A Client-Side GIS Web Application

THEERAPAT KHAMMUEANG

TASAPROM SUPTONGCHAI

BACHELOR OF ENGINEERING PROGRAM IN SOFTWARE ENGINEERING

INTERNATIONAL COLLEGE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2013

Thesis - Academic Year 2013

B.Eng. in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Title: A Client-Side GIS Web Application

Authors:

1. Miss Tasaprom Suptongchai Student ID : 53090012
2. Mr. Theerapat Khammueang Student ID : 53090015

Approved for submission

.....

(Asst.Prof.Dr. Visit Hirankitti)

Advisor

INTERNATIONAL COLLEGE

Date August 28, 2014

A Client-Side GIS Web Application

Miss. Tasaprom Suptongchai 53090012
Mr. Theerapat Khammueang 53090015
Asst.Prof.Dr. Visit Hirankitti Advisor
Academic Year 2013

ABSTRACT

A geographic information system (GIS) is a powerful tool which provides an effective means to support planning and decision making related to spatial information. GIS services can be best delivered to a large community via a web application. In this project, we developed a client-side GIS web application for emergency response and disaster alleviation to help people in a large scale.

Our client-side web application contains several components which are a map viewer, a set of GIS tools, a search facility, a data collection function from the user, a monitoring function for tracking progress of a submitted request. The GIS tools provide means for interaction with the map such as zooming, panning, measuring distance and area, including tracking user's current location and orientation. These components are developed using HTML5, JavaScript, jQuery, and OpenLayers where the latter is a JavaScript library for map visualization and interaction.

We have applied several development techniques for creating our application, including a technique, called 'Level of Details', for displaying map of different details. It allows the user to see more details of a map when the map scale is relevant to show them. Another technique is an event management when an event generated by OpenLayers conflicts with that generated by jQuery. In this case, we will give higher priority to the jQuery's one.

Acknowledgements

First and foremost, we wish to sincerely thank our project supervisor, Asst.Prof.Dr. Visit Hirankitti, whose contribution in stimulating suggestions and encouragement. Especially, his recommendation on this project topic, "A Client-side GIS web application", which gives us the opportunity to learn new things that we could not find in the classes, a challenging situation helping us to apply our knowledge in software development to use in the real working environment. We really appreciate his support towards the completion of this project. And, without his help and guidance, this project would not have been completed, especially in writing this project thesis.

Furthermore we would also like to show gratitude to our friends and all who contributed in one way or the other in the course of the project. Especially, the group whose work is to develop the server-side service. We also really appreciate a chance to have a collaboration with other groups as well.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Nowadays, the rapid growth of population has caused many changes around the world. The world is facing challenges such as how to improve management of resources on a map to help relieving people from an unexpected accident or even natural disaster occurred across a particular area. We need to use a Geographic Information System (GIS) which provides us an effective means to support planning and decision making concerned with spatial information.

Furthermore, today the Internet becomes an essential part of our everyday life as it provides ability to access to data and computer services from anywhere. A web application is a good example how people can access information via the Internet from anywhere and anytime. The web application can also be deployed on a variety of devices such as on web browser of any personal computers, mobile phones, and tablets. This emphasizes the ability of ubiquitous computing due to the fact that the accident on the map could happen anytime and anywhere. As a consequence, the users can access the web application immediately using their devices of choice.

However, developing a web application with spatial data is not enough. We intend to provide the user with a beautiful user interface and which is, absolutely, easy to use. Consequently, several programming languages involved in the development including HTML5, CSS3, JavaScript. Those are used to define a universal standard for web application and to build an exceptional interface for the user to use either using the traditional mouse or touch gesture from a mobile device. Additionally, OpenLayers, a JavaScript library, should be mentioned as well since it plays an important role in web development, especially, when we have to deal with the map.

As a result, we are confident that combining the technology of GIS with web application development will offer a good experience and provide a reliable service supporting multiple platforms for those people who are suffering from unexpected accidents or natural disasters.

1.2 Objectives

We are to develop a client-side GIS web application which provides the map with accuracy data. The application also provides tools for users to search for location, for example, a closest police station or a hospital in order to get there as soon as possible. Another interesting function is that the user is able to complete the form or send note, or even call via internet to request for emergency service. Our goal is to provide a better emergency service to help saving people's lives by applying GIS technology with the map. The web application to be developed has three main objectives as listed below:

1. The web application provides interaction with map and map layers such as zooming, panning, layer filtering, etc.
2. The web application provides many GIS tools for users to interact with the map such as to find area and distance, get the user's current location, etc.
3. The web application provides a GUI form for use in different contexts; for example, a function to send request, a pop-up balloon to show more information when a specific point is clicked, register form, responding form for staff.
4. The web application also provides client application such as search function, a layer filter function, request tracking, user notification, geolocation, map orientation

1.3 Contributions

We believe that our client-side GIS web application will play an important role in the adoption of many organizations that make use of geographic information system technology, especially for humanitarian organizations such as the Thai Red Cross. The main purposes is to help emergency response service and natural disaster which, as a result, may save people's lives.

1.4 Scope of work

The client-side web application developed in this project has three main objectives as follows:

1) The web application provides the accuracy data with easy-to-use user interface.

Works required to accomplish this task:

- The system is able to display the maps from OpenStreeetMap and Provincial Electricity Authority (PEA) map via the browser using OpenLayers 3 (beta).
- We have to develop user interface using HTML5 for the page markup and arrangement, and CSS3 for the page style.
- We have to develop GUI using JavaScript and jQuery for the data collection and processing, event handling, user interaction, and animation.

2) The web application provides many GIS tools for the user to interact with the map such as zooming, panning, measuring area and distance tools, finding current location, map rotation and orientation tools, and printing tool.

   Works required to accomplish this task:

   - We have to develop the GIS tools using classes, methods from OpenLayers 3 (beta) library.
   - The system should be able to handle events, for example, mouse clicking, mouse dragging, keyboard pressing, from OpenLayers 3 (beta) by the web application using JavaScript and jQuery.
   - On different projections on different maps, we need to convert one projection from one to another to be able to refer to the correct positions. For example, using Google Maps, a latitude and longitude position is used for a position while for our map we use a UTM based position.

3) The web application provides a GUI form to use in different contexts, for example, a register form, a request form, a response form for staff, search and request result management, and a pop-up balloon to show notice information.

   Works required to accomplish this task:

   - We have to create a neat form, we use CSS3 and JavaScript to style and provide a smooth transition when the user interacts with a web page.

4) The web application also provides client application such as search function, a layer filter function, request tracking, user notification, geolocation, map orientation.

Works required to accomplish this task:

- In order to do the above functions, we have to integrate the function from OpenLayers 3 (beta) to the page using JavaScript and jQuery. In addition, event handling is the most important part since if the events conflict with each other, the error will occur.
- For the search function, and other related functions that are required to communicate with the server side, the data to be sent must be structured in JSON format before it is sent using AJAX mechanism.

1.5 Structure of this thesis

The structure of this thesis is organized as follows:

Chapter 1 Introduction: This chapter discusses motivation, objectives, contribution, scope of work, and structure of this thesis.

Chapter 2 Related Work: This chapter discusses reviews of related works which are web applications developed by Metropolitan Electricity Authority, Telephone Organization of Thailand, and Google.

Chapter 3 Background Knowledge: This chapter discusses coordinate system, geographical data, projection, and visualization of GIS data model, client-side development languages, OpenLayers, gyroscope and orientation.

Chapter 4 Requirements and Analysis: This chapter discusses problem description, requirements specification including functional requirements and non-functional requirements, a use case diagram and its description.

Chapter 5 System Design: This chapter discuss system architecture, event-driven programming, graphical user interface design, and map display technique.

Chapter 6 Development: This chapter discuss the development of map viewer, geolocation and orientation, AJAX methods, and OpenLayers comparison.

Chapter 7 Result: This chapter discuss the result of our project including, system user interface, how each GIS tool works, some request sending and responding scenario, and some of our experimentations on GIS.

Chapter 8 Evaluation and Discussion: This chapter evaluates and discusses our project work as well as explains the user interface of the system in detail.

Chapter 9 Conclusions: This chapter includes project summary, lessons learned, problems and obstacles, and a plan for future work.

Chapter 2

Related Work

In this chapter, we look at some related works concerning with a GIS web application. Firstly, there is a similar GIS web application which was developed for an electricity company in cooperation with a team in our thesis supervisor's lab. Regarding to this GIS web application, the map layers are provided by Metropolitan Electricity Authority (MEA). Secondly, the iCoverage system developed for Telephone Organization of Thailand (TOT) by another team in the lab. Finally, we describe a web mapping application so-called a Google Maps developed by Google.

As a powerful tool, GIS is widely used by those web applications and the followings discuss all of them in detail and distinguish the point that is different from our web application.

2.1 Review of related work

2.1.1 MEA Better Care

MEA Better Care (as shown in Figure 2.1, source: http://www.giswebservice.com) is a location-based web application provided by Metropolitan Electricity Authority (MEA) which delivers a web map service with accuracy data of related electrical devices, location of MEA offices as well as fundamental landbase layers. It focuses on maintenance of electrical devices, resolving defective electrical equipment and electrical system design.

Figure 2.1 MEA Better Care

This web application is developed using OpenLayers 2. It has some similarities to our project; for example, search function and basic GIS tools like distance/area measurement and geolocation. However, there are also some differences which are first, in our project, we use OpenLayers 3 (beta) version to implement when we have to deal with the map in our web application. Second, we focus on developing emergency response for Thai Red Cross organization so we have functionalities to deal with user request and response which does not appear in MEA Better Care web application.

2.1.2 iCoverage by TOT

This web application is TOT iCoverage system (as shown in Figure 2.2, source: http://www.icoverage.tot3g.net) which provides map with data of TOT speed internet including 3G, 4G, hotspot and Wi-Fi. This application focuses on checking the speed of the internet provided by TOT at any location in Thailand.

It is developed using OpenLayers 2. The similarities between this web application and ours is that both applications create map which contains GIS data and use OpenLayers library to develop the applications. The differences are first, we have migrated to OpenLayers 3 (beta) version. Second, we have different types of functionalities; the TOT iCoverage provides search function for users to search for internet signal pole to check the speed in any area in Thailand but our project provides

search function for users to search for locations involved in emergency respond for example, police stations, hospital, and landmark buildings.

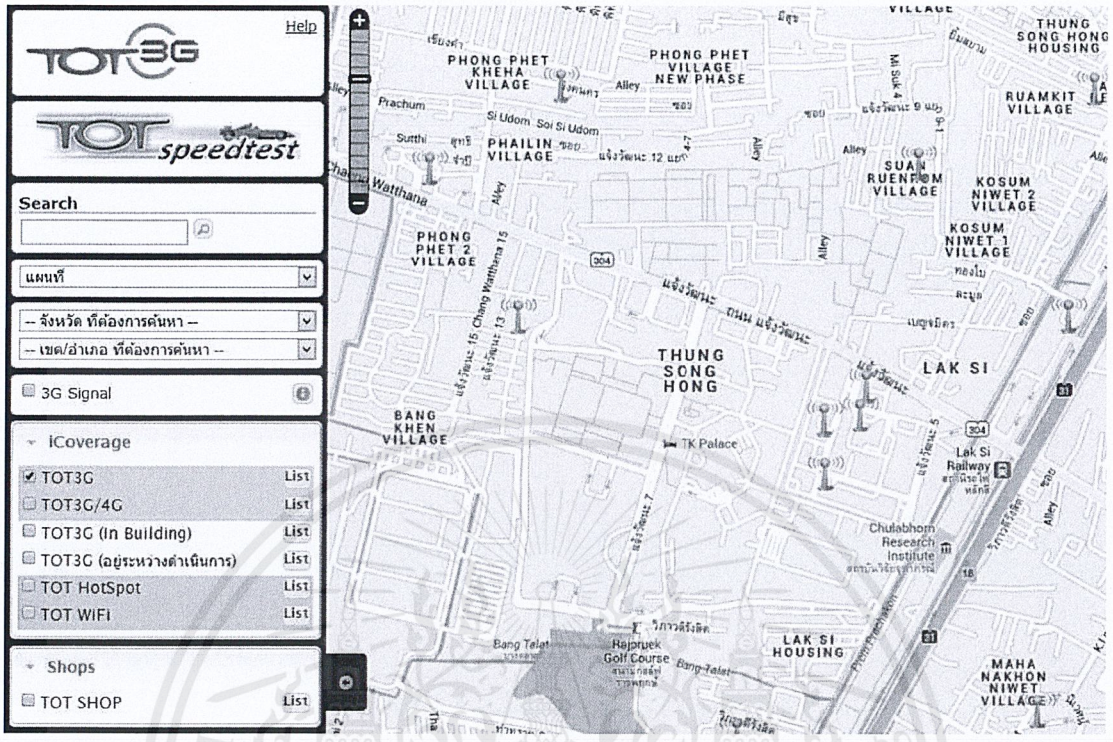

Figure 2.2 iCovergae by TOT

2.1.3 Google Maps

This web application (shown in Figure 2.3, source: https://maps.google.com) is a web mapping service application provided by Google. It offers its notable function called a 'route planner' for travelling on foot, by car or public transportation with approximate time and distance. This google maps can be embedded on third-party websites via Google Maps API.

The API is used in our project as well to provide an optional based-layer. The similarity between our project and Google Maps is that these two web applications both provide map viewer with ability to search for locations. The differences are first, we develop our project using OpenLayers which can call the map source via Google Maps API and other sources such as WMS and OSM. Second, Google Maps does not really provide GIS tools because there is no access to facilities from spatial database.

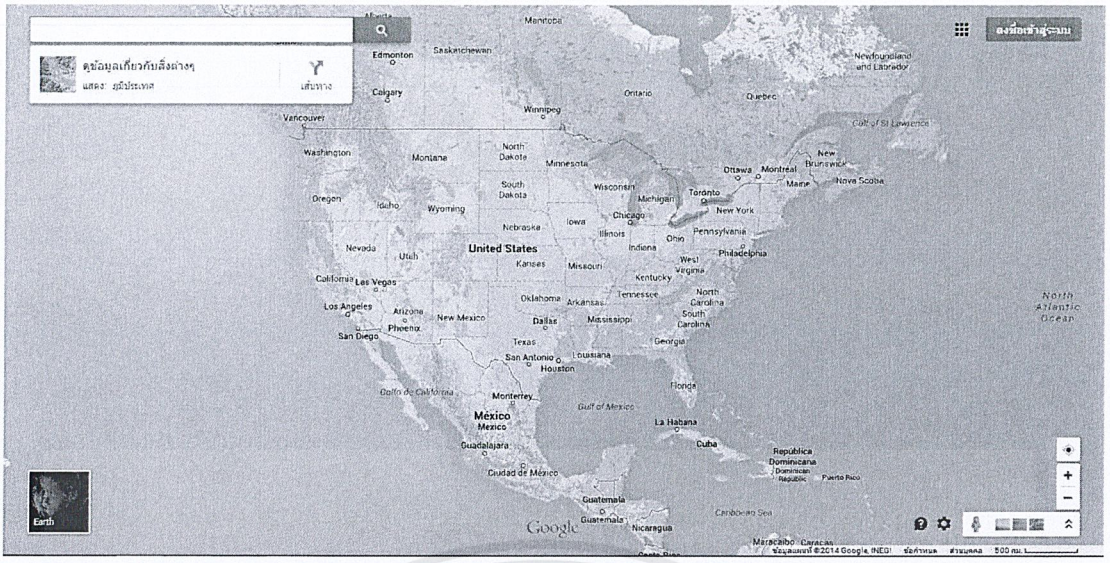

Figure 2.3 Google Maps

Chapter 3

Background Knowledge

3.1 Geographical Coordinate System

Latitude and longitude system is the system for defining and locating geographic positions on the surface of the earth. They are simply angular distance in degrees.

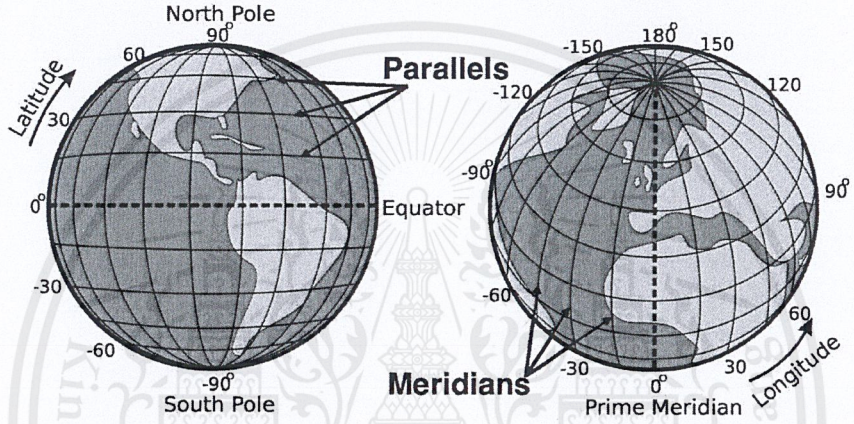

Figure 3.1 Latitude, longitude, and related components

From the Figure 3.1, Latitude lines are lines which runs horizontally. They are often referred as parallels because they are parallel and have an equal distance from each other. Degrees of latitude lines are from 0° to 90° north to south. Zero degree is the equator. 90° north is the North Pole and 90° is the South Pole. Latitude lines are a way to measure how far north or south of the equator. Each degree of latitude is approximately 69 miles or 111 kilometers apart.

