THRINITY COLLEGE DUBLIN

DUBLIN COMPASS MOBILE SYSTEM

Mobile-Based Travelling Information System

Jian Shen

A dissertation submitted to the University of Dublin, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

September 2010

Declaration

I declare that all the works described in this dissertation is entirely my own work, unless otherwise stated, and has been submitted as an exercise for a degree at this or any other University.

Signed: ______________________________

Jian Shen

13th September, 2010

Permission to lend and/or copy

Acknowledgements

First, I would like to thank my supervisor, Dr. René Meier, whose encouragement, supervision and support from the preliminary to the concluding level enabled me to develop an understanding of the subject.

Next, I am heartily thankful to Shane Brennan, a very obliging, amiable guy. Thanks for his helps during the development of the STIS services.

Of course, I also need to thanks for my parents who gave me the moral support I required.

Last, thank God!

Jian Shen

University of Dublin, Trinity College

13th September, 2010

Abstract

Ireland’s capital city, Dublin is at the center of the wonderful arc of Dublin Bay on the shores of the Irish Sea. Dublin is one of the top tourist destinations in Europe and each year welcomes many thousands of tourists to its streets. Currently, there are several tourist offices situated in Dublin that offer travelling information about tourist attractions and events, but it is far from meeting the needs of tourists. This dissertation focuses on ways to provide an open platform for mobile devices, it is named Dublin Compass Mobile System (DCMS), which allows tourists to query the information about the places of Interest (POI), create their own POIs, edit existing POIs, and plan a journey to somewhere. Moreover, under this platform, tourists also can share their travelling experiences with each other about a certain place by means of short reviews, rating, and photos.

For achieving above proposes, this platform is composed of three parts: POI Information System, STIS Web Service, and Prototype Mobile-Based Application running on the iOS (the Operating System of iPhone, iPod Touch, and iPad).

The POI Information System is proposed to provide some interfaces that can be used by client side applications to implement some particular functions concerning POIs, such as searching POIs by user-defined keywords or nearby POIs by a specific location (e.g. users’ current location), creating or modifying the information of POIs, or drafting reviews and rating for a certain place. The next part, STIS Web Services, is built on the Smart Travelling Information Server that is designed and developed by Shane Brennan. Under this service, client side applications can support planning multi-modal journeys using up to five different modes of transportation. The last part is an iPhone application that is used to demonstrate how a mobile-based application invokes above web services and interact with users.

Table of Contents

List of Figures

List of Tables

List of Code Tables

1. Introduction

“Innovation stems from Irritation.” – ***Tom Peters***

1.1 Motivation

For the traveller, in a imagining but strange city, a “guide” will be absolutely necessary for them. Generally speaking, the “guide” can be a book, a passer-by, and an assistant in a city tourist office, but sometimes they are not enough and convenient. For example, a guide book may provide some information concerning POIs for traveller, but it can’t direct them to get there or tell them where they are now; passers-by and assistants of tourist office may give travellers more, but they can’t always be at their side if they need a help. Therefore, who will be the “Right Man” if a traveller is lost in a city? The answer is the mobile device, especially the smartphone.

Modern technology moves fast and furious, especially when it comes to capabilities of the smartphone. At the essence smartphones are phones with open operating systems that allow free, third party development of software. Thus the smartphone differs from ordinary mobile phones in that they are capable of advanced functionality because they contain software applications that can be run directly from the phone itself. Today, the typical smartphone will accommodate the following capabilities:

- Datebook/Calendar;
- Advanced Internet connectivity;
- Photography;
- Music & video;
- Map & GPS locating;

Figure 1 - Smartphones

Based on these capabilities, mobile-based applications can easily implement several functionalities for travellers, such as invoking some online services to query POIs' information or locating themselves on a map.

It is easy to know that it is inefficient to store all of the travelling information, including POI and transportation data, into a mobile device. This is because some smartphones' memory is limited, and even if they can afford it, the stored data are maybe out-of-date. For example, the timetable data for some transportation are changed frequently, so if a mobile application only retrieves the local transportation data, the response for users is maybe incorrect. Therefore, considering this fact, some travelling online services should be involved in a mobile travelling system.

- *POI Information Service*. Usually, this type of Internet service provides some interfaces based on which client side applications are allowed to query and update the information about POI, such as searching POI information, manage POI information, drafting reviews for a certain place, etc. Moreover, some POI Information Systems also involve a user management service that allows users of the mobile device to register and manage their user accounts. Currently, some widely used POI information services are like *Yelp* (http://www.yelp.com) and *Lonely Planet* (http://www.lonelyplanet.com).

- *Route Planning Service.* The route service is also an indispensable service for a mobile travelling system. It can be invoked by clients to plan a route based on user-defined locations. A utilization of this type of service is to direct travellers from their current location to an interesting place. An example of the route planning service, currently, is the Smart Traveller Information Service [1], shortly named STIS, which is a system designed to provide users with a means of planning multi-modal routes, using the user preferences as the sole basis for creating route plans. This service coordinates between multiple transport networks to enable end-to-end planning using various modes of transport. Another example of the route planning service is the *Transport for London* that is a web service and provides the information on all forms of public transport in London, journey planner and online tickets sales. The *Transport for London* will be evaluated in detail in next chapter.

- *Translation Service*. As the name implied, this type of service is able to be used for translating information between different languages. One of sample utilizations of this service in a travelling system is to translate users' reviews into user-defined language (e.g. users' mother language). In this area, the Google Translation is one of the most widely used online translation service that can provide some interfaces use Java Ajax technology, allowing clients to translate text between up to 51 languages.

- *Weather Service*. Based on selected city, this type of service is able to offer its corresponding weather conditions in recent days. This is also a very useful service for mobile travelling system. Some sample online weather services are like "Yahoo! Weather" or "National Weather Service".

The project is aiming to use and combine above types of online services to design and develop a mobile travelling system for Dublin, named Dublin Compass Mobile System (DCMS). And based on this system, mobile device users can easily use several travelling services, such as searching and managing POI information, sharing travelling experiences with each other by means of reviews, rating, or photos, and planning a route based on user-defined locations.

1.2 Research Questions

This project is proposed to focus on the following research questions:

1. What is the *purpose* of this project?
2. Currently, are there some *relevant systems* that are similar with the DCMS? And through comparing with them, what the *advantages of DCMS*?
3. How to *design and implement* the DCMS?
4. How to *evaluate the DCMS* and what the *outcomes*?

1.3 Objectives

As discussion above, the purpose of this project is to design and implement a mobile travelling information system, the Dublin Compass Mobile System (DCMS), in order to provide some services that can be used for travelling purpose. Currently, there are some similar systems are used by travellers, such as *Transport for London* or *Yelp*. Therefore, one of the goals of this project is to present an overview of the current state of the art in mobile travelling information system, and based on the comparison with them to find what the advantages of the DCMS.

According to the purpose this this project, and based on the functionality, the DCMS is proposed to be divided into three components: POI Information System, STIS Web

Service, and Prototype Mobile-based Application.

- POI Information System. This system is proposed to offer some interfaces that allow client side applications to query and manage the information about POI.
- STIS Web Service. This service is built on the STIS, and its responsibility is to provide some web services that allow client side applications to query a route based on user-defined locations.
- Prototype Mobile-based Application. This application is just a prototype and proposed to run on the iOS platform. It is designed to use and combine the web services provide by above two systems to implement several functions for travelling purpose, such as POI information retrieval or route planning.

Therefore, another goal of this project is to design and implement these components.

The last goal in this project is to evaluate the DCMS. According to the common concerns of travellers, this project is proposed to evaluate the system from two aspects: system evaluation and functionality evaluation. The purpose of the system evaluation is to measure the performance of the POI Information System and STIS Web Service, and the functionality evaluation is aiming to evaluate whether the Prototype Mobile-based Application can work properly.

1.4 Contributions

According to the state-of-the-art study, one of the contributions of this project is to find the lacks of the current mobile travelling information system, and through the lacks evaluation to direct what improvements should be taken in travelling systems.

Another contribution of this project is the design and implementation of a mobile travelling information system. This system is composed of three components: POI

Information System, STIS Web Service, and Prototype Mobile-Based Application.

The *POI Information System* introduces the following features:

- The POI Information System provides a platform where tourists are allowed to share their travelling experiences.
- It provides an easier access to POI information. Therefore, based on this service, any applications, even if not mobile-based application, can easily implement some functions for retrieving POI information.
- Users are allowed to create and edit the POI information in this system. So after a period of years, it will create a Dublin-exclusive database that stores enormous quantities of POI information for Dublin.

The *STIS Web Service* introduces the following features:

- The STIS Web Service is just an upper layer of the STIS. The aim of this service is to interact with STIS and provide some web services to client side applications. Therefore, it can be regarded as an extension of STIS.

The *Prototype Mobile-Based Application* introduces the following features:

- The aim of this prototype application is to invoke above two services to implement some functionality for travelling purpose. The reason why it is called prototype application is because this application just provides a pattern to use and combine some utilizable online services to design a mobile travelling application. Therefore, although the prototype application in this project is proposed to be an iPhone application, based on this pattern, it will be easy to design and develop a mobile application running on the other mobile platform, such as Android.

The last contribution in this project is the outcomes of evaluating the DCMS. These outcomes can indicate where the DCMS should be improved in the future works.

1.5 Overview of Dissertation

The remainder of this thesis is structured as follows:

Chapter 2 introduces the concept of the Dublin Compass Mobile System, and presents an overview of the current state of the art in mobile travelling system. In this chapter there is also a comparison of each services and applications against the Dublin Compass Mobile System to be implemented for this project.

Chapter 3 is a description of the architecture design of the POI Information System, STIS Web Service, and Prototype Mobile-Based Application.

Chapter 4 illustrates the issues involved implementing the Dublin Compass Information System, the challenges, and how to be resolved.

Chapter 5 will evaluate the Dublin Compass Mobile System from two aspects: system evaluation and functionality evaluation.

Chapter 6 is a summary of the main conclusion of this project and potential future works.

Chapter 7 is a bibliography, listing all the resources, both printed and electronic, touched upon during the project.

2. State of the Art

*"A great part to the information I have was acquired by looking up something and finding something else on the way." – **Adams Franklin***

2.1 Introduction

The main concerns of travellers in a strange city generally are the POI information and the route information. Therefore, currently, there are a large number of, mostly mobile-based, travelling applications aiming to travellers, offering some services like route planner or POI information retrieval. Moreover, some advanced applications also allow users to create and edit POI information, and share travelling experiences with each other by means of reviews, rating, or photos.

Generally speaking, the main function that a routing service should provide is to plan a journey based on users' preference. Thus, one of the focuses of this chapter is to evaluate the routing service provided by current mobile travelling system. Also, this chapter will compare and contrast these systems with the Dublin Compass Mobile System in the scope of following issues:

- *Plan routes based on users' preference.* A routing service should allow users to plan a travelling journey based on their defined transport mode, start point, one or more waypoints, and destination.

- *Search parameter refinement.* This issue is used to evaluate whether a mobile travelling system can support the address refinement. The aim of the address refinement is to eliminate some input errors, such as spelling mistake, non-existent street names and/or missing address suffixes, and offer users a list of intended (corrected) locations.

- *Support multiple transport modes.* This issue is used to evaluate whether a mobile travelling system can allow users to plan a route across multiple transport modes.

- *The approaches of displaying route data.* The traditional approach to display a route to users is using textual instructions, but sometimes, for different users, it is ambiguity. Currently, most of mobile devices provide the map service, so based on this service, mobile travelling systems can also provide a more understandable way to display a route.

Next service a mobile travelling system should provide is POI information service. For this service, it should not only allow users to search POI information, but also manage POI information. Moreover, some advanced systems also need to provide a platform to users on which they can share their travelling experience by means of reviews, rating, and photos. And because the writers of reviews can be from different country using different language, a translation service will be a bonus point for the mobile travelling system to help travellers read all reviews. Thus, based on the services should be provided by the mobile travelling system, another focus of this chapter is to evaluate the POI information service, experiences sharing service, and translation service, and compare and contrast them with the Dublin Compass Mobile System in the scope of following issues:

- *POI information retrieval.* This issue is used to evaluate whether a mobile travelling system support POI information retrieval.

- *POI information management.* This issue is used to evaluate whether a mobile travelling system allows users to create and edit POI information.

- *Experiences sharing.* This issue can be used to evaluate a mobile travelling system whether allows users to draft a review and rate for a certain place.

- *Review translation service.* This issue is used to evaluate whether a mobile travelling system support review translation service.

2.2 Mobile Travelling Systems Researched

2.2.1 TomTom U.K. & Ireland

This is an iPhone application and produced by TomTom Company [2]. It is a traditional but easy-to-use and turn-by-turn GPS navigation application. Like most of GPS navigation apps, it can provide detailed and local maps, fast route calculation, POIs search, and so on.

Because the TomTom provides detailed local maps, when travelling aboard, it can still work without any Internet connections. In addition, because, on average, 15% of roads change every year [2], the TomTom allows users to upgrade the application's version to update local maps.

The TomTom application can not only provide a fast route calculation for users, but also help them calculate the accurate travelling duration based on its IQ routes technology. And after planning a route, it will display it on a map (see *Figure 2*) and associated with clear, spoken turn-by-turn instructions.

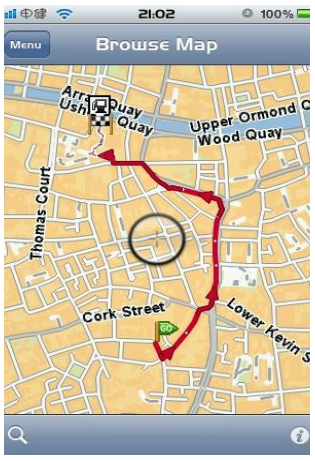


Figure 2 – Displays a route on the local map [TomTom]

The POI retrieval service provided by TomTom is based on the Google Place API. It allows users to find the countless shops and businesses wherever they are, and navigate directly there (see *Figure 3*).

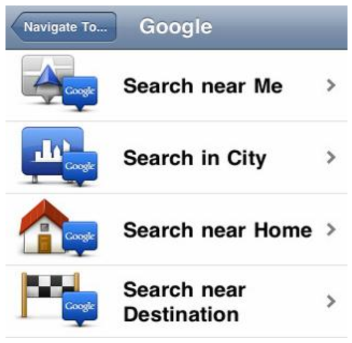


Figure 3 – POI information retrieval [TomTom]

Although the navigation functions offered by TomTom are much detailed and powerful, it is somewhat limited in that it only supports the single-modal transport for route planning. For example, the TomTom app only allows users to select a single transport mode, such as car or bicycle, before each route calculation. Moreover, the TomTome application supports POI information retrieval service based on the Google Place API, but it doesn't allow users to manage the POI information.

Note that the latest version of the TomTom app during this report is 1.4.1.

2.2.1.1 Comparison with DCMS

Actually, the TomTom and the DCMS are aiming to the different groups of people. The users of TomTom usually are drivers or cyclists, so their desired information will be like route data or navigating instruction. But for DCMS, its users are often travellers, so it should not only focus on planning a route, but also some services about POI information. Furthermore, the TomTom is an absolute "local" application, but the DCMS is based on the web services.

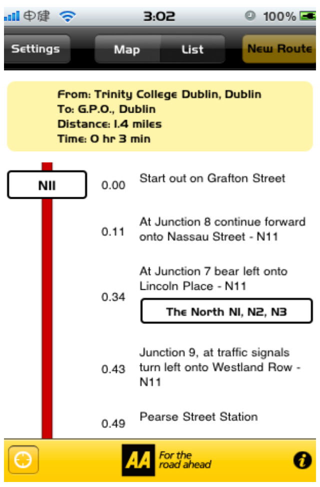
Contrast Issues	TomTom	DCMS
Plan routes based on users' preference	Yes	Yes
Search parameter refinement	Yes	Yes
Support multiple transport modes	No	Yes
The approaches of displaying route data	Map with the spoken instructions	Map & Text
POI information retrieval	Yes	Yes
POI information management	No	Yes
Experiences sharing	No	Yes
Review translation service	No	Yes

Table 1 – The results of comparison between TomTom and DCMS

2.2.2 *AA Route Planner*

The AA Route Planner is an iPhone application produced by Automobile Association Limited [3]. The major service this application provides is to allow users to select a start point and an end point to calculate a route. As a mobile travelling application, it has the following features.

All routes include directions, as users would see them on the road. For example, "at Stockwell Underground Station turn right (A123) signposted The City" (see *Figure 4*).



From: Trinity College Dublin, Dublin
To: G.P.O., Dublin
Distance: 1.4 miles
Time: 0 hr 3 min

0.00 Start out on Grafton Street

0.11 At Junction 8 continue forward onto Nassau Street - N11

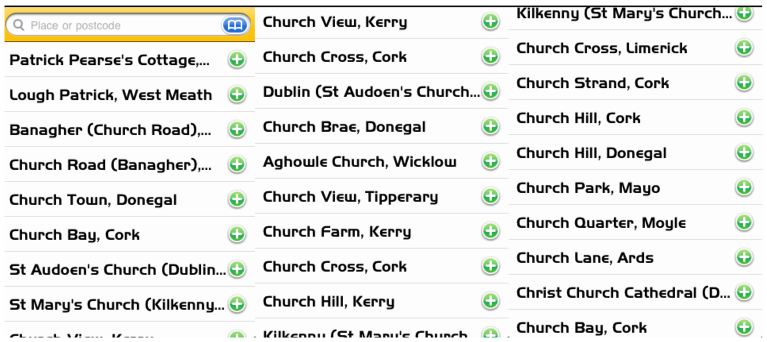
0.34 At Junction 7 bear left onto Lincoln Place - N11
The North N1, N2, N3

0.43 Junction 9, at traffic signals turn left onto Westland Row - N11

0.49 Pearse Street Station

Figure 4 – All routes include directions [AA Route Planner]

The AA Route Planner supports multiple types of parameters for searching a route, such as place name, address, postcode, and user's current location. And because of the inaccurate or imprecise travel information provided by users, it also support searching parameter refinement that will correct some input errors and offer a list of intended locations to resolve any ambiguity (see *Figure 5*). But because the AA routing system (a web service that is used by AA route planner) includes entire British and Irish road network, it can only provide limited refinement options.


Place or postcode

Patrick Pearse's Cottage,...
Lough Patrick, West Meath
Banagher (Church Road),...
Church Road (Banagher),...
Church Town, Donegal
Church Bay, Cork
St Audoen's Church (Dublin...
St Mary's Church (Kilkenny...

Church View, Kerry
Church Cross, Cork
Dublin (St Audoen's Church...
Church Brae, Donegal
Aghowle Church, Wicklow
Church View, Tipperary
Church Farm, Kerry
Church Cross, Cork
Church Hill, Kerry
Kilkenny (St Mary's Church

Kilkenny (St Mary's Church...
Church Cross, Limerick
Church Strand, Cork
Church Hill, Cork
Church Hill, Donegal
Church Park, Mayo
Church Quarter, Moyle
Church Lane, Ards
Christ Church Cathedral (D...
Church Bay, Cork

Figure 5 – Address options obtained by refining the keyword "St Patrick Church" [AA Route Planner]

The AA Route Planner also provides some options for users to optimize its routing service. For example, the options for avoiding motorways, toll roads and congestion charge, or an option for avoiding the roads unsuited for caravans.

Because the AA Route Planner is only a journey planning application, it doesn't provide any services about POI information.

Note that the latest version of the AA Route Planner during this report is 1.7.706.

2.2.2.1 Comparison with DCMS

The service of AA Route Planner is tailored predominantly towards vehicle commuters, so it only offers a basic routing service to the drivers with succinct information. Comparing with the DCMS, the similar area is to plan user-defined routes, but AA Route

Planner doesn't provide any services about POI information. So actually, as a travelling tool, it is unsuitable to travellers.

