Final Report
for
a Privacy-preserved
Instant Event Sharing System

Kwok Ching Hei Kevin (3035185031)

15 April 2018

Abstract

There are many forums and social media in the Internet presently, but no particular website focuses on location and real-time event information sharing. This project was initiated by the insufficiency of functions in the existing applications. The system aims to provide a platform for users to exchange the most updated information of specific locations.

This paper describes the design and implementation of "happens", the instant event sharing system. It has five important features, including the request-and-response function, photo sharing function, category system, the authentication system and the search function. The request-and-respond function let user inquire information based on location and the system connects users in the world by location. The other functions enhance the system.

The ultimate product of the system is to build a mobile application as well as a web application. The front-end application is written in Kotlin for Android, while the web version is written in React.js. Moreover, the backend server providing the database and API service is located in a Ubuntu virtual machine based on Amazon Elastic Compute Cloud (Amazon EC2) web service. The authentication service and the notification service are based on Google Firebase.

This paper is the final part of the project, which concludes the whole development process. The development process is divided into design phase, implementation phase and testing phase. It has been fully developed in April 2018. After the development, the main priority is to maintain a smooth running of the application and enhance the system to provide more new features to attract more users.

Different problems are appearing during the development, such as the suggestion of category system. These problems are dealt with different solutions; however, the system can be improved in the future.

This report marks the final milestone of the project. There are limitations and restrictions on this final year project, thus, recommendations are provided as a step stone for future development on location-based sharing system.

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.

Contents

List of figures

1 Introduction

This section introduces the project. The overview of the project is discussed, as well as describing the structure of this paper.

1.1 Background and Motivation

In the era of instant information, people are curious about the events happening in the world: A driver would like to know the traffic condition of his destination; a girl would like to know the queuing situation outside a shop selling limited edition bags. In the current development of information technology, end-to-end information sharing is a major focus. There are different blogs, forums and social networking sites providing a platform for users to exchange information. However, popular communication applications, such as Facebook and Twitter, are not location-oriented, thus it is hard to get the instant information of a certain place. Moreover, the information shared in the above application cannot be verified and this leads to a low accuracy in information exchange.

For solving the above problems, an instant event sharing system for requesting and uploading instant information by location is developed in this project. The system is a real-time application which provides a platform for users to request for the immediate situations of a certain place and retrieve the expecting information. The information is provided by the users in that certain place. It is foreseeable that the system will soon be widely used and lots of information can be exchanged through it. It aims to satisfy all kinds of curiosity and we believe that hands make light work. Therefore, the world would be connected if every user is willing to share events happening near them.

1.2 Scope of Study

This project is to develop an instant event sharing system. Different events can be shared through the system, such as a traffic jam in a specific district or the queuing information of the Book Fair. The system is a client-server application. The client side has mobile and web versions; while there is a back-end server storing data and doing cloud functions. It has five important features, including the request-and-response function, photo sharing function, category system, the authentication system and the search function. This system focuses on information exchange among all users, so there is no private message function included in the application.

1.2.1 Request-and-response function

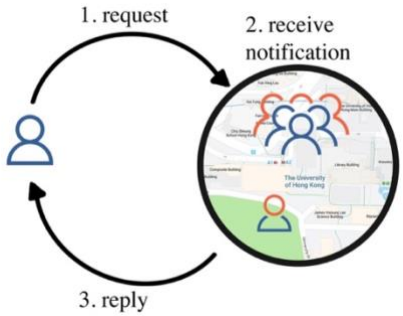


Figure 1 – System flow of the request-and-response function.

First, the request-and-response function is the main feature of the system. The system flow is shown in Figure 1. User can request for information of a certain place, and other users around that area will receive notification. They can then reply to the request by uploading images, showing the actual situation of the events happening at that location.

1.2.2 Photo sharing function

Second, it is the photo sharing function. Similar to other social media, users are allowed to upload photos and write description on the post. Moreover, user can categorise their post to specify what the post is about. The post can be publicly accessed by other users, and they can give comments as well as up-vote or down-vote the post. However, the system limits the photo uploaded must be directly captured using the app. This prevents old photos from being uploaded to the system because the system focuses on the instant event happening at the time, so only new image can be uploaded. Furthermore, voting and comment system are provided alongside with the photo sharing function. User can vote (similarly to like/dislike) and comment on each post.

1.2.3 Category System

Third, it is the category function. As our system are similar to a forum and it does not limit what user should post, a category system is developed to categorize the post. User can subscribe to any categories and read posts according to their preference easily.

1.2.4 Authentication system

Fourth, the authentication system register user for using the system. Although normal user without registration can use the system viewing the posts, the upload function is only limited to verified registered users. In addition, the system can block the user from using it for posting explicit images or giving inaccurate information. This prevents user from repeatedly misuse the system.

1.2.5 Search Function

Finally, there is a search function for the whole system which user can filter out the post by some keywords in the description of the posts. User can search for what they would like to read by providing the keywords.

1.2.6 Exclusion: Private Message Function

Private message function is a common function in forum or social networking sites. This can enhance the communication between users. However, our system focuses on information exchange among all users so there is no private message function. Users can respond and exchange any information by commenting in the posts.

1.3 Deliverables

Our project is to develop both mobile application as well as web application. It is named "happens" and it can be downloaded Google Play Store once it is completed and deployed to the public. A website has also been set up, which runs the web version of the system. The web address to the website is https://happens.hk/.

1.4 Structure

This paper mainly contains two parts; First, it introduces the system. The background of the system and the development process, which includes the methodology of the study, is discussed in the first part of the paper.

Second, the remainder of the paper concludes the project and introduces the future development of the project. The conclusion is discussed in the latter part of the paper. As this is the final part of the project, recommendation for future development is also discussed in this paper and the system can have improvements afterwards.

2 Methodology

This section is about the methodology of the project. Since this is a developmental project, the analysis of the set-up and design is discussed.

2.1 Platform setup

The system is client-server model application. The frontend application is deployed on both mobile and web platform; while there is a backend server for storing data and providing the API to the frontend application. In this section, the platform setup is discussed.

2.1.1 Backend server

The backend server is supported by the Amazon Web Service Elastic Compute Cloud (Amazon EC2). Amazon EC2 is a web service that provides secure, resizable compute capacity in the cloud, where developers can easily manage the cloud computing system [1]. Using Amazon EC2, the computing environment can be scaled up according to the size of the system and charged accordingly. There is a Ubuntu virtual machine running where the API server and the database is hosting on it. Also, NGINX is used as the HTTP server. NGINX is a more lightweight server compared to Apache, another famous HTTP server. The app engine is written on Python which respond to HTTP request and connect to database to return corresponding data back to client. The API server is publicly available on the domain https://api.happens.hk. The above domain has been purchased and managed in https://www.namecheap.com.

2.1.2 Frontend Client Application

There are mobile and web versions for the frontend application. There are some hardware and software requirements to ensure the system provides the same user experience in both versions. Although two different versions are set up for the development, the functionalities of the two versions will be made unanimous; and the user interface (UI) would be similar. This is because it is important to give the same user experience to any user, regardless they are using mobile or web version.

