Individual Final Report
for
a Privacy-preserved
Instant Event Sharing System

Chan Kin Long (3035184922)
Cheung Tin Long (3035187833)
Kwok Ching Hei Kevin (3035185031)

15 April 2018

Abstract

Social networking websites and forums have been popular for over a decade. Each of team provides some common features such as imaging sharing and private messaging and different systems have different strengths. However, a huge missing feature, real-time event information sharing, is not found in the existing applications. This project intends to provide and develop a new web and mobile system that fill in the gap.

This paper describes the design, development and testing of "happens", the real-time event information sharing application available on mobile and web platforms. It includes three major functionalities – photos sharing, category-location subscription, request-and-respond. These features allow anonymous users around the world to connect each other based on their locations and interests.

At the end of this project, two prototypes – web and Android applications will be available for the demonstration purpose. The web application is written in React (a JavaScript library) and Android application is written in Kotlin. Backend server used a number of technologies, including Amazon Web Services Elastic Compute Cloud (AWS EC2) used for the hosting of web application programming interface (API) and primary database MySQL. Another service used is Firebase which provide authentication system, real-time data access, file storage and cloud messaging system.

This paper marks the end of the project. This suggests that two porotypes are designed, implemented and tested. These stages do not imply the applications are production ready in terms of performance, user interface and completely bug-free, but they provide a reasonable user experience to explore our application. Though the project is completed, the application may receive feedback from users. The applications will be kept maintained and improved. To include a large amount of iOS user, an iOS application will be ported in the near future.

Acknowledgement

We would like to express our gratitude to our supervisor Dr. TW Chim, who initiated the idea and guides us throughout the project. This project would not be on the right track without Dr. Chim's timely advice. We would like to also show our appreciation to the Computer Science Department of the Faculty of Engineering of the University of Hong Kong, which provides resources for us to develop the system.

Table of Contents

List of figures

1. Introduction

This section introduces the project, overview of the application and describe the remaining structure of this paper.

1.1. Background and Motivation

The rapid development of the internet has led to a lot of information sharing system – social network sites, online media and forum. Most of these systems provide excellent user experience in achieving different goal such as connecting real life friends, initiating discussion, supporting various types of media sharing. However, none of these enables user sharing real time location information easily. Forum-like applications though provide a platform for general discussion, they do not provide a good experience for a request-and-respond use case.

To address the issues, a platform that is specially designed is expected to have demands. Therefore, in this project, it is aimed to develop a prototype that demonstrate how users can exchange real time information based on their current locations in different situation. In the application, users may customize the information they share to others and type of information received. With a successful implementation and marketing strategy, it is foreseeable that the public are willing to explore this system, helping users who do not know each other, but able to connect each other.

1.2. Scope of Study

This project successfully developed the essential features which fulfil the requirements. The scopes can be divided into three sections: client side, server side and not included functionality.

1.2.1. Client-side application

1.2.1.1. Authentication

Though it is not necessary, the system encourage all users to register to get access to all the functions. They only have right to read the public data (i.e. reading the information other users have shared) but they cannot write any data (i.e. sharing, requesting or responding information) for other users. Non-registered users also do not benefit from category-location subscription system which will be mentioned in the coming section.

1.2.1.2. Photo sharing

Similar to other social networking sites, users may share information actively to other users. Moreover, the sharer also needs to specify the categories and the regions of the information. For instance, an image with category “traffic” and location “Central, Hong Kong”. Classifying the information this way can help other users searching for their interested information. The set of information including photos and text is called “a post” which will be referenced in the other sections of the paepr. On the other hands, other users may also interact and express their opinion to the uploaded information by upvoting, downvoting or commenting on it.

1.2.1.3. Request-and-respond

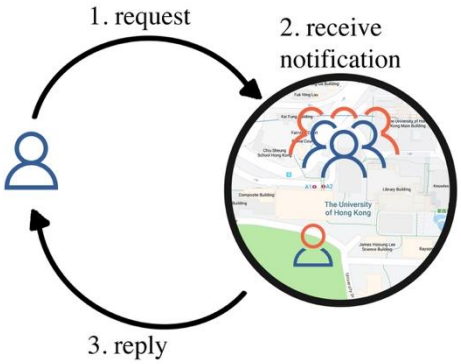

Figure 1 System flow of the request-and-response function.

The main feature of the application is request-and-respond function. A user may choose a location on the map send his/her request to the server. The request will immediately available and all other users are able to see that an information request is posted. The other users who are near to the request may respond the relevant photos and text. Meanwhile, upon receive a request, the server automatically search for users who were recently there to encourage more users responding to the request.

1.2.1.4. Category-location classification

The system allows users to subscribe to the information they have interested in. By providing classification information during the uploading information period, the system can easily filter relevant information for the users. For example, a user may subscribe the posts about traffic located in Mongkok or news of Hong Kong.

