COMP4801 Final Year Project: First Deliverable

A smartphone application for recording and sharing moods

By

Ng Helen Hoi Ling 3035269629
Wong Tin Chun Jeremy 3035298151
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, choosing Firebase as the cloud platform. Machine learning were also implemented in the project by Vision and natural Language API developed by Google Cloud Platform. By using the technologies discussed, the application is able to provide a social media platform for users to share their moods and interact with one another. Further testing on the machine learning analysis will be done. The next phase of the project is to implement all functions of the application.
# Table of Contents

Abstract  2

Table of Contents  3

1. Introduction  4
   1.1 Background  4
   1.2 Current Status of Industry  4
   1.3 Outline of Report  4

2. Objective  5
   2.1 Mobile Application Development  5
   2.2 Server Development  5

3. Methodology  7
   3.1 Mobile App Development  7
   3.2 Backend Development  8
   3.3 Machine Learning  9

4. Current Progress  11
   4.1 Design works  11
   4.2 Implementations  18

5. Conclusion  19
   5.1 Challenges  19
   5.2 Future Plan  20
   5.3 Summary  20
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. Then it will discuss the methodology used and justification of the methods chosen. Lastly, the report concludes with the current work in progress of our team.
2. Objective

This project aimed to develop a mobile application that records users’ daily mood and provides a social platform for users’ to share their mood with their friends. Machine learning technology will also be used in the mood detection feature.

2.1 Mobile Application Development
A mobile application will be developed in this project. It will provide an interface for user to use our matching application. The followings are the key features that we planned to implement:

2.1.1 Cross Platform
iOS and Android operating system dominating the entire smartphone market. According to research[4], 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 [5]. 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.

2.1.3 Data Collection
Our app shall collect as much metadata as it could for further analysis. The analysis results would help us to understand more about the user behaviour and hence we would improve the user experience.

2.2 Server Development
A server will be developed in this project to maintain the app’s social platform. The server shall at least serves the followings:

2.2.1 Data storage
The server shall store all matching related data and user data. The data shall organised in a user-oriented manner since it is a user-based application. We will design the detail of the database in this project.

2.2.2 Endpoint Connection
The server shall accept requests from the client (the app) and response to them accordingly. To facilitate the communication, an API will be develop for this application. The API shall use a RESTful approach to
keep the server stateless. A stateless server do not hold any user’s session, so it would able to allocation more resources to other operation, in our case, the matchmaking operation.

2.2.3 Social Platform
The server shall be able to operate social platform function. For instance, it would maintain a large amount of users information, the relationship between each user and the custom setting of each user. Furthermore, operations, for instance initiating posts, searching for friends and commenting should also be supported.

2.3 Machine Learning
Machine learning technology will be adopted in this project. To facilitate the user to identify their mood, whenever users initiate a post, they could include also a photo (with their face) and a description. Photo and description will be process by the machine learning unit.
3. Methodology

This chapter discusses the methodology we adopted to achieve our objective. The below includes our methods of developing mobile app and server. Also includes our considerations on choosing cloud services and the way of how our team collaborate during the software development of this project.

3.1 Mobile App Development

App development frameworks are mature and there are many choices. Each framework have its own characteristic and nature. Before we adopt to a framework, we need to understand clearly with our app development requirements.

3.1.1 Requirements

The requirements listed are the technical elaborations of the objective of Develop a Mobile Application (Chapter 2.1). The following are our key requirements and it provided a guideline for us to choose a framework that the framework shall satisfy all of these requirements.

1. Run on both iOS and Android platform

   It requires the framework support cross platform output. The output operation is foreseen to run quite frequently, we are more inclination to the framework that provides a light and reliable cross platform output.

2. Transfer data through RESTful API endpoints

   It requires the framework having a HTTP packaging tool and able to send them to a given address. Built-in packaging tool is not necessary, other extensions of the framework provide the same functionality are also acceptable.

3. Use of push notification

   It requires the framework calling the native notification library. This relies on the native connection of the framework.

3.1.2 Selection

After considering all of the requirements, we plan to use Flutter as our development framework. Flutter is a cross platform app development framework which developed by Google. It 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). Nevertheless, it satisfies all requirements we stated in the previous chapter.