The mobile version is deployed in Android first. It is written in Kotlin language using Android Studio. Kotlin is the latest officially supported language for Android development. Comparing to Java, the most common language for writing Android

application, it has a lot of advantages such as null safe and smart casts [2]. Although it is a new language for us to learn, it has the interchangeability with Java so it is possible to have Kotlin and Java classes together within the same project.

For the web platform, React – a JavaScript library, is used to create a single page application (SPA). With this library, developers can easily make reusable components and handle a large amount of data. Also, a single page application can enhance the user experience since the client browser only needs to make a partial refresh on the webpage to retrieve updated information. This reduces the webpage loading time and make the application more interactive. The web application would also be deployed to the Amazon EC2 server upon completion.

2.1.3 Version Control

As there are different group-mates responsible for the implementation of the system, GitHub is used for the version control. GitHub is a web-based version control and source code management system widely used in software development. Our system includes personal data which should not be accessed by the public, so the repository is private and only accessible by us.

2.2 System Architecture

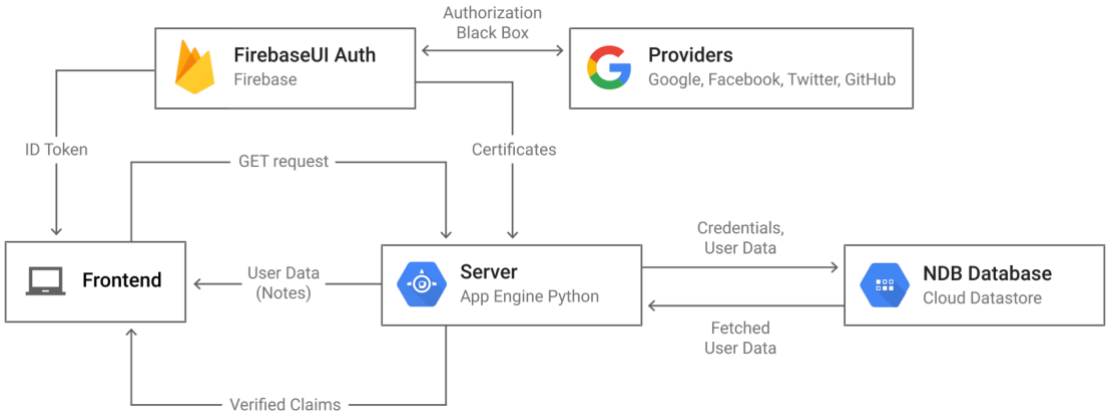


Figure 2 – Authentication flow using Firebase

A system architecture is the model that defines the system structure, describing how the system is organised to have different behaviours and interactions. As this system is a client-server model application, the architecture focuses on the interaction between client and server in order to perform the five features stated in the scope in section 1.2.

Figure 2 shows the flow of the system architecture. The client application sends HTTP request to the server. The server handles different requests with different logic. If the request is getting data from the database, it connects to the database and trigger corresponding SQL. The database returns the data and the server loads them into JSON format to return to the client. If the request is updating the database, it requires authentication token from Firebase. The authentication and the details of the database is discussed in section 2.4 and section 2.5.

2.3 User Interface Design

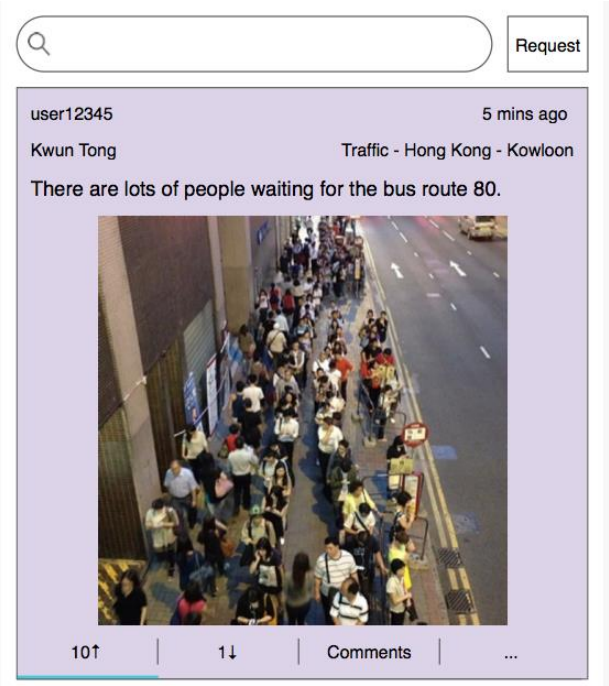


Figure 3a

Post view of the application

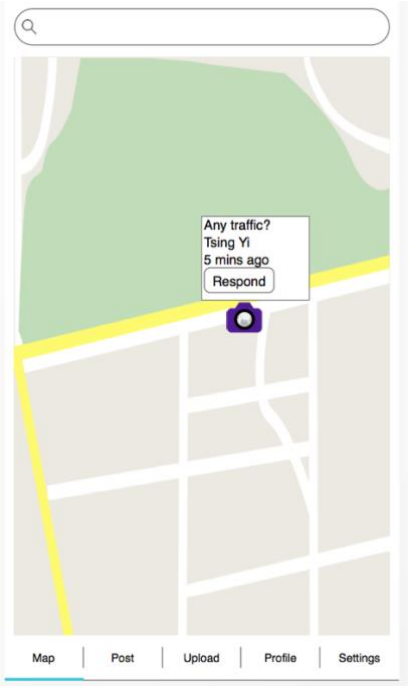


Figure 3b

Map view of the application

Figure 3 – User interface designs prototype of the mobile version of the application

The user interface (UI) design of a system plays a vital role in attracting user to use the application. A friendly interface is a must as it provides enhanced user experience to the users. Therefore, the frontend mobile application prototype has a simple UI design containing only a main panel and a bottom tab-selection bar in each page. This design is consistent throughout every page in the system.

The post view and post view of the system are selected to shown as an example.

For the post view, the posts are sorted one by one in a card form (see Figure 3a). 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 Figure 3b), 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 actual implementation would be shown in section 3 of this report.

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.

2.4 Database Design

In this project, the database is important in the backend server. Personal information for the authentication system, as well as the image and information of each post are stored in the database. It is important to have an organised database for handling substantially large amount of data. Our system uses hybrid database system, i.e. both SQL and NoSQL database system are used.

MySQL database management system is used for storing data. MySQL is an open source relational database management system (RDBMS) based on Structured Query Language (SQL) and it is used by many famous websites such as YouTube and Airbnb [3]. MySQL can achieve high levels of scalability, availability and performance with very low cost, so it is suitable for our system [4]. Moreover, Firebase is also used for storing some real-time data. Firebase is a mobile and web application development platform running by Google. It is specifically designed for real-time and scalable application, since it can accommodate a large application that has over a million users. Firebase is capable of providing real-time data update between devices, even with a huge amount of data [5]. It uses NoSQL key-and-value structure, so it is much faster than relational database in loading data. In our system, the posts data which uses the large amount of storage space is stored in the MySQL database; while authentication and notification data is stored in Firebase.