1.2.2. Server-side application

The server is set up to interact with the client devices.

1.2.2.1. Database and storage

The server stores all the users data and the sharing information securely in database. Uploaded images are stored separately (i.e. not inside a database) and they will references with a uniform resource locator (URL).

1.2.2.2. Web Application Programming Interface

In no circumstances that a client device can write directly to the database. Therefore, a web API could provide an entry point for all client devices to communicate with the server and enforcing all submitted data to database are checked and valid.

1.2.2.3. Server configuration and protection

The server hosts the API and the web application. It is essential to be configured correctly including processing concurrent request and preventing in proper access to the database with firewall.

1.2.3. Not included

1.2.3.1. Private messaging

Private messaging is not included in the current and the future plan of the development. Although this feature is common in other similar applications, this application intends to share the public data only. Therefore, all the information except personal data (e.g. email address, password, etc) should be available to every user.

1.3. Deliverables

This project is to develop a prototype of an Android application and a web application. Both of the applications only work as a prototype and are not public available in Google Play Store and a public accessible URL. However, the mobile application can be installed directly from the source code and the web application will be set public accessible during demonstration.

1.4. Structure

This paper mainly contains four parts. First, it describes the implementation details of the system. Second, it will address the problems encountered during the development and the solution to it. Next, a brief future plan will be covered. Finally, the details of application functions and user interface with screenshots will be shown.

2. Methodology

This section describes the methodology of the project which includes the details of platform setup, database and storage, authentication, user interface and system architecture.

2.1.Platform setup

2.1.1. Front-end

2.1.1.1. Android application

To target the most number of smartphone users, Android is chosen as the first mobile platform to work on. [1] Though there are other choices such as React Native to develop a cross platform application, considering one major feature, Google Map, being not officially supported and lacking some essential functions including marker clustering, it is decided to firstly work on native language.

The application is written in Kotlin with Android Studio. Another popular programming language choice is Java. Since Kotlin's code produced the java compatible bytecode, performance can be considered the same. However, Kotlin provide a number of features that Java does not have. For instance, data class, companion object and inline function are frequently used in the project. Therefore, the Kotlin is selected to be the primary language.

2.1.1.2. Web application

Web application is also prioritized in the project as it is expected that the system will expose to a lot of data. A web application which can be viewed from a computer's browser can be a better choice. Also, apart from uploading photos to the system, there are also a number of other functions that can be used without camera such as viewing information, requesting information and commenting. It is deemed a web version can attract more users to this system.

The web application is written in React, a JavaScript library which is specialized to create a single page application (SPA). The benefit of build a SPA is that it can reduce the website loading time. The loading time can be reduced because some fixed resources such as header and footer do not need to reload. Also, during loading new data, the browser could show relevant animation such spinner to notify the user. However, for a traditional website, it is not uncommon that after clicking on a link, the main screen changes nothing while waiting for server response. Therefore, SPA could provide a better user experience. Another reason of using SPA is that it can share the same web API with the Android application because it returns JSON can the browsers can easily process it to HTML.

2.1.2. Back-end

2.1.2.1. Server

A web server running Ubuntu has been set up on AWS EC2. Its roles are to host the web application, web API and database. One major reason of choosing AWS as the cloud computing service is he has the most number services which suits for most system. Considering this system have the potential to grow rapidly, scalability is a focus. In the future, it is possible to migrate different applications to their different cloud services such as Elastic Beanstalk for web API, Amazon Relational Database Service (RDS) and S3 for React application.

Inside the virtual machine, a major system application running is NGINX which is used as proxy server. By default, Apache was installed with Ubuntu. However, it is an old technology and designed for general use case while NGINX is good at reserve proxy, loading balancing, scalability, etc. Therefore, it is a better choice for a freshly new project when there is no other existing application to interact with.

2.1.2.2. Web API

The web API serves as the access point for communication of different client applications and database. It safeguards all the data submitted to the database, ensuring invalid data will be rejected and will not be delivered to other client applications.

The web API is written in Python with two important packages, Flask and Sqlalchemy. The first package is web framework which can listen to HTTP requests, based on the uniform resource identifier (URI) to process and return different result. The latter package is used to communicate between the server and the database. One feature it provides is Object Relational Mapper – using python object to interact with the database without using raw SQL.

2.1.2.3. Version control

Git and GitHub are the version control system used in the project. It is used to track the changes of computer files, especially sources code. Its mechanism allows different developers to cooperate for the same project. By setting up different branches, it ensures that developer only makes changes to files that do not affect production system. It also allows quite revert if after committing new changes, the new system does not work as expected.

2.2. Database and storage

Since the system exposes to a large amount of data and files, it is required to used database management system and storage services to store and structure the data.

