PROMOTION AND RESTAURANT FINDER ON ANDROID

CHAIYAPONG PIPITHANACHOTI

PAKORN PHATIKORNWONG

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

IN COMPUTER SCIENCE (INTERNATIONAL PROGRAM)

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2011

Thesis Title	Promotion and Restaurant Finder on Android		
Students	Mr. Chaiyapong	Pipithanachoti	Student ID: 51051174
	Mr. Pakorn	Phatikornwong	Student ID: 51051208
Degree	Bachelor of Science		
Program	Computer Science (International Program)		
Year	2011		
Thesis Advisor	Dr. Rungrat Wiangsripanawan		

ABSTRACT

This special project aim is to develop an Android application called "KinArai". The main objectives of the application are to provide the latest restaurants' promotions and to provide a list and maps of restaurants located near the GPS location of the user. Therefore, the application consists of two main functions: promotion and nearby. The nearby function also allows the user to search for restaurants near the specified place. In the implementation, promotion information are retrieved from the feed of 5 selected promotion sites and stored in the database implemented on appserv (Apache, PHP MySQL and phpMyAdmin). XML and JSON are used for the message exchanged formats between the android device and the web server. The Google API is used to provide the map functions.

Keywords: Android, Google API, KinArai, Promotion, nearby, location, RSS, LBS

ACKNOWLEDGEMENTS

This special project Promotion and Restaurant Finder would not have been possible without the help from many people.

Special thanks to our parents for their unconditional loves and supports. Thanks for cheering us up, teaching us and comforting us when we confronted with the hard problems.

Special thanks to Dr.Rungrat Wiangsripanawan for her advices and encouragement to solve problems that we had in this special project. We would like to thank for all her comments which gave us a lot of motivation.

Special thanks to Assoc.Prof.Dr.Jeeraporn Werapun and Asst.Prof.Dr.Korakot Prachumrak for their suggestions on the improvement of our application and Dr.Warangkhana Kimpan for the suggestions on the database design and implementation.

Last but not least, special thanks to all our friends particularly - P'Ming (Nikom Suwankamol) for his help and suggestions on PHP implementation, Eakawat Tantamjarik for his help on fixing bugs on Android, Parawadee Khraichom, Noppawan Chumnanchang and Mathus Maninoi for their warmth support and help in doing this special project.

Chaiyapong Pipithanachoti

Pakorn Phatikornwong

Contents

Contents (Cont.)

Contents (Cont.)

Page

Contents (Cont.)

Contents (Cont.)

Contents (Cont.)

Contents (Cont.)

List of Tables

List of Figures

List of Figures (Cont.)

List of Figures (Cont.)

List of Figures (Cont.)

List of Figures (Cont.)

List of Figures (Cont.)

List of Figures (Cont.)

List of Figures (Cont.)

Chapter 1

Introduction

1.1 Importance and cause of problem

Nowadays, the mobile device, in particular the smart phone, has become an important tool that everyone should have. Not only used for voice communication as in its twenty years ago version, the smart phone can facilitate people for all activities in their daily lives. Meanwhile, the Internet has also become another essential tool for people in this decade. It is the source of all information, a place for social meeting, a place for entertainment etc. The rapid growth of the mobile device, the Internet and both of their technologies has strongly affected the society in all aspects such as in communication, business and entertainment. In particular, it creates several new life styles which require new kinds of applications that could not be found or imagined of a decade ago. Fast access to the Internet and easy to use are two basic requirements. Other specific requirements depend on the needs of their users. This opens an opportunity for developers to create several new mobile applications. One of the new application types popularly released onto the market is the Location Based Service (LBS) applications. These applications use the GPS (Global Positioning System) feature on the smart phone for some specific location purposes. Its example is Foursquare.

Therefore, for our project, we decide to implement a Location Based Service (LBS) mobile application on Android smart phone, "KinArai". Our application aims at two objectives. The first objective is to provide the locations and maps of the restaurants located near the user's phone. The second objective is to allow the user to search for the latest restaurant promotion. Our application uses the features of the smart phone such as GPS together with a service from Google API such as Google Map. Since it is implemented particularly as an application for Android phone, we

believe that our application will give better satisfaction and result to the user than using the web application (Webapp) on the phone.

1.2 Purpose of the special project

1.2.1) To update new promotions of restaurants.

1.2.2) To search for promotions of the restaurant.

1.2.3) To search and navigate to restaurants nearby.

1.3 Coverage of the special project

1.3.1) Use Google Map API as the basic tool to create the map.

1.3.2) Use RSS **(Really Simple Syndication)** to retrieve promotions from the web promotion sites.

1.3.3) Use database to collect and store the location and promotion information which HTTP Get/Post are used to retrieve data from the database.

1.3.4) Use PHP to get RSS feed from promotion website and filter only tags that we want in XML files and store the extracted value in the database.

1.3.5) Use the web server to get requests from an Android device and respond back in JSON format to show the user.

1.4 Expected benefits

1.4.1) Developer

1.4.1.1) Gain knowledge in Java language to develop programs on Android OS device.

1.4.1.2) Gain knowledge in PHP language for developing the web service.

1.4.1.3) Gain knowledge for using Google Map API to calculate distance from latitude, longitude and method to show map of restaurant.

1.4.1.4) Gain knowledge to manage the database.

1.4.2) User

1.4.2.1) Knowing the latest restaurant promotion from this application.

1.4.2.2) Knowing how many restaurants that surround them.

1.4.2.3) Helping the user to make a decision.

1.5 Implementation procedures

1.5.1) Study Google API for android mobile device.

1.5.2) Study RSS feed.

1.5.3) Study JSON format.

1.5.4) Study java language for coding in Android platform.

1.5.5) Design an interface of our application.

1.5.6) Analyze and design the system.

1.5.7) Setup database

1.5.8) Install the web server.

1.5.9) Install the android emulator.

1.5.10) Implement.

1.5.11) Test filtering promotion procedures at the back-end.

1.5.12) Test information insertion procedures for the database at the back-end.

1.5.13) Test they system whether it can provide the information to android correctly.

1.5.14) Debug and test our application.

1.5.15) Write the documentation.

1.6 Organization

This special project consists of 5 chapters.

Chapter 1: Introduction

Chapter 2: Background

Chapter 3: System Analysis and design

Chapter 4: Implement of this special project

Chapter 5: Conclusion of this special project

1.7 Equipments used to implement this special project

1.7.1) Hardware:

- HTC Desire S (with android 2.3.5)
- Three laptops:
  - Two laptops are used to code Android application and PHP for web service.
  - One laptop is used to install database and web server.

1.7.2) Software :

For the database and server:

- Apache 2.2.8
- PHP 5.2.6
- MySQL 5.0.51b
- phpMyAdmin 2.10.3
- Windows 7

For developing applications on Android OS:

- Java Runtime Engine JRE version 6
- Eclipse IDE for Java Developers indigo version
- Android SDK version 16.0.0
- Google API version 10

Chapter 2

Background

This chapter represents the basic of Android OS, Android SDK, how to develop the Android platforms. Our application also uses PHP and MySQL for the database, Apache for the web server and RSS Feed to obtain the real time information from the websites. The NearBy function uses GPS and Google API for its location service.

2.1 Android [1]

2.1.1 What is an Android?

Android is a software that its structure is similar to stack model. Android consists of multiware and application, which is developed for small devices such as smart phones and tablets. Android is created base on Linux Kernel. To develop an android application, Android SDK (Software development kit) is used. Android program uses java language for development.

2.1.2 Android background

Android is an operating system for mobile devices such as smartphones and tablets. It was developed by Andy Rubin in October 2003. In August 2005, Google has bought android from Andy since his company had financial problem. Since then Google started to get involved in mobile phone market.

2.1.3 Android Libraries

Android provides several library tools, which aims at facilitating developers to develop the application more easily.

Dalvik Virtual Machine (VM) is the virtual machine created specifically to develop android applications. It is a part of the emulator.

Integrated Browser provides a webkit library for implementing web Browser.

Optimized Graphic provided to support graphics in 2D and 3D mode through OpenGL.

SQLite is the internal database that is stored in the mobile phone.

Media Support supports the multimedia functions such as audio or video file.

GSM Telephony supports mobile phone functions for Bluetooth, EDGE, 3G, and WIFI.

Rich development Environment supports functions for developing the application such as emulator, debugging tools.

2.1.4 Android Architecture

The Android Architecture consists of 4 main layers:

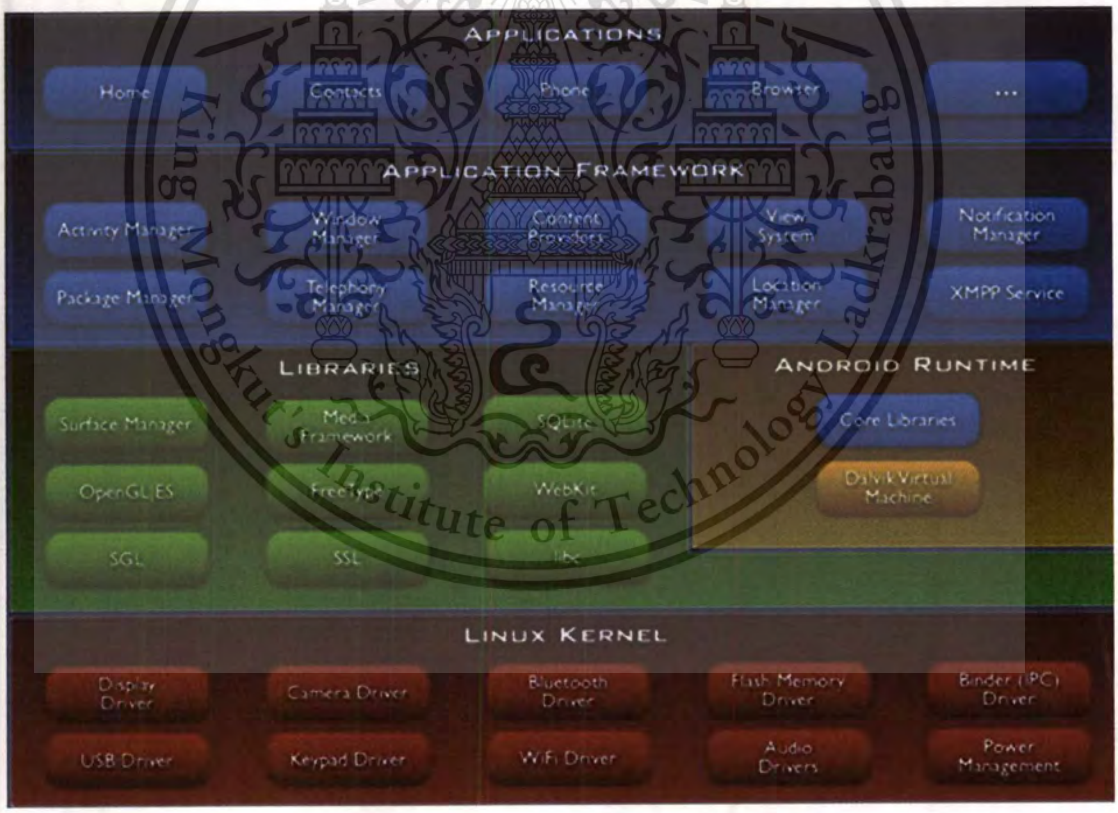


Figure 2.1 Layer of android architecture

Application Layer

This layer is on the top of Android architecture. This is the part of application such as web browser and phone dial. Data type is .apk file.

Note that android provides a list of core applications such as an email client, SMS program, calendar, maps and browsers.

Figure 2.2 Application layer

Application Framework

Normally, the developer uses Android API (Application Programming Interface) since Android is designed to reduce a duplication of application component.

Figure 2.3 Application Framework

Library Layer

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are used by developers through the Android application framework.

Figure 2.4 Library layer

Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

Figure 2.5 Linux Kernel

2.1.5 Android version 2.3 (Gingerbread) [2]

Gingerbread is the Android codename for the version 2.3 update of the open source Android mobile operating system. Gingerbread made its debut in December 2010 for a variety of smartphones, introducing Google Voice over Wi-Fi, enhanced gaming functionality and improved Google Apps. In our project we use Android version 2.3 (Gingerbread).

Display

WXGA (1280 x 768) or higher

Performance

- All-around performance improvements.
- Easy to use keyboard with multitouch.
- Better and global copy & paste.
- Improved power management. (battery life)
- Native VOIP Internet calling.

2.1.6 Android SDK [2]

For developing the application on android, developers use java language on ADT (Android development Tools). ADT is a part of IDE that uses for coding the program and ADT is also a part of Android SDK.

2.1.7 What is an Android SDK? [3]

A software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications. Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use, which runs on top of a Linux kernel.

2.2 Really Simple Syndication (RSS Feed) [4]

Figure 2.6 RSS Feed symbol

RSS is also called **web feeds**. RSS is a content delivery vehicle. It is the format used when you want to syndicate news and other web content. When it distributes the content it is called a feed. You could think of RSS as your own personal wire service.

RSS feeds benefit publishers by letting them syndicate content automatically. A standardized XML file format allows the information to be published once and viewed by many different programs. They benefit readers who want to subscribe to timely updates from favored websites or to aggregate feeds from many sites into one place. http://www.press-feed.com/howitworks/rss_tutorial.php

2.2.1 Advantages of RSS Feed

- It gives the user the latest updates. Whether it is about the weather, new music, software upgrade, local news, or a new posting from a rarely-updates site learn about the latest as soon as it comes out.
- It gives the power of subscription to the user. Users are given a free hand on which websites to subscribe.
- It saves on surfing time. RSS feed provides a summary of the related article.
- It is spam free. RSS does not make use of your email address to send updates thus your privacy is kept safe from spam mails.
- It can be used as an advertising or marketing tool. Users who subscribe product on websites.

2.2.2 RSS Feed XML Structure

```
<?xml version="1.0" ?>
    <rss version="2.0">
        <channel>
            <title>XULandXML</title>
            <link>http://www.xul.fr/en/</link>
            <description>XML graphical interface
                etc...</description>
        </channel>
</rss>
```

- **rss**: The global container.
- **title**: The title of the channel. Should contains the name.
- **link**: The URL of your website:
- **description**: Summary of the article limit about 200 characters.

2.3 XML [5]

Extensible Markup Language (XML) is a set of rules for encoding documents in machines-readable form. The design goals of XML emphasize simplicity, generality, and usability over the Internet. XML is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere

2.3.1 XML Syntax rule

In XML, it is not allowed to omit the closing tag. All elements **must** have a closing tag.

This is a paragraph

This is another paragraph

2.3.1.1 XML tags are case sensitive

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>. Hence, the these two tags are different.

<Message>This is incorrect</message>

<message>This is correct</message>

2.3.1.2 XML elements must be properly nested

In HTML, you might see improperly nested elements:

This text is bold and italic

In XML, all elements **must** be properly nested within each other:

This text is bold and italic

2.3.1.3 XML documents must have a root element [5]

XML documents must contain one element that is the **parent** of all other elements. This element is called the **root** element.

```
<root>
 <child>
  <subchild>.....</subchild>
 </child>
</root>
```

2.3.1.4 XML attribute values must be quoted

XML elements can have attributes in name/value pairs just like in HTML. In XML, the attribute values must always be quoted.

```
<note date="12/11/2007">
        <to>Tove</to>
        <from>Jani</from>
</note>
```

2.3.1.5 Entity references

Some characters have a special meaning in XML. If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

The following example will generate an XML error:

```
<message>if salary < 1000 then</message>
```

To avoid this error, replace the "<" character with an **entity reference**:

```
<message>if salary &lt; 1000 then</message>
```

There are 5-entity references in XML as shown in Table 2.1.

Table 2.1 Entity reference table

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

2.4 PHP [6]

PHP is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages. PHP code is embedded into the HTML source document and interpreted by a web server. PHP can be deployed on most web servers.

2.4.1 What can PHP do?

- Dynamically edit, change or add any content to a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the results to a browser
- Customize a Web page to make it more useful for individual users
- Provide security since your server code cannot be viewed from a browser

2.4.2 Get start with PHP [7]

PHP must start with <?php and end with ?> or start with <? and end with?>. The PHP language can be inserted anywhere in HTML.

```
<?php
        include (“moreFunction.php”);
?>
<html>
<head> <?php
        function hello()
        { echo "Hello World"; }
?>
</head>
<body> <?php hello(); ?>
</body>
</html>
```

- **Standard form**

  <?php Your code ?>

- **Short form**

  <? Your code ?>

- **Script form**

  <script language="php"> Your code </script>

- **Echo and Print**

  Both Echo and Print are basic PHP commands to print text out. Two of them can be used in replacement. PHP programmers usually use Echo.

2.5 Web Server

The most common use of web servers is to host the web sites. Other uses are data storage or applications running. The primary function of a web server is to deliver web pages on request to clients.

2.5.1 What is web server use for?

A web server is a program that is based on the client/server model and uses the World Wide Web's Hypertext Transfer Protocol (HTTP) to serve files from web pages to web users (whose computers contain HTTP clients that forward their requests). Every computer on the Internet that contains a web site must install a web server program. There are 2 lead web server program products. The first one is **Apache** which is the most widely-installed web server product. The second one is **Microsoft's Internet Information Server** (IIS).

2.5.2 Apache [8]

Apache is a freely available web server program that is distributed under an "open source" license. Version 2.0 runs on most UNIX-based operating systems (such as Linux, Solaris, Digital UNIX). According to the web server surveys, 60% of all web sites on the Internet are using Apache (62% including Apache

derivatives). This makes Apache more widely used than all other web server products combined.

Figure 2.7 Apache's logo

2.6 MySQL [9]

Figure 2.8 MySQL logo

MySQL is a relational database management system that runs as a server providing multi-user access to a number of databases. MySQL is a popular choice of database for using in web applications. MySQL is written in C and C++. It is based on the structure query language (SQL), which is used for adding, removing, and modifying information in the database. Standard SQL commands, such as ADD, DROP, INSERT, and UPDATE can be used with MySQL.

MySQL can be used for a variety of applications, but it is most commonly found on web servers. A website that uses MySQL may include web pages that access information from a database. These pages are often referred to as "dynamic", meaning the content of each page is generated from a database as the page loads. Websites that use dynamic web pages are often referred to as database-driven websites.

Many database-driven websites that use MySQL also use a web scripting language like PHP to access information from the database.

MySQL commands can be incorporated into the PHP code, allowing part or all

of a web page to be generated from database information. Because both MySQL and PHP are both open source (meaning they are free to download and use), the PHP/MySQL combination has become a popular choice for database-driven websites.

2.7 phpMyAdmin [10]

phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the world wide web. phpMyAdmin supports a wide range of operations with MySQL. The most frequently used operations are supported by the user interface (managing databases, tables, fields, relations, indexes, users, permissions, etc), while you still have the ability to directly execute any SQL statement.

2.8 Google API

Figure 2.9 Google API

2.8.1 What is a Google API? [11]

The Google API stands for **Google Application Programmable Interface**. It is an interface that queries the Google database to help programmers in the development of their applications. It is important to remember that all of Google's APIs are only available in beta version, which means they are mostly still in their initial trial release and that there could still be a few adjustments required to some of

them. Google API's consists basically of special web services and programs and special scripts that enable Internet application developers to better find and process information on the web.

In our project, we use one API which is **Google Map API.**

2.8.2 Google Map API [12]

The Google Maps Javascript API lets you embed Google Maps in your own web pages. Version 3 of this API is especially designed to be faster and more applicable to mobile devices, as well as traditional desktop browser applications.

The API provides a number of utilities for manipulating maps (just like on the http://maps.google.com web page) and adding content to the map through a variety of services, allowing you to create robust maps applications on your website.

Figure 2.10 Example of Google API and Geolocation

2.8.3 Geolocation [13]

Geolocation refers to the identification of the geographical location of a user or computing device via a variety of data collection mechanisms. Typically, most geolocation services use network routing addresses or internal GPS devices to determine this location. Note that geolocation is a device-specific API; some browser/devices support it, while others do not (or cannot), so you cannot assume that geolocation is always possible for a web application.

2.8.4 Developing for mobile devices (Android device) [13]

The Google Maps API V3 has been designed for fast loading and suitable to work with mobile devices. Mobile devices have smaller screen sizes than typical browsers on the desktop.

2.8.5 Google Places API [14]

The Google Places API is a service that returns information about places defined within this API as establishments, geographic locations, or prominent points of interest using HTTP requests. Place requests specify locations as latitude/longitude coordinates. There are four basic place requests that are available. **Place Searches** return a list of nearby places based on a user's location. **Place Details** return more detailed information about a specific place. **Place Check-ins** allows you to report that a user has checked in to a place. Check-ins are used for place's popularity; frequent check-ins will boost a place's ranking in your application's place search results. **Place Reports** allow you to add new places to the place service, and to delete places that your application has added. In our application, we use Place Search Requests. Hence, more details of this type request will be given next.

2.8.5.1 Place Search Requests

A Place Search request is an HTTP URL of the form shown below. Its output may be either in the JSON or XML values. JSON, the recommended one, indicates output in JavaScript Object Notation (JSON).

2.8.5.2 Place Search Request: Parameters in XML platforms

Certain parameters are required to initiate a Place Search request. The parameters and their descriptions are briefly described below.

- **Location** (required): The latitude/longitude around which are used to retrieve place information. This must be specified as latitude and longitude.
- **Radius** (required): The distance (in meters) within which to return place results. The recommended best practice is to set radius based on the accuracy of the location signal as given by the location sensor.
- **Types** (optional): The results will be restricted to Places matching at least one of the specified types.
- **Language** (optional): The language code indicates in which language the results should be returned.
- **Name** (optional): A term to be matched against the names of places. The results will be restricted to those containing the passed name value.
- **Sensor** (required): It indicates whether or not the place request came from a device using a location sensor (e.g. a GPS) to determine the location sent in this request. This value must be either true or false.
- **Key** (required): Your application's API key. This key identifies your application for purposes of quota management and so that Places added from your application is immediately made available to your app.

2.8.5.3 Example in XML code for Place Search Request

```
<?xml version="1.0" encoding="UTF-8"?>
<PlaceSearchResponse>
 <status>OK</status>
 <result>
  <name>Zaaffran Restaurant - BBQ and GRILL,
  Darling Harbour</name>
 <vicinity>Darling Drive, Darling Harbour,
 Sydney</vicinity>
 <type>restaurant</type>
  <type>food</type>
 <type>establishment</type>
 <geometry>
  <location>
 <lat>-33.8712950</lat>
 <lng>151.1984770</lng>
 </location>
  </geometry>
 <icon>http://maps.gstatic.com/mapfiles/place_api/icons/
 restaurant-71.png</icon>
<reference>CpQBjAAAAJLniknTlvTxJJ5d </reference>
 <id>677679492a58049a7eae079e0890897e
 b953d79b</id>
 </result>
 </PlaceSearchResponse>
```

2.9 Global Positioning System (GPS) [15]

The Global Positioning System (GPS) is a space-based global navigation satellite system (GNSS) that provides location and time information in all weather, anywhere on or near the Earth, where there is an unobstructed line of sight to four or more GPS satellites.

2.9.1 How GPS works?

The Global Positioning system consists of three main components:

- GPS Ground control stations.
- GPS satellites.
- GPS receivers

The ground stations send control signals to the GPS satellites; the GPS satellites transmit radio signals and the GPS receivers, receive these signals and use them to calculate their position.

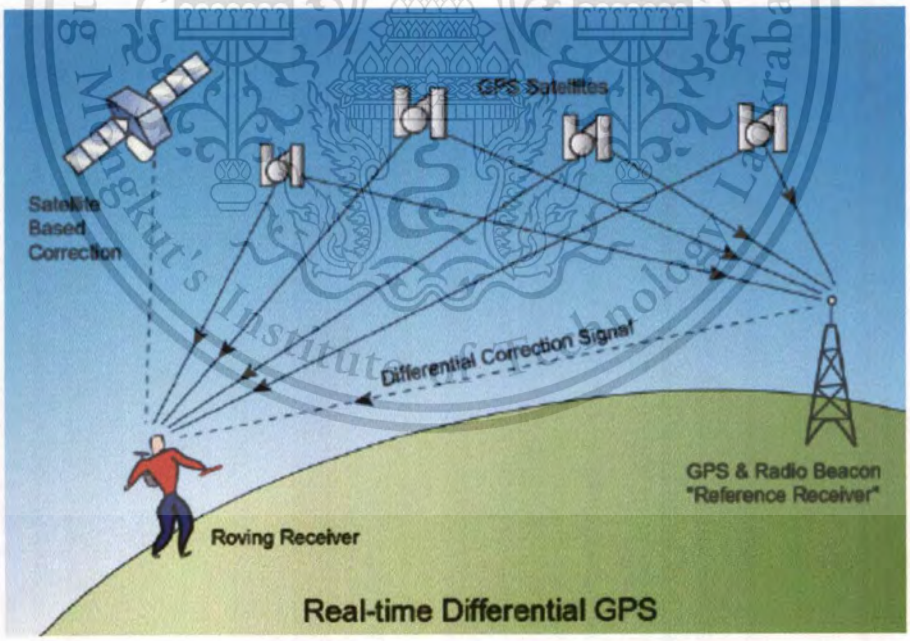


Figure 2.11 How GPS works

2.10 Location-based service (LBS) [16]

A Location-Based Service (LBS) is an information service, which can be accessible by mobile devices through the mobile network. It is used for the

geographical position of the mobile service. LBS includes several services such as a service to identify a location of a person. LBS also includes parcel tracking and vehicle tracking services. Moreover, it also includes mobile commerce (M-Commerce) such as advertising directly at customers based on their current locations.

2.10.1 LBS applications

Some examples of Location-Based Services are listed below:

- recommending social events in a city services
- requesting the nearest business or service
- turn by turn navigation to any address services
- locating people on a map displayed on the mobile phone services
- receiving alerts, such as notification of a sale on gas or warning of traffic jam services
- location-based mobile advertising services
- asset recovery services which may combine with active RF (Radio Frequency), for example, finding assets in containers where GPS wouldn't work.

2.11 JSON format [17]

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family 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, which in various languages, this is known as an object, record, structure, dictionary, hash table, keyed list, or associative array and an order list of values, which in most languages, this is known as an array, vector, list, or sequence. These structures are universal data structures. All modern programming languages support them in one form or another. It

makes sense that a data format that is interchangeable between programming languages is based on these structures.

In JSON an object is an unordered set of name/value pairs which begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).

An array is an order of values collection which begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma). A string is a sequence of zero or more unicode characters, wrapped in double quotes, using backslash escapes. A single character string represents a character. String in JSON is similar to JAVA or C String. Likewise, JSON's numbers are similar to a C or Java numbers, except that the octal and hexadecimal formats are not used.


```
{
  - data: [
      - {
            id: "1",
            name: "ครัวนารี อาหารตามสั่ง",
            lat: "13.720841",
            long: "100.775485",
            pic: "http://kinarai.no-ip.org/kinarai/kinarai_php/imglogo/KinArai.jpg",
            tellnum: "",
            location: "Latkrabang ลาดกระบัง"
        },
      - {
            id: "2",
            name: "พิกุลแก้วฟาร์ม เมี่ยงปลาเผา",
            lat: "13.722041",
            long: "100.776808",
            pic: "http://kinarai.no-ip.org/kinarai/kinarai_php/imglogo/KinArai.jpg",
            tellnum: "",
            location: "Latkrabang"
        },
      - {
            id: "3",
            name: "ข้าวต้มโต้รุ่ง เจ๊ภีร์",
            lat: "13.722045",
            long: "100.776735",
            pic: "http://kinarai.no-ip.org/kinarai/kinarai_php/imglogo/KinArai.jpg",
            tellnum: "",
            location: "Latkrabang"
        },
      - {
            id: "5",
            name: "โรงอาหารคณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง",
            lat: "13.729067",
            long: "100.778403",
            pic: "http://kinarai.no-ip.org/kinarai/kinarai_php/imglogo/KinArai.jpg",
            tellnum: "",
            location: "Latkrabang"
        },
```


Figure 2.12 JSON format

Chapter 3

System Analysis and Design

3.1 System Analysis

In this chapter, we will give the analysis of our system and its design. Basically, there are two main functional requirements in our application. The first requirement is a function that provides a map of the restaurants which their locations are near the user. The second requirement is a function to provide the lists of the restaurants' promotion information.

We will use case diagram, Sequence diagram and Activity diagram, which are based on UML structure as our tool to describe our system.

3.1.1 Use case diagrams of our project

We will begin by using the Use case diagram to describe our system. Use case diagram is a simple diagram which shows an overview of functional requirements of the system. That is, what the system can provide to its user. For our system, there are two functions: the NearBy function and the Promotion Function.

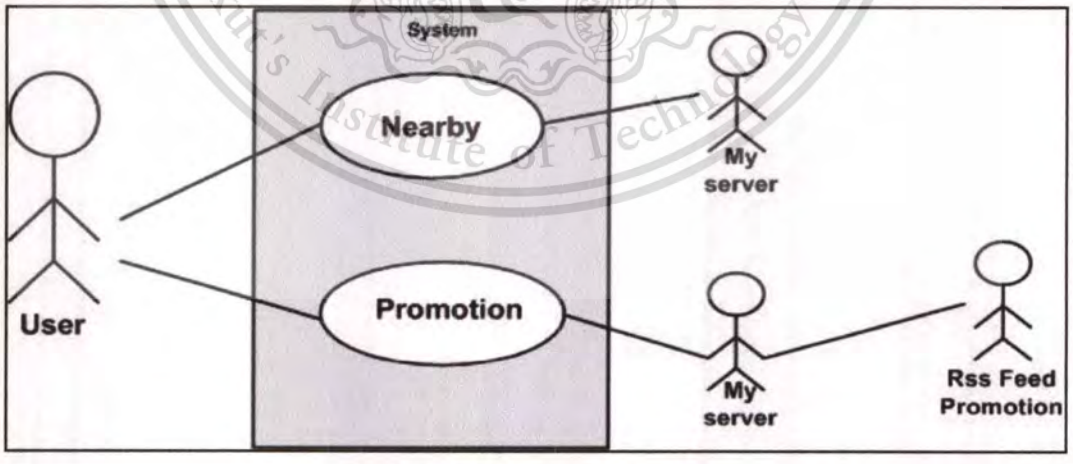


Figure 3.1 Use case diagram

3.1.1.1 Use Case Name: Nearby Function

The Nearby function uses the latitude and longitude (obtaining from the GPS device on the mobile phone) to calculate the current location between

the restaurants in the database around users. The locations of each restaurant are stored into database with latitude and longitude of them. The system will download the list of restaurant in database and then calculate to show the results.

3.1.1.2 Use Case Name: Promotion Function

The Promotion function is the function to provide the restaurants' promotion information to user. Promotion information is fetched from popular web promotions.

We note that in our project we use 5 web promotions, which are www.promotiontoyou.com, www.promotions.co.th, www.proded.com, www.welove pro.com, www.goodpromoguide.com. We choose these web promotions because they have RSS feed services that allow everyone to use them for free.

3.1.2 Activity Diagram of our project

Activity diagram is the diagram for showing the work flow of the system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. We divide the participant into 3 parts: the user (who owns the mobile phone), the system (the system part which interacts with the user directly, which it functions as the front end of the system) and the server (performs the system request and store some information functions as the backend of the system.).

For our project, there are 4 activity diagrams. The Activity Diagram of Front-End Promotion and the Activity Diagram of Back-End promotion are for web promotion function. The Activity Diagram of Nearby is for the location of the near restaurants. The last one is the Activity Diagram of inserting new restaurant to the system.

3.1.2.1 Activity Diagram of Front-End Promotion Function

Figure 3.2 shows the promotion function at the front end. First, when the user touches menu promotion on his mobile, the system displays a list of

restaurants to the user. Then, the user will select the restaurants that he is interested in its promotion. After the system receives the promotion request from the user, the system will query the database from the back-end part. The back-end will provide the data that filtered and sends back to the system in XML format.

When the system receives the data from the server, the system will display the promotion to the user.

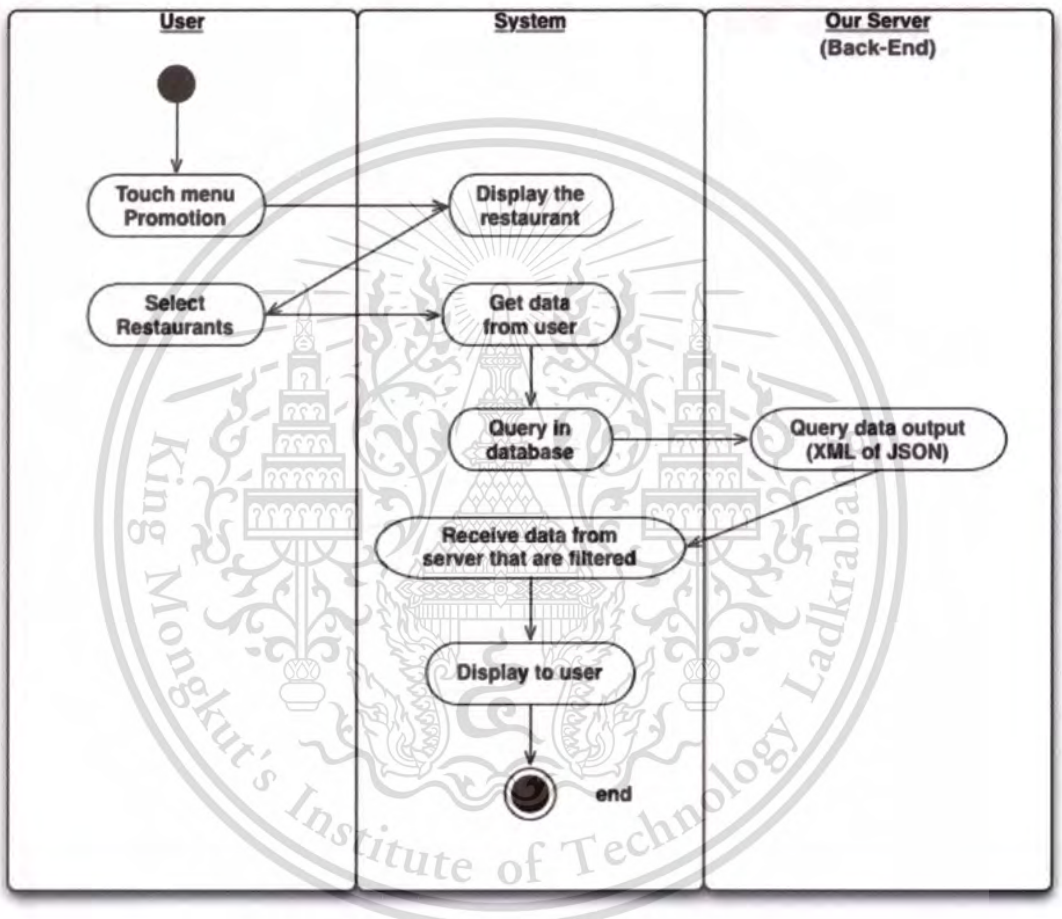


Figure 3.2 Activity diagram of front-end promotion function

3.1.2.2 Activity Diagram of Back-End Promotion Function

Figure 3.3 shows the activity diagram of the back-end promotion function. It includes the database, web server and web promotion sites . At first, our server uses PHP to get the data in XML format from the web promotion sites. Then, the site sends back the data to our server. Next, the server filters the keywords by using PHP. The PHP will check the keyword using tags of the XML file. The

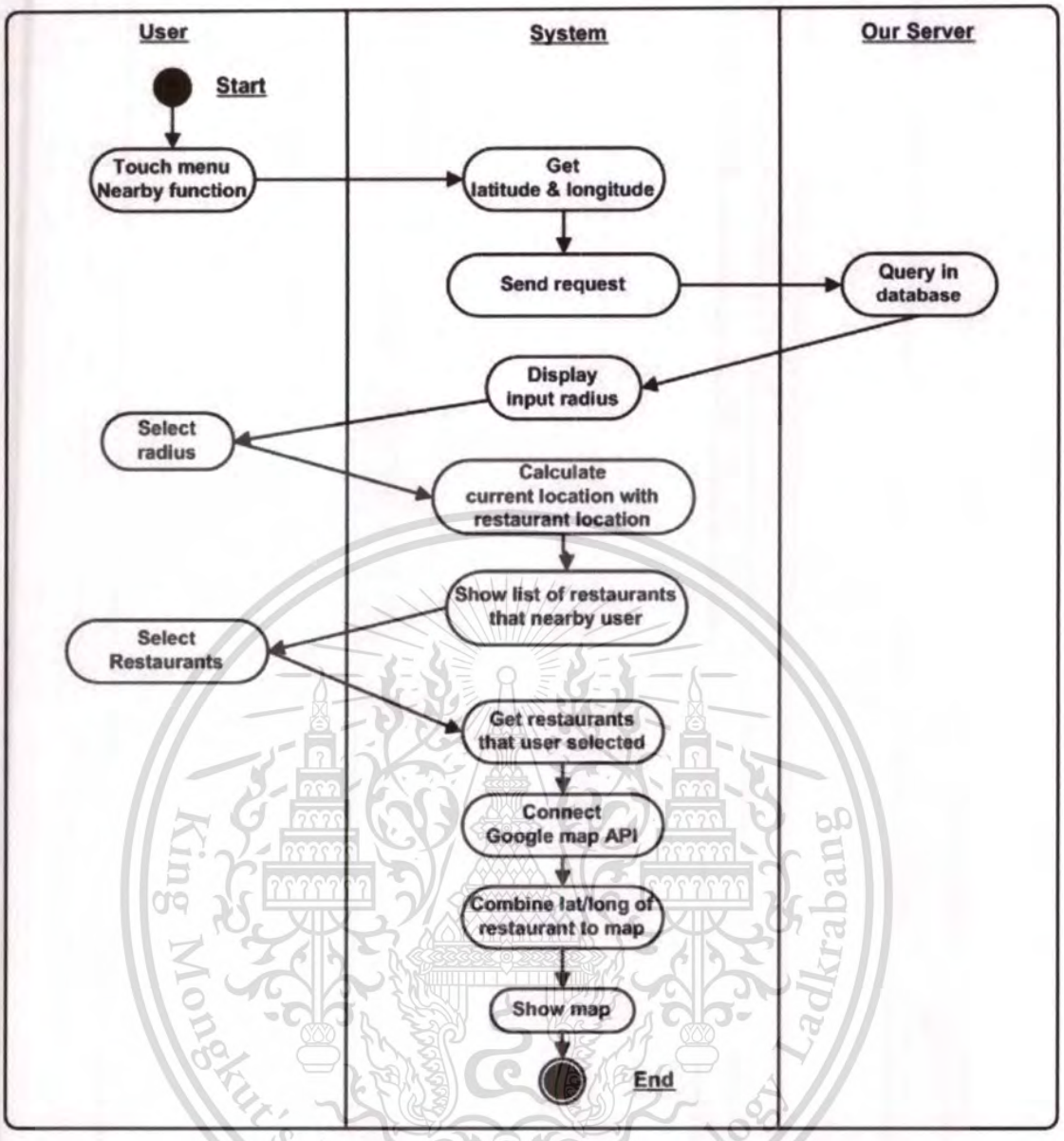


Figure 3.4 Activity diagram of front-end nearby function

3.1.2.3 Activity Diagram of Front-End Nearby Function

Figure 3.4 shows the activity diagram of front-end nearby function. First user selects the nearby menu on the mobile. The system will get latitude and longitude from phone's device. Then, the system will send a request to the database for all restaurants from the server. After the server finish querying, it will send back the restaurant information to the system. The system will temporary save this information to the phone. Then, it will ask the user to input his required radius. Next, the system will use the radius that user have chosen together with the current location of the user to find the restaurants located within the radius distance of the

phone. The results will be displayed on the screen as a list of the restaurant names and their distances from the user's location. When the user needs to know the route of any restaurant, he simply touches at that restaurant name. The system will then display the map and the direction to the restaurant.

