DISTANCE BASED SEARCHING BY AREA FOR REAL ESTATE IN BANGKOK

CHATCHOL ISARANGKUL

THAMONWAN KAJORNSAKSOPON

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR

THE DEGREE OF BECHELOR OF SCIENCE

INTERNATIONAL PROGRAMS, FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

Special Project Title	Distance Based Searching by Area for Real Estate in Bangkok	
Name	Mr.Chatchol Isarangkul	ID: 48050887
	Ms.Thamonwan Kajornsaksopon	ID: 48050890
Faculty	Science	
Program	Computer Science (International Program)	
Special Project Advisor	Mr.Suntana Oudomying	

Abstract

We propose an application for ranking results of users' interest in terms of distance to the intersection of their selection. This application helps users searching and adding properties to database based on Google maps using MySQL to store information. In our implementation, the system shows intersection (blue markers) and distance to real estate (red markers). This project shows how real-estate buying / selling websites are supposed to be.

Acknowledgement

We would like to express my gratitude to all those who gave us the possibility to complete this project.

The authors wish to express our deep gratitude to Mr. Suntana Oudomying our advisor, for this valuable guidance, attention, and encouragement throughout this research.

We are greatly appreciated all the professors who have given invaluable knowledge while we were studying in the department of computer, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang.

We are also would like to give the special thanks to all of our friends at the department who had encouraged us.

Lastly, the authors would like to express our deepest appreciation to our dearest fathers, mothers, brothers and sisters for love, care and encouragement. They are the most important in our lives forever.

Mr. Chatchol Isarangkul

Ms. Thamonwan Kajornsaksopon

Table of Contents

Chapter 1

Introduction

1.1 Rationale

Evolution in geographic information technology has improved dramatically in recent years – in particular digital map. Real estate websites in Thailand include static map images to better communicate with their users about the location of the properties listed on the site. Nowadays, embedding a map into websites is as easy as using Google maps API. With Google Map, users may zoom in / out to understand the real time map. In general, Google Map improves the way to convey location information.

Still real estate websites are accustom to traditional way for users query. All websites we found list the properties queried in term of district. At best, they list properties in term of particular area where new projects are dense, for example On-Nut. In practice, a person wants to be able to specify the area of his interest to the degree of intersection. (Hence, On-Nut mentioned above is the only exception.) For example, he is interested in properties within 10 kilometers from Nida intersection regardless of the district of the properties.

With two motivations in mind, we propose a web application embedded with Google map for users to easily retrieve properties located within the radius from the area the user specified.

1.2 Objective

Our web application stores built-in latitude, longitude of well-known intersections in Bangkok. Properties listed for sales are also stored with latitude and longitude of the locations.

With the information, our application ranks the result set differently from buyer selected intersection to another.

1.3 Scope of study

Since Google map does not recognize location addresses in Thailand, sellers have to pin the location of his property manually. Therefore, the application does not guarantee the accuracy of the pinning. Secondly, our project focuses on the interface of Google map, selection criteria are subjected to only the type of the property, namely – house, townhouse, condominium.

1.4 Research Methodology

To create our system, we have studied and research some relate Google map API, JavaScript, PHP, MySQL and basic of map. All related we will provide in chapter 2. Chapter 3, it shows how our system works from the beginning and how it shows the result. In chapter 4, we evaluate the system to show how well it works and show how to use it. The last part, chapter 5, we discuss about system's conclusion and recommendation. In the appendix

Chapter 2

Related Literature

In this chapter we give an overview of technologies and tools involved with application used in our project. These are Google maps API, PHP, MySQL, NetBeans IDE and Apache Web Server.

2.1 Google Maps API

The Google map is a free web mapping application and technology provided by Google that powers many map-based services including the Google Maps website, Google Ride Finder, Google Transit and embedded maps on third-party websites via the Google Maps API. It offers street maps, a route planner for traveling by foot, bicycle, car, or public transport and an urban business locator for numerous countries around the world. It also can help with finding businesses.

In late 2006, Yahoo began a campaign to upgrade their maps, to compete better with Google Local and other online map companies. Several of the maps used in a survey were similar to Google maps. Google Maps actively promotes the commercial use of their API. One of its earliest adopters at large scale are real estate mashup sites. Google's case study is about Nestoria, a property search engine in the UK and Spain. By using the Google Maps API it is possible to embed the full Google Maps on an external web site. Developers are required to request an API Key which is bound to the web site and directory entered when creating the key. Creating your own map interface involves adding the Google JavaScript code to your page, and then using Javascript functions to add points to the map. When the API first launched, it lacked the ability to geocode addresses, requiring you to manually add points in (latitude, longitude) format.

This has since been rectified. At the same time as the release of the Google Maps API, Yahoo! released their own Maps API. Both were released to coincide with the O'Reilly Web 2.0 Conference. Yahoo! Maps lacks international support, but included a geocoder in the first release.

As of October 2006, Google Gadgets' Google maps implementation is much easier to use with just the need of one line of script. The drawback is that it is not as customizable as the full API.

Google created the Google Maps API to facilitate developers integrating Google Maps into their web sites with their own data points. It is a free service which currently does not contain ads, but Google states in their terms of use that they reserve the right to display ads in the future.

Figure 2.1 shows Nestoria page.

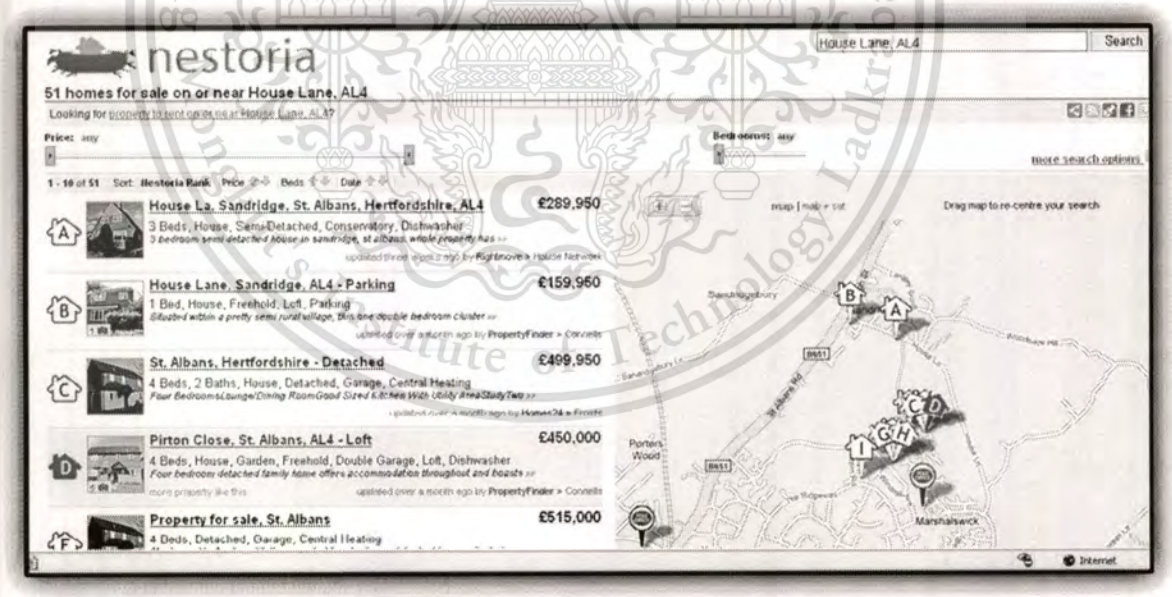


Figure 2.1 Nestoria home page

2.1.1 Google map API Features

1. Help users to find location. It offers street maps, a route planner for traveling by foot, bicycle, car, or public transport and an urban business locator for numerous countries around the world. Google map allows users to input location name then point marker for their location and show detail of location. Figure 2.2 shows the example to find location on Google maps.

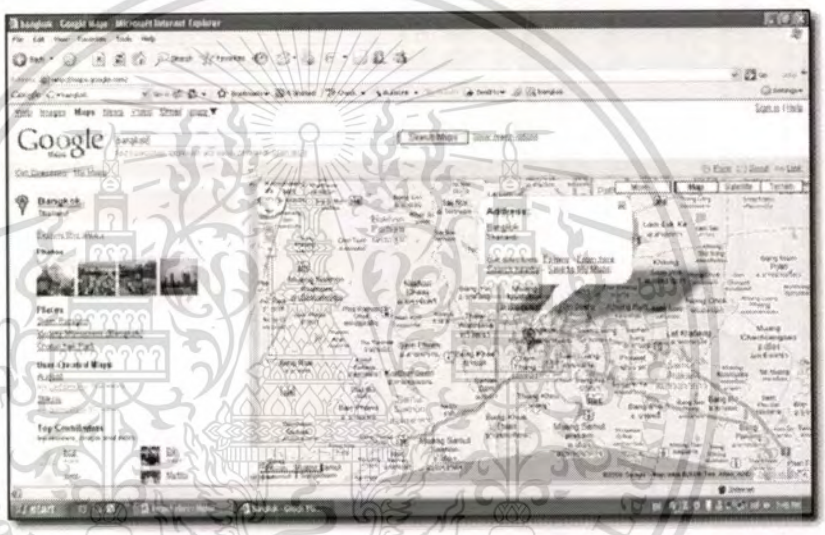


Figure 2.2 Find location

2. Local business center. Use the Local Business Center to create your free listing. When potential customers search Maps for local information, they will find your business your address, hours of operation, even photos of your storefront or products. It is easy, free, and you do not need a website of your own. Figure 2.3 shows added local business.

Figure 2.3 Local Business Center Process

2.1.2 Google maps API function

2.1.2.1 Map events

JavaScript within the browser is *event driven*, meaning that JavaScript responds to interactions by generating events, and expects a program to *listen* to interesting events. For example, within browsers, user mouse and keyboard interactions create events that propagate within the DOM. Programs interested in certain events will register JavaScript **event listeners** for those events and execute code when those events are received.

2.1.2.1.1 Event Listener

Events in the Google Maps API are handled by using utility functions within the GEvent namespace to register event listeners. Each Maps API object exports a number of named events. For example, the GMap2 object exports click, dblclick, and move events, and a host of others as well. Each event happens within a given context, and can pass arguments that identify that context. For example, the mousemove event fires when the user moves the mouse within a map object, and pass the GLatLng of the geographic location in which the mouse is located.

To register for notification of these events, use the static method GEvent.addListener(). That method takes an object, an event to listen for, and a function to call when the specified event occurs.

2.1.2.2 Map Control

The maps on http://maps.google.com contain UI elements for allowing user interaction through the map. These elements are known as *controls* and you can include variations of these controls in your Google Maps API application. You can also build your own custom controls by subclassing the GControl class. You add controls to the map with the GMap2 method addControl().

The Maps API comes with a handful of built-in controls you can use in your maps:

- GLargeMapControl3D - a large pan/zoom control as now used on Google Maps. Appears in the top left corner of the map by default.
- GLargeMapControl - a simpler large pan/zoom control. Appears in the top left corner of the map by default.
- GSmallMapControl - a smaller pan/zoom control. Appears in the top left corner of the map by default.
- GSmallZoomControl3D - a small zoom control (with no panning controls) as now used on Google Maps.
- GSmallZoomControl - a small zoom control (no panning controls) used in the small map blowup windows used to display driving directions steps on Google Maps.
- GScaleControl - a map scale
- GMapTypeControl - buttons that let the user toggle between map types (such as Map and Satellite)

- GHierarchicalMapTypeControl - a selection of nested buttons and menu items for placing many map type selectors.
- GOverviewMapControl - a collapsible overview map in the corner of the screen
- GNavLabelControl - a dynamic label indicating the "address" of the current viewport, appropriate to zoom level.

