The University of Hong Kong

Department of Computer Science

A Privacy-Preserved Instant Event Sharing System

Final Report

Cheung Tin Long 3035187833

Abstract

The project aims to develop an instant event sharing system, which serves as a platform for people to exchange real-time data all over the globe in a fast, reliable and efficient way. Despite all the existing communication tools and applications that are widely used, there are still no particular software excelling in real-time event information sharing with accurate categories and classified locations. Therefore, the project is to implement a software application that solves the above problem and advances the progress of information sharing.

The system, named “happens”, symbolizes “to explore what is happening anywhere in the world”. To achieve this purpose, The system has five main features: request-and-response function, media-sharing function, category system, an authentication system and a search engine. With the request-and-response function, users can basically request all kind of data anywhere and receive a response from another user at that place. Other features are equipped to enhance the performance and improve user-experience.

The final deliverable of the project is a mobile application and a web application. This is a common approach to most popular social media applications (e.g. Facebook, Instagram, Reddit, Twitter). Swift and Kotlin are used for the front-ends of the mobile application for iOS and Android platforms respectively, while React.js is used for the front-end of the web application. Both the mobile application and the web application share the same back-end. Ubuntu server is used as the back server, which is provided by Amazon Elastic Compute Cloud.

Acknowledgements

The accomplishment of the project relied effort from different parties. The project is accomplished by a group of three, including me, Tony and Kevin. Therefore, I would like to express my great appreciation to my groupmates for all the planning and coding for the project. Working in a group to develop a software is no easy task, especially for students without much experiences in workplace.

I would also like to thank our supervisor Dr. TW Chim for initiating the idea and guiding us throughout the development of the system. Thanks to Dr. Chim's timely advices to our system, more features has been added to the system time by time that make it a more completed system. I would also like to express my gratitude to the Department of Computer Science of the University of Hong Kong for all the support and resources.

Usage of third-party tools

Several development frameworks are used and some libraries have been imported to implement the instant event sharing system respectively for both the mobile and the web application.

Mobile applications:

(Refer to the report of my groupmates Tony and Kevin)

Web applications:

- Google Maps JavaScript API
- jQuery library, version 3.3.1
- React.JS framework, version 15.0
- Babel-preset-ES2015
- Webpack 4.4.0

Table of Contents

List of Figures

Chapter 1
Introduction

People always want to get instant information of different places: A hungry boy wants to know the immediate situation of restaurants nearby to avoid a long queue; Girls who planned to shop would like to know if the place was already crowded. Although there are already many popular communication applications for users to exchange information, we still find it hard to crave for instant information of certain places.

To satisfy all kind of curiosities, we will develop an instant information sharing system for requesting or uploading instant information by location and category.

The deliverable of this project will be a real-time application to provide a platform for users to ask for the immediate situation of wherever they are curious about and retrieve the expecting information that are shared by users at certain places. It is hoped that the application will soon be widely used, so that it could satisfy our curiosity.

1.1. Background

End-to-end information sharing has long been one of the main purposes of the development of information technology. Thanks to the breakthrough of the development of internet, there were numbers of online platforms for internet users to share information such as blogs, forums and various public websites. The Internet serves as a large public “storage”, from which user can search for whatever they want.

However, most of the websites and applications we are using nowadays are “static” [5]. That is, user has to send a request for new information such as pressing the key “F5” to refresh a certain website. This is an unpleasant single-user experience, in which contents only change when user makes a request to reload the information [2]. Thus, we believe that a real-time service would be much more engaging.

In this regard, our system takes references from existing applications which real-time service, especially a real-time map, provided.

1.2. Motivation

Instant event sharing provide a way that people are able to get what they want instantly. This fantastic feature can enhance the connection between people. Yet implementing a satisfying instant event sharing system is not an easy task.

Most existing communication applications use point-to-point approach to exchange data in an efficient way. This approach requires users at two nodes know each other in advance or hold the personal information of each other. For example to communicate in WhatsApp requires the phone number of the app user in order to send a message to that user. For Facebook, it is necessary to know the profile (Facebook name, email address or phone number) of an user if you want to send messages to that user. In this way, if we want to get some instant information of a certain place, for example I want to know the situation of Mc Donald's in Mong Kok, I have to know one thing in advance: A friend in my friend list who are currently at Mc Donald's in Mong Kok. In fact, it is almost impossible to have friends in different places in the world who can provide you instant information of that place. Thus, it is not the most ideal method for an instant event sharing system.

Based on the brief analysis above, we would like to develop a well-functioning instant event sharing system that solve the above-mentioning difficulty.

1.3. Scope of Study

In this project, we focus on the performance of sharing tons of instant information from a large amount of users. By the use of categorizing mechanism, various kinds of events can be shared through the system, such as a road condition in a specific place or the queuing information of the Food Expo.

The system architecture of the application is a client-server architecture in which there are three client programs, iOS, Android and web versions and there is a back-end server program for storing data and doing cloud functions. It has five important features, including the request-and-response function, media-sharing function, category system, the authentication system and the search engine.

1.3.1. Request-and-response function

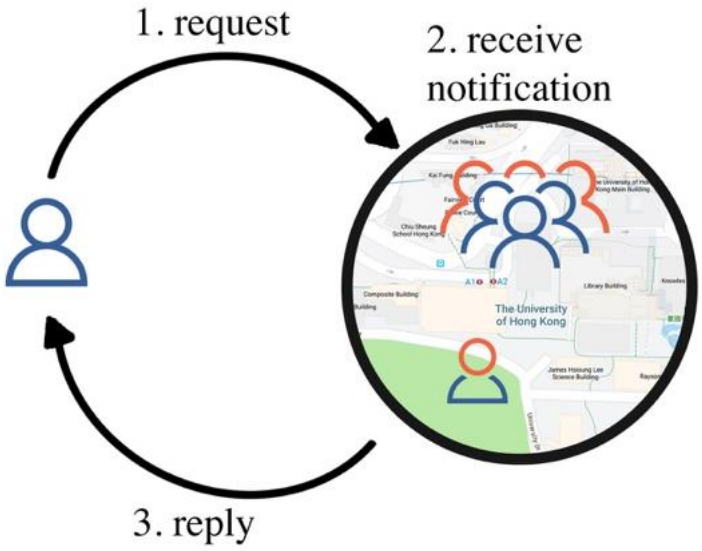

Figure 1 A top-level view of the requests-and-response function

The main feature of the system is the request-and-response function. With this function, users are able to request for whatever they want and notify others to respond. Moreover, the system will automatically notify the requesting user when a response comes. A top-level view of the approach is shown in Figure 1. User send a request for craving any information of a certain place. Then, other users nearby that area receive a notification of the request. After that, they can reply to the requesting user by taking photos with the details of the events happening at that area.

1.3.2. Media-sharing function

The second feature is the media-sharing function. A post generally consists of a description of the information, photos related to the information, and other minor information such as the number of votes, the creation time of the post, the user who created the post etc. The post can be publicly accessed by all users, and they can comment on the post as well as showing their opinion by up-voting or down-voting the post. As the system focuses on instant information, all the information uploaded should be new. In order to ensure the photos that being uploaded are new, the system limits that the photos must be directly captured using the app. In addition, voting and comment functions are implemented alongside with the media-sharing function. Every user can vote and comment on every post.