**3.1.3 Justification**

React-Native is another cross platform app development framework that similar to Flutter and also satisfies all requirements. Based on the React framework, React-Native uses javascript and 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.

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 Backend Development**

There are lot of choices of backend solution. Besides the traditional servers that require lots of setup and configuration in a machine, cloud service appears to be another approach as most of the time it covers all requirement we need. Before we adopt to any of these framework, we have to list out all of the required components.

**3.2.1 Requirements**

The requirements listed are the technical elaborations of the objective of *Develop a Server* (Chapter 2.2). The following are our key requirements and it provided a guideline for us to choose a server development framework and set up a hosting environment.

1. *Provides RESTful API to our app*
It requires the server listening to a certain TCP/IP port, and giving responses accordingly back to the client. Supports of Apache [8] is required for packaging the RESTful HTTP requests and responses.

2. **Synchronise / Realtime database**

It requires the server have a good synchronisation to the data as our app is serving a social platform. Data would update and create rapidly, meanwhile users in their app should able to view the most updated information.

3. **Capable of handling hundred of requests in one minute**

It requires our setting to the hosting machine able to handle around 100 requests per minute. The number of request needed to handle is predicted based on an average API calls of an app when there are 1000 active users [9].

3.2.2 Selection

We plan to use Firebase as our backend. Firebase is a mobile and web development framework 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.

3.2.3 Justification

The main reason that we choose a cloud platform over tradition server is the development time. Compare to tradition server, cloud platform does not require the set up on hosting, network configuration and security. This save lots of time and our team can focus more back into the feature development.

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.

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.

3.3 Machine Learning

Machine learning can be done locally, for instance, by a python library, or done by the cloud services, for instance Amazon Web Service and Google Cloud Service.
3.3.1 Requirements
In this project, we will adopt machine learning technology in the mood analysis. We are planning to combine two different methods to analyze user’s mood.

Photo Analysis
The first method is photo analysis, in particular, the analysis of facial expression. By analyzing user’s facial expression, we are expected to know the mood of that user in different category and a score representing the level of intensity of that mood.

Natural Language Analysis
The second method is natural language analysis. By analyzing the content in user’s post, we are expected to know the sentiment expressed by that post in some score representation.

3.3.1 Selection
We will use the natural Language API and Vision API by Google Cloud Platform. The natural Language API provides a natural language analysis, for instance, if a message ‘I am sad’ is sent to this API for analysis, it would report that the statement ‘I am sad’ is somehow negative and presents it with a score. While Vision API is a photo analysis API, which can detect any human face in a photo and analysis their emotion.

3.3.2 Justification
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.
4. Current Progress

This chapter reports our current progress of the project. We have both design works and implementation done at this interim stage.

4.1 Design works
By following the principle of software engineering, design works shall be done before we go into the actual implementation. In current stage, we completed the design of both user interface (UI) and our mood representation model.

4.1.1 UI & Flow Design
Our app’s UI is designed under the guidelines of Google Material Design and our rule of thumb is to keep our interface simple and intuitive.

1. General Design

We use a light blue and white as our theme colour and we want to give our users feel relax when using our app. The following table shows the general styling specification:

<table>
<thead>
<tr>
<th>Primary colour</th>
<th>#ADD8E6 (light blue)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secondary colour</td>
<td>#FFFFFF (White)</td>
</tr>
<tr>
<td>Button colour (normal)</td>
<td>#FF0080</td>
</tr>
<tr>
<td>Button colour (highlight)</td>
<td>#81BEF7</td>
</tr>
<tr>
<td>Font family</td>
<td>Montserrat, Sans-serif, Roboto</td>
</tr>
<tr>
<td>Font size (title)</td>
<td>32 px</td>
</tr>
<tr>
<td>Font size</td>
<td>16 px</td>
</tr>
<tr>
<td>General padding</td>
<td>8 px</td>
</tr>
<tr>
<td>View padding</td>
<td>10 px</td>
</tr>
<tr>
<td>Icon size</td>
<td>16x16 px</td>
</tr>
</tbody>
</table>
1. Login view