2.2.1. MySQL

MySQL is one of the database used in the system. It stores all the data submitted from the client applications in different tables. A relational database manage system is chosen because the data in this system are highly relational. For instance, there are different "things" mentioned in previous sections – user, post, request, category, etc.

Post is one of the major element the system, but it also associates to a lot of other tables (things). For instance, a post has a creator (user), and it is mapped to different categories. Furthermore, other users may also leave comments and up or downvote to a post. Modelling the post data in a relational manner ensure that they can be retrieved in various ways. This allows the functionalities of searching posts by time, user, category or/and location.

Moreover, relational database management system provides functions to enforce the above relationships, a post must be associated to a user (its creator) or a comment must be linked to a comment.

2.2.2. Firebase

Firebase is a web and mobile application development platform. It provides real time database and file storage. They are used together with the MySQL database to provide data access by the user.

Firebase real time database is used in the client applications to provide real time database access. The libraries provided for the client-side applications allow them to listen the data changes and therefore real time update. For example, whenever a new post is added to the Firebase, the client devices are able to detect this change and update the browser or mobile screen accordingly. Furthermore, the data are stored in key-and-value format inside the database. Therefore, the client applications can process it in a way like JSON or HashMap which is easy and efficient.

Another function provided is Cloud Storage. All the photos will be uploaded directly to Firebase. The MySQL and Firebase databases use URL to refer to different photos.

All these two functions are integrated with the Firebase Authentication. This ensures only authenticated users are allowed to read and write to the data. The details will be discussed in the coming section.

2.3. Authentication

Authentication is an important part of the system. While all the users can read the public data such as posts, requests and responses, authenticated users can do more by sharing their information. This can enhance the system security from abusive use by blocking the users who violate the rules. Currently, an email-password authentication system has been implemented. The users are also required to validate their email addresses before actually enjoying the functions of a registered user. Also, like other membership system, Firebase provides “forgot email” function which allows users to reset their passwords easily. More importantly, these two functions, validating email address and password reset can be easily customized with the Firebase web console using their email templates.

Firebase Authentication does not only allow users to gain access to other Firebase services such as real time database and cloud storage, but it can also generate ID token which is used to authenticate a user in the custom web API. The flow of actions of the registered users will be discussed in the next section.

2.4.System Architecture

The system architecture of the project contains two parts, one is the model used for Firebase real time database and cloud storage access and another one is client-server to communicate between client application and web API.

2.4.1. Firebase model

The authentication provided by Firebase is well integrated with its other services including real time database and cloud storage. Therefore, Firebase has provided libraries for Android and web to perform action with Firebase services.

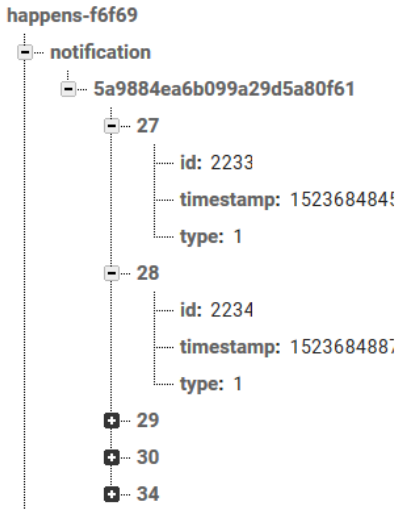

Figure 2 Notification node of Firebase real time database

First, a user logs in with Firebase using their devices (e.g. Android phone). After successful login, he/she can read or write to the Firebase. For example, a logged-in user can upload files (photos) to the cloud storage. It is also possible to restrict the detailed of access of a user. For example, a user can only read some certain nodes (places) of the database such as all the child nodes "/notification/<user_id>" where the <user_id> must equal to the logged in user id (see Figure 2). All these implementation and verification details are automatically handled by Firebase.