1.3.3. Category System

The category system is implemented to classify information into customized locations and genres. With the category system, the searching result can be optimised. For example, a post created with location “51. Nathan road” is categorised into “Hong Kong – Mong Kok”. If another user wants to know the road condition of Mong Kok to make a path-decision, the user would probably search for “Mong Kok”. The post above is closely relevant to “Mong Kok” and thus is shown in the search result. Moreover, Subscription is implemented to allow user subscribes to their interested categories. All posts under the subscribed category will be shown in the post feed of the user.

1.3.4. Authentication system

The fourth feature is the authentication system. The first function of the system is user-registration. To encourage users adopted into the system (actively provide instant information of different places), only registered users can view the details of a request and response. For non-registered users, they can only read posts created by users but not included any requests or responses.

Besides user-registration, the authentication system also serves as a management tool to keep up users of their activities. Users who tried to post explicit information would be warned (or blocked) by the system. Blocked users will be deprived of all the functions of the system such as commenting, voting, requesting and responding. This is to prevent users misusing the system.

1.3.5. Search engine

The search engine is implemented to optimise the search experience. Three genres are defined for users to search. The first genre is “User”. It is believed that some active users who contribute a lot to the system will be followed by other users. Thus, a single user can be search via the search engine. Searching the email or username in the field will result in the profile of that user if the provided information matches an existing user. The second genre is “Category”. This is designed for user to subscribe to their interested categories. With the help of the category system, user can search related words to get the specific category. For example if user enter “basketball” in the search field under the genre “Category”, then “sports” and “basketball” would probably be the search results. Furthermore, users can subscribe to the specific category in order to receive all posts under the category without searching the category again. The third genre is location. It is similar to the search engine in google maps.

1.4. Deliverables

The deliverables are a mobile application as well as a web application. The name of the application is “happens”. It can be freely downloaded from the Apple App Store and also Google Play Store. A web application has also been implemented, which is the web version of the system. The web address to the website is https://happens.hk/.

1.5. Outline of the report

This paper describes the full progress of the implementation of the system from inception to completion. The report is organised as follows. The first part of the paper will include the methodology of the system and of the implementation of the system. The second part will include the current progress of the project, some problems encountered during implementation and the respective solutions. The third part will give the summary of the project and the future tasks that increase the diversity and scalability of the project. The last part of the paper will include the workflow of different features with screen shots and descriptions.

Chapter 2
Methodology

In this chapter, a system of methods used in the implementation of the system will be introduced. There are various tools and programming languages used to develop mobile and web applications. The clarification of the choices of development techniques and tools the system adopted will be describe in details.

2.1. Platform setup

The platform setup makes an application-specific program configuration for an ideal working environment. It defines and implements the platform settings in the most optimal way. The setup is divided into three parts: back end, front end and version control.

2.1.1. Back End

The back end of the system includes the server that runs both client and server programs, the web services provided by different authority. The details are as follows.

The server is provided by Amazon Elastic Compute Cloud. The server runs in the operating system Ubuntu. The reverse proxy server used is Nginx. The domain of the web application is provided by HKDNR and the domain name server hosting service is provided by Namecheap, which is an ICANN-accredited registrar. The server program is written in Python.

2.1.2. Front End

For the mobile platform, the Android application is written in Kotlin using Android Studio. Android Studio is the official integrated development environment (IDE) for Android application.

For the web platform, the user interface of the web application is built using React.js, which is a JavaScript library that is used to create a single page application (SPA). With React.js, developers are able to easily reuse customised components to handle a large amount of data. In addition, a single page application improves user-experience because the browser does not have to refresh the full website in order to make a partial refresh on the webpage to retrieve updated information. This

reduces the webpage loading time to a large extent and thus make the application more interactive. There is no particular choice of the development environment but the web program will be deployed to the Ubuntu server upon the completion.

2.1.3. Version Control System

The version control system is important when it comes to cooperating a software by multiple developers. The version control system used in the project is Git. It helps tracking changes among multiple computer file to achieve source code management. Moreover, all the documentation files are stored in Github for developing purposes.

2.2. Database & Storage

As for database architecture, the system adopts both relational and non-relational database for different purposes. For relational database, MYSQL is used to manage all the data of the system, including all temporary or permanent data shown in the application. For non-relational database, Firebase is used to manage all the real-time data of the system. This approach is used because Firebase works well for real-time data. It provides API to listen and keeps track of the changes of real-time data. When there are new data or changes in data, the listener function will be triggered, and the event handler will manage the data appropriately.

Furthermore, Firebase pushes the update of the data automatically. As a result, concurrent updates are ordered chronologically and the system will not have data concurrency issue. It is not needed for developer to design an algorithm for safe data update. Firebase Real-time Database is a NoSQL Database in which data are organised in a key-and-value pair. With a well-designed data structure, same piece of data can be stored in multiple nodes. This technique is useful in a way that the data

can be stored in form of exactly how they will be retrieved later from client program and shown on the user interface. Thus, the server can retrieve the data directly without using SQL to join multiple tables for a query which is relatively inefficient.

As for the storage, the photos are stored in Firebase Storage. Since the file size of the original photo uploaded by users is relatively large, the performance of displaying such large images in client user interface would not be good. To make the display images smaller, the original photos are resized into thumbnails after storing into the storage. Reference links to these photos are then stored in MYSQL and firebase database in order to relieve the load of database.

2.3. Authentication

The system uses Firebase as the service provider for authentication and authorization. Authentication service from Firebase is used to manage validating user credentials and establishing the identity of the user while authorization service from Firebase is used to handle access restrictions, such as the permission of users' access to different features of the system.

In addition, Security is one of the most concerning points in the system. Since the system requires sensitive data from the user (such as user's location), security measure has been done accordingly to protect users' personal data. The location of each user would not be stored in the database to reduce the chance of data breach. The system asks for user permission before getting the user location. Users can deny the permission at any time and change the setting in the application. Safety measure has also been done to prevent leakage or unnecessary disclosure of personal data. Therefore, the system is privacy-preserving in a way that the privacy of users is preserved.

2.4. User Interface

The design of the user interface (UI) is important to attract more users to use the system. An appealing interface is necessary as it provides comfortable experience and impression to the users. Hence, the front-end application prototype has been made in the early stage of the project. The prototype displays a simple UI design of different pages.