Contrast Issues	AA Route Planner	DCMS
Plan routes based on users' preference	Yes	Yes
Search parameter refinement	Yes	Yes
Support multiple transport modes	No	Yes
The approaches of displaying route data	Map &Text	Map & Text
POI information retrieval	No	Yes
POI information management	No	Yes
Experiences sharing	No	Yes
Review translation service	No	Yes

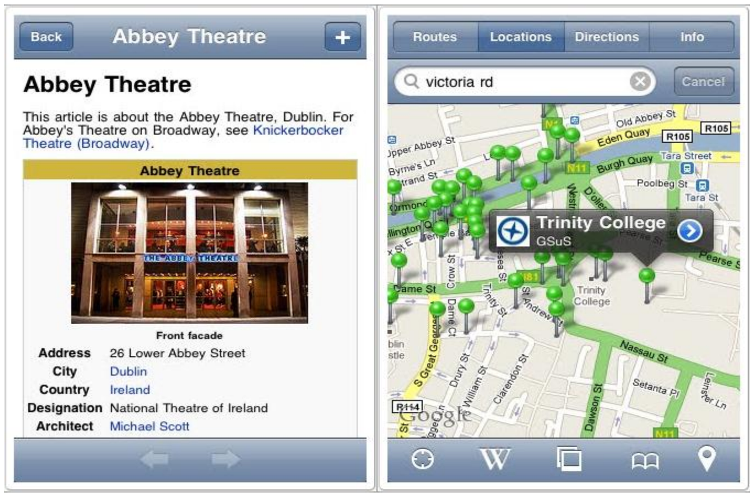
Table 2 - The results of comparison between AA Route Planner and DCMS

2.2.3 Dublin Buster

The Dublin Buster application is an iPhone application written by Yunhe Shi [4]. It is designed for convenient access to the information relevant to traveling on Dublin Bus.

Dublin Buster allows users to search bus routes based on route numbers and locations, locate nearby bus stops, display bus routes on the map, and access bus timetables. Moreover, this app has a local database of more than 160 bus routes in Dublin, so in the no Internet environment, the users also can easily and fast get possible bus routes between two locations.

In the paid version of this app, it also provides some advanced routing services, such as offering easy-to-use local timetables and directions from users' current location to a specified bus stop. Moreover, based on the Wikipedia location-based article search service and Panoramio location-based photo search service, the Dublin Buster app offers users nearby POIs information and photos for exploring and learning about Dublin (see *Figure 6*).



Abbey Theatre

This article is about the Abbey Theatre, Dublin. For Abbey's Theatre on Broadway, see Knickerbocker Theatre (Broadway).

Abbey Theatre

Front facade

Address	26 Lower Abbey Street
City	Dublin
Country	Ireland
Designation	National Theatre of Ireland
Architect	Michael Scott

Figure 6 – Search and display POIs [Dublin Buster]

Note that the latest version of the Dublin Buster app during this report is 1.2.12.

2.2.3.1 Comparison with DCMS

So far, of the various mobile applications, the Dublin Buster is the most similar to the proposal of DCMS. It can not only offer a route planning service between users-defined locations, but also provide POI information. But it also has its limitations. The first one is it is can only plan a route based on a single transportation mode - bus. The public transportation network in Dublin is very complex and extensive, so it will be a good solution for routing service if integrating various transportation modes into a single user-defined journey. Although the Dublin Buster allows users to search POI information, it doesn't support experience sharing and POI information management.

Contrast Issues	Dublin Buster	DCMS
Plan routes based on users' preference	Yes	Yes
Search parameter refinement	No	Yes
Support multiple transport modes	No	Yes
The approaches of displaying route data	Map	Map & text
POI information retrieval	Yes	Yes
POI information management	No	Yes

Experiences sharing	No	Yes
Review translation service	No	Yes

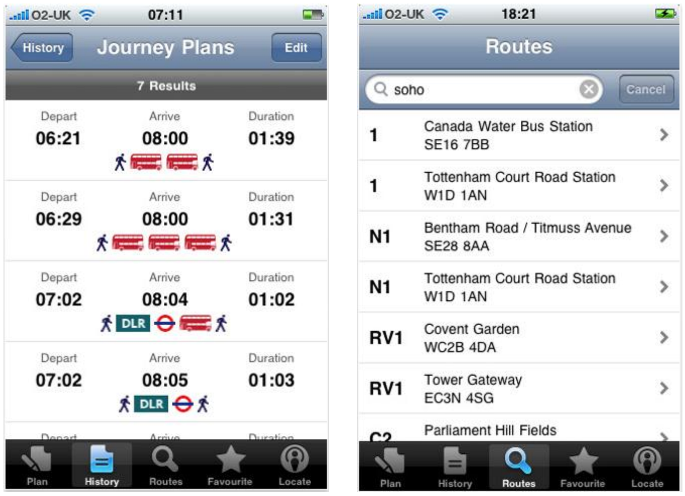
Table 3 - The results of comparison between Dublin Buster and DCMS

2.2.4 *London Bus*

The London Bus is written by Malcolm Barclay and running on the iPhone platform [5]. Its major responsibility is to offer the information about the London Bus networks. This app covers the main daytime and nighttime network, containing nearly 600 detailed routes and more than 20,000 individual bus stops. Moreover, it also provides a journey planner that can help users plan a route base on multiple London's public transports, such as Bus, Tube or Rail.

The journey planner service provided by London Bus is based on the Transport for London (Tfl) web service, so it can help users plan a journey using a number of means of public transport. The principle public transportation modes supported by the London Bus app are Tube (underground), Bus, DLR, Rail, Tram, Coach, River, and cycling service. Under this service, users firstly need to provide a start position and a destination with refined parameters, such as partial address or POI, postcode, transport station, and their current location. And then they need to select transport modes that they want to use. Last the departure or arrive time also need to be provided to complete the journey plan. The results of planning a journey are shown on *Figure 7*. Furthermore, this app also offers some advanced settings can optimize the planning requirements, such as the maximum walking time, walking speed, or cycling options. In this service, the Google Map service is used for displaying bus stop for any part of journey.

Besides the above mentions, the London Bus app also offers a service to find the bus routes by entering a locality name, for example "soho", or route number (see *Figure 7*). And by using the "Locate Me" function, this app can help users to find the nearest 20 bus stops and routes that run around users. In additional, the route timetables information also be provided by this app.



7 Results

Depart	Arrive	Duration
06:21	08:00	01:39
06:29	08:00	01:31
07:02	08:04	01:02
07:02	08:05	01:03

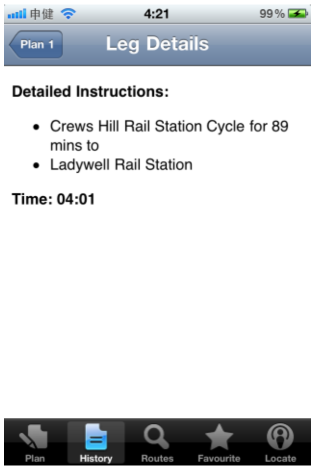
1	Canada Water Bus Station SE16 7BB
1	Tottenham Court Road Station W1D 1AN
N1	Bentham Road / Titmuss Avenue SE28 8AA
N1	Tottenham Court Road Station W1D 1AN
RV1	Covent Garden WC2B 4DA
RV1	Tower Gateway EC3N 4SG
C2	Parliament Hill Fields

Figure 7 – Journey plans and bus routes [London Bus]

Note that the latest version of the London Bus app during this report is 3.0.2.

2.2.4.1 Comparison with DCMS

If only evaluate the route planning service, based on the TfL, the London Bus is perhaps the most well implemented, and user friendly. It not only supports planning a route based on user-selected transport mode, start point, and destination, but also can calculate the route across multiple transport modes. Moreover, it also provides some advanced settings that can be used to optimize the planning requirements.



Detailed Instructions:

- Crews Hill Rail Station Cycle for 89 mins to
- Ladywell Rail Station

Time: 04:01

Figure 8 – The route plan instruction [London Bus]

There are however a number of drawbacks comparing with the DCMS. Chiefly among the limitations is the quit simple instruction for a route plan. It can only represent a route plan by text instruction (see *Figure 8*). Moreover, when using the London Bus to plan a route, users can't select waypoints. Because the London Bus is just a route planer tool, it doesn't support any services about the POI information.

Contrast Issues	London Bus	DCMS
Plan routes based on users' preference	Yes	Yes
Search parameter refinement	Yes	Yes
Support multiple transport modes	Yes	Yes
The approaches of displaying route data	Text	Map & text
POI information retrieval	No	Yes
POI information management	No	Yes
Experiences sharing	No	Yes
Review translation service	No	Yes

Table 4 - The results of comparison between London Bus and DCMS

2.2.5 Yelp

Yelp.com is a social networking, user review, and local search web site. As the early of 2010, it has more than 31 million monthly unique visitors [6]. Currently, the services it provides are as following.

One of services the Yelp provides is the POI information retrieval. This service allows users to find the POIs concerning a certain topic, not only the place name. A typical search includes what the user is finding (e.g. taco, cheap dinner, Max's) and the location from which the search is to be performed, entered as an address, neighborhood, city, state, or zip code (see *Figure 9*).

Not only provides the POI information retrieval, the Yelp also allows users to create their own POI information or edit existing POI information.

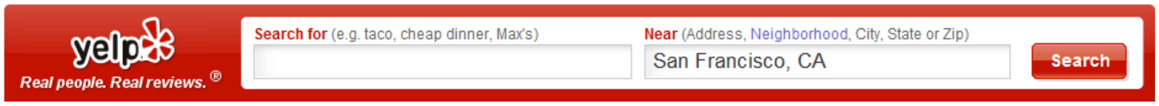

yelp
Real people. Real reviews.®
Search for (e.g. taco, cheap dinner, Max's)
Near (Address, Neighborhood, City, State or Zip)
San Francisco, CA
Search

Figure 9 – POI Information search service [Yelp.com]

Moreover, the Yelp provides a platform where users can share their travelling experience with each other by means of reviews, rating, and photos.

In August of 2007, the Yelp released a free RESTful and JSON-based application programming interface. These API provides access to POI information, reviews, photos, and ratings.

Based on its API, the Yelp also develops some mobile-based applications across multiple mobile platforms, such as Yelp for BlackBerry, Yelp for iPhone & iPod Touch, Yelp for Android, and Yelp for Palm Pre. Through combining the capabilities of mobile devices, the Yelp mobile applications also provide some services that are not included in its web-based application. For example, searching nearby POI information based on a user's current location (see *Figure 10*).

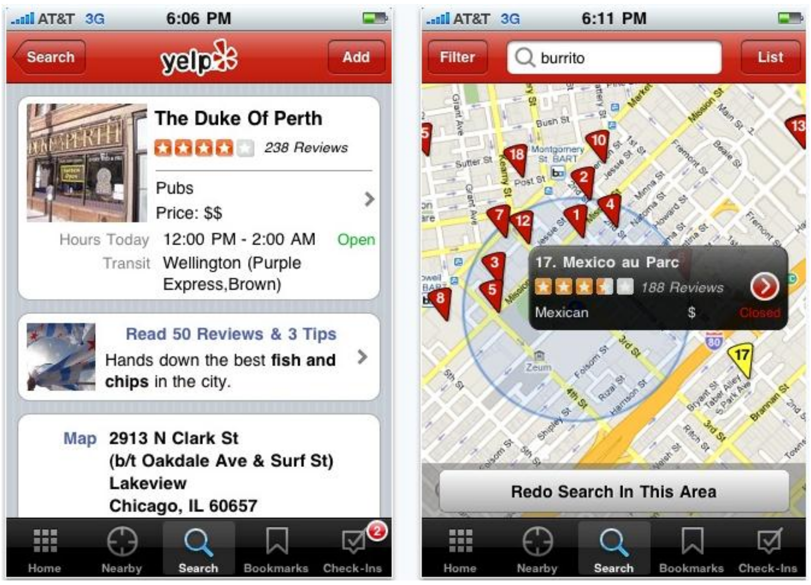


Figure 10 – Search nearby POI information in Yelp's iPhone application [Yelp]

2.2.5.1 Comparison with DCMS

If viewed solely from the POI Information service, the Yelp and DCMS are much similar. They all support POI information retrieval, POI information management, and travelling experiences sharing, but comparing with DCMS, the Yelp also has two small drawbacks. First, it doesn’t support the review translation service. Because most of reviews in Yelp are written in English, they can’t be understood by non-English speaking users without translations. Second, the Yelp doesn’t provide some API that can be used to manage POI information, so its mobile-based applications will not allow users to add and edit POI information.

Because the Yelp focuses on the business ratings and reviews, it doesn’t support any types of journey planning.

Contrast Issues	Yelp	DCMS
Plan routes based on users’ preference	No	Yes
Search parameter refinement	No	Yes
Support multiple transport modes	No	Yes
The approaches of displaying route data	No	Map & text
POI information retrieval	Yes	Yes
POI information management	Yes	Yes
Experiences sharing	Yes	Yes
Review translation service	No	Yes

Table 5 - The results of comparison between Yelp and DCMS

2.3 Summery

The most of applications or systems evaluated in this chapter are for travelling purpose. According to their functionalities, roughly, they can be divided into three categories:

- The applications focus on route planning, such as London Bus or TomTome.
- The applications focus on POI information service, such as Yelp.
- The applications focus on both POI information service and route planning service.

If only evaluated the route planning service, the London Bus, including its service provider Transport of London, is the most similar one to the DCMS. And the followed similar applications or systems are TomTom, and then AA Route Planner. But if viewed solely from the POI information service, the closest system currently available to the DCMS is the Yelp and its mobile-based applications. And for the Dublin Buster, although it focuses on both route planning service and POI Information service, these two services are limited.

Thus, to sum up the points which discussed in this chapter, the critical areas or other difference between the DCMS proposal and the current state of the art can be summarized as following:

- The DCMS will provide a route planning service that can calculate a route across different individual public transport service. For the complex and extensive public transportation networks in Dublin, integrating various transportation modes into a single user-defined journey will be the best solution for route planning.

- Some users will often provide inaccurate or imprecise travel information. Spelling mistakes, non-existent street name and or address suffixes (street/road/lane), all lead to uncertainty in planning a route. Therefore, the address refinement service is necessary that can be used to eliminate input errors, offering a list of intended (corrected) locations to resolve any ambiguity [1].

- The previous travelling experiences will be very helpful for the travellers in a strange city. Thus collecting POIs' reviews and providing a friendly reviews query interface will be necessary considerations by a travelling application. The DCMS will not only offer these functions, but also allow travellers to rate a certain place. This will bring about a distinct feeling to the other visitor how well a place is.

- An excellent POI review without understanding is unhelpful, so a translation service should be involved in a mobile travelling system. In order to implement this service, The DCMS will use a Machine Translation online service (e.g. Google Translation service) to provide a not perfect (the accuracy and fluency of the machine translation result are not satisfied currently) but very useful function for the travellers from different countries.

3. Design

"Imagination is more important than knowledge." – ***Albert Einstein***

3.1 System Requirements Specification

According to the state of the art studies in the previous chapter, the central idea behind this project is proposed to design and implement a travelling information mobile system in order to provide some web services for the mobile-based travelling application. And based on the functionality, this system is divided into three components: POI Information System, STIS Web Service, and Prototype Mobile Application.

3.1.1 POI Information System

This system can offer some interfaces that allow clients to query or provide information about POIs. The specific services is proposed to be provided in this system are as following.

- *POI information retrieval*. As a POI information system, it will certainly provide a service that is responsible for searching POIs information. Under this searching service, client side applications are allowed to query POI by its name or related tags. Tags are like keywords or labels that users add to POI to make it easier to find by the others. For example, users can tag a restaurant with phrases like "pizza", "cheap", "delicious", and "city center". Later if somebody looks for restaurants of pizza, they can just provide a keyword "pizza" to get all restaurants that have been tagged that way. Moreover, this POI system also offers a service that allows clients to query nearby POIs by a center location. Based on this service, client side applications easily implement some POI searching functions by location, such as providing POIs nearby users' current location.

- *POI information management*. Under this service, each registered user of client side applications is allowed to create and edit POI information. That means all of the information of POIs is provided by users. The information for creating a new POI should be provided by registered users is a place name, address, phone number, description, geo-location (latitude and longitude), and relevant tags.

- *User management*. This type of service allows client side applications to register/edit users in the POI information system. The reason why this system involves user management is because, for the secure purpose, it needs to know who provides information (e.g. POI information, reviews, rating, and photos). The information should be provided by users when they register an account is username, password, and default language. The default language normally is a user's mother language, which will be used by the POI Information System when a user searches relevant reviews for a certain place.

- *Travelling experience sharing*. Experience sharing will be another welcome service should be involved in a travelling system. The proposal approaches for sharing experience will be offered in the POI information system are reviews, rating, and photos. For the review, application users can draft some reviews that are used to describe what the impression about a certain place. Besides reviews, users can also rate a place by stars (the full mark is five stars). This will give the other users a distinct feeling how well this place is. The last approach is using photos to show what users saw in a certain place. In addition, a translation function will be involved in this service. The reason why this service need a translation function is because the review will be drafted by travellers in different language, in order to let every user read every review, the translation function is necessary.

3.1.2 STIS Web Service

The Smart Traveller Information System is a part of iTransit System and designed by Shane Brennan in 2006. It is a service which assists a traveller in choosing, refining, and then following the route of a pre-determined journey from a chosen starting point to a destination. The aim of this system is to provide users with a means of planning multi-modal routes, using user preferences as the sole basis for creating route plans. It coordinates between multiple transports networks to enable end-to-end planning suing various modes of transport. The STIS supports multi-modal routes using up to five different modes of transportation, such as Walking, Driving, Cycling, Bus, and Luas. [1]

The STIS Web Service is built on the STIS. Through interacting with STIS, the STIS Web Service is proposed to provide some interfaces that allow client side applications to plan a route on one or more user-defined transportation modes.

3.1.3 Prototype Mobile-based Application

The prototype mobile-based application is proposed to use and combine above two systems to implement following corresponding functionalities for travellers:

- Register/edit user information.
- Login/logout.
- Search POI information by place name or tag.
- Search nearby POI information by a user selected center location and radius.
- Display search results (POIs) on the map.
- Manage POI information.
- Search reviews and rating for a certain place.
- Translate the contents of the review based on user preference.
- Draft reviews and rating for a certain place.

- Plan a route based on user-defined start location, destination, and some waypoints.
- Draw the route line on the map.

3.2 Use Case Scenario

As the discussion in the system requirements specification, the DCMS is composed of three components: POI Information System, STIS Web Service, and Prototype Mobile-based Application. Therefore, in this section, three Use Case Diagrams will be involved that helps discover the requirements of these components from the user's perspective.

3.2.1 POI Information System Use-Case

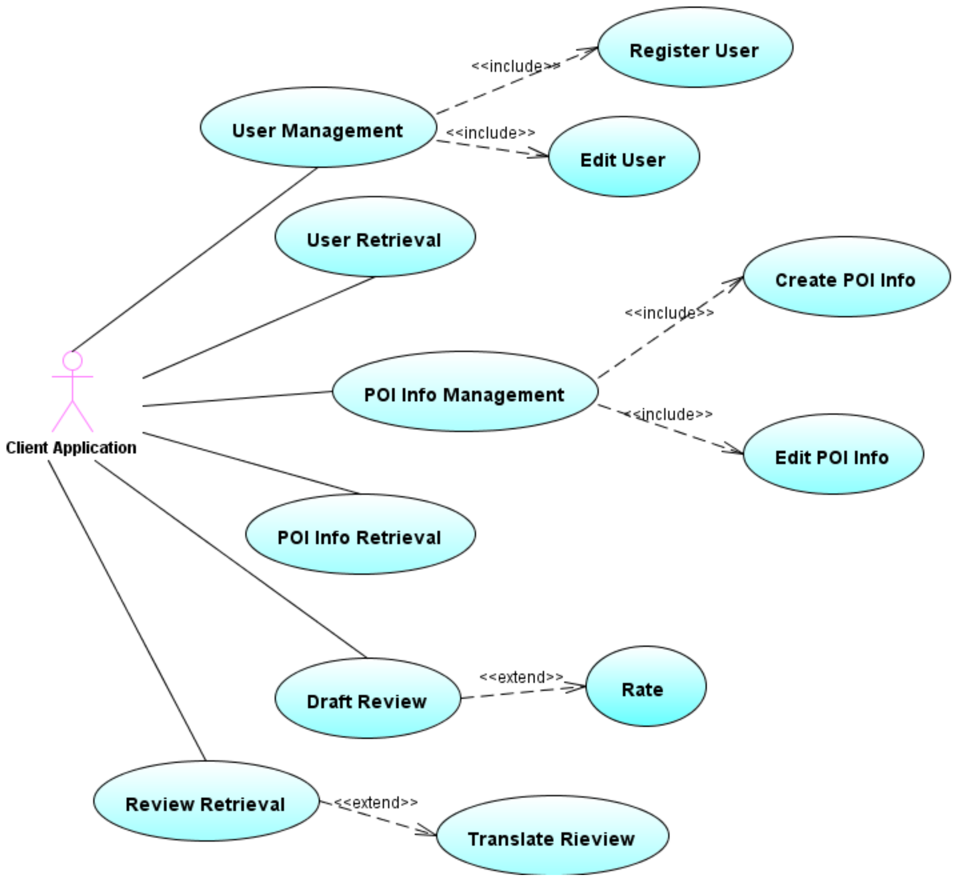


Figure 11 – Use Case Diagram of the POI Information System

The use-case model in above figure captures the usage scenario of the POI Information System, from the perspective of a client application.