2.4.2. Client-server model

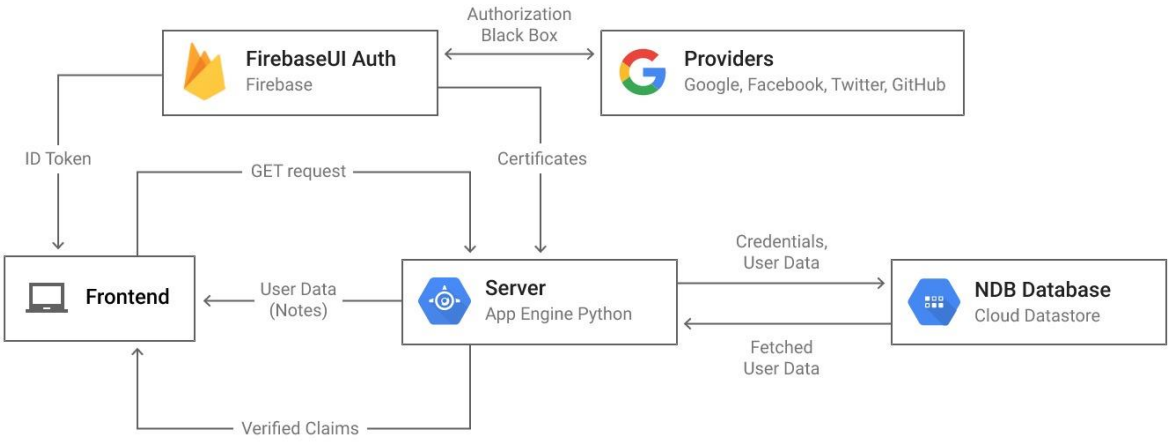

Figure 3 System architecture

The system architecture is a client-server model (see Figure 3). Except the Providers which is not currently used in the system, it contains about four parties, client applications, authorization server, web API. To implement this model, Firebase implemented another mechanism which allows custom server (web API) to verify the data submitted from the client applications.

ID Token Payload Claims		
exp	Expiration time	Must be in the future. The time is measured in seconds since the UNIX epoch.
iat	Issued-at time	Must be in the past. The time is measured in seconds since the UNIX epoch.
aud	Audience	Must be your Firebase project ID, the unique identifier for your Firebase project, which can be found in the URL of that project's console.
iss	Issuer	Must be "**https://securetoken.google.com/<projectId>**", where **<projectId>** is the same project ID used for **aud** above.
sub	Subject	Must be a non-empty string and must be the **uid** of the user or device.
auth_time	Authentication time	Must be in the past. The time when the user authenticated.

Figure 4 Attributes of Firebase generated ID token

First, using Firebase client libraries, an ID token containing the several attributes (see Figure 4) can be created for authenticated user. Two key attributes used in the web API are “exp” Expiration time and “sub” user id. Upon the server receiving a HTTP request, if the certain action requires authentication, it check the presence of ID token and also the expiration time. After

validating the token, the server directly uses the user id stored in the ID token to perform restricted action such as creating a request with the user id.

The valid time of the ID token is designed to be short (e.g. less than one hour) such that the web API does not need to verify each token with the authorization every time. Instead, the web API directly uses the user id to perform the requested action. The short live time of ID token prevent from causing huge damage for exceptional cases. For example, a user may change his/her password, but the access token would still valid for its remaining time. Another example is the leaked the of ID token. Even if the ID token is stolen by hacker, the hacker cannot perform a lot of actions due to its short live time.

Upon verifying the user identify or if the action does not require authentication, the web API communicates with the database to retrieve or write data.

Finally, the web API returns a response in JSON format which is commonly used in web application as it can be understood and processed easily by both browsers and mobile devices.

2.5. User interface

The user interface (UI) is how the application interact with users and it is also the key to provide good user experience. Using a consistent layout helps the user to predict the functions of the controls (e.g. links and buttons).

2.5.1. Android application

The Android application has three main parts that are used across the application.

2.5.1.1. Bottom navigation

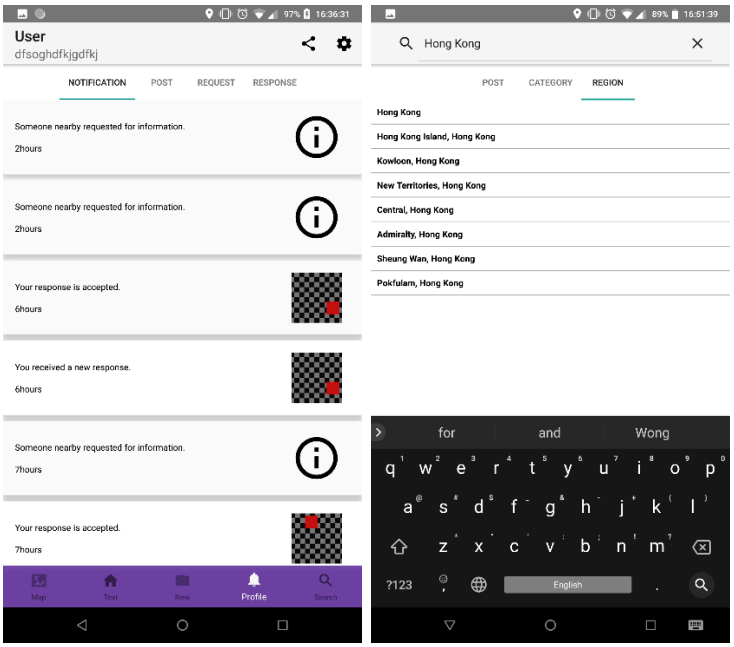

Figure 5 Examples of bottom navigation and tab layout

The bottom navigation (see Figure 5) is shown on the main activity and therefore is available on most of the screen. It is designed to put the front screen (map, text, etc) to different tabs such that users can move to the corresponding tab with a single tap. It also clearly shows the current tab which the user is using. If the user is deeply nested the current tab, for instance, clicking on a post, clicking on a user then clicking on the user's post and so on, he/she can click again the bottom navigation to remove all the visited and instantly back to the front screen.