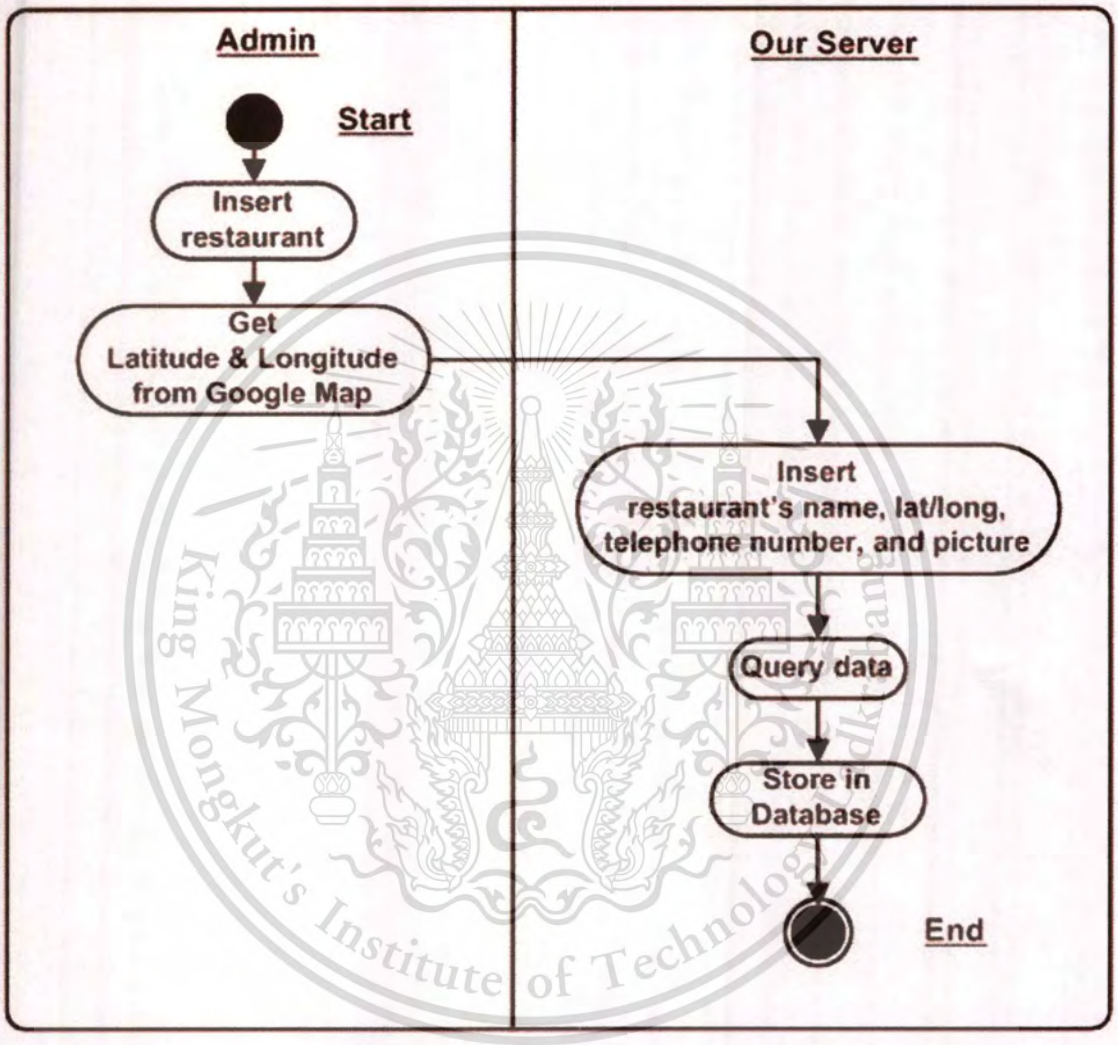


Figure 3.5 Activity diagram of restaurant insertion function on the back-end

3.1.2.4 Activity Diagram of Restaurant Insertion Function on the Back-End

The restriction is that we allow only the administrator of the system to add the new restaurants. Figure 3.5 shows the activity diagram of the function that only allows the administrator to insert new restaurant to the server on the

back-end site. The administrator must have this information: location of each restaurant (that gets from Google map), telephone number, and picture of restaurant.

3.1.3 Sequence Diagram [18]

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

We further use the sequence diagrams to explain our system in more details.

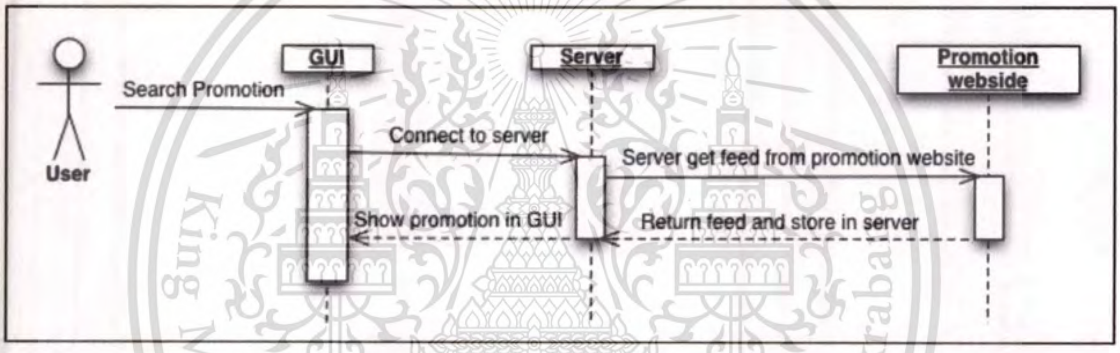


Figure 3.6 Sequence diagram of promotion function

3.1.3.1 Sequence Diagram of Promotion Function

Figure 3.6 shows the sequence diagram of the promotion function. The user searches for some restaurant promotions using our application through its GUI. The GUI will send his request to the server for the promotions he wants. The server then returns the required information. It is noted that the server has functions to get promotions from the web promotion sites which will be done automatically using RSS feed method.

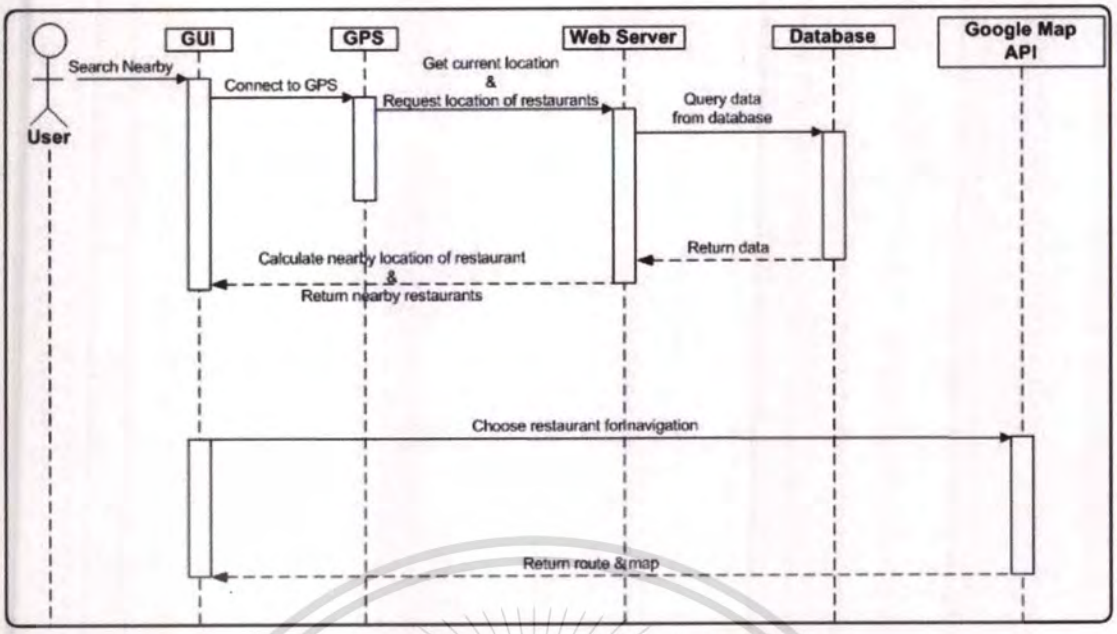


Figure 3.7 Sequence diagram of nearby function

3.1.3.2 Sequence Diagram of Nearby Function

Figure 3.7 shows the sequence diagram of nearby function. This diagram extends the diagram in Figure 3.4 to show details of front-end components and their functions over time. When the user wants to search some restaurants, that are located nearby his current location, he does that through the GUI of our application. The system will automatically connect to the GPS function on the user's mobile to get its current geographical location and then the system will request data from the web server. The web server will query data from the database and respond to the system in JSON format. After that, the system will use data from the web server to calculate and find the nearby restaurants according to the radius that the user inputs. The result (nearby restaurants will return to display on GUI. If the user wants to know the direction to go to any particular restaurant, they simply choose the restaurant on the list. Then, the system will combine latitude and longitude of each restaurant with Google Map API which consists of a map and the route to the destination.

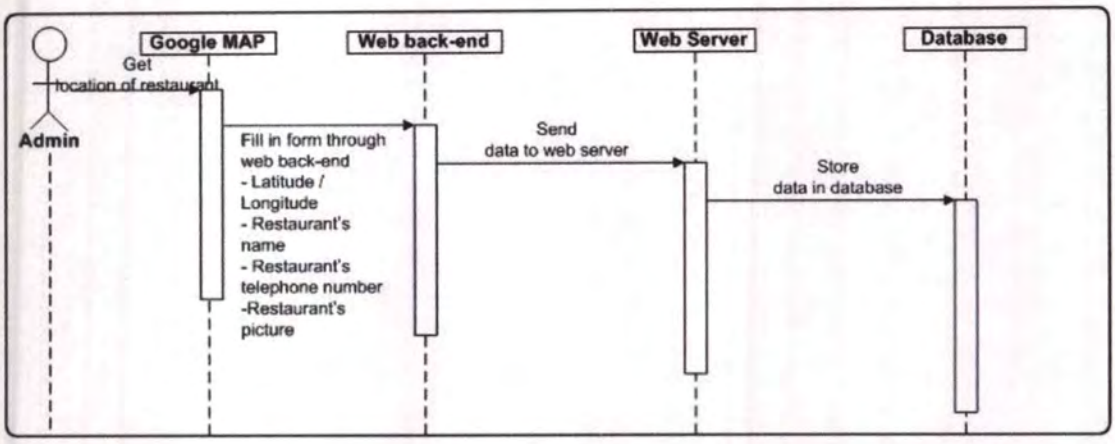


Figure 3.8 Sequence diagram of restaurant insertion on the web back-end

3.1.3.3 Sequence Diagram of Restaurant Insertion on the web back- end

In Figure 3.8, the administrator can add the new restaurant through the web back-end. First, he has to obtain the restaurant location. Then, he fills in all required information such as the restaurant latitude and longitude, the restaurant's telephone number and its picture. This information will then submit to the database server. Note that the location of the restaurants can be obtained from other trusted sources or the administrator can find it himself through the Google Map. To illustrate, the administrator can open his Google Map and marks the restaurant's location on the map for its position's latitude and longitude or he can use his position on the map (if he is at that restaurant).

3.1.4 Programming process diagram of front-end application

3.1.4.1 The main programming process diagram

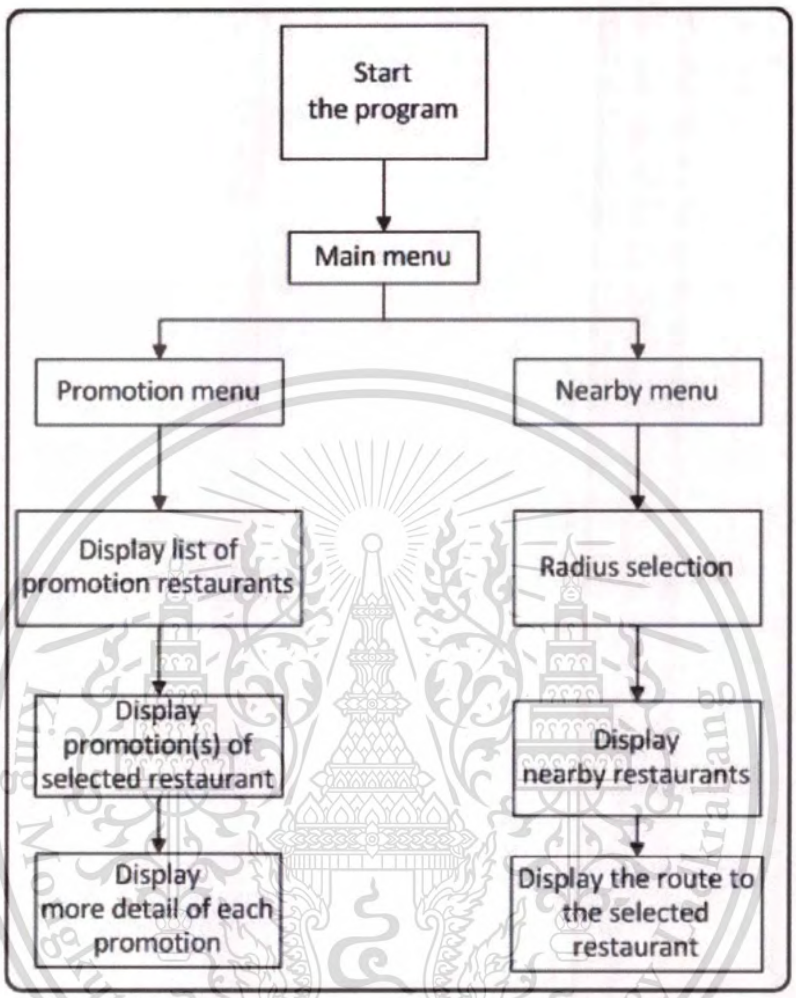


Figure 3.9 Main program's process diagram

3.1.4.2 The process of promotion function diagram

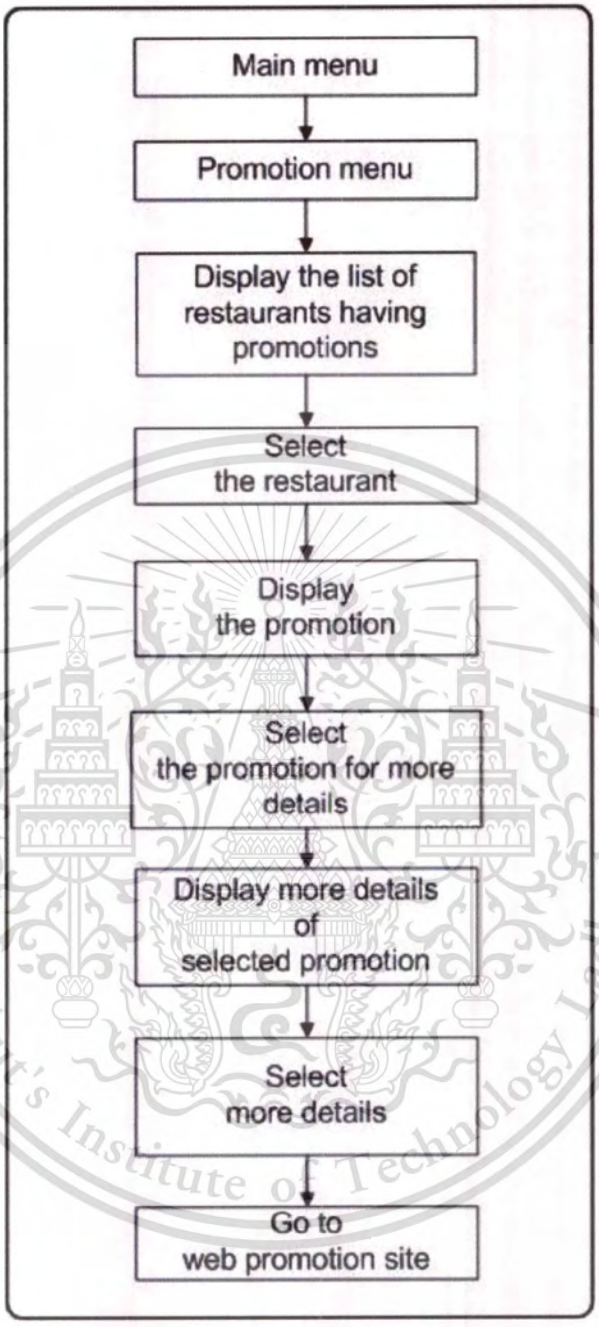


Figure 3.10 Promotion function process's diagram

3.1.4.3 The process of nearby function diagram

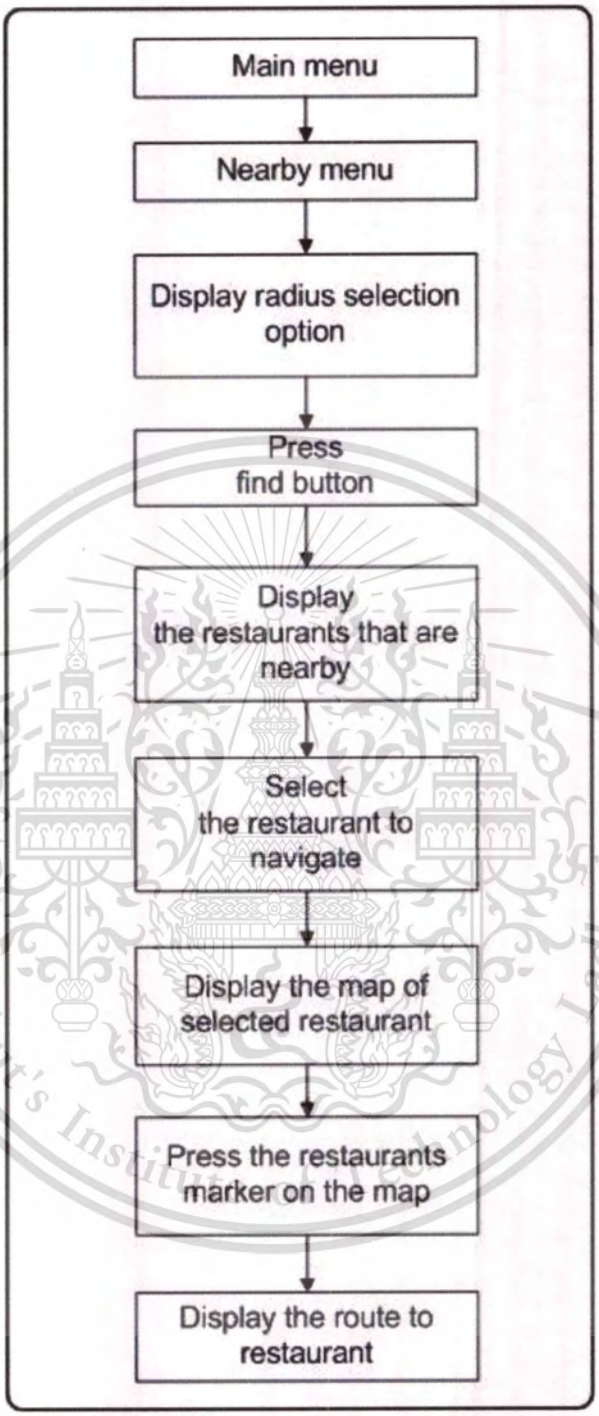


Figure 3.11 Nearby function process's diagram

3.1.5 Programming process diagrams at the back-end

3.1.5.1 Back-end Process: Adding promotion keywords diagram

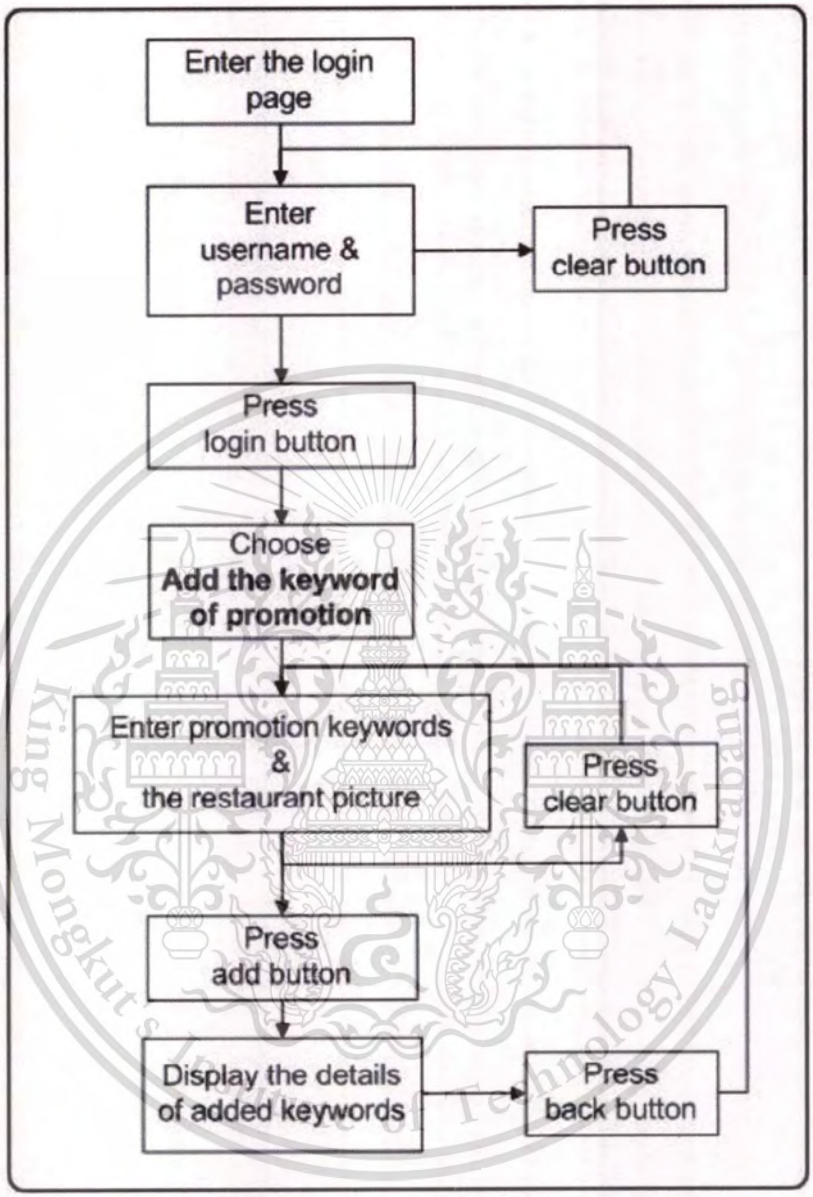


Figure 3.12 Adding promotion keywords diagram

Keywords are used to extract the promotion information from web promotion sites. The system will receive feed that contain all information from the web promotion sites but it will only extract elements that contain keywords. This is because web promotion sites provide not only restaurant promotions but also others such as clothes and hotels. Figure 3.12 shows the process to add the keywords required into the **restaurant table.** In our system, we uses two keywords for each restaurant.

The first keyword is either the abbreviation or some part of the restaurant names such as Starbuck for Starbucks for Starbuck coffee. The second keyword is the restaurant name in Thai.

3.1.5.2 Back-end Process: restaurant insertion diagram

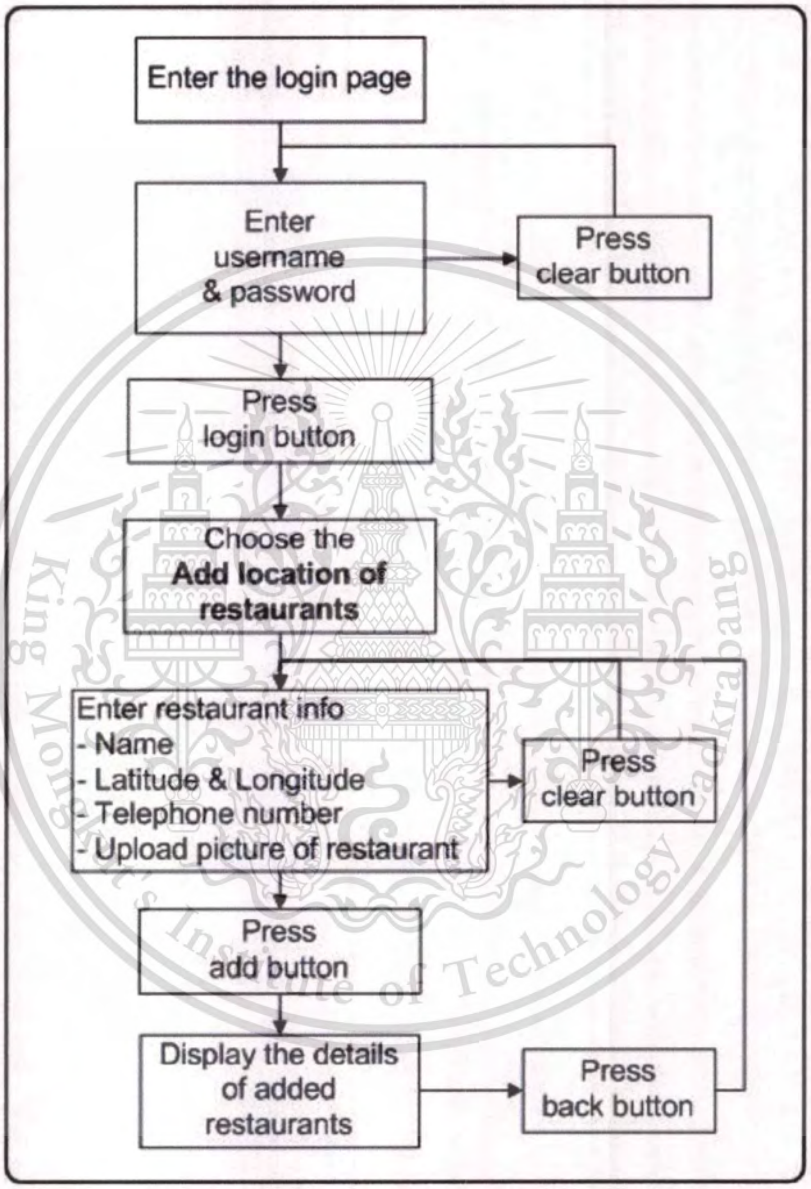


Figure 3.13 Restaurant insertion diagram

The restaurant location information that we added are obtained by the methods described in section 4.1.5 (see section 4.1.5 for how to obtain these values).

3.2 Design

As emphasized above, our application consists of two main functions: finding restaurants' promotion and searching for nearby restaurants. Both functions interact with the web service for their requests. The design of our system is based on the client-server architecture. The servers are very important components because they are centers for storing and providing all information to our application on the android phone. Hence, the server should be able to support all requests.

As a result, there are three main components in our system. Two of them deal with tasks at the front-end which are searching for the nearby restaurants and providing restaurant promotions. The third component consists of the web server and the database server which are responsible for functions in the back end part.

The databases in the back end part store four tables: the restaurant table, the promotion table, the member table and the restaurant location table respectively.

The **restaurant table** as shown in Table 3.1 stores restaurant information including the restaurant id, two keywords, the restaurant name and the restaurant's picture. This table is used for filtering the promotions from web promotion sites.

Table 3.1 Restaurant table

Attribute Name	Data Type	Collation	Reference	Key
id_restaurants	int(5)		auto_increment	PK
k_resEng	varchar(20)	utf8	unique	
k_resThai	mediumtext	utf8		
nm_restaurant	varchar(20)	utf8		
img_logo	varchar(20)	utf8		

The **promotion table** as shown in Table 3.2 stores promotion information. Each record stores information of each promotion that our system has. It includes the restaurant id (what restaurant that owns this promotion), the promotion id, the title of

the promotion, the description of the promotion, the image of the promotion, the link to web promotion sites that the system acquired the promotion from and the date that this promotion is published on the web promotion site.

Table 3.2 Promotion table

Attribute Name	Data Type	Collation	Reference	Key
id_restaurants	int(5)			FK
Id_promotions	int(10)	utf8	auto_increment	PK
title	varchar(333)	utf8	unique	
description	longtext	utf8		
image	longtext	utf8		
link	longtext	utf8		
pubdate	varchar(35)	utf8		

The **member table** as shown in Table 3.3 stores the username and password of the administrators who can add restaurants and their keywords and restaurant's location to the database. It consists of the user id, the user name and his password. The maximum number of the user is 99 (limited by its data type)

Table 3.3 Member table

Attribute Name	Data Type	Collation	Reference	Key
id_users	int(2)		auto_increment	PK
user	varchar(10)	utf8	unique	
pass	mediumtext	utf8		

Table 3.4 Restaurant Location table

Attribute Name	Data Type	Collation	Reference	Key
id	int(10)		auto_increment	PK
mmrestaurants	text	utf8		
lat1	double			
long1	double			
pic	text	utf8		
tellnum	tinytext	utf8		
location	text	utf8		

The **restaurant location table** as shown in Table 3.4 stores the restaurants' location information. It includes the id, the name of the restaurants, its latitude, its longitude, URL of the restaurant's picture on our server and the restaurant's telephone number .

For our database query and insertion, they are done through the web server rather than allowing the system on android phone to connect directly to the database.

For implementation, we decide to use MySQL for database and Apache for web server components. We will use PHP to communicate with the MySQL.

3.2.1 Promotion Function

To make the system more automate and does not depend on an administrator to input the promotion information, we obtain the promotion information from popular web promotions instead. Hence, we are only interested in the web promotion sites that provide the RSS feed service.

We use information from five web promotion sites (to prove that our design and implementation are correct. Then, to add more of them can be done easily later). By using the keywords "Promotion " and "โปรโมชั่น" to search on Google search engine, we took the first five popular websites that were appeared on the search

list. If any chosen website did not have the RSS feed service, we skipped that website for the next one on the search list. Table 3.5 shows the top five websites of each keyword that we used. We found that www.shopping.com did not provide RSS feed service so we did not use it. Table 3.6 shows the top five web site promotions that provide promotion through Rss feed channel.

Table 3.5 Top 5 of search keywords

	www.google.co.th (October 11, 2011)	
	Keyword "Promotion"	**Keyword "โปรโมชั่น"**
1	www.promotiontoyou.com	www.promotiontoyou.com
2	www.promotions.co.th	www.promotions.co.th
3	www.goodpromoguide.com	www.proded.com
4	www.welovepro.com	www.welovepro.com
5	www.shoppening.com	www.goodpromoguide.com

Table 3.6 Result of our selected web promotion sites

Our Selected web promotion sites
www.promotiontoyou.com
www.promotions.co.th
www.proded.com
www.welovepro.com
www.goodpromoguide.com

3.2.1.1 XML structure of promotion websites

The benefit of the RSS feed service is that it provides the feed information in XML structure form which we can filter for our interest information through the tags.

These are tags of XML structure, that we are interested in

```
<item>
    <title>…</title>
    <description>…</description>
    <image>…<image>
    <link>…</link>
    <pubDate>…</pubDate>
</item>
```

Figure 3.14 to Figure 3.23 show the index of the selected web pages and their XML structure data that they provided to the RSS feed functions. The red highlight frames represent our interested information which each site provided (ones that we can filter and store into our server). Once we obtain the promotion information, we store this information into the promotion table in our database.

Figure 3.14 Index page of promotiontoyou.com

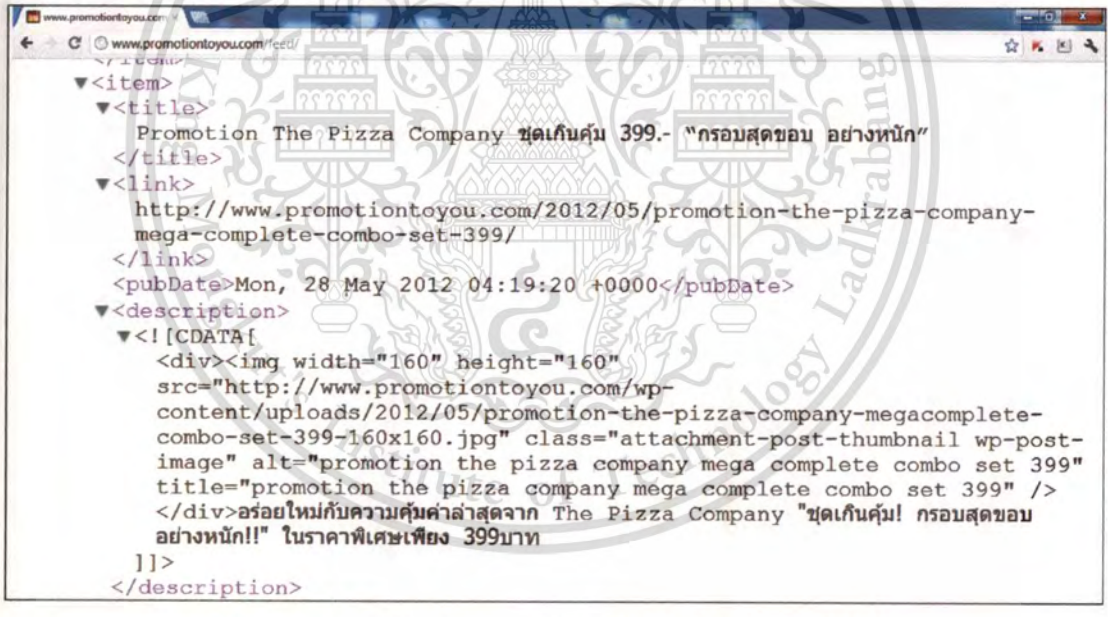


Figure 3.15 XML structure of promotiontoyou.com

Figure 3.16 Index page of promotions.co.th

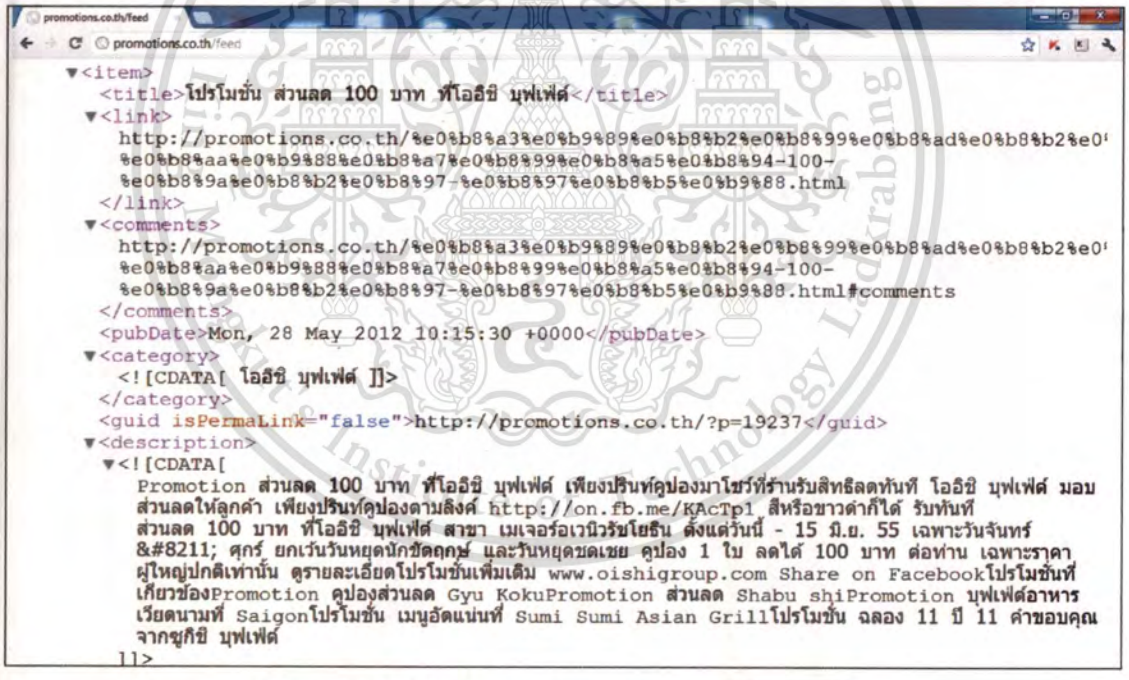


Figure 3.17 XML structure of promotions.co.th

Figure 3.18 Index page of proded.com

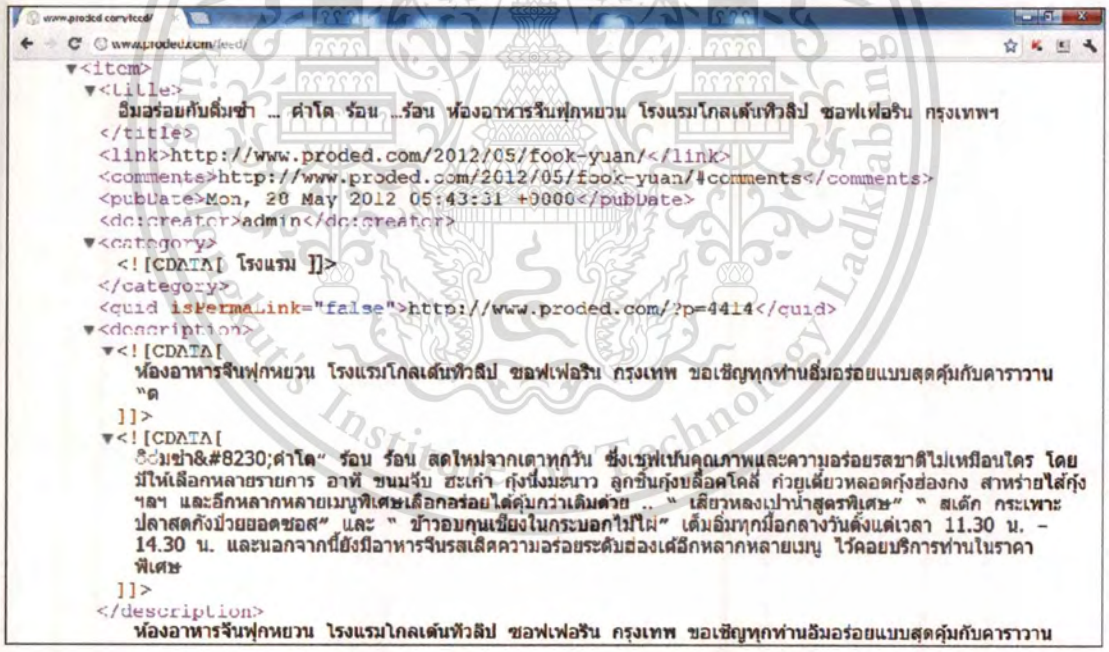


Figure 3.19 XML structure of proded.com

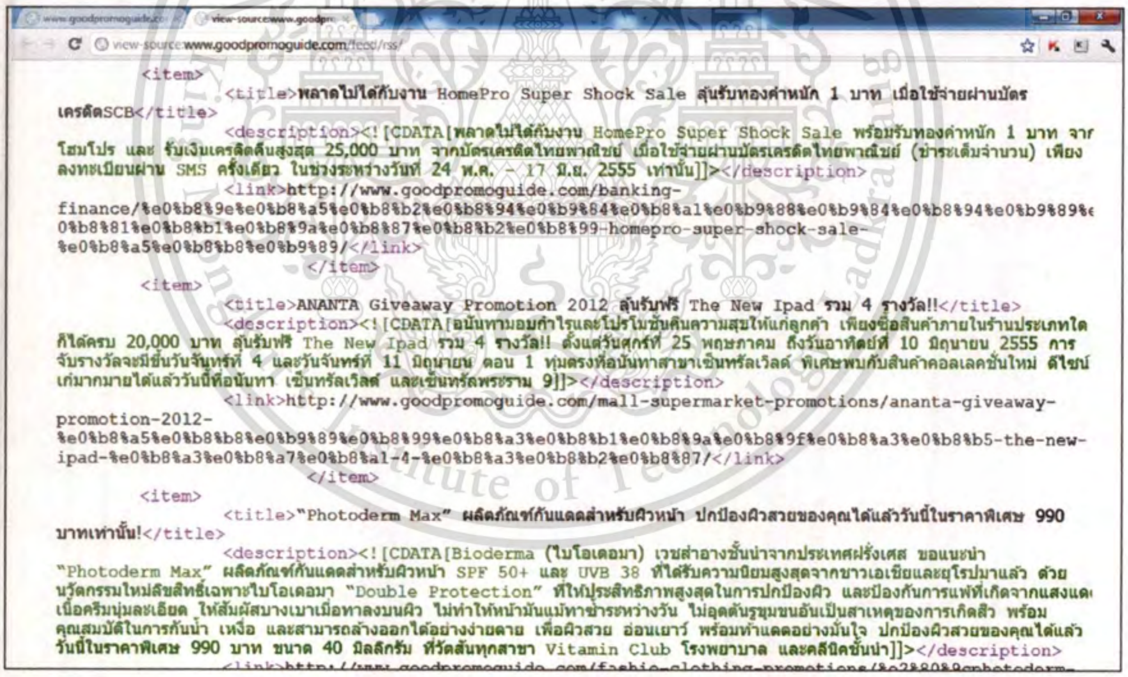
Figure 3.20 Index page of welovepro.com

```
                </item>
                        <item>
                <title>เครื่องดื่ม อิชิตัน แลกซื้อสุดคุ้ม 2 ขวด 25 บาท ที่ เซเว่นอีเลฟเว่น</title>
                <link>http://www.welovepro.com/ichitan-2-for-25-baht/</link>
                <comments>http://www.welovepro.com/ichitan-2-for-25-
baht/#comments</comments>
                <pubDate>Mon, 28 May 2012 10:03:49 +0000</pubDate>
                <dc:creator>reborn7</dc:creator>
                                <category><![CDATA[อาหาร-เครื่องดื่ม]]></category>
                <category><![CDATA[โปรโมชั่นแนะนำ]]></category>
                <category><![CDATA[อิชิตัน]]></category>
                <category><![CDATA[เซเว่น อีเลฟเว่น]]></category>
                <category><![CDATA[แลกซื้อ]]></category>

                <guid isPermaLink="false">http://www.welovepro.com/?p=57144</guid>
                <description><![CDATA[<p><img src="http://www.welovepro.com/wp-
content/uploads/2012/05/ichitan-2-for-25-baht.jpg"/></p>โปรโมชั่น ลดราคา sale ชิ่งโชค ส่วนลด
discount ลดกระหน่ำ]]></description>
                <wfw:commentRss>http://www.welovepro.com/ichitan-2-for-25-
baht/feed/</wfw:commentRss>
                <slash:comments>0</slash:comments>
                </item>
                        <item>
                <title>ฉลองเปิดตัวครั้งแรกของแบรนด์ BASIC HOUSE รับส่วนลดพิเศษ 20%</title>
                <link>http://www.welovepro.com/basic-house-special-discount-20-
off/</link>
                <comments>http://www.welovepro.com/basic-house-special-discount-20-
off/#comments</comments>
```