Longitude lines are lines which runs vertically. They are often referred as meridians. Longitude lines are positioned with one end at the North Pole and another at the South Pole. Zero degree longitude is called the Prime Meridian, it is located at Greenwich, England. Longitude lines are a way to measure how far east or west of the Prime Meridian, there are 180 lines east and 180 lines west. The 180 degree line is directly opposite of the Prime Meridian, it is called the International Date Line. Each degree of longitude is approximately 69 miles or 111 kilometers apart which is the same as latitude.

3.2 Geographical Data

Geographical data can be referred to data objects identified in a geographical space. Data objects can be both natural features like oceans, terrain, mountains, and constructed features like buildings, streets, landmark point.

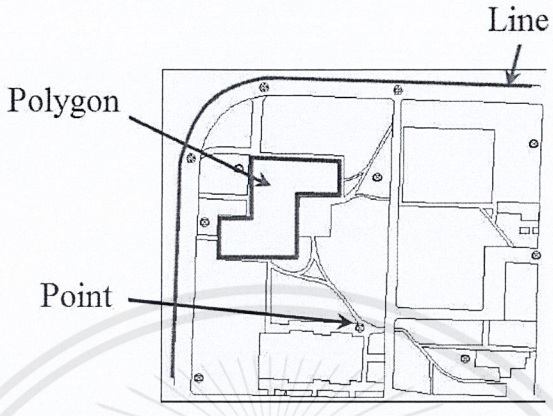

Figure 3.2 Geographical Data

Geographical data (see Figure 3.2) is usually stored in the form of points, lines, polygons and other geographic and geometric data primitives. It is can be accessed, manipulated and analyzed through Geographical Information System (GIS).

3.3 Projection

Projection is a mathematical conversion from spherical to planar (flat surface) coordinates. Map projections ensure a known relationship between locations on a map and their true locations on the earth. There are many types of map projection but any representation of the earth's surface in two dimensions always distorts shape, area, distance, or direction. Therefore, different type of projections produce different distortions.

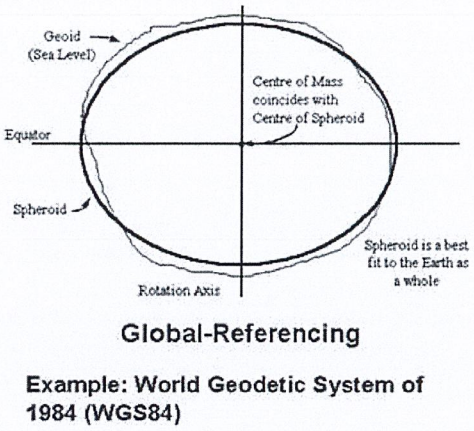

Figure 3.3 World Geodetic System

Projections in GIS are commonly referred to by their EPSG codes, identifiers managed by the European Petroleum Survey Group. One common identifier is EPSG:4326 which is also called the World Geodetic System (WGS). This projection is used by the GPS satellite navigation system and for NATO military geodetic surveying. The coordinate origin of EPSG:4326 is meant to be located at the Earth's center of mass as shown in Figure 3.3. It comprises a standard coordinate system which describes maps where latitude and longitude are treated as X/Y values. In our project, we use this projection as a standard projection to communicate with the server side.

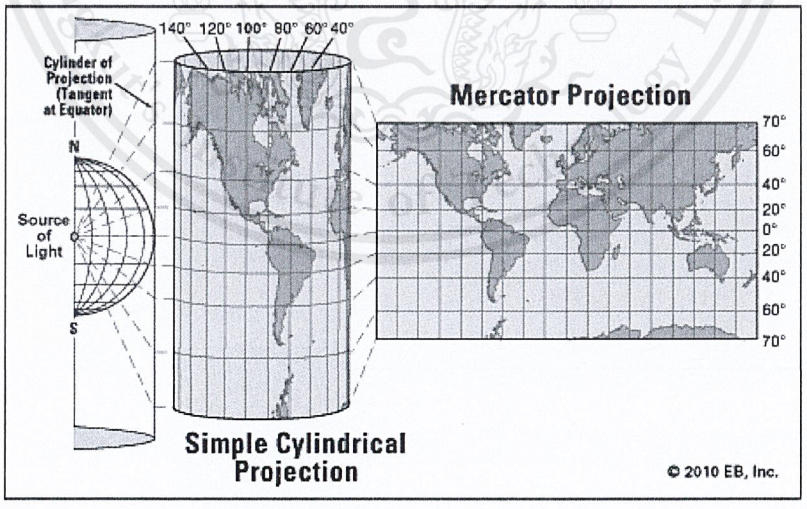

Figure 3.4 Mercator Projection

But the projection we use in our map is Spherical Mercator projection which treats the earth as a sphere instead of an ellipsoid as shown in Figure 3.4. This

projection is used by Google Maps and OpenStreetMap, which we use on our application. Spherical Mercator has an official designation of EPSG:3857. However, before EPSG:3857 was established, a large amount of software used the identifier EPSG:900913, including OpenLayers 2. This projection describes coordinates in meters in X/Y values. Therefore, when we communicate with the server side, we have to transform the projection using the method provided by OpenLayers library.

3.4 Visualization of GIS data model

In order to map 3D world into 2D, we need to present each type of dataset as a map layer.

Map layer is a visual geographic dataset represents in digital map. It is a slice of the reality in particular area as shown in Figure 3.5. It collects individual feature with geographic location as well as information which stored as an attribute. Each layer is independent from each other. In Bangkok, for example, people, buildings, rivers might be considered different layers.

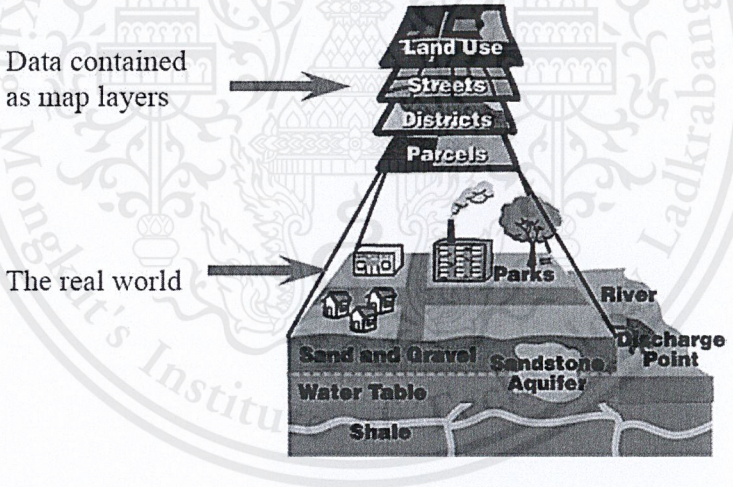

Figure 3.5 Conceptual model of GIS

3.5 Client-side Application Development

3.5.1 HTML5

HTML5 is the newest hypertext markup language for web application from the World Wide Web Consortium (W3C). In 2011, HTML5 was released and people started to talk about it and use it, but the support in different browsers was still poor. Today all

major browsers (Chrome, Safari, Firefox, Opera, Internet Explorer) offer HTML5 support, therefore the newest HTML technology can be used at its best today.

HTML5 specifies scripting application programming interfaces (APIs) that can be used with JavaScript. There are important APIs we use in this project, which are:

- Geolocation is used to get the geographical position of user. Since this can compromise user privacy, the position is not available unless the user approves it. Geolocation is more accurate when the user uses devices with GPS. It is supported by Internet Explorer version 9 or later, Chrome, Safari and Opera.
- Web Notification allows the web to display notification for given events to end users. The notification displays in the areas outside the web page for example, bottom-right corner of the desktop or the home screen of a mobile device. It is still in draft spec, not yet in any standard.

3.5.2 JavaScript

The web application is developed using a client-side scripting language, JavaScript, and though it benefits from:

Usability: It can modify a page without having to post back to the server; as a result, it helps reducing server workloads implying faster UI.

Efficiency: It can make small, quick changes to page without waiting for server since the business logic lives on the client side as well as other related resources are stored on the server.

Event-driven: It can respond to user actions like clicking and key pressing. Next section, we discuss this topic in more detail.

Client-side scripting language, JavaScript in this case, is often integrated within the HTML document. However, we can create it on separate file which is referenced by the document that uses it; for example, you can just simply include it in the HTML header like:

```
1  ...
2  <head>
3  <script type="text/javascript"
4  src="site_media/scripts/mapcontrol.js"></script>
5  </head>
6  ...
```

From the Figure 3.6, when request occurred (1), the necessary files are sent to the user's computer by the web server on which they reside (2 and 3). The user's web browser executes the script, then displays the document (4), including any visible output from the script. Client-side scripts may also contain instructions for the browser to follow in response to certain user actions, (e.g., clicking a button). Often, these instructions can be followed without further communication with the server.

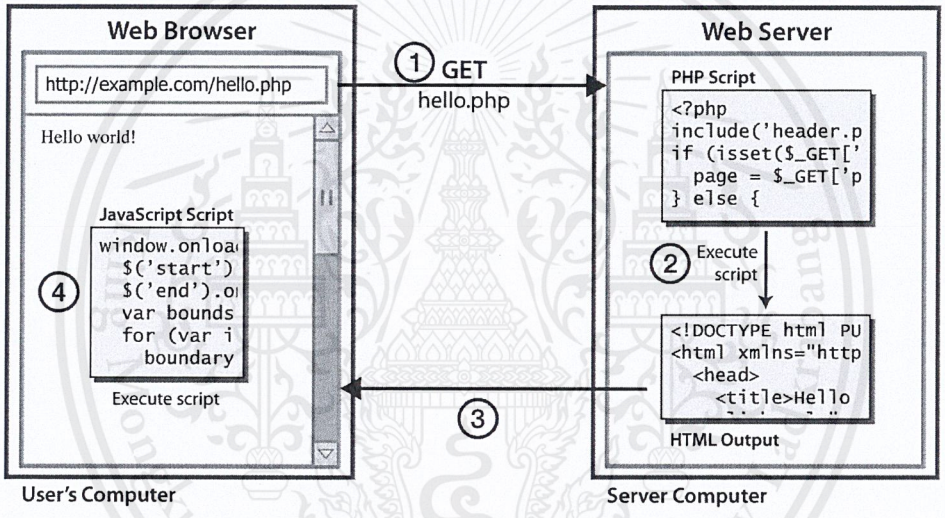

Figure 3.6 Client-side scripting mechanism

3.5.3 jQuery

jQuery is the most popular JavaScript library in use today, it is free and open-source. It is aimed to make the program easier to navigate a document, select DOM objects, handle events, create animations and develop AJAX applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets. Since it is a JavaScript library, it benefits the followings:

- Code runs in browser (executed on the client-side) after page is sent back from server.
- This code manipulates the page or responds to user actions.

- This type of computer programming is an important part of the Dynamic HTML

3.5.4 AJAX

AJAX stands for Asynchronous JavaScript and XML. It is a mechanism describes the process how data is sent and received between browser (client-side) and server. By using this method, we can send a request to and receive the response from a server without refreshing the whole page as shown in Figure 3.7. For example, every time the user pan the map around, OpenLayers makes a series of requests to get new map images to the server. The server then sends a new map image back as a response for OpenLayers to display on web browser.

Without using AJAX, the user has to refresh the web page every time the user interact with the map.

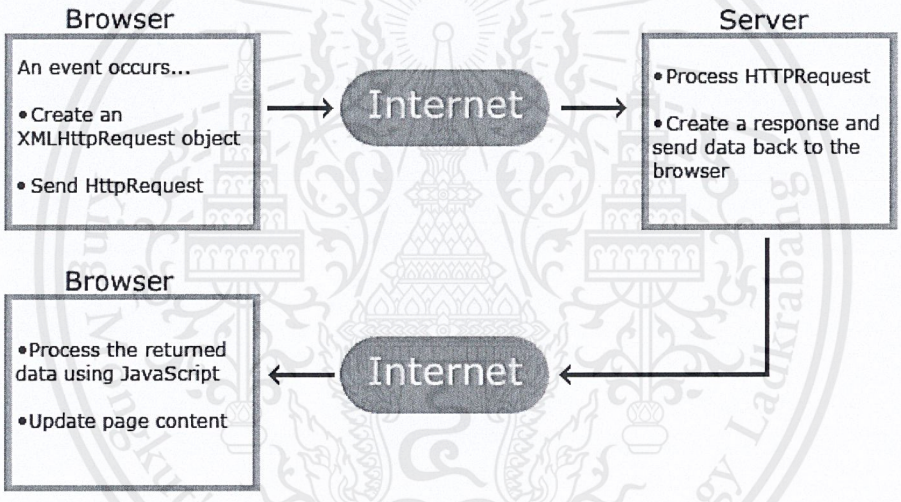

Figure 3.7 AJAX

3.5.5 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is human-readable. It is also easy for machines to parse and generate. Even though JSON is derived from JavaScript Programming Language but it is a language-independent. It uses conventions which are recognized by programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In most languages, this is realized as an object, record, dictionary, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

3.6 OpenLayers

OpenLayers is a client-side JavaScript library for displaying map data on web browser without server-side dependencies. It is very powerful, free, open source and has a strong community behind it. OpenLayers lets the user to build web map applications from the ground up and also customize any part of the map. It is also cross- browser compatible.

OpenLayers lives on the client-side. The main task is to get map images from a map server. Every time the user navigate, zoom or pan the map, the client has to send new requests to the server via asynchronous JavaScript (AJAX) which handled by OpenLayers.

OpenLayers is currently in version 2.13.1 (May 19, 2014) but the developing team have already begun to make an effort toward version 3 which provides many major changes. OpenLayers 3 is a rewrite of the library and editing functionality from OpenLayers 2. It is now released as a beta version. The new version focuses on friendlier visual components with lighter builds, performance improvement and more. These are the example of the major highlights;

- WebGL – OpenLayers 3 will offer WebGL which brings 3D capabilities and increased performance for all mapping needs to the latest browsers.
- Cesium – OpenLayers 3 will integrate new Cesium library to enable fill 3D spinning globe capabilities.
- Closure Compiler – It is a tool for making JavaScript download and run faster. Instead of compiling from a source language to machine code, it compiles from JavaScript to better JavaScript. It parses your JavaScript,

analyzes it, removes dead code and rewrites and minimizes what's left. It also checks syntax, variable references, and types, and warns about common JavaScript pitfalls. By utilizing the Closure Compiler, applications developers will be able to create smaller and faster libraries, easing the use of the extensive OpenLayers 3 toolkit.

- New codebase – It offers an opportunity to clean up some of the complicated ways of doing things in OpenLayers.

Even OpenLayers 3 is a bata version but we choose this version to implement our project. The reason is because the project will be able to develop and support more functionalities and features furthermore in the future.

3.7 Gyroscope and Orientation

A gyroscope is a device for measuring or maintaining orientation, based on the principles of angular momentum. Mechanical gyroscopes typically comprise a spinning wheel or disc in which the axle is free to assume any orientation. In Figure 3.8 shows how mechanical gyroscope looks like and its components. And Figure 3.9 shows all three axis of forces associated with gyro wheel.

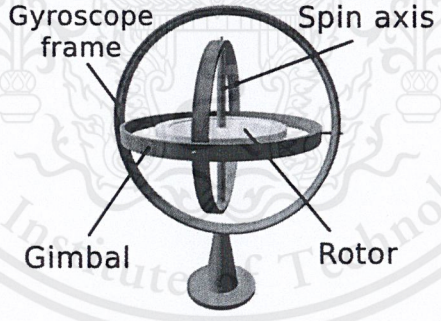

Figure 3.8 Components of Gyroscope

This concept of gyroscope is used in mobile technology industrial to produce microchip-packaged MEMS (Micro-Electro Mechanical System) gyroscope which can measure the orientation of the devices. It can sense motion including vertical and horizontal rotation. There are a lot of practical uses of gyroscope especially in mobile games and other applications.

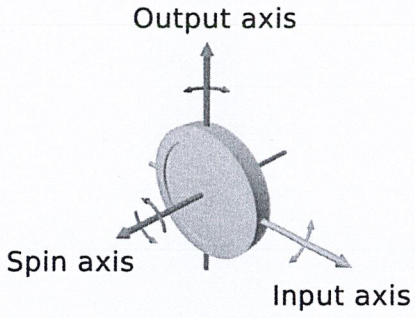

Figure 3.9 A Gyro Wheel

In this project, we create web application which can run on mobile device. It is very nice to have a use of gyroscope interacting with the map. We provides a function called tracking which show the current location of the user in real-time with map rotating when the user turn left or right corresponding to gyroscope.