2.5.1.2. Tab layout

The bottom navigation aims to categorize the front screen only while the tab layout (see Figure 5) is also used for grouping similar pages of a category. For example, in user screen, a user contains different data such as posts and requests he/she created. Another example is searching for different type of data. Using the tab layout is able to group different data to different screens. This can help the user to locate the information they are interested.

2.5.1.3. Action bar

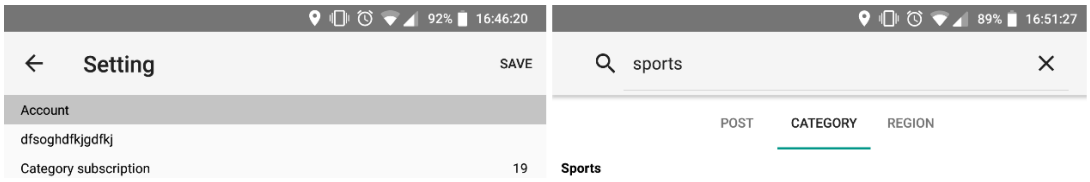

Figure 6 Examples of action bar

The action bars are available in all the screens and it serve two purposes, provide up navigation (going back to the previous screen) and indicating the current page (as opposed to current tab mentioned in the section of bottom navigation) the user is viewing.

2.5.2. Web application

The web application targets the user using a computer with browser. Therefore, there are some differences in designing the user interface for the web application than the Android application due to the larger browser screen and control (mouse and keyboard). The web application has two main parts that are used across the application.

2.5.2.1. Header

Figure 7 Header in web application

Header (see Figure 7) serves similar purpose of the Android's navigation bar, but it is put on the top. It allows users quickly traverse different pages

2.5.2.2. Popup windows

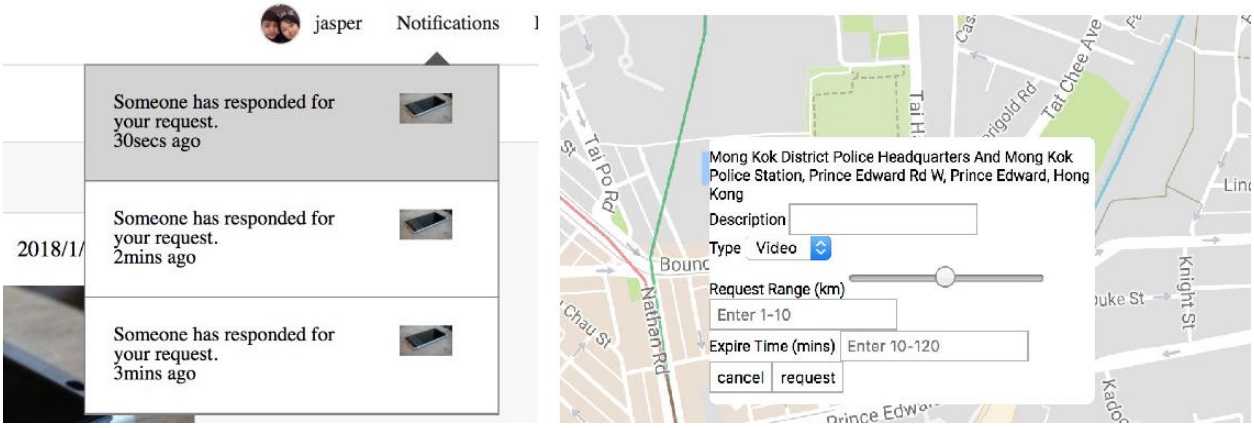

Figure 8 Examples of popup windows

Since the browser screen is big enough, it is not necessary to change the replace the entire screen to show a little data. Instead, a small window hovering on the original content is sufficient to show the new information.

3. Current status

This section describes the details of completed functions in client applications, web API, server configuration and testing.

3.1. Android application

The prototype has included several major functions enabling sharing, requesting and responding information.

3.1.1. Posting information

The posting information functions allow user to upload up to ten photos. After taking photos, the user needs to input additional information such as description and the location to illustrate the details. To associate a post to category-location system, the user is also required to choose up to three regions and categories from the pre-defined lists.

3.1.2. Viewing posts and requests on the map

Each post on the map is represented with a circle marker and they are updated in real time (i.e. colour changing, removing olds post and adding new posts). The colour indicates the posted time of the post. Newer posts are shown in green while the older ones are in red. The user may define the the type of posts and range of time of the posts to show on the map in the setting page. Clicking on a marker brings the user to the text view page to view the details of the post. The setting and text view pages will be introduced in the coming sections.