3.2.2 STIS Web Service Use-Case

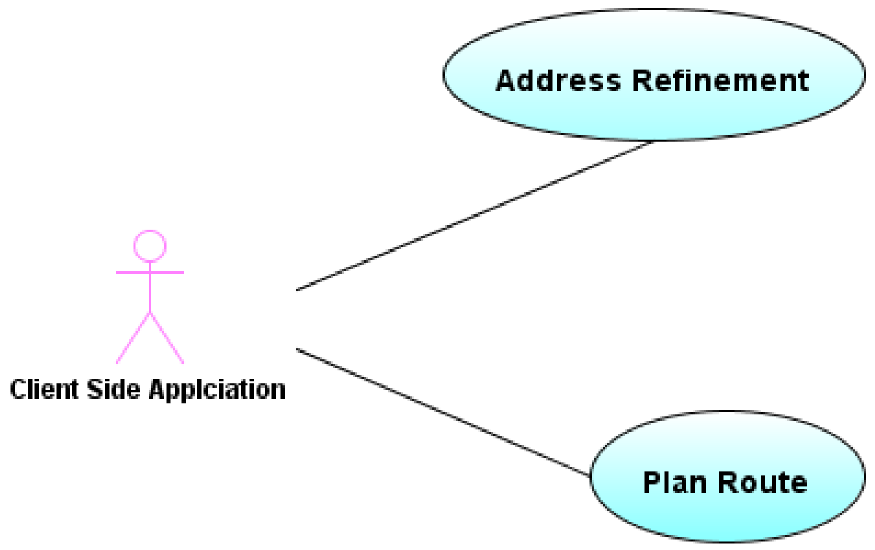


Figure 12 – Use Case Diagram of STIS Web Service

This use-case model discovers the requirements of the STIS Web Service from users' perspective. In this case, the user case "Address Refinement" shows a web service that helps clients to refine addresses. The reason why the STIS Web Service provides the address refinement web service is because some users will often provide inaccurate or imprecise travel information. Spelling mistakes, non-existent street names and/or missing address suffixes (street/road/lane, etc.), all lead to uncertainty in planning a route. Therefore, based on the address refinement service provided by ***STIS***, the STIS Web Service offers this web service to eliminate these input errors, offering clients a list of intended locations to resolve any ambiguity.

3.2.3 Prototype Mobile-based Application

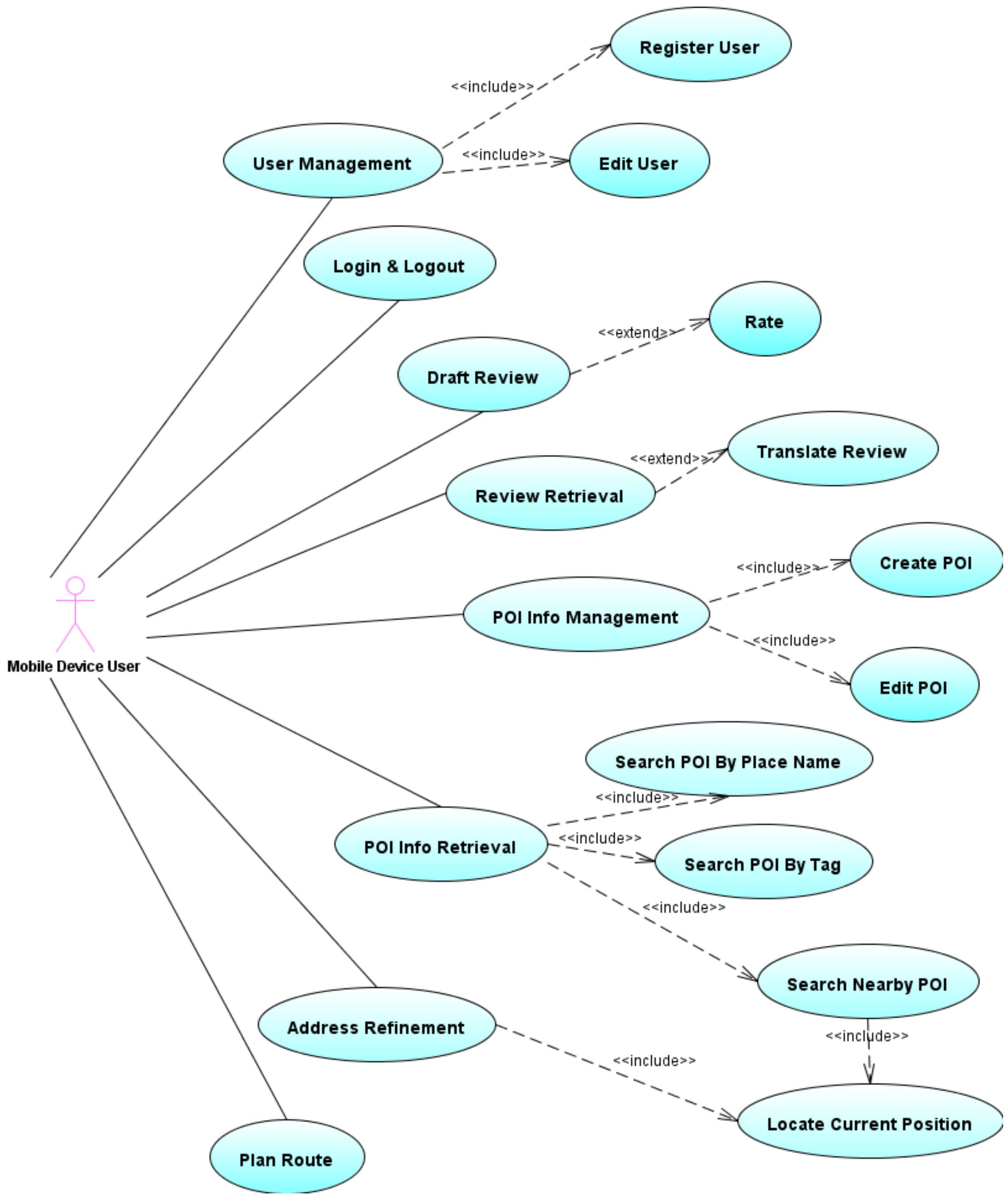


Figure 13 – Use Case Diagram of the Prototype Mobile-based Application

This use-case modal shows the requirements of Prototype Mobile-based Application from a mobile device user’s perspective.

3.3 System Architecture

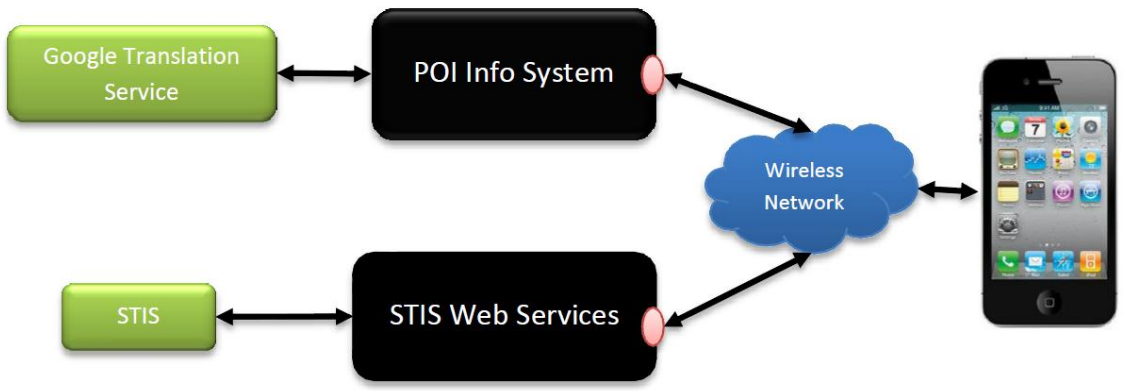


Figure 14 – High level system architecture

In this project, the Prototype Mobile Application is designed and developed on the iOS platform which is the operating system of iPhone, iPod Touch, and iPad. The major responsibility of this prototype application is to invoke some web services provided by POI Information System and STIS Web Service through a wireless network, and then based on these services to implement several functions for travelling purpose.

The POI Information System is responsible for offering some web services to the mobile-based application. In this project, these services are POI information retrieval, POI information management, review management, and user management. Moreover, the POI Information System also makes a method call to a translation service from the Google Translation. The aim of using a translation service in this system is to translate reviews to some users based on their mother language.

The STIS Web Service is built on the STIS and responsible to provide some interfaces that allow client side applications to plan a route on one or more user-defined transportation modes. Actually, the STIS has provided a handler to deal with the Http Requests from client side application. Therefore, the mobile application can connect the STIS directly without through the STIS Web Service.

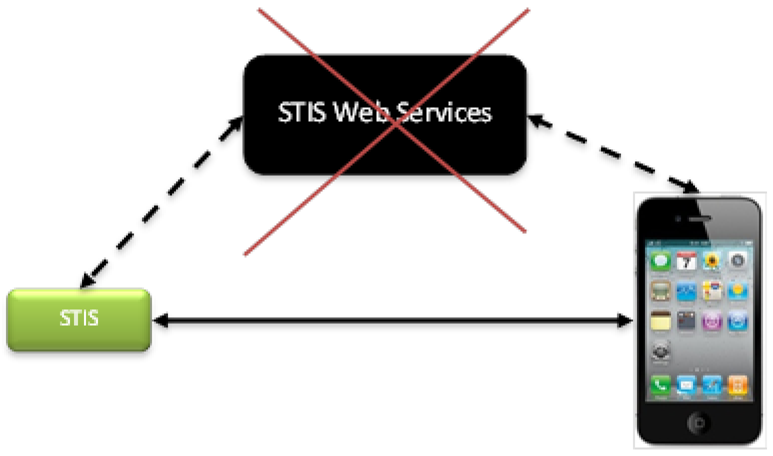


Figure 15 – Is the STIS Web Service necessary?

So, is the STIS Web Service necessary in this system?

Because STIS provide a XML interface for receiving user requests and sending response, if a mobile application intends to connect the STIS, it should set up a HTTP connection to the STIS Servlet, formatting application requests into XML, sending the requests over the HTTP link, parsing the XML responses, and passing the parsed information back to the corresponding method. So, now there is a problem that if a mobile application needs to call a route planning service provided by STIS, its developers have to know the XML schemas for the request and response, and then write lots of codes for establishing HTTP connection, formatting XML requests, and parsing XML response. Therefore, in order to solving this problem, the STIS Web Service should be involved.

The STIS Web Service can provide some web services that are designed based on the Simple Object Access Protocol (SOAP). Thus according to the WSDL file, the mobile application developers just need to use a *code generator* to generate some codes for calling SOAP services. For example, the wsdl2objc can generate Objective-C code from WSDL, and the JAX-WS can generate JAVA code from WSDL. So in brief, the aim of STIS Web Service is to provide an easier way to access the STIS services.

In the next sections, these three components will be explained in detail.

3.4 POI Information System

3.4.1 Database Design& Data Dictionary

As the discussion in the system requirements, the POI Information System will persist the POI data, user data, and review data to the database, so the database design is a necessary step involved in the system design. According to the requirements, the identified entities in this system are User, POI, POITag, Review, and Photo.

3.4.1.1 Data Dictionary of User

User: this table is used to hold the user data		
Attribute	**Type**	**Description**
userId	INT	It is the primary key of this entity and an auto incremental number.
Username	VARCHAR(256)	A unique name.
password	VARCHAR(32)	It is the user's password that is encrypted by MD5, so its size is 32.
defaultLanguage	VARCHAR(128)	The user's default Language, e.g. mother language.
isAvailable	INT	This filed indicates whether this user is available.

Table 6 – The data dictionary of entity "User".

3.4.1.2 Data Dictionary of POI

POI: this table is used to hold the POI data		
Attribute	**Type**	**Description**
poiId	INT	It is the primary key of this entity and an auto incremental number.
placeName	VARCHAR(128)	The place name.

address	VARCHAR(256)	The address of a place.
Latitude	DOUBLE	A parameter of Geo-Location.
Longitude	DOUBLE	A parameter of Geo-Location.
Phone	VARCHAR(64)	The office phone number of a place.
Description	LONGTEXT	The description of a place
isVisible	INT	This filed indicates whether this place can be retrieved.
lastEditeDate	DATETIME	The latest date and time of editing a place.
lastEditUser	INT	The latest user ID of editing a place. This is the foreign key of this entity and mapping to the User.userId.

Table 7 – The data dictionary of entity "POI".

3.4.1.3 Data Dictionary of POITag

POITag: this table is used to hold the tags information		
Attribute	**Type**	**Description**
tagId	INT	It is the primary key of this entity and an auto incremental number.
tagName	VARCHAR(64)	The text of a tag.
poiId	INT	The POI ID that a tag is associated with. It is the foreign key of this entities and mapping to the POI.poiId.

Table 8 – The data dictionary of entity "POITag".

3.4.1.4 Data Dictionary of Review

Review: this table is used to store the review data		
Attribute	**Type**	**Description**
reviewId	INT	It is the primary key of this entity and an auto incremental number.
Contents	VARCHAR(1024)	The contents of a review.
Rate	INT	The rate for a place.

createDate	DATETIME	The review creating date.
isVisible	INT	This filed indicates whether this review can be retrieved.
poiId	INT	The POI ID that a review is created for. It is the foreign key of this entity and mapping to the POI.poiId.
userId	INT	The user ID who creates this review.

Table 9 – The data dictionary of entity "Review".

3.4.1.5 The Relationship between Each Entity

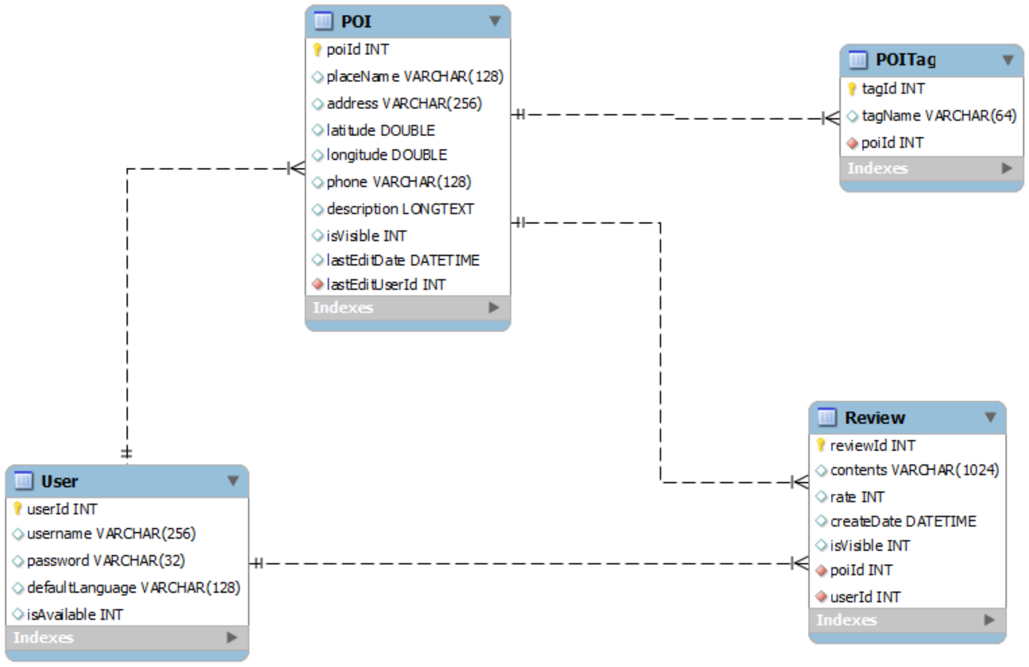


Figure 16 – The ER Diagram of the database in the POI Information System

The above figure shows the relationship between each entity.

- The relationship between User and POI is one to many. This is because one user account is allowed to create one or more POIs, but the last edit-user for a POI can only be one user.
- The relationship between User and Review is also one to many. The reason is like

above that one user account can draft one or more reviews, but a review only belong to one user.

- The relationship between POI and POITag is one to many. The same reason that one POI can own one or more tags, but one tag is only associated with one POI.
- The relationship between POI and Review is one to many. It is because one POI can own one or more reviews, but one review is only created for one POI.

3.4.2 Architecture Overview

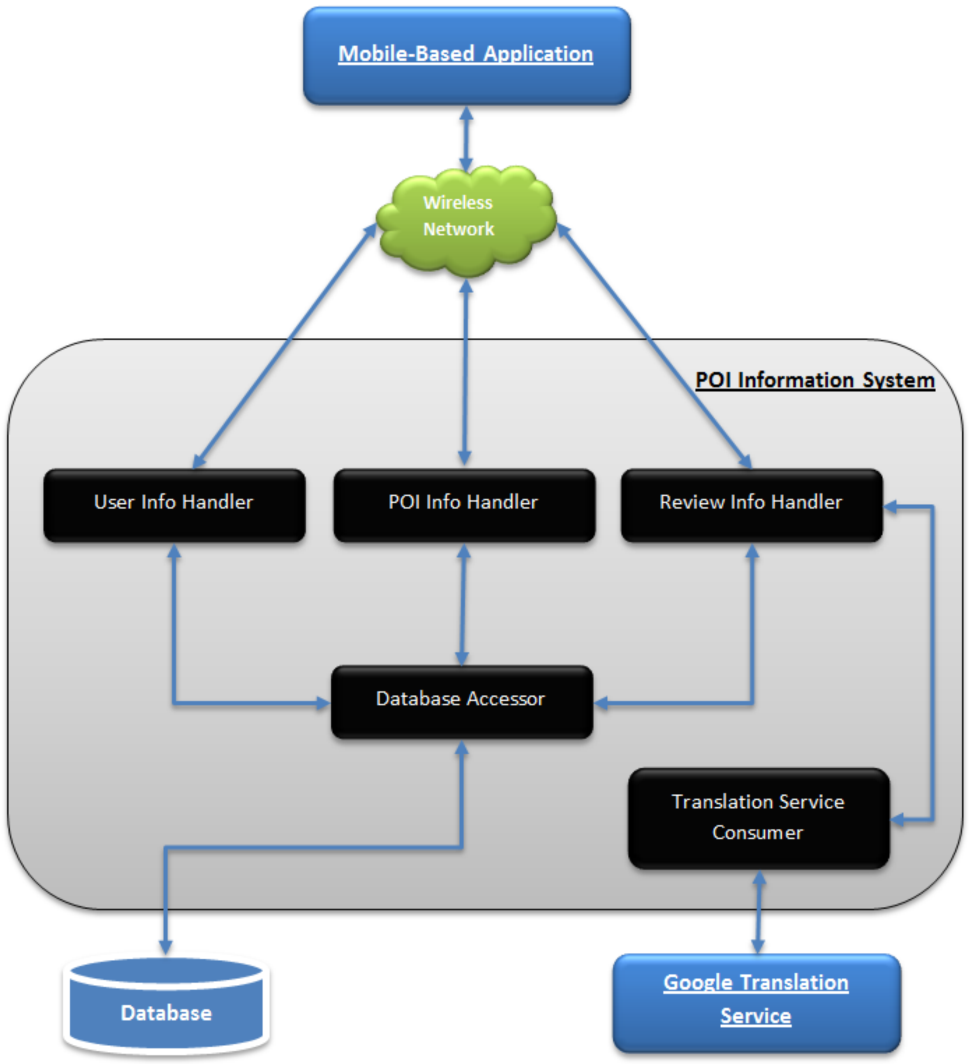


Figure 17 – The architecture of the POI Information System

3.4.2.1 User Information Handler

The responsibility of the User Information Handler module is to receive requests from client side application, and according to the type of requests, forward the formatted request data to the corresponding methods. And after the methods have dealt with the requests, this module will send a response back to the client.