Figure 3.21 XML structure of welovepro.com

Figure 3.22 Index page of goodpromoguide.com



```
view-source:www.goodpromoguide.com/feed/rss/
        <item>
                <title>พลาดไม่ได้กับงาน HomePro Super Shock Sale ลุ้นรับทองคำหนัก 1 บาท เมื่อใช้จ่ายผ่านบัตร
เครดิตSCB</title>
                <description><![CDATA[พลาดไม่ได้กับงาน HomePro Super Shock Sale พร้อมรับทองคำหนัก 1 บาท จาก
โฮมโปร และ รับเงินเครดิตคืนสูงสุด 25,000 บาท จากบัตรเครดิตไทยพาณิชย์ เมื่อใช้จ่ายผ่านบัตรเครดิตไทยพาณิชย์ (ชำระเต็มจำนวน) เพียง
ลงทะเบียนผ่าน SMS ครั้งเดียว ในช่วงระหว่างวันที่ 24 พ.ค. - 17 มิ.ย. 2555 เท่านั้น]]></description>
                <link>http://www.goodpromoguide.com/banking-
finance/%e0%b8%9e%e0%b8%a5%e0%b8%b2%e0%b8%94%e0%b9%84%e0%b8%a1%e0%b9%88%e0%b9%84%e0%b8%94%e0%b9%89%e
0%b8%81%e0%b8%b1%e0%b8%9a%e0%b8%87%e0%b8%b2%e0%b8%99-homepro-super-shock-sale-
%e0%b8%a5%e0%b8%b8%e0%b9%89/</link>
                        </item>
        <item>
                <title>ANANTA Giveaway Promotion 2012 ลุ้นรับฟรี The New Ipad รวม 4 รางวัล!!</title>
                <description><![CDATA[อนันทามอบกำไรและโปรโมชั่นคืนความสุขให้แก่ลูกค้า เพียงซื้อสินค้าภายในร้านประเภทใด
ก็ได้ครบ 20,000 บาท ลุ้นรับฟรี The New Ipad รวม 4 รางวัล!! ตั้งแต่วันศุกร์ที่ 25 พฤษภาคม ถึงวันอาทิตย์ที่ 10 มิถุนายน 2555 การ
จับรางวัลจะมีขึ้นวันจันทร์ที่ 4 และวันจันทร์ที่ 11 มิถุนายน ตอน 1 ทุ่มตรงที่อนันทาสาขาเซ็นทรัลเวิลด์ พิเศษพบกับสินค้าคอลเลคชั่นใหม่ ดีไซน์
เก๋มากมายได้แล้ววันนี้ที่อนันทา เซ็นทรัลเวิลด์ และเซ็นทรัลพระราม 9]]></description>
                <link>http://www.goodpromoguide.com/mall-supermarket-promotions/ananta-giveaway-
promotion-2012-
%e0%b8%a5%e0%b8%b8%e0%b9%89%e0%b8%99%e0%b8%a3%e0%b8%b1%e0%b8%9a%e0%b8%9f%e0%b8%a3%e0%b8%b5-the-new-
ipad-%e0%b8%a3%e0%b8%a7%e0%b8%a1-4-%e0%b8%a3%e0%b8%b2%e0%b8%87/</link>
                        </item>
        <item>
                <title>"Photoderm Max" ผลิตภัณฑ์กันแดดสำหรับผิวหน้า ปกป้องผิวสวยของคุณได้แล้ววันนี้ในราคาพิเศษ 990
บาทเท่านั้น!</title>
                <description><![CDATA[Bioderma (ไบโอเดอมา) เวชสำอางชั้นนำจากประเทศฝรั่งเศส ขอแนะนำ
"Photoderm Max" ผลิตภัณฑ์กันแดดสำหรับผิวหน้า SPF 50+ และ UVB 38 ที่ได้รับความนิยมสูงสุดจากชาวเอเชียและยุโรปมาแล้ว ด้วย
นวัตกรรมใหม่ลิขสิทธิ์เฉพาะไบโอเดอมา "Double Protection" ที่ให้ประสิทธิภาพสูงสุดในการปกป้องผิว และป้องกันการแพ้ที่เกิดจากแสงแด
เนื้อครีมนุ่มละเอียด ให้สัมผัสบางเบาเมื่อทาลงบนผิว ไม่ทำให้หน้ามันแม้ทาซ้ำระหว่างวัน ไม่อุดตันรูขุมขนอันเป็นสาเหตุของการเกิดสิว พร้อม
คุณสมบัติในการกันน้ำ เหงื่อ และสามารถล้างออกได้อย่างง่ายดาย เพื่อผิวสวย อ่อนเยาว์ พร้อมท้าแดดอย่างมั่นใจ ปกป้องผิวสวยของคุณได้แล้ว
วันนี้ในราคาพิเศษ 990 บาท ขนาด 40 มิลลิกรัม ที่วัตสันทุกสาขา Vitamin Club โรงพยาบาล และคลีนิคชั้นนำ]]></description>
```


Figure 3.23 XML structure of goodpromoguide.com

3.2.1.2 Web back-end's GUI: Adding promotion keywords

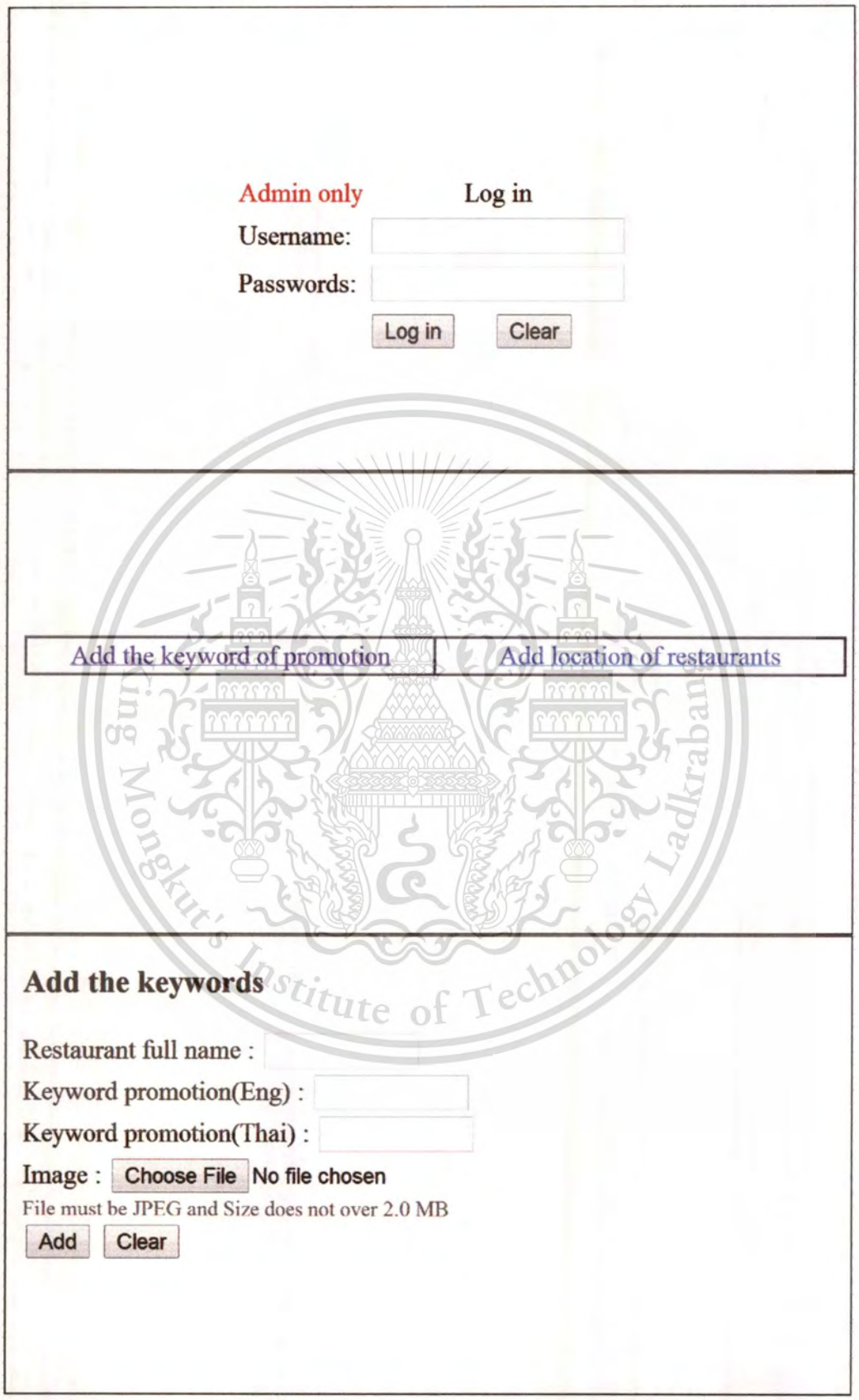


Figure 3.24 Web back-end's GUI: Adding promotion keywords

3.2.1.3 Front-end GUI: promotion function

There are five pages as shown from Figure 3.25 to Figure 3.29.

Figure 3.25 Main menu of promotion function

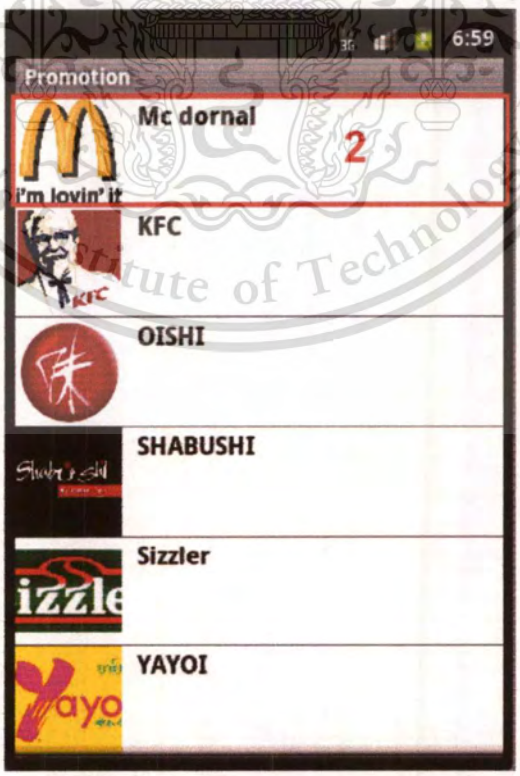


Figure 3.26 Select the McDonald

Figure 3.27 Select the promotion for more details

Figure 3.28 Press more details button to go to the web promotion site

Figure 3.29 Original web promotion site

3.2.2 Nearby Function

This feature provides the names and locations of the restaurants located near the user. It first uses the location service in android (this android phone must have GPS device on it) to give the current location of the user. The location is given as the latitude and longitude of the user's position. This location will be combined with the user's input for the area size. Then, this area will be sent to query the database for the nearby restaurants in the area.

The result will be displayed as all restaurants will be showed by list view in android device. User will choose from list view and the map obtained from the Google Map through the Google API will appear.

This feature also provides the place of restaurants that located around the user. This can be done by using location services in android to identify the current location of user. Then, we use the current location to calculate the place of nearby restaurants in the database.

3.2.2.1 Web back-end's GUI: new restaurant insertion

This feature allows the administrator of the system to add new restaurant into database. As previously described in the analysis section, it can be done through the web back-end. Latitude and longitude of restaurants will be obtain from Google MAP API.

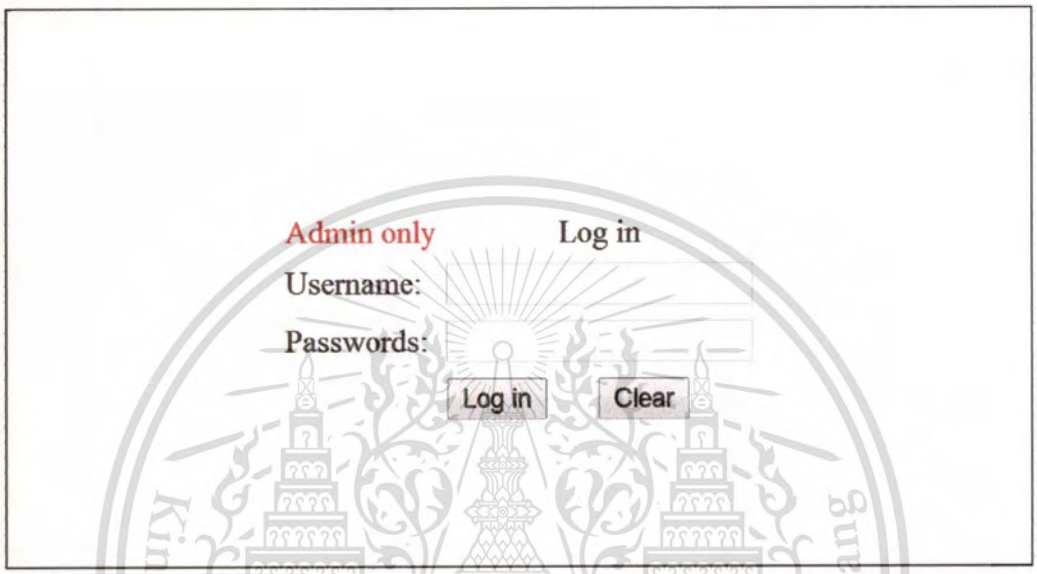


Figure 3.30 Log in page

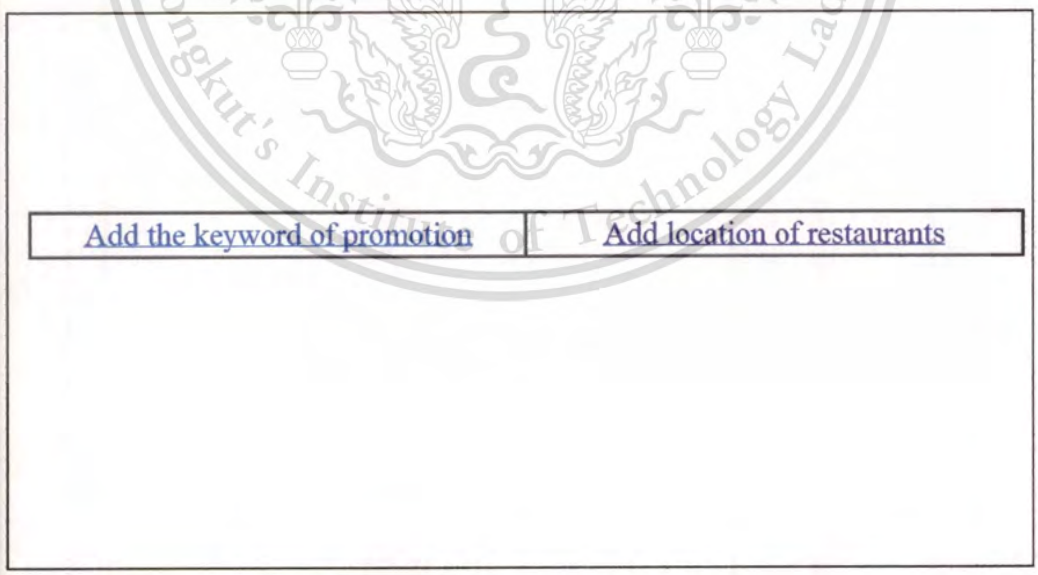


Figure 3.31 Select *add location of restaurants*

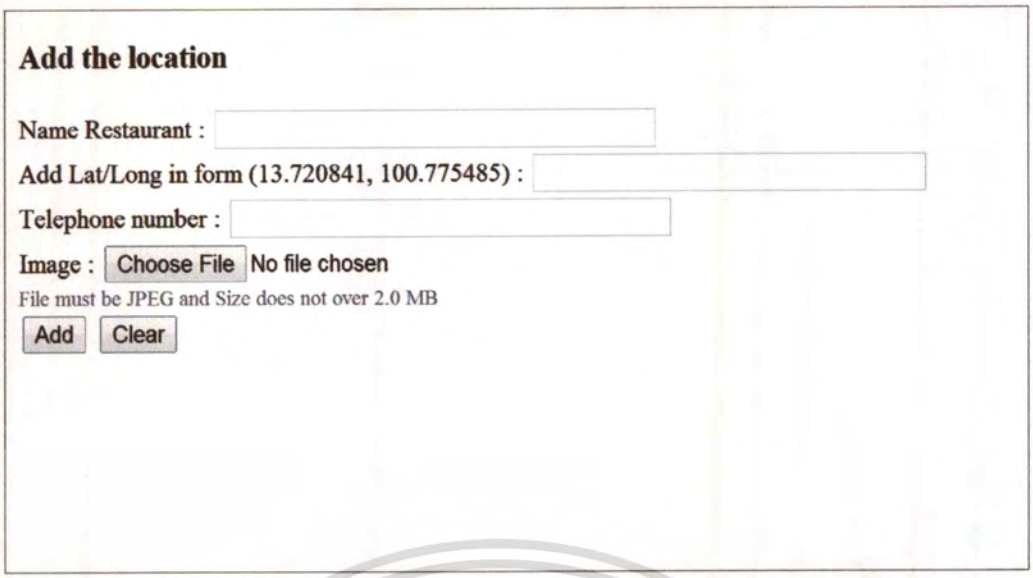


Figure 3.32 Fill in form to add restaurant location information

3.2.2.2 Front-end GUI: nearby function

There are four pages as shown from Figure 3.33 to Figure 3.36

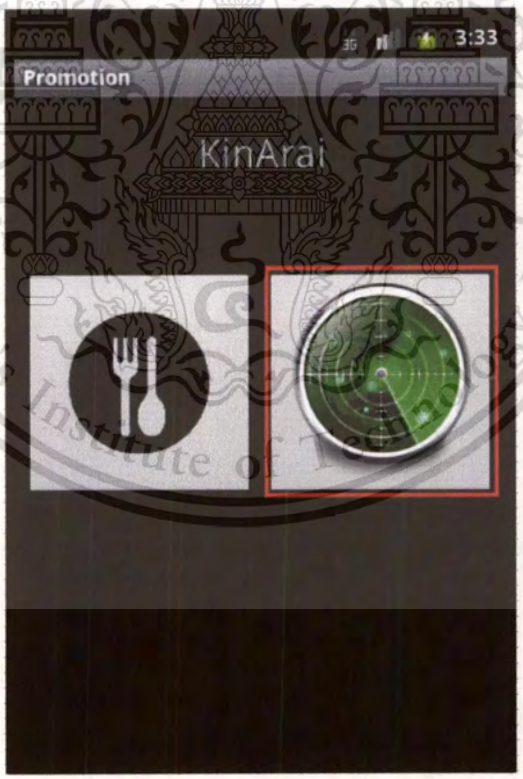


Figure 3.33 Main menu of nearby function

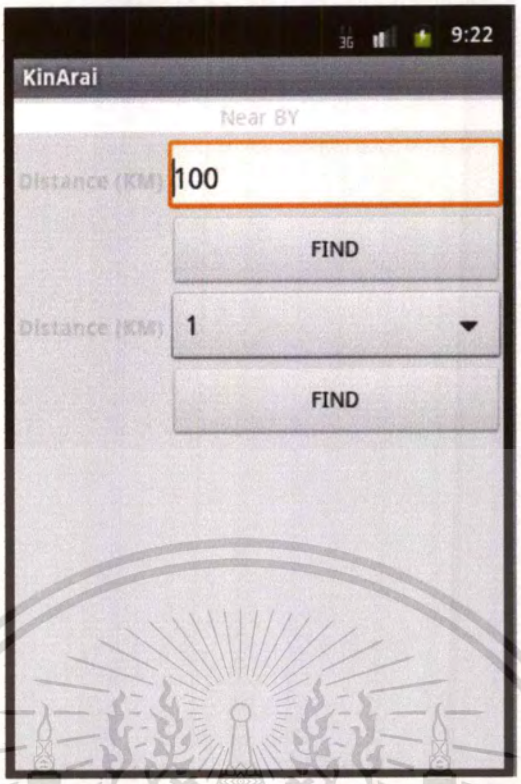


Figure 3.34 Enter the radius

Figure 3.35 Select the restaurant for navigation

Figure 3.36 Press the marker to navigate

3.3 Implementation

3.3.1 Implementation at the Web back-end

3.3.1.1 Extracting the restaurant promotion

Since web promotion sites provide several types of promotion, our system must provide a mechanism to extract only restaurant promotion item. See Figure 3.37 for a flowchart of the method that our system will use to extract restaurant items from all promotions. Note that before storing data into the promotion table, the system also has to extract the restaurant item for values for the field *title*, *description*, *link*, *puBdate* and *image*.

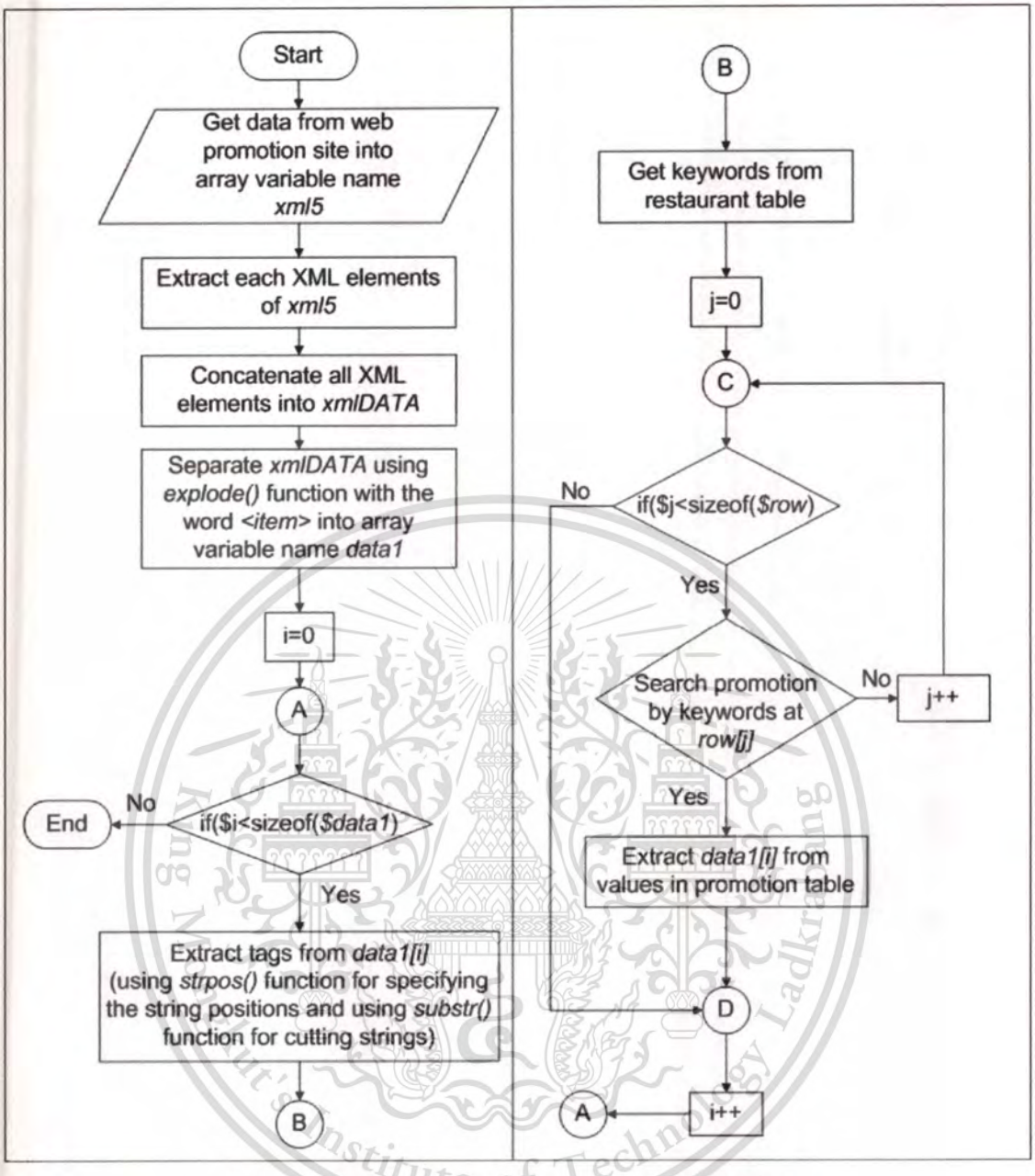


Figure 3.37 Extracting the restaurant promotion

3.3.1.2 PHP function: Cutting String

In Figure 3.38, PHP have functions to specify the words or sentences for cutting the sentence or string that you are interested in. For these functions to be well used, it must be combined with other functions. For example, the *strpos* function is used to specify the word position. The *substr* (subtract function) is used to cut the words or sentences that you specified the positions. The format of the *strpos* is *strpos(string, findstring)*. The first parameter is a string to search such as a

sentence. The second parameter is a string that we want to find. The output of the *strpos* is the first position of the *findstring*. The format of the *substr* is *substr(string, start_position, length)* . The first parameter is the source string. The second parameter is the starting position of the string for extraction. The third parameter is the number of characters to extract. For example, strpos($value,"<title>") and strpos($value,"</title>") are used to specify the string's position of the sentence. In this example we specify the two strings <title> and </title> and then use substr($value,strpos($value,"<title>"),strpos($value,"</title>")) to extract the sentence using the string's starting and ending positions received from strpos() function. The strip_tags() function is used for striping a string from HTML, XML, and PHP tags (get rid of the 'tag' to only the normal message left). The last function is trim() function used for removing whitespace from both side of a string.

```
21  $iTitle[$key]=trim(strip_tags(substr($value,strpos($value,"<title>"),strpos($value,
    "</title>"))));
22  $iLink[$key]=trim(strip_tags(substr($value,strpos($value,"<link>"),strpos($value,"
    </link>")-strpos($value,"<link>"))));
```

Figure 3.38 Cutting string in PHP

3.3.1.3 PHP function: Searching the words

In Figure 3.39, PHP uses a function *eregi* to search for the required word with case insensitive in the string or sentence. If it is found, it will return true. Figure 3.39, shows part of the code that our system used to extract the required promotion item from all promotion received from the feed. We use *eregi* function to search for the existence of the restaurant keyword in the title and the description of each XML's items. For example, *eregi($rows['k_resEng'], $iTiTle[$key]*) is a code to search whether the keyword in field *k_resEng* can be found in the *$iTitle[$key]) . iTitle is an array which stores all titles extracted from the web promotion feed. $key is the index element in iTitle.*

```
50  if(eregi($rows['k_resEng'], $iTitle[$key])or eregi($rows['k_resEng'], $iDesc[$key])or
    eregi($rows['k_resThai'], $iTitle[$key]) or eregi($rows['k_resThai'], $iDesc[$key])){
```

Figure 3.39 Searching the words function

3.3.1.4 PHP functions: Working with database

The PHP provides the functions to connect and query data from the database. First, *mysqli_connect* function is used to connect to the database. Then, *mysqli_select_db* is used to select the database. The *mysqli_query* function is used to manage the database which consists of at least 2 parameters: the former one is used to indicate the connection to the database and the latter one is the SQL commands . See Figure 3.40 for each function's example.

```
41  $cn = mysqli_connect("localhost", "root", "sawasdee");
42  $cn->set_charset('utf8');
43  mysqli_select_db($cn, "kinarai");
54  $iTitle[$key]=addslashes($iTitle[$key]);
55  $iLink[$key]=addslashes($iLink[$key]);
56  $iDesc[$key]=addslashes($iDesc[$key]);
57  $iImage[$key]=addslashes($iImage[$key]);
58  $ipubDate[$key]=addslashes($ipubDate[$key]);
59  $sql_p = "INSERT INTO promotions
    VALUES('{$rows['id_restaurants']}','{$id_promotions}','{$iTitle[$key]}','{$iDesc[$k
    ey]}','{$iImage[$key]}','{$iLink[$key]}','{$ipubDate[$key]}');";
60  $result_p = mysqli_query($cn,$sql_p);
```

Figure 3.40 Collecting the data into database

3.3.1.5 PHP: Building the XML file

The XML files have to initiate with <?xml version="1.0" encoding="utf-8"?> header as shown in Figure 3.41

```
20      $data  = '<?xml version="1.0" encoding="utf-8"?>'."\n";
21      $data .= '<rss version="2.0">'."\n";
22      $data .= '<channel>'."\n";
```

Figure 3.41 XML header file

The body of XML file consists of element tags, started with start tag and ended with end tag. Note that the ended tag must include / before the tag name. Example tags are *<title>...</title>* and *<description1>...</description1>*. See Figure 3.42 for more tags and how to create the XML content in PHP.

```
46      $data .= '<item>'."\n";
47      $data .= '<title><![CDATA['.html_entity_decode(stripslashes($rows2['title'])).
']]></title>'."\n";
48      $data .= '<description1><![CDATA['.html_entity_decode(stripslashes($rows2[
'description'])).']]></description1>'."\n";
49      $data .= '<image>'.html_entity_decode(stripslashes($rows2['image'])).'</image>'
."\n";
50      $data .= '<link>'.html_entity_decode(stripslashes($rows2['link'])).'</link>'.
"\n";
51      $data .= '<pubDate>'.html_entity_decode(stripslashes($rows2['pubdate'])).'
</pubDate>'."\n";
52      $data .= '</item>'."\n";
```

Figure 3.42 Body of XML file on PHP

The *fopen()* function is used by PHP to create a file. The *fputs()* function is used to write the content into the file. After writing the contents successfully, *fclose()* function is used to close the file. Figure 3.43 shows how to create the XML file.

```
57      $f = fopen( "test".$rows['id_restaurants'].".xml" , 'w' );
58      fputs( $f , $data );
59      fclose( $f );
```

Figure 3.43 Building an XML file on PHP

3.3.1.6 Encoding data into JSON format

Figure 3.44 shows the php function to encode data from array to **JSON** format by using *json_encode()* function.

```
23  $arr = array(data => $Arrobj);
24  echo json_encode($arr);
```

Figure 3.44 Encoding data into JSON format on PHP

3.3.2 Implementing Android Application: Promotion function

3.3.2.1 Building the Android XML layout of the screen

To implement the android application, XML layout of every component on the main menu (first screen) must be written first. Figure 3.45 shows our application's first screen XML layout which consists of three components: one *Textview* and two *Imagebutton*. *Textview* is used to set the properties of text. *Imagebutton* is used to set the properties of pictures on the screen. Similarly, for other screen, we must create its XML layout before writing the functional codes. All layouts are stored in folder *layout* under folder *res*.

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/widget56"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_x="1dip"
        android:layout_y="17dip"
        android:gravity="center"
        android:text="KinArai"
        android:textSize="26px" >
    </TextView>

    <ImageButton
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="6dip"
        android:layout_y="112dip"
        android:src="@drawable/eat1" >
    </ImageButton>

    <ImageButton
        android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="162dip"
        android:layout_y="112dip"
        android:src="@drawable/nearby1" >
    </ImageButton>

</AbsoluteLayout>
```

Figure 3.45 The XML layout of the main menu

3.3.2.2 Main menu for the promotion function

After building the XML layout, we must set the content view for the main menu. Figure 3.46 shows method onCreate, which is used to create the content view for our application's main menu. *ImageButton* is the library that used to control tasks on the button. *.setOnClickListener* is a function which waits for the click from the user. If the user click, the program then calls method *onClick,* which it will call *Intent* function to change the screen page to the new class (in this case it is *NewKinAraiActivity.class*).

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.mainscreen);

    ((ImageButton) findViewById(R.id.btn1))
            .setOnClickListener(new OnClickListener() {
                // @Override
                public void onClick(View v) {

                    Intent myIntent = new Intent(PromotionActivity.this,
                            NewKinAraiActivity.class);
                    // myIntent.setClassName("kinarai.com",
                    // "kinarai.com.promotionlist");
                    // myIntent.putExtra("ARRIVING_FROM","1");
                    startActivity(myIntent);

                }

            });
```

Figure 3.46 Android: Implementation code of the main screen for promotion function

3.3.2.3 Obtaining the list of restaurants from the web server

We get the restaurant list from the web server by using XML feed. First, we declare local variables of this class. We use array name *feedres* to store feed information. *feedres* uses the FeedURL structure (see Section 3.3.2.9 for more details on FeedURL). The method to get feed from URL is created using the *getRSSFeed()*.

```
public class NewKinAraiActivity extends ListActivity {
    public static ArrayList<FeedURL> feedres = null;
    URL url ;
    private ProgressDialog dialog;
    TextView textViewNodata;
    @Override


    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        feedres = getRSSFeed();
        setListAdapter(new MyRssAdapter());


}
```

Figure 3.47 Declaring the method

3.3.2.4 Method to get XML URL

Figure 3.48 shows the code for the method to get feed from URL. The feed then is stored into the array list of *FeedURL*. URL location is specified in the bracelet after the URL function, which in our program we store in variable *url*. *url.getContent* indicates that the information will be retrieved from the *url*. *SAXParserFactory*, *SAXParser* and *XMLReader* are used to read XML content from the *url*. We use variable *rssXmlHandler* to read each tag information from the feed content. *rssXmlHandler* is a variable for *RssRes* class (see Section 3.3.2.10 for more details on *RssRes* class). *feedItemList* obtains the value from *rssXmlHandler.getFeedItem*.

```
private ArrayList<FeedURL> getRSSFeed() {
    try {
        url = new URL("http://kinarai.no-ip.org/kinarai/kinarai_php/restaurants.xml");
        InputStream is = (InputStream) url.getContent();
        SAXParserFactory spf = SAXParserFactory.newInstance();

        SAXParser sp = spf.newSAXParser();
        XMLReader xr = sp.getXMLReader();

        /** Create handler to handle XML Tags ( extends DefaultHandler ) */
        RssRes rssXmlHandler = new RssRes();
        xr.setContentHandler(rssXmlHandler);
        // xr.parse(new InputSource(sourceUrl.openStream()));
        InputSource inputSource = new InputSource(is);
        xr.parse(inputSource);
        ArrayList<FeedURL> feedItemList = rssXmlHandler.getFeedItemList();
        return feedItemList;
    } catch (MalformedURLException e) {
        Log.e("RSSFeed", "ERROR", e);
        return null;
    } catch (IOException e) {
        Log.e("RSSFeed", "ERROR", e);
        return null;
    } catch (ParserConfigurationException e) {
        Log.e("RSSFeed", "ERROR", e);
        return null;
    } catch (SAXException e) {
        Log.e("RSSFeed", "ERROR", e);
        return null;
    }
}
```

Figure 3.48 Method to get feed from URL

3.3.2.5 Class to get and store data into the list view

This class *MyRssAdapter* is used to get data from the array list into each list on the list view screen. *LayoutInflator* is used to instruct the program to get data into the list view. *getCount* is used to obtain the number of the items in the

list. *getItem* is used to obtain each item number in the list. See Figure 3.49 for more details of the code. Figure 3.50 shows the code that displays the data in the list view.

```
class MyRssAdapter extends BaseAdapter {

    LayoutInflater inflater;

    public MyRssAdapter() {
        inflater = getLayoutInflater();
    }

    @Override
    public int getCount() {
        // TODO Auto-generated method stub
        return feedres.size();
    }

    @Override
    public Object getItem(int position) {
        // TODO Auto-generated method stub
        return feedres.get(position);
    }

    @Override
    public long getItemId(int arg0) {
        // TODO Auto-generated method stub
        return 0;
    }
```

Figure 3.49 Getting the data to each of the list in the list view

```
@Override
public View getView(int position, View convertView, ViewGroup arg2) {

    if(convertView instanceof TextView){
        convertView = null;
    }
    View row = convertView;
    ViewHolder viewHolder = null;
    if(convertView == null) {
        row = inflater.inflate(R.layout.rss, null);
        viewHolder = new ViewHolder();
        viewHolder.tvTitle = (TextView) row.findViewById(R.id.tv_title);
        //viewHolder.tvdescription = (TextView) row.findViewById(R.id.tv_description);
        //viewHolder.tvDate = (TextView) row.findViewById(R.id.tv_date);
        viewHolder.tvimg = (ImageView) row.findViewById(R.id.thumb);
        row.setTag(viewHolder);
    }else
        row = convertView;
        viewHolder = (ViewHolder) row.getTag();

FeedURL feedItem = (FeedURL) getItem(position);

    viewHolder.tvTitle.setText(feedItem.getTitle());
    //viewHolder.tvdescription.setText(feedItem.getDescription());
    //viewHolder.tvimg.setTag(feedItem.getimage());
```

Figure 3.50 Displaying the data in the list view

3.3.2.6 Method to get the image

Figure 3.51 shows a method to get images from URL. We note that when we obtain the XML data, it provides only the link to download the pictures. We use *BitmapFactory.decodeStream* function to download the picture from the *url*. *viewHolder.tvimg.setImageBitmap* is used to set the image view in the layout.

```
        // url = null;
        try {
            url = new URL(feedres.get(position).getimage());
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        InputStream is = null;
        try {

             is = (InputStream) url.getContent();

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    Drawable d = Drawable.createFromStream(is , "src");
        Bitmap mIcon1 = null;
         try {

            mIcon1 =
                    BitmapFactory.decodeStream(url.openConnection().getInputStream());
        } catch (IOException e) {
            e.printStackTrace();
        }
        // viewHolder.tvimg.setImageBitmap(Bitmap.createScaledBitmap(mIcon1, 74, 74,false));

            viewHolder.tvimg.setImageBitmap(mIcon1);

        return row;
    }

    }
```

Figure 3.51 Method to get image from URL

3.3.2.7 Method to wait for the user click

Figure 3.52 shows the *onListItemClick* method which is a method to wait for the user to click on the list and then go to the next class which in this example it will go to *GetFeedpromotion* class. Note that *GetFeedpromotion* class is a class for screen in Figure 3.26 (Select the promotion for more details).

```
public void onListItemClick(ListView l, View v, int position, long id) {
    if (url == null){
        setContentView(R.layout.restaurant_main);
        v.setVisibility(View.VISIBLE);
    }else{
    Getfeedpromotion.profeed = feedres.get(position);
    Toast.makeText(NewKinAraiActivity.this, feedres.get(position).restaurant, Toast.LENGTH_LONG).show();

    Intent intent = new Intent(this,Getfeedpromotion.class);
    startActivity(intent);
    }
}
```

Figure 3.52 Method to wait for the user click on the list

3.3.2.8 Method to show more details of the promotion

The method *Moredetails* in Figure 3.53 shows the implementation of more details of the promotion when the user click on the list. First the local variables are declared. *promotiondetail* is the name of the variable of class *Feeditem*. See Figure 3.54 for *Feeditem* structure.

```
public class Moredetails extends Activity{

    public static Feeditem promotiondetail;
    private Button link;
    private Button back;
    private TextView txtShow;
    private TextView txtdes;
    private ImageView imgShow;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.diaosshow);

        txtShow = (TextView)findViewById(R.id.TextView01);
           txtShow.setText(promotiondetail.getTitle().toString());
           txtdes = (TextView)findViewById(R.id.textViewdetail);
           txtdes.setText(promotiondetail.des);
```

Figure 3.53 More details class

3.3.2.9 Class for storing data from XML tags

There are two classes: *FeedURL* and *Feeditem*. *FeedURL* will transform restaurants information from the server to three strings which are the restaurant title, its image and its link. *Feeditem* will transform the promotion information from the server to five strings which are the restaurant title, its link, its pubdate, and its image as shown in Figure 3.54.

```java
public class FeedURL {
    String restaurant;
    String image;
    String link;


    public String getTitle() {
          return restaurant;
      }
      public void setTitle(String title) {
          this.restaurant = title;
      }
      public String getLink() {
          return link;
      }
      public void setLink(String link) {
          this.link = link;
      }
      public String getimage() {
          return image;
      }
      public void setimage(String image) {
          this.image = image;
      }
}
```

```java
public class Feeditem {
    String title;
    String link;
    Date pubDate;
    String des;
    String image;

    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getLink() {
        return link;
    }
    public void setLink(String link) {
        this.link = link;
    }
    public Date getPubDate() {
        return pubDate;
    }
    public void setPubDate(Date pubDate) {
        this.pubDate = pubDate;
    }
    public String getDescription(){
        return des;
        }
    public void setDescription(String des){
        this.des = des;
    }
    public String getimage() {
        return image;
    }
    public void setimage(String image) {
        this.image = image;
    }
```