Chapter 4

Requirements and Analysis

4.1 Problem Description

The Geographic Information System (GIS) has been used widely, especially on web application to provide location services. Since the rapid growth of populations has cause the world to face challenges such as how to improve management of resources on map, and also, there are lots of accident happened every second, minute, and hour. Occasionally, we might confront with unexpected situations such as fire conflagration or road accident, thus, the speed and timing are very important in order to save people's lives and deliver help.

The faster we can get victims to the hospital, the better chance they can survive. With those in mind, we applied a diverse range of GIS tools with our web application to provide the user with the ability to specify the current location of the user as well as the exact position on the map. In addition, the user can make a request from that position and send a request to staff, a person who is responsible to take care the user's request, so we can ensure that the user will be delivered help in time. These are just scenarios out of many cases which become the objective of our project to use GIS technology to help with emergency service of the Thai Red Cross organization.

The main idea is to provide a web based GIS application for the user to ask for emergency response. First, we needs to create a map viewer with accuracy data using OpenLayers and also tools for user to interact with the map. Second, we have to provide the functions for users to send requests to the staff. Third, after users send the requests, we need to provide functions for staffs to handle and respond them. These three tasks are the major things in the application.

4.2 Requirements

4.2.1 Functional requirements

R1 The user can register for a new account to log in to the system.

R2 The system provides log in/log out function for registered user.

R3 The user can filter the map layers; for example, building layer, center line layer, landmark point layer, etc.

R4 The user can search for location either by area or layer types.

R5 The system provides GIS tools for the user such as

- A function to get the user's current location and pin that location with a marker as well as provide a pop-up balloon showing that location's details.
- A function to rotate the map. This function is capable for use with tradition mouse or, in the case that the user use touch mobile devices, touch gesture.
- A function to measure both area and distance.
- A function to print the current view of the map or export it as a .png file.

R6 The system allows the user to pin a marker on any location on the map as well as view location detail when the user clicks on that marker. They can also choose different types of marker provided by the system.

R7 The system allows the user to submit emergency request form. The information required includes the details of the accident, the level of seriousness, expected response time, etc. Additionally, the user can attach files such as picture to the system.

R8 The system allows the user to view progress of submitted requests and get notifications. If the user is a type of 'Staff', he can view all user requests.

R9 The system allows the staff to handle and respond to the request.

R10 The system allows the user to get notified when their request has been responded by staff.

4.2.2 Non-functional requirements

N1 The system is developed using HTML5, CSS3, and JavaScript to provide the user with an east-to-use user interface. In addition, two JavaScript libraries which are jQuery and OpenLayers are also used as well.

N2 The system applied AJAX techniques frequently so the web page can respond to the user request without having to refresh the page.

N3 The system can be viewed across a variety of devices such as desktop PC, mobile phone or tablet.

N4 The system should have reasonable performance.

N5 The system is reliable. It should display everything on the map precisely and the position on the map should be geographically correct.

N6 The system is flexible. It is designed for future maintenance and development. It can be upgraded to have more features.

N7 The system should be deployed on Mozilla Firefox mobile browser in order to fully support both device rotation and orientation.

4.4 Use Case Diagram

The entire system composes of two sides; those are, a server-side service and a client-side application. The Figure 4.1 shows a use case of the entire system. Our client-side system is shown in the red box. For more details, please see it in the next section.

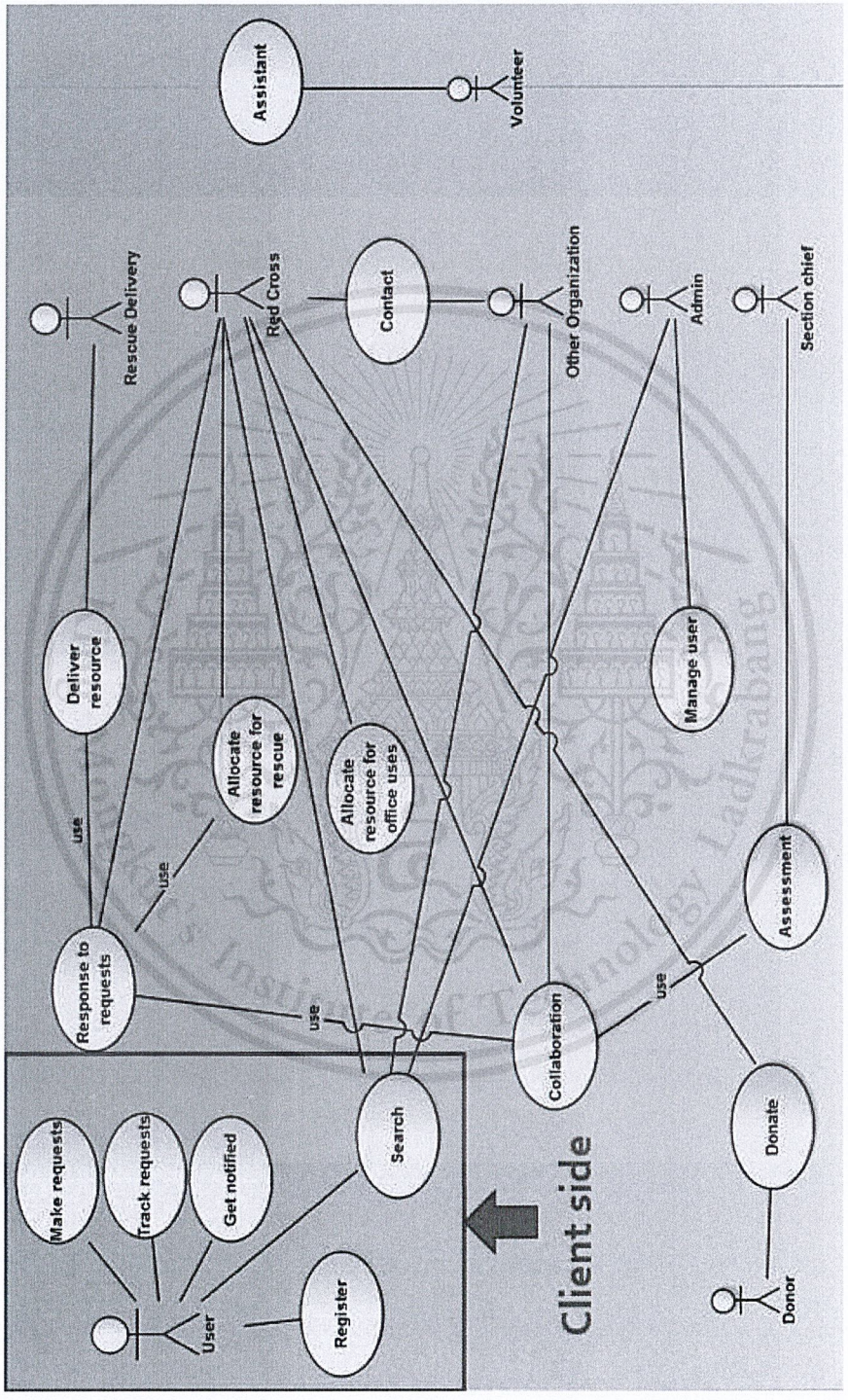

Figure 4.1 Use case diagram of the entire system

The following Figure 4.2 shows a use case diagram for client-side GIS web application for which we are responsible.

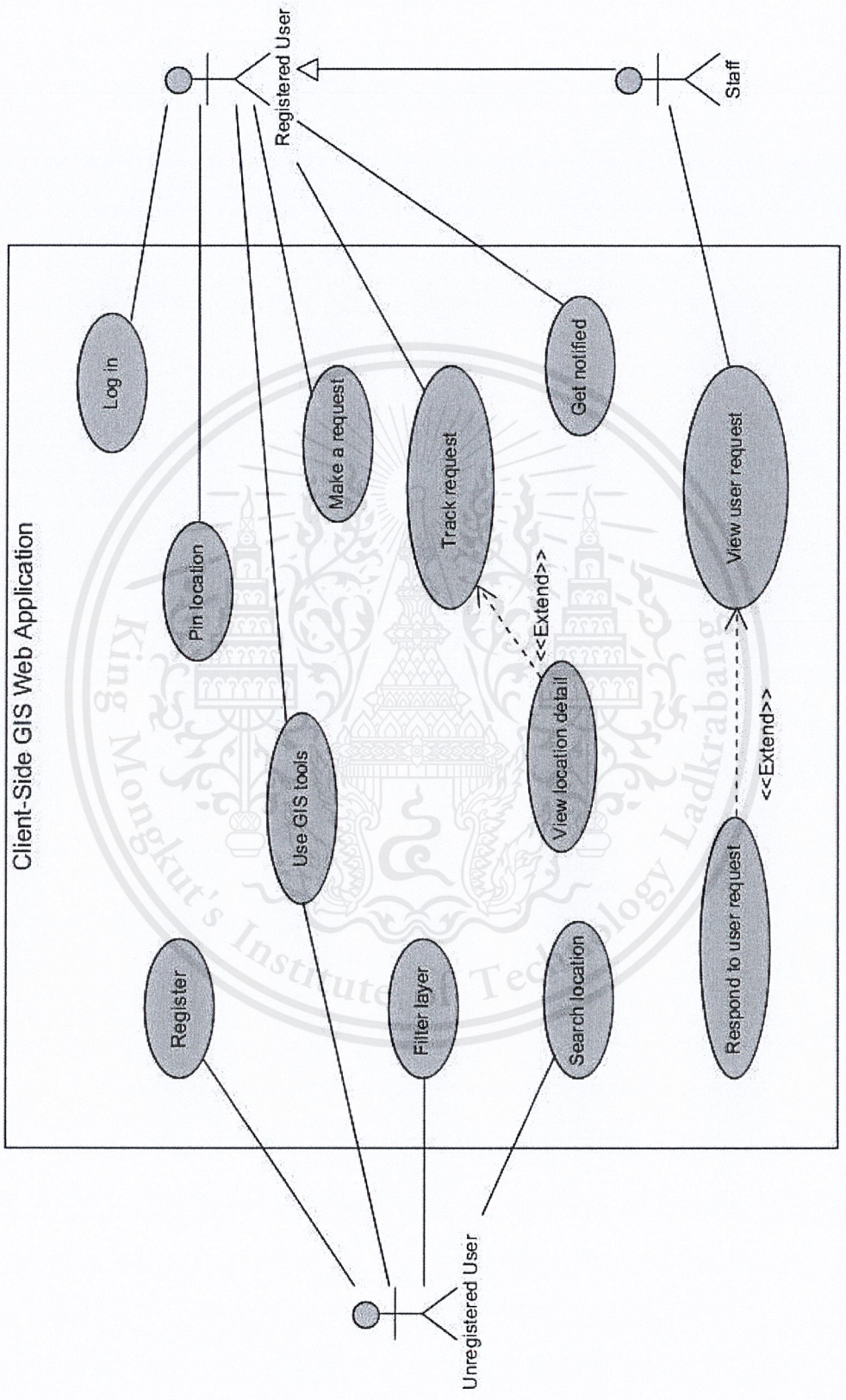

Figure 4.2 Use case diagram for client-side application

4.4.1 Use Case Descriptions

The following use case descriptions describe each use case in detail including,

- **Use Case** which mentions the name of that use case.
- **Cross-reference** which refers to its corresponding functional requirement.
- **Actor** which describes all of the actors involved to that use case.
- **Description** which describe each use case in detail.

Please refer to section 4.2.1 Functional requirements for the cross-reference of each use case.

Use Case: Register	**Cross-reference:** R1
Actors: Unregistered User	
Description: This use case provides a register form for new user to create a new account. When the user clicks on a register button, he is required to fill in his information to login to the system. The information includes account type, username, first name, address, etc.	

Use Case: Log in	**Cross-reference:** R2
Actors: Registered User, or Staff	
Description: This use case is for a registered user or staff who is already registered to the system. When the user clicks on a register button, he is required to fill in his information to login to the system. The information includes account type, username, first name, address, etc.	

Use Case: Filter layer	**Cross-reference:** R3
Actors: Unregistered User, Registered User, or Staff	
Description: This use case allows the user to filter which layer to be displayed on the map. The user can also select multiple layers at the same time. However, there are a group of landbase layer which causes the behavior of subsequent layers. If the landbase layer is not selected the subsequent layers cannot be selected as well.	

Use Case: Search location	**Cross-reference:** R4
Actors: Unregistered User, Registered User, or Staff	
Description: This use case provides a search function for the user to search any locations that are available in a specific district. The specific district is provided by the server side. The user can also search location by select a specific are or select by layer. When the result is displayed, the user can click on the location title and the map will automatically pan to that location.	

Use Case: Get current location	**Cross-reference:** R5
Actors: Unregistered User, Registered User, or Staff	
Description: This use case allows the user to get their current location when used. It will automatically pan the map to that location. Additionally, the marker will be pinned on that location as well.	

Use Case: Rotate map	**Cross-reference:** R5
Actors: Unregistered User, Registered User, or Staff	
Description: This use case provides the map rotation function for the user. This function allows the user to rotate the map either in clockwise or counterclockwise direction. In addition, this use case can be used by traditional mouse or touch gesture.	

Use Case:	Measure distance	**Cross-reference:**	R5
Actors:	Unregistered User, Registered User, or Staff		
Description: This use case provides a measurement function to measure the distance on map.			

Use Case:	Measure area	**Cross-reference:**	R5
Actors:	Unregistered User, Registered User, or Staff		
Description: This use case provides a measurement function to measure the area on map.			

Use Case:	Print map	**Cross-reference:**	R5
Actors:	Unregistered User, Registered User, or Staff		
Description: This use case allows the user to print the map. The map to be printed out will match the current viewport. For example, if the user zooms into the map for a certain level and they check on the building layer. As a result, the map will be printed out with that zoom level as well as the building layer.			

Use Case:	Export map as PNG	**Cross-reference:**	R5
Actors:	Unregistered User, Registered User, or Staff		
Description: This use case behaves similarly to the 'Print map' use case. Except that instead of printing out, the system will automatically export the map as a '.png' file with the same details of the 'Print map' use case.			

Use Case: Pin location	**Cross-reference:** R6
Actors: Registered User, or Staff	
Description: This use case allows the user to pin any locations on the map and then they can fill in the request form and submit it to the server. The user can also attach a file or image. There are multiple types of marker available for the user to choose including default marker, ambulance marker, etc.	

Use Case: View location detail	**Cross-reference:** R6
Actors: Registered User, or Staff	
Description: This use case allows the user to view location detail; for example, the geographical coordinate, of that location pinned by the user.	

Use Case: Make a request	**Cross-reference:** R7
Actors: Registered User, or Staff	
Description: This use case allows the user to pin any locations on the map and then they can fill in the request form and submit it to the server. The user can also attach a file or image. There are multiple types of marker available for the user to choose including default marker, ambulance marker, etc.	

Use Case: Track request	**Cross-reference:** R8
Actors: Registered User, or Staff	
Description: This use case allows the user to track their submitted request in order to check whether the request has been responded by staff. The user can also filter the request status they would like to see; for example, 'All', 'Requested', 'Pending', or 'Done'. When the result is displayed, the user can click on the request title and the map will automatically pan to that location.	

Use Case: View user request	**Cross-reference:** R8
Actors: Staff	
Description: This use case behaves similarly to the 'Track request' use case except it differs in detail and can only be used by 'Staff'. This use case allows the actor to view all of the user requests which correspond to the area the staff is responsible. For example, if the staff is responsible for 'Klong Toei District', and when he uses this use case the result will display only the request from 'Klong Toei District'.	

Use Case: Respond to user request	**Cross-reference:** R9
Actors: Staff	
Description: This use case allows staff to respond to a user request for a particular area for which he is responsible. The information provides to the user includes name of the staff, arrival time, helping detail, etc.	

Use Case: Get notified	**Cross-reference:** R10
Actors: Registered User, or Staff	
Description: This use case allows the user to be notified in real-time manner as soon as their request is responded by staff. The information when notification pops up include the topic of that request and the name of responsible staff. The user can view more details by clicking on a pop up balloon.	