All of these controls are based on the GControl object.

2.1.2.3 Map Overlays

Overlays are objects on the map that are tied to latitude/longitude coordinates, so they move when you drag or zoom the map. Overlays reflect objects that you "add" to the map to designate points, lines, or areas.

The Maps API has several types of overlays:

- Points on the map are displayed using **markers**, and often display a custom icon. Markers are objects of type GMarker and may make use of the GIcon type.
- Lines on the map are displayed using **polylines** (representing a collection of points). Lines are objects of type GPolyline.
- Areas on the map are displayed either as **polygons** if they are areas of an arbitrary shape or as **ground overlays** if they are rectangular. Polygons are similar to polylines in that they consist of a collection of points with a closed loop and may take any shape. Ground overlays are often used for areas that map either directly or indirectly to tiles on the map.

- The map itself is displayed using a **tile overlay**. You can modify this with your own set of tiles by using a GTileLayerOverlay or even by creating your map type using a GMapType.
- The **info window** is also a special kind of overlay. Note, however, that the info window is added to the map automatically, and that there can only be one object of type GInfoWindow attached to a map.

Each overlay implements the GOverlay interface. Overlays can be added to a map using the GMap2.addOverlay() method and removed using the GMap2.removeOverlay() method. (Note that the info window is added by default to the map.)

2.1.2.4 Map Services

The Google Maps API is regularly extended, adding new functionality and features that are often released on maps.google.com first. This section covers these services. Basically, this section is where we put all of those neat things that we could not put anywhere else.

Note that, our project did not use this section.

2.2 PHP (PHP Hypertext Preprocessor)

PHP Language is open source programming originally designed for producing dynamic web pages. PHP is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. It generally runs on a web server, taking PHP code as its input and creating web pages as output. It can be deployed on most web servers and on almost every operating system and platform free of charge. PHP is installed on more than 20 million websites and 1 million web servers.

PHP originally stood for Personal Home Page. It began in 1994 by Rasmus Lerdorf American programmer. He initially created this Personal Home Page using C programming Language and Perl to maintain his personal homepage. The tools were used to communicate with

database called "Form Interpreter (FI)" used to perform tasks such as displaying his résumé and recording how much traffic his page was receiving. He combined these binaries with his Form Interpreter to create PHP/FI which the beginning of PHP. PHP/FI included a larger implementation for the C programming language and could communicate with databases, enabling the building of simple, dynamic web applications.

Lerdorf released PHP publicly on June 8, 1995 to accelerate bug location and improve the code. This release was named PHP version 2 and already had the basic functionality that PHP has today. This included Perl-like variables, form handling, and the ability to embed HTML. The syntax was similar to Perl but was more limited, simpler, and less consistent.

After that PHP is very popular, users was demand to develop efficiency for PHP/FI. Zeev Suraski and Andi Gutmans, two Israeli developers at the Technion IIT come to help Rasmus Lerdorf to rewrote the parser in 1997 and formed the base of PHP 3. Stig Bakken supports PHP for communicate with Oracle, Shane Caraveo on Window 9x/NT, and Jim Winstead changed the language's name to *PHP: Hypertext Preprocessor*. The development team officially released PHP/FI 2 in November 1997 after months of beta testing. Afterwards, public testing of PHP 3 began, and the official launch came in June 1998.

On May 22, 2000, PHP 4, powered by the Zend Engine 1.0, was released. On July 13, 2004, PHP 5 was released, powered by the new Zend Engine II. PHP5 included new features such as improved support for object-oriented programming, the PHP Data Objects extension and numerous performance enhancements.

Nowadays developer used PHP to develop web site more than 5,100,000 sites all over the world. PHP can support in many database as follows:

Adabas D	Ingres	Oracle (OCI7 and OCI8)
Dbase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

Among the above databases, MySQL is the most popular database on PHP because MySQL is a freeware software and it is supported on many operating systems (OS). We will explain in detail about MySQL in the next topic.

2.3 MySQL Server 5.1

MySQL is a relational database management system (RDBMS) which has more than 11 million installations. The program runs as a server providing multi-user access to a number of databases. It supports sql command for store data and used together with another tools or program such as PHP, ASP, .NET, JSP , as well as can support Application Program such as JAVA, C#, etc.

MySQL developed by 2 Swede and Finn name David Axmark, Allan Larsson and Michael "Monty" Widenius. It was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now a subsidiary of Sun Microsystems, which holds the copyright to most of the codebase. The project's source code is available under terms of the GNU General Public License, as well as under a variety of proprietary agreements.

MySQL provides graphical tools for managing MySQL database namely MySQL Administrator and MySQLQuery browser.Figure 2.4 shows MySQL query browser interface.

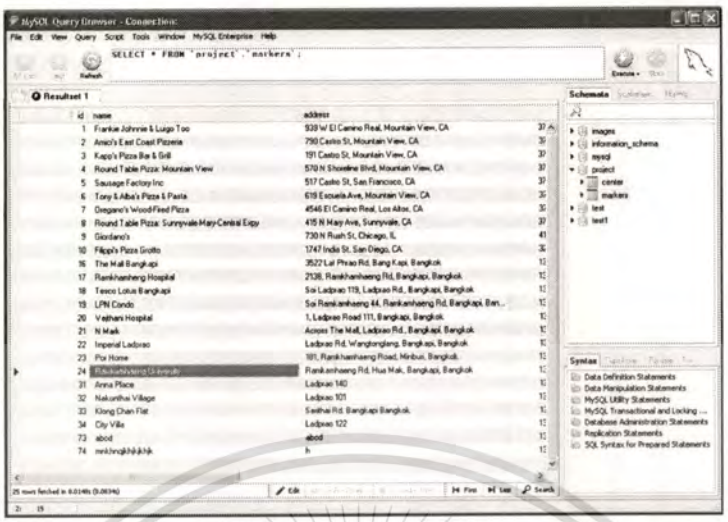


Figure 2.4 MySQL query browser

2.4 NetBeans IDE

NetBeans is an integrated development environment (IDE) for developing applications. The tool allows programmer to develop applications using java, JavaScript, PHP, Python, Ruby, Groovy, C, and C++.

NetBeans began in 1997 a student project under the guidance of the Faculty of Mathematics and Physics at Charles University in Prague. Roman Stanek later formed a company around the project and produced commercial versions of the NetBeans IDE until it was bought by Sun Microsystems in 1999. Sun open-sourced the NetBeans IDE in June of the following year. The NetBeans community has since continued to grow, thanks to individuals and companies using and contributing to the project.

Nowadays developer can use another programming language(C/C++, Ruby, UML, SOA, Web Application, Java EE, Mobility (Java ME), Java FX, Java Script, etc.) to development application by installing "Add-On" from netbeans web site or update center. NetBeans version 6.0 and later releases equipped with a number of Add-On programs.

The current version is NetBeans IDE 6.5, which was released in November 2008. NetBeans has won several awards for the Developer.com products of the year awards 2009: Development Tool, Development Utilities,

Wireless / Mobile, Java Tool, and Open Source. Figure 2.5 shows Netbeans first page.

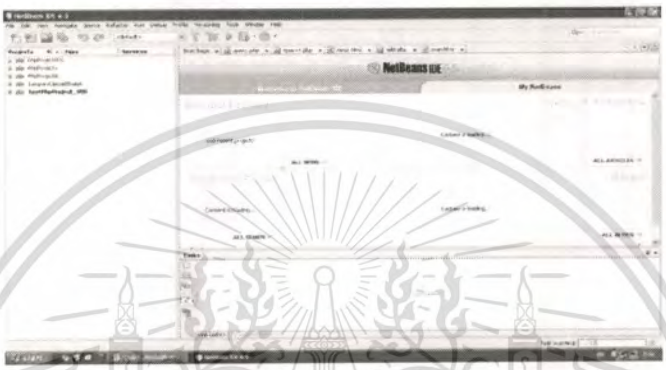


Figure 2.5 Netbeans IDE

2.5 Apache Web Server

Apache Web Server (Apache HTTP Server) is a web server notable for playing a key role in the initial growth of the World Wide Web. Many web applications are designed expecting the environment and features that Apache provides. Apache is the web server component of the popular LAMP web server application stack, alongside MySQL, and the PHP/Perl/Python (and now also Ruby) programming languages. Apache is used for many other tasks where content needs to be made available in a secure and reliable way. One example is sharing files from a personal computer over the Internet. Programmers developing web applications often use a locally installed version of Apache in order to preview and test code as it is being developed.

The first version of the Apache web server was created by Robert McCool, who was heavily involved with the National Center for Supercomputing Applications web server, known simply as NCSA HTTPd. When McCool left NCSA in mid-1994, the development of httpd stalled, leaving a variety of patches for improvements circulating through e-mails. These patches-

were provided by a number of other developers besides McCool: Brian Behlendorf, Roy Fielding, Rob Hartill, David Robinson, Cliff Skolnick, Randy Terbush, Robert S. Thau, Andrew Wilson, Eric Hagberg, Frank Peters and Nicolas Pioch, and they thus helped to form the original "Apache Group". There have been two explanations of the project's name.

According to the Apache Foundation, the name was chosen out of respect for the Native American tribe of Apache (Indé), well-known for their endurance and their skills in warfare. However, the original FAQ on the Apache Server project's website, from 1996 to 2001, claimed that "The result after combining [the NCSA httpd patches] was *a patchy server.* The first explanation was supported at an Apache Conference and in an interview in 2000 by Brian Behlendorf, who said that the name connoted "Take no prisoners. Be kind of aggressive and kick some ass". Behlendorf then contradicted this in a 2007 interview, stating that "The Apache server isn't named in honor of Geronimo's tribe" but that so many revisions were sent in that "the group called it 'a patchy Web server'". Both explanations are probably appropriate though the pun explanation has fallen into disfavor. Version 2 of the Apache server was a substantial re-write of much of the Apache 1.x code, with a strong focus on further modularization and the development of a portability layer, the Apache Portable Runtime. The Apache 2.x core has several major enhancements over Apache 1.x. These include UNIX threading, better support for non-Unix platforms (such as Microsoft Windows), a new Apache API, and IPv6 support. The first alpha release of Apache 2 was in March 2000, with the first general availability release on April 6, 2002. Version 2.2 introduced a more flexible authorization API. It also features improved cache modules and proxy modules.

Chapter 3

Distance Based Searching by Area for Real Estate in Bangkok

This chapter will cover designs we have for this project. These designs are system analysis, application design, and database design and user interface design. In our project, we categorized process of developing into 2 parts: System analysis process and system design process. System analysis explains the idea behind this project while system design process explains technical designs. The UIs will be shown in Chapter 4.

3.1 System Analysis

3.1.1 Problem

Real estate websites in present use maps only for the real estate location, while in practice, buyers would prefer to see list of properties listed in the area of buyer's interest in one single map.

3.1.2 Solution

Our application ranks the resultset based on distance calculation function using Haversine formula and present the result on a Google map. The application functions for its users include:

For buyers

- Selecting center location for this project.
- Choosing radius for coverage area.
- Filtering by type of real estate (House, Condo and Space).