Based on the system requirements, three web services should be provides in this module:

- *registerUser*. This aim of this service is to allow clients to register a new user account. And the parameters this service needs are username, password, and default language.
- *editUser*. This service will provide an interface that allows client side applications to edit existing user information. The parameters accepted by this service are username, old password, new password, and default language.
- *getUser*. This service is responsible for user retrieval. Under this service, the client side applications can pass a username and a password to query corresponding user information.

3.4.2.2 POI Information Handler

This module is responsible for handling requests from client side about POI information. When this handler receives a request, it will invoke a relevant method to deal with it, and then return the processing results back to the client.

According to the system requirements, the POI Information Handler should provide the following web services:

- *addPoi*. This service provides an interface through that the client side applications can easily implement a function to create a new POI in the POI Information System. The parameters of this web service are username, password, place name, address, latitude, longitude, phone, description, and tags. And if the client side only provides the address for identifying the place's location, this service also need to convert this address to the geo-data (latitude and longitude), and vice versa.

- *editPoi*. This service is responsible for editing existing POI information. The parameters needed by this service are username, password, place ID, place Name, address, latitude, longitude, phone, and description.
- *addTags*. This service allows client side applications to assign tags to a certain place. The parameters that client applications should provide for this service are tags string, and place ID. The tags string is a string that contains the tags that will be assigned for a certain place. And each tag in this string is separated by a comma, e.g. "pub, drink, nice place, beer".
- *getPoiByTitle*. This service allows client side applications to search POI information according to a place name. The parameter with this web service is keyword. Based on this service, all of the places will be searched if their place names include this keyword.
- *getPoiByTag*. The responsibility of this service is to search POI information according to a tag. The parameter of this web service is tag name.
- *getNearbyPoi.* This service can help client side application to search all places that are located in a certain area. The parameters with this web service are latitude, longitude, and radius. In these parameters, the latitude and longitude are used to describe the certain position of an area, and the radius is used to identify the scope of an area.

3.4.2.3 Review Information Handler

The responsibility of this module is to receive requests about the review information from the client side applications, and then invoke proper methods to handle them. The types of requests can be handled by this module are review creation request and review retrieval request, so two web services are created in this module:

- *addReview*. This service allows client side applications to create new reviews in this information system. The parameters used in this service are username, password, contents, rate, and POI ID.
- *getReview*. It is responsible for retrieving reviews for a certain place. Its parameters are POI ID, page, and target language. The page parameter is used to control which page of reviews should be returned. In default, the count of reviews in each page is 10. And the target language parameter is the language that will be used to translate the target reviews. If this parameter is null, it means the target reviews don't need to be translated.

3.4.2.4 Data Accessor

The Data Access module is responsible for accessing the database to query and update data. Because it is proposed to be built on the Hibernate that is an open source Java persistence framework, so it will be composed of the following components:

- *Data Access Objects.* They are some classes that derive from the Data Access Object (DAO) design pattern, providing an efficient way to query and update data in the database. The idea behind the DAO design pattern is to separate out the POJOs from the logic used to persist them into, and retrieve them from, the database. In this design pattern, the DAO class represents the operations that can be performed on a POJO type, such as UserDAO (a DAO class) for User (a POJO class).
- *Plain Old Java Object (POJO).* The POJO is just a normal Java object that can be persist to the database. Although a POJO can be constructed out of a selection of

table columns, or several POJOs can be persisted into a single table, in this project, each table in the database will be mapped to one POJO. Therefore, according to the database design, the POJO will be User, Review, Poi, and Poitag.

- *Hibernate Configuration File and Hibernate Mapping File.* The Hibernate Configuration file is a XML-based file that points to the database and XML mapping files. And the mapping files for each of the POJOs that map properties to columns in a table.

3.4.2.5 Translation Service Consumer

This translation service consumer module is about to be used by the Review Information Handler module for translating review contents based on the user demand. And in this module, the translation service will be provided by an online translation service – the Google Translation.

3.4.3 Class Design

According to the discussion about the system architecture, the class design will be as following:

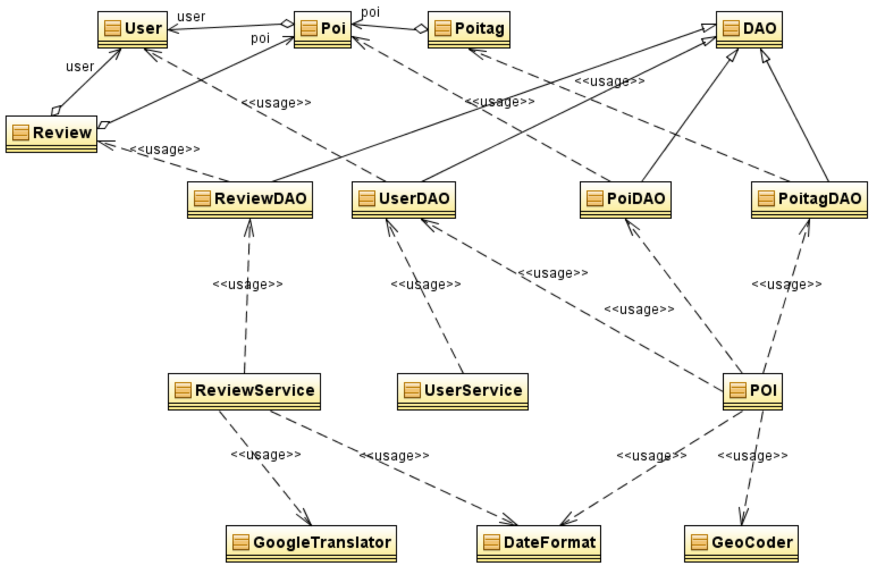


Figure 18 – The class diagram of the POI Information System

In this class diagram, the classes ReviewService, UserService, and POI are the handler modules discussed above. And all objects whose names include “DAO” are the Data Access Objects, such as ReviewDAO, UserDAO, PoiDAO, and PoitagDAO. The POJOs in the class design are represented by User, Poi, Poitag, and Review classes. In addition the class GoogleTranslator is the Translation Service Consumer module, and the class GeoCoder is a functional class that is used to convert the address to the geo-data (latitude and longitude) and reverse the geo-data to the address.

3.5 STIS Web Service

3.5.1 *Architecture Overview*

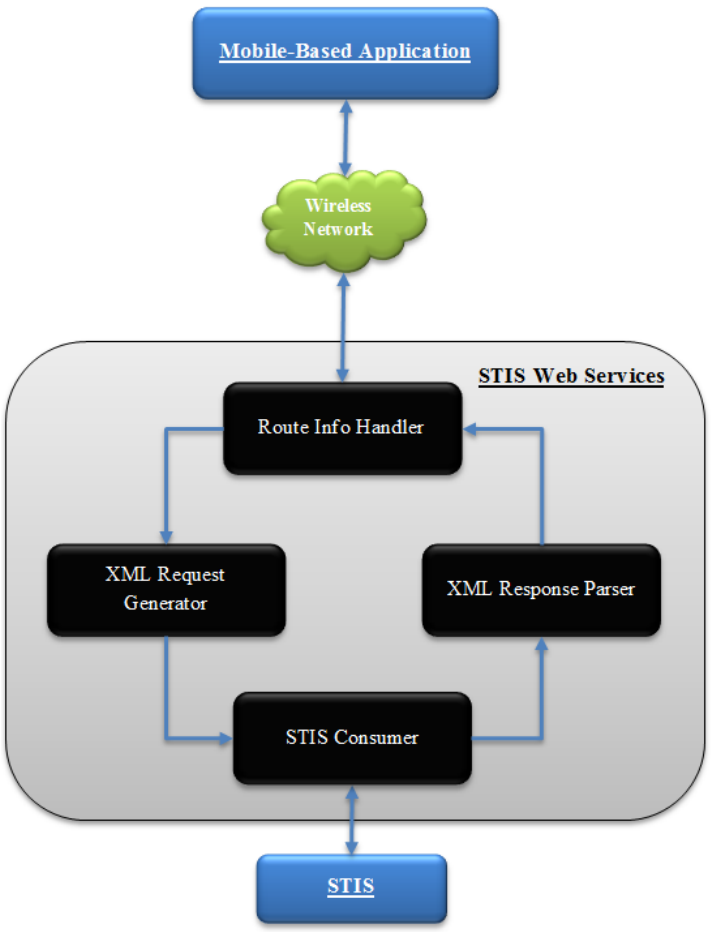


Figure 19 – The architecture of the STIS Web Service

3.5.1.1 Route Info Handler

The Route Info Handler module is responsible for receiving the requests from the client side applications, and then based on the type of receiving requests, selecting proper method to deal with them. In this module, there are two web services will be provided to the client applications:

- *getRefinementOptions*. It provides an address refinement service to the client side applications. Under this service, the inaccurate or imprecise address will be eliminated, and offering clients a list of intended locations to resolve any ambiguity. The parameters of this web service are a list of addresses.
- *getRoute*. Under this service, the client side applications can plan a route based on user-defined preferred transport mode, start location, destination, and some waypoints. The parameters should be provided to this service are a list of addresses that are used to represent the transport mode, start location, some way point locations, and destination.

3.5.1.2 STIS Consumer

This consumer module is responsible for interacting with the STIS. In the architecture diagram, it shows when this module receives a XML request that is created by XML Request Generator module, it will forward this request to the STIS. And after receiving the XML response from the STIS, it will pass it to the XML Response Parser to extract useful data for the Route Info Handler.

3.5.1.3 XML Request Generator & XML Response Parser

Based on the STIS request XML schema, the XML Request Generator module provides a function to create a XML request using the data passed from the Route Info Handler.

Similarly, based on the STIS response XML schema, the XML Response Parser can parse the receiving response from the STIS, and pass the parsed data to the Route Info Handler.

3.5.2 Class Design

According to the architecture of the STIS Web Service, the class design should be as following:

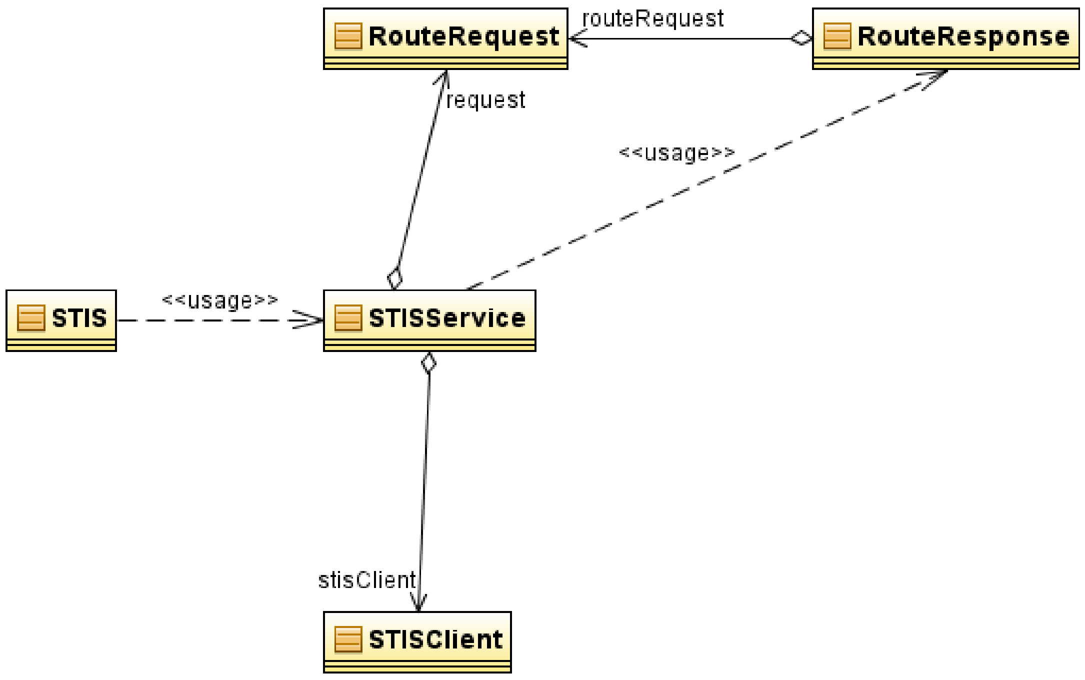


Figure 20 – The class diagram of the STIS Web Service

In this class diagram, the classes STIS and STISService are used to implement the functions of the Route Info Handler module, the class STISClient is representing the STIS Consumer module, and the classes RouteRequest and RouteResponse are, certainly, the XML Request Generator module and XML Response Parser module.

3.6 Prototype Mobile-based Application

Because, currently there are some kinds of mobile operation systems in the market, such as iOS, Android, Windows Mobile, RIM, Symbian, and Linux, the biggest challenge of designing architecture for the mobile application is platform independent. That means based on this design, the developers can implement a mobile application in any mobile platform.

3.6.1 Architecture Overview

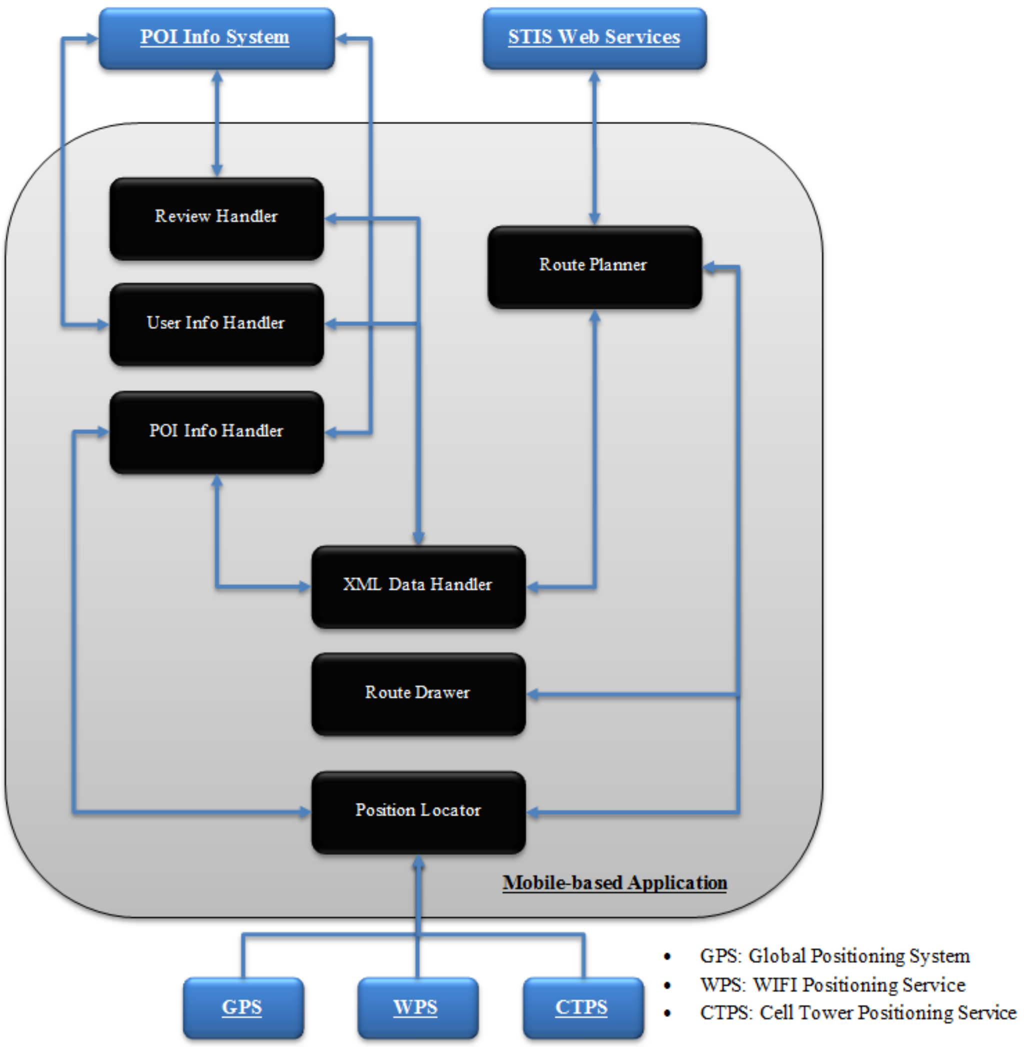


Figure 21 – The architecture of the Prototype Mobile-based Application

3.6.1.1 User Info Handler

According the system requirements, the User Info Handler module is proposed to provide the following functions to users:

- *Register user*. This function is implemented by invoking the web service *registerUser* from the POI Information System.
- *Edit user*. This function is based on the web service *editUser* that is also provided by the POI Information System.
- *Login and Logout*. Similarly, this Login function will invoke the web service *getUser* from the POI Information System, and according to the return result to determine whether to set this user's status as logged. And the Logout function is just to set the status to unlogged.

3.6.1.2 POI Info Handler

The POI Info Handler module is used to deal with any requests from users about the POI information. Therefore, based on the system requirements mentioned in previous section, this module should provide some functions as following:

- *Create POI*. Based on the web service *addPoi* provided by the POI information system, this function will allow users to add new place information. And when a user creates POI, this function not only allows this user to enter an address to identify this place's location, it should also provide a map view where the user can pin a position to obtain the geo-data (latitude and longitude).
- *Edit POI*. This function can be implemented by invoking the web service *editPoi* from the POI Information System. It allows users to edit existing place information. Like the Create-POI function, the Edit-POI function can also allow users to select a position in a map view.

- *Add tags*. By calling the web service *addTags* from the POI Information System, this function can be used to add some tags for a certain place.
- *Search POI by place name*. Based on the web service *getPoiByTitle* provided by the POI Information System, this function is proposed to allow users to enter a keyword, and then by using this keyword to search qualified places information. And if there is no result found, this function should notify users whether to create a new place.
- *Search POI by tag*. This function can be implemented by calling the web service *getPoiByTag* from the POI Information System. It allows users to enter a tag according to that to search POIs.
- *Search nearby POI*. By invoking the web service *getNearbyPoi* from the POI Information System, and based on user-defined center position and radius, this function allows users to search the places that are located in a certain area. Certainly, this function also provides a map view on that the users can pin a position to get geo-data for the center position.
- *Display POI on the map*. By using each POI's latitude and longitude, the POI Info Handler module can pin the searched results (qualified POIs) on the map.

3.6.1.3 Review Info Handler

This handler module is proposed to provide two functions to users about the review information:

- *Draft review*. Based on the web service *addReview* provided by the POI Information System, this function can allow users to draft a review for a certain place. When drafting a review, the users can not only write a short comment for a certain place, but also rate it with stars (5 stars is the full mark).
- *Search reviews for a certain place*. Based on a selected POI, this function can call

the web service *getReview* from the POI Information System to provide a list of relevant reviews for users.

3.6.1.4 Route Planner

According to the web services provided by the STIS Web Service, the Route Planner module is proposed to provide two functions about planning a route:

- *Address refinement.* Based on the web service *getRefinementOptions* provide by the STIS Web Service, the Route Planner module offers this function to refine a list of addresses provided by users to eliminate the inaccurate and imprecise input, and providing a list of intended locations to resolve any ambiguity.

- *Route planner*. By calling the web service *getRoute* from the STIS Web Service, this function can calculate a route based on the user-defined preferred transport mode, start point, some way points, and destination. After getting the route, the Route Planner module can draw this route line on the map view and also provide a list of instructions about this route, such as duration, distance, or involved transport mode's details.

3.6.1.5 XML Data Handler

Because the travelling mobile-based application needs to invoke a series of web services to implement its functions, it will design a handler module to deal with the XML request and XML response. Therefore, the XML Data Handler is just for that: generate XML requests and parse XML response.

3.6.1.6 Route Drawer

Briefly, this module is just used to draw a route line on the map view. Because its implementation depends on the mobile platform, a sample implementation in the iOS will be discussed in next chapter.

3.6.1.7 Position Locator

The position locator module can provide two functions for users:

- *Locate a position on the map*. This function allows users to select a position on a map view, and then based on this position to calculate the real world geo-data (latitude and longitude) and address.