Each request on the map is represented with two colours of exclamation mark. A red marker indicates the request is not resolved, i.e. there are not response that are accepted by the requester. The resolved requests are also shown on the map but indicates in green since it is possible that other users may also be interested in the information in that area.

Considering that users may be interested in lots of information, hence lots of posts and requests possibly show on the map, marker clustering is implemented to group nearby markers. Clicking on cluster item lead to the map zooming in to view the details of the map.

3.1.3. Requesting information

To request information, a user may long press on the map, a marker of the application logo shows on the map and ask whether the user wants to request information in marker location. Then the user needs to input the details of the requests including the description, how long the request should hold and how far away the other users can respond.

3.1.4. Responding information

Users can view the details of requests, if they are within the requester's defined range, they can respond to requests by taking photos. Afterward, they can optionally fill in extra information such as description. This step is marked as optional since reducing the effort of responding is possible to encourage more users helping each other.

Furthermore, there are two options to boost the information exchange.

First, by default, the response is also delivered to nearby requests. It is because it is possible that many requests are within a small area. It will be inefficient and discouraging for the responders to take photos and respond to each of the requests. However, this will lead to the problems that irreverent responses being sent to the requesters, to reduce the possibility, the nearby range is set to a sufficient small to 100 meters and the responders may choose which nearby requests they also want to respond to.

Second, the responder may also set the response to create a new post. However, choosing this option require the responder to fill in the description and also picking regions and categories.

3.1.5. Viewing posts in text

There are two forms of text view – showing a list of posts and view one post only. While the map helps users knowing the location of the information, it is essential to have a text view in order to read the details. The text of a post contains username, posted time, description, location, regions and categories. As mentioned in early sections, the application allows to interact. Therefore, the score (number of upvotes minus number of downvotes) and comments will be display as well.

User may also click on different areas to go to different page.

1. Click on the username to see the user view of that user.
2. Click on the image to view the image in full screen.
3. Click on the location to view the exact location on the map.
4. Click on the category to view all the posts associated category
5. Click on the region to view all the posts associated to that region
6. Click on the comment icon to start to write a new comment

Users may click on the upvote or downvote button as well. However, once upvoted or downvoted, it cannot be undone.

Users may click on the option button to share or report the post.

3.1.6. Commenting on post

Users may be interested in knowing more about the post. The commenting function allows to communicate with each other. The comments are updated in real time.

3.1.7. Category-location subscription

Users may want to keep getting updates from some categories and locations. The subscription system allows user to subscribe a pair of category and location, for instance, (Mongkok, traffic) and (Hong Kong, weather). Having this subscription list, users may view the posts the match the criteria on the map or in the text view.

3.1.8. Viewing posts by categories and locations

Users may view the posts of a location or category. For example, while viewing a post, a user may click on the category, lists of posts associated with be loaded accordingly. The user may then further filter the posts by choosing different locations. For example, the user can end up with viewing traffic posts in Mongkok, Prince Edward, Yau Ma Tei only. Similarly, users can also first filter the posts by location then by categories.

3.1.9. Viewing user profile

The user profile page allows users to view posts, requests and responses that the user have published. There is an extra tab – notification tab to show if the user is viewing his own profile. The details will be discussed in the coming section. There is also a share button allowing user to share to profile to others easily.

3.1.10. Search

The search page currently allows users to search for posts, categories and location. For post searching, if a post's description contains the search string, it will be return to the user. For category and location search, the system matches the search string with their title. Users can then click on the category or location and view the posts that category or location.

3.1.11. Setting

The setting page allows users to manage the following:

1. Log in or log out
2. View subscribed categories and unsubscribe
3. Set application start up screen
4. Manage types of notifications to receive
5. Enable or disable the request function on the map
6. Manage the posts and maps shown on the map
7. Report issues

3.1.12. Notification

There are two types of notification – in app notification which the users can view in their profile page. Another type is push notification such that even if the users have close the application, they will still receive updates about their posts, requests and responses.

Currently, there four types of notification.

1. A new request is nearby
2. User's request is responded
3. User's response is accepted by the requester
4. Post containing inappropriate images is processed (temporarily/permanent removed or restored) by the server

3.1.13. Localization

The default language developed for the application is English. However, considering that this is Hong Kong and potentially Hong Kong people would be a large portion of users. Most of the text used in the application are translated to traditional Chinese. The language setting follows the device language. Therefore, for devices running the operating system in Chinese, the application shows Chinese text and otherwise, English is shown.

3.2. Web application

The implemented functions for web application are very similar to the Android's. However, the photo sharing function is not implemented because not every computer has a webcam and more importantly, the webcam is not designed for taking photos but video conferencing.