Figure 2 The text view of the posts

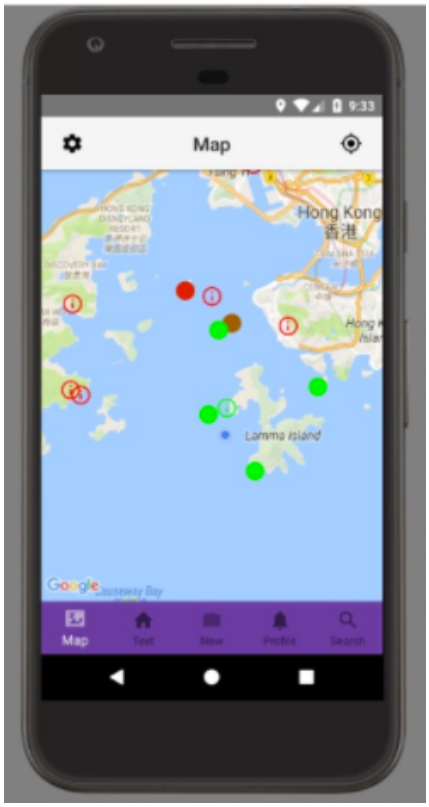

Figure 3 The map view of requests, responses and posts

For the mobile application, in each page there are a main panel and a bottom tab-selection bar in each page. This design is consistent throughout every page of the application. The post view and map view of the system are used to be the examples.

For the post view, the posts are sorted one by one in a card form (see Figure 2). In each post, information such as location, description and category of the post are shown along with the image. There are buttons below each post for users to vote, comment, or share the post. This is a likewise design to other communication application. Hence, users could smoothly adapt to this new system. Furthermore, users can efficiently explore posts they would like to read by entering searching criteria on the top search bar. The posts below will then be filtered correspondingly.

For the map view of the system (see Figure3), the main panel is a map with markers. The markers cluster the events happening in adjacent locations and user can click on the markers to see what is happening around. Moreover, user can see the requests on the map. User can click on the marker and choose to respond to the requests.

The system uses common UI designs appearing on most of the existing applications presently, therefore, users can learn to use this new system in a short period of time.

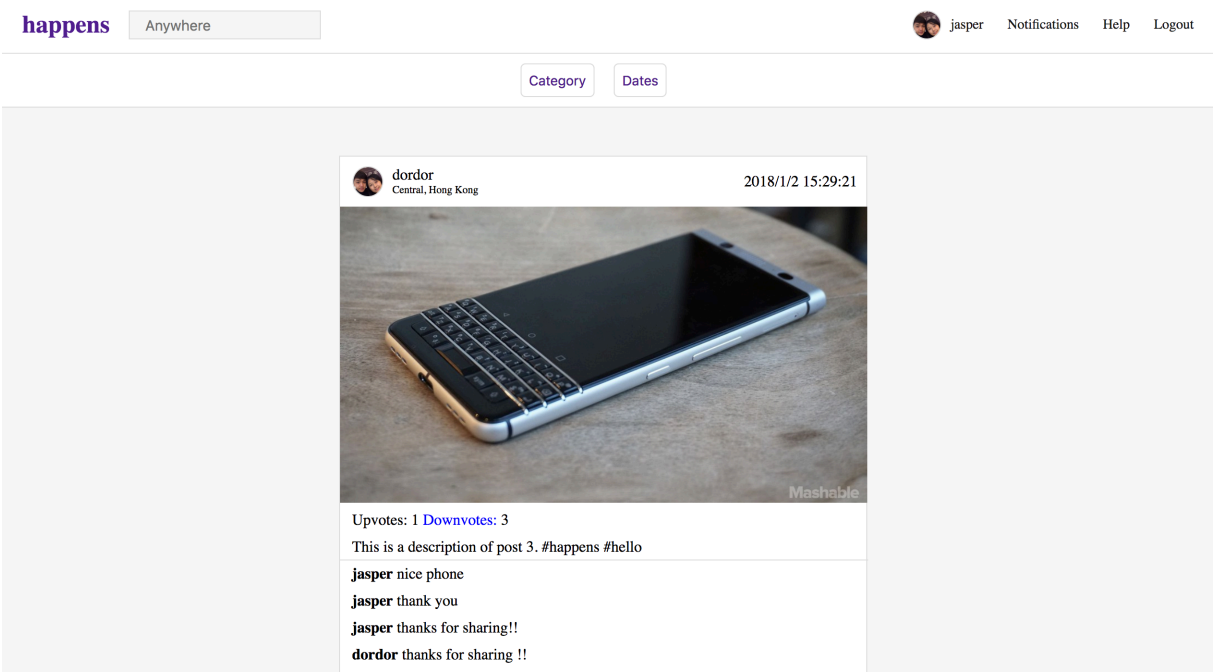

Figure 4 The home page of the web application of happens

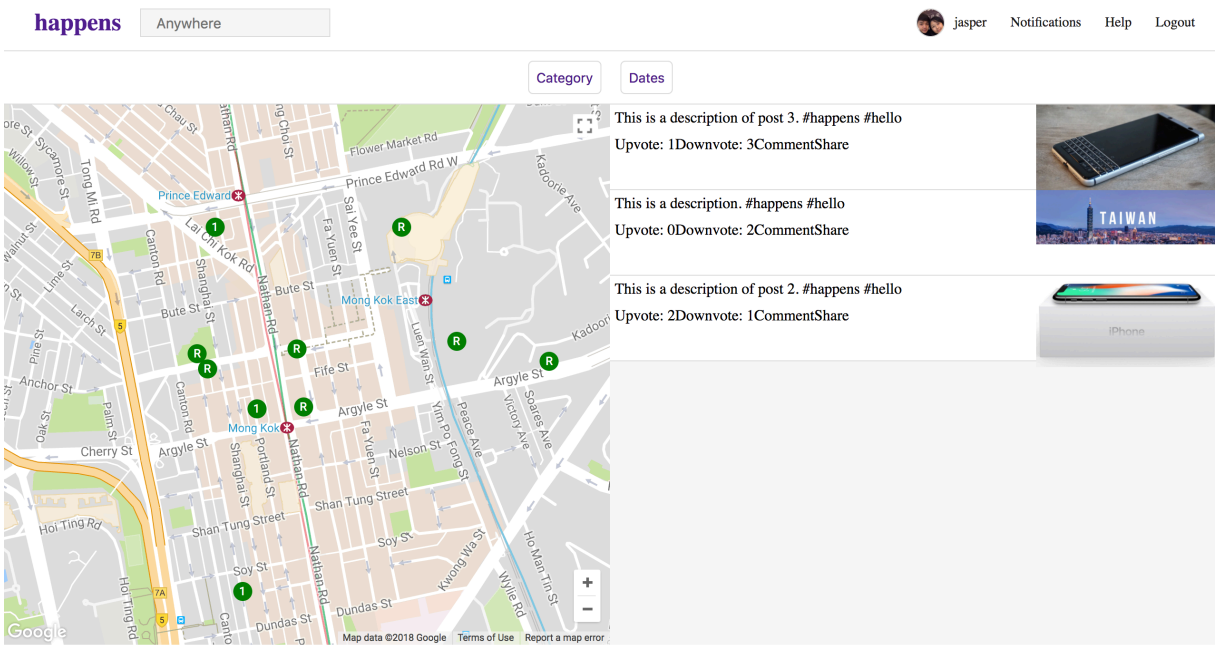

Figure 5 The map view of the web application of happens

For the web application, the two main pages are selected to introduce. The Home page is shown in Figure 4 and is designed as follows. Firstly, on top of the page there is a navigation bar in the header. It controls what to display in the main panel. For the management of user's profile, the username at the navigation bar on top-left position can be clicked to adjust user's profile information. In addition, users can view their request content in the profile page. For notification tab next to the username, it can be clicked to view the notification from the system (such as the notice of receiving a response from another user).