- *Locate user's current location*. Based on three technologies, this function has the ability to determine where in the world. These technologies are GPS, cell tower triangulation, and Wi-Fi Positioning Service (WPS). GPS is the most accurate of the three but is maybe not available on some mobile devices. GPS reads microwave signals from multiple satellites to determine the current location. Cell tower triangulation determines the current location by doing a calculation based on the locations of the cell towers in the phone's range. Cell tower triangulation can be fairly accurate in cities and other areas with a high cell tower density but becomes less accurate in areas where there is a greater distance between towers. The last option, WPS, uses the IP address from mobile device's Wi-Fi connection to make a guess at the user's location by referencing a large database of known service providers and the areas they service. WPS is imprecise and can be off by many miles. [7]

3.6.2 Limitations of the Architecture

Although this architecture design tried to achieve platform independent, because of the different requirements of each function, it also has limitations in some specific functions:

- Because most of functions implemented in the mobile-based application need the Internet connection, the mobile device must have the Internet connectivity capability.

- For the “*locate a position on the map”* function, it requires a mobile OS that can support an API to convert a point (coordinate point: x & y) to the geo-location (latitude and longitude).
- In order to implement the “*locate user’s current location*” function, the mobile device has to support one of the three locating technologies at least. And correspondingly, the mobile OS should provide some APIs to obtain the geo-data.

4. Implementation

"The important thing in life is to have a great aim, and the determination to attain it." – ***Johan Wolfgang von Goethe***

4.1 Development Environment

As the discussion in the previous chapters, the Dublin Compass Mobile System is composed of three components: POI Information System, STIS Web Service, and Prototype Mobile-based Application. The development environments of the POI Information System and STIS Web Service are similar.

Hardware: Desktop PC

- Processor: Intel 2.2GHz Core 2 Duo T7500.
- Memory: 2GB DDR2-667 SDRAM.
- Video Card: NVIDIA GeForce 8600M GT with 256MB GDDR3.
- Hard Disk: 500GB 5400RPM SATA HDD.

Software:

- Operating System: Windows 7 Ultimate Version 64bits.
- Programming Language: Java SE (JDK 6 Update 21).
- Development IDE: Netbeans 6.9.1.
- Development IDE for STIS: Eclipse Europa 3.3.0.
- Application Server for the POI Information System & STIS Web Service: GlassFish Server Open Source Edition 3.0.1.
- Application Server for STIS: Apache Tomcat 6.0.26.
- Database Management System: MySQL Community Server 5.1.50.
- Unit Testing Tool: Junit 4.5 Testing Framework.

For the Prototype Mobile-based Application, because it will be implemented as an iPhone application in this project, the development environment is completely different with the MS Windows platform.

Hardware: MacBook

- Processor: Intel 2.0GHz Core 2 Duo.
- Memory: 2GB DDR3-1067MHz SDRAM.
- Video Card: NVIDIA GeForce 9400M GT with 256MB GDDR3.
- Hard Disk: 160GB 5400RPM SATA HDD.

Software:

- Operating System: Mac OS X 10.6.4.
- Programming Language: Objective-C.
- Development IDE: Xcode 3.2.3.
- Base SDK: iPhone Device 4.0.
- Testing Device: Simulator 4.0.

4.2 Common Implementation Issues

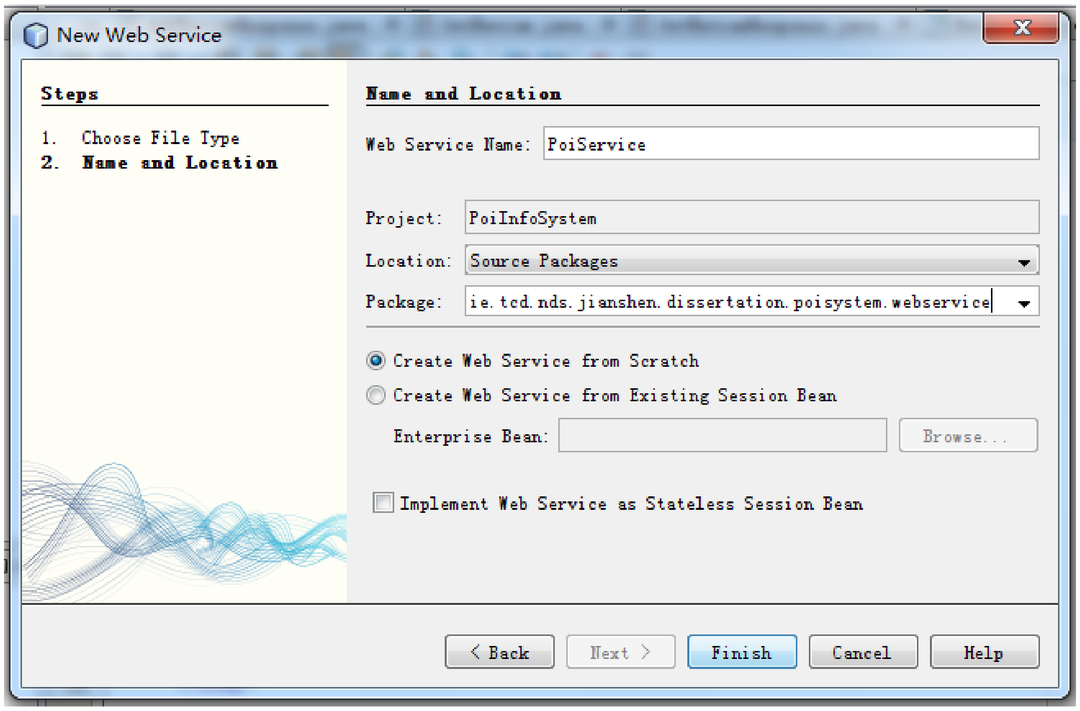
Based on the functionality design discussed in last chapter, the three system components will have a common implementation issues – web service. For the POI Information System and STIS Web Service, they need to provide some web services to the client side applications. That means they should implement a series of fundamental issues relating to monitoring Http connection, parsing XML request and generating XML response. Conversely, because the Prototype Mobile-based Application need to invoke some web services from the two components, it should also implement a number of issues to be a web service client, such as establishing Http connection, generating XML request and parsing XML response. Therefore, this section will focus on how to implement this common issue in these three system components.

4.2.1 *Web Service Implementation*

Depending on the Netbeans's powerful capability, it is very easy to create the web services for the POI Information System and STIS Web Service, nay, without any coding.

Here is a sample that will show how to create an operation *getPoiByTitle* in the web service *PoiService* for the POI Information System:

1. In this system, create a new web service called PoiService:



New Web Service

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Web Service Name: PoiService

Project: PoiInfoSystem

Location: Source Packages

Package: ie.tcd.nds.jianshen.dissertation.poisystem.webservice

(●) Create Web Service from Scratch

( ) Create Web Service from Existing Session Bean

Enterprise Bean:

[ ] Implement Web Service as Stateless Session Bean

Figure 22 – Create a new web service file in Netbean 6.9.1

2. Then in the PoiService console, click "Add Operation" button.

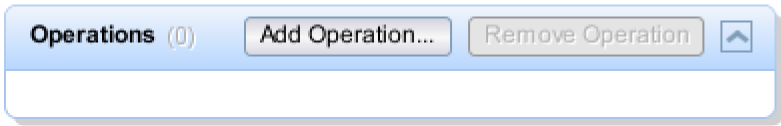
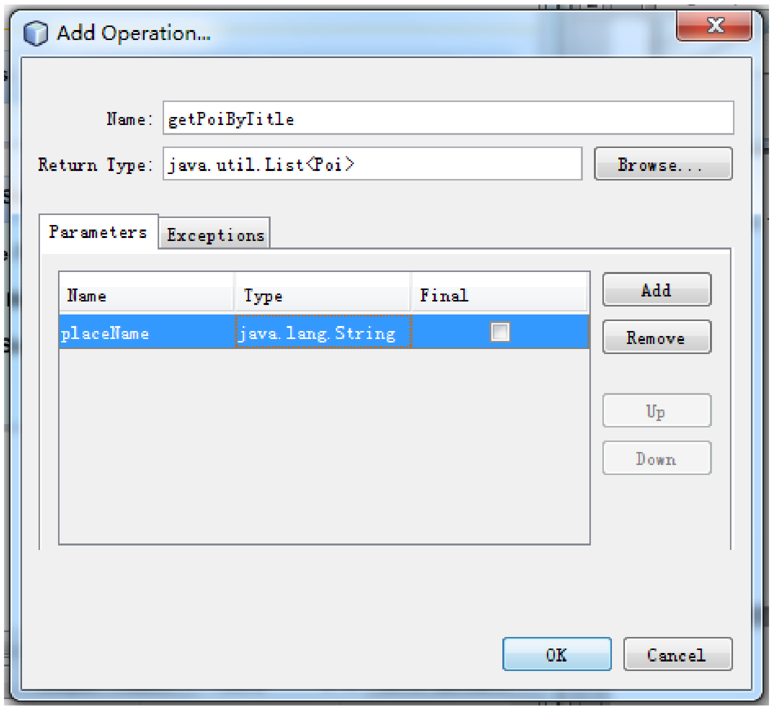


Figure 23 – Add an operation in the web service file

3. Create an operation called getPoiByTitle.



Name: getPoiByTitle
Return Type: java.util.List<Poi>

Name	Type	Final
placeName	java.lang.String	☐

Figure 24 – Create an operation called getPoiByTitle

4. Now the web service getPoiByTitle is created, and the next thing for developers is just to add some codes to make this service work.

```
@WebService()
public class PoiService {
    /**
     * Web service operation
     */
    @WebMethod(operationName = "getPoiByTitle")
    public java.util.List<Poi> getPoiByTitle(@WebParam(name = "placeName")
    String placeName) {
        // write some codes here to implement this service
        return null;
    }
}
```

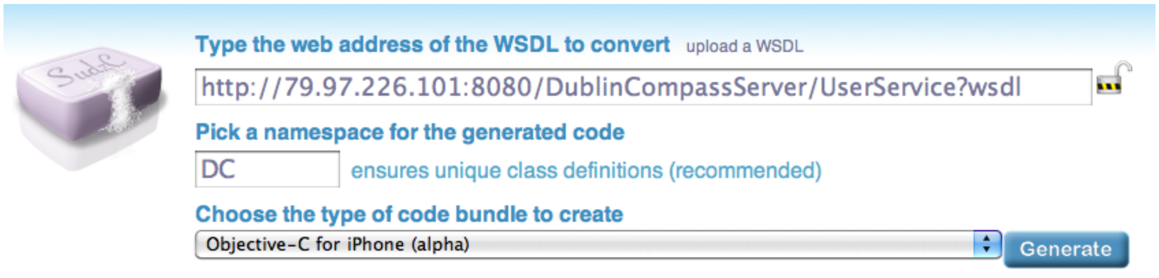
Code Table 1 – The codes of web service getPoiByTile that are generated by Netbeans 6.9.1

4.2.2 Web Service Client Implementation

Although the Xcode IDE didn't support the codes generation for implementing the SOAP web service client, but, fortunately, there are some tools for that purpose, such as wsdl2objc [8] or Sudz-C [9]. In this project, the selected tool is Sudz-C that is a .Net web site solution and uses XSLT to transform SOAP-based web service definition WSDL files into complete code package. While SudzC supports multiple platforms, its main focus is the code generation for Objective-C libraries for iPhone development.

By using this online tool, it is quite easy to implement the web service client in the Prototype Mobile-based Application. The following is a sample that shows the steps to generate Objective-C codes based on a WSDL file.

1. Enter a valid WSDL link into the proper textbox, and optionally pick a namespace for the generated code if necessary.


Type the web address of the WSDL to convert
upload a WSDL
Pick a namespace for the generated code
ensures unique class definitions (recommended)
Choose the type of code bundle to create
Generate

http://79.97.226.101:8080/DublinCompassServer/UserService?wsdl

DC

Objective-C for iPhone (alpha)

Figure 25 – By using the SudzC, to generate the Objective-C codes for implementing a web service client

2. After clicking the "Generate" button, a compressed file that includes the generated codes will be downloaded.

3. Unzip the downloaded file to obtain the generated classes.

4. Based on these generated classes, it is quite easy to implement the web service calling without any coding works about establishing a connection, generating XML

request or parsing XML response.

The following is the sample codes that can be used to call a web service *getUser.*

```
{
   @DCUserService *service = [DCUserServiceservice];
   [service getUser:self action:@selector(handleGetUser☺ username:@“name”
                                                              password:@“password”];
}

- (void) handleGetUser☹id)result{
         DcgetUserResponse *response = [DcgetUserResponse newWithNode:
                                                       [result objectAtIndex:0]];
         DcuserContents *userContents = response._return;
}
```

After the discussion about the common implementation issue, in the coming sections, some individual implementation issues for each system component will be explained.

4.3 POI Information System

As the discussion about the architecture design in last chapter, the POI Information System should implement five modules: Poi Info Handler, User Info Handler, Review Info Handler, Data Accessor, and Translation Service Consumer.

In order to implement the three handler modules, some web services need to be created and implemented, such as registerUser, addPoi, getPoiByTitle, getReview, etc. But in this section, it won’t focus on how to implement these three handlers, because, first like mention in the last section, it is quite easy to create web services by using Netbeans, and second there is no challenge to implement each web service. For example, in order to implement the web service registerUser, the only thing for developers is to encapsulate the receiving user’s information into a *User* object and invoke the method *add(User user)* that is provided by the class UserDAO (a DAO class that will be detailed in the section 4.4.1) to persist this object to the database.

Therefore, the focus of this section will be on the implementations of the Data Accessor and Google Translator.

4.3.1 Data Accessor

Like mentioned before, the Data Accessor is composed of three components: Hibernate Configuration File and Hibernate Mapping File, POJOs, and Data Access Objects. In this section, these components will be discussed.

4.3.1.1 Hibernate Configuration File

The configuration file is used by Hibernate framework to point the database link and POJO mapping files.

```
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/pois</property>
   <property name="hibernate.connection.username">root</property>
   <property name ="hibernate.connection.password">123</property>
    <mapping resource="ie/tcd/nds/jianshen/dissertation/server/pojo/Review.hbm.xml"/>
    <mapping resource="ie/tcd/nds/jianshen/dissertation/server/pojo/Poi.hbm.xml"/>
    <mapping resource="ie/tcd/nds/jianshen/dissertation/server/pojo/Photo.hbm.xml"/>
    <mapping resource="ie/tcd/nds/jianshen/dissertation/server/pojo/User.hbm.xml"/>
    <mapping resource="ie/tcd/nds/jianshen/dissertation/server/pojo/Poitag.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

Code Table 2 – An sample of hibernate configuration file

In this sample configuration file, the property “hibernate.connection.url” is used to represent the database link, the property “hibernate.connection.username” represents the connecting username, the property “hibernate.connection.password” represents the connection password, and the properties “mapping” are used to link the mapping files.

4.3.1.2 Plain Old Java Object & Hibernate Mapping File

The POJO is just a normal Java object that can be persist to the database. It can be constructed out of a selection of table columns, or several POJOs can be persisted into a single table, but in this project, each table in the database will be mapped to one POJO. The Hibernate Mapping files for each of the POJOs that map properties to columns in a table. The following is a sample that shows how to create and map a POJO for the table *User.*

```
Public class User  implements java.io.Serializable {
    private Integer userId;
    private String username;
    private String password;
    private String defaultLanguage;
    private Integer isAvailable;
    private Set reviews = new HashSet(0);
    private Set pois = new HashSet(0);

   //some getter & setter methods for each property
}
```

Code Table 3 – The sample codes for creating a *User POJO*

The properties *reviews* and *pois* (see *Code Table 3*) are used to represent the relationships between User and Review, and User and Poi. And its relevant mapping file is as following (see *Code Table 4*).

```
<hibernate-mapping>
  <class catalog="pois" name="ie.tcd.nds.jianshen.dissertation.server.pojo.User"
table="user">
    <id name="userId" type="java.lang.Integer">
      <column name="userId"/>
      <generator class="identity"/>
    </id>
    <property name="username" type="string">
      <column length="256" name="username"/>
    </property>
    <property name="password" type="string">
      <column length="32" name="password"/>
    </property>
    <property name="defaultLanguage" type="string">
      <column length="128" name="defaultLanguage"/>
    </property>
    <property name="isAvailable" type="java.lang.Integer">
      <column name="isAvailable"/>
    </property>
    <set cascade="save-update" inverse="true" lazy="false" name="reviews">
      <key>
        <column name="userId" not-null="true"/>
      </key>
      <one-to-many class="ie.tcd.nds.jianshen.dissertation.server.pojo.Review"/>
    </set>
    <set cascade="save-update" inverse="true" lazy="false" name="photos">
      <key>
        <column name="userId" not-null="true"/>
      </key>
      <one-to-many class="ie.tcd.nds.jianshen.dissertation.server.pojo.Photo"/>
    </set>
    <set cascade="save-update" inverse="true" lazy="false" name="pois">
      <key>
        <column name="lastEditUserId" not-null="true"/>
      </key>
      <one-to-many class="ie.tcd.nds.jianshen.dissertation.server.pojo.Poi"/>
    </set>
  </class>
</hibernate-mapping>
```

Code Table 4 – The sample hibernate mapping file for *User POJO*

4.3.1.3 Data Access Objects

In order to implement the data access objects, the base class used to manage the *connection session* should be created first. The benefit of implementing the base class is that provides an efficient way for a thread to retrieve and (if necessary) create its sessions with a minimal impact on the clarity of the code.

```
Public class DAO {
    private static final ThreadLocal cache = new ThreadLocal();
    private static final SessionFactory sessionFactory =
            new AnnotationConfiguration().configure().buildSessionFactory();
    protected DAO(){}
      //get the hibernate session
    public static Session getSession(){
        Session session = (Session) DAO.cache.get();
        if(session == null){
            session = sessionFactory.openSession();
            DAO.cache.set(session);
        }
        return session;
    }

     //begin transaction
    protected void begin(){
        getSession().beginTransaction();
    }

    //commit the transaction
    protected void commit(){
        getSession().getTransaction().commit();
    }

    //roll back the transaction's effect
    protected void rollback(){
        try{
            getSession().getTransaction().rollback();
        }catch(HibernateException e){
            e.printStackTrace();
        }
        try{
            getSession().close();
        }catch(HibernateException e){
            e.printStackTrace();
        }
        DAO.cache.set(null);
    }

     //close the session
    public static void close(){
        getSession().close();
        DAO.cache.set(null);
    }
}
```

Code Table 5 – The sample codes for the base class of the DAO design pattern

And then through inheriting this base class, the DAO classes can be built. Here will list a sample DAO class *UserDAO* that will offer some operations to perform on the POJO class

User.

```
Public class UserDAO extends DAO {

    public - 58 -oolean add(User user) throws DAOException {
        - 58 -oolean result = false;
        try {
            - 58 -oolean isAvailable = checkUsername(user.getUsername());
            if (isAvailable) {
                begin();
                getSession().save(user);
                commit();
                close();
                result = true;
            }
        } catch (HibernateException e) {
            rollback();
            close();
            throw new DAOException(e.toString());
        }
        return result;
    }

   public - 58 -oolean checkUsername(String username) throws DAOException {
   //some codes here to check whether the given username is existed in the database
    }

    public User get(String username, String password) throws DAOException {
    //some codes should be here to implement the user retrieval
    }

    public void update(User user) throws DAOException {
    //some codes should be here to implement the user modification
    }
}
```

Code Table 6 – The sample codes for a DAO class *UserDAO*

This sample codes (see the *Code Table 6*) shows how to use the base class and provides some necessary methods to retrieve an existing User object, create a new User object, and update a User object. Changes to the object in question will be persisted to the database at the end of the transaction. But due to limitations on space, it won't give the whole details of the UserDAO class.

4.3.2 Google Translator

Because the Google Translation Service originally only supports JavaScript, developers have a need to access the AJAX Language API from the Non-JavaScript environments have to use the RESTful interfaces. Based on the provided interface, a sample command performs a Language Translation (/ajax/services/language/translate), for *Hello World* (q=hello%20world) from English to Italian (langpair=en%7Cit) is as following [10]:

```
http://ajax.googleapis.com/ajax/services/language/translate?v=1.0&q=hello%20world&lan
gpair=en%7Cit
```

Code Table 7 – The sample command performs a Google

The response is the JSON format and shown below [10]:

```
{"responseData": {
   "translatedText":"Ciao mondo"
 },
 "responseDetails": null, "responseStatus": 200}
```

Code Table 8 – The response data with the JSON format from the Google Translation Service

In this response, the content of the *responseText* is the translating result.

Therefore, based on RESTful interface, the Google Translator module should first establish a connection with the Google Translation Service.