**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 or Facebook.

**Flow:** When ‘sign up with Facebook’ button is pressed, our app will connect to Facebook and allow users to sign in with their Facebook account. When ‘sign up with Email’ button is pressed, our app will allow users to sign up with their email. When ‘Already have an account?’ button is pressed, our app will allow users to sign up with their account.
2. Sign up view

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 ‘Continue’ 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.
3. Home view

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

**Flow:** When the ‘Like’ 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.
4. Rate mood and intensity view

**Layout:** Figure 4 shows the design of mood representation view. This view appears when the user want to record their mood. For X-axis, the mood is represented from sad to happy. For Y-axis, the mood is represented in intensity.

**Flow:** When the user press or drag their finger on the screen, a grey circle will appear. The user can rate their mood according to emotion then intensity. The example shown on Figure 4 represents a very happy and high intensity mood.
5. Add description view

**Layout:** Figure 5 shows the design of *add description view*. The app will show this page after the user rate their mood.

**Flow:** When ‘arrow’ button is pressed, the user will post their mood and description. Their post will be seen on their friend’s mood feed. The app will send the user to the mood view (Figure 3) after description are added.
4.1.2 Mood Representation Model Design

It is necessary to integrate the result of the cloud service APIs (Vision API & Natural Language API) and interpret them in our own format, or the result would be vague and meaningless to the user. Based on this reason, we designed a mood representation model for our app.

1. General Representation

This model attempt to represent mood in a two dimensional spectrum. We defined the X direction as the category of mood, from -100 (Extreme negative) to 100 (Extreme positive). While the Y direction is defined as the intensity of the mood, from 0 (Extreme weak) to 100 (Extreme strong). A mood in our application will be a point which locate this two dimensional space. The intensity value is compute by the level of confidence of the analysis result, so a strong intensity mood is not only mean the user has a ‘strong’ mood but also a probability of our application has a ‘strong belief’ on the user by having that mood. Based on this fact, whenever we need to sort the importance of the posts, the one having stronger mood intensity will tend to show first, as we believe that it is more likely the ‘true’ mood of the user.

2. Conversion from the Vision API

The Vision API gives 4 emotional related result for each face in a photo. They are joyful, sorrow, anger and surprise. Each of these emotion have 5 level of result, they are ‘very unlikely’, ‘unlikely’, ‘natural’, ‘likely’ and ‘very likely’. Also, there is a detection confidence, which is from 0 (not confident) to 1 (very confident).

At this stage, we defined a positive mood as a composition of ‘joyful’ and ‘surprise’, and their ratio is 67% and 33% respectively. Similarly for negative mood, it would be a composition of ‘sorrow’ and ‘anger’ and their contributing ratio would be both 50%.

The begin the interpretation by determining whether the mood is positive or negative. We consider the level of result of each emotion, and we only consider the level with likely or above. Then, we try to calculate a score. A ‘likely’ result score 1 and a ‘very likely’ result score 2, and we calculate a score by summing up the positive component and negative component (with negative value). If the final score is between -0.3 to +0.3, it will be a ‘Natural’ result in our model, value from -0.3 or below will be ‘Negative’ and similarly, value with +0.3 or above will be ‘Positive’.

3. Conversion from the Natural Language API

Natural Language API gives a rather simple result that a score from range -1.0 (negative sentiment) to +1.0 (positive sentiment). There are also a magnitude value which is a strength of sentiment regardless the score. Its range is from 0 to infinity. A higher magnitude in a post means there would be some stronger sentiments expressing the post.
We observed that if the magnitude value in a sentence is more than 0.5, that sentence is very likely expressing the writer’s emotion and hence the score of the sentence would be confident by us. We use this behaviour to determine the level of confidence of the result.

This score level of this API is quite similar to ours model, so we would just map the score to our mood that -1.0 is mapped to -100 and 1.0 is mapped to +100.

4. **Combining both result**

The mechanism is still under design, but we have a basic direction.