For sellers

- Adding their real estate information to database (Name, address, type, description, images).

3.1.3 The Haversine Formula

The Haversine Formula is used generally for computing great-circle distances between two pairs of coordinates on a sphere.

Figure 3.1 shows the example of SQL statement that will find the closest 20 locations that are within a radius of 25 miles from the 37, -122 coordinate. It calculates the distance based on the latitude/longitude of that selected intersection and the target latitude/longitude, and then asks for only rows where the distance value is less than 25, orders the whole query by distance, and limits it to 20 results.

```
SELECT id, ( 6371 * acos( cos( radians(37) ) * cos( radians( lat ) ) * cos( radians( lng ) - radians(-122) ) + sin( radians(37)
) * sin( radians( lat ) ) ) ) AS distance

FROM markers

HAVING distance < 25

 ORDER BY distance

LIMIT 0 , 20;
```

Figure 3.1 Example of Haversine Formula

3.2 System Requirement

3.2.1 System structure

This process will show how system works by using 2 diagrams: dataflow diagram and activity diagram.

3.2.1.1 Dataflow diagram

This section explains the overall activities of our application. There are 3 groups of persons involved to it – sellers, buyers, and an administrator. Sellers can add their properties information which will be stored in markers table. The administrator adds intersections information which includes their coordination for later calculation. Buyers can search locations of their interest by selecting an intersection. The application selects properties from marker table that are within the radius chosen relative to the selected intersection. Figure 3.2 shows dataflow of our system.

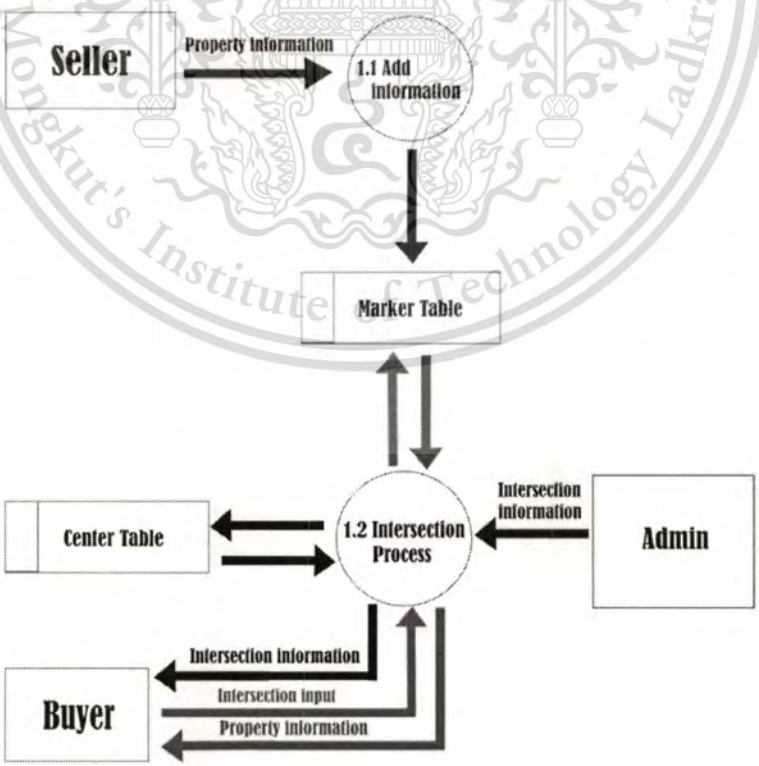


Figure 3.2 System dataflow diagram

3.2.1.2 System's overall diagram

This diagram shows overall in our system. The intersections are added manually by the administrator. A seller inputs data in HTML form on "input.html" page. When the seller clicks the submit button, data will be sent to AddFunction for inserting into database. A Buyer selects his conditions in HTML drop-down list. When submit button on the search page is clicked, two functions are called – SearchLocationNear and SearchLocation. SeachLocationNear executes query for selecting qualified properties in Marker table such that CreateMarker will plot them in red on Google map. Similarly, SeachLocation execute Query1 to retrieve selected intersection from Center Table to be plotted in blue. System's overall diagram shown in figure 3.3

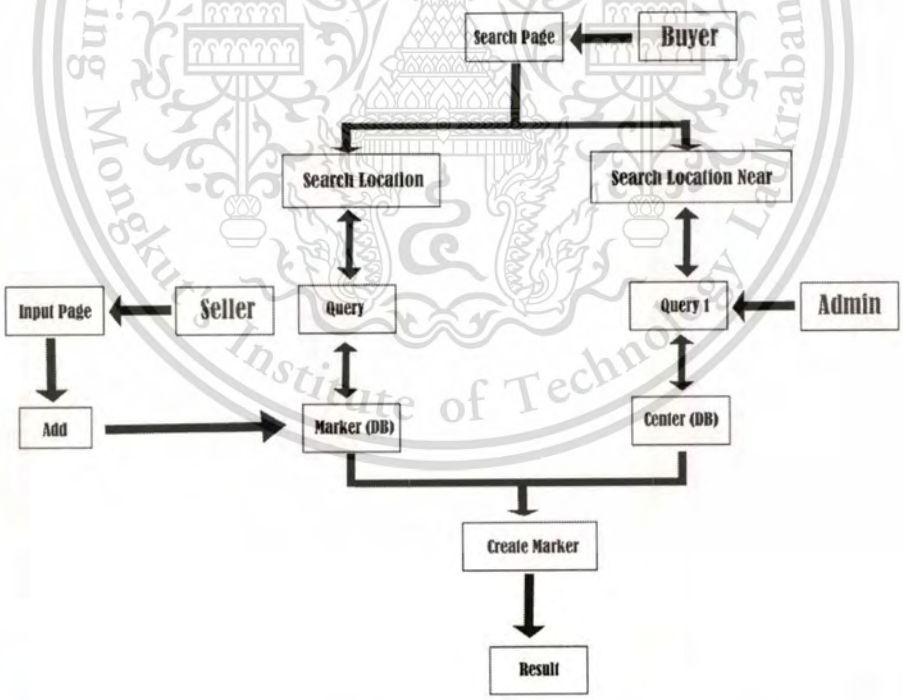


Figure 3.3 System's overall diagram

3.3 Database Design

We create two tables for storing center values and marker values. We will explain their schemas below.

3.3.1 Center Table

Center Table shown below in Table 3.1 is for storing intersection data. Its structure contains ID, name, address, latitude and longitude. The address attribute is for administrative purpose. Only the name value are populated into the drop down list in the search page.

Table 3.1 Center Table

Attribute	Data Type	Description	Example
ID	INTEGER()	Intersection Code	
Name	VARCHAR(60)	Name of intersection	Nida Intersection
Address	VARCHAR(80)	Address of intersection	Nida Intersection
Latitude	FLOAT(10,6)	Lat of intersection	13.76
Longitude	FLOAT(10,6)	Lng of intersection	100.63

3.3.2 Markers Table

Marker table shown in Table 3.2 is for storing real estate. Data is this table created by sellers. Its structure contains information of real estate (ID, Name, Address, Lat, Long, Type, Description, Images and Contact info).

Table 3.2 Marker Table

Attribute	Data Type	Description	Example
ID	INTEGER()	Place Code	
Name	VARCHAR()	Name of location from users	City Villa
Address	VARCHAR()	Address of location from users	Ladprao 122
Latitude	FLOAT()	Latitude of location from users	13.77
Longitude	FLOAT()	Longtitude of location from users	100.62
Type	VARCHAR()	Type of property from users	Condo
Description	VARCHAR()	Describe about data	3 Bedrooms, 2 Bathrooms
Images	VARCHAR()	Store image file name	001.JPG
Contact info	VARCHAR()	Store contact information	Tel: 089-xxx-xxxx

3.4 System development process

We create five separated files for input script. First, we put database connection information in the file named "DB.php". This is generally a good idea whenever you are using PHP to access a database, as it keeps your confidential information in a file instead of leaving it as part of normal code. It also allows other function, such as searching, to consistently use the common information. This file should look like Figure 3.4

```php
<?php
$DBServer = "localhost";
$username="root";
$password="adminadmin";
$database="project";
?>
```

Figure 3.4 Database information files

Secondly, a file named "input.html", shown below, was created to display page for users to input their data. The submitted data will be passed to the file named "add.php". Figure 3.5 shows "input.html" source code.

```
<form action="add.php" method="post"
ENCTYPE="multipart/form-data">
        Name : <input type="text" size="60" name="name"/> <br>
        Address : <input type="text" size="60" name="address"/> (ex.1234, ABC Rd., Bangkapi, Bangkok)<br>
  Type :<select name="Type">
    <option value="House">House</option>
    <option value="Condo">Condo</option>
    <option value="Space">Space</option>
  </select><br>
  Description: <textarea name="Description" rows="6" cols="40">
  </textarea><br>
        <input id="marker_lat" type="hidden" value="" name="lat"/>
        <input id="marker_lng" type="hidden" value="" name="lng"/>
  <!--Image: <input type="file" name="image" size="40"><br> -->
```

Figure 3.5 "input.html"

The add location function for sellers, to input their data (name, address, latitude, longitude, description, type and images) into map, is contained in the file named "add.php". The application will insert their data into MySQL database (marker table). Figure 3.6 shows "add.php" source code.

```
$strSQL = "INSERT INTO markers (name, address, lat, lng, type, description)";

    $strSQL = $strSQL."VALUES (\"".$_POST["name"]."\", \"".$_POST["address"]."\", ".$_POST["lat"].",      ".$_POST["lng"].",
\"".$_POST["Type"]."\", \"".$_POST["Description"]."\")";
```

Figure 3.6 "add.php"

Then, we create a file named "query.php" for creating XML document by connecting to the database, executing the SELECT by distance query discussed above from the markers table. The final task of this step is processing the resultset. For each row in the resultset (each location), create a new XML node with the row attributes as XML attributes, and append it to the parent node. Then dump the XML outputted to the screen. Figure 3.7 shows "query.php" code.

```
require("DB.php");

$center_lat = $_GET["lat"];

$center_lng = $_GET["lng"];

$radius = $_GET["radius"];

$type = $_GET["type"];

. . .

if ($type != All) {

$query = sprintf("SELECT address, name, lat, lng, type, ( 6371 * acos( cos( radians('%s') ) * cos( radians( lat ) ) * cos(
```

```
else

{$query = sprintf("SELECT address, name, lat, lng, type, ( 6371 * acos( cos( radians('%s') ) * cos( radians( lat ) ) * cos(
radians( lng ) - radians('%s') ) + sin( radians('%s') ) * sin( radians( lat ) ) ) ) AS distance FROM markers HAVING distance <
'%s' ORDER BY distance LIMIT 0 , 20",
```

Figure 3.7 "query.php"

The "query1.php" file in figure 3.8 is for creating XML document by obtaining intersection data, populated by the application administrator, from center table.

```
require("DB.php");

. . .


$query = "SELECT name, address, lat, lng FROM center WHERE name='".$center_name."'";
```

Figure 3.8 "query1.php"

For "map.html", we will explain details in separate parts, and then put all code together. First input JavaScript to retrieve map result from Google maps server for create our map that shown in figure 3.9. In our map, we specify only area in Bangkok.

```
map = new GMap2(document.getElementById('map'));
```

Figure 3.9 Loading maps from Google's server (parts of "map.html")

Secondly, the searchLocation function uses latitude and longitude of the selected intersection to perform the nearest top 20 properties selection. You can use this function, shown below in Figure 3.10, which just receives the intersection parameters from the textbox to get values in XML document that query from center table.

```
var searchUrl = 'query1.php?name=' + name;

...
```

Figure 3.10 SearchLocation function (parts of "map.html").

