re-train by our own data, however in this project, we remain to use the default models, which the result are satisfying after certain tests.
5. Conclusion

5.1 Challenges
Several challenges were encountered while building this app, the three main challenges are server cold start, NoSQL database design and Flutter’s Widget Structure.

5.1.1 Server Cold Start
Server cold start is the long response time when the virtual server needs to reboot. The cold start time are fluctuated in accordance to the virtual server boot times. If the virtual server running the app is awake, the functions are triggered immediately. But if the server needs to be start up from limbo, more time is needed to initiate the functions of the app.

Cold start affects the run time performance unpredictably. For example, a simple hello world function can take 3ms or 100ms to complete. These fluctuations are an unavoidable aspect of cloud functions from any provider. It is the nature of cloud servers and can only be minimized by caching the data or use hacks to keep the function warm.

5.1.2 NoSQL Database Design
Relational Database Management System(RDBMS) are mature and adopted by many enterprises. It is stable and richly functional, many features are well implemented and supported. However for NoSQL, it is more difficult to design the database for more efficient querying. NoSQL databases drastically differ from RDBMS depending on the type of NoSQL database. There are no universal best practices or principles for designing the database schema. This creates the challenges in designing the best practice for the respective NoSQL database type.

For this project, document database are adopted. Several factors were considered when designing the database to retrieve items efficiently, collection size limits, available query types and indexing supports. It is challenging for first time users of NoSQL to design a comprehensive and efficient database.

5.1.3 Flutter’s Widget Structure
Widgets are the building block of Flutter. Everything from a container to a button is a widget. One of the challenges of using Flutter widgets effectively is designing the lowest level of widget to be reusable. For example, a widget for the container of a post would be reused a lot in the app. It is important to design the widgets in way so they can be reused and no need to rewrite it everytime.
Another challenge for the Flutter widgets is how to effectively make use of the StreamBuilders for effective data retrieval. Apps are highly asynchronous. Dart streams supports the asynchronous of data. StreamBuilder is a widget that builds itself on the latest snapshot of interaction with the Dart stream. The challenge for this project is to design which data are needed to be “listened” in this stream. If a large amount of data are listened, the network data flow is large. On the other hand, if some required data are not listened, the app is not asynchronous to the latest version. It is a challenge for the team to design which data are needed for the stream. For this project, several streams were built including authentication, user profile and notifications.

5.2 Future Works

Recommendation for future works of this project is to design a better NoSQL database for this app to scale up. The current cloud Firestore is suitable for a smaller amount of users. However, if the number of users grow and there are more data stored in the Firestore, a more efficient way of querying is needed.

The photo recognition of the project uses the Google Vision API. It is currently an auxiliary tool in the project which compares the emotion detected of the face in the photo and text. The Google Vision API is a powerful tool which can also detect entities. Further development on integration of detecting entities can better help the app to detect emotions. For instance, a background with the sea or a sunny day could imply a happy emotion.

5.3 Summary

In this project, a cross-platform app for recording and sharing emotions is developed. It is a social media app which users can create posts about their current moods and share it with their friends. Flutter is chosen as the mobile app development platform and Firebase is the backend of the app. Cloud Firestore is the cloud database for this app and cloud functions were implemented to read, write and update documents that store information. Attractive features were also implemented, for instance, emotion recommendation and Google Map integration. Vision API is also adopted for photo emotion detection. Other attractive features include gift shop for users to buy and send virtual gifts to their friends and sad warning which notifications are sent to friends when a user is unhappy for a long period of time. However, this app is not perfect and there are ways to improve it. A better NoSQL database could be designed and further application of the Google’s Vision API could help improve the emotion detection feature for future uses.
References