Figure3.54 Class for storing data from XML tags

3.3.2.10 Class for storing XML tags from feed

RssRes is a class to collect the data from each XML tag.

```java
public class RssRes extends DefaultHandler {
    Boolean currentElement = false;
    String currentValue;
    ArrayList<FeedURL> feedresList = new ArrayList<FeedURL>();
    FeedURL currentFeedItem;

public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {

        currentElement = true;
        if (localName.equals("item")) {
            currentFeedItem = new FeedURL();
        }
    }

    @Override
    public void endElement(String uri, String localName, String qName) throws SAXException {
        currentElement = false;
        if (currentFeedItem != null) {
            if (localName.equals("restaurant")) {
                currentFeedItem.setTitle(currentValue);
            } else if (localName.equals("link")) {
                currentFeedItem.setLink(currentValue);
            } else if (localName.equals("image")) {
                currentFeedItem.setimage(currentValue);
            } else if (localName.equals("item")) {
                feedresList.add(currentFeedItem);
            }
        }

    }
```

Figure3.55 Class for storing XML tags from feed

3.3.3 Implementing Android Application: Nearby function

3.3.3.1 Checking the current location of the user

In the main Nearby screen, our application checks the current location of the user by using GPS or 3G services on the phone. Restaurants' location information is obtained from the web server using JSON format. To check for the enabled location service in the android device, function *getsystemService (Context.LOCATION_SERVICE)* is used. Then, function *isProviderEnabled* is used to check whether the particular network (specified in the second parameter) is enable. For our application, we check for two networks: GPS and 3G. Function *requestLocationUpdates* is used to obtain the current location from the enabled services which the last parameter of the function is used to store the obtained location (*loclistener*). Function *getRestaurantModelFromService* from class *ServiceRestaurant* is used for getting the restaurants location information from the server. See Section 3.3.3.2 for more details of *ServiceRestaurant* class.

```
((ImageButton) findViewById(R.id.btnZ))
        .setOnClickListener(new OnClickListener() {
            // @Override
            public void onClick(View v) {


                locManager = (LocationManager) PromotionActivity.this.getSystemService(Context.LOCATION_SERVICE);
                try {
                    gps_enabled = locManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
                } catch (Exception ex) {
                }
                try {
                    network_enabled = locManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
                } catch (Exception ex) {
                }

                if (gps_enabled) {
                    locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locListener);
                }
                if (network_enabled) {
                    locManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locListener);
                }

                if (NearbyActivity.arrayListRestaurantModel == null) {
                    ProgressDialog dialog = ProgressDialog.show(PromotionActivity.this, "", "Loading. Please wait...",
                            true);


                    ServiceRestaurant service = new ServiceRestaurant();
                    NearbyActivity.arrayListRestaurantModel = service.getRestaurantModelFromService();

                    dialog.dismiss();
                }
```

Figure 3.56 Getting locations of the current user and all restaurants

3.3.3.2 Class to load JSON object from the web service

The *ServiceRestaurant* class as shown in Figure 3.57 is a class to load JSON data from the URL *http://kinarai.no-ip.org/kinarai/Kinarai_Service/Service.php?v=JSON* from the web server. *jsonArray* (a variable class *JSONArray*) is used to obtain JSON data from *getJSONArray* function. The system will extract each element in JSON object by using *getJSONObject* function. For example, *String id = jsonObject.getString("id")* is a code to extract the *id* value of each item to store in string *id*. This extraction is done until the last element of JSON is extracted. Lastly, all string values (*id, name, lat, ...,location*) will be put in the array list name *arrayListSubjectModel*.

```
public class ServiceRestaurant {

    public ArrayList<RestaurantModel> getRestaurantModelFromService() {

        ArrayList<RestaurantModel> arrayListSubjectModel = new ArrayList<RestaurantModel>();
        // StudentModel studentModel = null;

        JSONObject json = JSONfunctions.getJSONfromURL("http://kinarai.no-ip.org/kinarai/Kinarai_Service/Service.php?v=json");
        try {// success
            JSONArray jsonArray = json.getJSONArray("data");

            for (int i = 0; i < jsonArray.length(); i++) {
                JSONObject jsonObject = jsonArray.getJSONObject(i);
                String id = jsonObject.getString("id");
                String name = jsonObject.getString("name");
                String lat = jsonObject.getString("lat");
                String lon = jsonObject.getString("long");
                String image = jsonObject.getString("pic");
                String phone = jsonObject.getString("tellnum");
                String location = jsonObject.getString("location");
                arrayListSubjectModel.add(new RestaurantModel(id, name, lat, lon, image, phone,location));
            }

            return arrayListSubjectModel;

        } catch (JSONException e) {

            Log.e("log_tag", "Error parsing data " + e.toString());

        }
        return arrayListSubjectModel;
    }
}
```

Figure 3.57 Service Restaurant Class

3.3.3.3 Declaring the local variables of the Nearby class

We declare the local variables of the nearby class as shown in Figure 3.57. This class is used for the screen in Figure 3.30 (Enter the radius) *dis1* is a variable used for the drop down distance (1 to 10).

```
public class NearbyActivity extends Activity {

    String dis1[] = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10" };


    public static ArrayList<RestaurantModel> arrayListRestaurantModel;
    public EditText editTextKM;
    public ArrayList<RestaurantModel> arrayListRestaurantModelInKilo;

    int curLocation = -1;
    int locationBuffer = 3;
    Location[] pastLocations = new Location[locationBuffer];
    Spinner spinner_dist1;


    ArrayList<RestaurantModel> placelocation = new ArrayList<RestaurantModel>();


    String url1 = "http://kinarai.no-ip.org/kinarai/Kinarai_Service/Service.php?v=";


    static final String[] Tag = new String[] {
        "Centralworld",
        "Latkrabang","Ratchathewi","Pathumwan","ChongNonsi","Suvarnabhumi","Bangna","Bangcare","Bangluk","Bangkaie"
    };
```

Figure 3.58 Local variables of the nearby class

3.3.3.4 Method onCreate in Nearby class

In Figure 3.59, we declare the variable to be use for coding in this class.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.findnearby);


     final AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.autocomplete_places);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, R.layout.tag_item, Tag);
        textView.setAdapter(adapter);


    arrayListRestaurantModelInKilo = new ArrayList<RestaurantModel>();

    spinner_dist1 = (Spinner) findViewById(R.id.dist1);
    ArrayAdapter<String> adapter_dist1 = new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, dis1);
    adapter_dist1
            .setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spinner_dist1.setAdapter(adapter_dist1);
    spinner_dist1.setOnItemSelectedListener(new MyOnItemSelectedListener());


    editTextKM = (EditText) findViewById(R.id.distxt);
```

Figure 3.59 Method onCreate

3.3.3.5 Searching from the user specified input

Our application provides three approaches to search for the nearby restaurant. The first one, the user can specify the radius by himself. The user enters his specified value in the text box(*editTextKM* in Figure 3.60). The value is changed from string to double. *CheckListRestaurantKilo* is used to check for the restaurants that their distance from the current location is less than the variable

kilodouble, which they will be shown in the list on the screen in Figure 3.31 (Select the restaurant for navigation). It has a method to sort the result from the minimum to the maximum value (*Collection.sort*).

```
((Button) findViewById(R.id.btn1))
        .setOnClickListener(new OnClickListener() {
            // @Override
            public void onClick(View v) {

                // //mapShowOS();
                // Intent myIntent = new Intent(nearby.this ,
                // HelloMapsActivity.class);
                // startActivity(myIntent);


                // My Code
                // Check
                if (!editTextKM.getText().toString().equals("")) {
                    String kilo = editTextKM.getText().toString();
                    double kiloDouble = Double.valueOf(kilo);

                    checkListRestaurantKilo(kiloDouble);
                }


                Collections.sort(arrayListRestaurantModelInKilo, new Comparator<Object>(){
                    public int compare(Object o1, Object o2) {
                        RestaurantModel p1 = (RestaurantModel) o1;
                        RestaurantModel p2 = (RestaurantModel) o2;
                       return p1.kilo.compareToIgnoreCase(p2.kilo);
                    }
                });


                ListRestaurantActivity.arrayListRestaurantModelInKilo = arrayListRestaurantModelInKilo;


                Intent myIntent = new Intent(NearbyActivity.this,ListRestaurantActivity.class);
                startActivity(myIntent);

            }
        });
```

Figure 3.60 Searching by user specified radius

3.3.3.6 Searching from the input from dropdown menu

The second approach, the user simply chooses the distance of 1 to 10 from the dropdown (*spinner_dist1.getSelectedItem()*). Similar to the code int he previous section, *CheckListRestaurantKilo* is used to check for the restaurants that their distance from the current location is less than the variable *kilodouble*. For the rest of the code is the same as the code in the previous Figure 3.60.

```
((Button) findViewById(R.id.btn2))
        .setOnClickListener(new OnClickListener() {
            // @Override
            public void onClick(View v) {


                // Check Kilo Spinner
                double kiloDouble = Double.valueOf(spinner_dist1.getSelectedItem().toString());

                checkListRestaurantKilo(kiloDouble);

                Collections.sort(arrayListRestaurantModelInKilo, new Comparator<Object>(){
                    public int compare(Object o1, Object o2) {
                        RestaurantModel p1 = (RestaurantModel) o1;
                        RestaurantModel p2 = (RestaurantModel) o2;
                       return p1.kilo.compareToIgnoreCase(p2.kilo);
                    }
                });
                ListRestaurantActivity.arrayListRestaurantModelInKilo = arrayListRestaurantModelInKilo;


                Intent myIntent = new Intent(NearbyActivity.this,ListRestaurantActivity.class);
                startActivity(myIntent);

            }

        });
```

Figure 3.61 Searching by dropdown distance

3.3.3.7 Searching for the restaurants near the specified place

The third approach, the user is allowed to type the name of the place, district and the restaurant name. Then, the system will search for any restaurants near these landmarks. The user does not have to enter his landmarks full name. Once its initial is entered, the screen will list all places/districts or restaurant names that started with this initial (See method onCreate in Figure 3.59 for more details). The name then is combined with the value in variable *url1* and stored in variable *feedplace. json* variable is used to tell the system to go to the *feedplace* (through *getJSONfromURL* function) for JSON value. Similar to a code in Section 3.3.3.2, *jsonArray* (a variable class *JSONArray*) is used to obtain JSON data from *getJSONArray* function which the system will extract each element in JSON object by using *getJSONObject* function. All string values (*id, name, lat, ...,location*) will be put in the array list name *placelocation* (*placelocation.add*). The *Intent* function is used to change the screen. In this code, it changes from the Nearby screen to the Map screen.

```
((Button) findViewById(R.id.f1)).setOnClickListener(new OnClickListener(){

        @Override
        public void onClick(View arg0) {

            if(!textView.getText().toString().equals("")){

                String place = textView.getText().toString();
                String feedplace = url1+place;
                JSONObject json = JSONfunctions.getJSONfromURL(feedplace);
                try {// success
                    JSONArray jsonArray = json.getJSONArray("data");

                    for (int i = 0; i < jsonArray.length(); i++) {
                        JSONObject jsonObject = jsonArray.getJSONObject(i);
                        String id = jsonObject.getString("id");
                        String name = jsonObject.getString("name");
                        String lat = jsonObject.getString("lat");
                        String lon = jsonObject.getString("long");
                        String image = jsonObject.getString("pic");
                        String phone = jsonObject.getString("tellnum");
                        String location = jsonObject.getString("location");
                        placelocation.add(new RestaurantModel(id, name, lat, lon, image, phone, location));


                    }} catch (JSONException e) {

                    }

                MapresActivity.placelocation = placelocation;
                Intent myIntent = new Intent(NearbyActivity.this,MapresActivity.class);
                startActivity(myIntent);

                }

        }

    });
```

Figure 3.62 Searching by entering place/district/name

3.3.3.8 Method for converting latitude and longitude distance to kilometers

The *checkKilo* method calculates the distance from the user current location to each restaurant in latitude and longitude values. Then, converts the distance into kilometer.

```
public double checkKilo(String lat, String lon) {
    Location dest = new Location("");
    dest.setLatitude(Double.valueOf(lat));
    dest.setLongitude(Double.valueOf(lon));

    DecimalFormat twoDForm = new DecimalFormat("#.##");
    float meters = PromotionActivity.currentLocation.distanceTo(dest);
    float miles = meters * 0.000621371192f;
    float kilo = miles * 1.609344f;

    return Double.valueOf(twoDForm.format(kilo));
}
```

Figure 3.63 Method for converting distance to kilometer

3.3.3.9 Method to check the restaurants within the nearby area

In this method, it compares the user selected distance (*kiloNomore*) with the distance returned from *checkKilo* (*direction*). If *checkKilo* is less than *kiloNomore,* it means that this restaurant location is within the user specified radius so this restaurant information will be stored in the *arrayListRestaurant ModelInKilo.*

```
public void checkListRestaurantKilo(double kiloNoMore)
{
    // Check Kilo
    if (arrayListRestaurantModel != null) {
        arrayListRestaurantModelInKilo.clear();
        for (int i = 0; i < arrayListRestaurantModel.size(); i++) {
            double direction = checkKilo(arrayListRestaurantModel.get(i).lat,arrayListRestaurantModel.get(i).lon);
            arrayListRestaurantModel.get(i).kilo = direction+"";
            if (direction <= kiloNoMore) {
                arrayListRestaurantModelInKilo.add(arrayListRestaurantModel.get(i));

            }
        }
    }
}
```

Figure 3.64 Method to store the nearby restaurant into the list

3.3.3.10 Class to display the restaurants within the nearby area

This class checks whether the *arrayListRestaurantModel InKilo.size* is not zero. If so, the restaurant list will be shown. Otherwise, it will show that there is no data available. *listBaseAdapter* is a method to show the restaurant list. *listView.setAdapter(listBaseAdapter)* and *listView.setOnItemClickListener(this)* are functions that allow the user to click on any of the list.

```
public static ArrayList<RestaurantModel> arrayListRestaurantModelInKilo = null;

public ListView listView;
ListAdapterRestaurant listBaseAdapter;
TextView textViewNodata;

@Override
public void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.restaurant_main);

    textViewNodata = (TextView)findViewById(R.id.textViewNoData);
    listView = (ListView) findViewById(R.id.List_RestaurantActivity);
    listView.setBackgroundColor(android.R.color.white);

    if (arrayListRestaurantModelInKilo != null) {

        if(arrayListRestaurantModelInKilo.size()!=0){

            textViewNodata.setVisibility(View.GONE);
        }
        else
        {

            textViewNodata.setVisibility(View.VISIBLE);
        }

        listBaseAdapter = new ListAdapterRestaurant(this,arrayListRestaurantModelInKilo);
        listView.setAdapter(listBaseAdapter);
        listView.setOnItemClickListener(this);
    }
    else
    {

        textViewNodata.setVisibility(View.VISIBLE);
    }
}
```

Figure 3.65 Class to display the nearby restaurant's list

3.3.3.11 Method waiting for the user click

OnItem is a method to send the result of the array list in to the next class. So, after the user in Section **3.3.3.10** clicks the list, the program will go to the screen as appear in Figure 3.32 (Press the marker to navigate).

```
@Override
public void onItemClick(AdapterView<?> arg0, View arg1, int position, long arg3) {
    // TODO Auto-generated method stub
    //Go To Map
//  RestaurantModel restaurantModel = arrayListRestaurantModelInKilo.get(position);
//  Intent intent = new Intent(android.content.Intent.ACTION_VIEW,Uri.parse("http://maps.google.com/maps?saddr=" + "&daddr="
//  + restaurantModel.lat + "," + restaurantModel.lon));
//  startActivity(intent);

    MapDetailActivity.restaurantModelClick = arrayListRestaurantModelInKilo.get(position);

    Intent intent = new Intent(this,MapDetailActivity.class);
    startActivity(intent);

}
```

Figure 3.66 Method waiting for the user click

3.3.3.12 Class to displaying the result in map view

This class will display the map of the chosen restaurant. It will also draw a marker of the restaurant's position on the map. *Lm.requestLocation Updates* is a function to get the current location of the user phone (see Figure 3.67). There are four parameters. The first one *LocationManager.GPS_PROVIDER* indicates that the location is obtained from the GPS. 10000 is the maximum time that you can view the map. 1 is the starting latitude. Then, the current location is stored in the fourth parameter *locationListener*. *mapView* is a variable to display a google map. It use *MapView* library. *mapView.setBuildInZoomControls(true)* allows the user to zoom in and out of the map. *mapView.setStreetView(true)* is a function to enable the map on android to show street however this feature is currently not allowed to be used. To set the zoom value, first we need to set mapView controller (*mc = (MapController) mapView.getController*). Then, we simply set the zoom value by using setZoom function (*mc.setZoom(14)*).

To set the marker onto the map, we first have to set an overlay variable (*starbuckItemOverlay*). The marker symbol on the overlay is set during the *starbuckItemOverlay* declaration in *RestaurantItemOverlay* method parameter (*getDrawable(R.drawable)*). To put the marker on the map, it is required to convert the latitude and longitude with the specific formula as shown in new *GeoPoint* code part in Figure 3.67. To show more of the maps' symbols and features such as compass functions and the symbol for the user's current location, it can be done through *myLocationOverlay* variable which is created based on *MyLocationOverlay* method. To add all the features to the mapView, we use function *mapView.getOverlay.add* which its parameter is the overlay variable. *mapView.getControl.animateTo(gp) forces* the marker to be appeared in the middle of the map.

```
requestWindowFeature(Window.FEATURE_NO_TITLE);
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);
setContentView(R.layout.map);

lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
locationListener = new MyLocationListener();
lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10000, 1,
        locationListener);

mapView = (MapView) findViewById(R.id.mapview1);

mapView.setBuiltInZoomControls(true);
mc = (MapController) mapView.getController();
mapView.setStreetView(true);
mapView.invalidate();
mc.setZoom(14);


//test convert data
starbuckItemOverlay = new RestaurantItemOverlay(getResources().getDrawable(R.drawable.bubble));

final GeoPoint gp = new GeoPoint(
        (int) (Float.valueOf(restaurantModelClick.lat) * 1E6),
        (int) (Float.valueOf(restaurantModelClick.lon) * 1E6));
starbuckItemOverlay.addOverlayItem(new OverlayItem(gp,restaurantModelClick.name, "Phone:"+ restaurantModelClick.phone));


MyLocationOverlay myLocationOverlay = new MyLocationOverlay(this,mapView);
myLocationOverlay.enableMyLocation();
myLocationOverlay.enableCompass();
mapView.getOverlays().add(myLocationOverlay);
mapView.getOverlays().add(starbuckItemOverlay);

mapView.getController().animateTo(gp);
```

Figure 3.67 Displaying the result in the map view

3.3.3.13 Method onTap

This method used for sending the result of the latitude and longitude of the marker to the map application in the phone. Function Intent is a function to change the page on Android. For the code in Figure 3.68, function *Intent* is used to trigger the map application.

```
@Override
protected boolean onTap(int index) {

    OverlayItem oi = overlayItemList.get(index);

    Toast.makeText(MapDetailActivity.this, restaurantModelClick.name, Toast.LENGTH_LONG).show();

    Intent intent = new Intent(android.content.Intent.ACTION_VIEW,Uri.parse("http://maps.google.com/maps?saddr=" + "&daddr="
    + restaurantModelClick.lat + "," + restaurantModelClick.lon));
    startActivity(intent);

    return super.onTap(index);

}
```

Figure 3.68 Method onTap

Chapter 4

Implementation

The implementation process is divided into 2 parts: back-end and front-end parts. The back-end part consists of processes on the web and database servers which prepare the data and services for the front-end. The front-end part consists of all processes on Android device.

4.1 Back-End

There are four main tasks at the back-end. The first task is to add the restaurant's promotion keywords into the restaurant table (see Section 4.1.1). The second task is to add the restaurant location to the restaurant location table (see Section 4.1.2). These two tasks are done by the application administrator which he has to login to the system first. The third task is to add restaurant promotions to the promotion table which is done automatically by the system every 30 minutes (see Section 4.1.3). The fourth task deals with services providing information to the front-end (see Section 4.14).

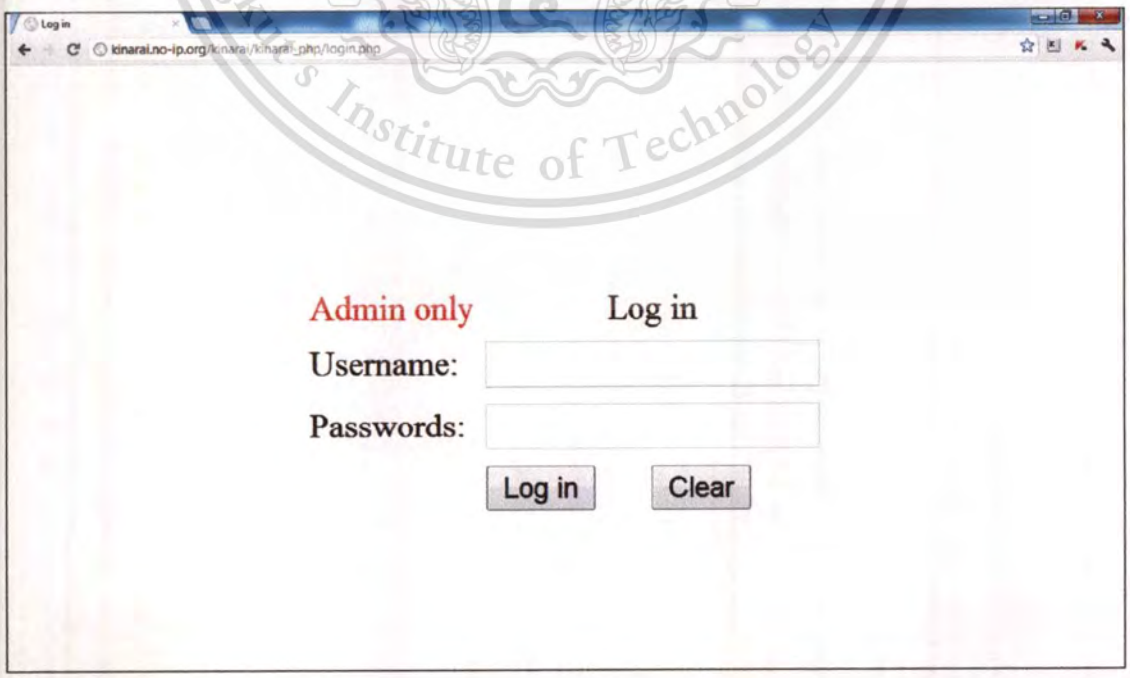


Figure 4.1 Log in page

By entering URL **http://kinarai.no-ip.org/kinarai/kinarai_php/login.php,** the login page as shown in Figure 4.1 will appear. The user fills in his username and password and presses log in button. If the username or password is incorrect, the system displays the warning message as shown in Figure 4.2. Note that the user in this case is the user with our application administrator's right.

Figure 4.2 Warning page

After the user has logged into the system with correct username and password, the system asks whether he wants to add either the promotion's keyword or to add the location of the restaurants (see Figure 4.3).

4.1.1 Adding the promotion keywords

As previously described in Chapter 3, keywords are important to our application since they are used to filter only restaurant promotions from all feed's promotions sent from our selected web promotion sites. Therefore, the administrator has to specify these keywords before. To add the keywords, the user first clicks **Add the keyword of promotion** button in Figure 4.3. Then the page as shown in Figure 4.4 will appear. The page in Figure 4.4 will ask the user to input the restaurant name, its keyword in English (it is not necessary that the keyword must be the restaurant name),

its keyword in Thai and the restaurant's logo. See Figure 4.5 for an example of adding the new keywords.

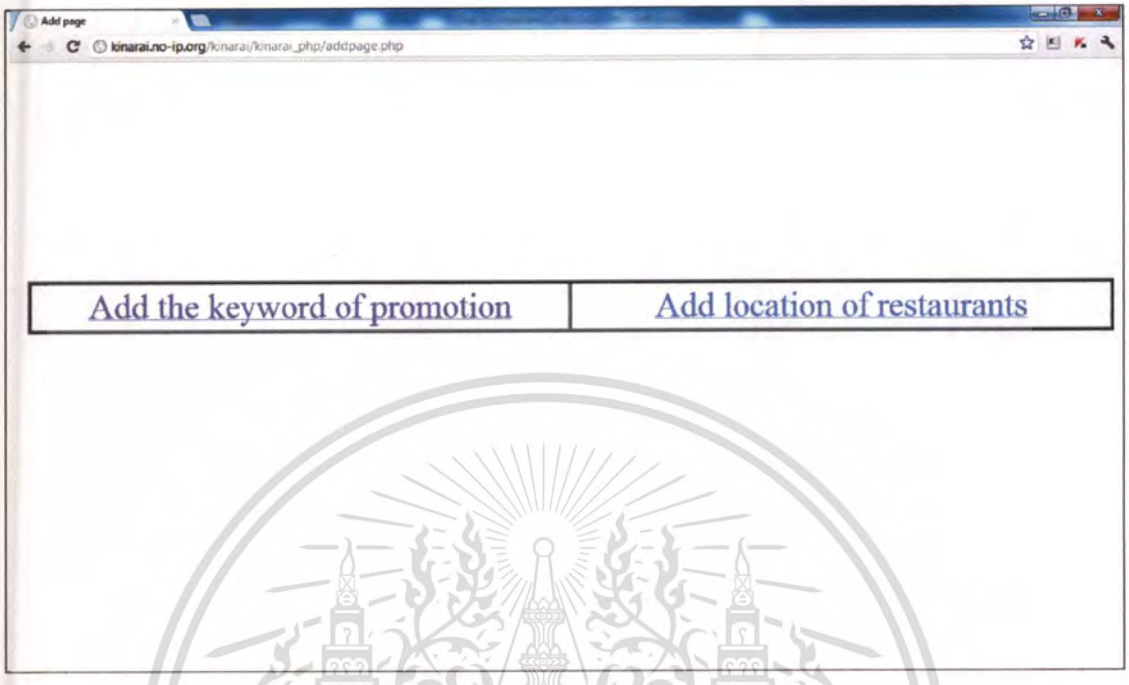


Figure 4.3 Server function's selection

Figure 4.6 displays the result of the data that the administrator has added into the database. If the system displays this page, it means that the promotion additional process is completed. If the user would like to add other keywords, he simply presses back button and adds more keywords.

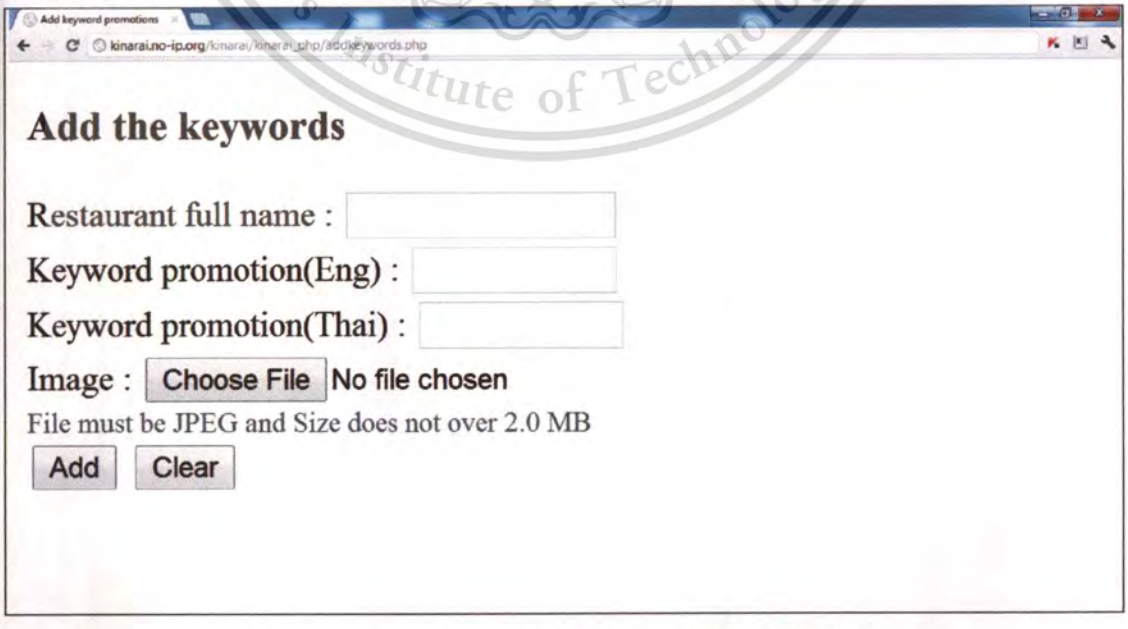


Figure 4.4 Adding the promotion keywords page

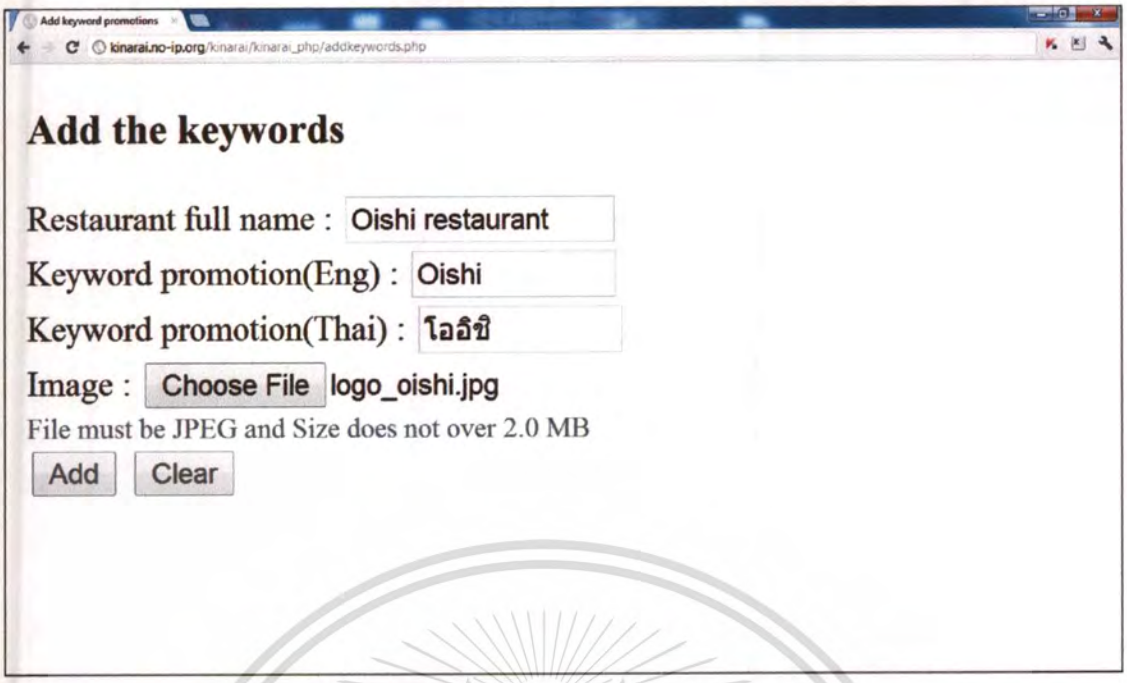


Figure 4.5 Example of adding keywords

Figure 4.6 Keyword adding result page

4.1.2 Adding the restaurant location

When the administrator wants to add the restaurant with its location, he selects **Add location of restaurants** menu as shown in Figure 4.7. Then, the page as shown in Figure 4.8 will appear.

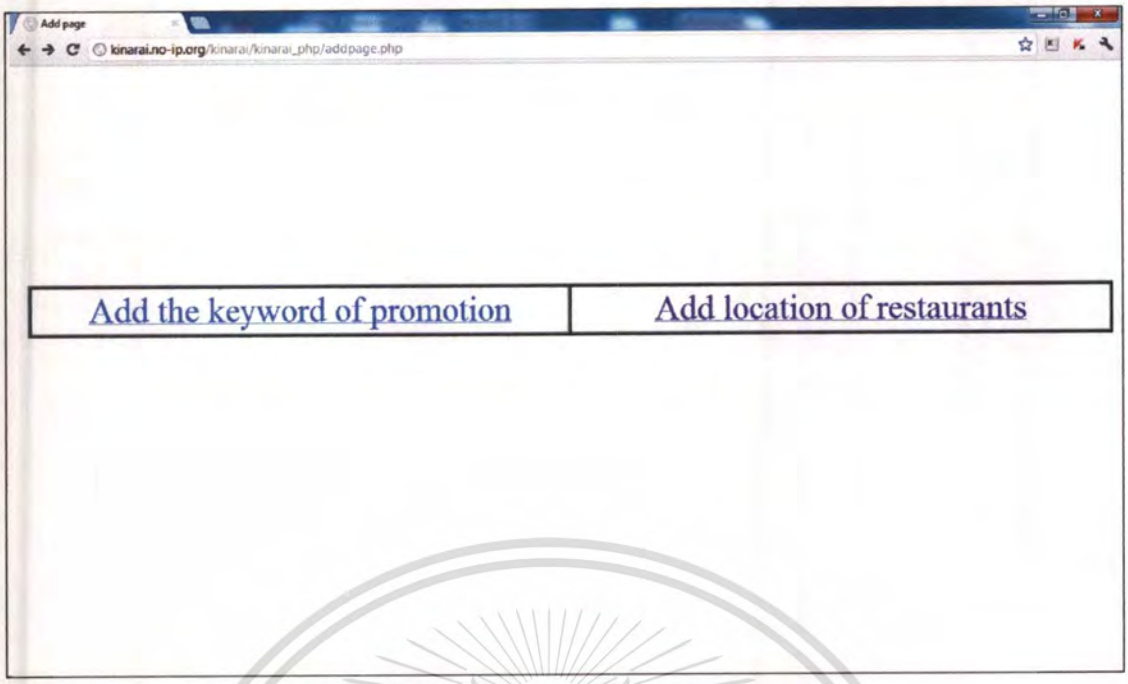


Figure 4.7 Server function's selection

The Add Location page in Figure 4.8 asks the user to input the restaurant name, its location in latitude and longitude form (see Section 4.1.5 for how to obtain these values), its telephone number and its image. We note that the restaurant image in this task is not needed to be a restaurant logo as in the previous task. It can be any picture of this restaurant. Once all details have been filled, press add button.

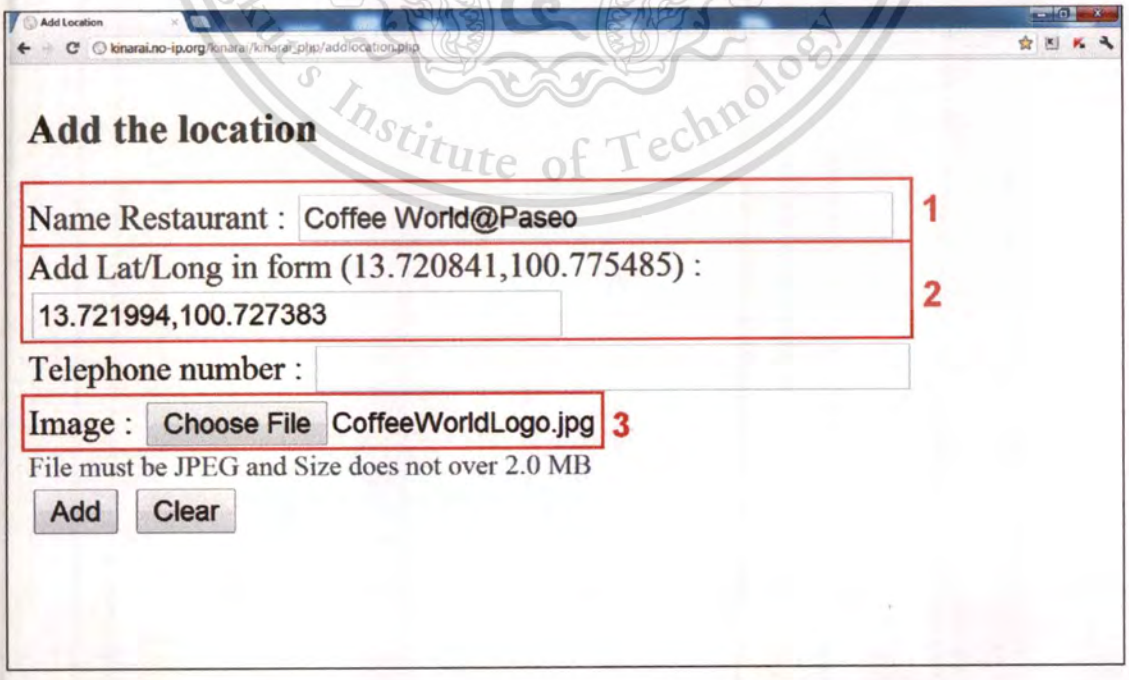


Figure 4.8 Page to add the restaurant location

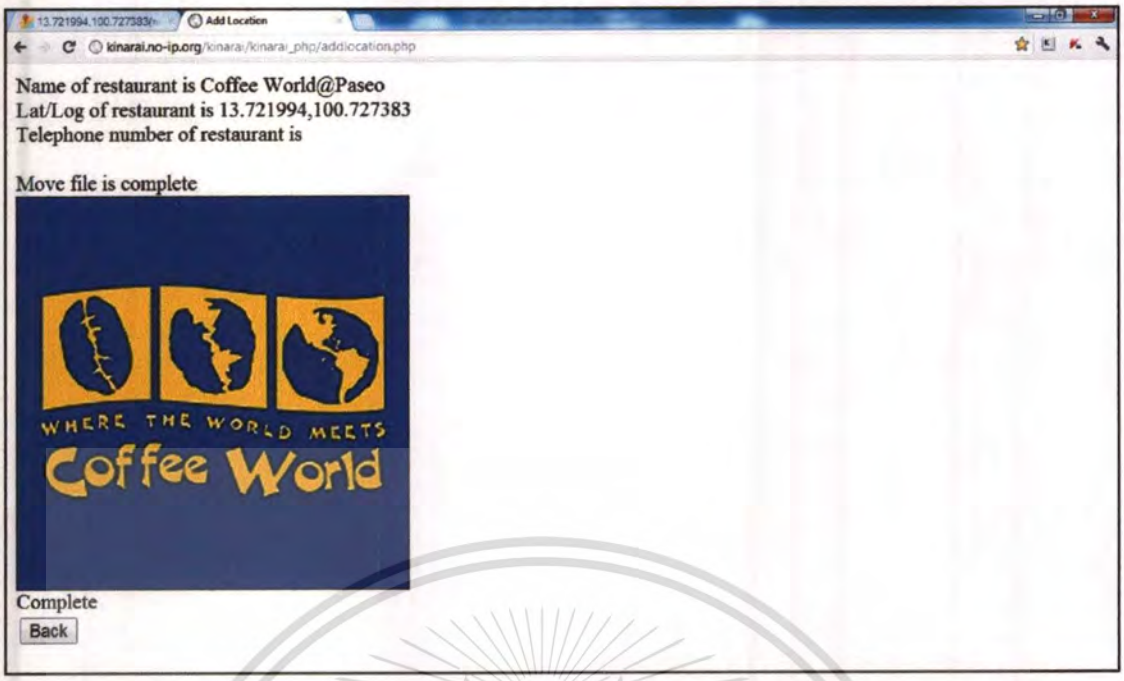


Figure 4.9 An example of a page when the location adding is completed

Figure 4.9 shows the result after the addition is completed. If there is more restaurants to be added, the user just clicks at the back button on the page.

4.1.3 Adding the filtered promotion to promotion table

This task can be divided into two main functions. The former is to filter the restaurant promotions from all promotions' information sent from web promotion sites. The latter is to add these filtered promotions into the promotion table. The idea is that keywords that are stored in the restaurant table will be used to search each feed item's title and description tags. If the keyword is found, the promotion item of those tags will be extracted and stored in the promotion table automatically. These functions refresh themselves every 30 minutes. Note that the administrator must open the web browser and enter the following URLs, since he starts the back-end process.

- http://kinarai.no-ip.org/kinarai/kinarai_php/promotion2u_v1.php
- http://kinarai.no-ip.org/kinarai/kinarai_php/promotion_v1.php
- http://kinarai.no-ip.org/kinarai/kinarai_php/proded_v1.php
- http://kinarai.no-ip.org/kinarai/kinarai_php/welovepro_v1.php
- http://kinarai.no-ip.org/kinarai/kinarai_php/goodpromoguide_v1.php

To illustrate, each link represents the URL of the php file, used to filter and extract the promotion from each web promotion site. The result of each file is the filtered promotion from the original promotion site (see Figure 4.10 to 4.14). In case that there is no promotion of any keyword found on the web promotion site, the result will be as shown in Figure 4.13 and 4.14 (no result).

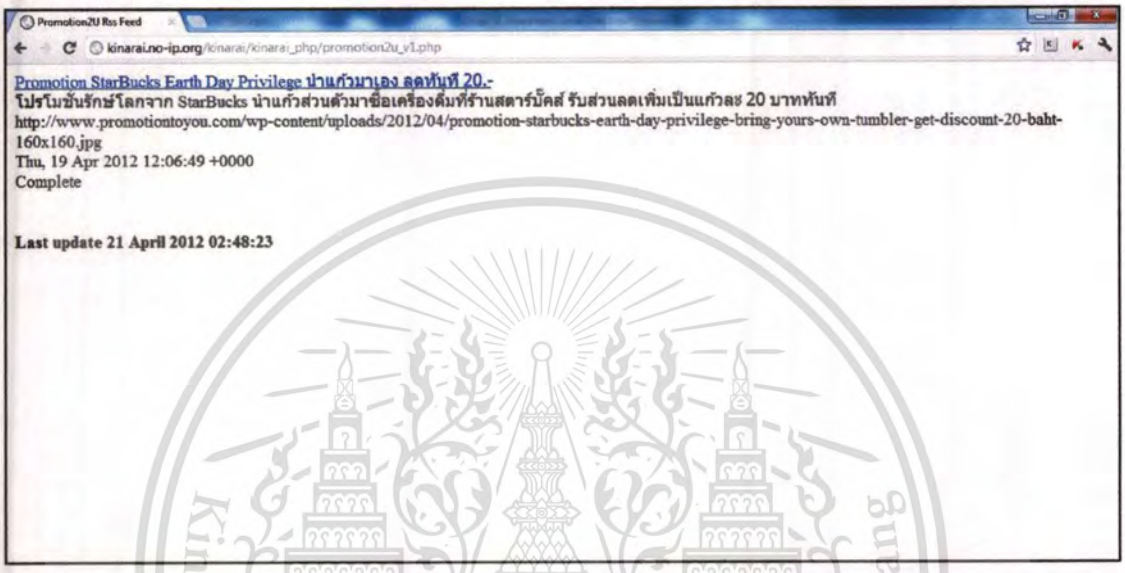


Figure 4.10 Promotion from promotiontoyou.com (file promotion2u_v1.php)

Figure 4.11 Promotion from promotion.com (file promotion_v1.php)

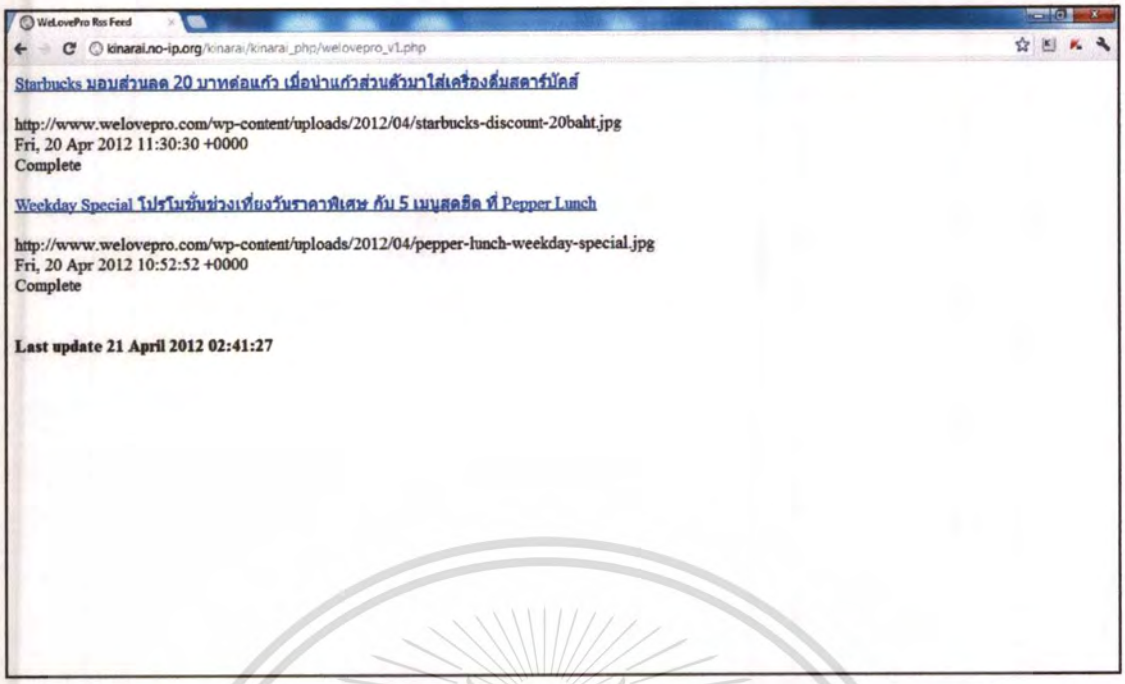


Figure 4.12 Promotion from welovepro.com (file welovepro_v1.php)

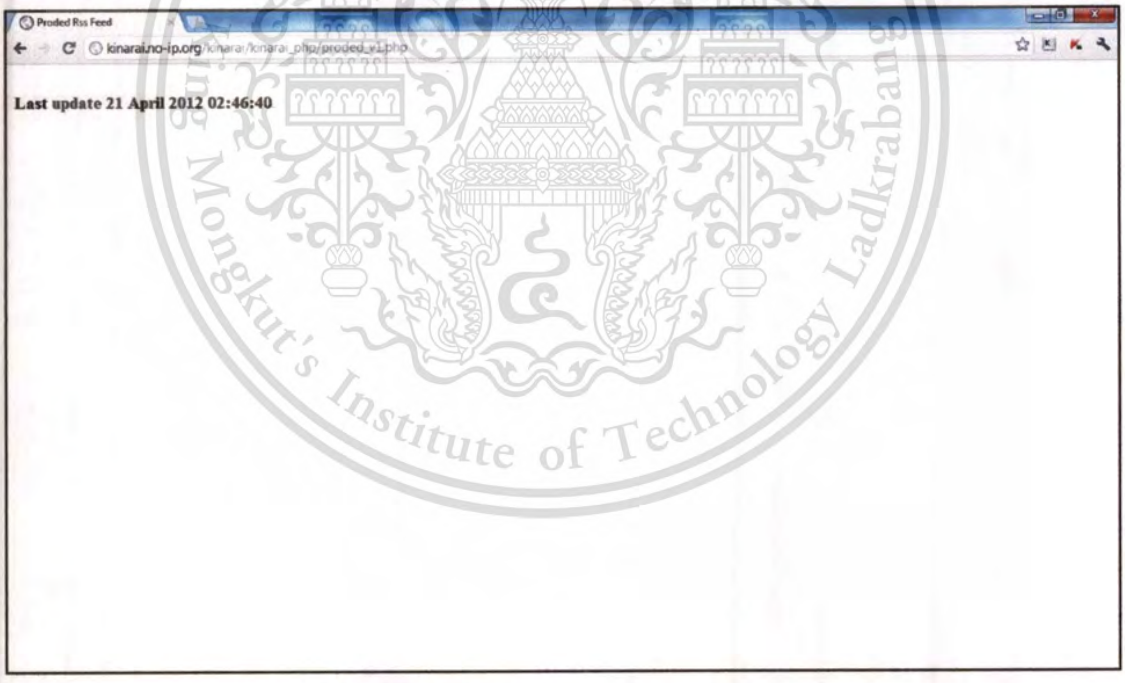


Figure 4.13 Promotion from proded.com (file proded_v1.php)

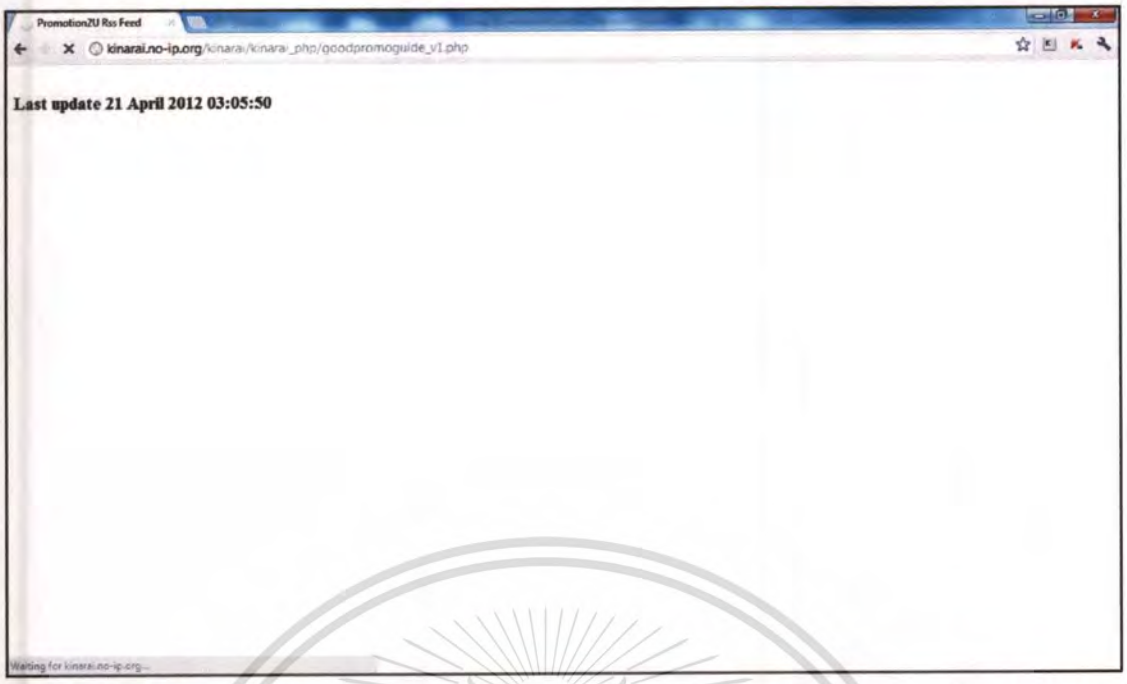


Figure 4.14 Promotion from goodpromoguide.com (file goodpromoguide_v1.php)

4.1.4 Providing information to android

Our server uses two mechanisms in providing information to the application on the android phones: through XML files and through JSON.

4.1.4.1 Creating the XML files for android

For the XML mechanism, the system stores necessary information into the XML file. The android goes to the file's URL and extracts the content of the file. The server uses the XML file interface for two purposes: to provide the restaurant information to the restaurant list view and to provide the promotion information of any chosen restaurant to the promotion list view. The restaurant information is stored in the *restaurants.xml* file. Each of its item has three components: the restaurant name, its logo and the URL of the promotion file. The promotion file stores each restaurant promotions. Each file is used each restaurant. The name of each file always starts with the word *test* followed by the restaurant id. For example, file *test1.xml* is for Mcdonald's promotion (Mcdonald's ID is 1).

We divide the process to create the XML file into two processes. The first process is to create the *restaurant.xml* file. The second process is to create the XML promotion files.

- The *buildrest.php* is the php file that the system uses to create the *restaurants.xml*. The buildrest.php must be run by the browser after the administrator has added new restaurant information into the restaurant table. Note that the buildrest.php URL is http://kinarai.no-ip.org/kinarai/kinarai_php/buildrest.php. Figure 4.15 shows buildrest.php code. Figure 4.16 shows the result of buildrest.php when it runs on the browser.