Chapter 5

Software Design

In this chapter, we describe more detail of our system which includes:

- The system architecture
- The client-side scripting language
- The event-driven programming
- Graphical user interface design of the system
- Techniques used in development

5.1 Overall GIS Architecture

The architecture of the entire system can be described as follows (see Figure 5.1):

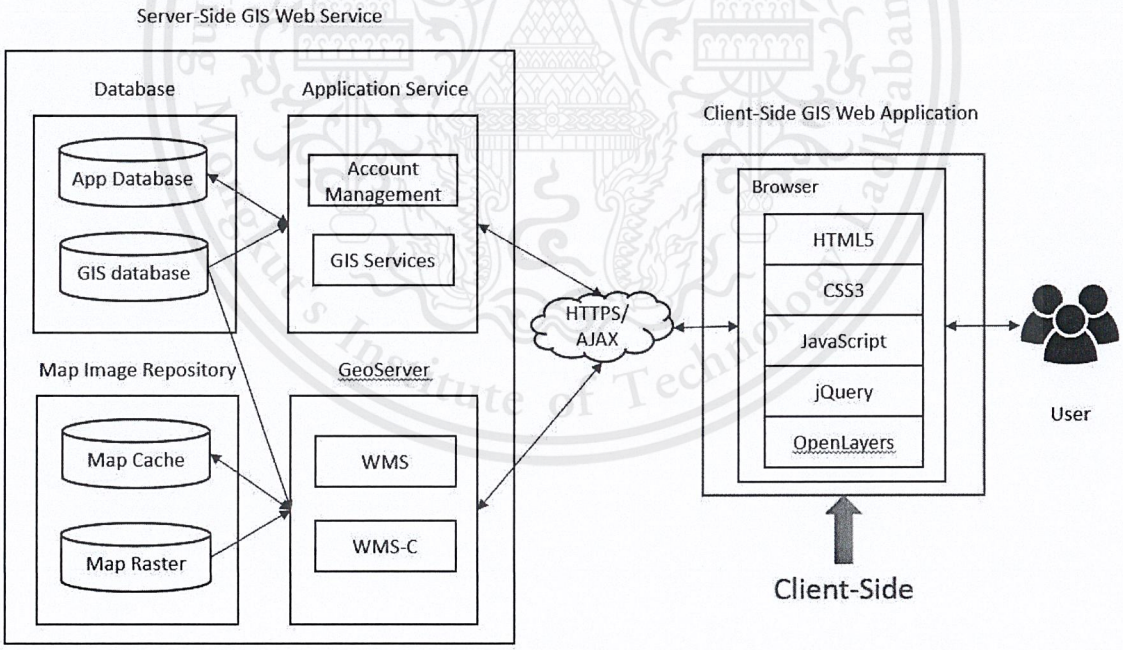

Figure 5.1 System Architecture

The whole system consists of server side and client side. For this project, we are responsible for the client side. The following describes the overview for each side, so you can get the idea and understand the responsibility for each of them.

GIS Web Server: It is the server side of this system providing:

- User database such as storing user's account information, etc.
- GIS database such as map configuration, detail of each map layer, etc.
- Map image repository such as map cache
- Application services such as account management, GIS services

GIS Web Client: It is the client side of the system that we are responsible to develop. It includes:

- OpenLayers which is a JavaScript library responsible for GIS-related tools development
- jQuery and JavaScript are used to manage the behavior of web page as well as handling driven by the user's event. It is also used when we interact with the server side
- HTML5 which is used to markup our web page to ensure that the design is easy to use
- AJAX which is used to deal with data communication between client-side web application and server-side web service (see Figure 5.1). The web application is first loaded and styled with HTML5, CSS3, respectively, and the browser will execute JavaScript files necessary for first time loading. The data is interchanged via XMLHttpRequest using JSON.

The Figure 5.2 shows the system architecture in detail for client-side web application. The main components include external and internal elements. The external elements are user and server-side part. The internal element, the client-side part, is a component which directly interacts with user providing a user interface. On the other side, it provides the communication part to send data to and receive data from the server.

There are plenty of sub components involved in the system. Most of them are omitted to simplify the system architecture and reduce complexity. However, we will explain them as a scenario and include those components when necessary.

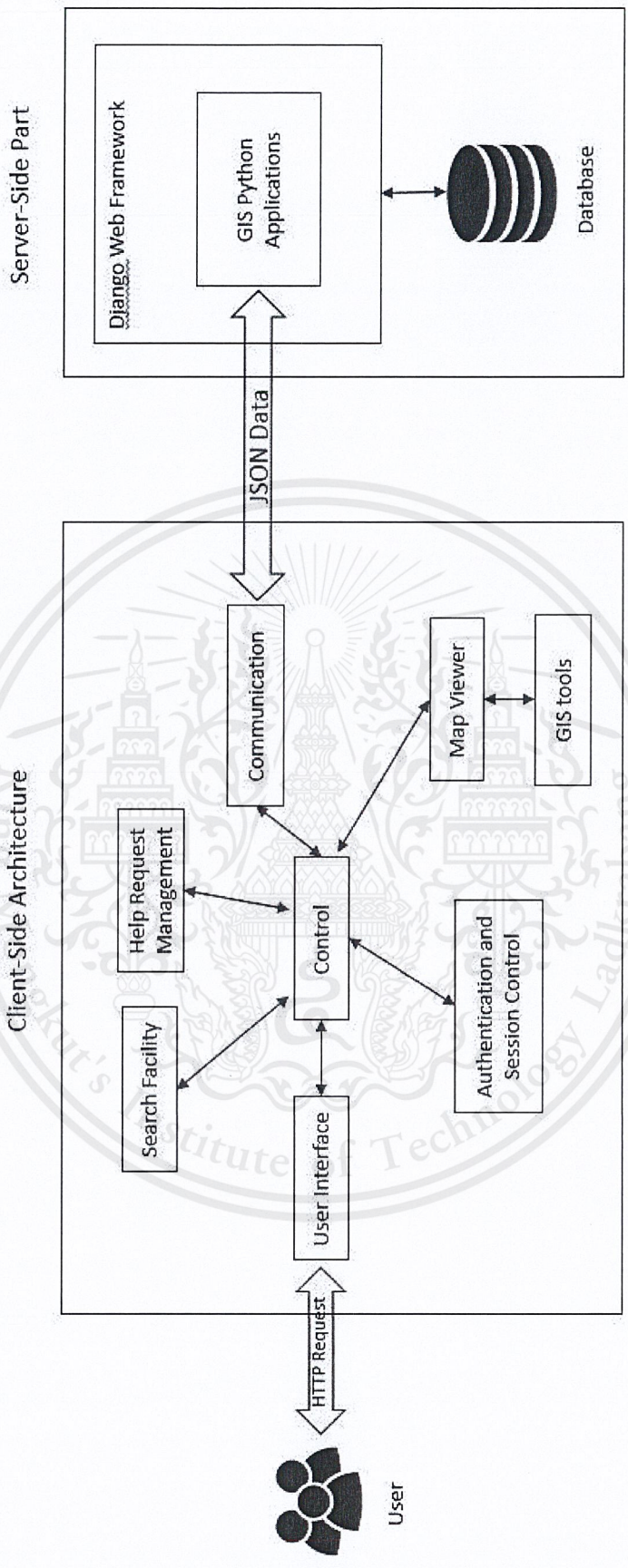

Figure 5.2 Client-Side Architecture

From the figure above, we describe each module on how it works as a scenario. Please note that the underlined words enclosed in a single quote are the main components corresponding to the above system architecture and those without the underline are sub components.

- The page will be loaded up with HTML, JavaScript and other related elements for the first time the user enters the system via HTTP Request. The system allows the user to interact with the map by providing a **'User Interface'**. The user interface interacts with **'Control'** to provide functionalities such as:
  - **'Search Facility'** which is responsible to provide search location function such as search by boundary or search by layer.
  - **'Authentication and Session Control'** provides the user with log in/log out interface as well as the registration window. It also provides session mechanism in order to keep session of the current user, thus, they do not have to log in again when they close the browser.
  - **'Map Viewer'** which is responsible to display the map to the user as well as to initialize map such as setting the center of the map when the page is loaded, setting the current view, initialize the base layer of the map. It is also responsible for projection transformation in order to convert to a proper unit agreed with the server side.
  - **'Layer Management'** is a conceptual model represents a layer filter function. It allows the user to turn the visibility of each layer to be on or off including a group of layers (please refer to section 4.4.1, navigate to 'Filter layer' use case for more detail).
  - **'GIS Tools'** includes an implementation of GIS tools which are geolocation, orientation, distance and area measurement, printing mechanism. For more detail, please refer to Chapter 7.
- Once the user logs into the system, they are allowed to use several services provided by the system including:
  - **'Help Request Management'** includes the form for users to fill in details of submitting request. It also provides **'Track Request'** function which is used to track user's submitted request to check

whether it is handled by staff or not. The status of a request includes 'Requested', 'Pending', or 'Done'. Additionally, it services a **'Request Form'** which includes the form for users to fill in details of submitting request.

- **'Communication'** provides a channel for the client side to send data to and receive data from the server. When the user uses a service, says 'Make a request', the data of the user will be processed in the form of JSON data and it is sent to the server using AJAX code via HTTP Request.
- The server-side part is developed using **'Django Web Framework'**. Once the data arrives at the server-side part, the help request data for example, it will be processed via **'GIS Python Application'**. The staff can then view or respond to user request using database provided by the GIS python application which is further connected to database to store user information.

Figure 5.4 shows how we structure each component from Figure 5.2 on our client-side web application. The following describe each component in detail and illustrate some scenarios corresponding to its component.

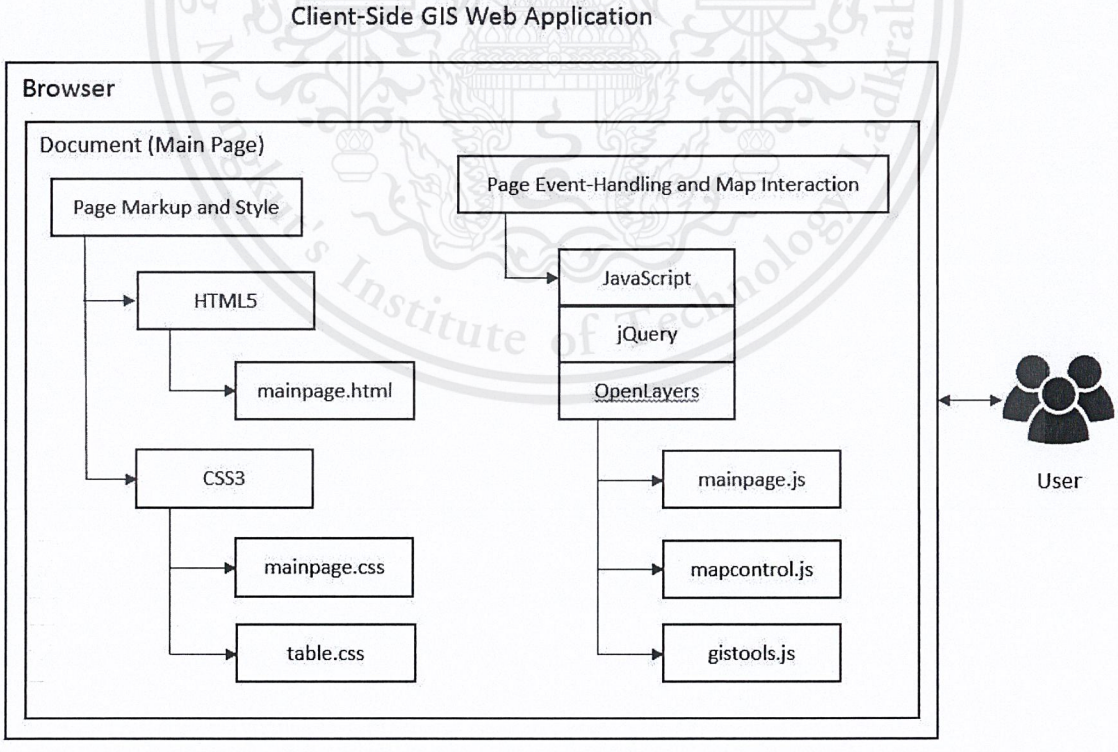

Figure 5.3 Client-Side Web Application Structure

- A 'mainpage.html' file is responsible for marking up the main page of the system. It is rendered together with 'manipage.css' and 'table.css' files to style the page according to our design. The design includes the layout of side menu, a set of GIS tools, map layout. In addition, the fill-in forms are also defined here; for example, log in form, request form, respond form. The 'table.css' file is a style sheet for table
- A user enter the system via http://www.i-emergency.com/gis/, the page is loaded up when a registered user would like to make a request, he must log in to the system.
- Any GIS-related tools are dealt with 'gistools.js' including a function to get user's current location, a function to measure distance and area, etc.
- A 'mapcontrol.js' is responsible for map initialization, map management, and other relate settings, for example, setting the level of details (please refer to section 5.4 for more information), setting of a map viewer (section 6.1), etc.

5.2 Event-Driven Programming

In order to deliver the user with interactive web application, we have used JavaScript to handle the event-driven throughout the project (see Figure 5.4). There are many points we created interactive functions for the user to interact with the map including when the user wants to measure the area, the user can drag the cursor to specify the starting point and the end point of that area.

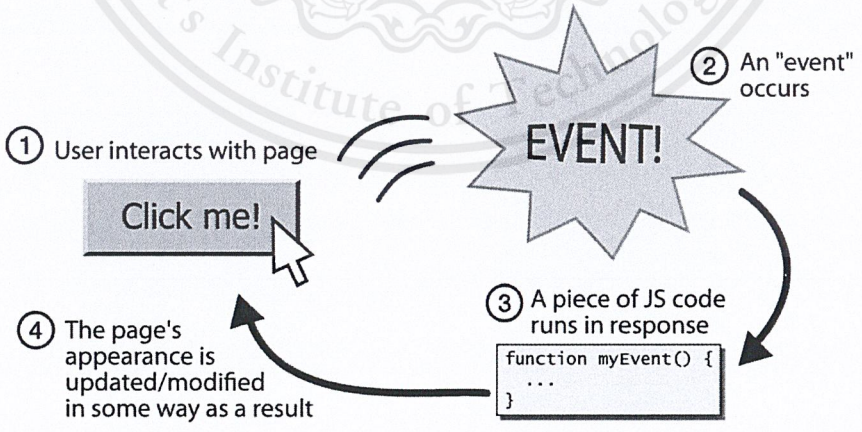

Figure 5.4 Event-driven scenario

5.3 Graphical User Interface Design

A good web application should have a beautiful design as well as well-organized functions. Such design makes the user feel comfortable and convenient to use while we can still deliver them with a feature-rich web application. Similarly, our client-side web application is not an exception. The GUI of our web application was inspired by the works mentioned in Chapter 2 (Please refer to section 2.1 Review of related work). The interface consists of three main components which can be illustrated in the following figure.

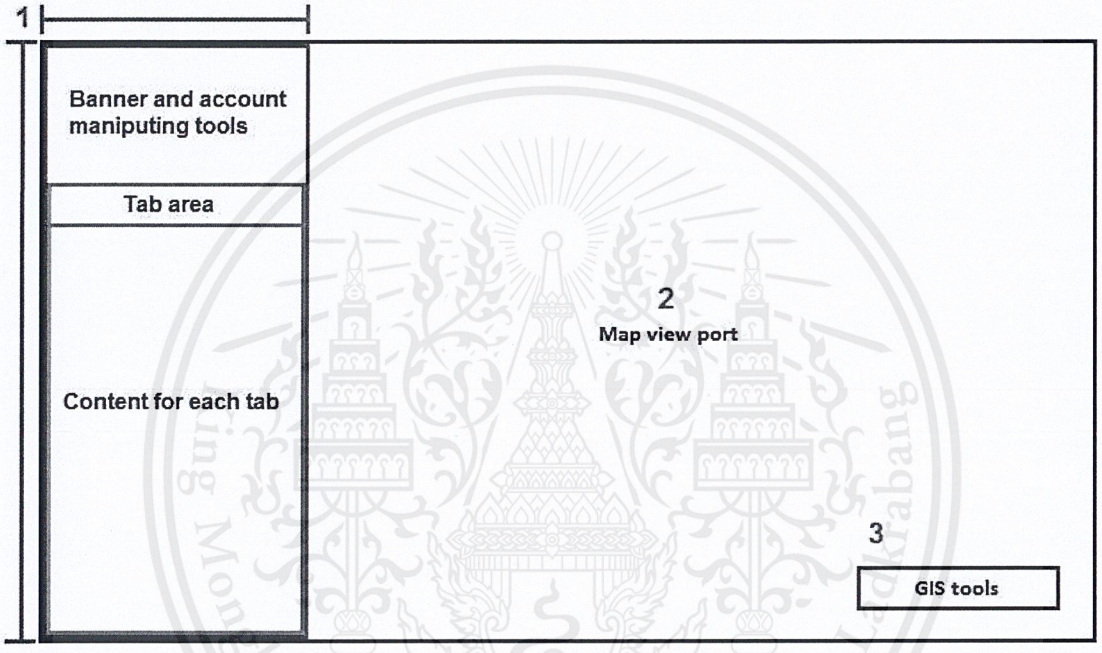

Figure 5.5 The design structure of the web page

From Figure 5.5, we can describe each element in detail as follows:

Firstly, the side menu (denoted as 1), it is the main component that the user uses frequently. It is designed to behave like a toggle menu which allows the user to toggle it on or off. To put it simply, if the user click to toggle the menu off, it will be animated smoothly to the left and be hidden there until the user click to toggle it on.

This is to provide the user with a bigger viewport of the map, so the user can enjoy navigating the map and making use of that available space.