```
String URL = "http://ajax.googleapis.com/ajax/services/language/translate?v=1.0&langpair=";
String TEXT = "&q=";
String fromCode = "en"; //"en" here is the code of English language
String toCode = "it"; //"it" here is the code of Italian language
StringBuffer urlBuffer = new StringBuffer();
urlBuffer.append(URL).append(fromCode).append("%7C").append(toCode).
        Append(TEXT).append(URLEncoder.encode(text, "UTF-8"));
URL url = new URL(urlBuffer.toString());
HttpURLConnection con = (HttpURLConnection) url.openConnection();
```

Code Table 9 – The codes for establishing a connection with the Google Translation Service

And after receiving the response, the Google Translator needs to parse the response data and extract the translating result.

```
String result = "";
JSONObject jsonObject = JSONObject.fromObject(buffer.toString());
//the result is the translating result.
Result = ((JSONObject) jsonObject.get("responseData")).getString("translatedText");
```

Code Table 10 – The codes for parsing the response data

4.4 STIS Web Service

According to the architecture design mentioned in the last chapter, the STIS Web Service should implement the following modules: Route Info Handler, STIS Consumer, XML Request Generator, and XML Response Parser. But for the last three modules, they have been implemented in the STIS by the class STISClient, RouteRequest, and RouteResponse, so when implementing the STIS Web Service, these three classes are quoted into this new system. Therefore, in this section, it will focus on the only one module – Route Info Handler.

Based on the system requirements, two web services need to be created in the Route

Info Handler module: getRefinementOptions and getRoute. Like mentioned in last section, the implementation about how to create these two web services won't be discussed in this section. Instead, it will explain two methods, *refineAddress(String address)* and *getRoute(List<String> message)*, that will be used to implement these two web services.

4.4.1 Method "refineAddress" Implementation

The method is aiming to refine an address to a list of intended (corrected) locations to resolve any ambiguity. The parameter of this method is the address with the *String* data-type, and the return value is a list of intended locations with the *List<String>* data-type. The implementation of this method is very simple, the developer just need to invoke the method *refinePreferences(String address)* provided by the class STISClient to get a XML response, and then use the method *parseXML(String xml)* to parse the receiving response and extract a list of results (see Code Table 11). The codes of method *parseXML* are also quoted from the STIS.

```
ArrayList<String> options;
String xmlResponse = stisClient.refinePreferences(address);
options = parseXML(xmlResponse);
return options;
```

Code Table 11 – The codes for implementing the method "refineAddress"

4.4.2 Method "getRoute" Implementation

This method is used to calculate a route by using user-defined preferred transport mode, start point, some waypoints, and a destination. Therefore, the parameter of this method will be a list of string in that the first string is used to represent the preferred transport mode such as "Bus"; the second string is the address of start point; the last string is the address of destination; and the others are the addresses of each waypoint. The return value of this method is a *RouteResponse* object.

When implementing this method, the developer first needs to encapsulate the parameter into two *Vector* objects:

```
String travelMode = null;
Vector waypoints = new Vector();
Vector waypointTypes = new Vector();
//message is the parameter
for (int I = 0; I < message.size(); i++) {
   if (I == 0) {
   //the first string in the parameter is the user preferred mode
      travelMode = message.get(i);
      continue;
   }
   String value = message.get(i);
   if (value.startsWith("Junction")) {
      String id = getTag(value, "(", ")");
      waypoints.add(id);
      waypointTypes.add(RouteRequest.JUNCTION_TYPE);
   } else if (value.startsWith("Luas-Stop")) {
      String id = getTag(value, "(", ")");
      waypoints.add(id);
      waypointTypes.add(RouteRequest.LUAS_TYPE);
   } else if (value.startsWith("Bus-Stop")) {
      String id = getTag(value, "(", ")");
      waypoints.add(id);
      waypointTypes.add(RouteRequest.BUS_TYPE);
   } else {
      String id = getTag(value, "(", ")");
      waypoints.add(id);
      waypointTypes.add(RouteRequest.JUNCTION_TYPE);
   }
}
```

Code Table 12 – The codes for encapsulating the user-defined preferred transport mode, start point, waypoints, and destination into two Vector objects.

And then, the two *Vector* objects and travelMode will be encapsulated into a *RouteRequest* object.

```
RouteRequest request = new RouteRequest(waypoints, waypointTypes, travelMode);
```

Code Table 13 – The codes for encapsulating the two Vector objects and preferred travel mode into a RouteRequest object.

Last, by using the method *findRoute* provided by the class STISClient, a RouteResponse object will be returned.

```
RouteResponse response = stisClient.findRoute(request);
```

Code Table 14 – The codes for calculating a route

This returned response is the result that includes all of the information of calculated route.

4.5 Prototype Mobile-Based Application

Based on the architecture design discussed in the last chapter, the Prototype Mobile-Based Application needs to implement seven modules: Review Info Handler, POI Info Handler, User Info Handler, Route Planner, XML Data Handler, Route Drawer, and Position Locator. For the first four modules, the approaches of implementation are similar. The developers only need to invoke relevant web services (based on the users' requests) and display receiving results on the screen. As the discussion in the section 4.2, most of works relating to web services invocation have been done by the SudzC, so in this section, it won't focus on the implementation of these four modules. Of course, because of the same reason, the XML Data Handler isn't the point of this section as well.

Therefore now, the implementations of the remaining two modules, Route Drawer and Position Locator, will be discussed in detailed in this section.

Note: all implementation approaches used in the Prototype Mobile-Based Application are only for the iOS platform, so it maybe can't work on the other mobile platform.

4.5.1 Route Drawer

So far, the iPhone SDK 4.0 hasn't provided an API that can be used to draw a route line on the map view. So, in this project, a methodology designed by Craig [11] will be used to implement the Route Drawer.

Because the iPhone SDK doesn't support drawing a line on top of the MKMapView (a map controller that is available in iOS 3 and later) directly, the first methodology used in this project is to create a custom drawn UIView and place it over the map. Make it identically sized to the MKMapView. Next, use the MKMapView member functions *convertCoordinate:toPointToView:* to determine View coordinates of the points. Last send these points to the custom drawn UIView and have it render those paths by using the drawRect method. Under the methodology, a route line can be drawn on a map view, but the map view can't respond users' zooming/panning actions.

In order to solve the zooming/panning problem, Craig's methodology is involved in this project. The trick here is instead of doing the drawing on the drawRect method of the customer UIView, using the drawRect method of a custom MKAnnotationView. By using this methodology, the Route Drawer can work properly.

Because there are too many codes for implementing this module, it won't list any codes in this report.

4.5.2 Position Locator

Like mentioned in the last chapter, the Position Locator should implement two functions: locate a position on the map and locate the user's current location.

The idea of implementing "*locate a position on the map*" function is quite simple. The first use the *hitTest:withEvent:* method of UIView to obtain a point (it is a point in a twodimensional coordinate system) where the user selects on a map view. And then send this point value (x and y) to the *convertPoint:toCoordinateFromView:* method of the MKMapView and convert it to a map coordinate (latitude and longitude).

```
//define a variable in the header file of a custom UIView
CGPoint aPoint;
```

```
//in the corresponding implementation file, obtain a point value from the method
//hitTest:withEvent: of the custom UIView
- (UIView *) hitTest: (CGPoint) point withEvent: (UIEvent *) event {
   aPoint = point;
}
```

```
//use the convertPoint:toCoordinateFromView: method to convert aPoint to the a map
//coordinate
CLLocationCoordinate2D aCoordinate = [mapView convertPoint: aPoint toCoordinateFromView:self];
double latitude = aCoordinate.latitude;
double longitude = aCoordinate.longitude;
```

Code Table 15 – The sample codes for locating a position on the map

In iPhone SDK, the *Core Location* framework (available in iOS 2.0 and later) can provide an ability to determine where in the world. Like mentioned in last chapter, there are three technologies that *Core Location* can leverage to do this: GPS, cell tower triangulation, and Wi-Fi Position Service (WPS) [7]. And the technologies that *Core Location* depends on are hidden from the application. The developers don't need to tell *Core Location* whether to use GPS, triangulation, or WPS. They just tell it how accurate they would like it to be, and it will decide from the technologies available to it which is best for fulfilling the project. The following steps will show how to use the Core Location to obtain a user's location.

1. Create an instance of the *CLLocationManager*, usually referred to as the Location Manager, which is the main class of the *Core Location*. And then in order to let the Location Manager start polling for a location, the developers have to assign a

delegate to the Location Manager. The Location Manager will call delegate methods when location information becomes available or changes.

```
//create an instance of the Location Manager

CLLocationManager *lm = [[CLLocationManager alloc] init];

//assign a delegate to the Location Manager

Lm.delegate = self;
```

Code Table 16 – Create an instance of the Location Manager

2. Set the desired accuracy. The accuracy is set using a *CLLocationAccuracy* value.

```
//kCLLocationAccuracyBest is used to tell Core Location to use the most accurate
//method that's currently available.

Lm.desiredAccuracy = kCLLocationAccuracyBest;
```

Code Table 17 – Set the desired accuracy

3. Start the Location Manager.

```
[lm startUpdateLocation];
```

Code Table 18 – start the Location Manager

4. Obtain the current location from the delegate method *locationManager:didUpdateToLocation:fromLocation:* of the Location Manager

```
- (void) locationManager: (CLLocationManager *) manager didUpdateToLocation:
(CLLocation *) newLocation fromLocation: (CLLocation *) oldLocation {

    CLLocationCoordinate2D coordinate = newLocation.coordinate;

    double latitude = aCoordinate.latitude;

    double longitude = aCoordinate.longitude;

}
```

Code Table 19 – Obtain the current location from the delegate method of the Location Manager

5. Evaluation

"Practice is the sole criterion of testing truth." – ***Karl Heinrich Marx***

5.1 Introduction

The main concerns of users about a mobile application are its functionality and performance. Therefore the evaluation scenarios for this project are proposed to evaluate from two aspects: functionality evaluation and system evaluation.

The aim of functionality evaluation is to evaluate whether the mobile application in this project can work properly under the different situation. In this evaluation scenario, the functionality of the mobile application will be evaluated in Internet environment and no-Internet environment, and the functionalities will be evaluated are:

- *User Management*. It includes new user creation, and user modification.
- *POI Management*. It includes POI information query, new POI creation, and POI modification.
- *Review Management*. It includes review retrieval for certain POI, write new reviews, and translate reviews.
- *Route Planning*. It includes planning a route with walking mode and a route with bus mode.

As this implementation of the Dublin Compass mobile system is a proof-of-concept, rather than an enterprise level software release, the non-functional performance of the system is important, but not critical. User experience and functional correctness were considered slightly higher priorities; however both are directly affected by the non-functional behaviour of the platform as a whole. The aim of the system evaluation is to evaluate the performance of the POI information system and STIS web services by means of latency. In this evaluation scenario, there will be different number of clients to access these two system at the same time, and then according to the corresponding response time to evaluate each system's scalability.

5.2 Scenario 2: System Evaluation

5.2.1 System Requirement

The platform of POI information system:

- Hardware: Dell XPS M1530.
    - Processor: Intel 2.2GHz Core 2 Duo T7500.
    - Memory: 2GB DDR2-667 SDRAM.
    - Video Card: NVIDIA GeForce 8600M GT with 256MB GDDR3.
    - Hard Disk: 500GB 5400RPM SATA HDD.
- Software:
    - Operating System: Microsoft Windows 7 Ultimate Version 32bit.
    - Development IDE: Netbeans IDE 6.9.1.
    - Application Server: GlassFish Server Open Source Edition 3.0.1.

The platform of STIS:

- Hardware: Dell XPS M1530.
    - Processor: Intel 2.2GHz Core 2 Duo T7500.
    - Memory: 2GB DDR2-667 SDRAM.
    - Video Card: NVIDIA GeForce 8600M GT with 256MB GDDR3.
    - Hard Disk: 500GB 5400RPM SATA HDD.
- Software:
    - Operating System: Microsoft Windows 7 Ultimate Version 32bit.
    - Development IDE: Eclipse Europa 3.3.0.

    - Application Server: Apache Tomcat 6.0.26.

The platform of STIS web service:

- Hardware: Dell XPS M1530.
    - Processor: Intel 2.2GHz Core 2 Duo T7500.
    - Memory: 2GB DDR2-667 SDRAM.
    - Video Card: NVIDIA GeForce 8600M GT with 256MB GDDR3.
    - Hard Disk: 500GB 5400RPM SATA HDD.
- Software:
    - Operating System: Microsoft Windows 7 Ultimate Version 32bit.
    - Development IDE: Netbeans IDE 6.9.1.
    - Application Server: GlassFish Server Open Source Edition 3.0.1.

The platform of evaluator:

- Hardware: Desktop PC.
    - Processor: Intel 2.33GHz Core2 Q8200.
    - Memory: 4GB DDR2-800 SDRAM.
    - Video Card: NVIDIA GeForce GTX 260 with 896MB GDDR5.
    - Hard Disk: 320GB 7200RPM SATA HDD.
- Software:
    - Operating System: Microsoft Windows 7 Ultimate Version 64bit.
    - Development IDE: Netbeans IDE 6.9.1.

Internet environment:

- Fiber Power Broadband from UPC: 30Mb/3Mb.

5.2.2 Evaluator

In order to evaluate the latency, an evaluator is designed in this project. The principle of this evaluator is very simple that creates a number of threads that will access an evaluated system at the same time, and then each thread will print out corresponding response time (latency).

The following codes are used to implement accessing an evaluated system and printing out its response time.

```
//codes are written in Java
//start timestamp
long startTime = System.currentTimeMillis();
//the method "getNearbyPoi" is a sample service provided by POI information
//system
getNearbyPoi(53.348925,-6.25989, 20000);
//end timestamp
long endTime = System.currentTimeMillis();
System.out.println(endTime - startTime);
```

Code Table 20 – The codes of Evaluator that are used to implement accessing an evaluated system and printing out its response time

The following codes are used to implement creating a number of threads and starting them.