Next, the searchLocationNear function shown below in figure 3.11 passes the latitude and longitude values to our PHP script, and use the asynchronous GDownloadURL function to load the XML outputted into our page. GDownloadURL is a wrapper for the XMLHttpRequest that's used to request an XML file from the server where the HTML page resides. It takes the URL as the first parameter and passes the retrieved data to the callback function, if successful.

```
var radius = document.getElementById('radiusSelect').value;
    var type = document.getElementById('typeSelect').value;
    var searchUrl = 'query.php?lat=' + center.lat() + '&lng=' + center.lng() + '&radius=' + radius + '&type=' + type;


    GDownloadUrl(searchUrl, function(data) {
     var xml = GXml.parse(data);
     var markers = xml.documentElement.getElementsByTagName('marker');
```

Figure 3.11 SearchLocationNear Function (parts of "map.html")

Then, the createMarker function shown below in figure 3.12 is for pinning a red marker for every locations returned from the query on your Google map. An info window (callout) for each property with its image is created. When a user clicks the red marker triggering the marker event listener, the callout is displayed.

```
function createMarker(point, name, address, type, marker, path) {
      var marker = new GMarker(point, customIcons[marker]);
      var str = "<img src= upload/" + path + " width=250 height=150></img><b>";
      var html = "<b>" + name + "</b> <br/>" + address + "<br>"+ type +"<br>" + str;
GEvent.addListener(marker, 'click', function() {
marker.openInfoWindowHtml(html);
```

Figure 3.12 CreateMarker function (parts of "map.html")

Finally, createSidebarEntry is responsible for displaying information received as the function parameters on record at a time. The information is located on the left hand side of our map. Another feature of the function is that it displays info window when a record is clicked as well as changing the text background color (a nice UI touch) on mouseover event. As shown in figure 3.13, the function create a div variable for each record with addDomListener for each event namely click, mouseover and mouseout.

```
function createSidebarEntry(marker, name, address, distance) {
    var div = document.createElement('div');
    var html = '<b>' + name + '</b> (' + distance.toFixed(1) + ')<br/>' + address;
    div.innerHTML = html;
    div.style.cursor = 'pointer';
    div.style.marginBottom = '5px';
    GEvent.addDomListener(div, 'click', function() {
      GEvent.trigger(marker, 'click');
    });
    GEvent.addDomListener(div, 'mouseover', function() {
      div.style.backgroundColor = '#eee';
    });
    GEvent.addDomListener(div, 'mouseout', function() {
```

Figure 3.13 CreateSidebarEntry function (parts of "map.html").

Chapter 4 shows the results of all techniques we have covered in this section.

Chapter 4

Results and Discussion

4.1 System Operation

4.1.1 Main page

Figure 4.1 shows the first page of our system. Buyers can select center, radius, type for generate result. If users select center, the page will display result base on intersection name. Selecting radius will display distance from intersection. And type will display type of properties (For example: House, condominium)

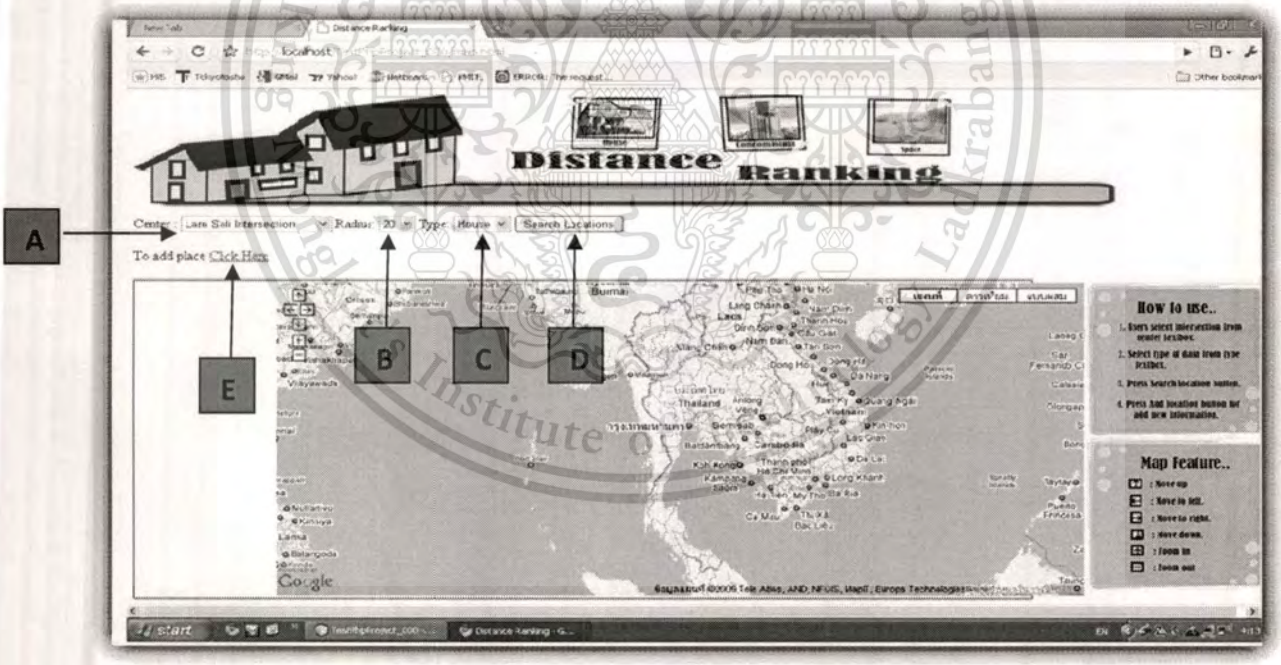


Figure 4.1 Main Page

A. Center List

B. Radius list

C. Type list

D. Search location button

E. Add place link

4.1.2 Add information page

If sellers need to add their location, they will click "Add place" then the system will show this page as in Figure 4.2. Sellers can input their information, upload image on this page and clicks submit to store data in database.

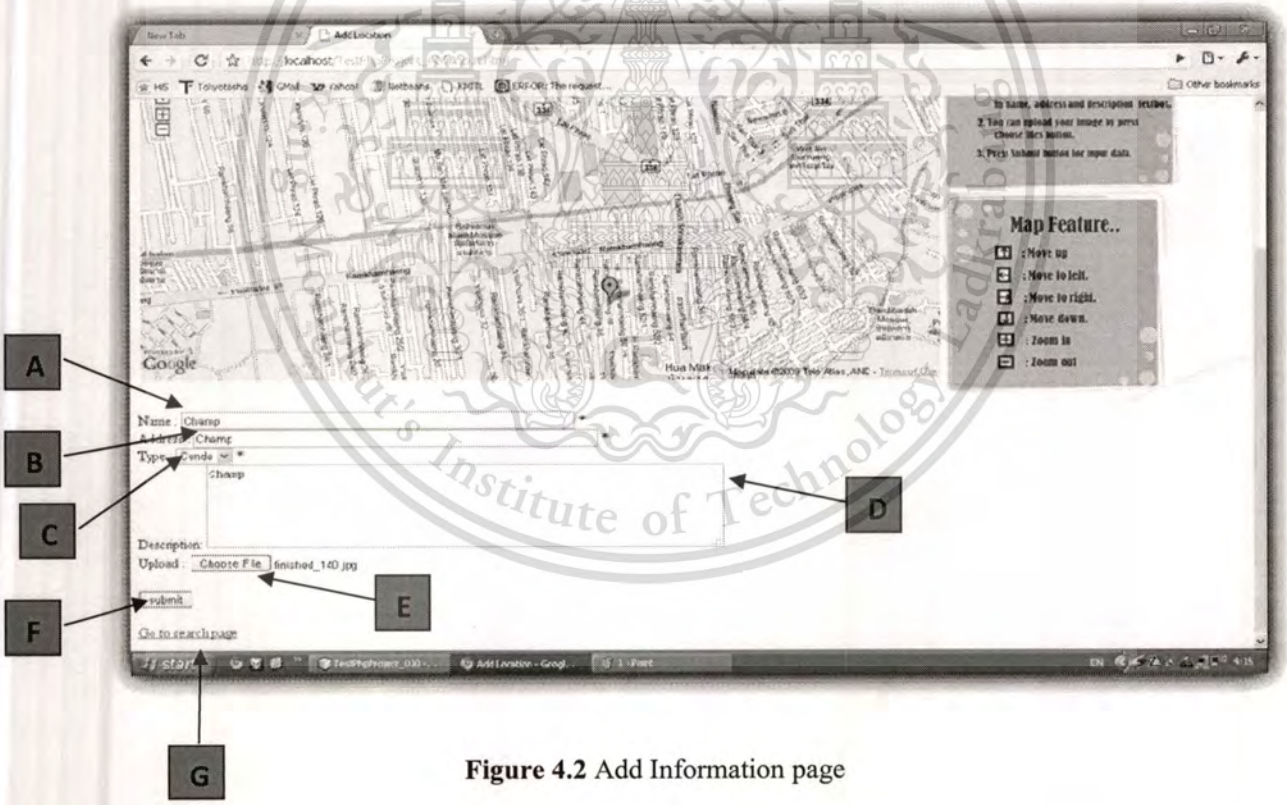


Figure 4.2 Add Information page

A. **Name Textbox:** to input location name.

B. **Address Textbox:** to input location address.

C. **Type:** select type of real estate.

D. **Description Textbox:** to input information about real estate.

E. **Upload button:** to upload image.

F. **Submit button:** for accept your information to store in database.

G. **Go to search page Link:** Back to users interface page.

Buyers can input information of their real estates and then click submit for adding to the database. If buyers input all of data that system required, after clicks submit, it will show "Add Successfully" but if some of fields are missed or forgotten to pin the marker on the map, it will show "Unable to Add".

4.1.3 Result page

When buyers select all of their interest and click "Search Location" they will shown in Figure 4.3. The result will display name, address of intersection, distance from intersection and display location image on map.

The system displays the selected intersection with a blue marker. Location markers are in red. Information and their ranked distances are shown in the left hand side of the screen. Clicking this information displays a table of the detail information with images and contact information.