Secondly, below the navigation bar is the filter bar. It is used to add the filter the posts shown below the bar. Initially, all posts under users' subscription will be shown in the main panel. By setting the filter options such as selected categories and a particular period, only posts related to the filter options will be shown in the main panel.

Thirdly, the main panel shows a post card. At the top of the card there are username of the post writer, location of the post and the related post-creation time. At the middle of the post card, images are display in this

area. For multiple images in a post, clicking “>” and “<” button can move to the next and the previous images respectively. At the bottom of the card, there are voting details (e.g. the number of up-votes and the number of down-votes) and comments.

Besides the Home page, another main page of the system is the Map page (see Figure 5). On the left of the main panel, google maps is rendered with different types of markers. Generally there are two types of marker.

1. The “Post” marker: For posts that are relatively new, its colour is green. For posts that are relatively old, its colour is red.
2. The “Request” marker: The request marker has “R” on top of the marker. If the request has been answered by a particular user, its colour will turn into red.

In addition, on the right of the main panel, there are the details of each post on the map such as the description and related images.

2.5. System Architecture

In this part, four main flows of the system will be included to introduce how different components of the architecture interact with each component.

2.5.1. Authentication

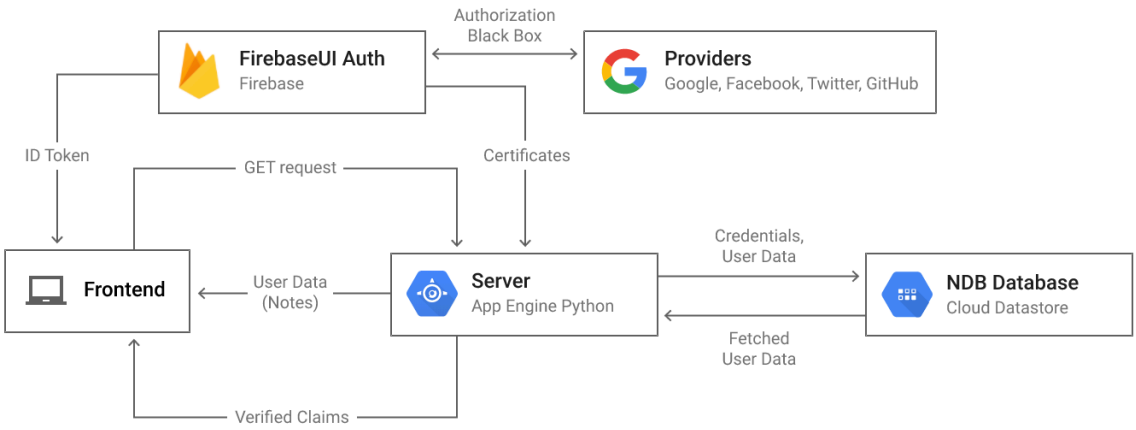

Figure 6 A top-level approach of authentication

The flow of the authentication is shown in Figure 6. Firebase verifies user's authentication and returns the ID token to client side upon successful authentication.

Firstly, the client program initiates the flow by log in and redirects the browser to Firebase, so the user can authenticate by different methods such as Google, Facebook, Twitter and GitHub.

Secondly, Firebase authenticates the user. If it is the first time the user goes through this flow, then a consent page will be displayed. Also in the consent page, the permissions that will be given to the system are listed. Thirdly, Firebase redirects the user to the application with an ID token in the hash fragment of the URI. The system can then extract the tokens from the hash fragment. In a Single Page Application (SPA) this would be done using JavaScript and in a Mobile Application this is typically handled by interacting with a Web View.

Last but not least, the app can use the ID token to call the API on behalf of the user.

2.5.2. Request-and-response

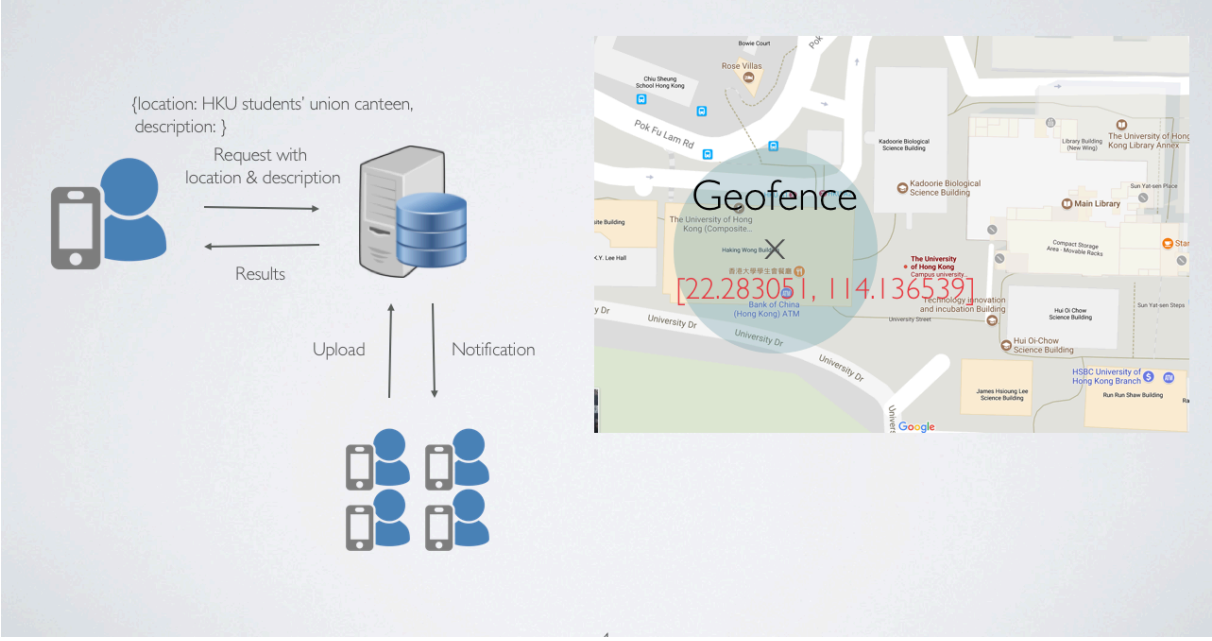

Figure 7 A top-level view of the flow of the request-and -response function

The approach of request-and-response is shown in Figure 7.
Once the user has logged in. The user granted the permission to send a request to the server with location name and description. In this example, a canteen in the University is used as the location. When the server receives the request, it will make use of Google Maps API to convert the location name to a corresponding longitude and latitude. The next step is to send a notification to the users nearby. Geo-fence is defined by the system. It is geographical boundaries on top of the map. When a device enters the boundaries, an alert is issued. The server creates a Geo-fence in which the centre is the location of the canteen in the University of Hong Kong. Users inside the Geo-fence are notified by the system. They could take a photo and upload to the server. Finally, the system answers the request.