```
//codes are written in Java
//variable "count" is the number of threads will be created
for(int I = 0; i<count; i++){
   //"PoiInfoEvaluatorHandler" is a thread class that includes above codes
   Thread t = new Thread(new PoiInfoEvaluatorHandler());
   //start thread
    t.start();
}
```

Code Table 21 – The codes of Evaluator that are used to implement creating a number of threads and starting them

5.2.3 Evaluation Outcome and Summary

Measurements of the response time, using different number of accessing clients, demonstrate the response time to be proportional to the number of clients (see *Figure 26*).

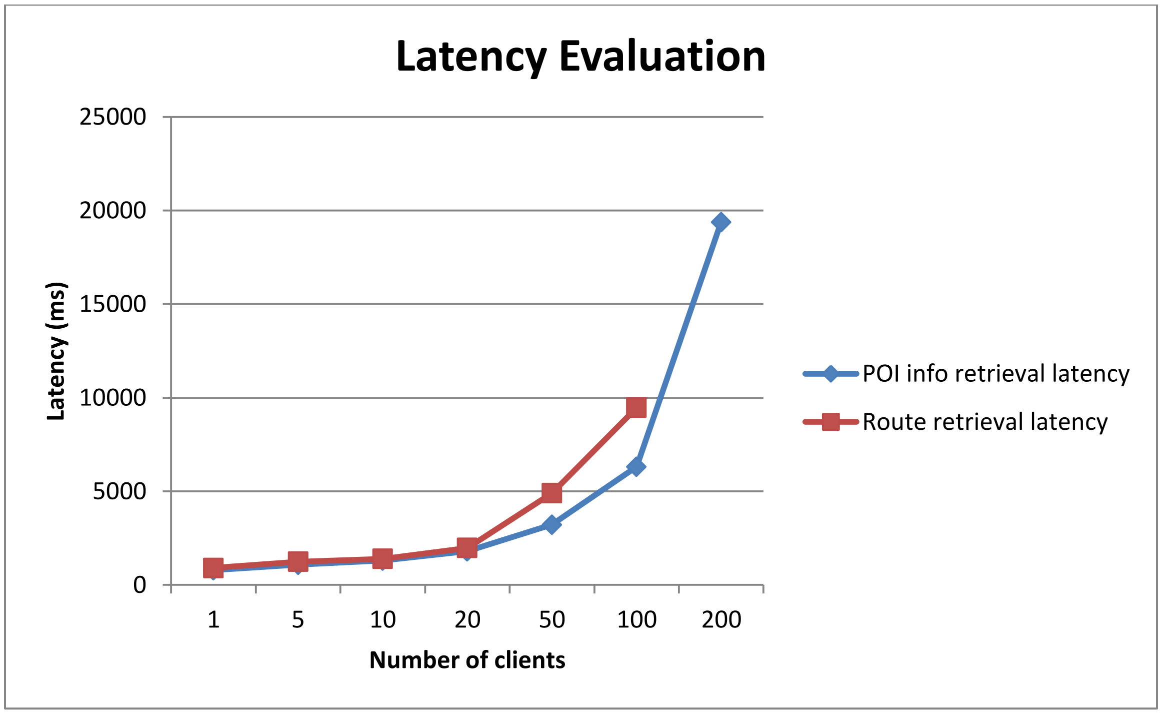


Figure 26 – Comparing response latency of POI info retrieval and route retrieval with number of accessing clients

According to Akamai and JupiterResearch identified, currently 4 seconds is the threshold of acceptability for web application response time [12], so the acceptable max number of accessing clients of POI information system is 50 to 100, and 20 to 50 for STIS web service. The reason why the STIS web service wasn't measured by 200 clients is because when 100 clients were accessing it simultaneously, the STIS (not STIS web service) throws an error at times.

The evaluation data are as following (each response time is an average value):

	1 clients	5 clients	10 clients	20 clients	50 clients	100 clients	200 clients
POI info system	786.16	1084	1302	1796	3216	6313	19382
STIS web service	907	1246	1396	1975	4905	9470	nil

Table 10 – The data for latency evaluation

5.3 Scenario 1: Functionality Evaluation

Because most of functions provided by the mobile-based application in this project can only work in the Internet environment, a no-Internet environment evaluation will be needed that is used to measure whether this application can work properly (provide appropriate alert and no crash) with no Internet connection.

5.3.1 Internet Environment

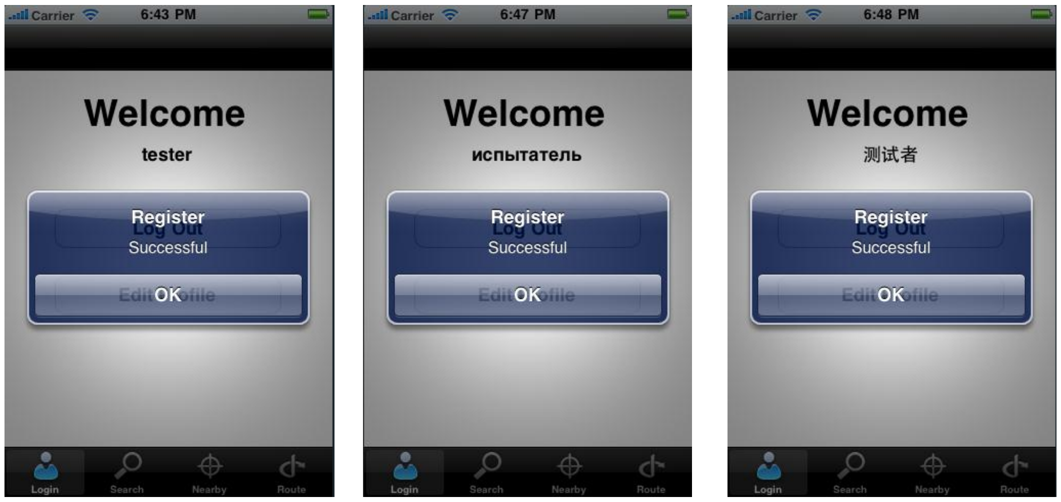
5.3.1.1 User Management Evaluation

The input data for ***evaluating new user registration*** are as following:

Task	Username	Password	Retype Password	Default Language
1	tester	1234	1234	[select in a table]
2	испытатель	1234	1234	[select in a table]
3	测试者	1234	1234	[select in a table]
4	tester	1234	1234	[select in a table]
5	tester1	1234	123	[select in a table]

Table 11 – The data for evaluating new user registration

For task 1, 2 and 3, the evaluating outcomes are same as the expectance.

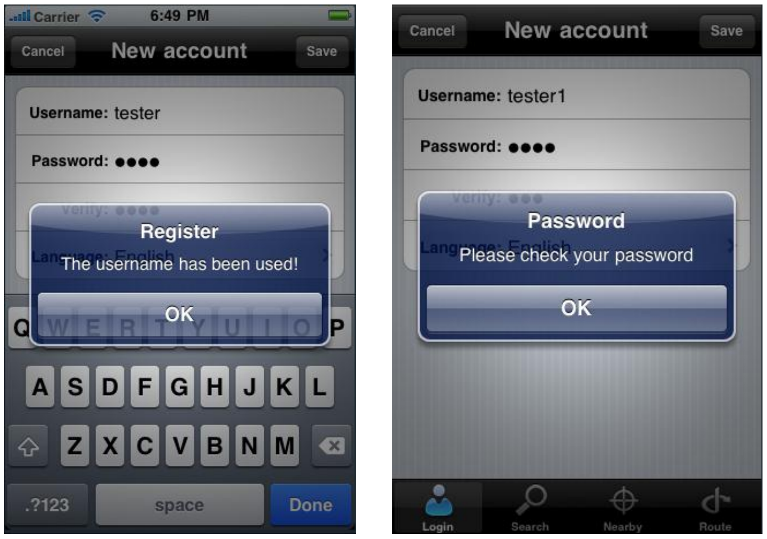


Register
Successful

Figure 27 – The outcomes of evaluating new user registration (1). The left image is the outcome of task 1, the middle image is for task 2, and the right image for task 3.

The images at above figure show that the POI information system supports multi-language as the username.

For task 4, the application shows an alert to notify users this username has been used (see the left image at below figure). And in task 5, an alert is shown to notify users the entered password is incorrect (see the right image at below figure).



Username: tester
Password: ●●●●
Register
The username has been used!

Username: tester1
Password: ●●●●
Password
Please check your password

Figure 28 – The outcomes of evaluating new user registration (2). The left image is the outcome of task 4, and the right image is for task 5.

The input data for ***evaluating user modification*** are as following:

Task	New Password	Retype new Password	Old Password	Default Language
1	[don't change]	[don't change]	1234 [correct]	Italiano
2	4321	4321	1234 [correct]	Italiano
3	1234	123	4321	Italiano
4	1234	1234	432 [incorrect]	Italiano

Table 12 – The data for evaluating user modification

For task 1 and 2, the evaluating outcomes are same as the expectance.

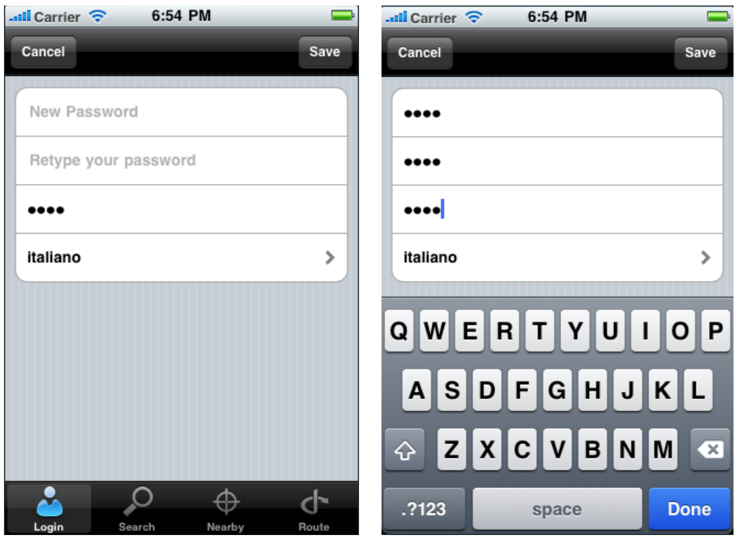
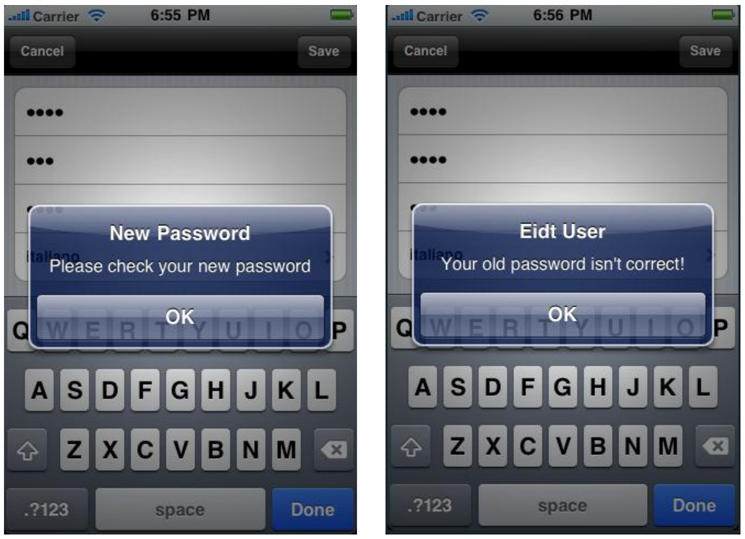


Figure 29 – The outcomes of evaluating user modification (1). The left image is the outcome of task 1, and the right image is for task 2.

For task 3, the application displays an alert that notifies users to check new password. And in task 4, an alert appears with a message "Your old password isn't correct".


New Password

Please check your new password

Eidt User

Your old password isn't correct!

Figure 30 – The outcomes of evaluating user modification (2). The left image is the outcome of task 3, and the right one is for task 4.

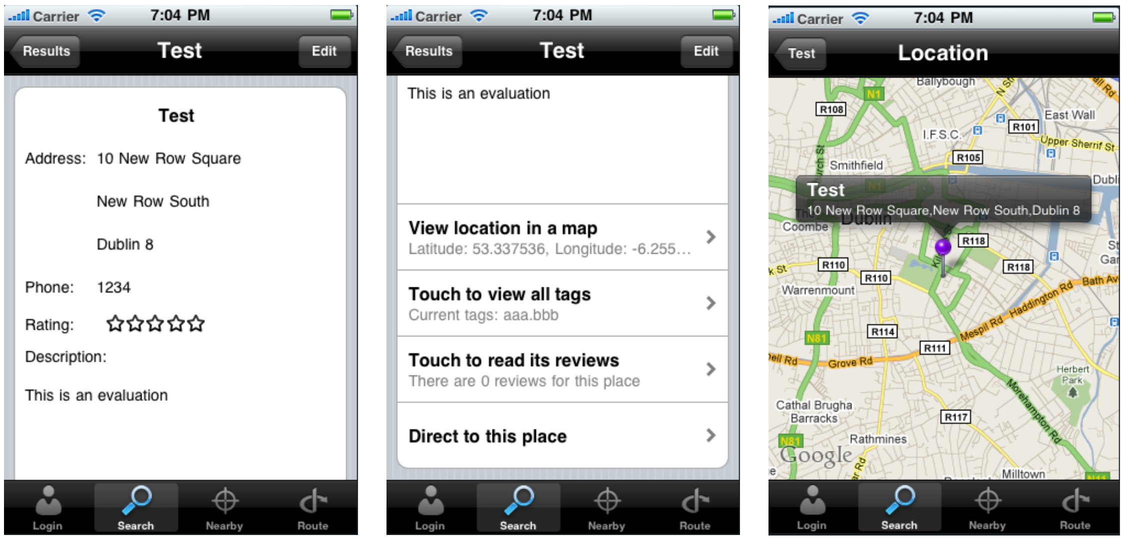
5.3.1.2 POI Management Evaluation.

The input data for ***evaluating POI creation*** are as following:

Task	Place Name	Address	Phone	Description	Location	Tags
1	Test	10 New Row Square, New Row South, Dublin 8	1234	This is an evaluation	[Select a location on the map]	aaa,bbb
2	Test 1	[No enter]	1234	This is an evaluation	[Select a location on the map]	[No enter]
3	Test 2	10 New Row Square, New Row South, Dublin 8	1234	[No enter]	[No enter]	aaa

Table 13 – The data for evaluating POI creation

For task 1, the evaluating outcome is same as the expectance, see below figure.


Carrier 7:04 PM

Results **Test** Edit

Test

Address: 10 New Row Square

New Row South

Dublin 8

Phone: 1234

Rating: ☆☆☆☆☆

Description:

This is an evaluation

Carrier 7:04 PM

Results **Test** Edit

This is an evaluation

View location in a map
Latitude: 53.337536, Longitude: -6.255…

Touch to view all tags
Current tags: aaa.bbb

Touch to read its reviews
There are 0 reviews for this place

Direct to this place

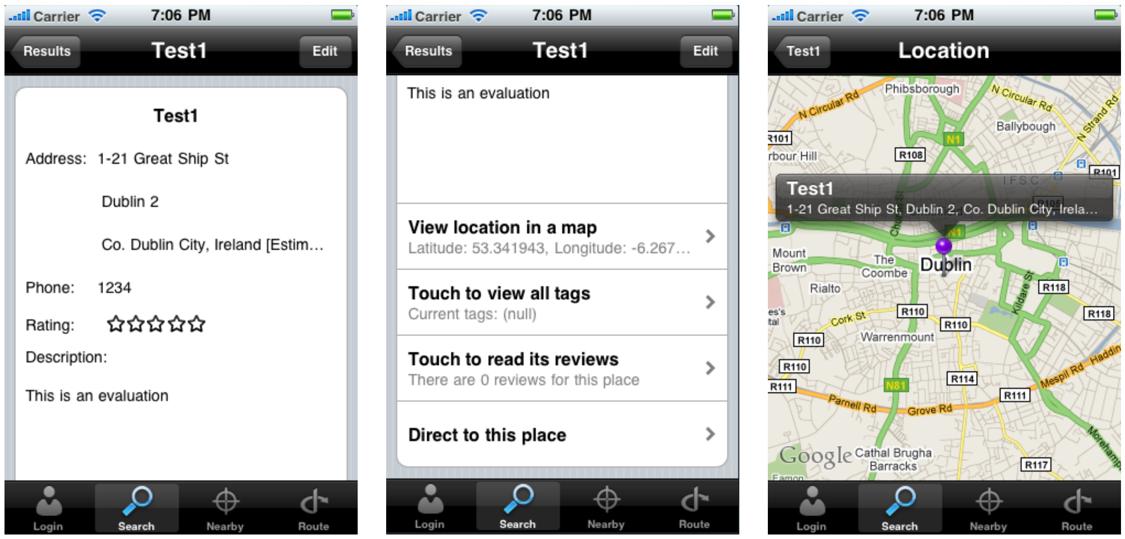
Carrier 7:04 PM

Test **Location**

Test
10 New Row Square,New Row South,Dublin 8

Figure 31 – The outcome of evaluating POI creation (1). All images are the outcome of task 1.

For task 2, the evaluating outcome is also correct. The POI information system not only saves this new POI into database, but also reverses the selected location (the location is described by latitude and longitude) to its corresponding address.


Carrier 7:06 PM

Results **Test1** Edit

Test1

Address: 1-21 Great Ship St

Dublin 2

Co. Dublin City, Ireland [Estim…

Phone: 1234

Rating: ☆☆☆☆☆

Description:

This is an evaluation

Carrier 7:06 PM

Results **Test1** Edit

This is an evaluation

View location in a map
Latitude: 53.341943, Longitude: -6.267…

Touch to view all tags
Current tags: (null)

Touch to read its reviews
There are 0 reviews for this place

Direct to this place

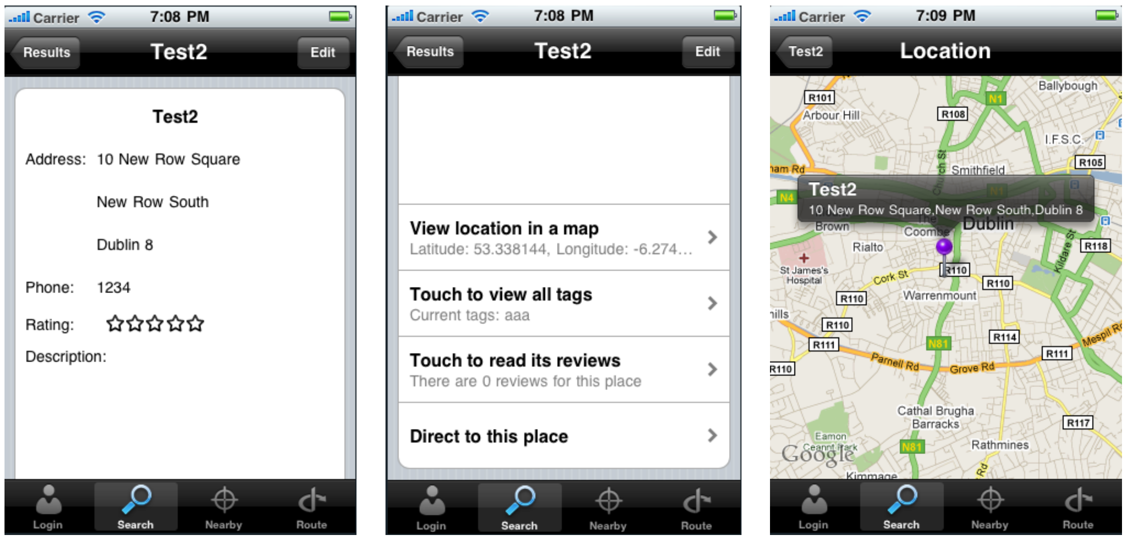
Carrier 7:06 PM

Test1 **Location**

Test1
1-21 Great Ship St, Dublin 2, Co. Dublin City, Irela…

Figure 32 – The outcome of evaluating POI creation (2). All images are the outcome of task 2.

For task 3, the evaluating outcome is correct. The POI information system also converts the entered address to its corresponding latitude and longitude.


Carrier 7:08 PM

Results Test2 Edit

Test2

Address: 10 New Row Square

New Row South

Dublin 8

Phone: 1234

Rating: ☆☆☆☆☆

Description:

Login Search Nearby Route

Carrier 7:08 PM

Results Test2 Edit

View location in a map
Latitude: 53.338144, Longitude: -6.274…

Touch to view all tags
Current tags: aaa

Touch to read its reviews
There are 0 reviews for this place

Direct to this place

Login Search Nearby Route

Carrier 7:09 PM

Test2 Location

Test2
10 New Row Square,New Row South,Dublin 8

Login Search Nearby Route

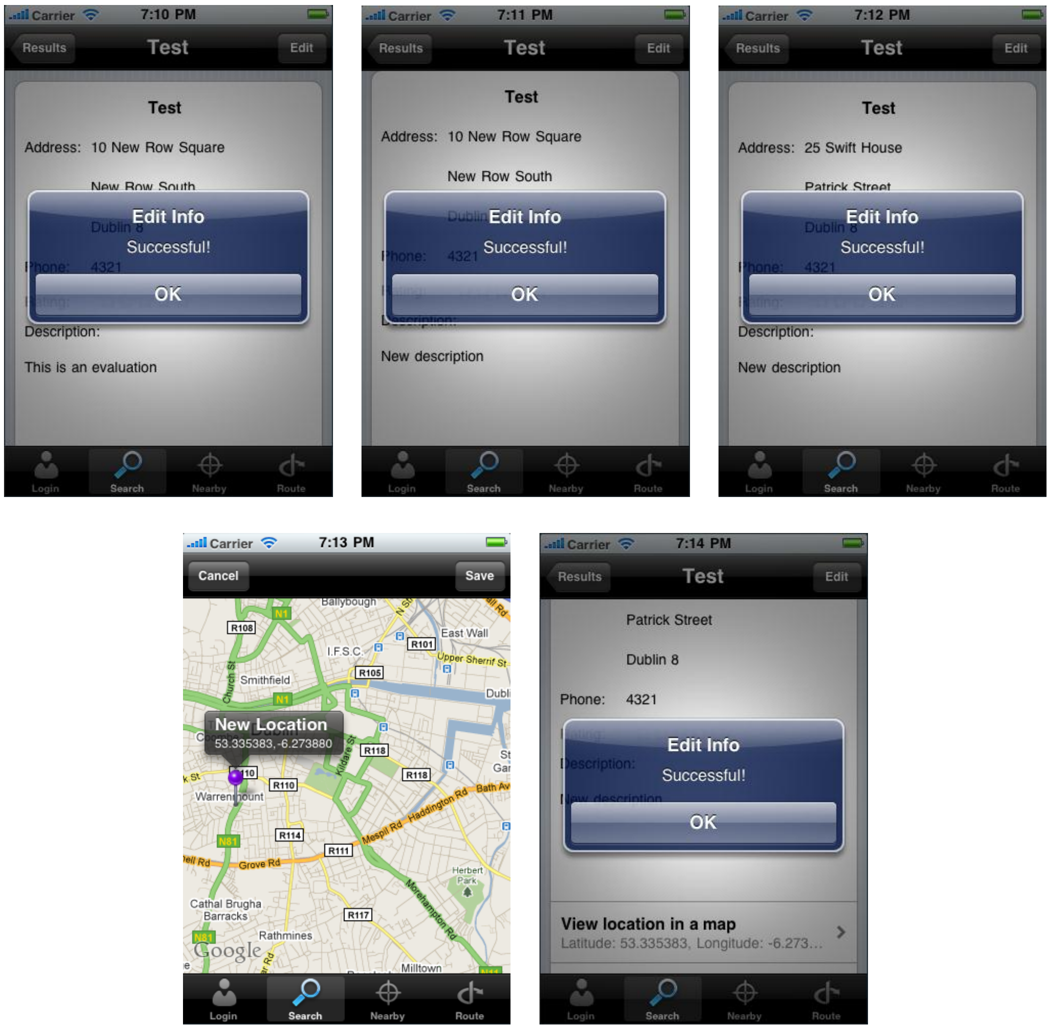
Figure 33 – The outcome of evaluating POI creation (3). All images are the outcome of task 3.

The input data for ***evaluating POI modification*** are as following:

Task	Address	Phone	Description	Location
1	[don’t change]	4321	[don’t change]	[don’t change]
2	[don’t change]	[don’t change]	New description	[don’t change]
3	25 Swift House, Patrick Street ,Dublin 8	[don’t change]	[don’t change]	[don’t change]
4	[don’t change]	[don’t change]	[don’t change]	[select a new location on map]

Table 14 – The data for evaluating POI modification

For task 1, 2, 3, and 4, the evaluating outcomes are same as the expectance.



Test

Address: 10 New Row Square

New Row South

Dublin 8

Phone: 4321

Description:

This is an evaluation