Additionally, the side menu consists of the following subcomponents:

- Banner and account manipulating tools: This area is used to place a desirable banner together with account manipulating tools such as register button, log in/log out button, the name of the user.
- Tab area: The tab area describes a way to group a collection of subsequent contents. By using tab menu, we can utilize such a small area to display more contents by hiding others when they are not in use.
- Content for each tab: This area provides space for the each tab's content. For example, the 'Filter map layer' tab has multiple layer checkers as its content.
- Map viewport (denoted as 2): The map viewport is one of the most important component because it also defines the way we arrange other components. Since we would like the user to experience the map as large area as possible, that is why we need to use the toggle side menu.
- GIS tools (denoted as 3): At the bottom right of the screen, the GIS tools provide the user with frequent used tools including 'Get your current location', 'Rotate map', 'Measure distance', 'Measure area', and 'Print map' as well as 'Export map as PNG'.

Please refer to Chapter 7 Result for more precise screenshots of the web application so can see each component in detail.

5.4 Map Display Technique

5.4.1 Level of Details

The technique we use in this application is layers shown on the map viewer called level of detail. We have designed how the map should display the layer so it can provides the most convenient way to the user.

For example, when the user views the map in max extent scale, the building layer should not be displayed to user, otherwise it will be seen as a big mass cover the area of the map. Therefore we put the condition for each layer in which zoom level should it displayed on map. The following partial code illustrates the mentioned idea so that you can understand easily.

To illustrate the idea, consider this example. The Figure 5.6 shows the build layer on the map without setting 'maxResolution' value. You can see a big mess of building blocks as a result you cannot specify the exact position on the map.

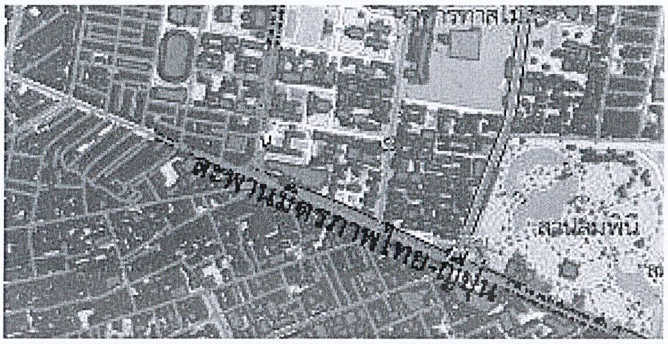

Figure 5.6 Building layer before without maxResolution

The Figure 5.7 below shows the build layer on the map with 'maxResolution' value. What you can see now is a typical OSM base map. The building layer is supposed not to be display at this zoom level.

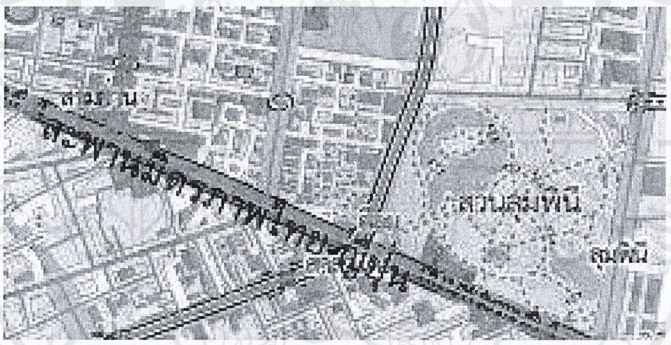

Figure 5.7 Building layer before with maxResolution

After a certain zoom level is reached (see Figure 5.8), the building layer will be appeared automatically. Thus, you can clearly see which building belongs to a label specified on the map.

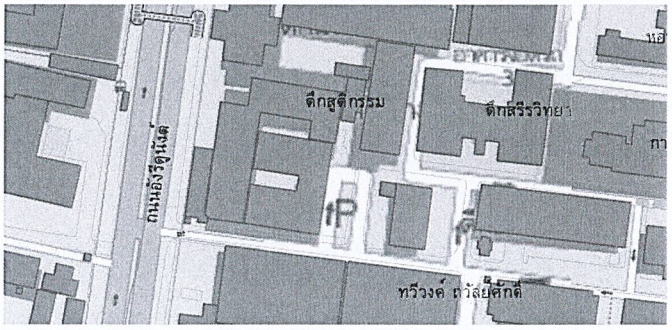

Figure 5.8 Building layer on certain zoom level

The follwing code shows how we apply 'maxResolution' to make 'Level of Details'.

```
1   …
2   }), new ol.layer.Group({
3         layers : [new ol.layer.Tile({
4               source : new ol.source.TileWMS({
5               url : 'http://www.i-
6   emergency.com:80/geoserver/Proj_GIS/wms',
7               params : {
8                     'LAYERS' : 'Proj_GIS:building',
9                     'TILED' : true
10                    },
11              serverType : 'geoserver'
12        }),
13        maxResolution: 2
14  }),
15  …
```

Line #13 demonstrates the attribute used to define the zoom level. When the user zooms map into a certain amount the map will automatically be displayed and be hidden it the user zooms out of the defined range.

Chapter 6

Development

In chapter, we discuss on system development in detail including how we can display the map on our web page, how we can deal with geolocation and map orientation. In addition, it also includes methods we use for data manipulating such as 'GET & POST', JSON data structure and, finally, the comparison between OpenLayers and OpenLayers 3 (beta).

6.1 Map viewer

The partial code below shows how we can display the map on the web browser.

```
<div id="map" style="width: 100%, height: 400px"></div>
<script>
  new ol.Map({
    layers: [
      new ol.layer.Tile({source: new ol.source.OSM()})
    ],
    view: new ol.View({
      center: ol.proj.transform([37.41, 8.82],
'EPSG:4326', 'EPSG:3857'),
      zoom: 2
    }),
    target: 'map'
  });
</script>
```

- **Map**

```
<div id="map" style="width: 100%, height: 400px"></div>
<script>
  var map = new ol.Map({target: 'map'});
</script>
```

The core component of OpenLayers is the map (ol.Map). It is rendered to a target container (e.g. a div element on the web page that contains the map). All map properties can either be configured at construction time, or by using setter methods, e.g. setTarget().

- **View**

```
1  map.setView(new ol.View({
2      center: ol.proj.transform([37.41, 8.82], 'EPSG:4326',
3  'EPSG:3857'),
4      zoom: 2
5  }));
```

ol.Map is not responsible for things like center, zoom level and projection of the map. Instead, these are properties of an ol.View instance - typically an ol.View2D for 2D maps. The reason for this abstraction is the idea of instantly switching e.g. between a 2D and a tilted 3D view, without the need to maintain two copies of the layers.

The center specify in latitude/longitude coordinate (EPSG:4326). Since the layer used is in Spherical Mercator projection (EPSG:3857), the x/y values need to be reproject on the fly to be able to zoom the map to the right coordinates.

- **Source**

```
1  var osmSource = new ol.source.OSM();
```

To get remote data for a layer, OpenLayers 3 uses ol.source.Source subclasses. These are available for free and commercial map tile services like OpenStreetMap or Bing, for OGC sources like WMS or WMTS, and for vector data in formats like GeoJSON or KML.

- **Layer**

```
1    var osmLayer = new ol.layer.Tile({source: osmSource});
2    map.addLayer(osmLayer);
```

A layer is a visual representation of data from a source. ol.layer.Tile is a type os layers. It is for layer sources that provide pre-rendered, tiled images in grids that are organized by zoom levels for specific resolutions.

6.2 Map geolocation and orientation

There are two important terms we used to implement 'Get current location' function and 'Rotate map' function. Those are 'Geolocation' and 'Map orientation'. First, 'Geolocation' which is used to identify the exact position (can be referred to location) of the user at that time. Second, 'Map orientation' which is, in this case, an event-handler we use to make the map to be oriented on any direction and in any devices.

- **Geolocation**

```
1  var geolocation = new ol.Geolocation();
2  geolocation.bindTo('projection', view);
```

This is to define Geolocation and takes the projection to use from the map's view using bindTo function.

```
1  geolocation.on('change', function() {
2        $('#accuracy').text(geolocation.getAccuracy() + '
3  [m]');
4        $('#altitude').text(geolocation.getAltitude() + '
5  [m]');
6        $('#altitudeAccuracy').text(geolocation.getAltitudeA
7  ccuracy() + ' [m]');
8        $('#heading').text(geolocation.getHeading() + '
9  [rad]');
10       $('#speed').text(geolocation.getSpeed() + ' [m/s]');
11 });
```

This to listen to the change in position and update the HTML page when the position changes.

```
1  geolocation.on('error', function(error) {
2        var info = document.getElementById('info');
3        info.innerHTML = error.message;
4        info.style.display = '';
5  });
```

This to handle the error caused by calling geolocation in case the browser is not support or users do not allow the device to user current location.

Put it all together.

```
var geolocation = new ol.Geolocation();
geolocation.bindTo('projection', view);


// update the HTML page when the position changes.
geolocation.on('change', function() {
      $('#accuracy').text(geolocation.getAccuracy() + '
[m]');
      $('#altitude').text(geolocation.getAltitude() + '
[m]');
      $('#altitudeAccuracy').text(geolocation.getAltitudeA
ccuracy() + ' [m]');
      $('#heading').text(geolocation.getHeading() + '
[rad]');
      $('#speed').text(geolocation.getSpeed() + ' [m/s]');
});

// handle geolocation error.
geolocation.on('error', function(error) {
      var info = document.getElementById('info');
      info.innerHTML = error.message;
      info.style.display = '';
});
```

When the user clicks on 'Get current location' button, it will automatically pan the map and pin a marker on the user's current location as shown in the Figure 6.1. For more details about the tool, please refer to Chapter 7 Result.

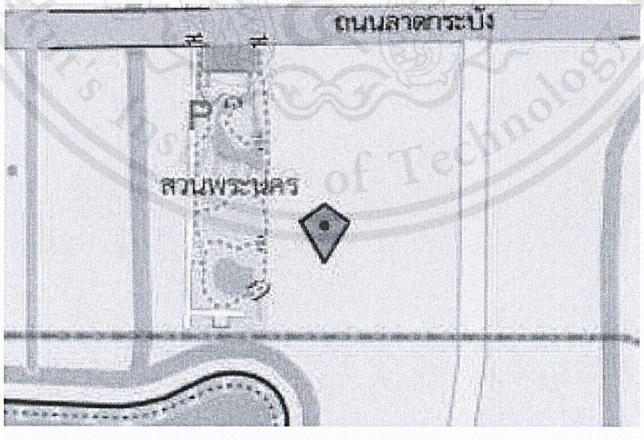

Figure 6.1 Example of using 'Get current location' tool

- **Map orientation**

```
1  var compass = document.getElementById('compass');
```

The compass is an image we will display in order to see the element rotating.

```
1  if(window.DeviceOrientationEvent) { //do something }
```

This line is to check if the browser support the Device Orientation Events API. If yes, the following code will work.

```
1  window.addEventListener('deviceorientation',
2        function(event) {
3              var alpha;
4              alpha = event.alpha;
5  }
```

This is to access to the orientation data of the device and attach an event listener to the device orientation event. The orientation data we concern on is only the data in alpha axis because it shows where the device is heading.

```
1   compass.style.Transform = 'rotate(' + alpha + 'deg)';
```

This line is to rotate the compass image by using the CSS3 transform property. The image will rotate according to the device orientation.

Put it all together.

```
 1  function init() {
 2        var compass = document.getElementById('compass');
 3        if(window.DeviceOrientationEvent) {
 4              window.addEventListener('deviceorientation',
 5  function(event) {
 6                    var alpha;
 7                    alpha = event.alpha;
 8                    compass.style.Transform = 'rotate(' +
 9  alpha + 'deg)';
10          }
11  }
```

When the user clicks on 'Rotate map' button or makes gesture on touch device, the user can rotate the map either in clockwise or counterclockwise direction as shown in the Figure 6.2. For more details about the tool, please refer to Chapter 7 Result.

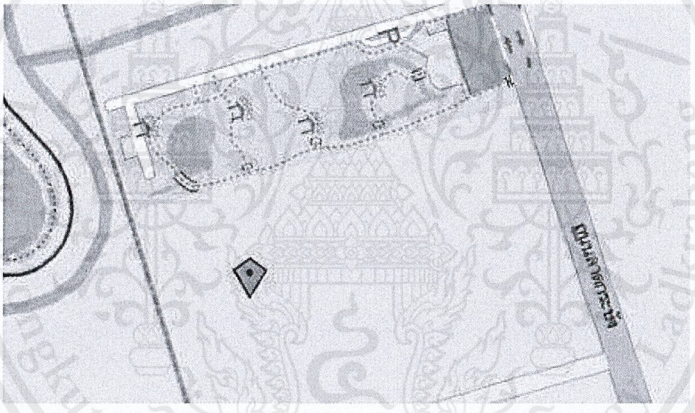

Figure 6.2 Example of using 'Rotate map' tool

6.3 GET & POST

In our project, there two common methods that we use frequently when we implement a function interacting with the server side; those are GET and POST method.

- **GET**: We use 'GET' to make a request to the server without attach any data. For example, we use this method in 'checkSession()' function.

```
1  function checkSession() {
2        //session_data = '{"session",""}';
3
4        $.ajax({
5              url : "http://www.i-
6  emergency.com/server/check_session/",
7              type : "GET",
8              dataType : 'json',
9              success : checkSessionSuccess,
10             error : processError
11       });
12 }
```

The partial code above shows the use of 'GET' method using AJAX in jQuery. Apparently, we have commented out line #2 since we do not need to send any data to the server, instead, what we have to do is to get data from it.

Line #2 specifies the url to which the data is sent. It is provided by the server side and it differs according to the functionality.

Line #7 specifies what method to be used, 'GET' in this case. When we successfully get data from the server, the data will be processed later with a function in line #9, 'checkSessionSuccess', if not it will be processed with the error handling part in line #10, 'processError'. The following partial code shows you the idea of how things are going on after we successfully get data from the server and how we make use of it.

```
1  function checkSessionSuccess(data) {
2        if(data.session == 'has_session') {
3              username = data.username;
4              var user_a = data.first_name + " " +
5  data.last_name;
6              $("#acc_div").css({
7                    'display' : 'none'
8              });
9              //do something
10       }
11 }
```

- **POST**: We use 'POST' when we would like to modify data in the server side including profile updating by the user, submitting help request to the server, or other related processes required data to be attached. To illustrate the idea of this method, please see the following code.

```
1  function authenticate() {
2        authen = '{"username" : "' + log_username.val() +
3  '","password" : "' + log_password.val() + '"}';
4
5        $.ajax({
6              url : "http://www.i-
7  emergency.com/server/login/",
8              type : "POST",
9              data : authen,
10             success : loginSuccess,
11             error : processError
12       });
13 }
```

The partial code demonstrates the 'authenticate()' function which is used to verify the account of the user who is accessing to the system.

Similarly, other attributes perform the same task as in the 'GET' method except that in line #9, the data is required to be sent to the server. By using this method, we can send or modify data in the server side.

The data to be sent must be in the form of JavaScript Object Notation (JSON) which will be discussed in the next section.

6.4 JSON Data

As we mentioned in section 3.5.4 JSON, it is a universal collection of key and value data used in data interchanging between the client side and the server side. The following partial code shows how we define JSON data when the user registers to the system and the user's data is sent to the server.

```
{
    "username":         "reg_username.val()",
    "password":         "reg_password.val()",
    "first_name":       "first_name.val()",
    "last_name":        "last_name.val()",
    "gender":           "gender.val()",
    "citizen_id":       "citizenID.val()",
    "email":            "email.val()",
    "telephone_number": "telephone_number.val()",
    "address_name":     "address_name.val()",
    "house_number":     "house_number.val()",
    "street_name":      "street_name.val()",
    "sub_district":     "sub_district.val()",
    "district":         "district.val()",
    "province":         "province.val()",
    "postcode":         "postcode.val()"
}
```

The followings show an example of data with GeoJSON data. This data is used in 'Submit request' function.

```
{
    "username": "username",
    "type": "FeatureCollection",
    "features": [
        {
            "geometry": {
                "type": "Point",
                "coordinates": [lon,lat]
            },
            "properties": {
                "topic": "topic.val() ",
                "expected_time": "expRes.val() ",
                "level_of_seriousness": "levOfSer.val() ",
                "request_type": "request_type ",
                "danger_id": "danger.val() ",
                "detail": "detail.val()"
            }
        }
    ]
}
```

Line #6 and line #8 states that this JSON is GeoJSON which is embraced by the square brackets.

6.5 Comparison between OpenLayers 2 and OpenLayers 3 (beta)

OpenLayers 2.0 was initially released eight years ago. It is designed to be flexible for further extending. The latest version of OpenLayers 2 pull off the latest browser advances property like touch events, offline caching, and CSS transitions. It is also widely used by people who create web mapping application. It works well and cover all users' needs but there are some reasons the developers are doing the version 3.

First problem of version 2 is the dichotomy of base layer and overlay. It is quite a challenge for user to know how to set properties for the map for example; resolution and maxExtend. Projection handling between map and each layer and the interplay between projection and properties are also complicated as well. Even expert users might be confused sometimes. In version 3, the developers have fixed the design and API flaws to solve this problem.