2.5 User Access and Security Design

As mentioned in section 1.2.4, an authentication system is needed in our project for ensuring no user uses our system to do illegal events or posting inappropriate images. Firebase provides an easy set-up for the authentication system. Users can register accounts by providing their email and password. Firebase can verify the email address by sending a verification email to the user. After the link in the verification email is clicked, the account is verified, and the upload function can be used by the user. This authentication process is essential to verify the identity of the users.

Security is important in the system as well. As the system would get sensitive location data from the user, security measure is done accordingly to protect personal data. The location data of each user is not saved in the database to minimise the opportunity of data breach. The system requires user permission to get the user location. Users can deny the permission at any time. Safety measure is done to prevent disclosure of personal data and the privacy of the user is preserved. In fact, Firebase in more than just database, authentication and storage systems and they are integrated to work with each other. For example, the database and storage can be configured to be accessed (read and write) by authorized users only. Authorization used to be a complicated issue for secure system because the server needs to store sensitive data including passwords or token. Transmitting these data between client and server also needs to be considered. As shown in Figure 2, Firebase generates token for each authenticated user. The client application sends request with the token for authentication in the server for some functions, such as send new requests or upload new responses. With the authentication token, the privacy of the user such as user location is preserved.

3 Results

This section is about the results of the project. The work accomplished to date and the problems encountered are discussed.

3.1 Work Accomplished to Date

The project development timeline is divided into three phases, including design phase, implementation phase and testing phase. The development process is done in April 2018.

3.1.1 Design Phase

The first phase of the project is design phase. It is about the design of the architecture of the system. This project was initiated in late August 2017. The original idea was to develop a road condition sharing system. Nevertheless, the idea was too narrow, and it was estimable that the target user group is small. After further discussion, the idea of information sharing is maintained, while the project is broadened to develop an instant event sharing system.

The detailed project plan is given to Dr. Chim, our supervisor, in late September 2017. After consultation, the scope of this project stated in section 1.2 is established. The approved design is concluded, and implementation is based on the discussion results. Furthermore, the set-up and designs discussed in section 2 is determined in this stage. This stage is completed in November.

3.1.2 Implementation Phase (Server)

The second phase of the project is implementation phase. This phase started immediately after the design phase.

The server set-up is implemented first. This is because it is the centre of the system and it provides all the functions for the frontend applications. The Amazon Elastic Compute Cloud (Amazon EC2) instance has been set up for hosting the API server and the web application. It is available at https://happens.hk.

The API server is written in Python for handling the routing to different services. There are different HTTP GET and POST method for handling different requests. For example, the URL https://api.happens.hk/post get the latest post by default. It has different parameters such as limit to limit the number of post returned. The URL https://api.happens.hk/post?limit=50 means it returns the latest 50 posts; while https://api.happens.hk/post/4115 get the post with id 4115. In conclusion, the API server has mainly four HTTP GET method route for getting post information, category information, request information and user information. On the other hand, there are six HTTP POST method route for uploading information, which are uploading posts, subscribing new category, unsubscribing category, making new request, responding to request and accepting the respond. All POST requests require the token generated by Firebase for authentication. Only verified users can upload information to the system.

```
SELECT post.id, post.description, post.username, post.location, post.latitude, post.longitude, post.date_created,
       IFNULL(upvote.count, 0) AS upvote, IFNULL(downvote.count, 0) as downvote, IFNULL(comment.count, 0) as comment_count,
       post.user_id, images.urls, category_ids.ids, location_ids.ids, voting.voted
FROM post
LEFT JOIN
    (SELECT post_id, count(post_voting.voted) AS count FROM post_voting WHERE voted = "up" GROUP BY post_id) AS upvote ON upvote.post_id = post.id
LEFT JOIN
    (SELECT post_id, count(post_voting.voted) AS count FROM post_voting WHERE voted = "down" GROUP BY post_id) AS downvote ON downvote.post_id = post.id
LEFT JOIN
    (SELECT post_id, count(post_comment.id) AS count FROM post_comment GROUP BY post_id) AS comment ON comment.post_id = post.id
LEFT JOIN
    (SELECT post_id, GROUP_CONCAT(url SEPARATOR ',') as urls FROM post_media_url GROUP BY post_id ORDER BY `post_media_url`.`post_id` ASC) images ON post.id = images.post_id
LEFT JOIN
    (SELECT post_id, GROUP_CONCAT(category_id SEPARATOR ',') as ids FROM post_category_association GROUP BY post_id ORDER BY `post_category_association`.`post_id` ASC) category_idsON post.id = category_ids.post_id
LEFT JOIN
    (SELECT post_id, GROUP_CONCAT(location_id SEPARATOR ',') as ids FROM post_location_association GROUP BY post_id ORDER BY `post_location_association`.`post_id` ASC) location_ids ON post.id = location_ids.post_id
LEFT JOIN
    (SELECT post_voting.post_id, post_voting.voted AS voted FROM post_voting WHERE post_voting.user_id = '') AS voting ON voting.post_id = post.id
WHERE  post.is_vision_removed = 0
ORDER BY post.id DESC
LIMIT 50
```

Figure 4 – SQL to get all posts

The other part of the server implementation is to write corresponding SQL to get the correct data from the database. Figure 4 shows an SQL which get all posts with related comments and category, etc. It is a straight-forward SQL which joins many tables together in order to get all related information. The performance of the server getting data from the database is discussed in section 3.1.4.

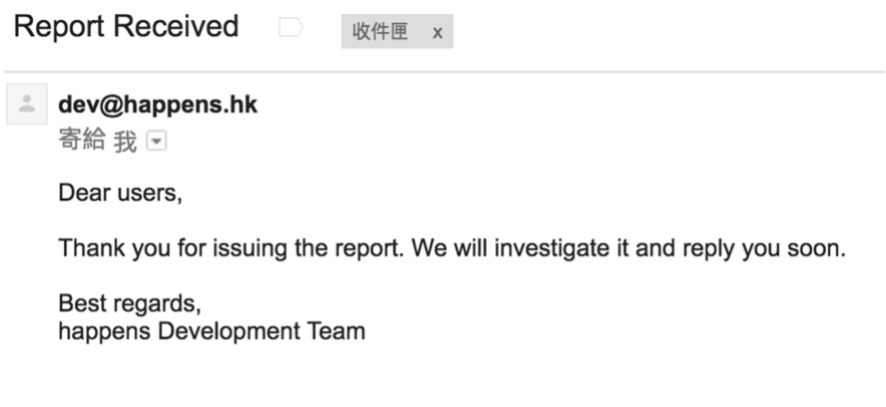


Figure 5 - Auto-response email from the server

Finally, our server connects with other API to perform different tasks. Google Gmail API is used to send auto-respond email back to the user when the user reports problems of the system. Also, Google Cloud Vision API is used for filtering indecent images uploaded. Google Cloud Vision API analyse the image for inappropriate content such as racy image. The system uses this features and filter the images uploaded. If the post is flagged with inappropriate tag, it will be hidden.