```
 6  <?php
 7  $cn = mysqli_connect("localhost", "root", "sawasdee");
 8  $cn->set_charset('utf8');
 9  mysqli_select_db($cn, "kinarai");
10  $sql = "SELECT nm_restaurant, img_logo, id_restaurants FROM restaurants;";
11  $result = mysqli_query($cn,$sql);
12      $data  = '<?xml version="1.0" encoding="utf-8"?>'."\n";
13      $data .= '<rss version="2.0">'."\n";
14      $data .= '<channel>'."\n";
15      $data .= '<title>Restaurant name</title>'."\n";
16      $data .= '<pubDate>'.date("D, d M Y H:i:s O").'</pubDate>'."\n";
17  while($rows = mysqli_fetch_array($result)){
18      $data .= '<item>'."\n";
19      $data .= '<restaurant>'.htmlspecialchars(stripslashes($rows[
    'nm_restaurant'])).'</restaurant>'."\n";
20      $data .= '<image>'.
    'http://kinarai.no-ip.org/kinarai/kinarai_php/imglogo/'.$rows['img_logo'].'
    </image>'."\n";
21      $data .= '<link>'.'http://kinarai.no-ip.org/kinarai/kinarai_xml/test'.
    $rows['id_restaurants'].'.xml'.'</link>'."\n";
22      $data .= '</item>'."\n";
23  echo stripslashes($rows['nm_restaurant']);
24  echo "<br/>";
25  echo $rows['img_logo'];
26  echo "<br/>";
27  echo "test".$rows['id_restaurants'].".xml";
28  echo "<br/><br/>";
29  }
30      $data .= '</channel>'."\n";
31      $data .= '</rss>'."\n";
32      $f = fopen( "restaurants.xml" , 'w' );
33      fputs( $f , $data );
34      fclose( $f );
35  mysqli_close($cn);
36  ?>
```

Figure 4.15 *buildrest.php*'s code

- Figure 4.15 shows the code to generate the *restaurants.xml* file. The codes from line no. 7-9 are about the database connection and the table selection. Line no. 10-11 shows SQL statement used to query the restaurant table which the result is kept in $result. The XML file has to start with the header like one in line no. 12. Codes from line 12 to 16 shows how to generate the XML header for this

file. The line no. 17 shows how to fetch the data from nm_restaurant, img_logo and id_restaurants fields from restaurants table. Then the line no. 18-22 shows the method to concatenate string (data$) from the data field nm_restaurant, image_logo, id_restaurants that fetched from the restaurant table to content between restaurant tags, image tags and link tags respectively. To illsturate, for ***<restaurant>'.htmlspecialchars(stripslashes($rows ['nm_restaurant'])).'</restaurant>***, the $rows['nm_restaurant'] means to fetch data from the nm_restaurant field. The stripslashes() function is used to remove backslashed added by the addslashes() function. The last function which is *htmlspecialchars*() function converts some predefined characters to HTML entities such as & (ampersand) becomes &amp. Note that the content in XML file must consist of the open tag and close tag. Line no 32 shows the part to create an XML file with fopen() function. The first parameter of fopen() function at line no 32 means the file name. The second parameter w means opening for write only. The fputs() function is used for writing the content into the file at the line no. 33. After writing the content successfully, close the file with fclose() function (line no.34).

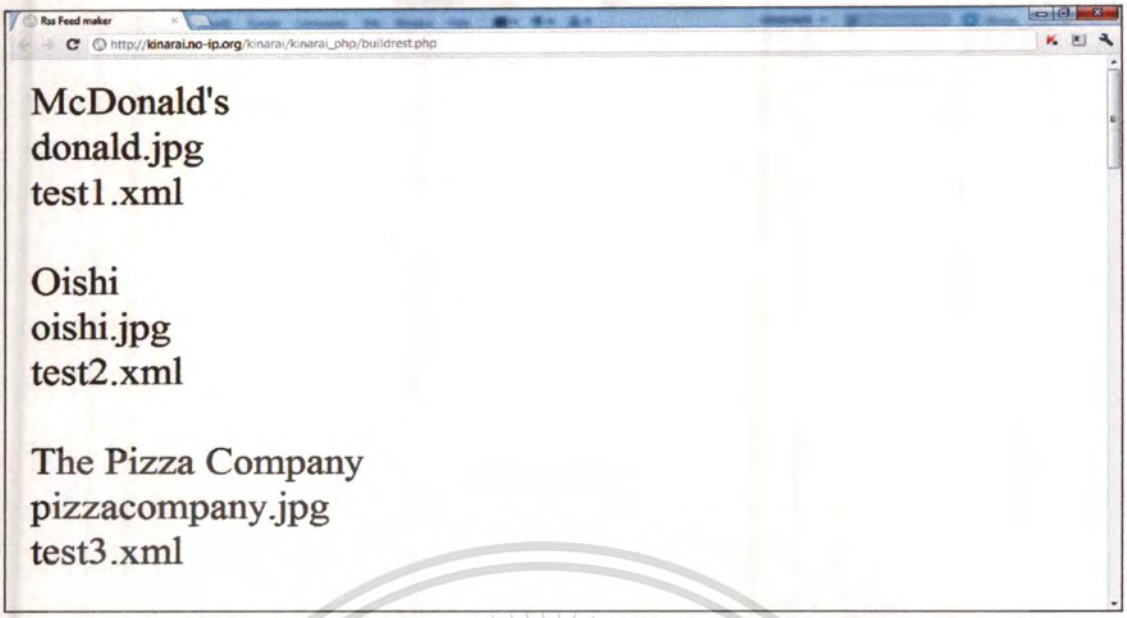


Figure 4.16 Result of *buildrest.php* when run on the browser

- The *mkxml.php* file is used to create the XML promotion files. It takes the promotion information from the promotion table and then creates all promotion files. This file will run automatically every 30 minutes. To operate this file, the system has to keep opening the link on the server's browser. The file URL is http://kinarai.no-ip.org/kinarai/kinarai_xml/mkxml.php. Figure 4.18 shows the result of *mkxml.php* when it runs on the browser.

```php
<?php
$cn = mysqli_connect("localhost", "root", "sawasdee");
$cn->set_charset('utf8');
mysqli_select_db($cn, "kinarai");
$sql = "SELECT DISTINCT id_restaurants FROM promotions;";
$result = mysqli_query($cn,$sql);
while($rows = mysqli_fetch_array($result)){
echo $rows['id_restaurants'];
echo "<br/>";
    $data  = '<?xml version="1.0" encoding="utf-8"?>'."\n";
    $data .= '<rss version="2.0">'."\n";
    $data .= '<channel>'."\n";
    $data .= '<title>xxxxxxxxx rss</title>'."\n";
    $data .= '<description>Rss XML</description>'."\n";
    $data .= '<link>http://www.webthaidd.com</link>'."\n";
    $data .= '<pubDate>'.date("D, d M Y H:i:s O").'</pubDate>'."\n";
$sql2= "SELECT * FROM promotions WHERE id_restaurants = ".$rows[
'id_restaurants']." ORDER BY id_promotions DESC;";
$result2 = mysqli_query($cn,$sql2);
    while($rows2 = mysqli_fetch_array($result2)){
    echo $rows2['title'];
    echo "<br/>";
    echo $rows2['description'];
    echo "<br/>";
    echo $rows2['image'];
    echo "<br/>";
    echo $rows2['link'];
    echo "<br/>";
    echo $rows2['pubdate'];
    echo "<br/>";
    echo "--------------------<br/>";
    $data .= '<item>'."\n";
    $data .= '<title><![CDATA['.html_entity_decode(stripslashes($rows2[
'title'])).']]></title>'."\n";
    $data .= '<description1><![CDATA['.html_entity_decode(stripslashes(
$rows2['description'])).']]></description1>'."\n";
    $data .= '<image>'.html_entity_decode(stripslashes($rows2['image'])).'
</image>'."\n";
    $data .= '<link>'.html_entity_decode(stripslashes($rows2['link'])).'
</link>'."\n";
    $data .= '<pubDate>'.html_entity_decode(stripslashes($rows2['pubdate'
])).'</pubDate>'."\n";
    $data .= '</item>'."\n";
    }
    $data .= '</channel>'."\n";
    $data .= '</rss>'."\n";
    $f = fopen( "test".$rows['id_restaurants'].".xml", 'w' );
    fputs( $f , $data );
    fclose( $f );
}
mysqli_close($cn);
?>
```

Figure 4.17 *mkxml.php*'s code

- Figure 4.17 shows the code used to generate the promotion XML file, which is quite similar to code in Figure 4.15.

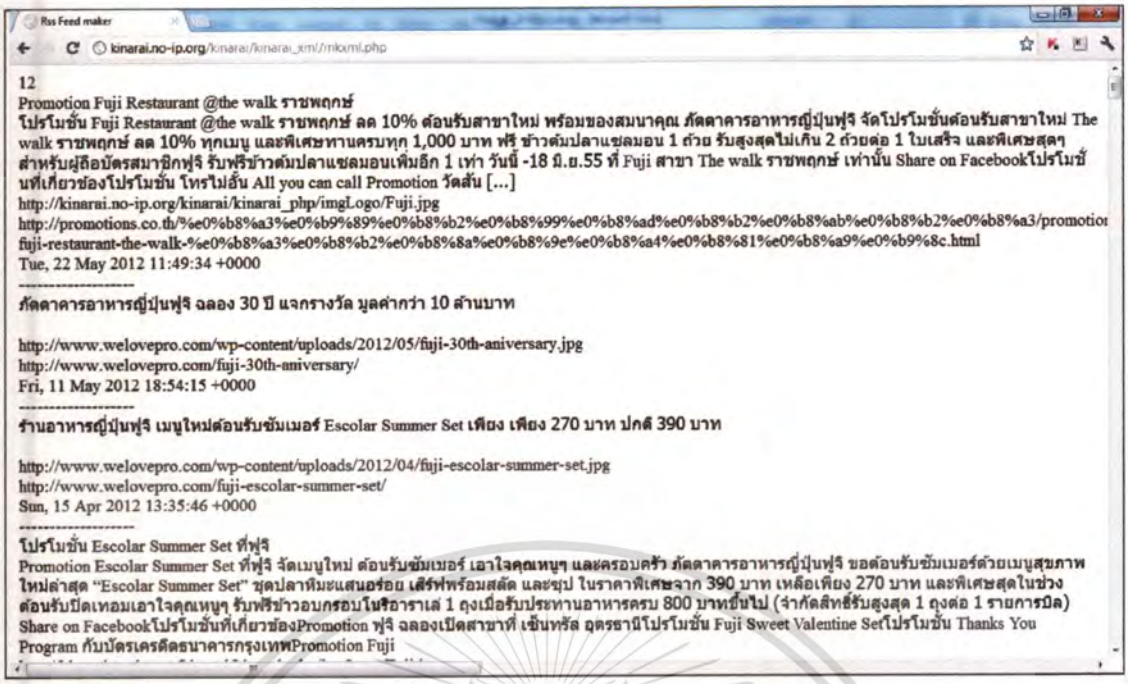


Figure 4.18 Result of *mkxml.php* when run on the browser

4.1.4.2 Creating the JSON object for android

For JSON, the system provides information to android by using HttpGet method. JSON performs the task according to the triggering word in the URL, which is the word that followed *?v=* . Our system uses the word *json* as the triggering word for the location service. That is, when our system receives http://kinarai.no-ip.org/kinarai/kinarai_service/service.php?v=json service, the server extracts the word *json*. Hence, it knows that there is a request for the restaurant location data from the database. Note that after querying data from the database, the system has to encode the data into JSON format before sending to the android. See Figure 4.19 for the implementation code for JSON's location service.