The new API for version 3 is designed to be more consistent and convenient than the old version. OpenLayers 2 uses following syntax:

```
1  var map = new OpenLayers.Map({
2      div: "map",
3      layers: [new OpenLayers.Layer.OSM()],
4      center: [-12245144, 5621521],
5      zoom: 4
6  });
```

In OpenLayers3's syntax looks like the following:

```
1    var map = ol.map({
2      renderTo: "map",
3      layers: [ol.layer.osm()],
4      center: [-110, 45],
5      zoom: 4
6  });
```

They actually go further than just a syntax changing, they have created new functionalities, new platforms, and apply with new technologies. 3D is what the developers are targeting. Since WebGL is the standard for 3D in browsers, OpenLayers 3 leverages WebGL by providing map renderer abstraction. It allows

different map renderer implementations. Moreover, getting the benefit of WebGL, the developers are going to implement version 3 to support very large vector layers, client-side re-projection of vector and raster data, 2D tiled and rotated views, and 3D terrain in local projection.

The 2D and 3D worlds should converge with seamless transition, this can be done by integrating virtual globes like Cesium.

OpenLayers 3 is based on the Closure library which is a unique high-quality tool for code verification and optimization. The Closure compiler produces extremely compact, high performance JvaScript through advanced optimizations like variable and property re-naming, and unused code removal. JSDoc-style annotations allow the compiler to do type checking which helps the developers to produce JavaScript code with fewer bugs. The code is easier to extend and maintain.

It is obvious that OpenLayers 3 stable release is going to be large collection of modules with automated tests and continuous integration ensuring that they always work well together. It will bring the huge advantages to the library users for example; the long-live maintenance and especially for large-scale projects. Being able to update one library is so much easier than tracking many different plug-ins form many different sources and having to manually deal with dependency management and integration problem.

Chapter 7

Result

This chapter discusses about the key functionalities provided with screenshots as well as the experimentation of our project.

7.1 Our GIS Web Application

Since we are responsible for a client-side web application development, we put an emphasis on how we can deliver to the user with an easy-to-use user interface as well as provide an essential set of GIS tools, and other related tools, for the user to use. Thus, we mainly focus on explaining the user interface in detail in this section. To see our client-side GIS web application in action please visit the following URL: http://www.i-emergency.com/gis/

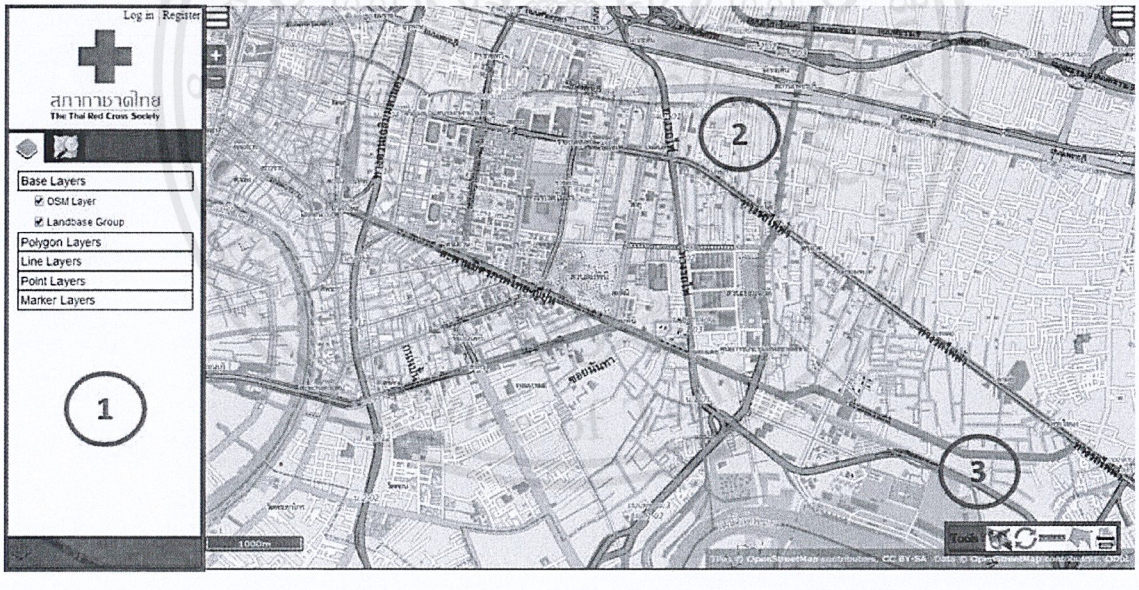

Figure 7.1 Main page of the web application

The Figure 7.1 shows the interface of the web application. As mentioned in Chapter 5 System Design (section 5.2 Graphical User Interface Design), the toggle side menu (1) is on the leftmost, a set of GIS tools (3) is at the bottom-right corner. And, apparently, most of the space of the page is reserved as a viewer for the map (2).

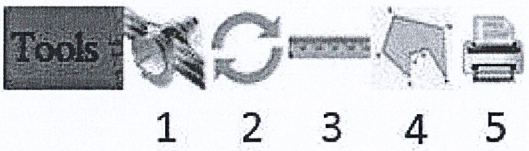

Figure 7.2 GIS tools

There are five GIS tools as shown in the Figure 7.2 which are geolocation, map rotation, distance and area measurement, and map printing respectively. We will demonstrate those tools in detail shortly.

1. Get current location (geolocation)

When the user clicks on 'Get current location' tool (1), the map will be automatically panned to the current location of the user as well as pin a marker on the map as shown in Figure 7.3.

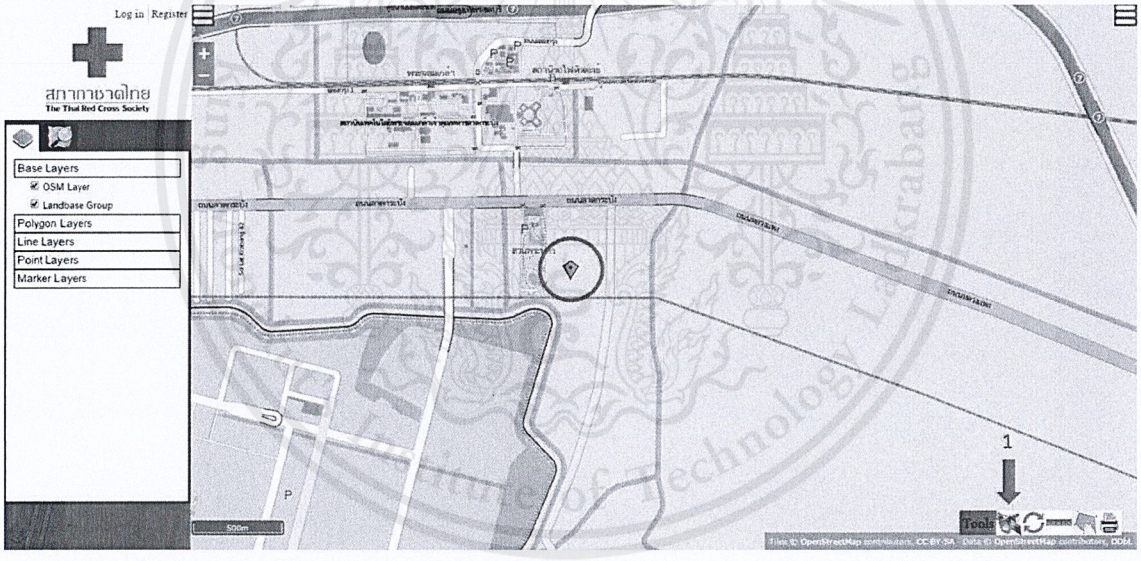

Figure 7.3 Geolocation

2. Rotate map

When the user clicks on map rotation tool (1), there will be a bar (2) for user to rotate the map either in clockwise or counterclockwise direction as shown in Figure 7.4. The user can also press and hold Alt+Shift and then drag the mouse to rotate the map. To return to the default rotating position, the user can click on the reset button (3).

Additionally, if the user is using smart mobile device they can rotate the map using touch gesture as well.

Figure 7.4 Map rotation

3. Measure distance

When the user clicks on 'Measure distance' tool (1), the application lets the user to select the start point and the destination point to measure the distance. The output is at the bottom-left corner of the map next to the side menu (2) as shown in Figure 7.5. The blue line (3) visualize the distance that we want to measure.

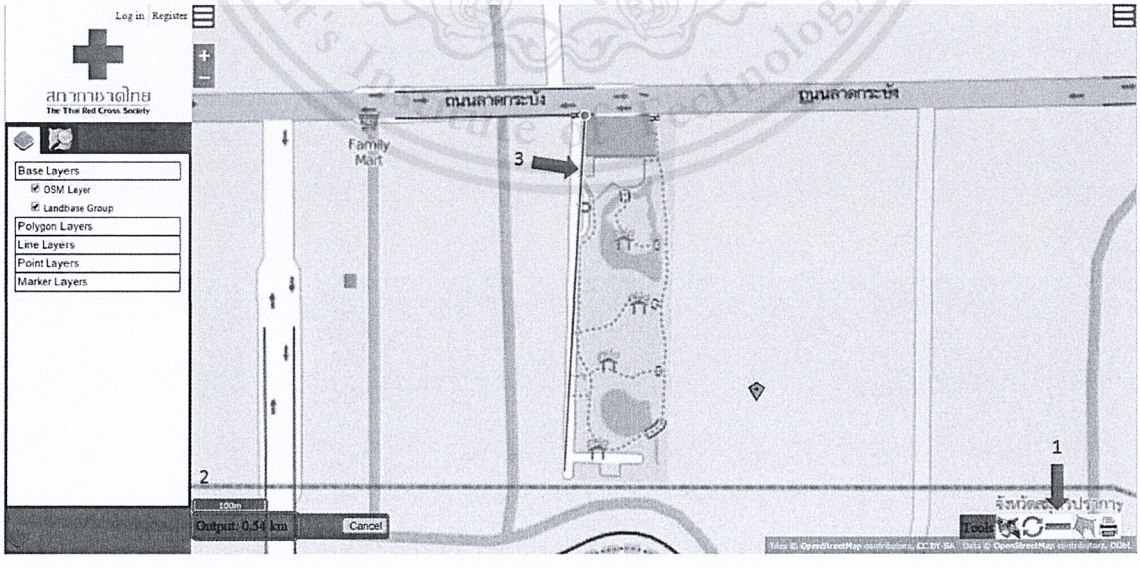

Figure 7.5 Distance measurement

4. Measure area

When the user clicks on 'Measure area' tool (1), the application lets the user to draw a polygon to measure the area which is covered by the blue lines and the area to be measured is shaded in light gray (3). The output is shown at the bottom-left corner of the map next to the side menu (2) as shown in Figure 7.6.

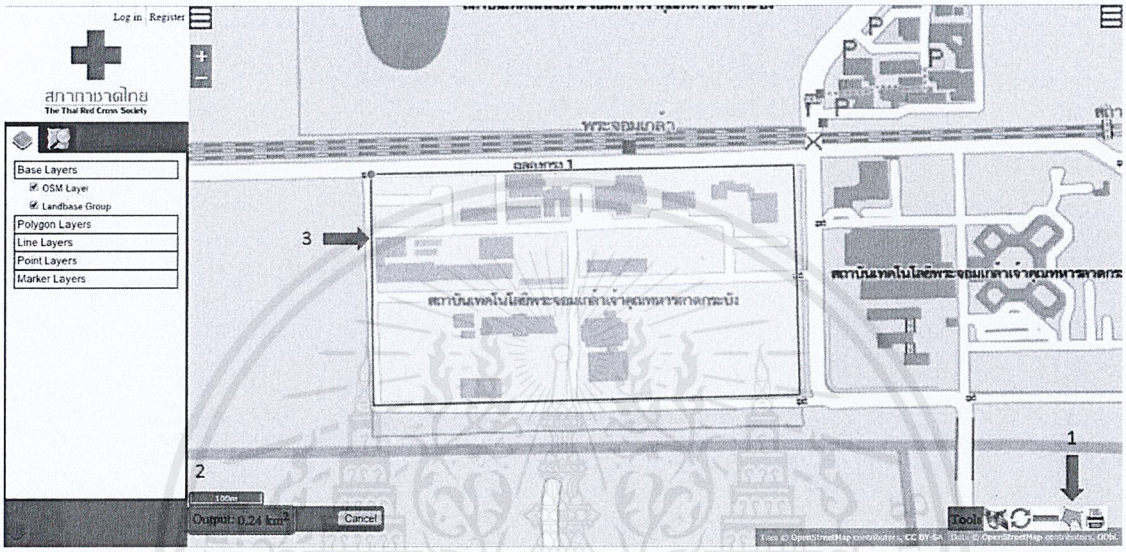

Figure 7.6 Area measurement

5. Print map

When the user clicks on 'Print map' tool (1), the application lets the user to choose whether he want to export the map as '.png' file or print the map immediately via a printer. Both functions are similar in the way that they capture the current view port. Take a look at the followings for the examples.

- Export the map as '.png' file

  When the user clicks on 'export as PNG file' button (see Figure 7.7), the system will automatically download the current will port and save it as '.png' file to the system (2) as shown in Figure 7.8.

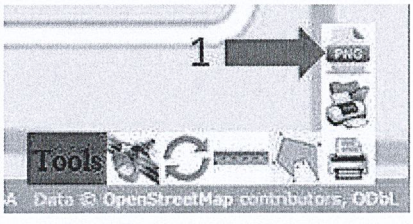

Figure 7.7 Export as PNG file

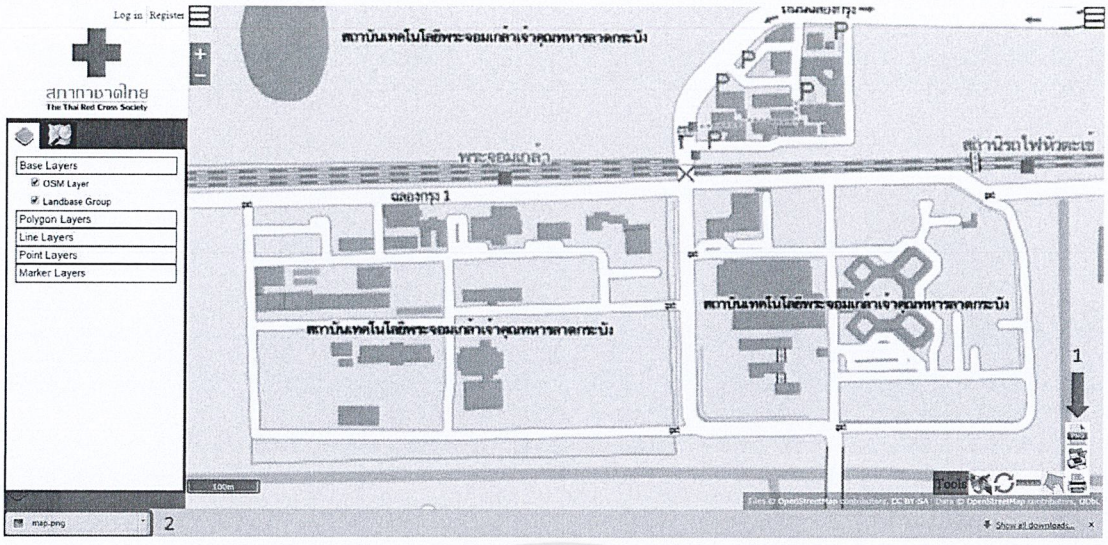

Figure 7.8 Save the file to user's computer

- Print the map instantly via a printer

  When the user clicks on 'Print the map' button (see Figure 7.9), the system will automatically pop up the print window Figure 7.10.

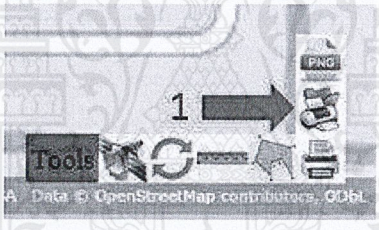

Figure 7.9 Print the map via a printer

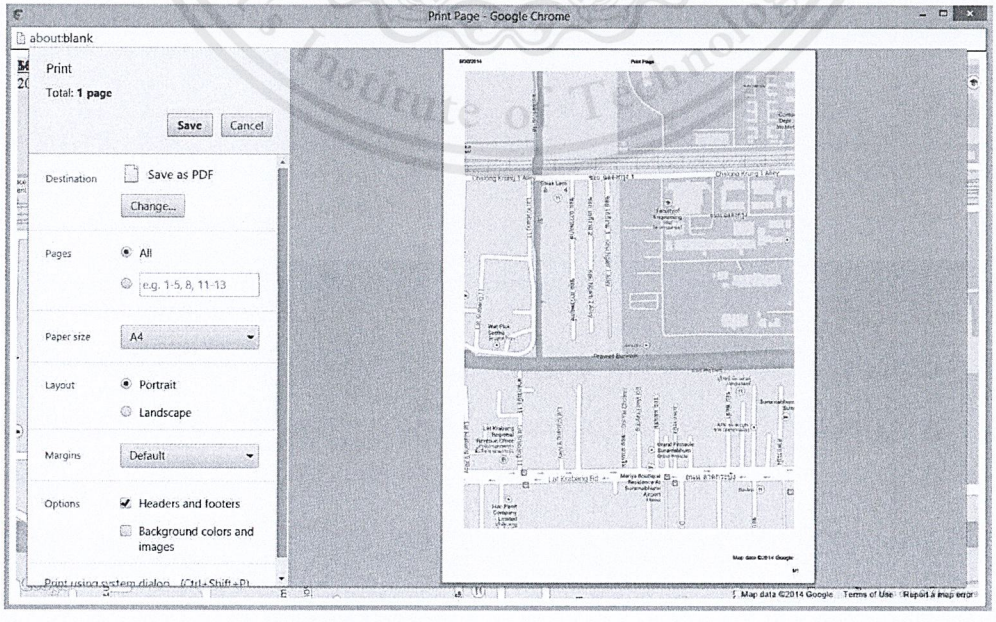

Figure 7.10 Pop up window for print page

For anonymous users who are not registered to the system, they are allowed to use all GIS tools but there are only two GIS services provided for them which are a function to filter the map layers and a function to search location.