3.1.3 Implementation Phase (Client)

The implementation for the client application in both mobile and web versions is done in parallel. The client application provides a user interface for user to use the system. The UI is implemented according to the design prototype in section 2.3. Two views of the application are chosen to be detailly explained, while other views are shown in Appendix.

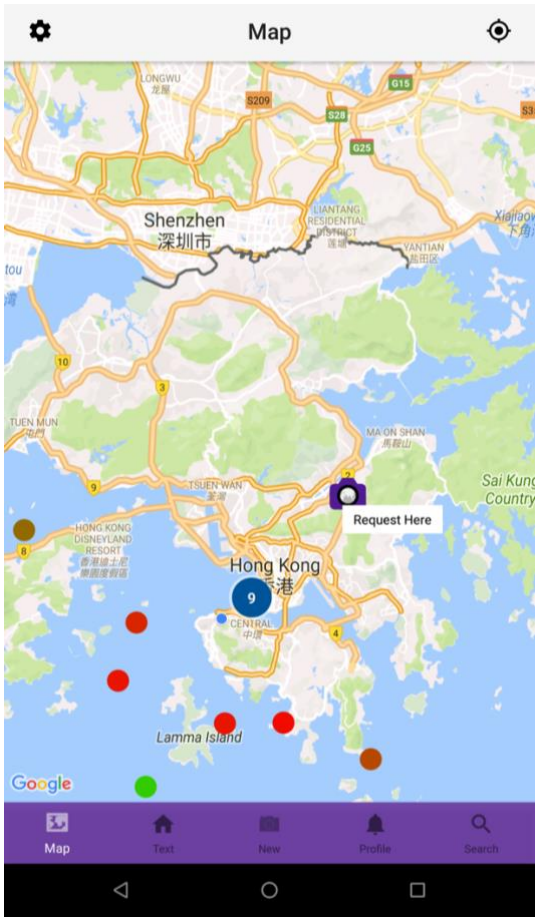


Figure 6a – Map view of the system

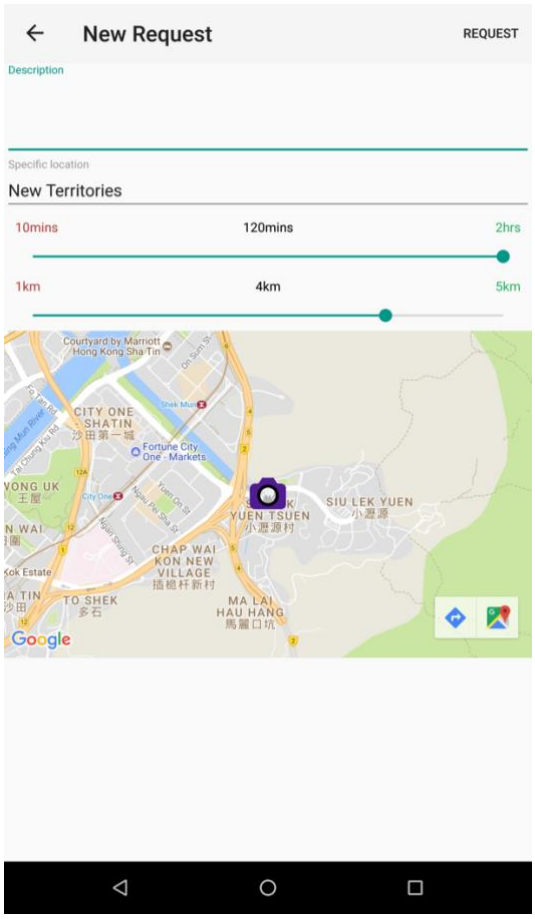


Figure 6b – Request view of the system

Figure 6 – Implemented user interface of the mobile version

Figure 6 is two screens shots from the Android mobile application of the system. The map view and the request view are two of the main page of the application. For the map view, it uses the Google Map API for Android which is an official API provided by Google which supports clustering markers. Our system utilizes this feature of the API and mark our posts on the map, while it could automatically cluster then when zoomed out. Also, by long-pressing on the map view, it selects the point and ask whether the user would like to request information from that point. After filling in corresponding data in the request page, the request is sent to server and notify users around that area.

Mobile application focuses the data transfer from the application to the server. Our system needs to upload images to the server; and it would take a lot of mobile data usage during upload. Therefore, our system compresses the image to around 100kb. This reduces the data usage significantly.

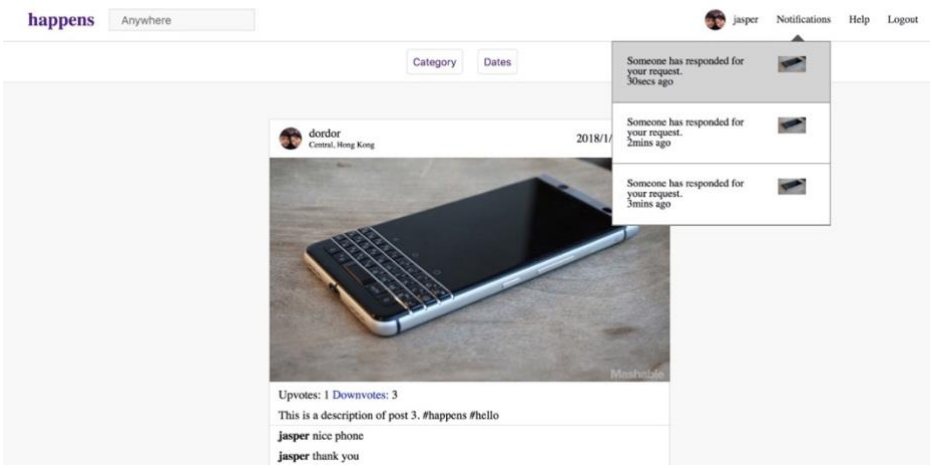


Figure 7 – Implemented user interface of the web version

Figure 7 shows the text view page of the web application. It has a similar user interface design with the mobile version. This provides an identical user experience to user in both mobile and web version. However, in the web application, it has fewer functions as a computer gets a less accurate location and camera is not always supported. Therefore, the respond to request function and the upload new post with photos function is disabled from the web version.

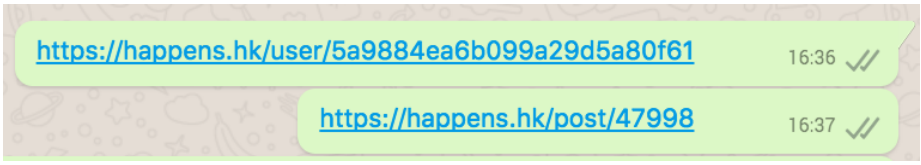


Figure 8 – Link sharing of happens in WhatsApp

Finally, there is an extra feature during implementation of the client application. As each post or user profile has a specific link in the server, our system supports sharing of the posts or user profile. Moreover, deep linking in enabled in the mobile application, i.e. by clicking the link the application will be opened automatically and routed to the specific page.