2.5.3. Firebase notifications

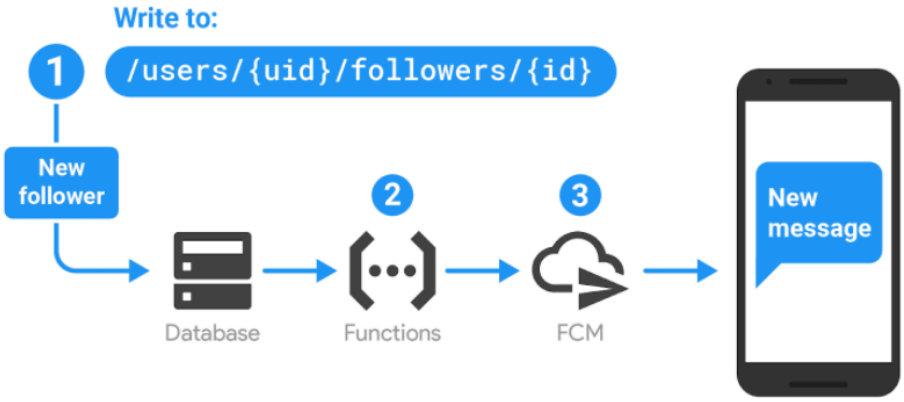

Figure 8 A top-level view of the flow of sending a notification

The flow of Firebase notifications is shown in Figure 8. Firebase cloud messaging is used in the system. Notifications is an important feature of the system because any requests should be sent to the users within the requesting location in order to ask them for the information. The best way to notify them is a message alert from their phone screen, especially when they are offline (not using happens). The following is the flow of the Firebase notifications.

Firstly, a user requests instant information of a particular location via happens.

Secondly, the sever updates the “requests” node in the real-time database in Firebase once the server received the request.

Thirdly, the update in Firebase will trigger a pre-defined cloud listener function in the server to send a cloud message.

Finally, the cloud message is created and send to the users nearby the location to notify them of the request details.

2.5.4. Photos upload

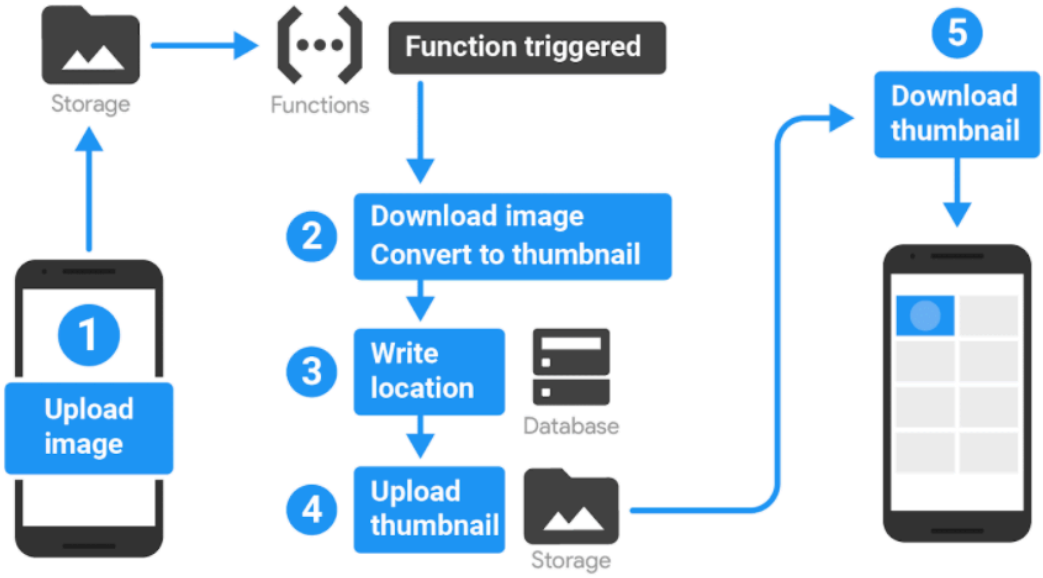

Figure 9 The flow of converting the original images to thumbnails

The flow of photos upload is shown in the Figure 9. As mentioned in section 2.2., photos upload by users are needed to resize into thumbnails in order to display a smaller-size image to the client user interface. The flow of making the thumbnails and storing the relative reference links are as follows.

Firstly, an user uploads an image to the storage of the server. Secondly, Firebase storage detects the addition of image upload by the user, an listener function is then triggered. The server automatically downloads the images from storage and then converts it into thumbnail. Thirdly, the reference link of the thumbnail is returned from the listener function and then it is written into the database.

Fourthly, the server uploads the thumbnail back into the storage after the addition of record in database.

Finally, when client program access the image, it is the thumbnail being downloaded in the client program but not the original image.

Chapter 3

The Progress

This chapter will focus on the current progress of the project. The progress is generally divided into two major parts. The first part is the implementation of the system, including both client and server programs. Moreover, the server configuration is also included. The second part of the progress is about the work related to test the system. This includes test harnesses for automation of tests, unit testing along with boundary value analysis. The result is satisfactory. In addition, this chapter will also include the problems encountered during implementation. Some solutions were adopted to improve the user-experience in happens.

3.1. Accomplished tasks

All the features and functions stated in the requirement specification have been completely implemented and tested. However, some adjustments to the system has to be made when there are more users using the system. In the following parts, the details of the implementation as well as the testing procedure will be introduced.

3.1.1. Mobile application

For the details of the implementation of the mobile application, please refer to section 3.1. of the report of my teammate Tony and section 3.1.3. of the report of Kevin.

3.1.2. Web application

For the web application, the front-end is built using React.js.
The main features have been implementation. The details are as follows.