```
2  if ($_GET["v"]==""){
3      exit;
4  }
5  if($_GET["v"] == json){
6   $sql = "SELECT * FROM location_rest";
7   } else {
8   $sql = "SELECT * FROM location_rest WHERE location LIKE '%{$_GET["v"]}%'
   or nmrestaurants LIKE '%{$_GET["v"]}%' ";}
9   $conn = ConnectMySql();
10  $arr = Array();
11  $result = mysql_query($sql);
12  $i = 0;
13    while($row = mysql_fetch_array($result)){
14      $obj = null;
15      $obj->id = trim($row["id"]);
16      $obj->name = trim($row["nmrestaurants"]);
17      $obj->lat = trim($row["lat1"]);
18      $obj->long = trim($row["long1"]);
19      $obj->pic = trim($row["pic"]);
20      $obj->tellnum = trim($row["tellnum"]);
21      $obj->location = trim($row["location"]);
22      $arr[$i] = $obj;
23      ++$i;}
24 CloseMySql($conn);
25 return $arr;
26 $arr = array(data => $Arrobj);
27 echo json_encode($arr);
```

Figure 4.19 JSON location service's code

- Figure 4.19 shows the implementation code our location service using JSON. Line no.2-5 are a part to get the service value from GET method. If the value is null, the program will not query the data from the database or do nothing. If the android sends the service with *json* word, the program will query the data from database (line no.5) Line no.9 and 11 are about the database connection and querying. Fetching data from the database is started at line 13. Line no. 15-21 shows how to fetch data from each field from location_rest table and store into each tag value. Line no. 24 shows how to disconnect from the database. To encode the data into JSON format, see Line no. 25-27.

4.1.5 Getting the location of restaurants

The location of the restaurant is obtained from the Google Map on the android device. The location's URL will be sent via email to the administrator. The administrator then opens the location's URL from his email using the web browser in which he will find an accurate latitude and longitude of the restaurant. These values

are then kept and entered into the restaurant location table through the page in Figure 4.8.

4.1.5.1 Getting latitude and longitude from the Google Map

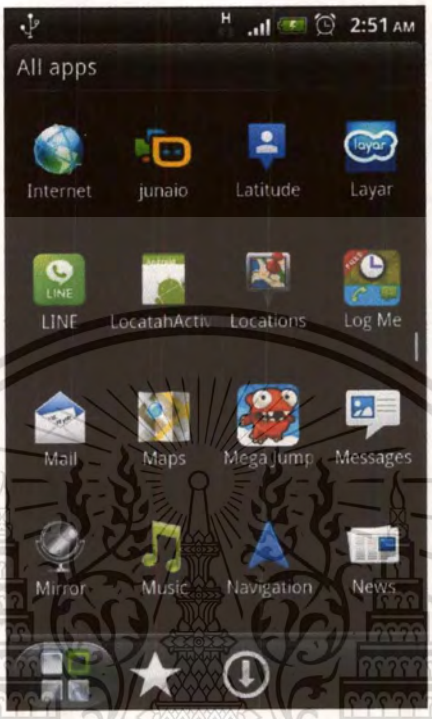


Figure 4.20 Main Menu

Figure 4.20 shows the main menu of our android device. To open the **Maps** application, the administrator selects the Maps icon.

Figure 4.21 Maps application

The screen similar to one in Figure 4.21 will appear. Note that, the map that is shown depends on the device's current location which might be changed each time. Since the Google Map allows users to move on the map, the user can navigate on the map to the location of the restaurant that he wants.

Figure 4.22 Location of restaurant on the map

When his target restaurant is spotted on the screen, the user simply touches the screen at that particular point for 2 seconds which the pop up as in Figure 4.22 will appear.

Figure 4.23 Maps application: Menu

Figure 4.23 shows Maps application option menu. The **share this place** menu is selected. Maps application allows its user several ways to share his location with other applications. In this case share this place via email by **Gmail** is selected (see Figure 4.24).

Figure 4.24 Maps application: share this place menu

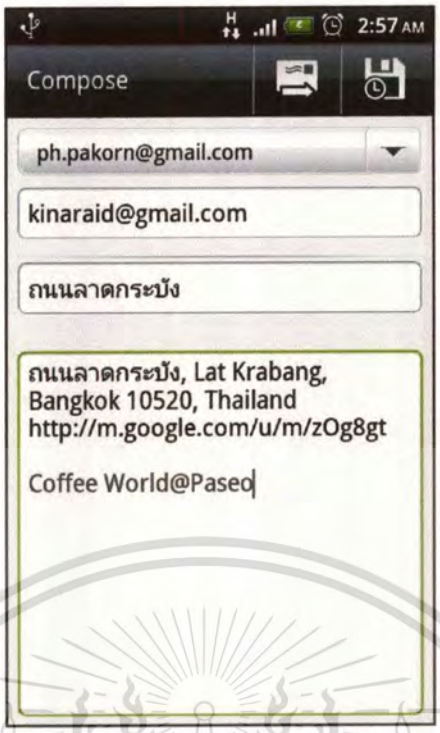


Figure 4.25 Share this place via email

After selecting share this place via email, the screen as in Figure 4.25 will display the detail of URL's location. If there is more detail to add such as the restaurant's name or its telephone number, it can simply be done by typing the extra information in the text box message. The picture of the restaurant can be attached in this process as well. Finally, the recipient email address must be entered before pressing the send button.

4.1.5.2 Getting URL of latitude and longitude by email

When the URL's location of the restaurant has been sent to the receiver inbox, he has to log in with his username and password into http://www.gmail.com as shown in Figure 4.26.

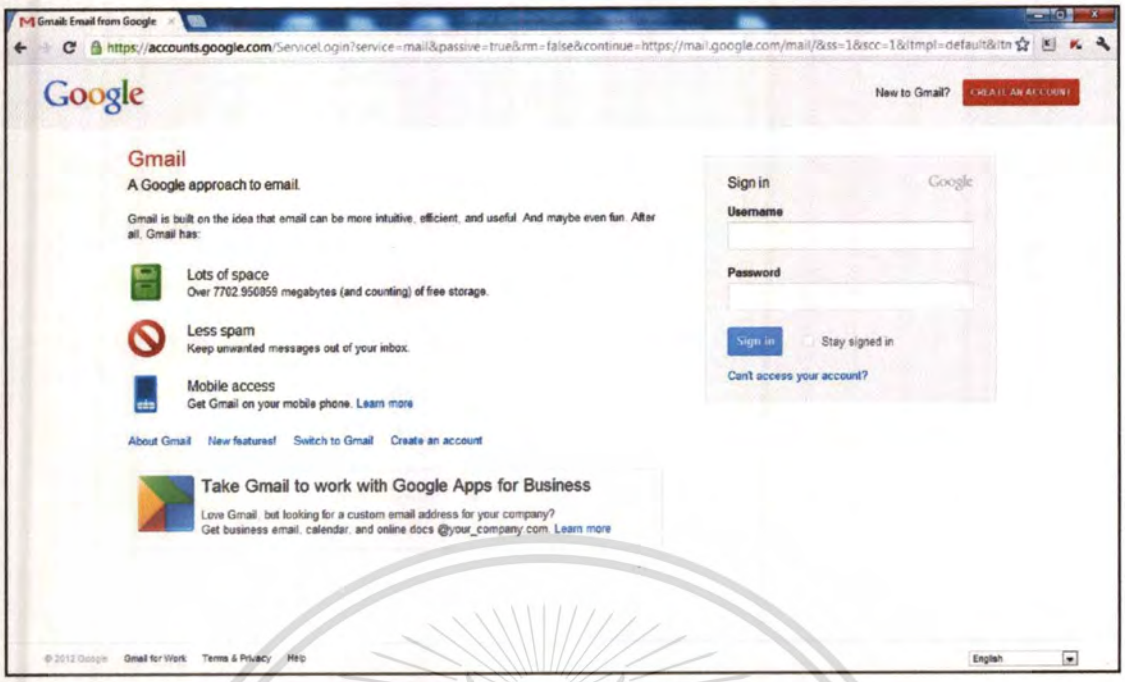


Figure 4.26 Gmail page

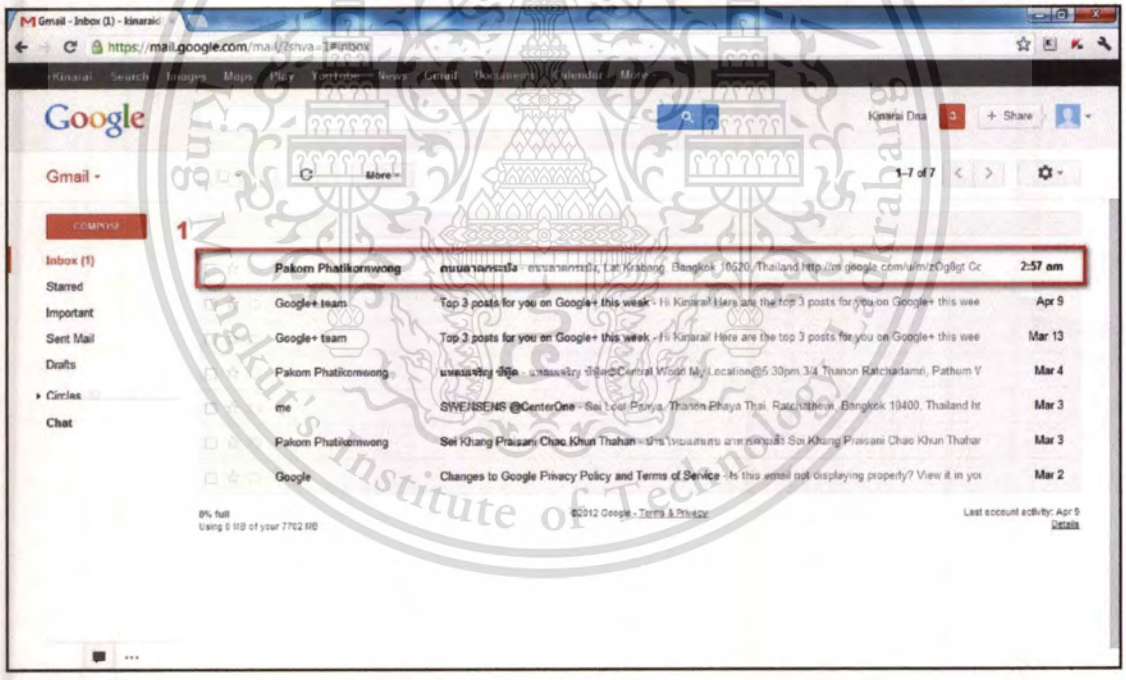


Figure 4.27 Inbox

Then he goes to read his new email as shown in the red frame in Figure 4.27.

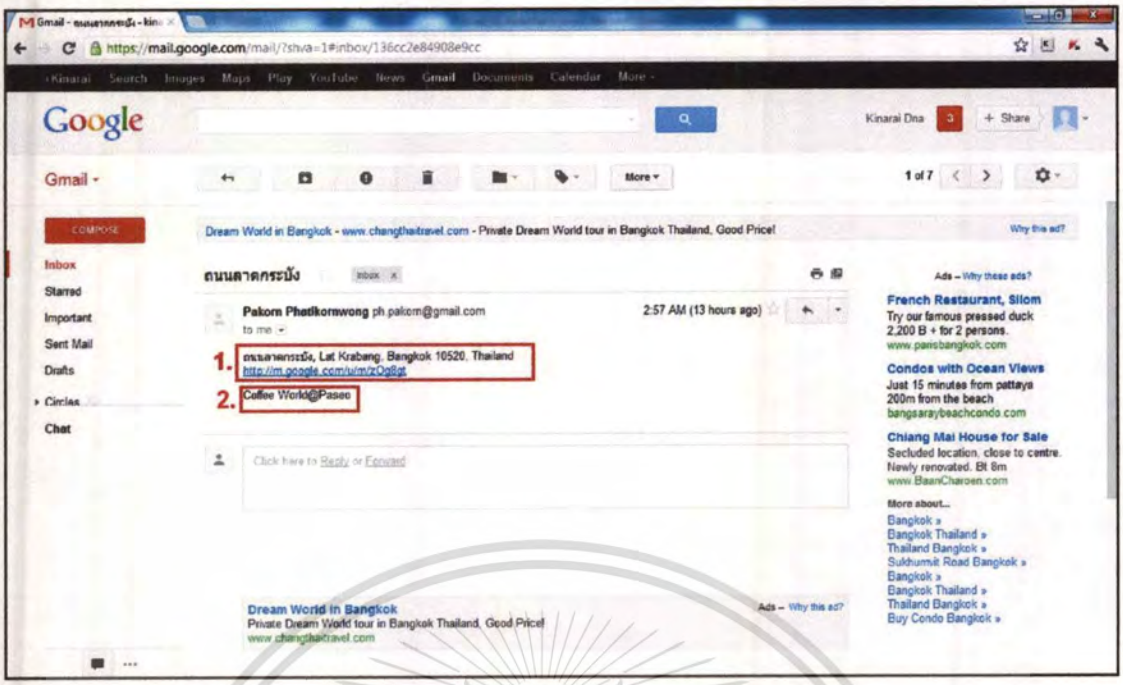


Figure 4.28 Location Details (in email)

Figure 4.28 shows the location information. The red frame in number 1 is the URL to get latitude and longitude value. The red frame in number 2 is the restaurant's name. The user then clicks at the link in the red frame in number 1 to open the Google MAP.

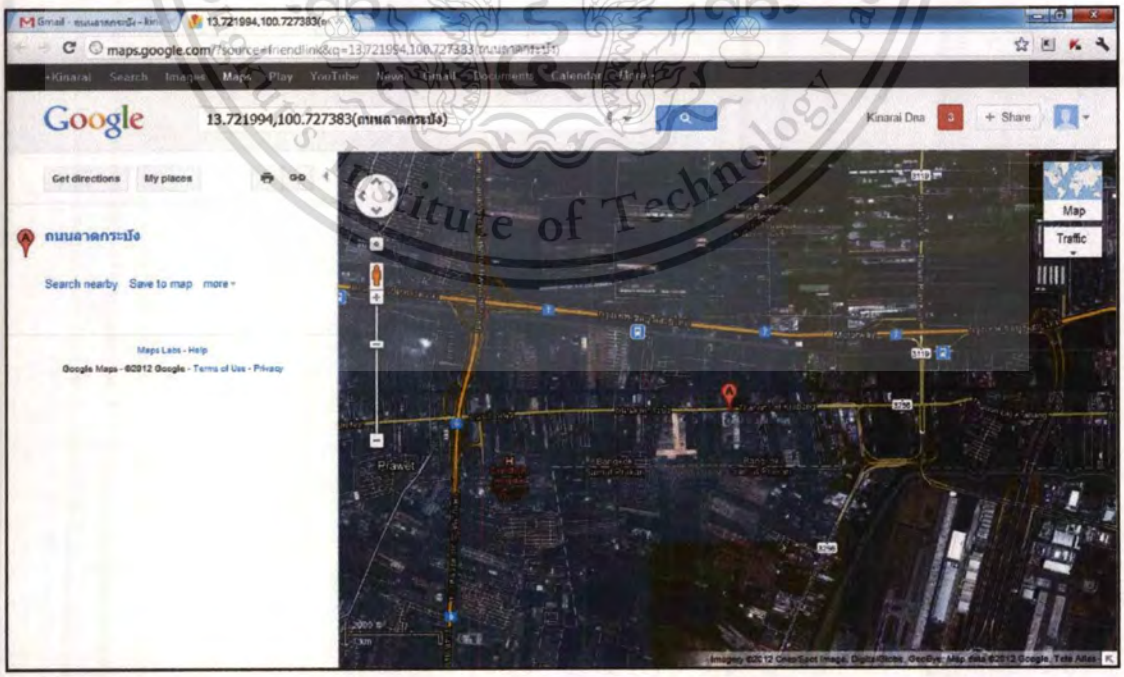


Figure 4.29 Map of the marked restaurant

After opening the link in Figure 4.28, the screen will display the map as seen in Figure 4.29. The marker A is the location of restaurant that has been marked from android device.

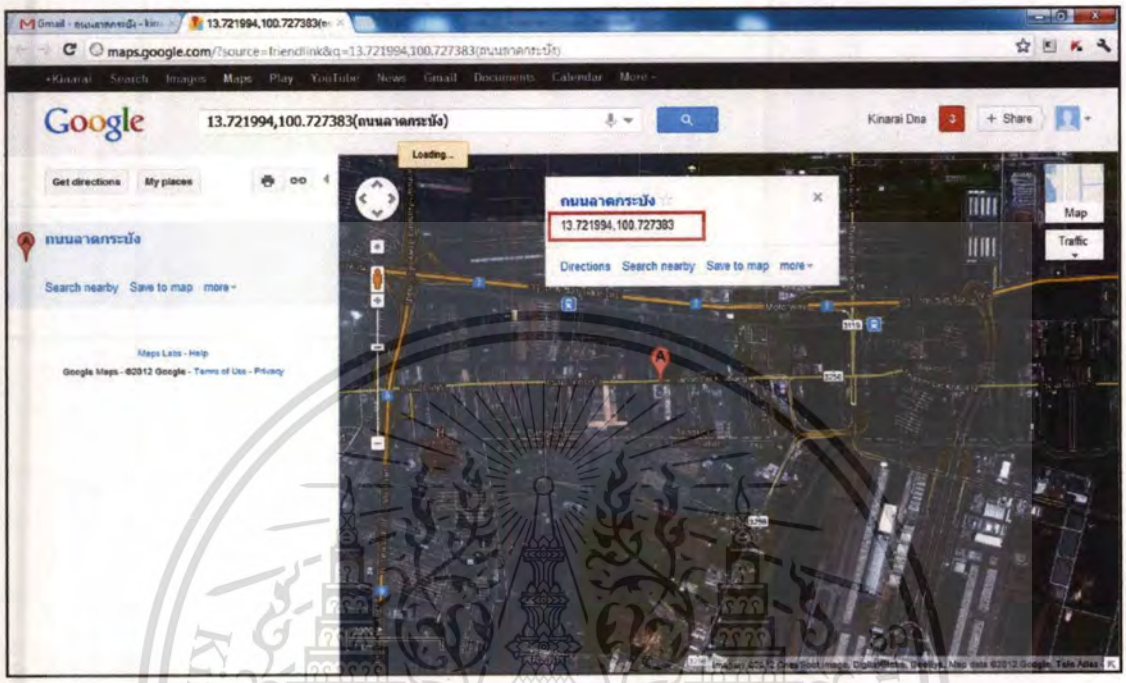


Figure 4.30 Getting latitude and longitude of the marked restaurant

When the user clicks on the marker A, the pop up will appear to show the latitude and longitude of the restaurant. In conclusion, the user will get latitude and longitude from Figure 4.30 and get restaurant's name from Figure 4.28.

4.2 Front-End

The **Kinarai** application have 2 main functions at the front-end. The first one is the promotion function which provides the list of latest restaurants' promotions in the list view. The second function is the nearby function which uses GPS to get the user current location and provides the list of restaurants located near the user (in the list view and map view).

When the users open Kinarai application, the main menu screen as shown in Figure 4.31 will appear. In the main menu, there are two icons: the icon on the left

(with fork and spoon) is for the promotion function where the icon on the right with the green radar is for the nearby function.

Figure 4.31 Main menu screen

4.2.1 Promotion function

The aim of the promotion function is to obtain the list of selected restaurants and show their promotions. We display this function in the following order: the list of restaurants (Figure 4.32), its promotions (Figure 4.33), the promotion in more detail (Figure 4.34) and the original page of web promotion site that this promotion is from (Figure 4.35). Note that the user can get back to the previous screen every time by pressing the phone's back button.

The followings are the brief description of each screen in the promotion function. After touching the promotion icon, the screen as shown in Figure 4.32 will appear. This screen shows a list of restaurant names and their logos that are obtained from the server.

Figure 4.32 List of the restaurants

Users can simply touch at the item that they are interested in to see whether their target restaurant has any promotion. When the users touch at their target restaurant, the program may take a few seconds before the target restaurant promotion page as in Figure 4.33 will appear.

Figure 4.33 List of the selected restaurant (McDonald) promotions

To see more detail of their interest promotions, the users touch at that promotion list then the page as in Figure 4.34 will appear.

Figure 4.34 More detail page (Mcdonald's drive thru)

If the users want to see more of their selected promotion (see Figure 4.34), the users can press More Details button, which will redirect the program to the original web promotion site as an example shown in Figure 4.35.

Figure 4.35 Promotion's original page from web promotion site (www.promotions.com)

In terms of implementation, the android client obtains the restaurant information and promotion from the web server through the XML files stored on our application server. First, android obtains the list of restaurant names and their logos used in Figure 4.32 from http://kinarai.no-ip.org/kinarai/kinarai_php/restaurants.xml. Each item of the *restaurants.xml* file (as shown in Figure 4.36) stores three components: the restaurant name, its logo and the URL of the XML file, which each of this XML file stores promotion's information for each restaurant so that when the user wants to see the promotion of each restaurant, the promotion information in this file is extracted and put in the list view. An example of this type of XML file *(test1.xml for McDonald's)* is shown in Figure 4.37. Note that the system names the promotion file by starting the file name with the word *test* and followed by its restaurant id.

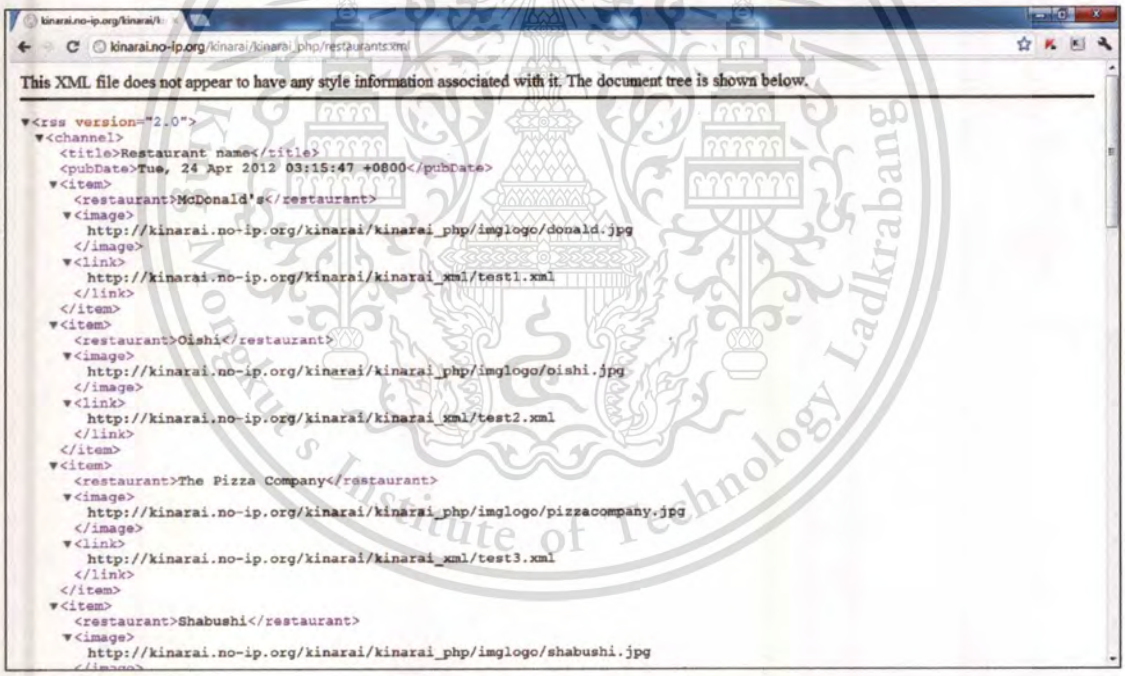


Figure 4.36 XML file of *restaurant.xml*

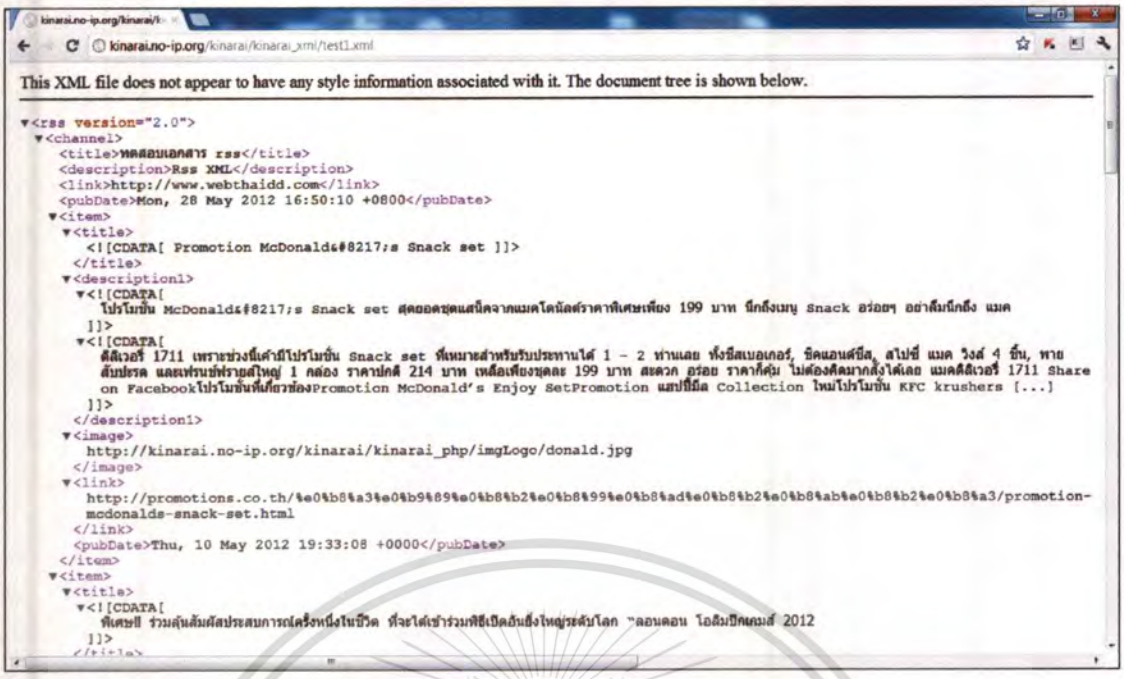


Figure 4.37 XML file of *test1.xml*

See Section 3.3.2.4 for the restaurant extraction and the promotion extraction mechanisms.

4.2.2 Nearby Function

The aim of the nearby function is to obtain the list of restaurants which are located near the user's current location. This function also allows the user to choose his interesting district or landmark such as Latkrang and Centralworld to see the list of the restaurants around that place including their maps. In addition, this function also allows the user to search for the restaurant by its name. We display this function in the following order. First, the users must select which option that they want to search (distance/place or restaurant name selection) (Figure 4.38). If the user select the distance options, the list of restaurants around the place as in Figure 4.40 will appear which will be followed by the screen to display that selected restaurant's map (Figure 4.41). For the place or name option, the screen as in Figure 4.39 will appear, in which the user can touch on the Map view tab on the top to change the display from the list view to the map view (Figure 4.41).

The followings are the brief description of each screen in the nearby function. First once the users touch at the nearby icon located on the right in the main menu, the distance screen as shown in Figure 4.38 will appear. This screen provides three options since the application allows users to choose the nearby place in three alternative ways. The first two are by distance selection. For the first option, users specify any distance in kilometers that they want (see Figure 4.38). For the second option, users select the distance (1-10 kilometers) through the dropdown list (see Figure 4.39).

Figure 4.38 Distance selection

Figure 4.39 Distance selection in dropdown list

The system then will display the list of the restaurants located within selected distance as an example shown in Figure 4.40. For each restaurant item in the list, there are four components: the restaurant logo (on the left), the restaurant name (in the middle), its telephone number (on the upper right) and the distance in kilometers from the user location (on the lower right).

Figure 4.40 List of the restaurants

To see the map of the restaurant that are they are interested, the users touch at that restaurant item and the map of the restaurant will appear (similar to one in Figure 4.41). The position of the restaurant is marked with the red bubble. The position of the user (phone) is marked with the blue bubble. Note that the application will always display the map with the selected restaurant in the middle of the map.

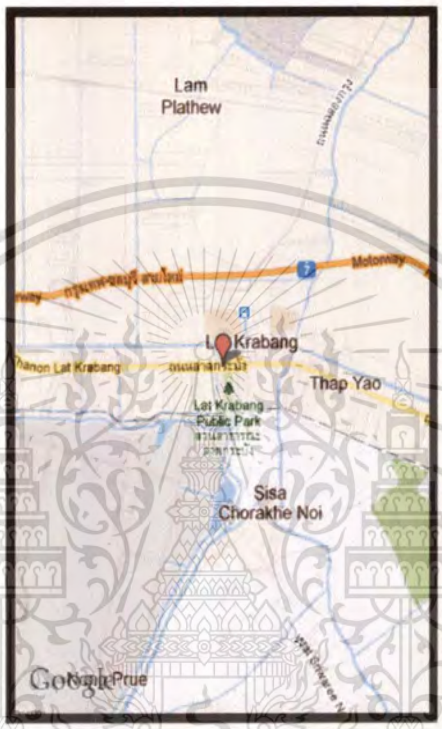


Figure 4.41 The restaurant map view screen

For the third option, users enter a name or a place (see Figure 4.42). Then, the screen as in Figure 4.43 will appear. In this screen, there are two tabs on the top, which the users can switch between the list view and the map view. The first tab is the list view tab which is the default display. Under the list view tab is the list of the restaurants which consists of the restaurant logo (on the left), the restaurant name (in the middle), its telephone number (on the upper right) and the distance in kilometers from the user location (on the lower right). For the map view of this selection's type, all restaurants in the area will be shown on the map. The positions of the restaurants are marked with the red bubble. When the user clicks at any red bubble (choose specific restaurant), the map will change the screen to the map view similar to one in

Figure 4.44, which will display the map with the selected restaurant in the middle with its location in latitude and longitude values.

For the nearby function's implementation, the program obtains the list of all restaurants information from the web server through JSON service. Note that this information is indeed the data in the restaurant location table, which is re-loaded every time the user enters the nearby function menu. This information is stored in the array list. See Section 3.3.3.2 for the mechanism to connect to the restaurant service and how to load the restaurant information to store in the array.