3.1.4 Testing Phase

Testing phase is done after the implementation phase in April. Different automated unit tests and integration tests are done to ensure the system works.

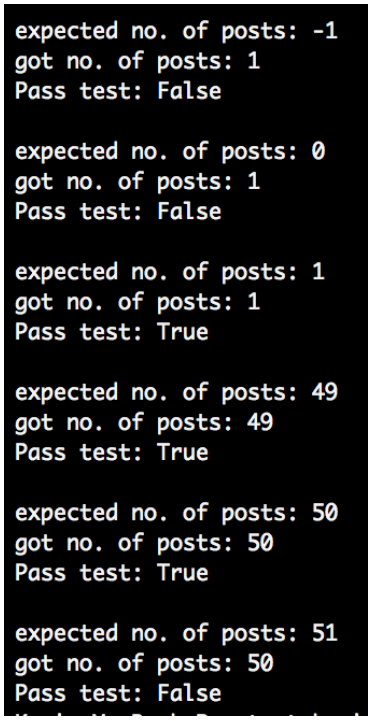


Figure 9 – Boundary Value Analysis Test Case for getting posts

First, boundary value analysis is used for the unit test. The defect of the system appears easily on the boundary. For example, in Figure 9, the API server is tested for getting specific number of posts. The limit is bounded from 1 to 50 so we tested the value right above and below the boundary. It returns posts limit from 1 to 50 but not returning error code for error inputs so the implementation is modified afterwards.

Second, integration test is done after every unit tested. We deployed the application to some real Android devices and tested the request function. It notifies the user in the range correctly. The application runs smoothly on different Android device so the system passes the testing phase.

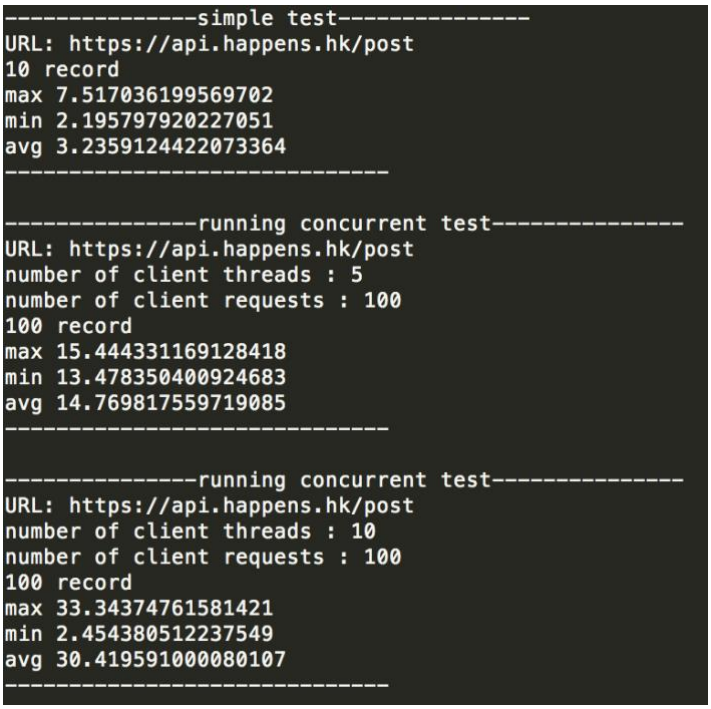


Figure 10 – Performance test result for getting posts

Finally, a performance test is done for testing the server for concurrency. Figure 10 shows the test results of the test in time measurement. The server responds in around 3 seconds for normal environment. However, when many clients send GET request to the server, the server responds slowly. 10 client threads request 100 time would trigger the server to respond in an average of 30 seconds. There is a time that the server cannot respond on time and triggered a gateway timeout. The server is unacceptable so recommendation is given in section 4.1.4.

3.2 Problems Encountered

There are different problems discovered during the development. The first problem is about the choice of programming language for the front-end application, while the second problem is suggesting category.

3.2.1 Choice of programming language for the front-end application

In the initial plan of the project, React Native was planned to be used for the implementation of the front-end application. React Native is a framework using JavaScript language developed by Facebook, Inc. announced in 2015. The major feature of React Native is that it can generate a cross-platform application, i.e. running on both iOS and Android platform, in just one single implementation [6]. For instance, Facebook and Airbnb are using React Native.

However, during research and implementation, the problem of lacking official support from Google or Apple is found. For example, there is no official support from Google Maps API in React Native. Google Maps in React Native is only supported by third party and some main features such as location markers clustering cannot be implemented. The map is a major component in the system and it cannot be developed without the map. After discussing with the supervisor, native mobile version of the system using Kotlin for Android is built. The platform setup is documented in section 2.1 and this problem is solved in the design phase.

3.2.2 Category Suggestion

Category system as mentioned in section **Error! Reference source not found.** is a major function that help the users to categorize their information and for the other users to search for it. In the early stage of the project, we planned to make each category in the form of type of the information plus the location information. The location information may further consist of three levels (country, state, city) to indicate the detailed location of the information. For example, a piece of information may be categorized to "Traffic in Mongkok, Kowloon, Hong Kong".

Nevertheless, during the implementation, it is hard to design and develop a system for the user to input this information. The users are required to input four fields and it is not user-friendly. The design is updated such that the category are in the form of one type plus one location.

However, Google Cloud Vision API is designed to be used to distinguish what the main object in a photo is and automatically suggest suitable category. This is not working because we have to build our own category list beforehand and apply deep learning to it. The learning machine is out of our scope and is difficult to build. Our solution to the problem is that the system suggest popular categories to the user. More development suggestion is discussed in section 4.

4 Recommendations

Due to the limitation of this final year project, this report marks the final milestone for the project. However, there are recommendations for enhancing the system, as well as providing suggestions to future development on instant event sharing system similar to this project.

4.1 Recommendations for enhancing the system

There are still many problems and space for improvements in the system. Therefore, there are recommendations for improving the system.

4.1.1 Implementation in iOS

Android and iOS platforms almost cover all smartphone users in the world, so the system should be migrated to and implemented in iOS version to let more users use the system. When there is more user in the system, there would be more information exchange among users.

4.1.2 Localisation

Also, localisation should be applied to the system. Currently the system is in English version and Chinese version is supported partially. Although English language are globally recognised, if we would like to attract users from different countries, we have to localise the application and translates the predefined list of categories and make a localised setting page.

4.1.3 Different Network Environment

The system is focuses on images exchange to provide actual data. If the mobile version is handled contrarily in different network environment, for example download a bigger image for Wi-Fi user and compress a smaller image for mobile data user, this would reduce the mobile data usage and may be an incentive to attract user to use the application.