3.1.2.1. Requesting for instant event at any place.

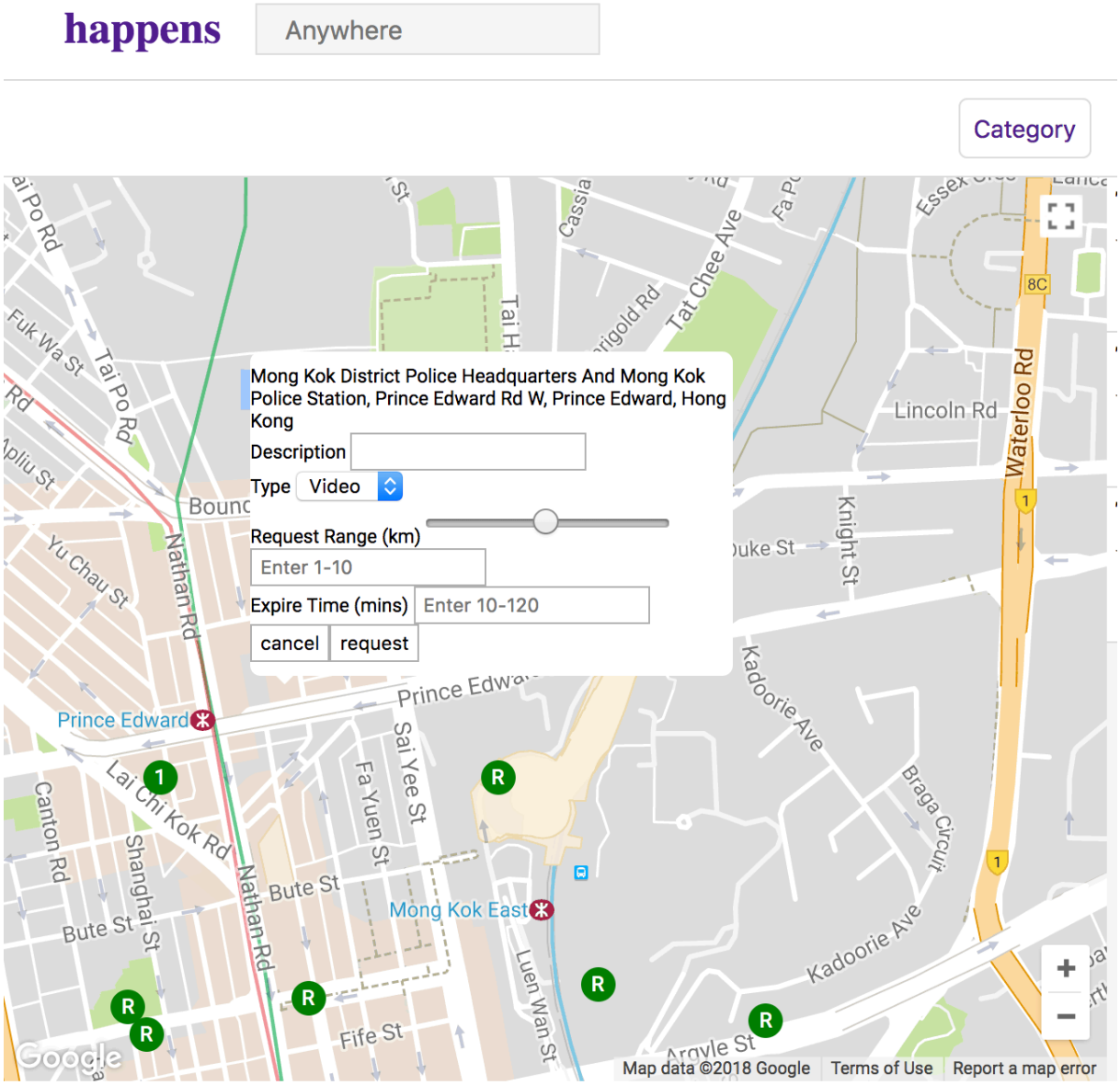

Figure 10 The request form in the map

In the Google Maps, users can make a request by clicking on the map. The request form will appear with the location of the point being clicked at the top of the form (see Figure 10).

Moreover, users are required to write a description of the request (such as the type of event). Considering the requesting point as a centre of a circle, the request range is defined as the radius of the circle such that other users inside the defined circle will be notified. The range is limited to 1-10 kilometres so as to ensure the performance. If the radius is too large, the circle (That is the Geo-fence) will be too large and other users who are far away from the location will be notified. If the radius is too small, only a minority of users will be notified and the requesting user may not be able to receive a responses.

In addition, since the application focuses on instant events, the events shown on the maps should not be outdated. Therefore the expire time is designed and is limited to 10-120 minutes, which means the request will only last for 10 to 120 minutes. Once the required inputs have been filled, users can send the request by clicking the “request” button.

The back-end flow described in section 2.5.2. has also been implemented. Thanks to the real-time service provided by Firebase, the new request is shown in the map automatically without a browser refresh.

3.1.2.2. Authentication & Authorization

Login to the system is required if users want to make a request or respond to a request. In this regard, the authentication provided by Firebase has been implemented in the system. The registration method used in the system is Email-and-Password and a username. A confirmation step is required after signing up for new accounts.

Email-and-Password method is used because this requires less personal data as possible. From the users’ prospect, providing less personal data to an application results in a better protection to their privacy. Other

registration methods are not considered by the time because those requires lots of personal data, such as Facebook login, Gmail login and so on.

Once the user have signed up for a new account, the server sends a confirmation email to the user for authentication. When the user is authenticated by the system successfully, the user grants the authorization to perform different actions in the application.

3.1.2.3. Search engine

The search engine is implemented in the web application. Three tabs are provided for searching: Location, Category and User. Searching texts under each tab can retrieve the relevant posts and user profile respectively. Moreover, users are able to search for posts within a particular period. The following describes the details of each tab.

Regarding searching by location index, the default location in the map is user's current location such that users can view any events nearby. In addition, the search box is interactive. It automatically shows the relevant places when users is typing inside the box. For example, when users enter "Mong" in the box, some related locations, such as "Mong Kok Hong Kong" and "Mongolia", are suggested in a list-form under the search box. When users entered a location name in the search box and submitted the form, a Google maps view will be centred at the entered location. Posts nearby this central point will be shown. On the right of the map, the posts are shown in detail (see Figure). The order of the posts is from the greatest to the smallest timestamp (i.e. from the most recent to the oldest).

As for searching by category index. The system allows events from all categories. Ranging from music, sports, transport, to entertainment and so on, users can share whatever they want via the system except inappropriate content, which will be discussed in section 3.2.3. However, the posts will be in a terrible mess if all types of posts are shown together in the user-interface. And this is the reason for the implementation of the search by category. When users entered a category name in the search box and submitted the form, only posts under the entered category will be shown on the map view and the text view.

Last but not least, searching by user is also implemented so that users can find the profile of a particular user to view all the posts created by the user. This is similar to YouTube. User can search and subscribe to another user in order to keep track of the latest video from that user. In our system, it is believed that there would be active users who usually share interesting things and other users would like to follow their status. In this regard, user can search for a user name. When users entered a username in the search box and submitted the form, the suggested user with similar name will be shown and their profile can be accessed upon clicking.

Moreover, user can subscribe the searched user. For all the user's subscription, an alert is issued when the subscribed user makes a new post under the subscription.

3.1.2.4. Post-sharing with voting and commenting

The post-sharing function is completed. User can create posts by taking photos and writing a description of the post. The new post will be shown on the map represented by a marker. Upon hovering the marker the description of the post will be displayed on top of the marker.

Moreover, the voting and commenting alongside with the post has also been completed. Each user can express their opinions by up-voting or down-voting on the post. The number of votes reveals the degree of concern that other users pay on the post. As for commenting on a post, each user can post a comment on the post to show their opinions. Moreover, user can also reply to a comment in a post for a discussion purpose.

3.1.2.5. Server-side configuration

The server set-up is completed.
Firstly, the cloud functions are implemented to perform several actions. The main function in cloud function is to send Firebase cloud message. It is triggered when there are new requests sent to the server.

Secondly, databases are also set up for data storage. MySQL is used for all kind of data while Firebase real-time database is used to manage real-time data. For the requesting function, a request data will first be sent to MYSQL to store all relevant information related to the request such as the request ID, description, the request type, the request range and the expire time. After that, the request ID will be sent to the Firebase real-time database. When the real-time database detects a new addition of request ID in the “request” node, the cloud function mentioned above will be triggered and the cloud message will be made and sent to particular users.

Moreover, an Amazon Elastic Compute Cloud (Amazon EC2) instance has been set up. Its operation system is Ubuntu and it has all programs for running the application. The domain of the website is https://happens.hk ,which was purchased and managed in https://www.hkdnr.hk/.