- Search function

The user can type a key word to search for location. The tab will expand to the right next to the side menu to show the search result. See the following Figure 7.11 for an example. In this example, we search for the word 'สยาม' and we search only in 'Landmark point' layer (2). It returned 13 search results (1) which are displayed in the form of two-column table, one for the title of that location and another for Geographical Identification number (GID).

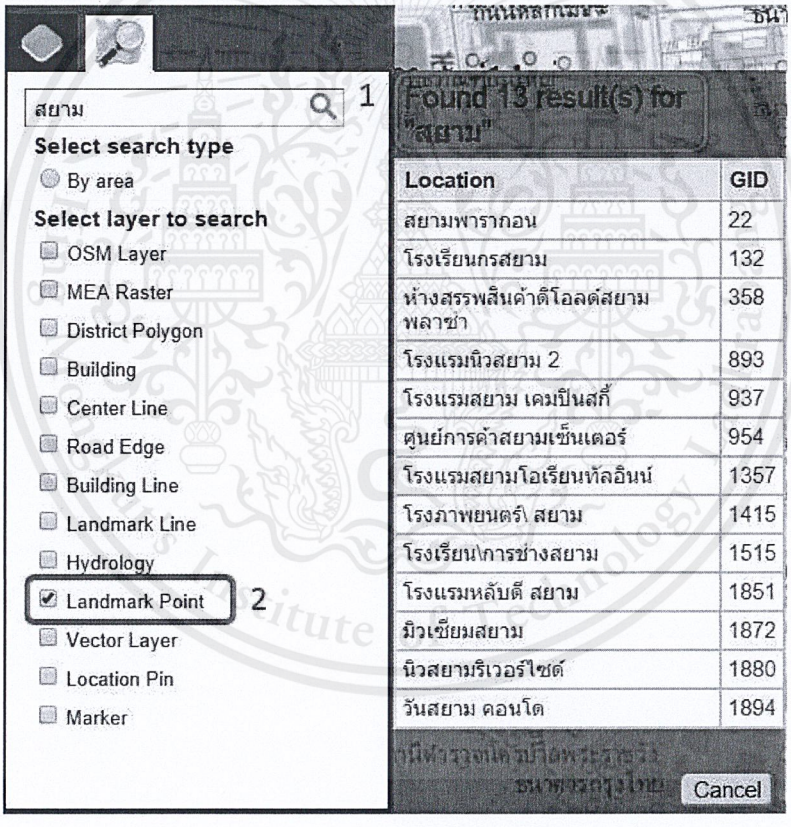

Figure 7.11 Search function

Once the result is displayed, the user can clicks on one of them (1), then the map will be automatically panned to that location (2) as shown in the Figure 7.12 which we clicked on 'นิวสยามริเวอร์ไซด์' as an example.

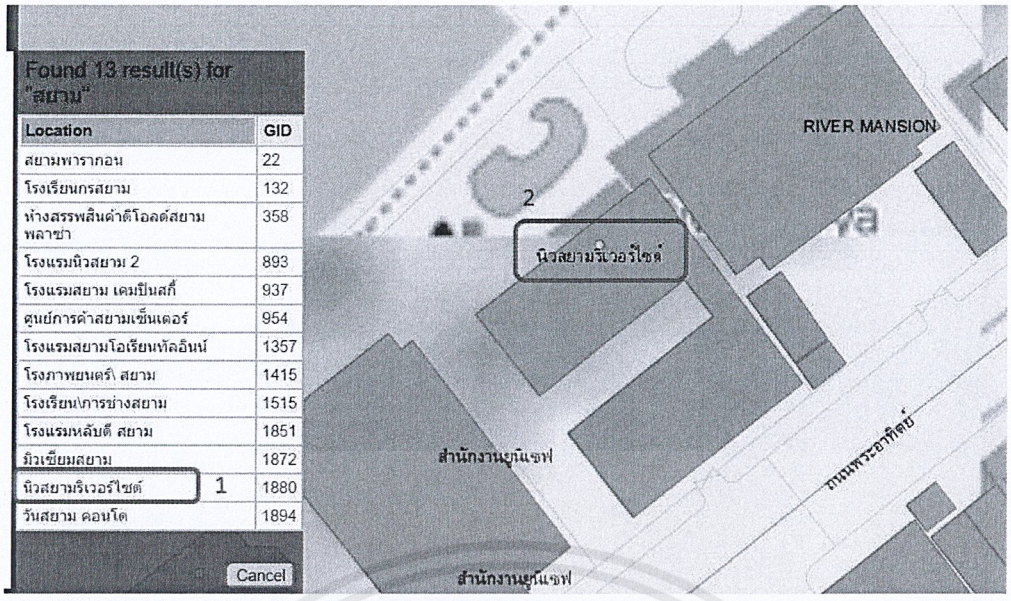

Figure 7.12 Click and pan to the location

- Layer filter function

The user can filter the map to see different layers as they want to. The Figure 7.13 shows the 'Filter map layers' tab and its expanded layers.

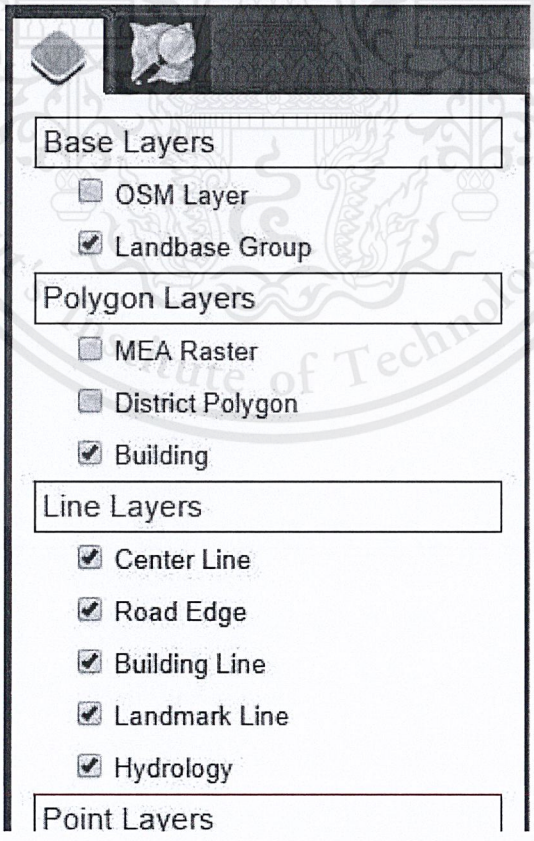

Figure 7.13 Filter map layers tab

The following figures do not have OSM Layer checked so it will not be displayed on the map. One of them have the 'Center Line' layer checked, and another one does not.

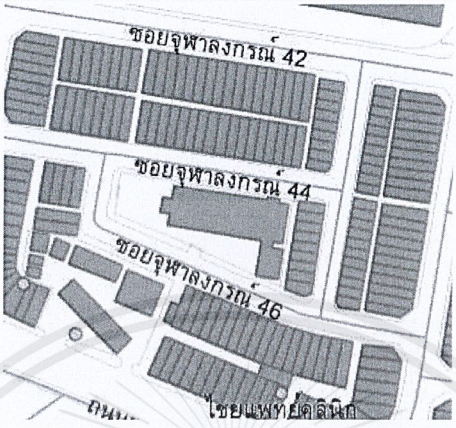

Figure 7.14 'Center Line' is checked

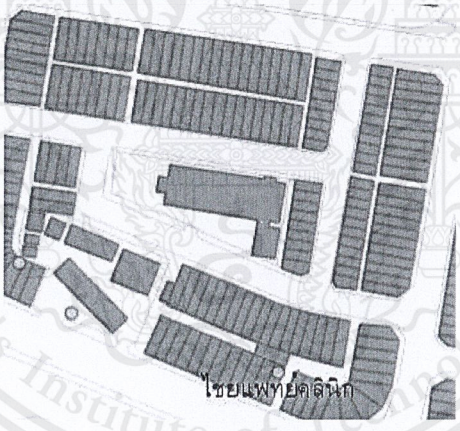

Figure 7.15 'Center Line' is not checked

For registered users, there are additional tabs called 'SERVICES'. This tab allows the user to use functions including 'Make a request' and 'Track request'.

- Make a request

To make a request, the user selects a marker type of their choice the marker selected is shaded in dark blue. Then, click on it and pin it on the map.

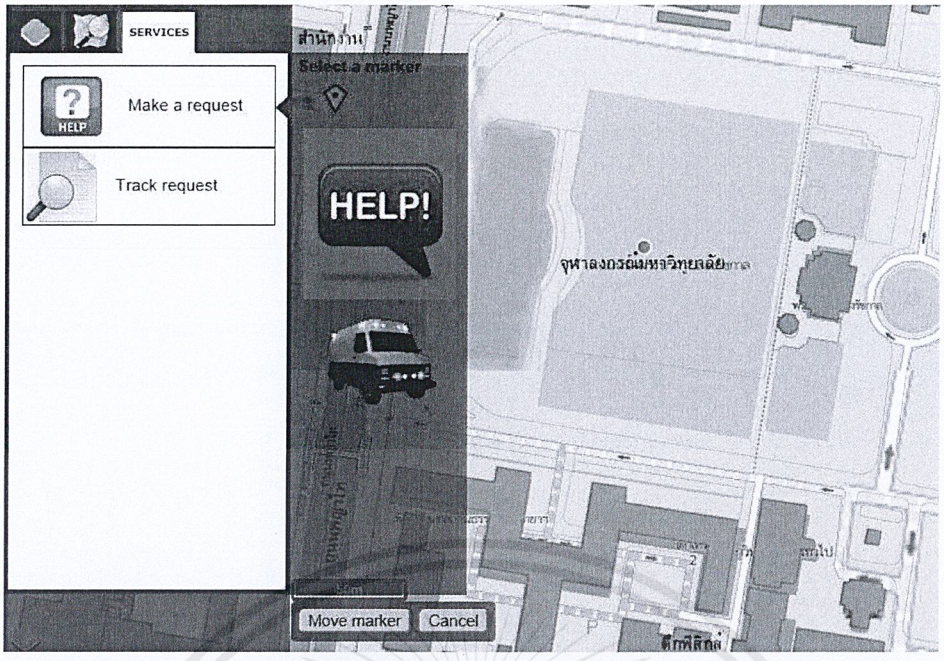

Figure 7.16 Marker selection

Once the maker is pinned, there will be a pop-up window for the user to fill in all necessary details to send to the staff as shown in the Figure 7.17. Additionally, the user can attach an image file as well.

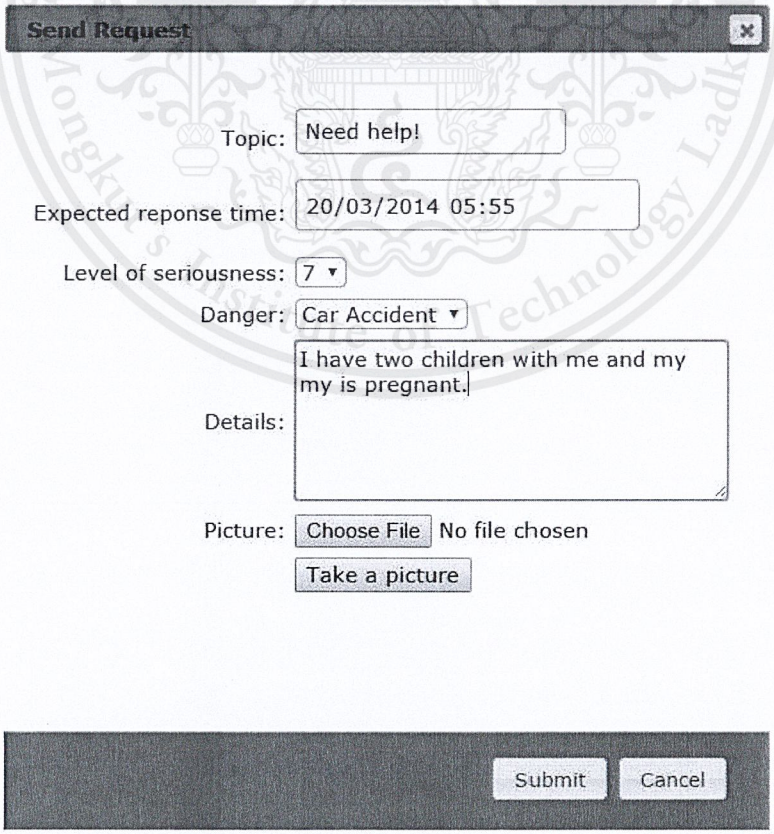

Figure 7.17 Request form

- Track request

To track request, the user selects 'Track request' service, all of the user's submitted request will be displayed in table as shown in the Figure 7.18. The table has five columns which are location, type, date and time of that submitted request, level of seriousness, and status. The status is further divided into 'Pending', 'Requested', and 'Done'.

The user can click on their request topic to pan to that location, for example, if the user clicks on 'fire!!!' (1). The map will be automatically panned to that location. Additionally, if the user clicks on the marker, it will shows the detail of that location, the exact position in this example (2).

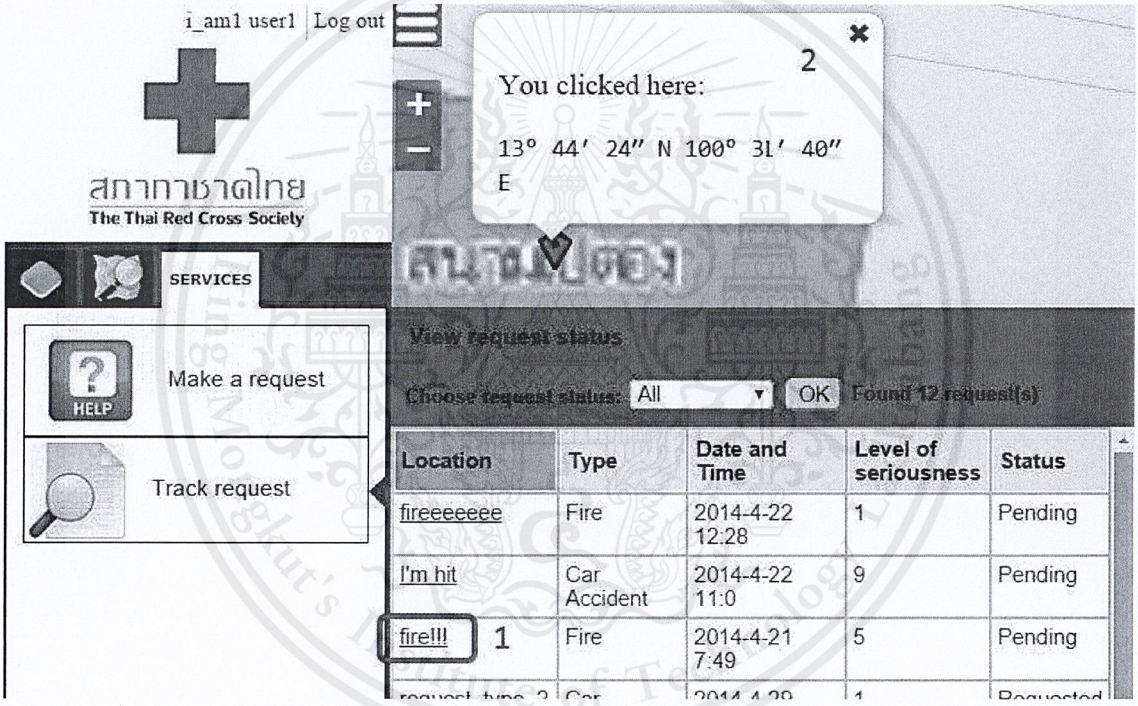

Figure 7.18 Track request scenario

The Figure 7.20 shows the overview of 'Track request' when it returns the result. You can see that as soon as the results are shown, the markers are automatically pinned on the map.