For the case that the level of confidence of both method is similar (similarly high or similarly low), we give a base level of intensity to this situation. Case 1, both of the result having a high level of confidence (greater than 0.8), we set the level of intensity to be at least 75. Case 2, both of them having a medium level of confidence (greater than 0.5), the level of intensity will be at least 50. Case 3, both of the level of confidence is less than (0.5), the level of intensity will be at least 20.

For the case that both result do not have similar level of confidence, we interpret this situation as the user have some complicated mood, like he looks like happy but deeply in his mind, he is sad. It is difficult to know his real mood, so we will put its intensity to a low value, which is around 30.

4.2 **Implementations**

The current implementations are mainly for our team to play around with the development environment. We update our implementation every week after reviewing the feedbacks.

4.2.1 **App Demo**

We followed our UI design and built an app demo. Integrated with Firebase authentication model and Firebase Realtime database, so we can perform login and logout in our app and also we can write some data to the database through the app.

4.2.2 **Firebase Backend**

We configured the Firebase authentication model and allows user to login to the app. In this stage, only ‘Email & Password’ method is implemented and we are currently testing with other method, such as Facebook login and Google login.

The Firebase Realtime database is also implemented for simple data storage. We defined some collections and start putting some dummy data in it for testing. We used the characteristic of the real-time database and syncing all user related data to the app instance, however, we found that this method would have a huge network overhead, so we are redesigning the structure of the collections to reduce the network overhead problem.
5. Conclusion

This chapter concluded our current progress by stating out our challenges and future plan. This report ends with a brief summary.

5.1 Challenges

We are facing many challenges throughout the process, some of them took us hours to solve while some of them took us a week or even more. We highlighted some of the big challenges and listed below:

1. **Design completeness**

   Our mood representation model is still not a well defined model. Whenever we put data to this model, there are always some exceptional cases that we did not think of how to handle yet. We believe time would solve this problem, but we do aware that time is not much and we shall settle this as soon as we can.

   Our app also facing a design incompleteness issue. When we design the app flow, we did not have a comprehensive error handling flow. So, we are still struggling in the app flow when the app encounter errors.

2. **Customized UI Widget**

   Flutter supports the implementation of custom UI widgets. There are some views required us to made our own UI view (not using the material UI), while there is one particular view challenged us. The rate mood intensity view is a 2D coordinate-like view and it is use for record users mood and its corresponding intensity. The challenging point is we have to map the screen space to the model space. Game engine, such as Unity provide handy library to do this, however, in Flutter, we have to do this ourselves, and until now, we still not satisfy with the outcome.

3. **Cloud service settings**

   Cloud service platform indeed very powerful and it provides various solutions to different need. However, to configure a service to match our need took us almost a month. The setting on Vision API and Natural Language API was simple, but connecting the app with the cloud service was complicate. Not only we have to set the API keys, we are also required to host a privacy policy web page because of the GDPR requirements. Besides, we have to set billing quota and limit, and manage and arrange the network usage quota.
5.2 Future Plan
We reviewed our schedule and found out we are behind it. Important design works such as database and mood representation model is not complete yet. Those designs are significant, so we agreed to arrange them to the highest priority work and plan to complete them by January.

The next major task is to distribute our workable demo app to serval testing devices and begin the phase 1 testing. Meanwhile, we will continue with our mood model implementation and it is expected to launch in February.

The following shows the revised version of our project schedule:

January 2019
- Complete the app demo with backend integration
- Begin distributing testing (iOS)

February 2019
- Complete mood representation model
- Begin distributing testing (Android)

March 2019
- Do testing, debug and final adjustment to the algorithm

April 2019
- Do testing and debug and finalise the product

5.3 Summary
This project has a very clear goal, which is to develop a cross-platform application for users to record and share their moods. Flutter is used for developing the app; Firebase is used as our backend; Machine Learning technology is adopted and our selections are Vision API and Natural Language API from Google Cloud Platform to facilitate the mood analysis in the app.

In this interim stage, design works of user interface and the main frame of mood representation model design are completed and we are testing it with our app demo together with the Firebase backend. Meanwhile, we registered account on cloud service platforms and playing around with their services. We are facing some challenges, for instance, the problem of design incompleteness, customised UI Widget and Cloud service settings. We are planning to complete the remaining functions by the end of February and do testing and debugging in March and April.