3.2. Problems encountered & solutions

In this section, we will introduce some problems we encountered during the implementation. The problems generally related to some performance issues.

3.2.1. Category suggestion list for images

This problem is about providing a category suggestion list for a photo. The starting point is taking a photo in our application. When users are planning to create a post or answer a request, taking photos in the application is required. After taking a photo, it would be interactive if the system could provide the photo a suggestion list for the category.

The suggestion list should contain the categories which are closely related to the objects in the photo. Thus, the ideal solution would be applying the image content analysis with vision API. Through the image content analysis, objects with a high probability of appearing in the image are returned. Moreover, a machine learning model is needed to classify the returned objects into particular category. Finally, a suggestion list of categories related to the image can be returned to the user interface.

An alternative solution of providing a suggestion list for the category is to return the user's subscription list of category. It is believed that the probability that the post's category belongs to one of user's subscription list would be high.

3.2.2. Handling huge amount of requests in the same location

This issue is interesting and challenging and appears when there are more users requesting via the system. Consider a case that there is an annual book exhibition in Wan Chai, Hong Kong. Many people would be curious about the situation in the exhibition and would send a request to the system time by time. A problem will be induced that there will be lots of request in the same location. The leading effect is threefold:

1. There will be lots of notifications being sent to the users in the exhibition that requesting for the same event.

2. A huge number of markers will be shown on maps for the same requesting event, making the map not user-friendly.

3. Most requesting users will not receive a reply because answering all the same request one-by-one would not be practical.

For the first effect, the system would display other nearby requests to the requesting user before the user submits a request. Moreover, users can follow the request from another user so that they will receive a notification when there is a response to the request. In this way, users can follow the request with the same event (“Book exhibition”) instead of make a new request.

For the second effect that lots of marker of the same request being shown on the map, the solution is to do google maps clustering (in other words, grouping markers that are close to each other). In this way, only one marker will be shown on the location of the exhibition to represent all the requests.

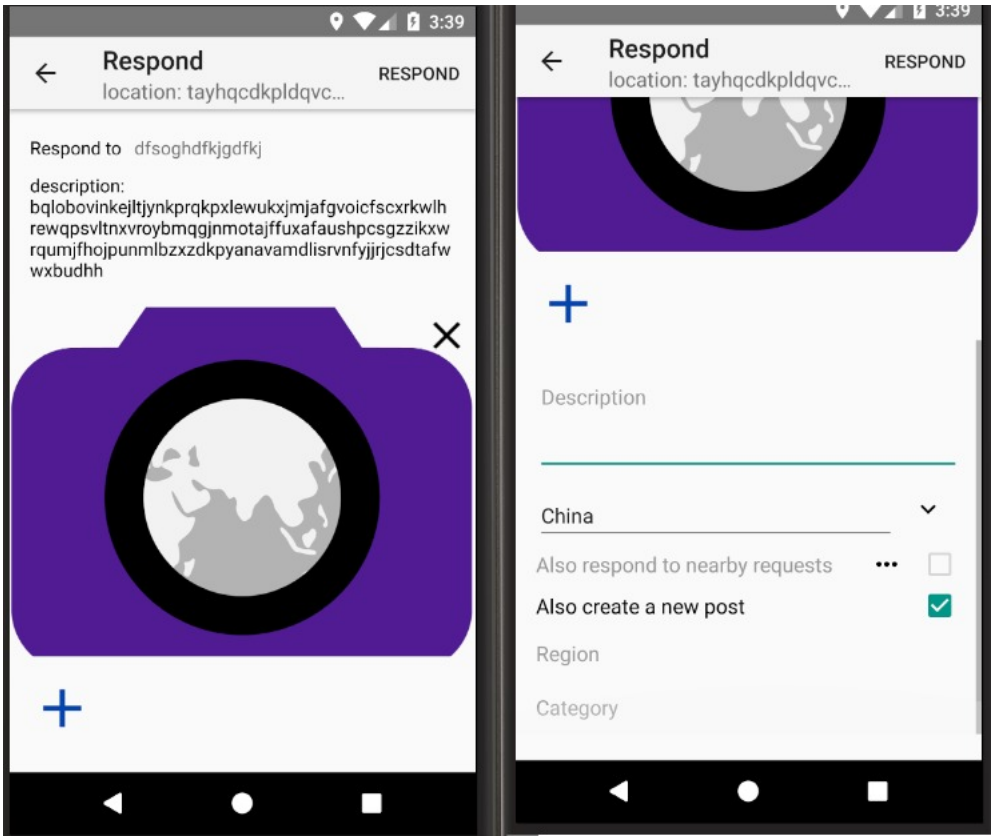

Figure 11 One-to-many responding implementation

For the third effect that most of the requests with the same description will not be answered, a one-to-many approach is implemented to solve the problem. Instead of answering one request at a time, users can choose to answer other nearby requests at the same time when they are answering a request (see Figure 11). There is an option “Also respond to nearby requests” for users to answer similar requests from other users.