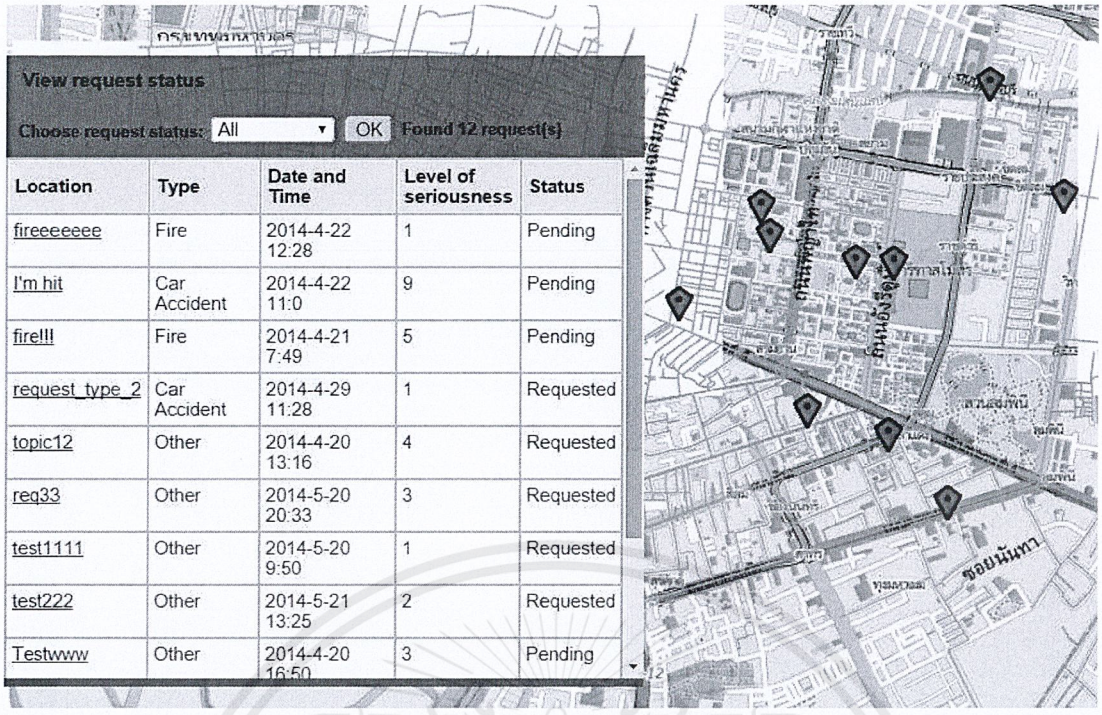

Figure 7.19 Track request overview

For staff users, they have one additional GIS service called 'RESPONSE' tab which is used to respond and handle user requests. All requests that registered users have sent to ask for help can be seen and responded by staff users only.

The mechanism for 'View all request(s)' function is similar to that already mentioned except that there is additional column called 'Action'. It is used to respond to a user request.

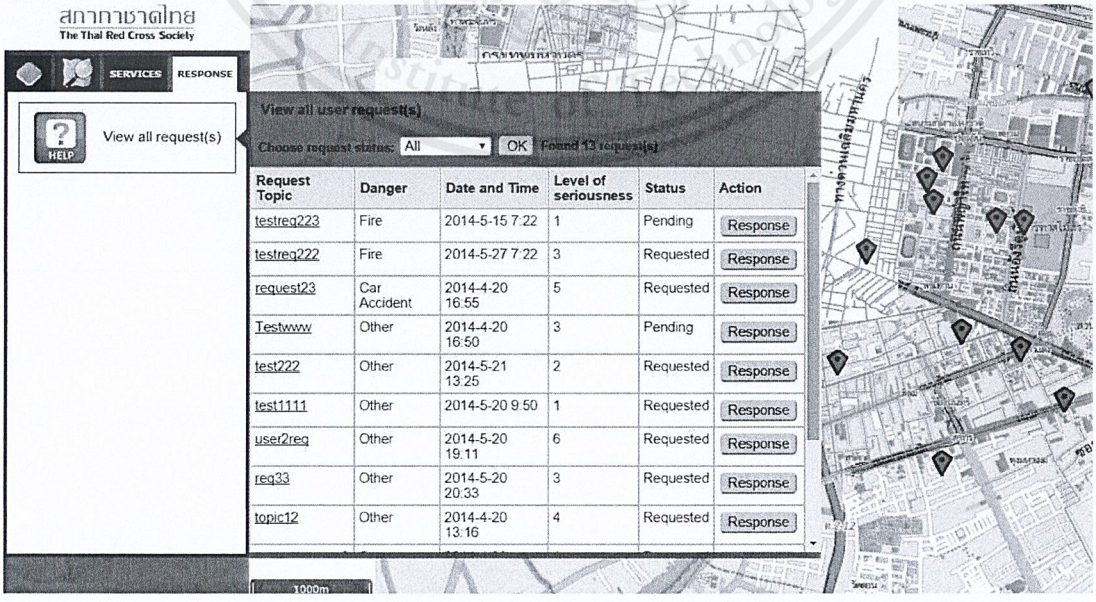

Figure 7.20 View user requests for staff users

When staff user clicks on 'Response' button, the 'Response to a request' window will be popped up. The staff can fill in necessary details to notify the user. Any requests responded by the staff will have their status change from 'Requested' to 'Peding'. To illustrate the help scenario, please see the Figure 7.21.

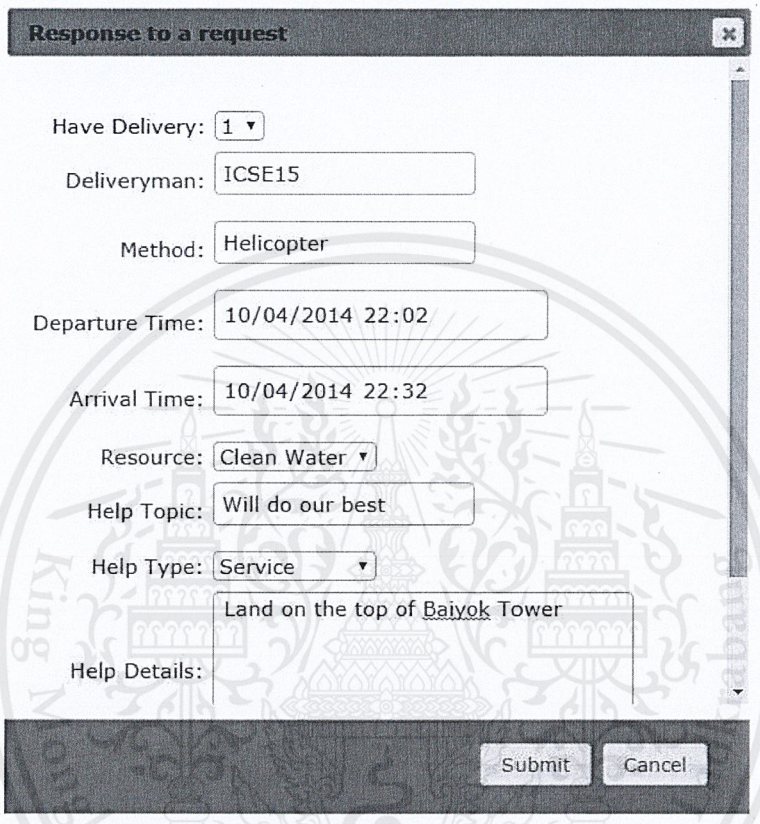

Figure 7.21 Response to a request form

7.2 Experimentation

We have done experiments of geolocation and orientation in order to increase and improve the performance.

First, the early version of geolocation is called via HTML5 which get the user current positions every second manually by the loop function we create using JavaScript. The performance is quite okay, the pin locates at the accurate position but the pin movement is not very smoothly due to the big change of position in every second. So the next version, we take the benefit of OpenLayers 3 which provide the method for geolocation called tracking. This method tracks the user location so the

pin will automatically move very smoothly when there is a change in user position. This way we have a good geolocation function provided on portable device.

Second, HTML 5 provides the Device Orientation API to get the orientation information and the movement of the device via the positional sensors. With the benefit of Device Orientation API, we create the web mapping application which automatically rotate the screen according to the user direction. To experiment the function, we have tested it on many browsers. The best browser to use this function is Firefox Mobile browser which support both the Orientation event and Rotation event. This browser is only provided in play store for android phones. From the experiment if we open this application on other mobiles or other browsers, the application will not perform perfectly for example, the map might not rotate or it rotates but not smoothly.

The Figure 7.22 below compares the compatibility of HTML5 orientation and rotation across different browsers.

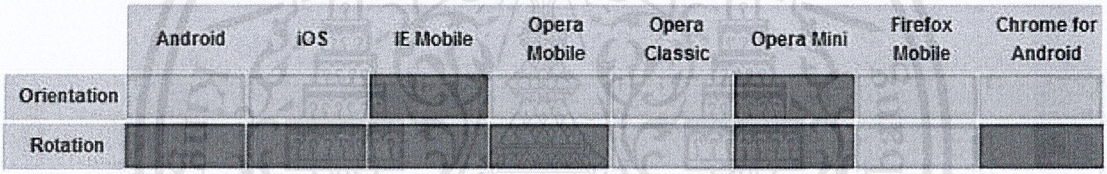

Figure 7.22 HTML5 orientation and rotation compatibility for each browser

Chapter 8

Evaluation and Discussion

In this chapter we talk about the evaluation of the project and also discuss all pros and cons of the software including the precision of the result given by the application and bugs. Especially, discuss in regard to the web application performance as well as distinguish the difference between OpenLayers 2 and OpenLayers 3 (beta). The completion of the project is also mentioned in this chapter.

8.1 Efficiency of OpenLayers

The efficiency of the web application is relatively desirable, the map can respond to users' action very quickly, due to the benefit from using AJAX, OpenLayers and cache provided by the server side. Also, the design is clean and simple. The color is good for long-time starring.

The only missing thing in the scope of the project which makes it is not totally complete is the notification. Although we have an agreement with the server-side group to implement polling technique for notification, we have not managed to finish it in time.

Due to the limitation of OpenLayers 2 at the beginning of the project in first semester, some of the tools and features could not fully perform. Until the second semester, the OpenLayers 3 has been release in beta version. This new library has provides more functions and features to the map; for example, better and various animated transition, better mapping performance.

On the other hand, the new problem is that it is only a beta version which is not stable. The library has bugs and some methods or functions provided in version 2 are not yet in version 3 and we have wasted a lot of time migrate from the old version to the new one. Anyway, we believe it is worth wasting time, the reason is because the project will be able to develop and support more functionalities and features furthermore in the future when the final version is released.

Each GIS tools perform and give a precise result except the distance and area measurements. In OpenLayers 2, there is a method provided to calculate the distance and area according to the shape of the earth which is curved while in OpenLayers 3, there is not yet such a method. As now we have migrated to version 3, the software can only measures both distance and area in planar surface for now. The stable version of OpenLayers 3 will definitely have the method to provide the precise result of measurement. The other GIS tools all work just fine. The geolocation result which represented by the pin display on the map is accurate.

8.2 Completion of Project

This section discusses the completion of our project compared to the requirements specification which are detailed in Chapter 4 Requirements and Analysis (section 4.2.1 Functional requirements). Most of the requirements were satisfied except 'R10' which is about developing a real-time notification to notify the user as soon as their requests are responded by staff.

As we have mentioned in section 8.1, we spent much of our time migrating from OpenLayers 2 to OpenLayers 3 (beta). This is due to incomplete document and the lack of example on the Internet. We experienced a hard time coding without proper examples and, thus, causing some bugs in the application which is a time-consuming tasks. As a result, we keep fixing and improving our system to ensure that it can run OpenLayers 3 (beta) smooth as possible, at least with its current release. So we cannot finish the notification function in time, however, we have studied some topics on how we can communicate with the server-side service using polling technique.

Chapter 9

Conclusions

9.1 Project Summary

We have applied GIS technology with web application to provide emergency service to help relieving people from an unexpected accident or even natural disaster occurred across a particular area. The project is responsible for development of a client-side GIS web application which runs on a web browser. It is also called web-based GIS.

We have developed a client-side GIS web application which contains several components including a map viewer, a set of GIS tools, a search facility, a data collection function from the user, a monitoring function for tracking progress of a submitted request. Those components are briefly discussed in the following paragraphs.

Firstly, the map viewer which is developed to provide the user with precise and accurate spatial data using OpenLayers. Map layers are provided by the server-side service using spatial database from Metropolitan Electricity Authority (MEA). Secondly, we have developed a set of GIS tools for the user to interact with the map such as zooming, panning the map, getting the current location, rotating the map, measuring distance and area, printing the map.

Thirdly, we have provided several GIS-related services including a search facility to search location by layer or boundary, a function to pin a marker on the map, a data collection mechanism to store user request information and send it to the server which is applied to a register function, a submit request function, etc. Finally, we have developed services to send a request, track request, and respond to user request which the latter is reserved for staff user only.

Those components are developed using several web programming languages such as HTML5, CSS3, JavaScript, as well as some of their libraries including OpenLayers 2 and OpenLayers 3 (beta), jQuery, AJAX. Those are used to provide the user with an intuitive user interface and which is, absolutely, easy to use. Furthermore, they are used to define a universal standard for web application and to

build an exceptional interface which can be used either by using the traditional mouse or touch gesture from a mobile device. Additionally, OpenLayers, a JavaScript library, should be mentioned as well since it plays an important role in GIS web development, especially, when we have to deal with the map.

As a result, we are confident that combining the technology of GIS with web application development will offer a great user experience and provide a reliable service supporting multiple platforms for those people who are suffering from unexpected accidents or natural disasters.

9.2 Lessons Learned, Problems and Obstacles

We have learned how to manage time and how to work in group. If we could go back in time, we would have started the project earlier so we could have more time doing and completing the project. Also, we would communicate to each other more about each responsibility in order to integrating parts of the project together. Being responsible and encouraged in teamwork are also important things in order to complete this project.

The problem we have found is the limit of OpenLayers 2 at the beginning of the project in first semester. Some of the tools and features could not fully perform. Until the second semester, the OpenLayers 3 has been release in beta version. This new library has provides more functions and features to the map for example the animation.

But the new problem is that it is only a beta version. The new version does not provide good documents and not enough examples to study. It is also not stable. The library has bugs and some methods or functions provided in version 2 are not yet in version 3 and we have spent a lot of time migrate from the old version to the new one. Anyway, we believe it is worth spending time, the reason is because the project will be able to develop and support more functionalities and features furthermore in the future when the final version is released.

Since we have spent much of the time changing from OpenLayers 2 to OpenLayers 3 (beta), as we mentioned there are differences in syntax, methods, and attributes as well as lacking of well-defined documents and some examples. As a result, we could not finish some functions in time. Especially, a notification center which is used to notify users when their requests are responded in real-time manner.

9.3 Future work

The web application is deliberately designed for future expansion. There are spaces available in the service tab and, in case the account type is staff, response tab to be added with more functionalities. So the web application should be added for more features and functions accordingly. Also, the web application should support all devices for example desktop version, mobile version and tablet version. The following lists are possible works that will probably be added to the system for the future work:

- In order to support a diverse range of devices, responsive design will be introduced. As a result, the user can get the same experience whatever the devices they are using.
- Add real-time notification when the user request is sent and responded. This enables a rapid help from the staff side and the user will be notified in real-time manner.
- Add the search capability; for example, search location by area manually defined by the user.
- Automatically calculate the time needed to deliver help compared to the user's and staff's current locations.
- Improve performance and efficiency for large scale use especially reducing the time the user use to fill in the form. It should be faster and deliver more automated artificial intelligence to the user.
- Add more tools such as a service from geo-communicator group to support users' call from different location simultaneously.

Bibliography

[1] Hazzard, E. 2011. **OpenLayers 2.10 Beginner's Guild.** Birmingham, United Kingdom: Packt Publishing.

[2] Perez, S.A. 2012. **OpenLayers Cookbook.** Birmingham, United Kingdom: Packt Publishing.

[3] Robbins, N.J. 2012. **Learing Web Design 4th: A beginner's Guide to HTML, CSS, JavaScript, and Web Graphics.** Sebastopol, United States of America: O'Reilly Media.

[4] Lawson, B. and Sharp, R. 2012. **Introducing HTML5 2nd Edition.** United States of America: Pearson Education.

[5] Chaffer J. and Swedberg K. 2011. **Learning jQuery 3rd Edition.** Birmingham, United Kingdom: Packt Publishing Ltd.

[6] Resig, J. and Bibeault, B. 2013. **Secrets of the JavaScript Ninja.** United States of America: Manning Publications Co.

[7] Crockford, D. 2008. **JavaScript: The Good Parts.** United States of America: O'Reilly Media, Inc.

[8] Freeman, E.T. and Robson, E. 2014. **Head First JavaScript Programming.** United States of America: O'Reilly Media, Inc.