4.1.4 Improve AWS settings

As mentioned in section 3.1.4, the server responds slowly when there are concurrent client requests. This would happen in real world usually, therefore we have to tackle the problem. Due to limitation of this project, the whole system is only built on the basic Amazon EC2 cloud computing instance. However, Amazon provides Amazon Relational Database Service (Amazon RDS) and AWS Elastic Beanstalk, where they are specific server for database and web engine respectively. If we can handle the traffic well among these services, the server response time can be increased. Moreover, the structure of the database may not be well-managed so the query time for getting posts is long. The structure of the database can be modified and take a balance between duplicating data and getting quicker response.

4.2 Suggestions to future development

Mobile crowdsourcing is a popular issue nowadays. This system acts as a prototype to future development on instant event sharing system. The technical suggestion to future development on similar project is shown above. From the idea point of view, instant event sharing system focuses on information exchange among users. Therefore, the system should encourage user to post accurate information. For example, the system could be more incentive-based. User should have a limit on posting and requesting each day. User can increase the limit by solving others request. This would increase the quality of the response and also encourage user to participate more in posting but not only request for information. The idea of mobile crowdsourcing is to provide a platform for user to exchange information, so future development should be concentrated on this way.

5 Conclusion

This report has shown the overall progress of the final year project. This project was initiated by the insufficiency of existing functions in the current social media applications. A new instant event sharing system is decided to be built and it is a platform that can encourage people to exchange real-time information on different places. It takes references from the strength of existing applications, as well as providing new features to cover the deficiency.

This paper described the design, implementation and testing phase of "happens", the instant event sharing system. It has five important features, including the request-and-response function, photo sharing function, category system, the authentication system and searching function. The request-and-respond function let user inquire information based on location and the system connects users in the world by location.

Moreover, this project is to develop the system in both mobile and web application. Therefore, this system can be reached by a significant number of users. The mobile application is native, i.e. written in officially supported language of Kotlin for Android. On the other hand, the web version is written in React.js, but it has no photo sharing function, as we limit the system only accept image instantly captured from the camera. In addition, the back-end server is organised using Google Firebase and Amazon Elastic Compute Cloud. Firebase is a software development platform, which provides all-in-one features to the development, such as authentication system and database. Amazon EC2 is a web service provides scalable compute capacity in the cloud. Our server is located in Amazon EC2, which includes the web application, API server and the MySQL database. Firebase NoSQL database is used for real-time data such as notification, while long term data such as posts are stored in MySQL database.

The development process of the project is done in April 2018. Previously, the design phase, implementation phase and the testing phase is finished one-by-one and the deliverables are deployed to the public.

Despite there are different problems appearing during the development, solutions are found to tackle them. The rationale to write native mobile applications, rather than using React Native to construct a cross-platform application, is that React Native is not officially supported by Google and Apple. This increases the difficulty of writing complicated functions, such as implementing the map and geo-fence. The other problem is that the deep learning category suggestion system is out of our scope so we choose to suggest only popular categories to the users when they choose category in the post.

During development, feedbacks are collected. Nevertheless, due to the limitation of this project, improvements can be made if there are future development. Recommendation for enhancing the system, such as localisation and improve AWS setting, should be done if the system keeps running in the future. This would attract more users. Besides, suggestions are given for future development on similar projects. Future development should be more incentive-based and this fits the current trend of mobile crowdsourcing.

References

[1] Amazon, “Amazon EC2,” Amazon, 2018. [Online]. Available: https://aws.amazon.com/ec2/. [Accessed 14 04 2018].

[2] J. Thornsby, “Kotlin vs Java: key differences between Android’s officially-supported languages,” Android Authority, 28 02 2018. [Online]. Available: https://www.androidauthority.com/kotlin-vs-java-783187/. [Accessed 14 04 2018].

[3] MySQL, “MySQL Customers,” MySQL, 2018. [Online]. Available: https://www.mysql.com/customers/. [Accessed 14 04 2018].

[4] MySQL, “A MySQL Strategy Whitepaper,” MySQL, 2011.

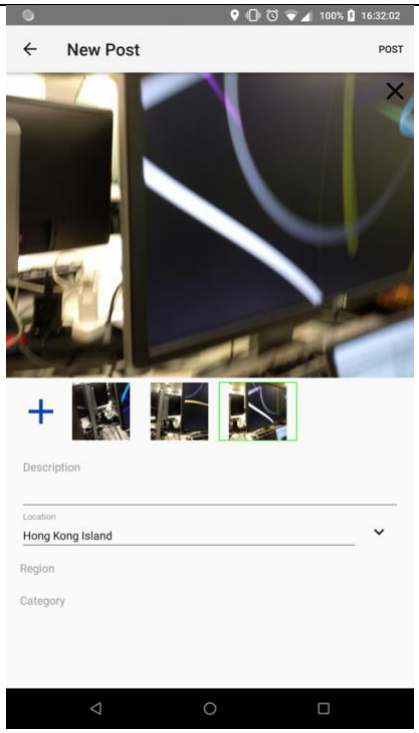
[5] Google Developers, “Firebase,” Google, 3 10 2017. [Online]. Available: https://firebase.google.com/docs/database/usage/limits. [Accessed 23 10 2017].

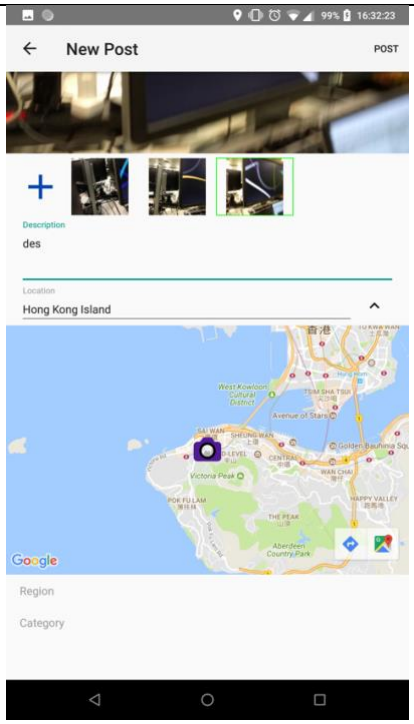
[6] A. Narayan, “React Native vs Native iOS/Android,” Course Report, 15 11 2017. [Online]. Available: https://www.coursereport.com/blog/so-you-want-to-build-a-mobile-app-react-native-vs-native-mobile. [Accessed 30 11 2017].