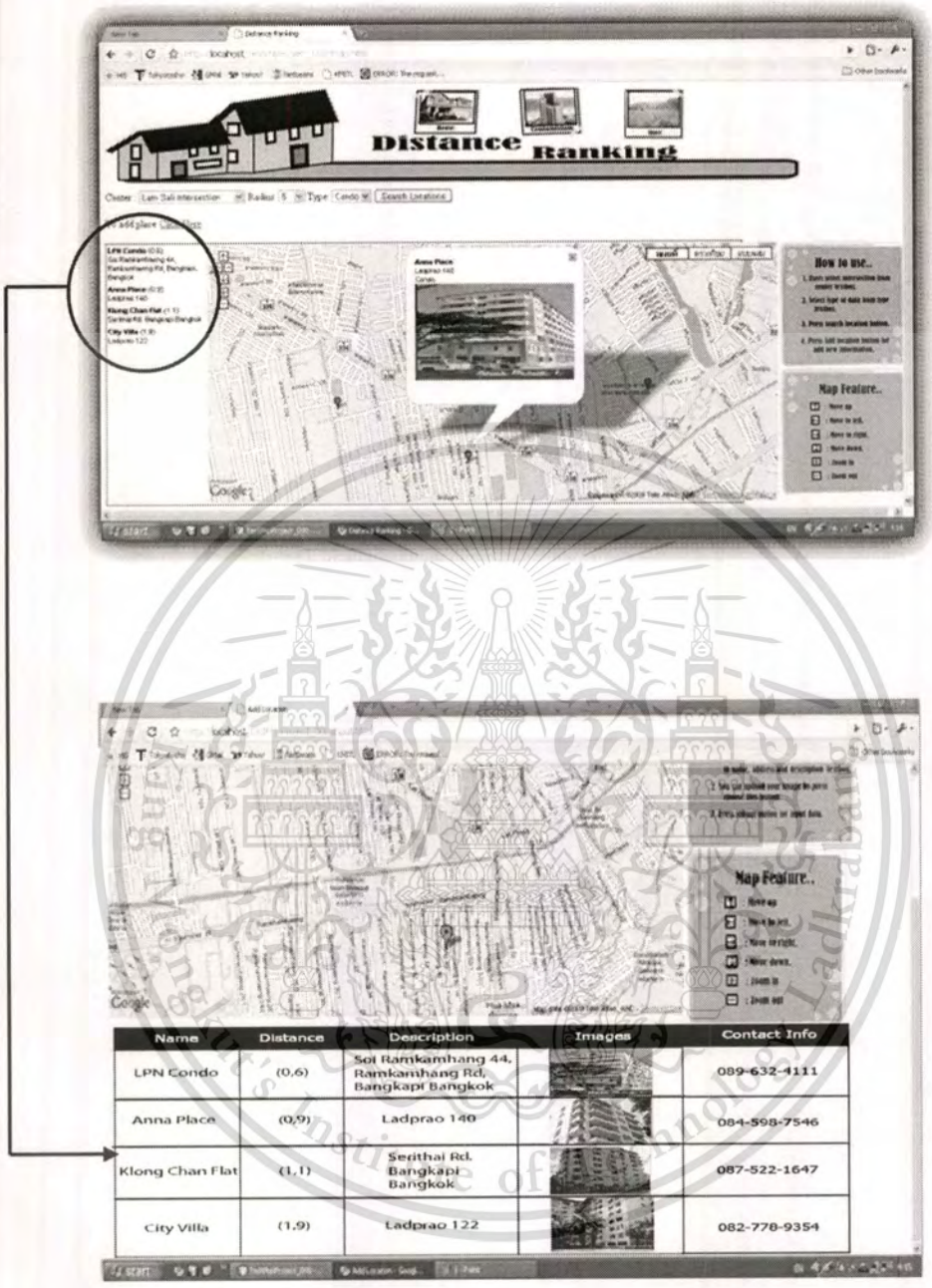


Figure 4.3 Result Page

Chapter 5

Conclusion and Recommendations

5.1 Conclusion

The objective of our project is to develop a property website by using Google maps. Our application uses Apache Web Server for running our project on localhost, Google maps API for display geographic information in our system we specific area in Ladprao road, HTML for create web user interface, PHP language for store and query data from MySQL.

The result page issuer friendly as it ranks the properties in term of vicinity to intersection selected.

5.2 Recommendations

There are many areas where we can improve our system. For example:

1. If the database is huge, it will need a well management. Partial database can be one of solutions to reduce search space.
2. An area could have many aliases; we have not allowed users to refer to the area by any other name.
3. More filtering options, for example Status checking, check status of properties that is sold or not.
4. Create a function like "Add to cart" in shopping website to collect properties information from users' interest for comparison regardless of the area.
5. Users can upload multiple images per property.

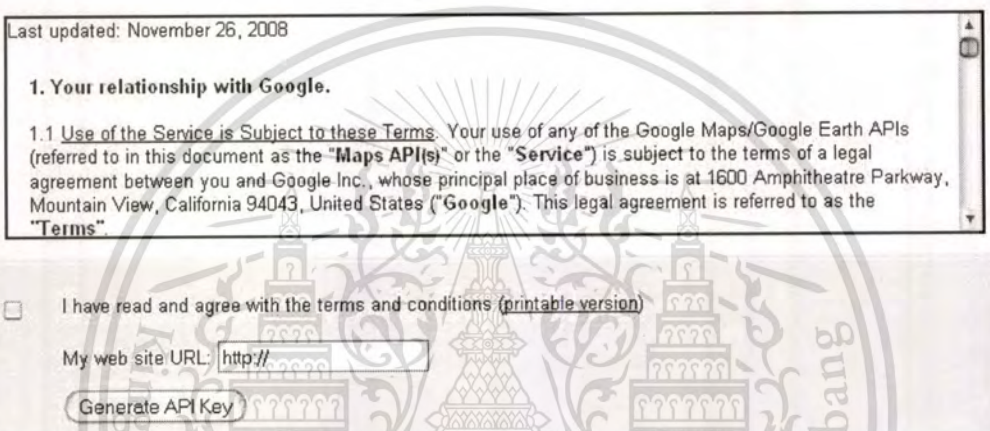
References

1. **http://www.php.com**
2. **http://www.mysql.com**
3. **http://www.apache.org**
4. **http://www.webmasterworld.com/php/3527779.htm**
5. http://en.**wikipedia**.org/**wiki/Google_Maps**
6. **http://www.tanomsak.com/index.php/2008/12/google-maps-insert-database/**
7. **http://code.google.com/support/bin/answer.py?answer=65622&topic=11364**
8. **http://www.movable-type.co.uk/scripts/latlong.html**
9. **http://www.nestoria.co.uk/**
10. **http://code.google.com/apis/maps/documentation/index.html**

Appendix A

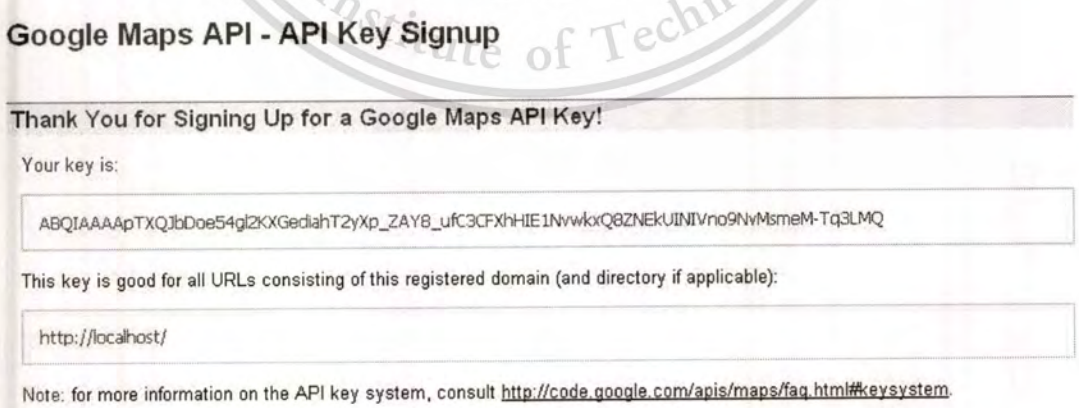
Registering for Google maps API key

Go to http://code.google.com/apis/maps/signup.html to register.



In my website URL box, in our case we use localhost and click Generate API Key.

- Google will generate key for your URL.



Key will put in the <script> tag in our project.

Appendix B

Google maps API reference

Google maps API can be separate into 6 sections, core class, base classes, event classes, control classes, overlay classes and services classes.

Core class:

- GMap2

This is the most important class within the Maps API. The other classes in this reference are grouped by their purpose.

Base Classes:

• GBounds	• GInfoWindowTab	• GMapOptions
• GBrowserIsCompatible	• GKeyboardHandler	• GMapPane
• GDraggableObject	• GLanguage	• GPoint
• GDraggableObjectOptions	• GLatLng	• GSize
• GInfoWindow	• GLatLngBounds	• GUnload
• GInfoWindowOptions	• GLog	

Event Classes:

• GEvent	• GEventListener

Control Classes:

• GControl • GControlAnchor • GControl • GControlPosition	• GHierarchicalMapTypeControl • GMapType • GMapTypeControl • GMapTypeOptions	• GMapUIOptions • GMenuMapTypeControl • GNavLabelControl

Overlay Classes:

• GCopyright • GCopyrightCollection • GGroundOverlay • GIcon • GLayer • GMarker • GMarkerManager • GMarkerManagerOptions • GMarkerOptions	• GMercatorProjection • GOverlay • GPolyEditingOptions • GPolyStyleOptions • GPolygon • GPolygonOptions • GPolyline • GPolylineOptions • GProjection	• GScreenOverlay • GScreenPoint • GScreenSize • GTileLayer • GTileLayerOptions • GTileLayerOverlay • GTileLayerOverlayOptions

Service Classes:

- GAdsManager
- GAdsManagerOptions
- GClientGeocoder
- GDirections
- GDirectionsOptions
- GDownloadUrl
- GFactualGeocodeCache
- GGeoAddressAccuracy
- GGeoStatusCode
- GGeoXml
- GGeocodeCache
- GGoogleBar
- GStreetviewLocation
- GStreetviewOverlay
- GStreetviewPanorama
- GStreetviewPanorama.ErrorValues
- GStreetviewPanoramaOptions
- GGoogleBarAdsOptions
- GGoogleBarLinkTarget
- GGoogleBarListingTypes
- GGoogleBarOptions
- GGoogleBarResultList
- GPov
- GRoute
- GStep
- GStreetviewClient
- GStreetviewClient.ReturnValues
- GStreetviewData
- GStreetviewLink
- GTrafficOverlay
- GTrafficOverlayOptions
- GTravelModes
- GXml
- GXmlHttp
- GXslt

We discuss some methods that often used in the table below.

Class GMap2

Instantiate class GMap2 in order to create a map. This is the central class in the API. Everything else is auxiliary.

Constructor

Constructor	Description
GMap2(container:Node, opts?:GMapOptions)	Creates a new map inside of the given HTML container, which is typically a DIV element. If no set of map types is given in the optional argument opts.mapTypes, the default set G_DEFAULT_MAP_TYPES is used. If no size is given in the optional argument opts.size, then the size of the container is used. If opts.size is given, then the container element of the map is resized accordingly. See class GMapOptions. Note: a Map needs to be centered before it can be used. You should immediately call GMap2.setCenter() to initialize a map created with this constructor.

Methods

Configuration

Method	Return Value	Description
enableDragging()	None	Enables the dragging of the map (enabled by default).
disableDragging()	None	Disables the dragging of the map.
draggingEnabled()	Boolean	Returns true iff the dragging of the map is enabled.
enableInfoWindow()	None	Enables info window operations on the map (enabled by default).
disableInfoWindow()	None	Closes the info window, if it is open, and disables the opening of a new info window.
infoWindowEnabled()	Boolean	Returns true iff the info window is enabled.
enableDoubleClickZoom()	None	Enables double click to zoom in and out (enabled by default).