3.2.3. Handling arbitrary requests and posts

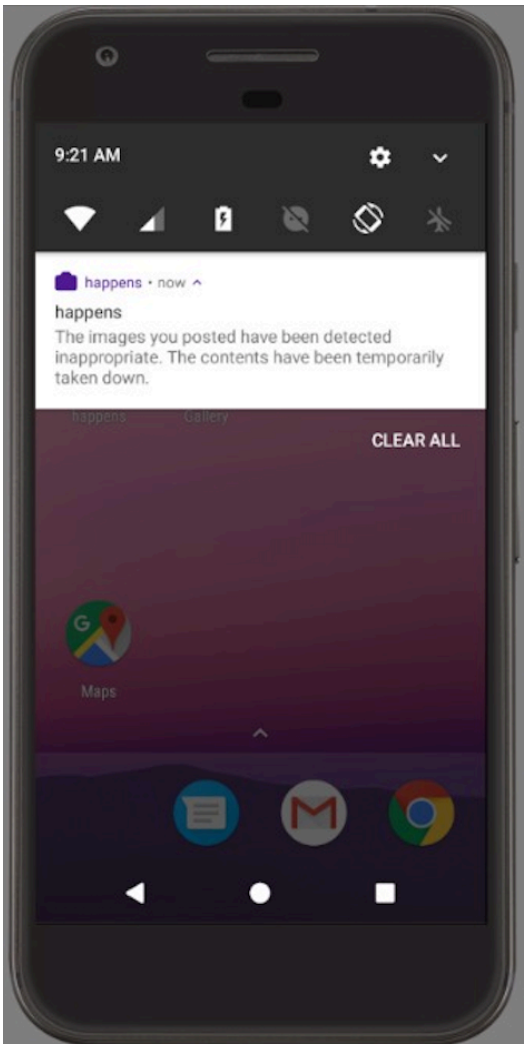

Figure 12 The notification of inappropriate images being taken down

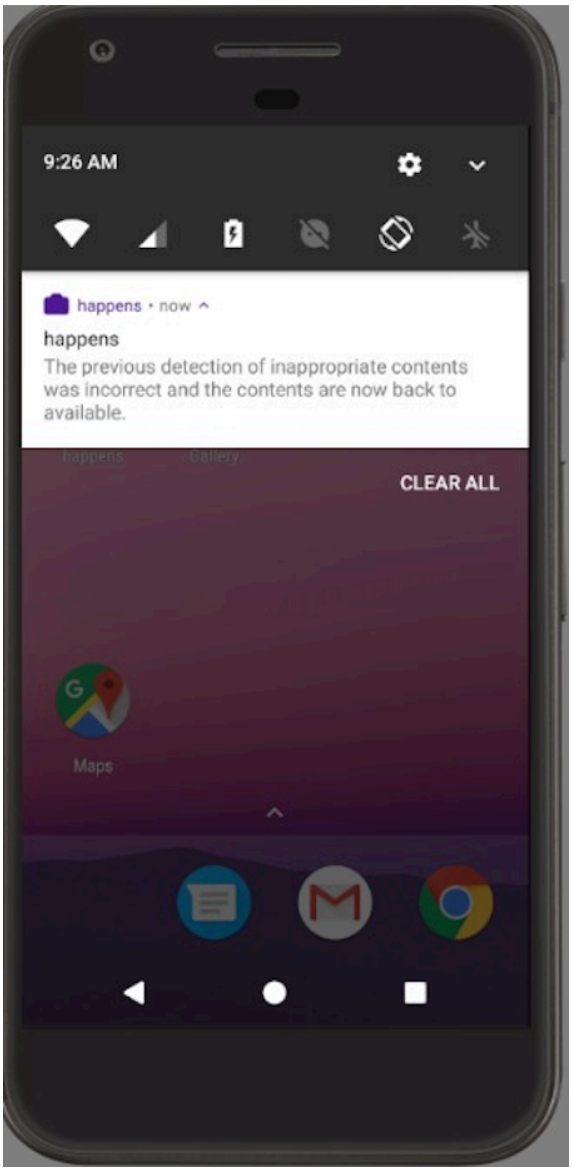

Figure 13 The notification of inappropriate images being determined appropriate

Although the system is designed for normal uses, there may be lots of arbitrary requests and posts created by some naughty users. For example, making some requests with meaningless description deliberately or posting some meaningless content and images. In this regard, the system has set a quota for the number of requests available for each user every day. With limited quota, user cannot make lots of arbitrary requests and posts.

Moreover, a report mechanism is used in the system to report arbitrary requests, posts and comments. A notification of the report will be sent to the user if the system received a report. After that the system will decide if the content is indeed inappropriate. If the content is determined “inappropriate”, the system will remove the relevant data and send a notification to the user (see Figure 12).
If the content is determined “appropriate”, the system will mark it as “available” (see Figure13).

Chapter 4

Review and future developments

In this section, the review of the system is carried out and also some possible future developments will be discussed. At the beginning of the project, the components in the system architecture are selected for a basic project, such as EC2, a basic virtual machine storing and running API. As the scalability and the functions of the system expands, more advanced server configuration is needed to maintain a satisfactory performance and provide great user-experience. The following sections regards various possible improvement that the system can adopt to perform better.

4.1. Migrating from EC2 to Elastic beanstalk & RDS

Amazon EC2 instance is used as the server of the program. It is basically a virtual machine running the API of the system. Moreover, there are python scripts to access the MySQL database. As the system is expanding more and more features, the database structure would become more and more complicated. Some queries might take lots of time to execute and return the result. In this regard, it is suggested that a migration of the EC2 instance to elastic beanstalk environment with Amazon relational database service (RDS) should be done. As a result, RDS makes it easy to operate and scale a relational database.

Moreover, RDS has cost-efficient and resizable capacity and automating time-consuming administration tasks, for instances, hardware provisioning, database setup, patching and backups [6].

4.2. Mobile data usage optimization

Since the system includes lots of images, it will use lots of data in downloading those images even through the original image has been converted into thumbnails. Hence, different approach should be taken to help user from saving mobile data.

For the application users who are connecting to Wi-Fi, images with the best quality will be downloaded in the client program. For the application users who are using mobile data, an alert should be issued to ask the users if they want to download the best quality images with larger sizes or the normal quality images with small sizes.

4.3. More features

The Karma feature is one of the interesting features that would probably be added into the system. Since the performance of the system highly relied on users' active participation such as sharing surrounding events, voting and commenting, an incentive feature would be implemented. For example, increasing the quota of requesting per day. If users actively respond to others' requests and accepted by the users, they can have the increase in quota of requesting per day.

Moreover, photo-editing may also be implemented in the system. It is common place for many popular social media applications to edit photos before sharing. Some animations, texts and colorings can be added on the photos in order to make it appealing.

In addition, a machine learning model will be built to analyze the content of different images. When there are more users sharing events in the system, the server will collect lots of images with the description and the category defined by the user. Those data are useful for data feeding to a machine learning model.

4.4. iOS application development

By now, the system only supports the Android version of the mobile application because Android has long been the most popular smartphone operating system in the world [7]. Yet the iOS users should by no means be neglected. An iOS version of happens will be developed and the features would be the same as the Android version.

Chapter 5
Conclusion

We build the instant event sharing system happens with mobile and web versions to let people request for what they are curious in. Moreover, several responding mechanisms were implemented for better performance. Voting and commenting features are include in the system to enhance the communication between users. Moreover, a report mechanism is designed to get rid of any explicit content. When the number of users increases, the system also serves as a tool for data collection in the image aspect.

Reference

[1] Firebase helps you build better mobile apps and grow your business.
Retrieved from: https://sjorshooijen.io/firebase-helps-you-build-better-mobile-apps-and-grow-your-business/

[2] Real-time application (RTA).
Retrieved from:
http://searchunifiedcommunications.techtarget.com/definition/real-time-application-RTA

[3] C. Anderson, Why You Should Consider React Native For Your Mobile App, 7 April, 2016
Retrieved from: https://www.smashingmagazine.com/2016/04/consider-react-native-mobile-app/

[4] J. Friberg, REACT NATIVE VS NATIVE IN MOBILE APP DEVELOPMENT, 6 July, 2017
Retrieved from:
https://www.varvet.com/blog/react-native-vs-native-in-mobile-app-development/

[5] T. Greene, “Why your app needs to be real-time “, 5 April 2013.
Retrieved from: https://venturebeat.com/2013/04/05/why-your-app-needs-to-be-real-time/

[6] Amazon Relational Database Service (RDS)
Retrieved from:

https://aws.amazon.com/rds/?sc_channel=PS&sc_campaign=acquisition_HK&sc_publisher=google&sc_medium=rds_b&sc_content=rds_e&sc_detail=amazon%20rds&sc_category=rds&sc_segment=186844484602&sc_matchtype=e&sc_country=HK&s_kwcid=AL!4422!3!186844484602!e!!g!!amazon%20rds&ef_id=WmITKQAAALGIIVHa:20180415084324:s

[7] Android Still More Popular Than iOS in U.S.
Retrieved from: https://www.huffingtonpost.com/entry/android-more-popular-than-ios_us_5678203be4b06fa6887de2e