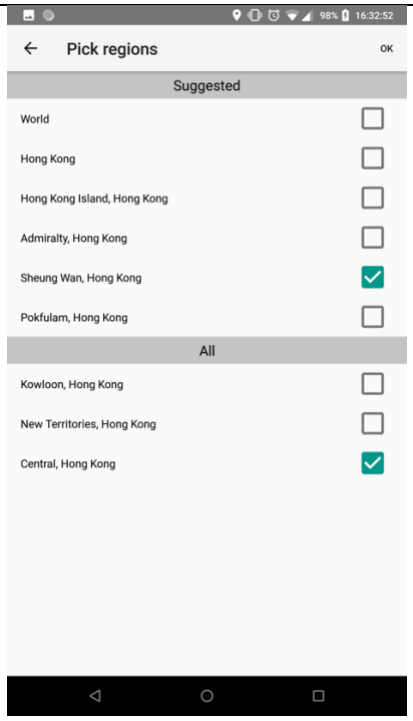
Appendix

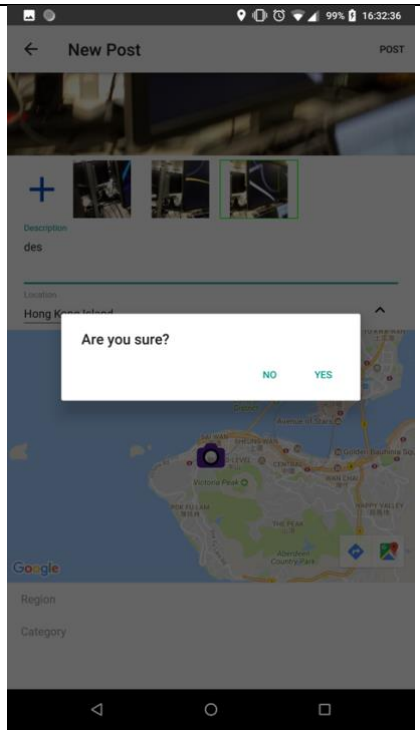
Implemented User Interface of the mobile application