On the other hands, the web application allows users to search for posts nearby to some areas. This function is not the same as viewing posts by region or searching for locations. These two functions mentioned in the Android section are designed to search for pre-defined locations while the location search function implemented in the web application is to search posts by the geo-location (i.e. latitude and longitude).

3.3. Web API

The web API allows the client applications to retrieve and submit data. The API functions that are fore the client applications such as posting information, requesting have been implemented. The following are the key actions the web API have been implemented.

3.3.1. Data validation

All the data adding to the database must be first verified by the web API. The following are some forms of validity checking:

1. Text length (e.g. post's description must not be empty and less than 22,000 characters)
2. Latitude and longitude
3. Requests valid time (a request must be valid for at most 2 hours)
4. Response valid distance (the distance of submitted response must be within the distance defined by the requester)

3.3.2. Searching for nearby users

Upon receiving a request, the web API needs to find the users that were recently nearby. Currently, the web API look for the users whose locations are within the requester's defined range in the last 30 minutes.

3.3.3. Inappropriate image detection

Upon receiving a new response and post, the web API will communicate with Google Vision API, sending the image URLs to the API and the result is whether the image is inappropriate such as containing adult and medical images.

However, the detection takes a few second for each photo, it is not reasonable for the users posting information to wait. Therefore, the current implementation is to allow all the images posted online first. Meanwhile, the web API passes the URLs to the API in background. If the result is containing inappropriate contents, the images are taken down temporarily but immediately and finally inform the posters. The whole process should take

less then minutes.

3.3.4. Email auto reply system

Keeping end user in touch is a key element to a successful application. Therefore, the opinions and feedbacks from the users should be taken seriously. The client applications have several options to report issues to the development team such as reporting posts and system-wide issue. Therefore, whenever the web API receive reports, it automatically replies to the users that the administrators will look into the issue.

3.4. Server configuration

The web server used is a virtual machine in AWS EC2. It plays a vital role to serve the web application, web API and database. Since the instance is externally accessible, this requires some proper set up before putting it alive.

AWS EC2 comes with a web console to allow view and control the server state. First, ensure the instance firewall to block all inappropriate incoming traffic where only port 22, 80, 443 are needed. With the IP address shown on the console, corresponding DNS record is also set to point to the web server.

Since both the web application and API are running on the same machine, a reserve proxy server is required to route HTTP requests to different ports (e.g. default port 8080 for React application and port 5000 for Flask application). By default, Apache is already installed in the server. However, NGINX is a newer and performance-better technology and therefore it is chosen to serve as a reserve proxy server 4.

Furthermore, since the web API may receive a huge number of concurrent requests, uWSGI is set up in the middle if NGINX and Flask application. It allows the web API to listen to more than one HTTP request at the same time which cannot be done without it. Currently, 10 processes are set and therefore, it is possible to handle 10 concurrent HTTP requests.

3.5. Testing

The project currently focuses on the web API testing which includes unit test and integration – the accuracy of data and the performance test. Both tests are done with Python.

3.5.1. Unit test

To check whether the data submitted to the web API are processed correctly, two sets of data are prepared for the tests. One is the data submitted to the server and another set is the expected result including the HTTP response code and the response body. The body to check against involves the type of data and also the actual data value. For example, the web API must return an Integer for post_id field and list of Integers of category_ids field. Currently, the tests have covered creating and getting posts, requests and responses including valid request and invalid request. In the cases of invalid requests. The web API should return response code such as 422 for invalid data and 403 for performing authenticated user's action without a valid ID token.

3.5.2. Integration test

One important integration test conducted is request system. The web API should search for nearby user when there is a request submitted from a user. Therefore, the test begins with setting an Android phone to a location. Then a Python application (emulating a web application or Android application) submitting a request, where the coordinate is near the previously set location, to the server. Lastly, the Android phone should receive a push notification to indicate there is a nearby request.

Another integration test conducted is create and retrieve function. When a client application creates a new post, it also receives its post_id. The test is to check the newly created post indeed exists with that post_id and the content such as description and location is the same submitted.

3.5.3. Performance test

As mentioned in the last section, the web API is set up to received concurrent requests, some of the request can be process-consuming. Therefore, the Python application is set up to emulate the actual traffic to the web API.

The tests running in Python are set up with 3 factors.

1. Number of threads – number of concurrent requests
2. Number of total requests
3. URL

The test client records the start time and end time of each request. Some minimum, maximum, average process time can be retrieved.

However, the tests results are not very satisfactory, and it will be discussed in the next section.

3.6. Problems encountered

There are several problems encountered in the project that include category suggestion, abuse of request system, and performance.