For the distance finding options, the android client will first convert the current location of the phone from latitude and longitude to kilometer. This is because our program uses the GPS device of the android phone to obtain its location which returns location values in latitude and longitude format but the application uses distance in kilometer format. Our program then converts the location of all restaurants obtaining from JSON which temporarily stored in array to kilometers and uses them to compare with the current location (in kilometers). Then, only the restaurant within the specified distance will be shown in the restaurant list view. See Section 3.3.3.9 for the mechanism to convert from the latitude and longitude to kilometer location and Section 3.3.3.8 for the mechanism to get the nearby restaurants and put them in the list view.

For the area or restaurant name or place selected option, the android client will use autocomplete textview function in android library to check the characters that the user types. The mechanism to implement the autocomplete item is shown in object onCreate in Section 3.3.3.4. The program then compares the restaurant and location names in each JSON object with the name that the user has typed or selected and put the matched objects in the list view. See Section 3.3.3.8 for the mechanism used in implementing this part.

Figure 4.42 Autocomplete text

To display the map as in Figure 4.41, the program will take the latitude and longitude of the selected restaurant and sends these values to the Google Map API. The Google Map API then will send back the map. As previously explained in Section 3.3.3.12, MapView library in android is used to communicate with the Google Map API. For our application, we set the zoom value to 17 (there are 21 levels for zooming Google map). See Section 3.3.3.13 for more details of the mechanism used to implement the map view part.

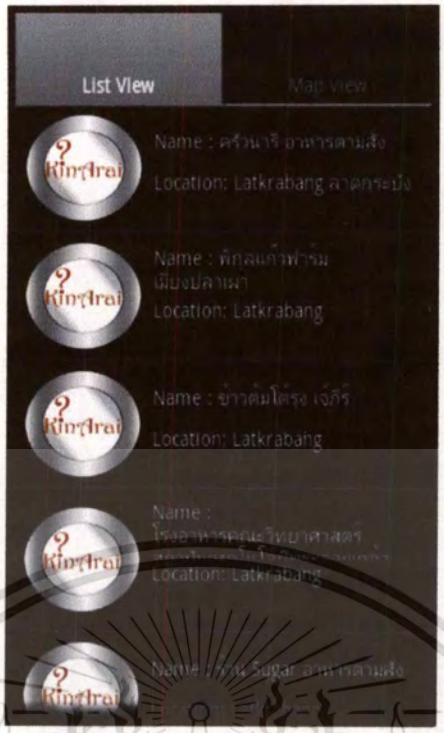


Figure 4.43 Nearby restaurants in list view

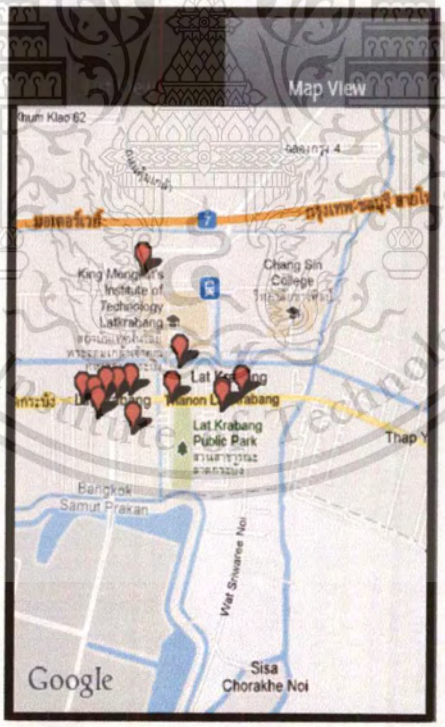


Figure 4.44 Nearby restaurants in map view

Chapter 5

Conclusion and Suggestion

5.1 Conclusion

The two main objectives of our application, KinArai Application, is to help the user to search for the restaurant's promotion and to help the user to search for restaurants that located nearby then provides the map to go to that particular restaurant. Our application is designed and implemented specially for the Android mobile devices with the OS version 2.3.3.3 and above. It is best fit on the screen with resolution WVGA 800x480. The system consists of two parts: the back end part which deals with the web server and the database functions and the front-end part which is the KinArai application on android device. For the back end part, we used appserv (Apache, PHP MySQL and phpMyAdmin) for the web and the database server. Our system obtains the promotion information from 5 web promotion sites' feed: www.goodpromoguide.com, www.proded.com, www.welovepro.com, www.promotion.com and www.promotiontoyou.com. XML and JSON's formats are used in the message exchange formats between android devices and the web server (for server's services). Google API is used to obtain the map from the Google Map website which will use the GPS device from the phone for the user location.

5.2 Limitation of our application

- When the application connects to the server to retrieve the XML files, the connection was too slow because our server is set up at home with the Internet upload bandwidth at only 512 KB.

- Sometimes our application maybe slow because for each promotion item in the promotion list view, our application has to download each promotion image from its source at the web promotion site and then resizes it to fit the image layout (75x75 dpi). If the size is large, it takes time to resize.

- If the user wants to use nearby function, he has to make sure that the GPS on his phone is already enable because if our application has not received the signal from the GPS device, it will set the current location to the default location which is King Mongkut's Institute of Technology's location.

- The accuracy of GPS on the android phone depends on the quality of the GPS device attached with the phone. Hence, if our application is run on the phone with low quality GPS device, the value of the user current location which will be used to calculate the distance between the phone and the restaurant will not be correct.

- There is a problem for the promotions that are expired since the XML feed from each promotion site that we use do not specify the expiry date tag. Even though the administrators might put the promotion duration in the description tag, it is hard to extract since they enter it with no exact format. Hence, at this stage, our system does not provide the mechanism to purge the old promotion items which if the application is used for a while, it is possible that many promotion items shown on the promotion list are outdated.

5.3 Suggestion and Future work

- For better performance and stability, we suggest using a server which installs the OS for server such as Microsoft Windows Server or Linux server (our system used Windows 7).

- Our server IP might be changed when the Internet connection is down or the router is restart because our server network is behind the ISP home broadband connection that assigns an IP dynamically. To solve this problem, our server subscribes to no-ip service which provides a program to install on our server. This program updates the current server's IP to the no-ip website every 5 seconds. The no-ip service also provides the DNS name for our application (kinarai.no-ip.org). We note that if the server is implemented with static IP. There is no need to use the no-ip service. The application can simply use the web server's DNS or the server's IP address.

- For more widely use, we suggest to implement the application on iOS devices.

- The .php files that control processes on the back-end should be run as the background process for better security and memory usage.

- We suggest for the future work to remove the outdated promotions. One alternative is to check with the web promotion sites if the content of each promotion that we took from is still available, if not this promotion should be removed from the promotion table. This can be done by using the value in field *link* in the promotion table (this field stores the original link of the promotion) and check if the link is still available. Nevertheless, some webs do not delete the URL files even though there is no content. Hence, another suggestion is comparing the string of the field *title* in promotion table to the html *title* of the promotion file content. If it is not found, it is likely that this promotion is removed which allows the system to delete this promotion record from the promotion table.

References

[1] C. So-In, P. Chanjoi. **Basic android app development.** SE-ED Thailand, August 2010

[2][Online] http://www.webopedia.com/TERM/G/gingerbread.html
http://developer.android.com/guide/basics/what-is-android.html

[3][Online] http://www.webopedia.com/TERM/A/android_sdk.html

[4][Online] http://www.press-feed.com/howitworks/rss_tutorial.php
http://astonfunds.com/rss
http://in.answers.yahoo.com/question/index?qid=20100117061601AA7JCDp

[5][Online] http://www.w3schools.com/xml/
http://en.wikipedia.org/wiki/XML
http://www.itsabacus.com/xml.html

[6][Online] http://en.wikipedia.org/wiki/PHP
http://www.linkedin.com/groups/PHP-is-generalpurpose-serverside-scripting-4077799.S.72211275

[7][Online] http://www.w3schools.com/php/default.asp

[8][Online] http://en.wikipedia.org/wiki/Apache_HTTP_Server
http://httpd.apache.org/

[9][Online] http://en.wikipedia.org/wiki/MySQL
http://www.mysql.com/
http://www.mysql-thailand.com/

[10][Online] http://www.phpmyadmin.net/home_page/index.php

References (Cont.)

[11][Online] https://developers.google.com/

[12][Online] https://developers.google.com/maps/mobile-apps

[13][Online] https://developers.google.com/maps/articles/geolocation

[14][Online] https://developers.google.com/maps/documentation/places/

[15][Online] http://www8.garmin.com/aboutGPS/
http://en.wikipedia.org/wiki/Global_Positioning_System
http://www.global5thailand.com/thai/gps.htm

[16][Online] http://en.wikipedia.org/wiki/Location-based_service
http://www.it24hrs.com/2011/location-based/

[17][Online] http://www.json.org/

[18][Online] http://en.wikipedia.org/wiki/Sequence_diagram

Appendices

Appendix A.

Appserv Installation

How to install AppServ

1. Download program AppServ from this website http://www.appservnetwork.com
2. Double click at file **appserv-win(32/64)-x.x.x.exe** to install the program (see Figure A.1).

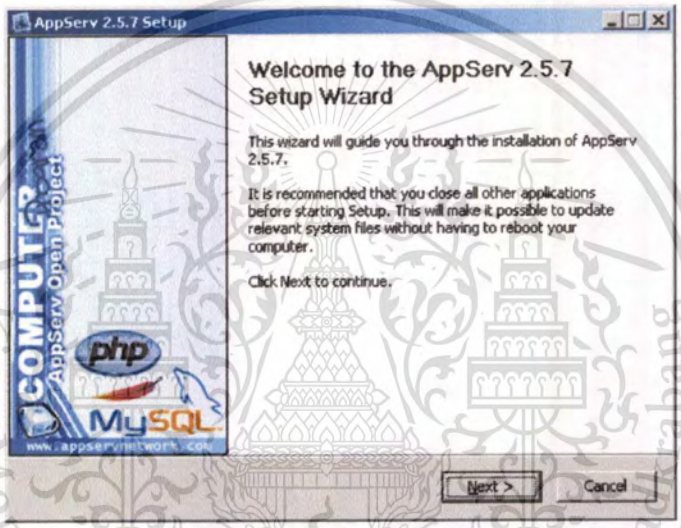


Figure A.1 AppServ installation

3. Click **Next** button, the license agreement window will appear (see Figure A.2)

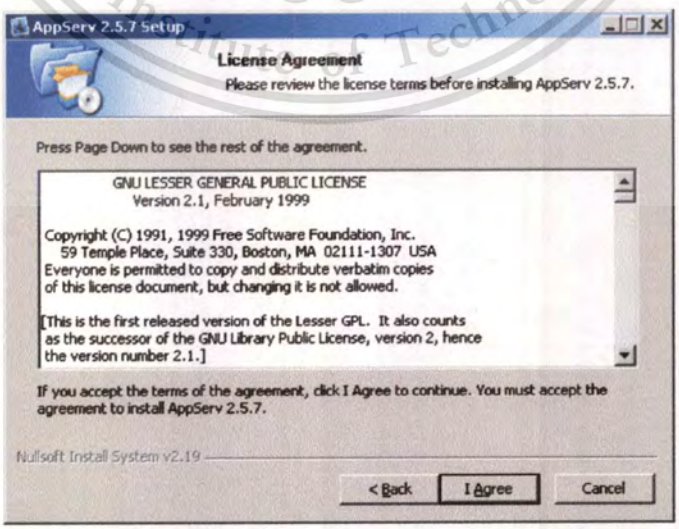


Figure A.2 License Agreement's window

4. Click **I Agree** button. The program will ask the destination folder to store the program. After finish selecting the destination folder, click **Next** button (see Figure A.3).

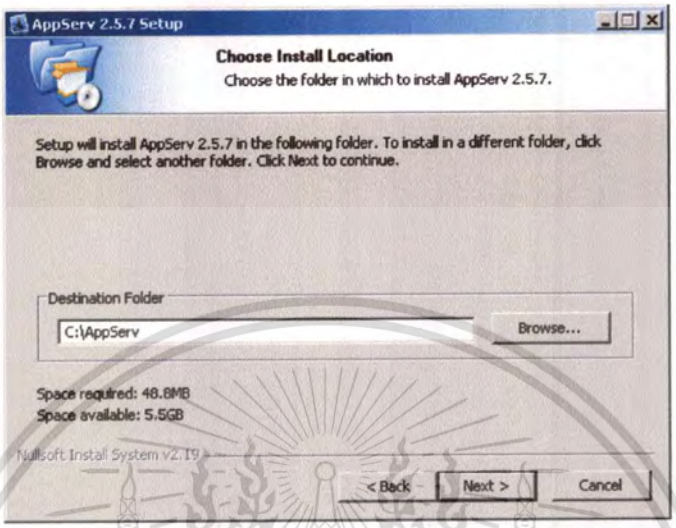


Figure A.3 Choosing installation's location

5. Select the package components that you want to install. Then click **Next** button (see Figure A.4).
   - Apache HTTP Server
   - MySQL Database
   - PHP Hypertext Preprocessor
   - phpMyAdmin

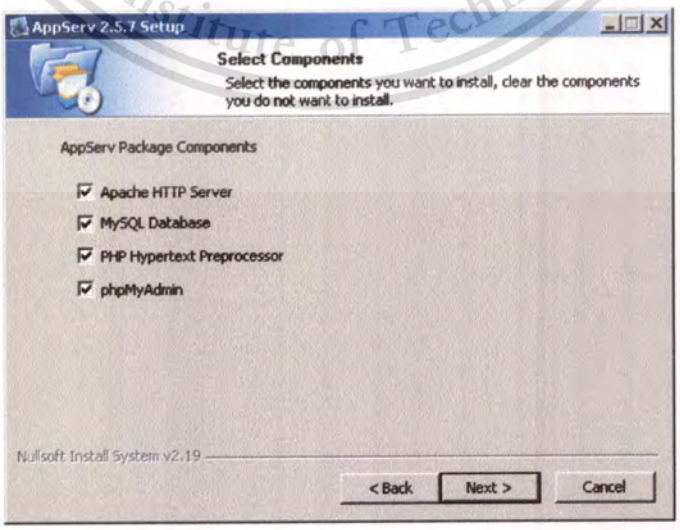


Figure A.4 Selecting package components

6. Configure the Apache Web Server and click **Next** button (see Figure A.5). Information required is listed below.
   - Server Name: Name of the webserver.
   - Admin's Email: Admin Email.
   - HTTP Port: Web Server's port number.

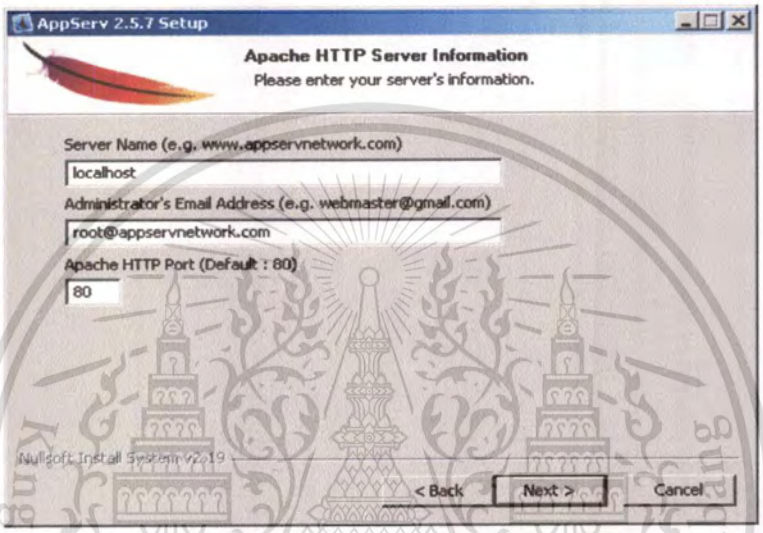


Figure A.5 Configuring the Apache Web Server

7. Configure the MySQL database as shown in Figure A.6. After configuring MySQL database, click **Install** button. Information required in this window are listed below.
   - Root Password: This is the password used as a root password to access the database.
   - Character Sets: It is used to define the encoding character system used in the database.

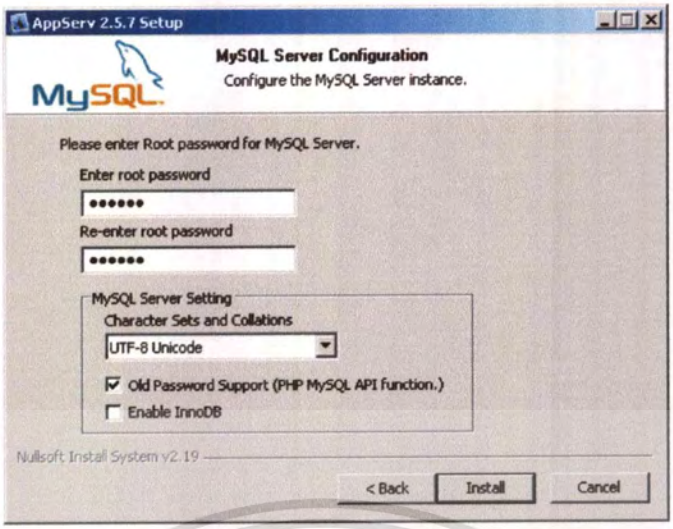


Figure A.6 Configuring the MySQL database

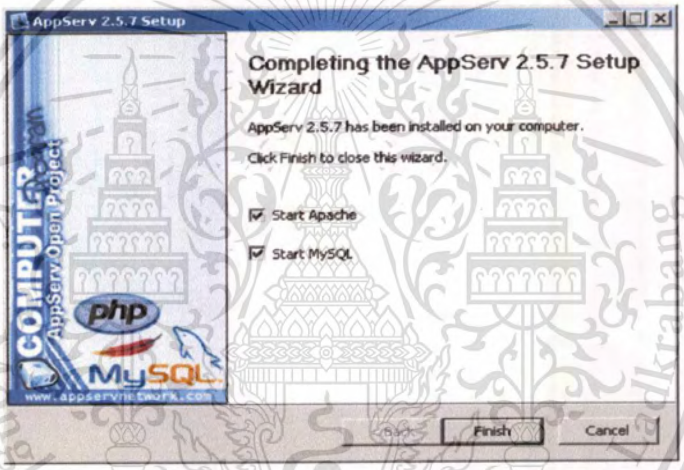


Figure A.7 Appserv installation completing page

8. Click **Finish** button to complete the installation (see Figure A.7).

Appendix B.

Eclipse and Android Plugins Installation

How to install Android SDK and "Eclipse"

1. Download Eclipse from https://www.eclipse.org/downloads/packages/eclipse-classic-362/heliossr2 (see Figure B.1)

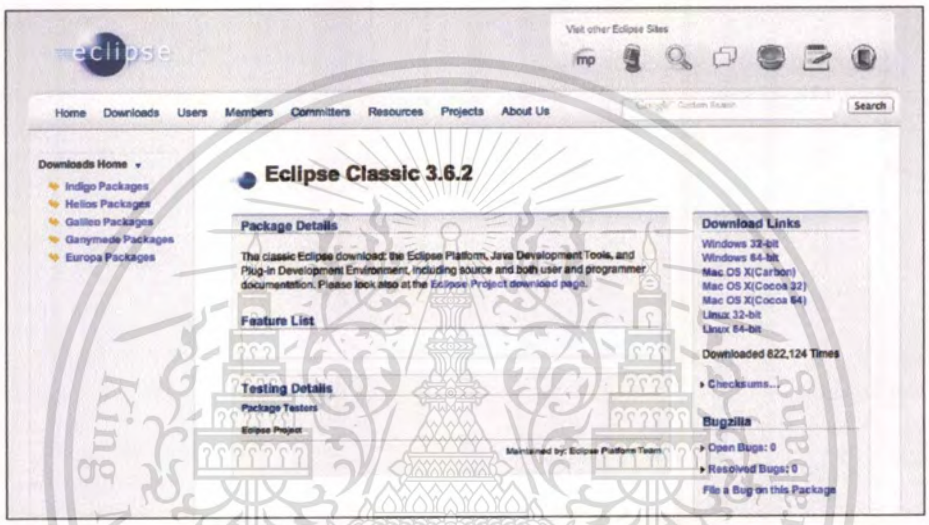


Figure B.1 Eclipse's downloading page

2. Download JDK and JRE from http://www.oracle.com/technetwork/java/javase/downloads/index.html (See Figure B.2)

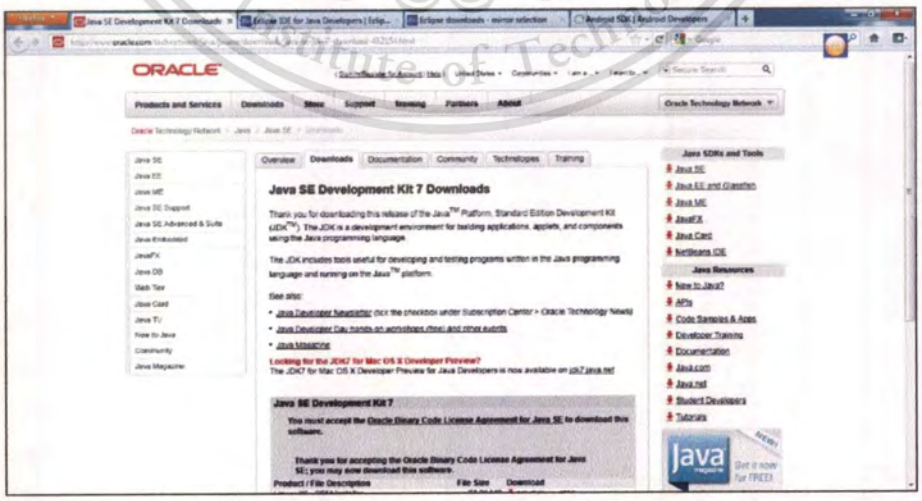


Figure B.2 JDK and JRE's downloading page

3. Download Android SDK from http://developer.android.com/sdk/index.html (See Figure B.3)

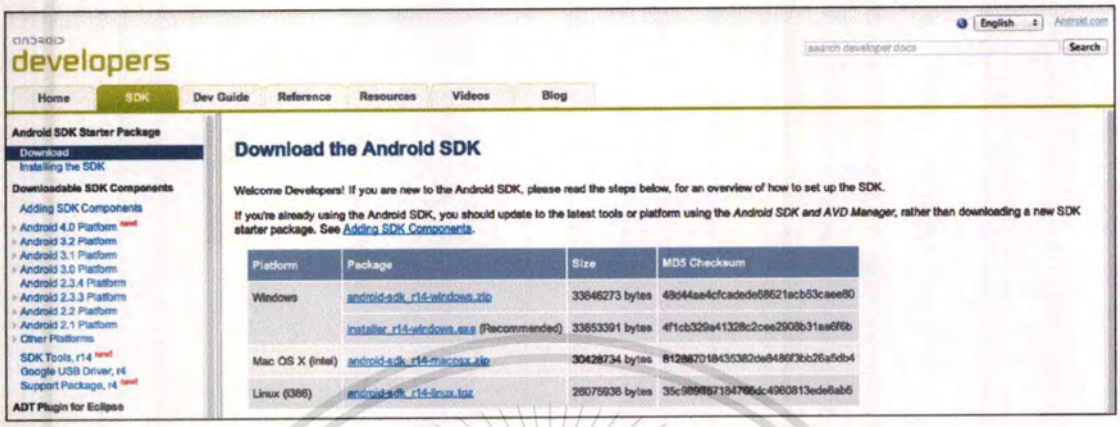


Figure B.3 Android SDK's downloading page

4. Install ADT plug in to Eclipse

   - Open program Eclipse then select **Help** from its menu bar (See Figure B.4).

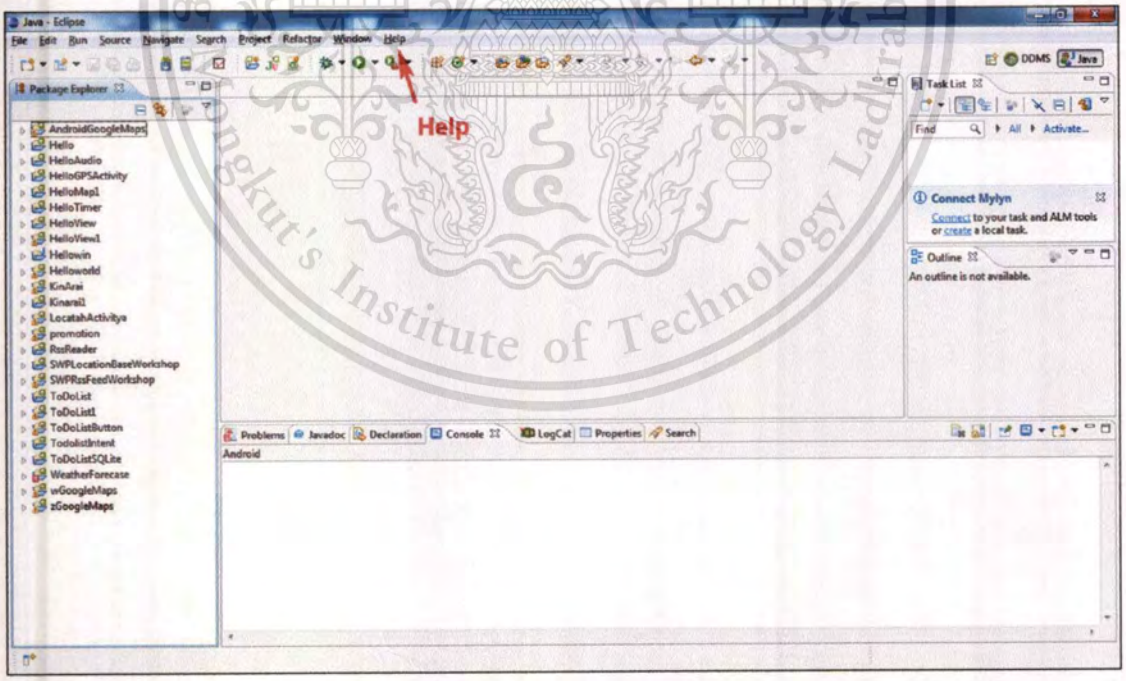


Figure B.4 Eclipse editor : Help button

- Select **"Install New Software"** (See Figure B.5).

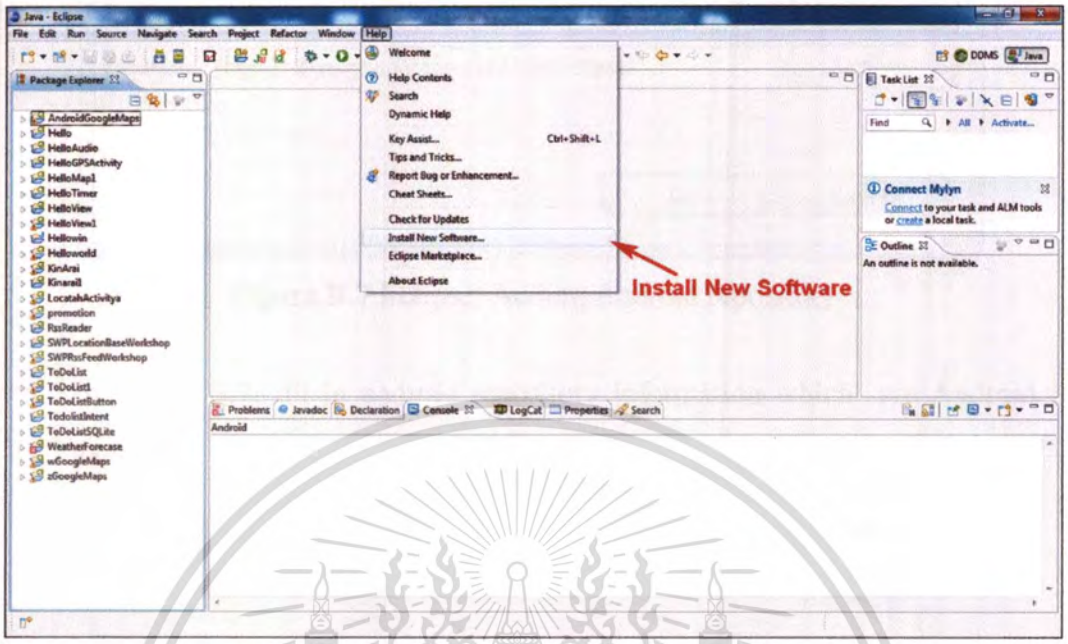


Figure B.5 Eclipse: Installation of new software

- In Figure B.6, click **Add** button so the next window as shown in Figure B.7 will appear.

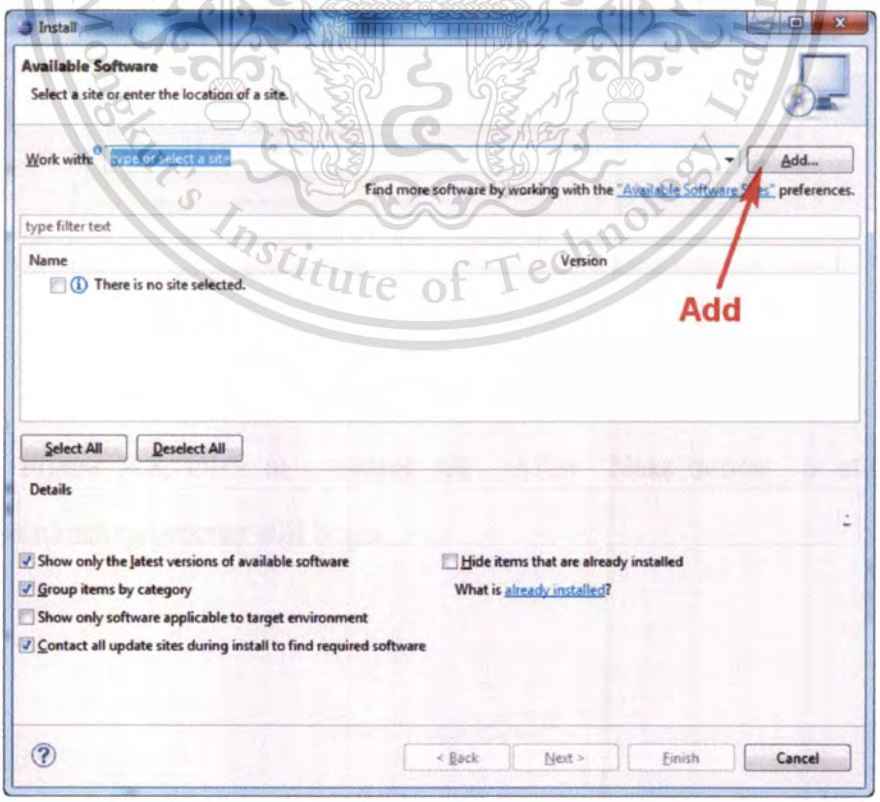


Figure B.6 Eclipse: Installation of Android SDK plugin window

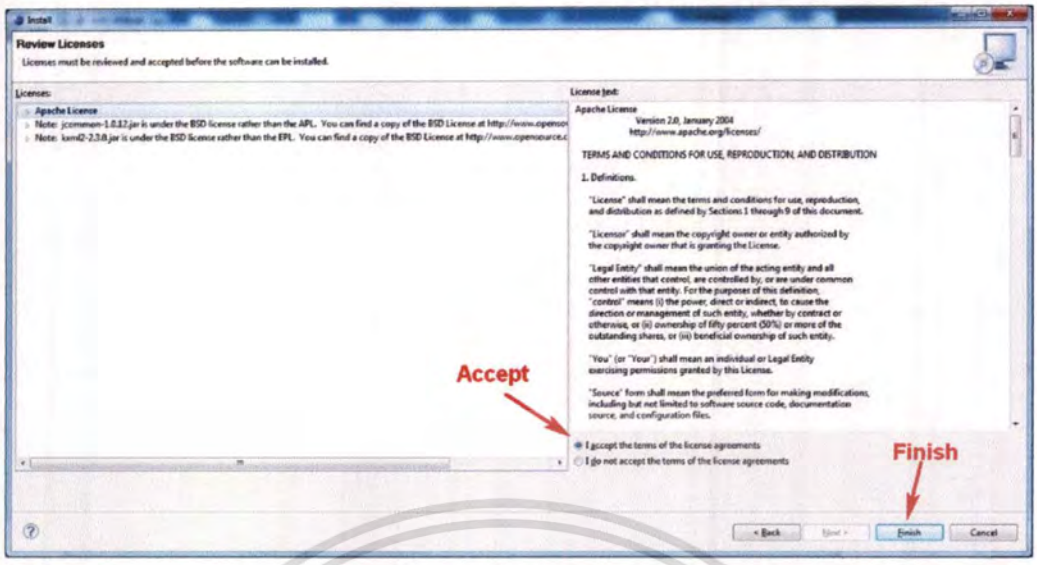


Figure B.9 Eclipse: SDK plugin's review license window

- In Figure B.9, when downloading process is finished, the system will ask for the acceptance of plugins' installation. Click at **I Accept**. Then, click at **Finish**.

Figure B.10 Eclipse: SDK plugin's review license window

- When the system finishes installing the plugins, press **Restart Now** as shown in Figure B.10. The system will restart the program.
- Open the Eclipse program and download the SDK version of android by clicking at **Android SDK Manager's icon** as shown in Figure B.11.

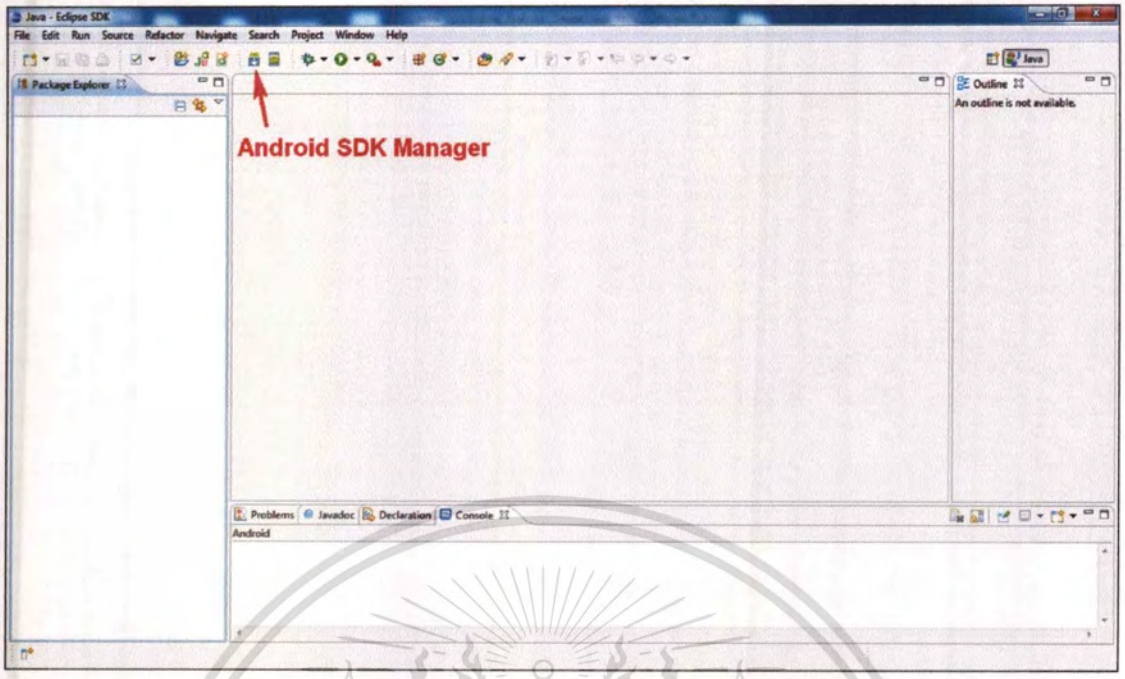


Figure B.11 Eclipse: Android SDK Manager's icon

- Android SDK manager allows its developers to choose the SDK version that they want to install, which more than one version can be installed at the same time. The choice of installation version depends on the developers' applications and devices. In our program, Android SDK version 2.3.3 was chosen which there were totally 24 packages as shown in Figure B.12. Note that the developer can omit some packages by expanding at the folder icon in front of the version number and deselect those packages then, click at **Install 24 packages** button.