Method	Return Value	Description
disableDoubleClickZoom()	None	Disables double click to zoom in and out.
doubleClickZoomEnabled()	Boolean	Returns true iff double click to zoom is enabled.
enableContinuousZoom()	None	Enables continuous smooth zooming for select browsers (disabled by default).
disableContinuousZoom()	None	Disables continuous smooth zooming.
continuousZoomEnabled()	Boolean	Returns true if continuous smooth zooming is enabled.
enableGoogleBar()	None	Enables the **GoogleBar**, an integrated search control, to the map. When enabled, this control takes the place of the default *Powered By Google* logo. Note that this control is not enabled by default.
disableGoogleBar()	None	Disables the **GoogleBar** integrated search control. When disabled, the default *Powered by Google* logo occupies the position formerly containing this control. Note that this control is already disabled by default.
enableScrollWheelZoom()	None	Enables zooming using a mouse's scroll wheel.

Method	Return value	Description
disableScrollWheelZoom()	None	Disables zooming using a mouse's scroll wheel.
scrollWheelZoomEnabled()	Boolean	Returns a Boolean indicating whether scroll wheel zooming is enabled.
enablePinchToZoom()	None	Enables pinching to zoom on an iPhone or iPod touch. Note: pinch to zoom is enabled by default.
disablePinchToZoom()	None	Disables pinching to zoom on an iPhone or iPod touch. Note: pinch to zoom is enabled by default.
pinchToZoomEnabled()	Boolean	Returns a Boolean indicating whether pinch to zoom is enabled.
getDefaultUI()	Object	Returns a GMapUIOptions object specifying default behaviour and UI elements for the Map, based on the UI of maps.google.com.
setUIToDefault()	None	Adds the default behaviour and UI elements specified in getDefaultUI() to the Map.
setUI(ui:GMapUIOptions)	None	Adds behaviour and UI elements specified in the ui parameter, which can be a modified version of the object returned from getDefaultUI().

Controls

Method	Return Value	Description
addControl(control:GControl, position?:GControlPosition)	None	Adds the control to the map. The position on the map is determined by the optional position argument. If this argument is absent, the default position of the control is used, as determined by the GControl.getDefaultPosition() method. A control instance must not be added more than once to the map.
removeControl(control:GControl)	None	Removes the control from the map. It does nothing if the control was never added to the map.
getContainer()	Node	Returns the DOM object that contains the map. Used by GControl.initialize().

Map Types

Method	Return Value	Description
getMapTypes()	GMapType[]	Returns the array of map types registered with this map.
getCurrentMapType()	GMapType	Returns the currently selected map type.
setMapType(type:GMapType)	None	Selects the given new map type. The type must be known to the map. See the constructor, and the method addMapType().
addMapType(type:GMapType)	None	Adds a new map type to the map. See section GMapType for how to define custom map types.
removeMapType(type:GMapType)	None	Removes the map type from the map. Will update the set of buttons displayed by the GMapTypeControl or GHierarchicalMapTypeControl and fire the removemaptype event.

Map State

Method	Return Value	Description
isLoaded()	Boolean	Returns true iff the map was initialized by setCenter() since it was created.
getCenter()	GLatLng	Returns the geographical coordinates of the center point of the map view.
getBounds()	GLatLngBounds	Returns the the visible rectangular region of the map view in geographical coordinates.

Method	Return value	Description
getBoundsZoomLevel(bounds:GLatLngBounds)	Number	Returns the zoom level at which the given rectangular region fits in the map view. The zoom level is computed for the currently selected map type. If no map type is selected yet, the first on the list of map types is used.
getSize()	GSize	Returns the size of the map view in pixels.
getZoom()	Number	Returns the current zoom level.
getDragObject()	GDraggableObject	Returns the draggable object used by this map.

Modify the Map State

Method	Return Value	Description
setCenter(center:GLatLng, zoom?:Number, type?:GMapType)	None	Sets the map view to the given center. Optionally, also sets zoom level and map type. The map type must be known to the map. See the constructor, and the method addMapType(). This method must be called first after construction to set the initial state of the map. **It is an error to call operations on a newly constructed GMap2 object until after this function is invoked.**
panTo(center:GLatLng)	None	Changes the center point of the map to the given point. If the point is already visible in the current map view, change the center in a smooth animation.
panBy(distance:GSize)	None	Starts a pan animation by the given distance in pixels.

Method	Return value	Description
panDirection(dx:one of -1 0 +1, dy:one of -1 0 +1)	None	Starts a pan animation by half the width of the map in the indicated directions. +1 is right and down, -1 is left and up, respectively.
setZoom(level:Number)	None	Sets the zoom level to the given new value.
zoomIn(latlng?:GLatLng, doCenter?:Boolean, doContinuousZoom?:Boolean)	None	Increments zoom level by one zoom level. If an optional latlng argument is provided, the map will attempt to keep that location present on the zoomed-in map. (Note that the latlng must currently be present on the zoomed-out map.) If the optional doCenter is provided, additionally, the map will be centered on the provided latlng.
zoomOut(latlng?:GLatLng, doContinuousZoom?:Boolean)	None	Decrements zoom level by one. If the latlng is provided, the map will zoom about that point. If doContinuousZoom is set to true, the map will use continuous zoom when zooming out.
savePosition()	None	Stores the current map position and zoom level for later recall by returnToSavedPosition().
returnToSavedPosition()	None	Restores the map view that was saved by savePosition().

Method	Return Value	Description
checkResize()	None	Notifies the map of a change of the size of its container. Call this method after the size of the container DOM object has changed, so that the map can adjust itself to fit the new size.

Overlays

Method	Return Value	Description
addOverlay(overlay:GOverlay)	None	Adds an overlay to the map and fires the addoverlay event.
removeOverlay(overlay:GOverlay)	None	Removes the overlay from the map. It is an error to try to remove an overlay that is not on the map. If the call is successful, it fires the removeoverlay event.
clearOverlays()	None	Removes all overlays from the map, and fires the clearoverlays event.

Method	Return Value	Description
getPane(pane:GMapPane)	Node	Returns a DIV that holds the object in the layer identified by pane. Used by GOverlay instances in method GOverlay.initialize() instances to draw themselves on the map

Info Window

Method	Return Value	Description
openInfoWindow(latlng:GLatLng, node:Node, opts?:GInfoWindowOptions)	None	Opens a simple info window at the given point. Pans the map such that the opened info window is fully visible. The content of the info window is given as a DOM node.
openInfoWindowHtml(latlng:GLatLng, html:String, opts?:GInfoWindowOptions)	None	Opens a simple info window at the given point. Pans the map such that the opened info window is fully visible. The content of the info window is given as HTML text.

Method	Return Value	Description
openInfoWindowTabs(latlng:GLatLng, tabs:GInfoWindowTab[], opts?:GinfoWindowOptions)	None	Opens a tabbed info window at the given point. Pans the map such that the opened info window is fully visible. The content of the info window is given as DOM nodes.
openInfoWindowTabsHtml(latlng:GLatLng, tabs:GInfoWindowTab[], opts?:GInfoWindowOptions)	None	Opens a tabbed info window at the given point. Pans the map such that the opened info window is fully visible. The content of the info window is given as HTML text.
showMapBlowup(latlng:GLatLng, opts?:GInfoWindowOptions)	None	Opens an info window at the given point that contains a closeup view on the map around this point.
updateInfoWindow(tabs:GInfoWindowTab[], onupdate?:Function)	None	Updates the content of the currently open GInfoWindow object, without repositioning. The info window is resized to fit the new content. The optional onupdate callback function is called after the info window content is actually changed.

Method	Return Value	Description
updateCurrentTab(modifier:Function, onupdate?:Function)	None	Updates the currently selected tab, causing a resize of the GInfoWindow object, without repositioning. The modifier function is used to modify the currently selected tab and is passed a GInfoWindowTab as an argument. The optional onupdate callback function is called after the info window displays the new content.
closeInfoWindow()	None	Closes the currently open info window.
getInfoWindow()	GInfoWindow	Returns the info window object of this map. If no info window exists yet, it is created, but not displayed. This operation is not influenced by enableInfoWindow().

Events

Event	Description
addmaptype(type:GMapType)	This event is fired when a map type is added to the map.
removemaptype(type:GMapType)	This event is fired when a map type is removed from the map.
click(overlay:GOverlay, latlng:GLatLng, overlaylatlng:GLatLng)	This event is fired when the user clicks on the map with the mouse. A click event passes different arguments based on the context of the click, and whether or not the click occured on a clickable overlay. If the click does not occur on a clickable overlay, the overlay argument is null and the latlng argument contains the geographical coordinates of the point that was clicked. If the user clicks on an overlay that is clickable (such as a GMarker, GPolygon, GPolyline, or GInfoWindow), the overlay argument contains the overlay object, while the overlaylatlng argument contains the coordinates of the clicked overlay. In addition, a click event is then also fired on the overlay itself.

Event	Description
dblclick(overlay:GOverlay, latlng:GLatLng)	This event is fired when a double click is done on the map. Notice that this event will not be fired if the double click was on a marker or other clickable overlay. The geographical coordinates of the point that was double clicked are passed in the latlng argument. The overlay argument is always set to null.
singlerightclick(point:GPoint, src:Element, overlay?:GOverlay)	This event is fired when the DOM contextmenu event is fired on the map container. If the right click was on a marker or other clickable overlay, then the overlay is passed to the event handler in the overlay argument. The pixel coordinates (in the DOM element that holds the map) of the point that was right clicked and the source element of the DOM event are passed in the point and src arguments respectively. Note that if it is a double right click and double click to zoom is enabled, then the map zooms out and no singlerightclick event is fired. If, however, double click to zoom is disabled, two singlerightclick events will be fired.

Event	Description
movestart()	This event is fired when the map view starts changing. This can be caused by dragging, in which case a dragstart event is also fired, or by invocation of a method that changes the map view.
move()	This event is fired, possibly repeatedly, while the map view is changing.
moveend()	This event is fired when the change of the map view ends.
zoomend(oldLevel:Number, newLevel:Number)	This event is fired when the map reaches a new zoom level. The event handler receives the previous and the new zoom level as arguments.
maptypechanged()	This event is fired when another map type is selected.
infowindowopen()	This event is fired when the info window opens.
infowindowbeforeclose()	This event is fired before the info window closes.

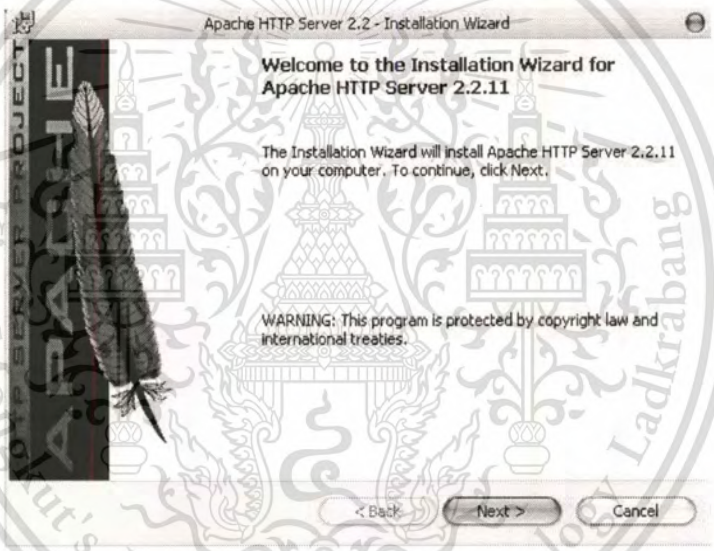
Event	Description
infowindowclose()	This event is fired when the info window closes. The event infowindowbeforeclose is fired before this event. If a currently open info window is reopened at a different point using another call to openInfoWindow*(), the events infowindowbeforeclose, infowindowclose and infowindowopen are fired in this order.
addoverlay(overlay:GOverlay)	This event is fired when a single overlay is added to the map by the method addOverlay(). The new overlay is passed as an argument overlay to the event handler.
removeoverlay(overlay:GOverlay)	This event is fired when a single overlay is removed by the method removeOverlay(). The overlay that was removed is passed as an argument overlay to the event handler.
clearoverlays()	This event is fired when all overlays are removed at once by the method clearOverlays().
mouseover(latlng:GLatLng)	This event is fired when the user moves the mouse over the map from outside the map.