3.6.1. Category suggestion

Initially, it is planned that when a user wants to create a new post, category and location will automatically be suggested to reduce to poster efforts. However, currently it is only possible to provide location suggestion based on the device location. For the category, original plan is to make use of Google Vision API, which can classify an image into different categories, to provide suggestion. However, it turns out that the Google Vision API returns the result that are very fundamental such as car, tree, person, etc. It is hard to map these object to the system's category list. The current and temporary is to provide the user's subscribed list instead. After the application and the data size grow, the images and user's classified category can be used for machine learning and to predict the image suggestions.

3.6.2. Too many data on the map and abuse of the request system

Initially, it is planned to only shows the unresolved request because once it is resolved, there is no reason to keep it on the map and ask for more responses. However, it is possible that some or many other users may be interested in the same area. Therefore, an accepted response may also be of another user's interest. This allow other users to view the information as well.

Another problem is that, in setting page, users can set the type of requests show on map. Although there is an option to show all the requests on the map, it is unrealistic to expect that users really want to see all the requests. Instead, they may set it to show only the nearby requests or no requests are shown at all. Therefore, the requester actually will not know whether a similar request has been made. The solution to this problem is show the nearby requests when a user is trying to request for information and allow he/she to follow that request.

Furthermore, a quota system is set to limit the amount of request a user can make. Currently, only one request can be made by each user every day.

3.6.3. Web API poor performance

In the previous testing section, it is mentioned that concurrent requests are made to test the web API response time. However, the result is not satisfactory.

First, the tests include a simple API calls that return a static result – no interaction with the database. This result is the average response time is less than 0.1 second. Next, the test tries to create posts concurrently and the average response time is around 1 second (the image uploading time is not included in the test) which is also acceptable.

However, the problem comes when trying to retrieve the data. For a simple non-concurrent request, the web API takes more than 1 second to respond and more than 10 seconds in average when the client requests concurrently. The poor performance is unexpected but understandable due to the limit of relational database. Currently, a get post HTTP request contains a lot of data in the response data – original post content, voting score and number of comments. This involves 4 tables (post table, image table, voting table and comment table). Performing query to retrieve data requires query optimization which is not consider.

Apart from indexing on all required columns, no other solution has been tried to increase the performance. However, there will be three possible actions to solve. One is to optimize every SQL. Another one is separate the database server from the web API server. Although there will be network transferred time added to it, it can avoid the web API application and database fighting for the same hardware resources. Lastly, it is to move post related data to a NoSQL database such that the image URLs, comments and voting data are stored together with posts.

4. Future planning

4.1. Server migration

Currently three applications – web API, web application and database are running on the same instance and causing performance issue. For a production ready system, the applications should be separated to different server. For example, web API can be moved to AWS Elastic Beanstalk, React application can be moved to Amazon S3 and database can be moved to Amazon Relational Database Services

4.2. Client application optimization

This application is data-driven application and it requires a lot of network transferred. This is a huge burden to the smartphone as their network speed and quota are limited. Therefore, two optimizations can be made as follow.

1. Image compression – currently, the client application compresses the images to around 100KB before uploading to server. However, in other social networking sites, they have several sets of images such as full size, medium size and thumbnail version. For example, when users a viewing the notification page, a small image is show side by side with the text. Currently, it is using the original photo which is not necessary for less than 100 pixels times 100 pixels screen size.
2. Handling mobile network and WIFI network differently – in the case of having multiple sets of images, it can be that when the application is running with mobile data, only a medium-size image is loaded. When the application is running with WIFI, a full-size image can be loaded.

4.3. Advanced membership system

As mentioned in the previous section, each user is limited to send one request every day. Considering that the system is to encourage more users to help each other, for the users who actively provide useful information. They should earn some credits and they can be used to create more requests. On the contrary, for the user who do not follow rules by make inaccurate information, the system can also decrease their quota of requesting, posting or even commenting.

5. Conclusion

This report describes the different phases of project “happens”. The main motivation of the project is to encourage users sharing useful information to each other. This functionality is something that no other social networking websites are providing.

Then the paper moves on to discuss the functions and implementation details of the application. Several key features such as photo sharing, request-and-respond and category-location system are introduced. The system runs mainly in a client-server model and occasionally with Firebase real time database to provide real time data update while users are using the application. Then the server-side roles are also introduced – verifying data, inappropriate image detection, searching for nearby users, etc. The prototype of Android application and web application are available for internal testing and demonstration.

Next, the paper describes the issues that the system has faced and is still facing, performance being the largest problems. For each problem, actions have been applied to resolved and potential fixes are suggested.

Finally, the prototype is ready for demonstration and user feedback. The application will continue to be maintained and updated. After the client application has received some other feedbacks, improved accordingly and thoroughly tested, also the web API issue is fixed, it can be officially launched for the public.

6. References

[1] Global mobile OS market share in sales to end users from 1st quarter 2009 to 2nd quarter 2017 Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/ [Accessed 14 Apr 2018].