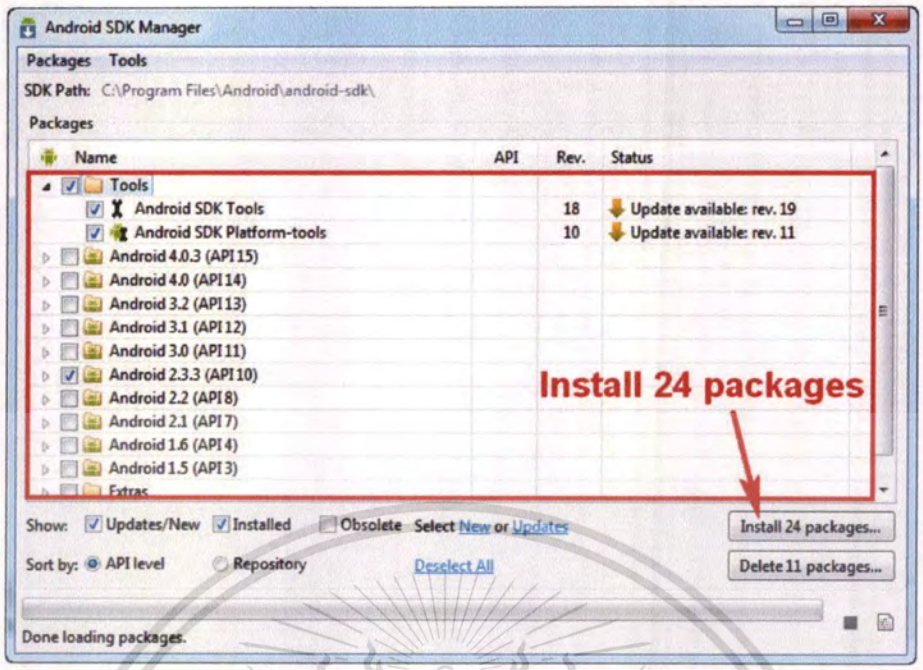


Figure B.12 Android SDK manager: version selection

- In Figure B.13, select **Accept All** and click at **Install** button.

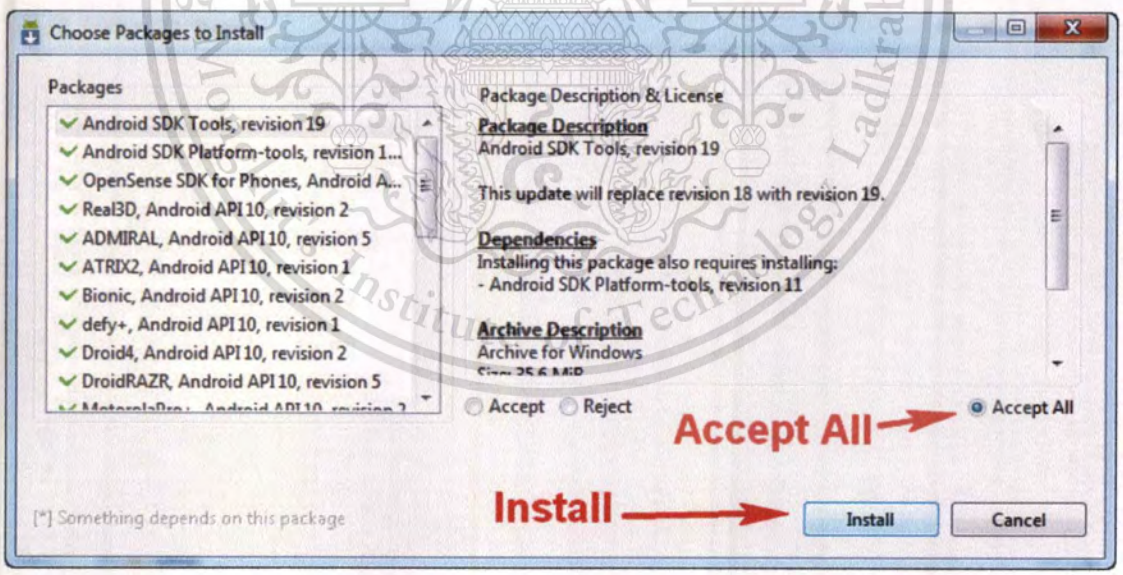


Figure B.13 Android SDK manager: Package description and license acceptance

- Figure B.14 shows the installation of android SDK which when the installation is finished, the program will ask to restart ADB (Android Debug Bridge). Click **Yes** to restart as shown in Figure B.15.

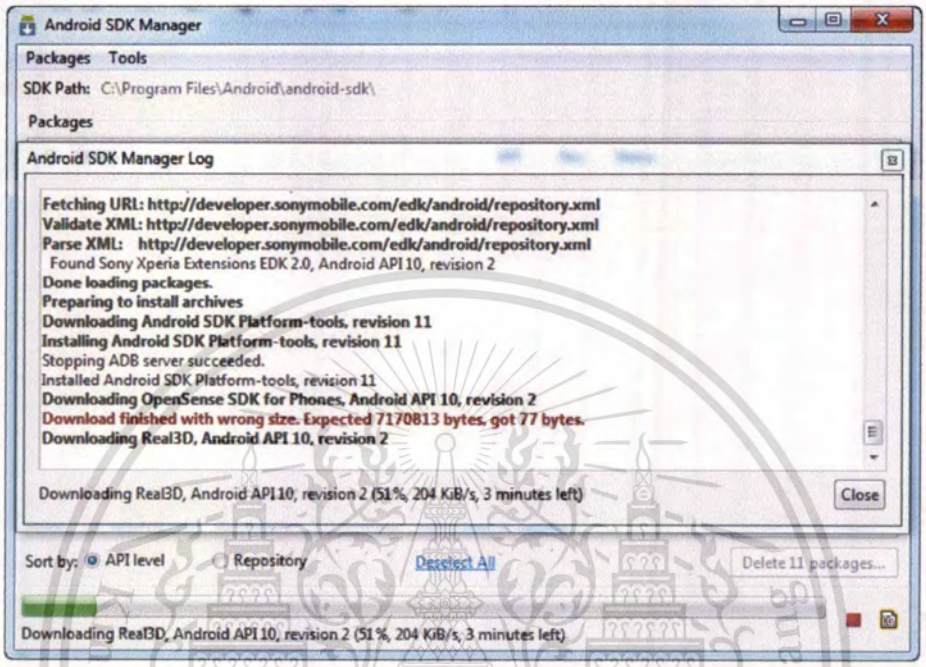


Figure B.14 Android SDK Manager: Installation of android SDK

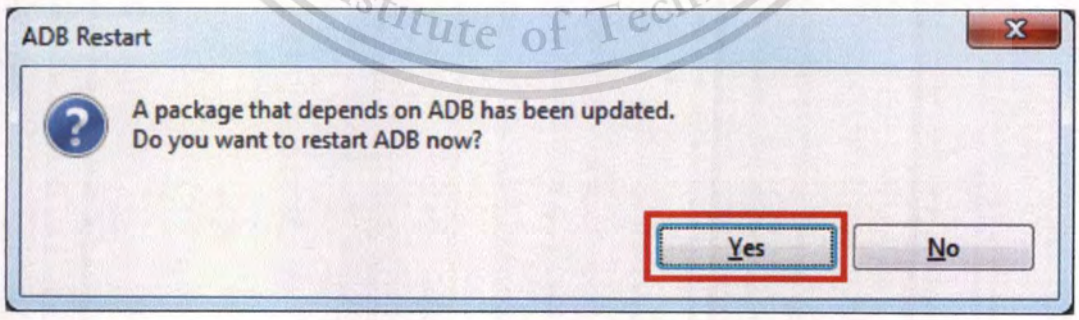


Figure B.15 Android SDK Manager: Restart ADB

- Figure B.16 shows the Android SDK manager log after the installation is successful. Click **at Close** button to prepare for next step.

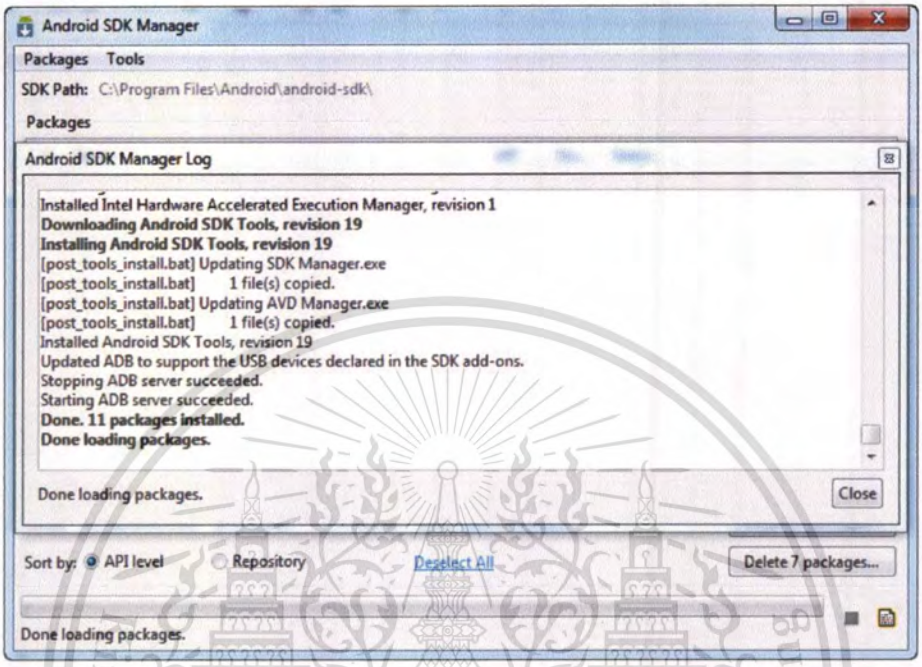


Figure B.16 Android SDK Manager: Android SDK Manager Log

5. Start the android emulator by pressing **File** → **New** → **Other** as shown in Figure B.17.

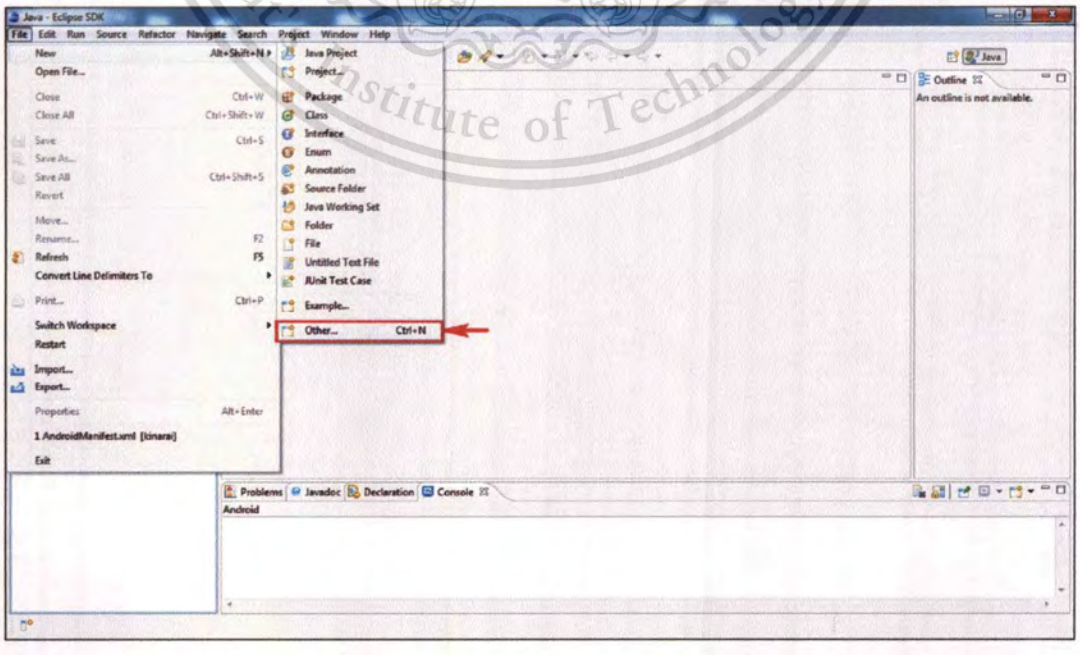


Figure B.17 Eclipse: Starting the android emulator

- To create an android project, click at **Android** → **Android Project** → **Next button** as shown in Figure B.18.

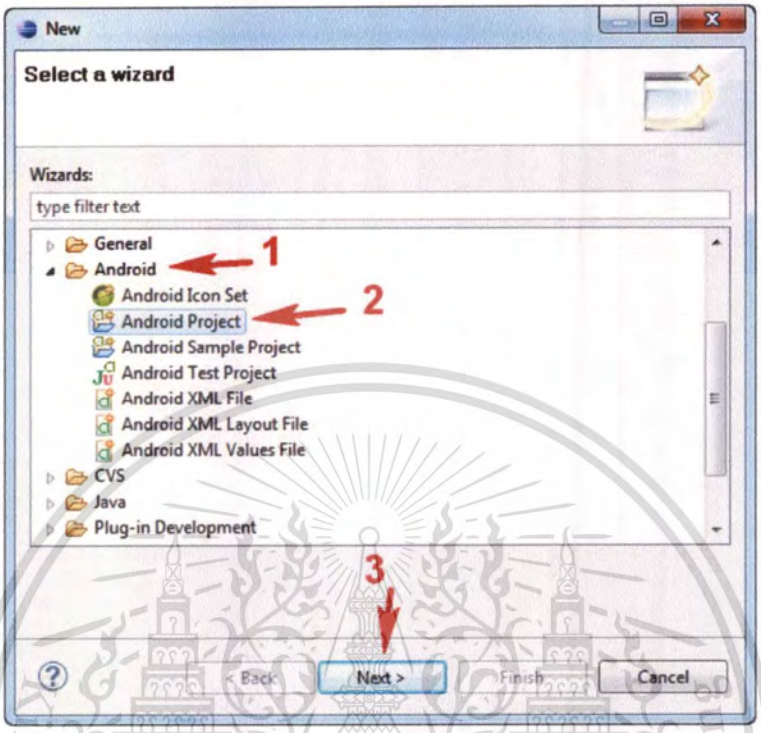


Figure B.18 Eclipse: Creating an android project

- As seen in Figure B.19, type your project name and click at **Next** button.

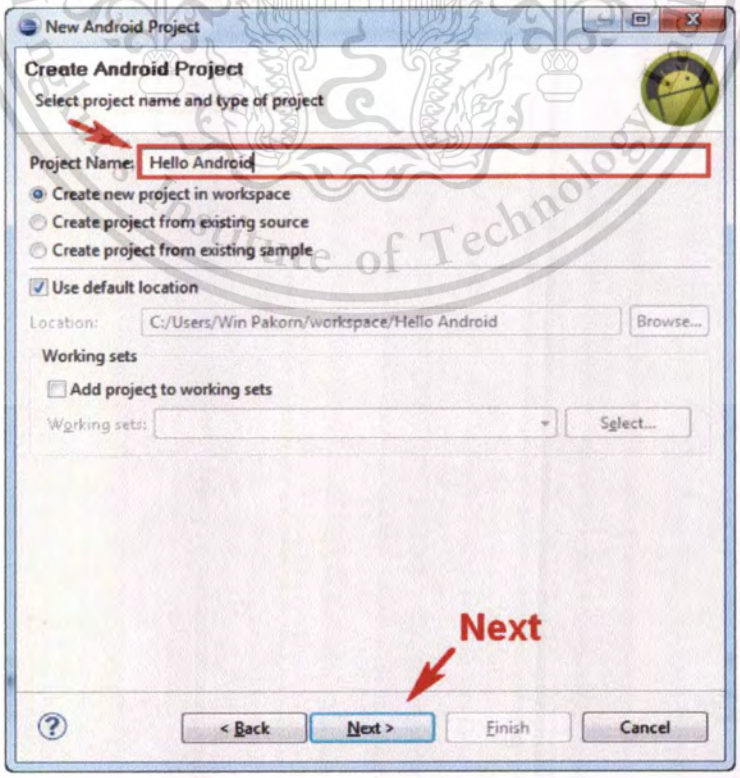


Figure B.19 Eclipse: New Android Project

- Then, checking at your target version and click **Next** button as shown in Figure B.20.

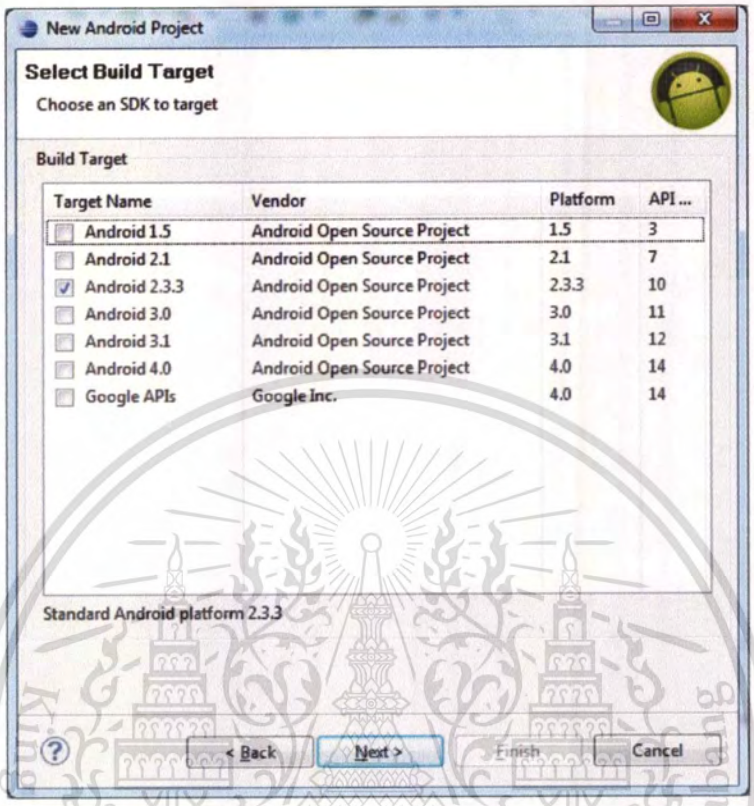


Figure B.20 Selection of android version

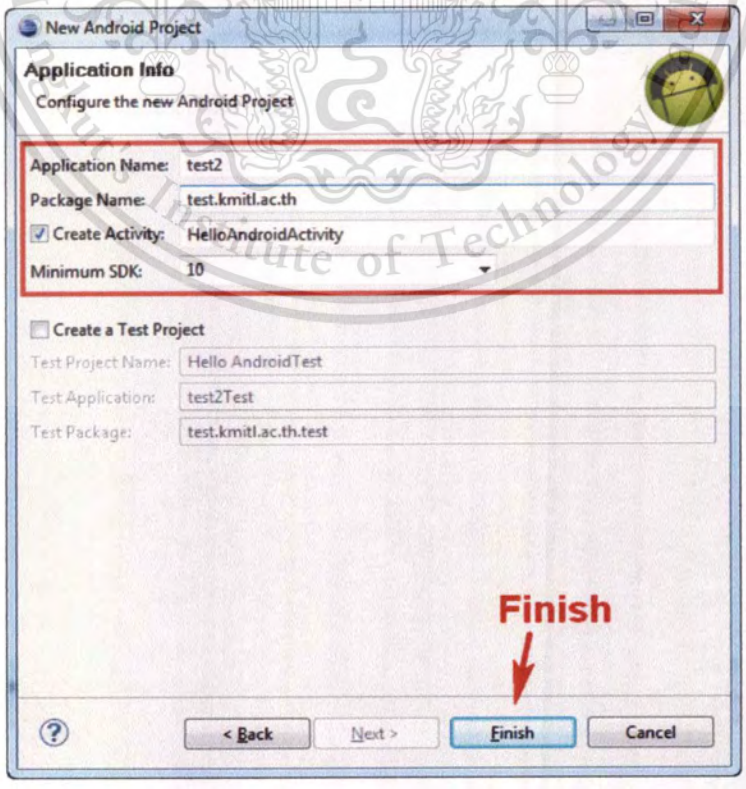


Figure B.21 Android project application information setting

- An application information is configured in Figure B.21. The developer is required to fill the **application name** *(test2)* and the **package name** *(test.kmitl.ac.th)*. **Create Activity** box is checked and the activity name is derived from the project name (*HelloAndroidActivity*), which is the project name followed by the word "Activity". A value next to **Minimum SDK** indicates the lowest Android SDK version that the project can run. When the configuration is done, click **Finish** button and the window as shown in Figure B.22 will appear. Now the program test2 can be compiled and seen in the emulator.

Steps to Compile and Run Android Project

Steps to compile and run the android projects are listed as follows:

1. **Right click** at the android project **Hello Android** as shown in Figure B.22.

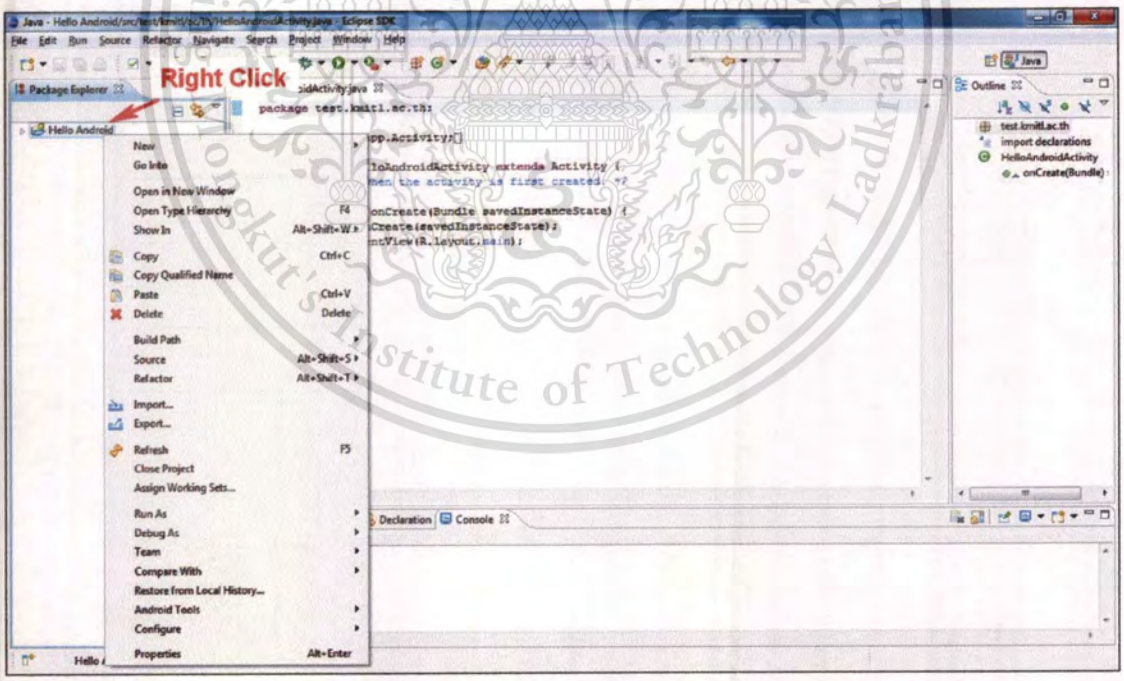


Figure B.22 Running a program on the emulator: Step 1

2. Select **Run As** → **Run Configurations** as shown in Figure B.23.

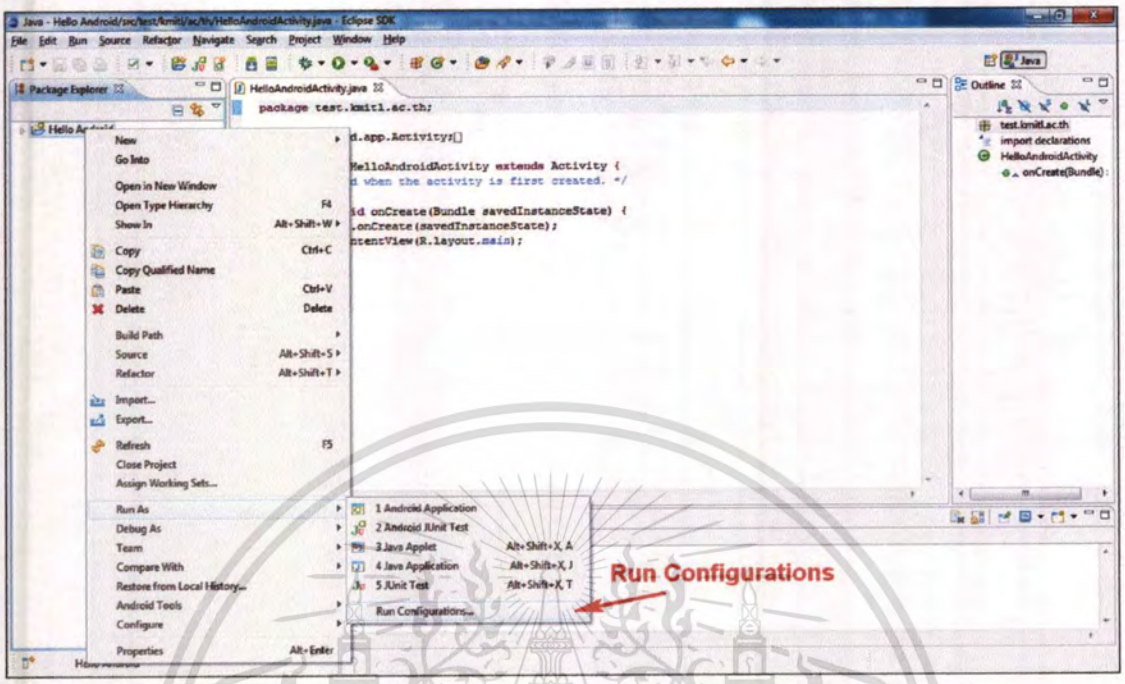


Figure B.23 Running a program on the emulator: Step 2

3. In Figure B.24, **double click** at **Android Application.** The **New Configuration**, and the Android tab will appear as shown in Figure B.25.

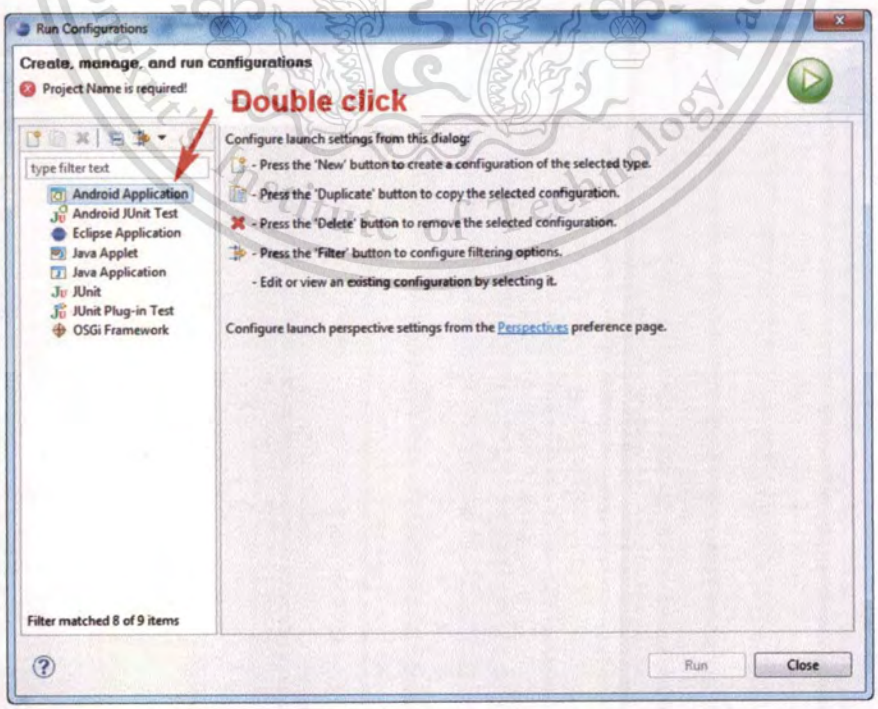


Figure B.24 Running a program on the emulator: Step 3

4. Next, enter the name of the project in the box next to **Name:** *(Test)* . Then, click at **Browse** button to select your project *(HelloAndroid)*.

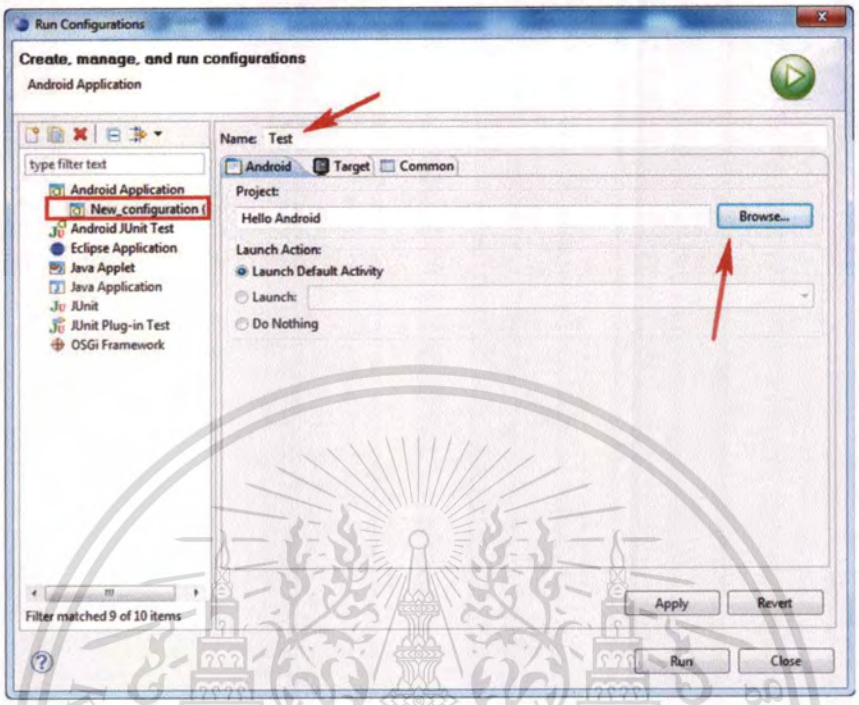


Figure B.25 Running a program on the emulator: Step 4

5. Click at **Target tab** as shown in Figure B.26. Then, tick box in front of your chosen Android Virtual Device under the **AVD Name.** Next, click at **Apply** button and **Run** button.

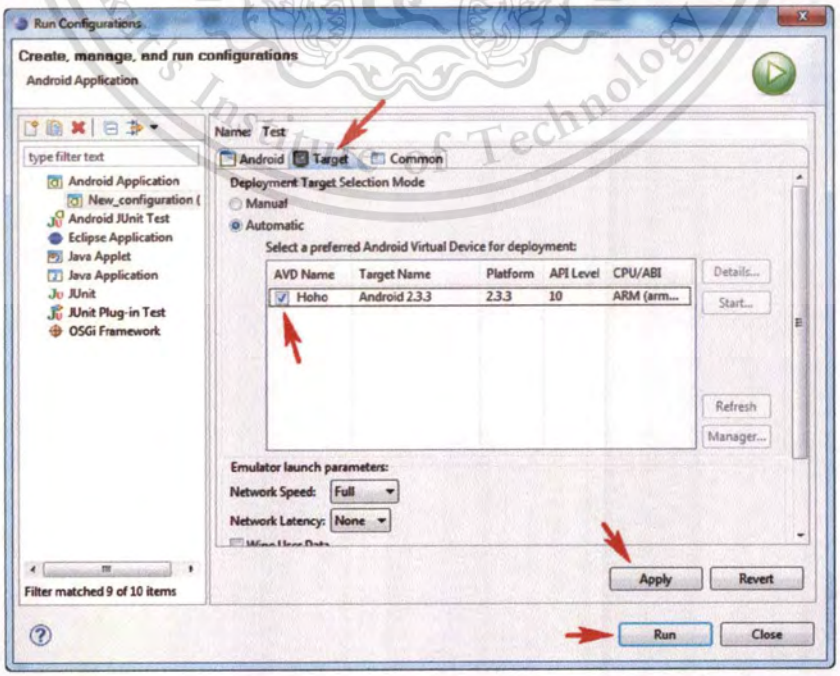


Figure B.26 Running a program on the emulator: Step 5

6. After clicking **Run** button in Figure B.26, the emulator will show and load the project as shown in Figure B.27.

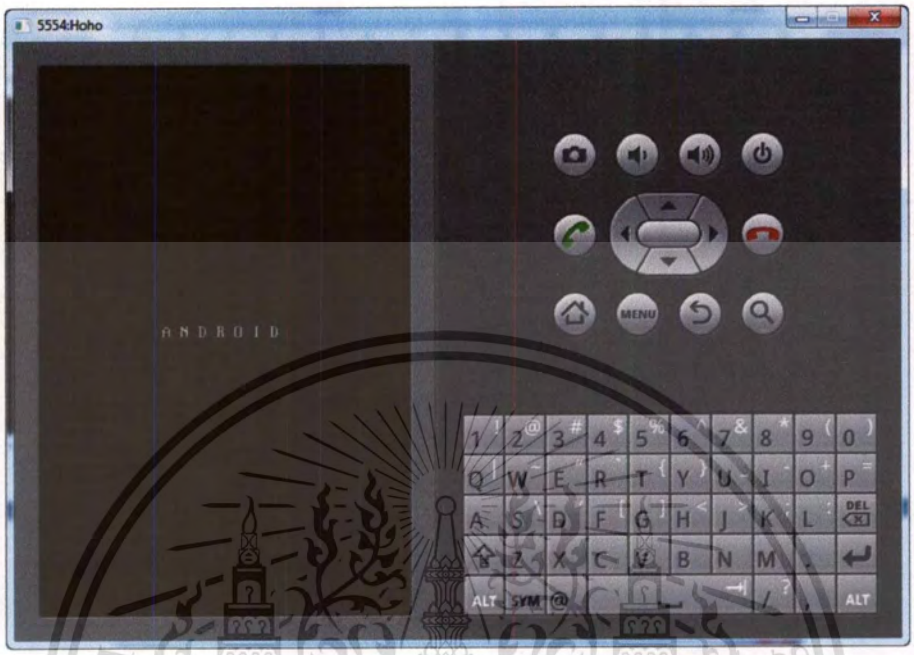


Figure B.27 Running a program on the emulator: Step 6

7. Figure B.28 shows the result of the project that are built.

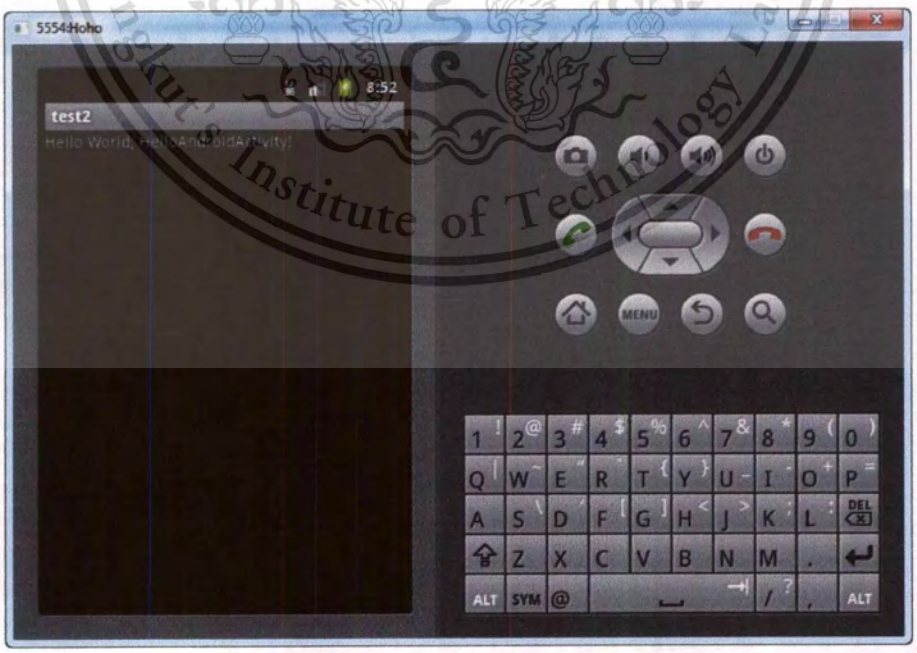


Figure B.28 Running a program on the emulator: Step 7

Appendix C.

How to export to the APK files

Export to the APK file.

When the users want to install our Kinarai application on their android mobiles, they need .apk file. The instructions to export the application from Eclipse SDK to the APK File are listed below.

1. Right click at the android project that you want to export under the tap **Project Explorer**.

2. Go to **Android Tools** and click at **Export Signed Application Package** (as shown in Figure C.1). The screen as in Figure C.2 will appear.

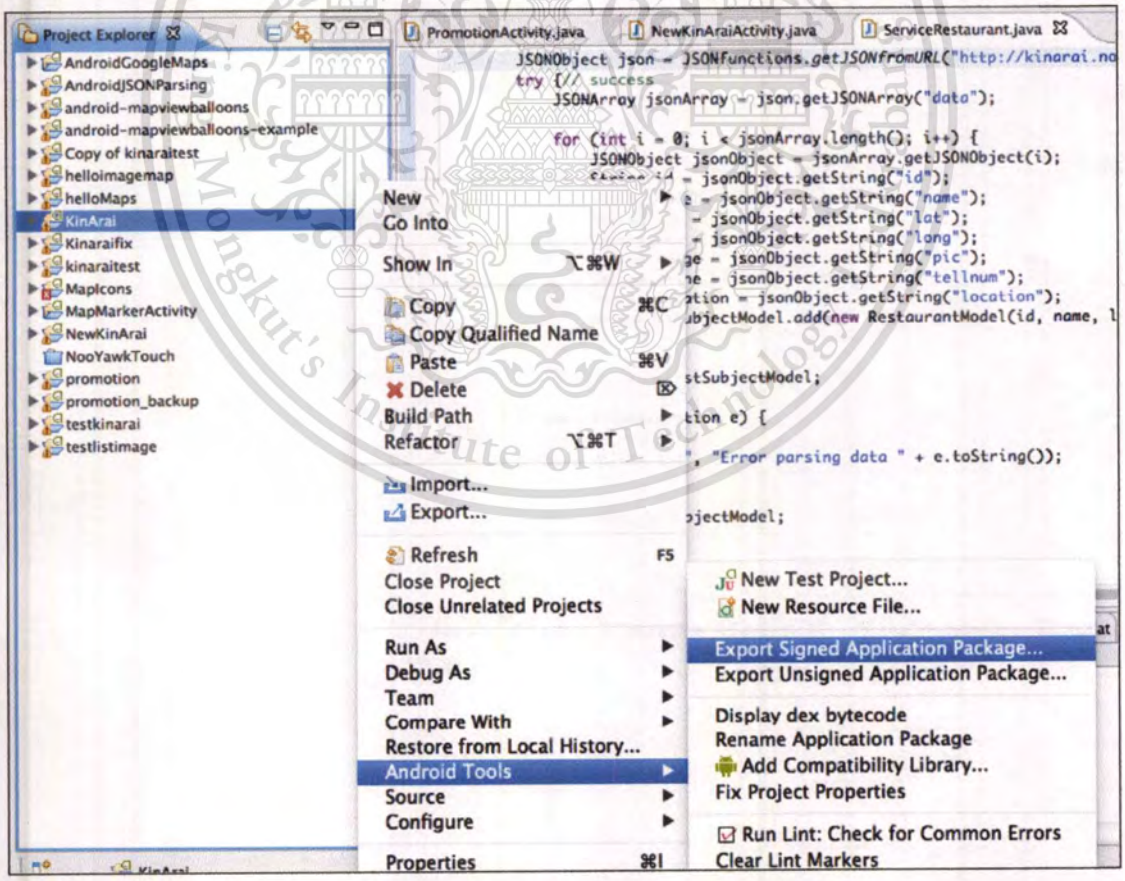


Figure C.1 Exporting android project to an APK file

3. **Browse** to the project name that you want to export and click. Then, click **Next.**

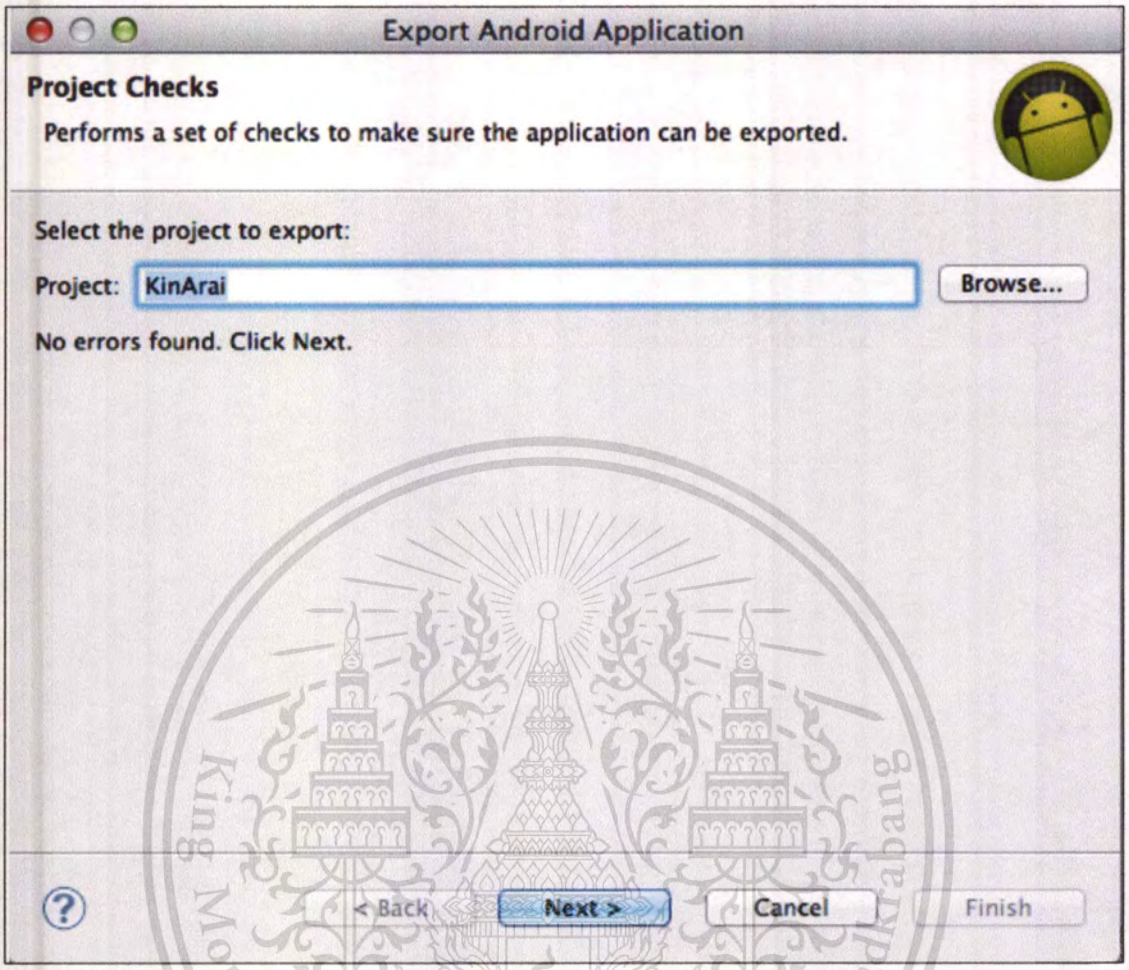


Figure C.2 Selecting the project

4. It will ask for the keystore. If the user already has set up the keystore, select **Use existing keystore** otherwise select **Create new keystore.** Then, type the password in both password and confirm textboxes. For **Use existing keystore**, enter the keystore file name or click **Browse** to browse for the existing keystore file as shown in Figure C.3. For **Create new keystore**,click Browse for the location that you want to save the keystore or otherwise use the default location which it is /Users/patipat/Chaiyapong as shown in Figure C.4. Then, click **Next.**

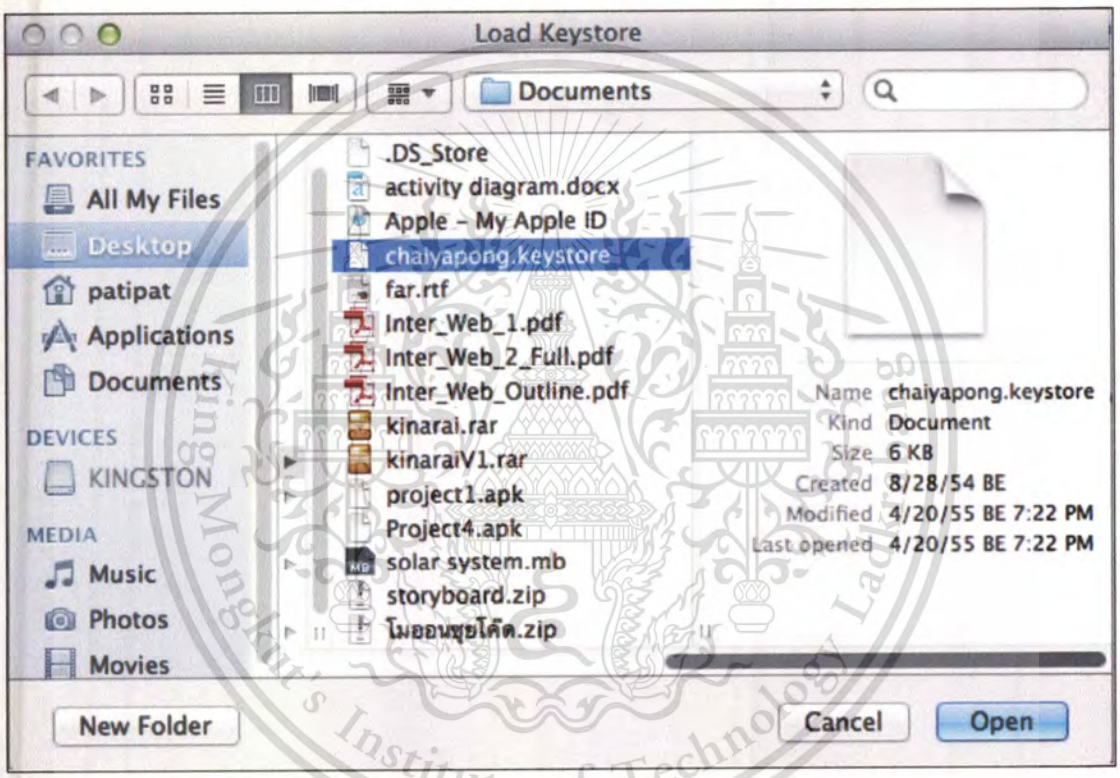


Figure C.3 Keystore selection (1)

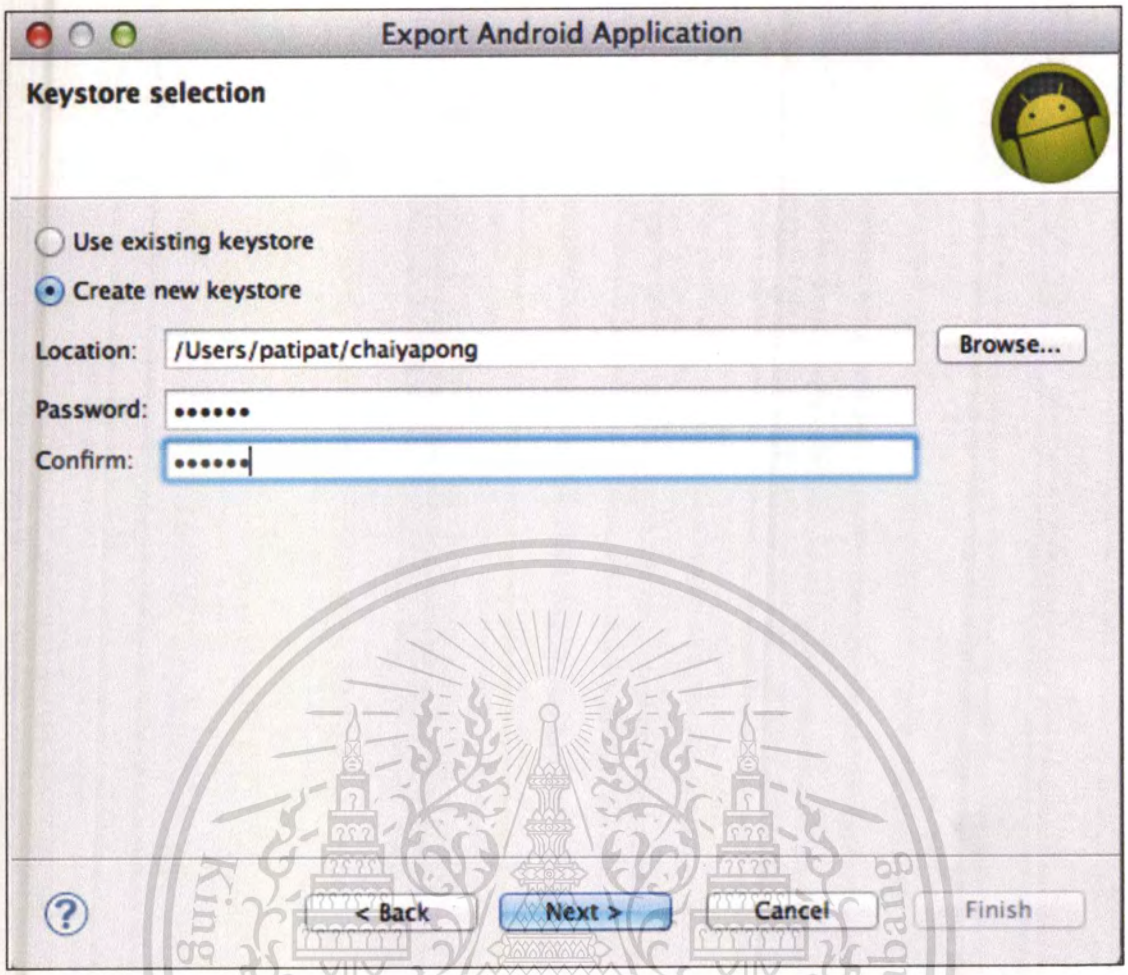


Figure C.4 Keystore selection (2)

5. It will ask for the key alias which is the alias of the developer who develops this application. Since we already have it so we create the new one as an example (see Figure C.5). Then, click **Next.**

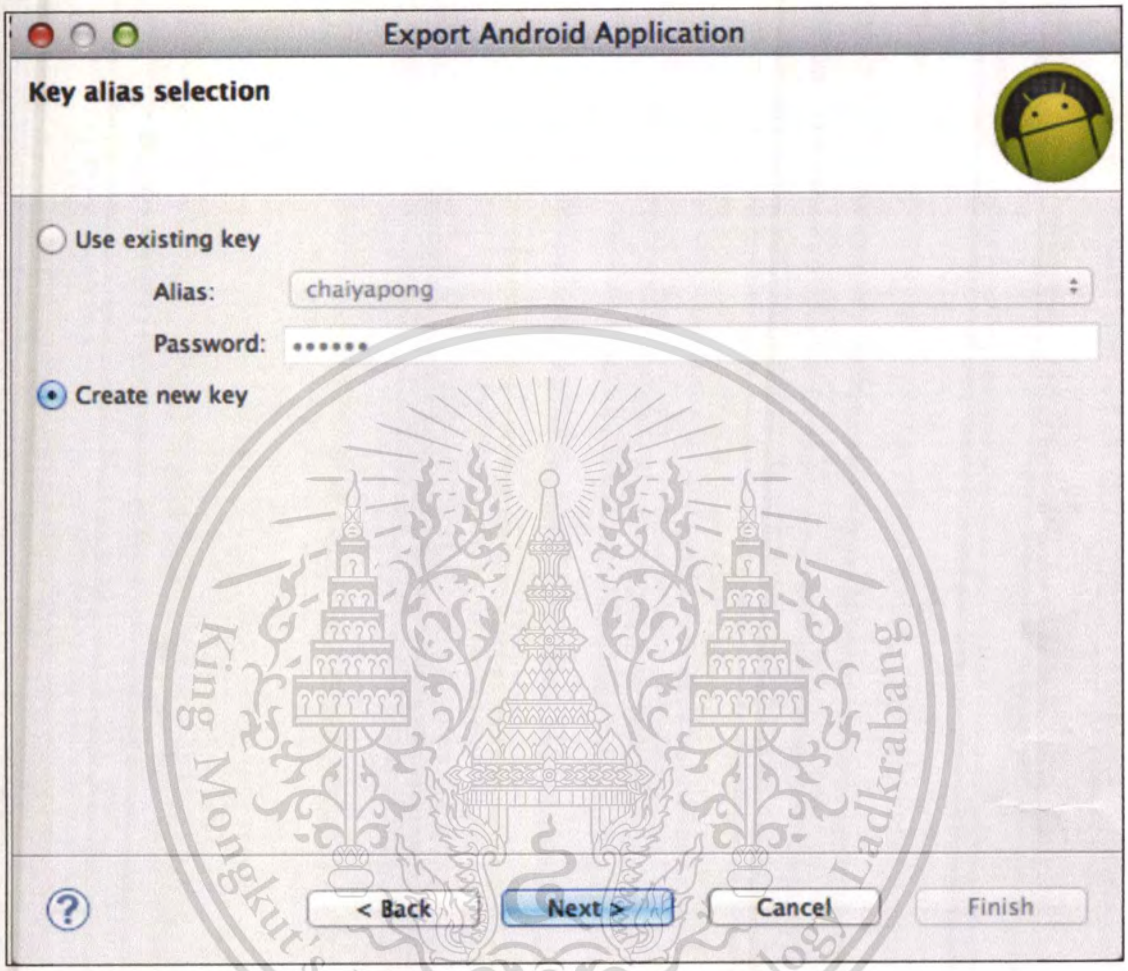


Figure C.5 Asking for key alias

6. The screen as in Figure C.6 which asks for the developer information will appear. Click **Next** after you fill all required information.

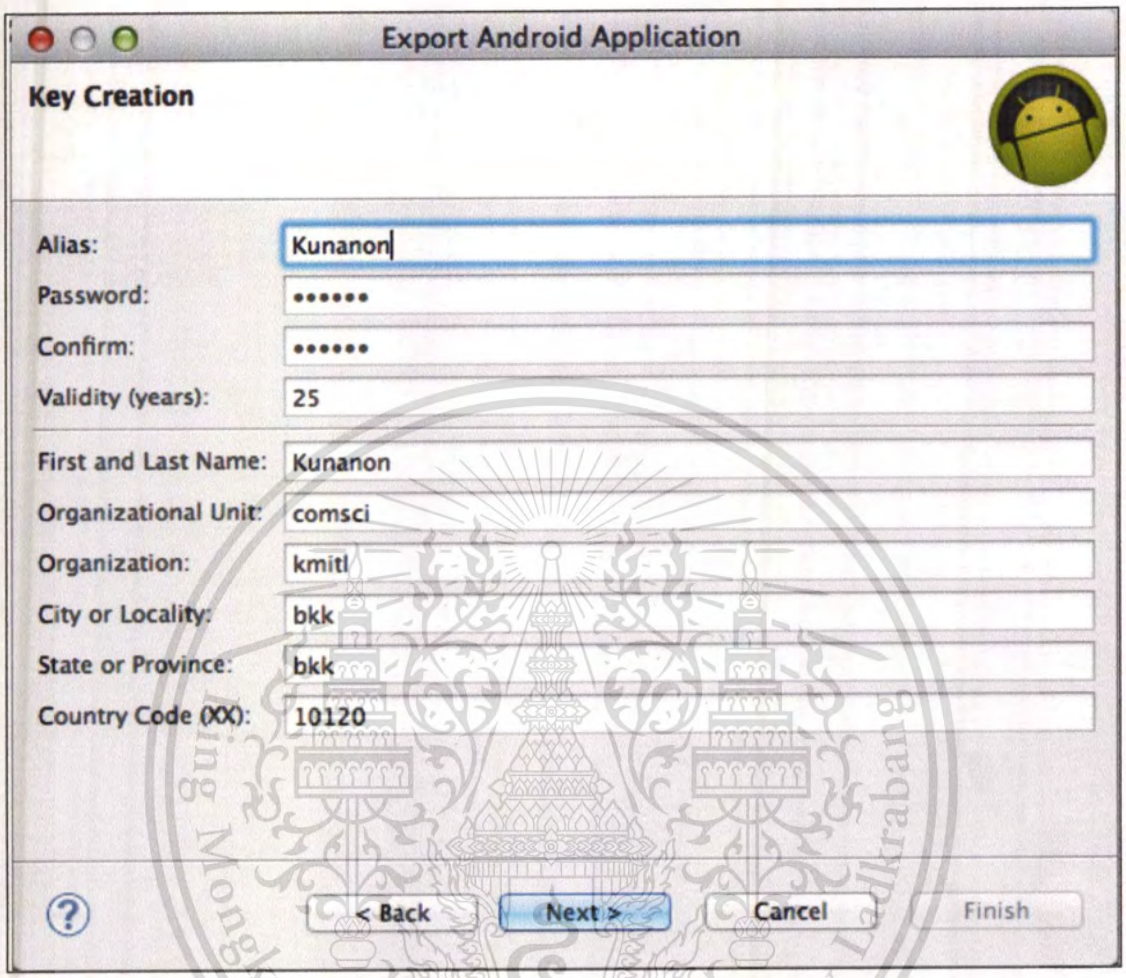


Figure C.6 Creating the key alias

7. Enter or browse for the destination path to export the file to. Then, click **Finish** (see Figure C.7).

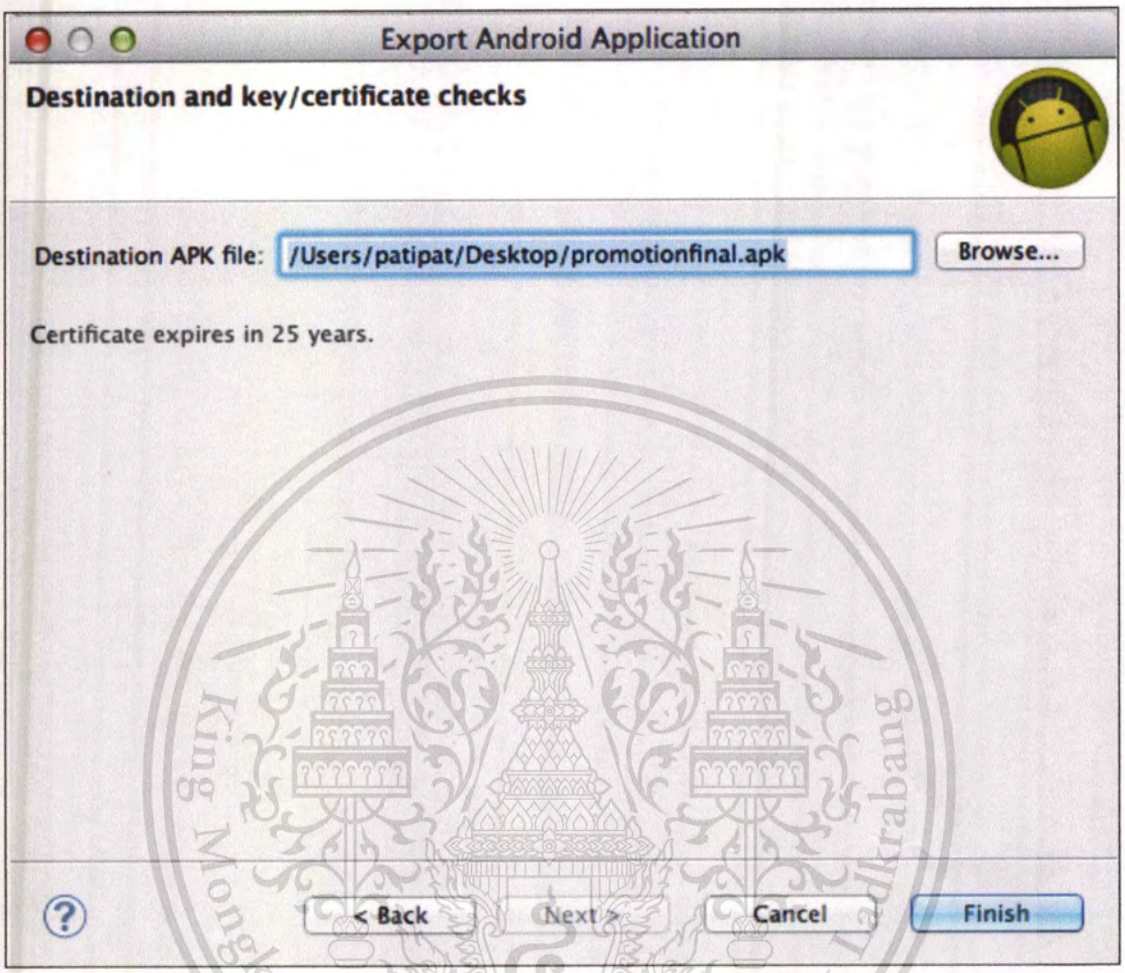


Figure C.7 Selecting the destination path

Appendix D.

User Manual

User Manual for KinArai Application

The **Kinarai** application provides 2 main functions. The first function is the promotion function which provides the list of latest restaurants' promotions. The second function is the nearby function which uses GPS to obtain the user current location and provides the list of restaurants located near the user (in the list view and map view).

When the users open Kinarai application, the main menu screen as shown in Figure D.1 will appear. In the main menu, there are two icons: the icon on the left (with fork and spoon) for the promotion function and the icon on the right with the green radar for the nearby function.

Promotion Function

The aim of the promotion function is to show the promotions of the user's selected restaurant. The usage steps are as follows:

1. In the main menu screen of the **KinArai** Application, click icon on the left hand side.

Figure D.1 Main screen of KinArai Application

2. List of restaurants obtaining from the Kinarai web server will be displayed as shown in Figure D.2.

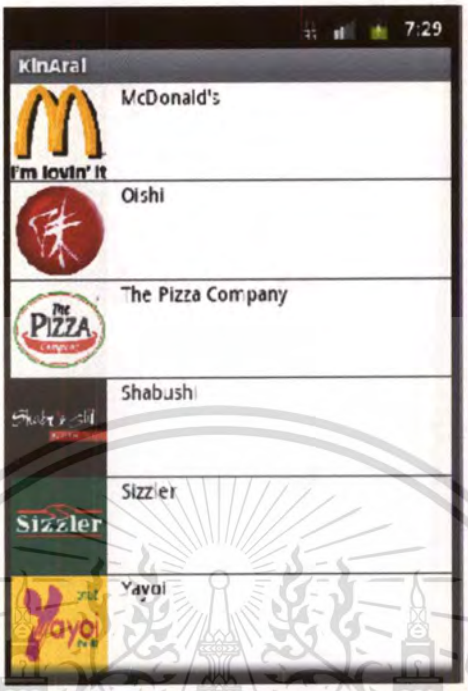


Figure D.2 KinArai Application: List of Restaurants

3. Touch the item of the restaurant that you want to know its promotion. The program then will obtain promotion information of the restaurant from the Kinarai server. In the example in Figure D.3, "Shabushi" was chosen.

Figure D.3 KinArai Application: List of the selected restaurant'spromotions

4. Touch at the promotion item that you are interested in for the promotion's description and picture in full page. The full page promotion will appear.

Figure D.4 KinArai Application: Full page of the selected promotion

5. If you want to see more detail for the selected promotion, touch at **More detail** button. The application will redirect to the original page of the promotion at the web promotion sites.

Figure D.5 KinArai Application: Display original website

Nearby Function

The aim of the nearby function is to provide the list of restaurants which are located near your current location. This function also allows you to choose your interesting district or landmark such as Latkrang and Centralworld to see the list of the restaurants around that place including their maps. Moreover, the function allows you to search for the restaurant by typing its name. Nearby function provides 3 options for finding the restaurants.

- Search by user specified distance. The first one allows you to type the distance number in kilometers.
- Search by using the distance drop-down list. The second one allows you to select number between 1 to 10 kilometers from the dropdown menu.
- Search by user specified name or the place. The third one allows you to enter the place or the restaurant name that you want to search for.

The usage steps are as follows:

From the main screen of KinArai Application as shown in Figure D.6, select the right icon.

Figure D.6 Main screen of KinArai Application

Search by user specified distance:

1. Enter the distance (radius) in the distance textbox (Figure D.7). Then, click **Find** button.

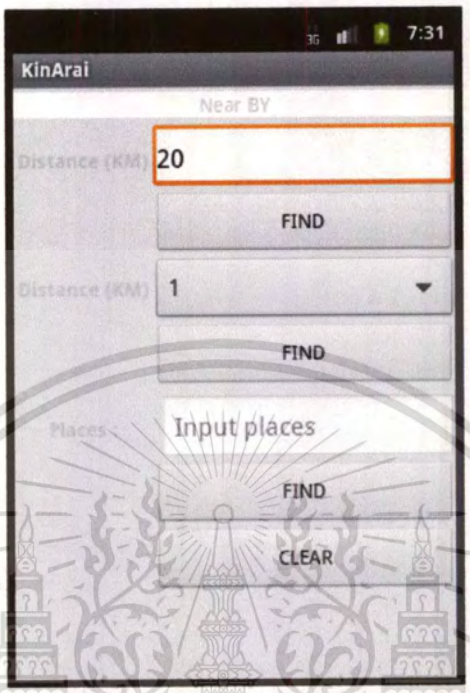


Figure D.7 KinArai Application: Search option pages

2. The application will display the result in the list of the restaurants with their logos, names and distances (Figure D.8).

Figure D.8 KinArai Application: List of result search location

3. Select the restaurant that you want to go for its map by simply touching at its item. The map with the red bubble marked on the restaurant location will appear.

Figure D.9 KinArai Application: Show the marker of selected restaurant

4. For the direction to the destination, click at the map again, the program will redirect to the **Maps application**. The **Maps application** will display the route from your current location to the selected restaurant as shown in Figure D.10.

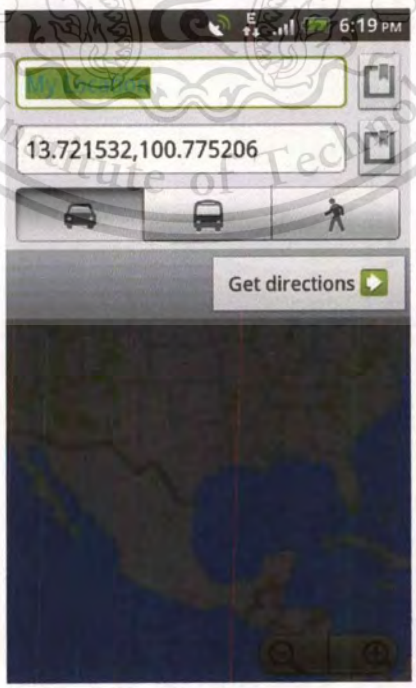


Figure D.10 KinArai Application: Maps application display

5. Click at **Get Direction** button to get the direction (Figure D.11). The result will appear in Figure D.12

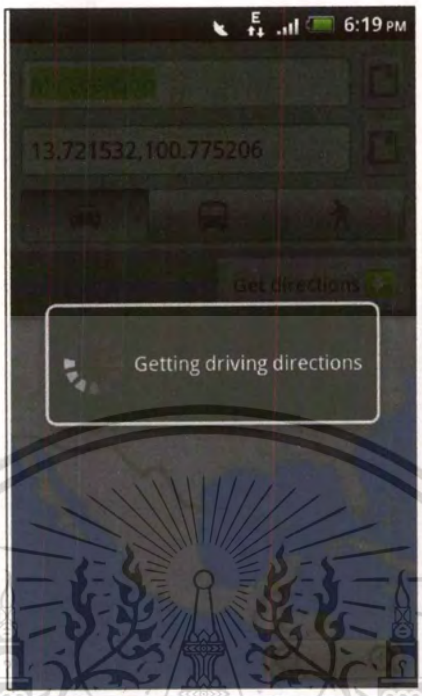


Figure D.11 Map application waiting to calculate the direction

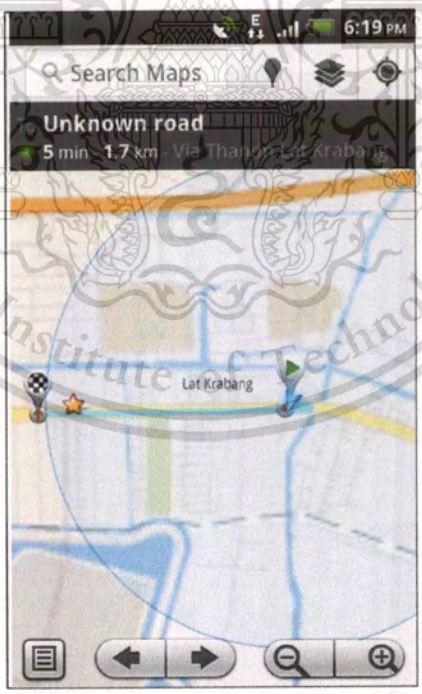


Figure D.12 Map application showing the direction

Search by using the distance drop-down list:

Go back to the nearby first screen (Figure D.7). Click at the distance dropdown list, list of the number between 1 to 10 will appear as shown in Figure D.13.

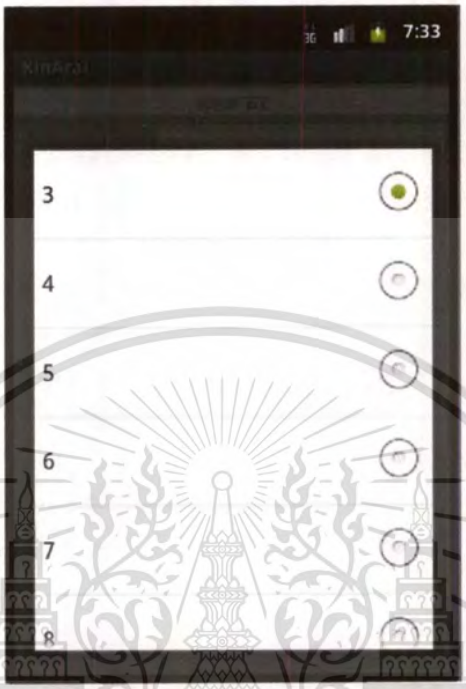


Figure D.13 KinArai Application: Show the numbers between 1 to 10

Select the number that you want, the application will process with the same step as search by user specified distance in Figure D.8 to Figure D.12.

Search by user specified name or the place:

Go back to the nearby first screen (Figure D.7). Type the place name or area that you want to search for the restaurant around such as **Centralworld, Latkrabang, Ratchathewi, Pathumwan** (see Figure D.14, Figure D.15). Then, click **Find** button.

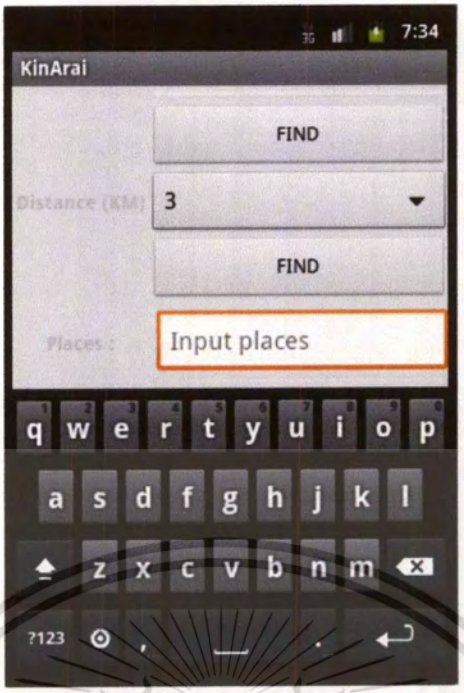


Figure D.14 KinArai Application: last type for searching

In this type of searching, the application provides a suggestion to places that are available on the system. So you can just simply type the initial letter/letters of the name, and the suggestion places will be shown (see Figure D.15).

Figure D.15 KinArai Application: Input place for searching

The the application will display a screen as in Figure D.16. In this screen, you will see 2 tab bar menus: the list view and the map view. The list view (which is the default setting) shows the result with the list of names and locations (default screen) (Figure D.16).

Figure D.16 KinArai Application: Display the result of searching in list view

To see the map, touch at **Map View** Tab. The map view's screen shows a map with the red bubbles spotted on the restaurants' locations (see Figure D.17).

Figure D.17 KinArai Application: Display the result of searching in map view

1. Press at the marker of the restaurant that you want to go. The screen as in Figure D.18 will appear.

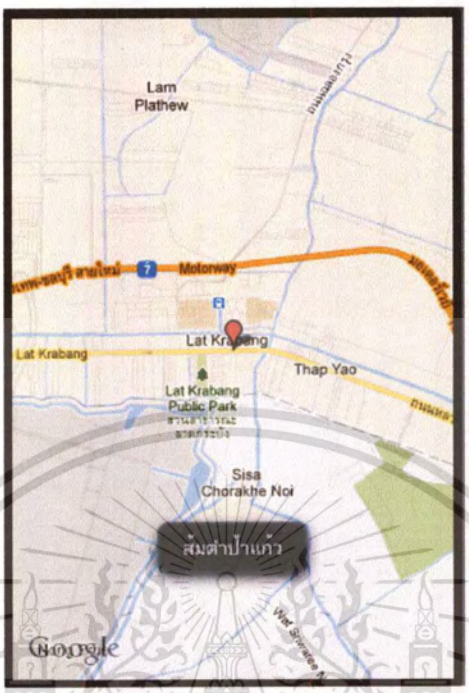


Figure D.18 KinArai Application: Display the marker restaurant's selected

2. Click at the map to redirect to the **Maps application** for the direction to the selected restaurant.