Event	Description
mouseout(latlng:GLatLng)	This event is fired when the user moves the mouse off the map.
mousemove(latlng:GLatLng)	This event is fired when the user moves the mouse inside the map.
dragstart()	This event is fired when the user starts dragging the map.
drag()	This event is repeatedly fired while the user drags the map.
dragend()	This event is fired when the user stops dragging the map.
load()	This event is fired when the map setup is complete, and isLoaded() would return true. This means position, zoom, and map type are all initialized, but tile images may still be loading.

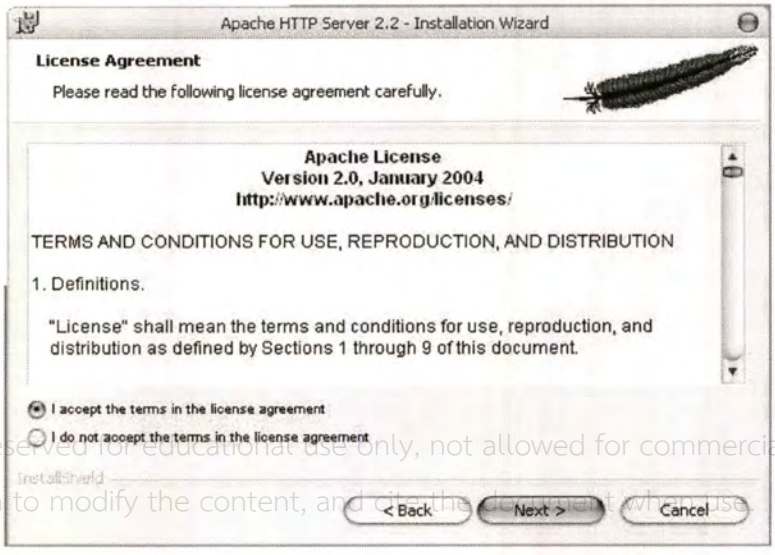
Appendix C

Install Apache HTTP Server

1. Double click on installation files. Program will come to the part of installation. Then click "Next"



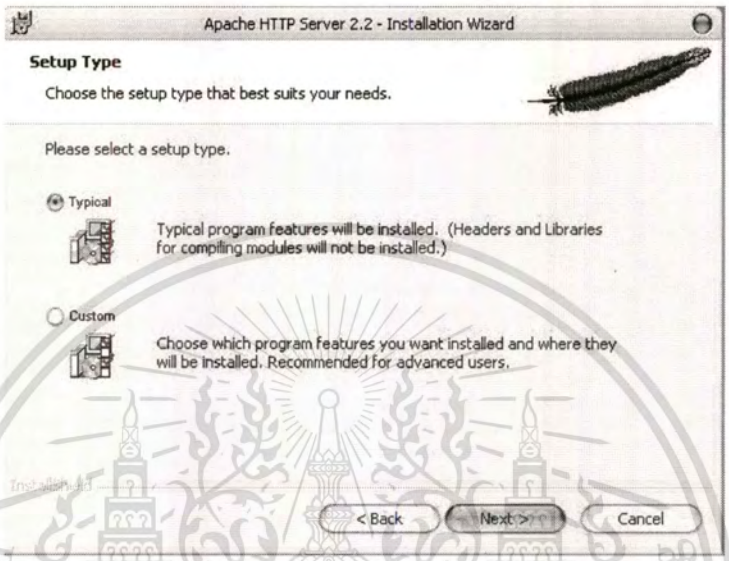
2. In the license agreement, click "I accept" and "next".



3. In the Read this first, it shows details of Apache HTTP server. Click Next

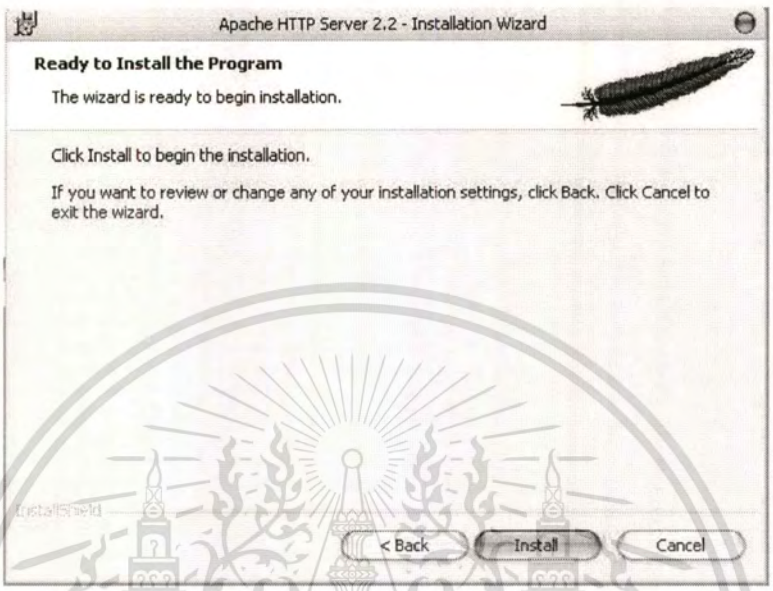
4. In the Server information page, you must define network domain, server domain and e-mail address. In the bottom of page, you can define priority of users to use this program.

5. This step is to select type of installation, we recommend selecting Complete. Then click next.

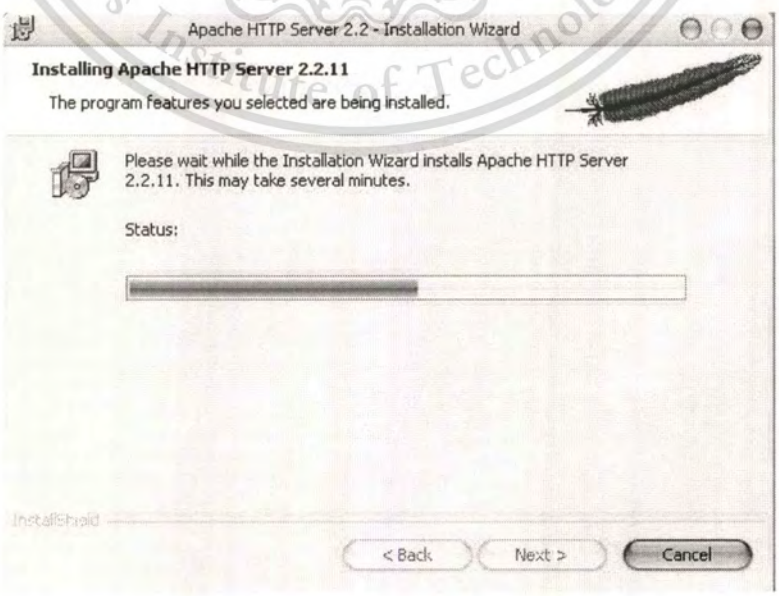


6. Select the destination folder and click next.

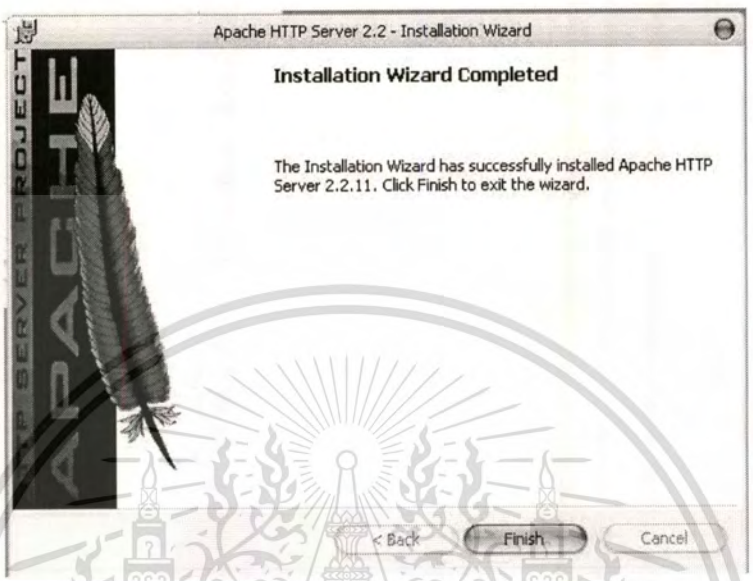
7. This page tells you it ready to install. Click install.



8. Installing… Please wait a few minutes

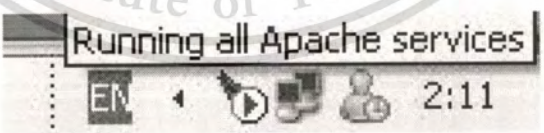


9. Installation wizard completed. Click Finish



When you installed Apache completely, you can see in the right-bottom of the screen.

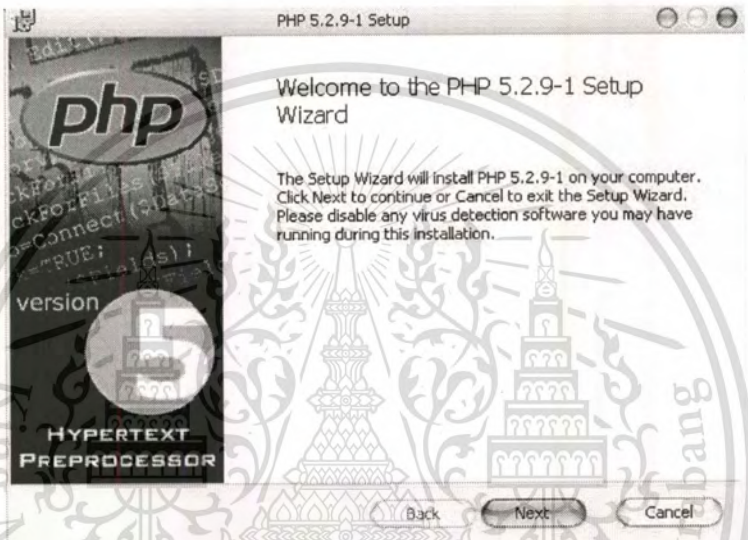
You will see the icon of the program that means your computer install Apache successfully.