New Post Flow

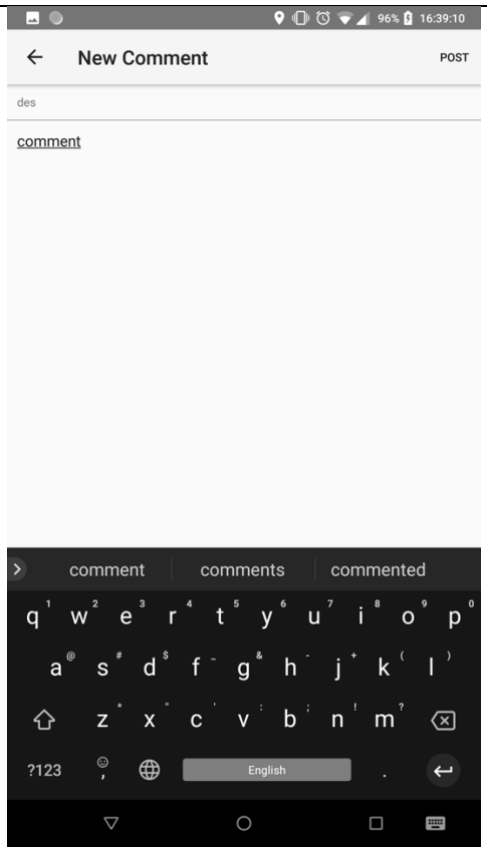


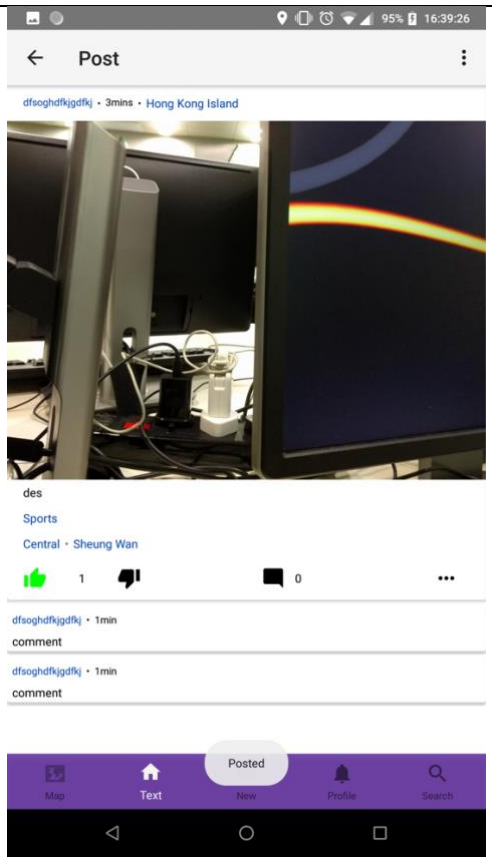




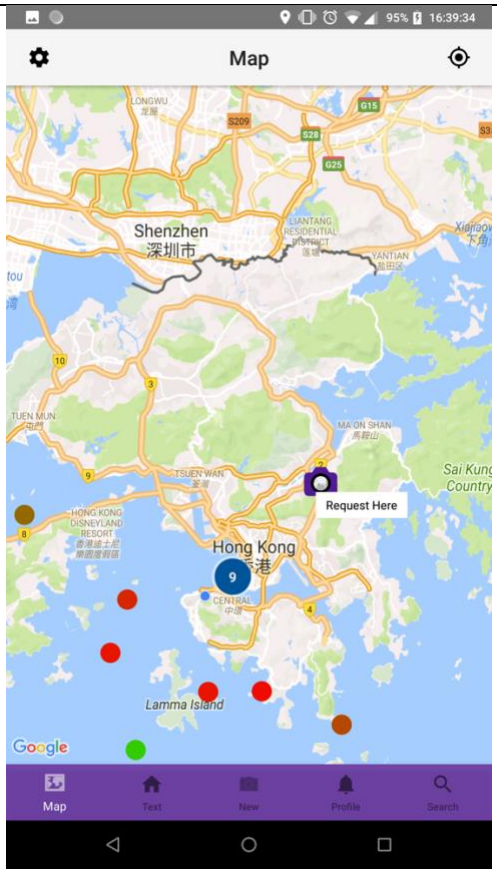


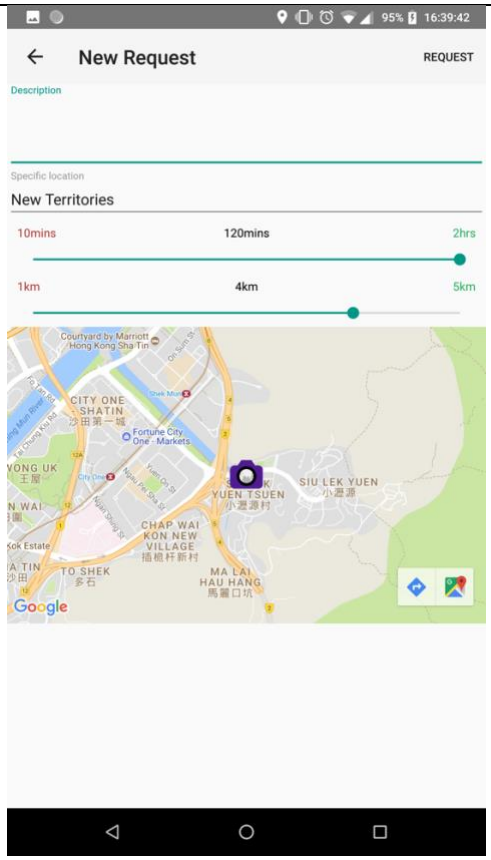
Text View Vote and Comment Flow

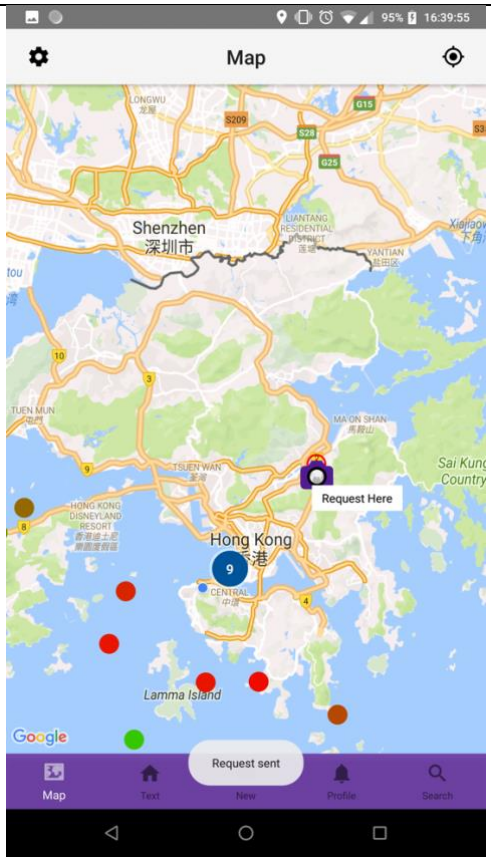




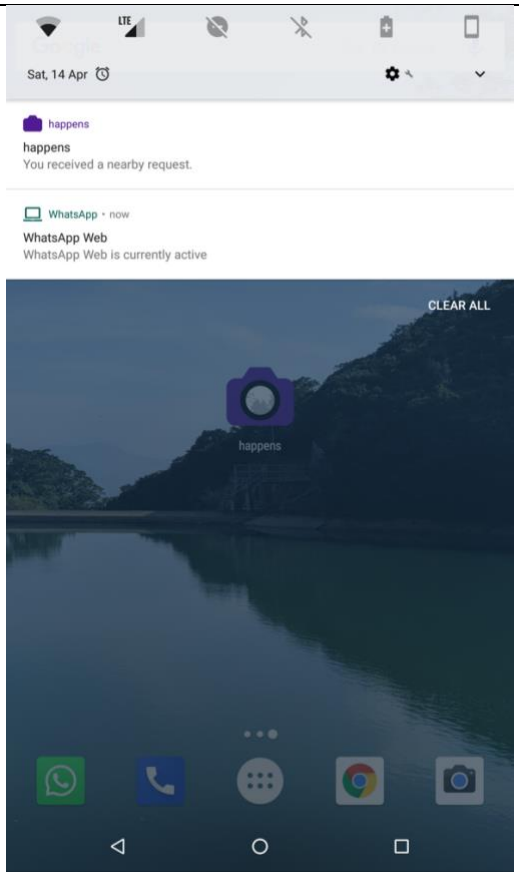
New Request Flow

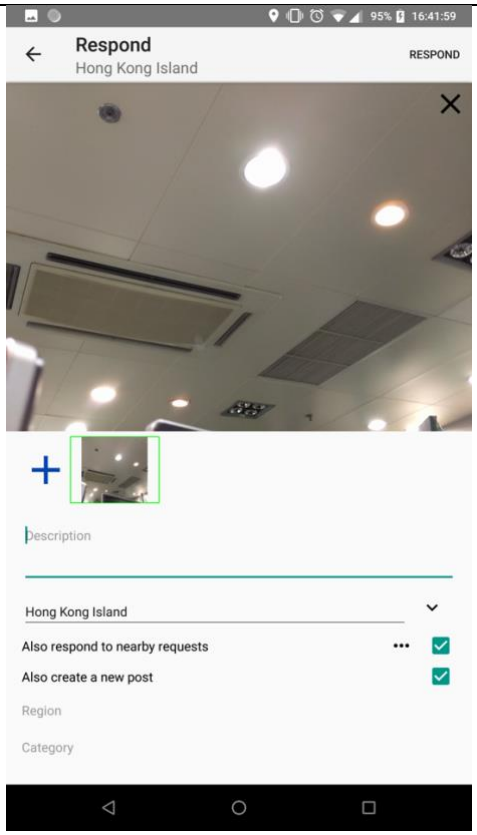




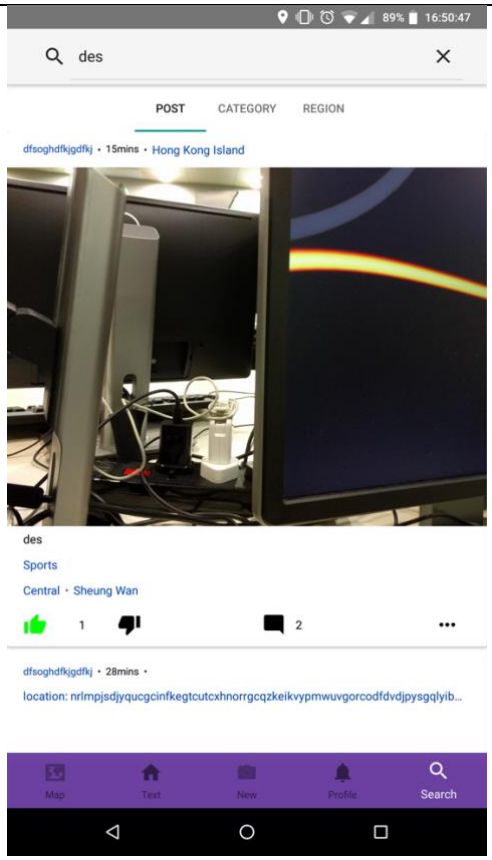


New Respond Flow

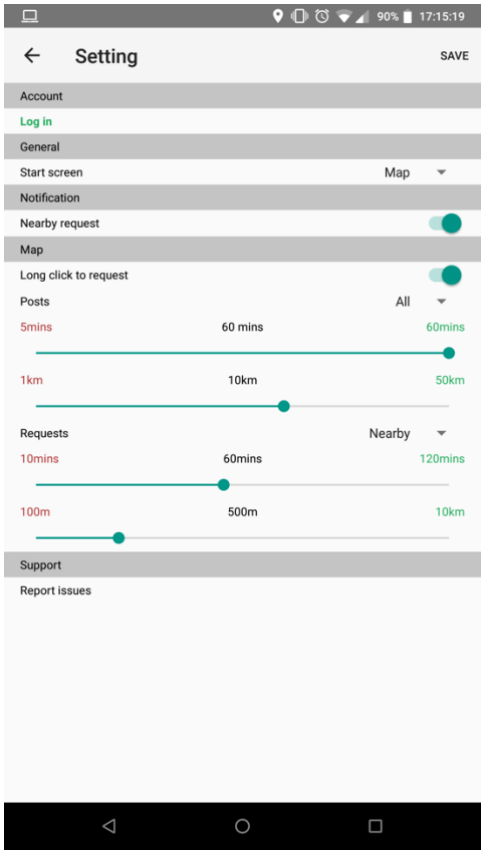




Searching Flow



Settings Page



Report Page