Appendix D

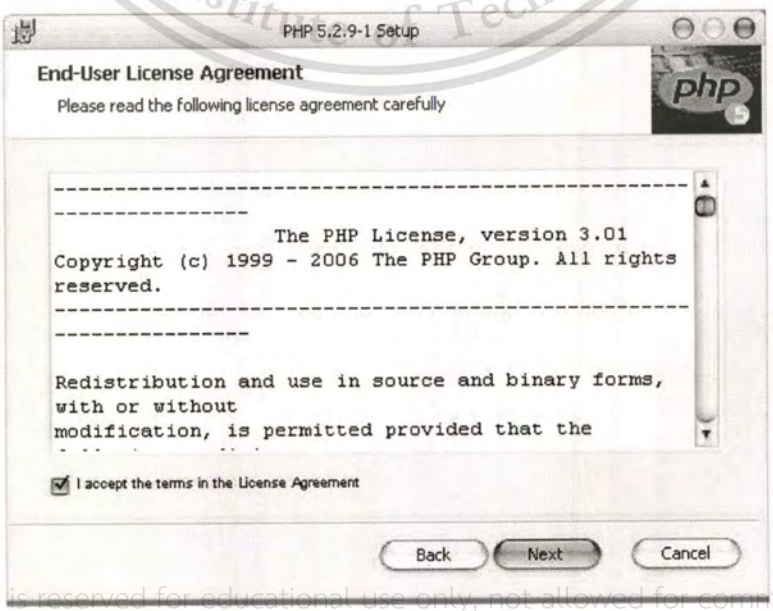
PHP Installation

1. Double click the installation file, this screen will appear. Click next.

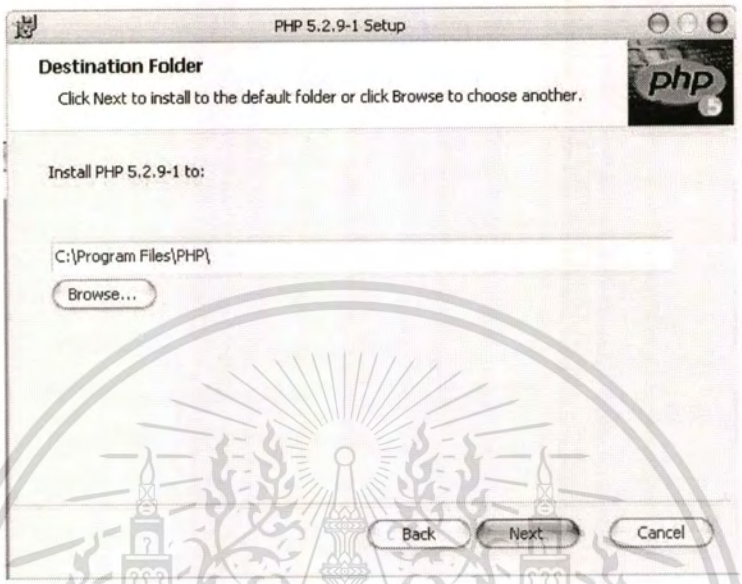


2. This screen tells about license agreement, check "I Agree" and click next

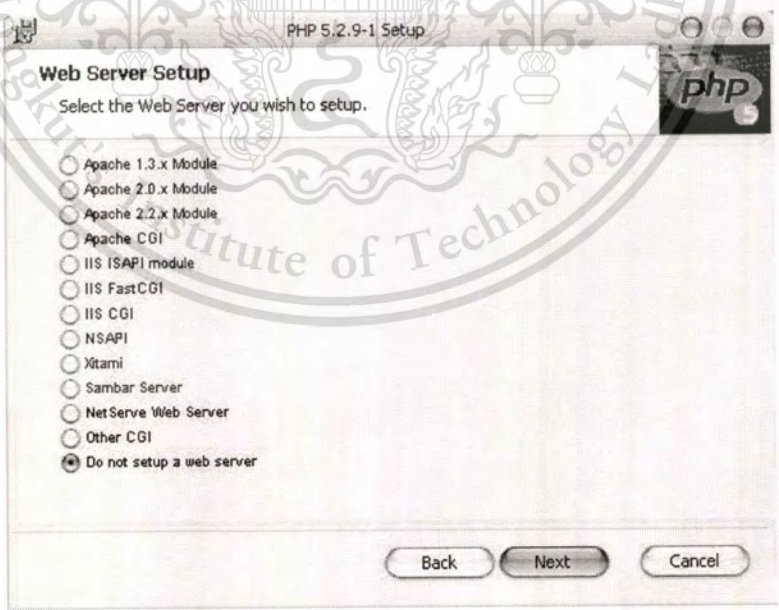
.



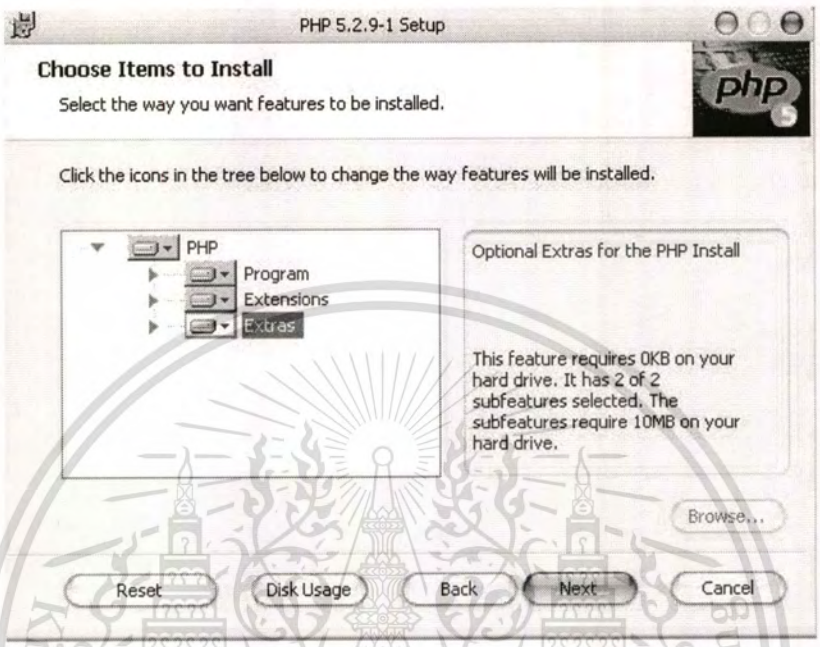
3. Select the destination folder and click next.



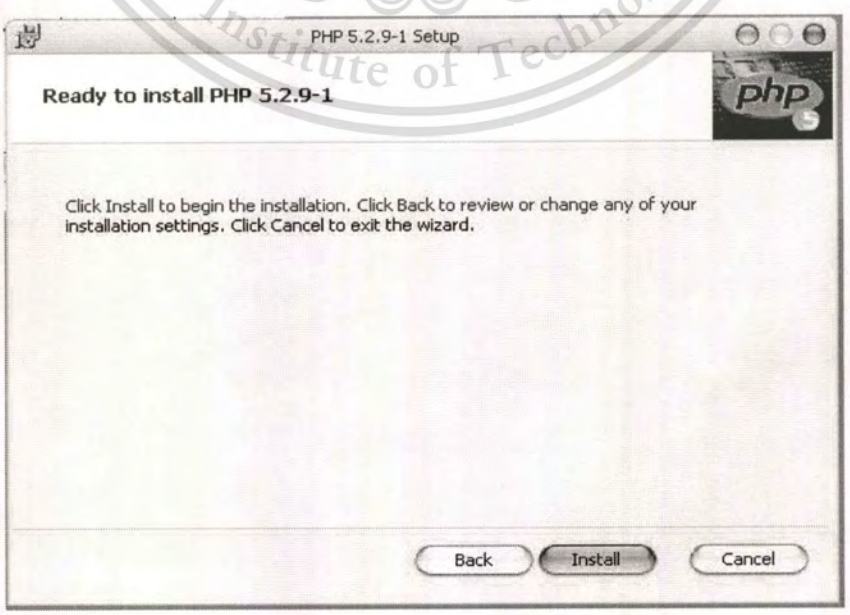
4. Select web servers to install. If you installed Apache before, select do not setup and click next.



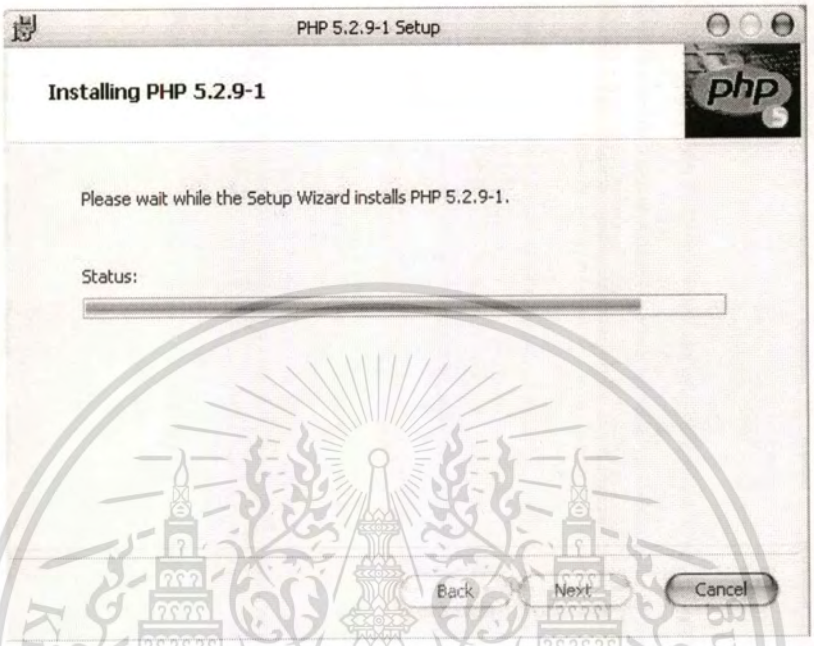
5. Select PHP features to install. We recommend selecting all. Then click next.



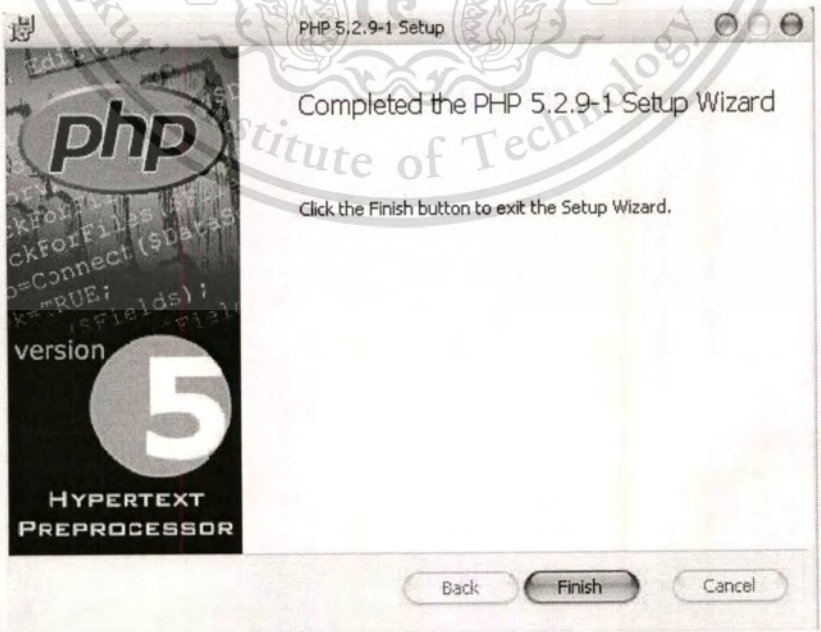
6. It will start to install to your computer. Click install.



7. Installing… Please wait



8. Click finish to complete the installation.



Author Biography

Name: Mr.Chatchol Isarangkul

Birth: 21 April 1987

Place of birth: Bangkok

Education: 1998-2004 High school and junior high school

Triamudomsuksa Pattanakarn school

Bangkok, Thailand

Name: Ms.Thamonwan Kajornsaksopon

Birth: 20 April 1987

Place of birth: Bangkok

Education: 1998-2004 High school and junior high school

Bodindecha 2 school

Bangkok, Thailand