Edit Info

Successful!

OK

Test

Address: 10 New Row Square

New Row South

Phone: 4321

Description:

New description

Edit Info

Successful!

OK

Test

Address: 25 Swift House

Patrick Street

Dublin 8

Phone: 4321

Description:

New description

Edit Info

Successful!

OK

Patrick Street

Dublin 8

Phone: 4321

Description:

New description

Edit Info

Successful!

OK

View location in a map

Latitude: 53.335383, Longitude: -6.273…

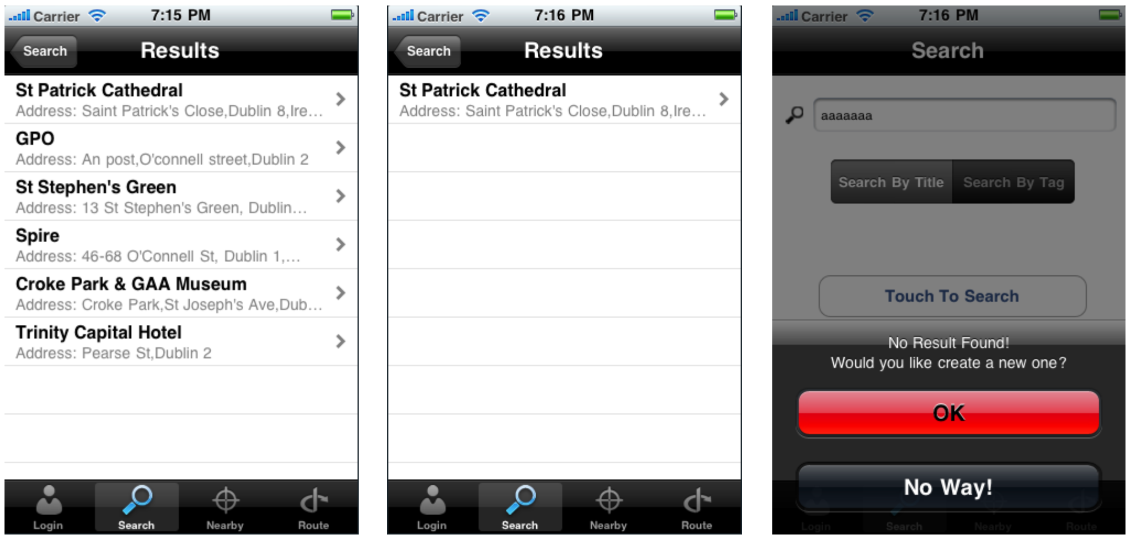
Figure 34 – the outcomes of evaluating POI modification. The top-left image is the outcome of task 1, the top-middle image is for task 2, and the top-right image is for task 3. The two bottom images are the outcome of task 4. The bottom-left image shows the new location selection, and the bottom-right image is the result.

The input data for ***evaluating POI retrieval by place name*** are as following:

Task	**Keywords**
1	P [search all the POIs whose name includes "p"]
2	St Patrick Cathedral [exact place name]
3	aaaaaaa [no POI with this name]

Table 15 – The data for evaluating POI retrieval by place name

In task 1 and 2, the application lists all desirable POIs. And in task 3, a notification appears to ask users whether create a new POI with the name “aaaaaaa”.



St Patrick Cathedral
Address: Saint Patrick's Close,Dublin 8,Ire…

GPO
Address: An post,O'connell street,Dublin 2

St Stephen's Green
Address: 13 St Stephen's Green, Dublin…

Spire
Address: 46-68 O'Connell St, Dublin 1,…

Croke Park & GAA Museum
Address: Croke Park,St Joseph's Ave,Dub…

Trinity Capital Hotel
Address: Pearse St,Dublin 2

St Patrick Cathedral
Address: Saint Patrick's Close,Dublin 8,Ire…

aaaaaaa

No Result Found!
Would you like create a new one?

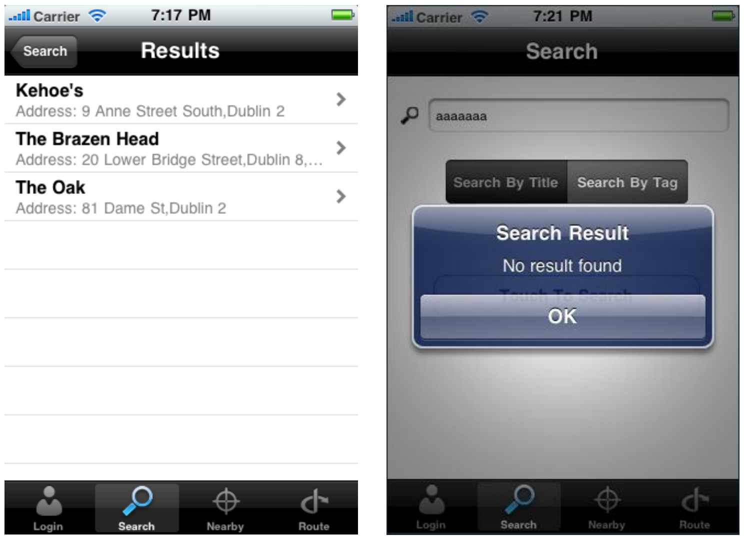
Figure 35 – The outcomes of evaluating POI retrieval by place name. The left image shows the outcome of task 1, the middle image is for task 2, and the right image is for task 3.

The input data for ***evaluating POI retrieval by tag*** are as following:

Task	Keywords
1	pub [search all the POIs that are associated with the tag “pub”]
2	aaaaaaa [no this tag]

Table 16 – The data for evaluating POI retrieval by tag

The result of task 1 is to list several desirable POIs in a table, but for task 2, an alert appears to notify users “No result found”.



Kehoe's
Address: 9 Anne Street South,Dublin 2

The Brazen Head
Address: 20 Lower Bridge Street,Dublin 8,...

The Oak
Address: 81 Dame St,Dublin 2

aaaaaaa

Search Result
No result found

Figure 36 – The outcomes of evaluating POI retrieval by tag. The left image shows the outcome of task 1, and the right one is for task 2.

The input data for ***evaluating nearby POI retrieval*** are as following:

Task	Searching Radius	Center Location
1	2 km	[locate user's current location]
2	0.5	[select a location on map]
3	2 km	[select the same location with task 2]
4	5 km	[select the same location with task 2]

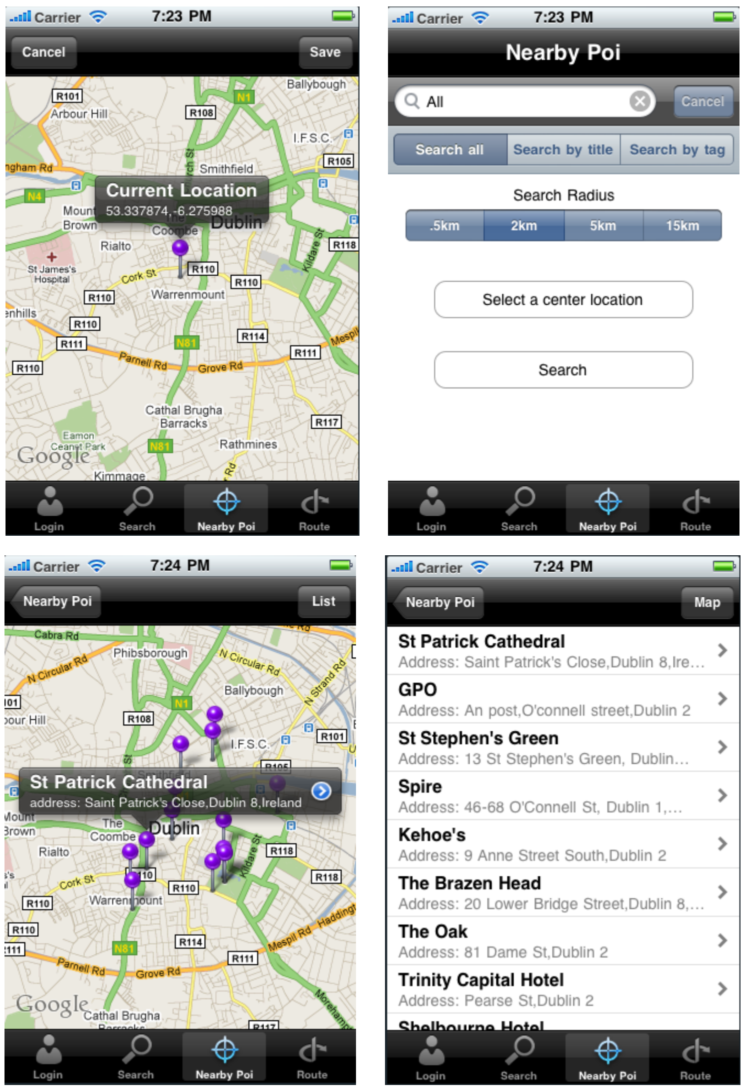
Table 17 – The data for evaluating nearby POI retrieval

In the task 1, within 2km, all eligible POIs around a user are annotated on a map, and also listed in a table.

The top-left image at below figure shows a user's current position. This position is located by the application automatically.

The image on the top-right shows the selected search radius is 2km.

The lower two images illustrate the searching results that are represented by means of map and table.

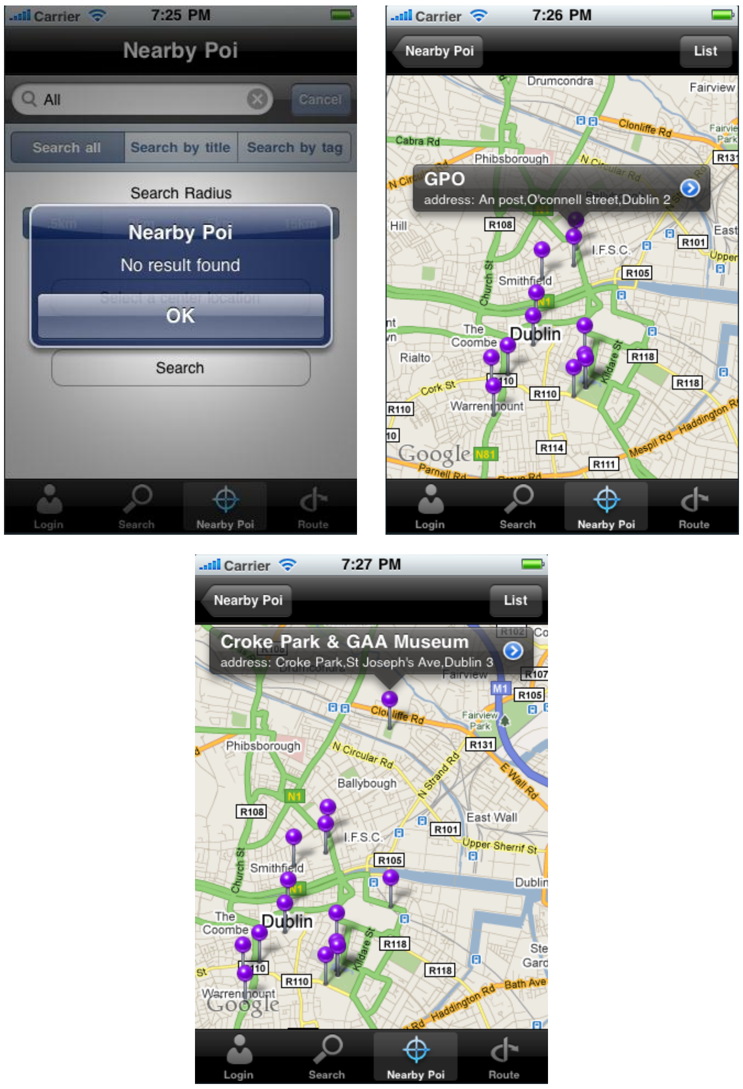


St Patrick Cathedral
Address: Saint Patrick's Close,Dublin 8,Ire...
GPO
Address: An post,O'connell street,Dublin 2
St Stephen's Green
Address: 13 St Stephen's Green, Dublin...
Spire
Address: 46-68 O'Connell St, Dublin 1,...
Kehoe's
Address: 9 Anne Street South,Dublin 2
The Brazen Head
Address: 20 Lower Bridge Street,Dublin 8,...
The Oak
Address: 81 Dame St,Dublin 2
Trinity Capital Hotel
Address: Pearse St,Dublin 2
Shelbourne Hotel

Figure 37 – The outcome of evaluating nearby POI retrieval (1). All images shows the outcome of task 1.

For the task 2, 3, and 4, all eligible POIs are found.

Because the searching radius in task 2 is only 0.5km, so there is no result found in this case (see the image at the top-left of below figure). And when the radius increases to 2km in task 3, several POIs are searched and displayed on the map (see the image on the top right of below figure). In the last task, the radius increases again to 5km, two more POIs are found (see the lower image in the below figure).



Nearby Poi
No result found
OK

GPO
address: An post,O'connell street,Dublin 2

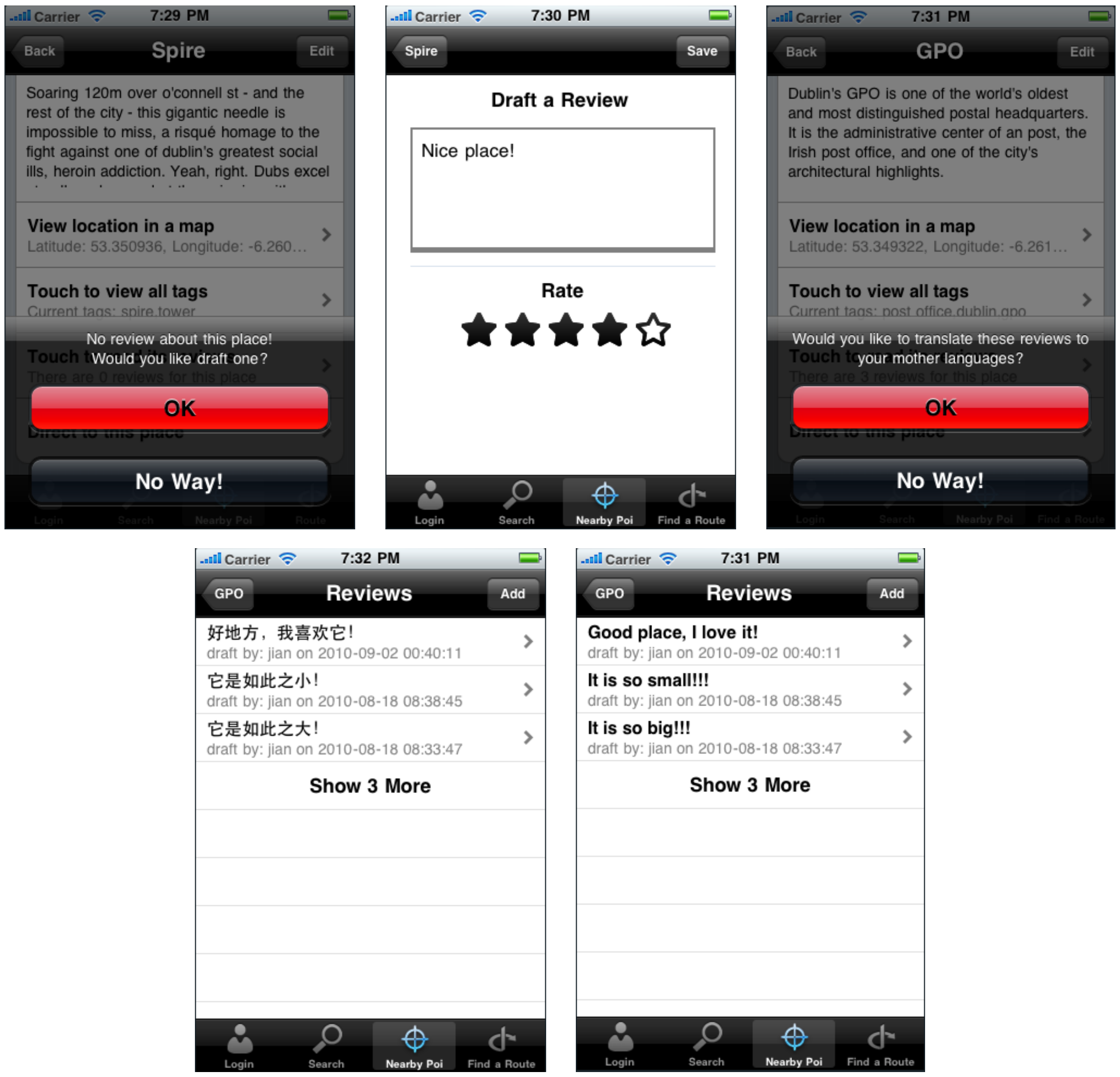
Croke Park & GAA Museum
address: Croke Park,St Joseph's Ave,Dublin 3

Figure 38 – The outcomes of evaluating nearby POI retrieval (2). The image on the top left shows the outcome of task 2, the image on the top right is for task 3, and the lower one is for task 4.

5.3.1.3 Review Management Evaluation.

When a user touches to read POI's reviews, if there is no review for this POI, a notification will be displayed to ask this user whether creates a review (see the image on the top-right of below figure). And if the user touches "OK", then a new view for drafting a review will appear (see the top-middle image).

But if there are some reviews for this POI, a notification will be also displayed to ask this user whether translate these reviews to his default language (see the top-right image). If the user touches "OK", all of the related reviews are translated and displayed in a table (see the lower-left image). Otherwise, the application will just display the original reviews (see the lower-right image).



Soaring 120m over o'connell st - and the rest of the city - this gigantic needle is impossible to miss, a risqué homage to the fight against one of dublin's greatest social ills, heroin addiction. Yeah, right. Dubs excel

View location in a map
Latitude: 53.350936, Longitude: -6.260…

Touch to view all tags
Current tags: spire,tower

No review about this place!
Would you like draft one?

There are 0 reviews for this place

Draft a Review

Nice place!

Rate

Dublin's GPO is one of the world's oldest and most distinguished postal headquarters. It is the administrative center of an post, the Irish post office, and one of the city's architectural highlights.

View location in a map
Latitude: 53.349322, Longitude: -6.261…

Touch to view all tags
Current tags: post office,dublin,gpo

Would you like to translate these reviews to your mother languages?

There are 3 reviews for this place

好地方，我喜欢它！
draft by: jian on 2010-09-02 00:40:11

它是如此之小！
draft by: jian on 2010-08-18 08:38:45

它是如此之大！
draft by: jian on 2010-08-18 08:33:47

Show 3 More

Good place, I love it!
draft by: jian on 2010-09-02 00:40:11

It is so small!!!
draft by: jian on 2010-08-18 08:38:45

It is so big!!!
draft by: jian on 2010-08-18 08:33:47

Show 3 More

Figure 39 – The outcomes of evaluating review Management

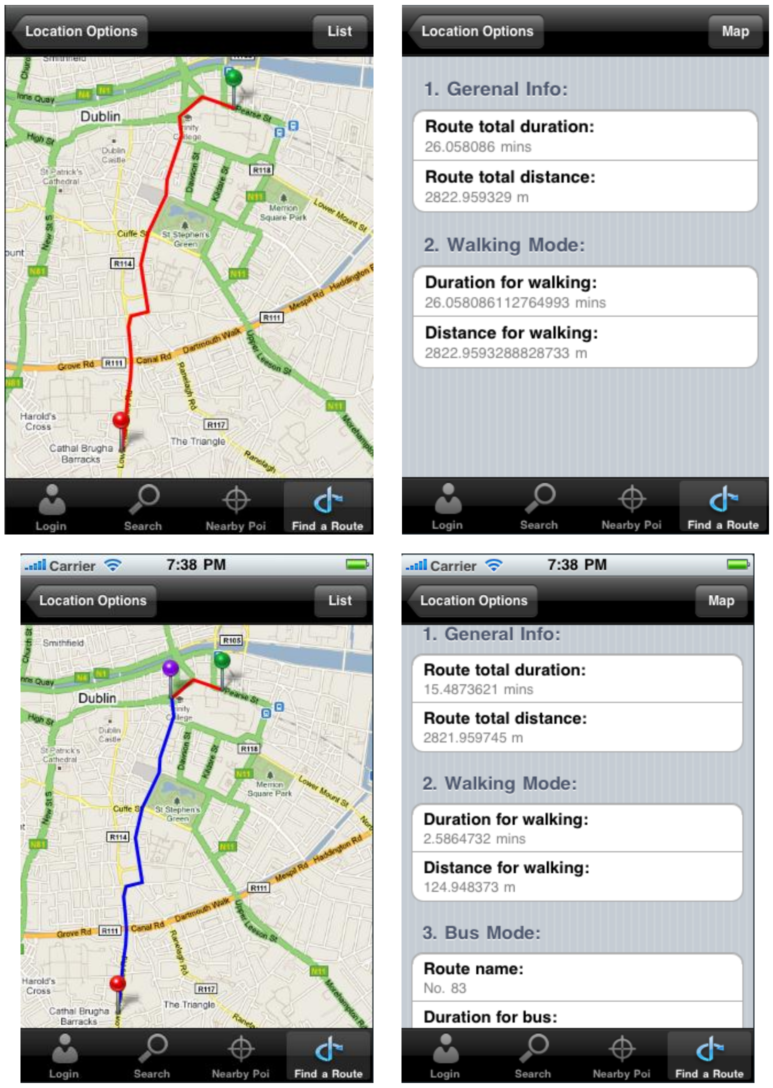
5.3.1.4 Route planning Evaluation.

The input data for ***evaluating the route planning under different transport modes*** are as following:

Task	Start	Destination	Transport Mode
1	Pearse Street	Rathmines Road	Walking
2	Pearse Street	Rathmines Road	Walking/Bus

Table 18 – The data for evaluating the route planning under different transport modes

The evaluating results of each task are same as the expectance.



1. Gerenal Info:

Route total duration:
26.058086 mins

Route total distance:
2822.959329 m

2. Walking Mode:

Duration for walking:
26.058086112764993 mins

Distance for walking:
2822.9593288828733 m

1. General Info:

Route total duration:
15.4873621 mins

Route total distance:
2821.959745 m

2. Walking Mode:

Duration for walking:
2.5864732 mins

Distance for walking:
124.948373 m

3. Bus Mode:

Route name:
No. 83

Duration for bus:

Figure 40 – The outcomes of evaluating the route planning under different transport modes. The top two images show the outcome of task 1, and the lower two images are for task 2.

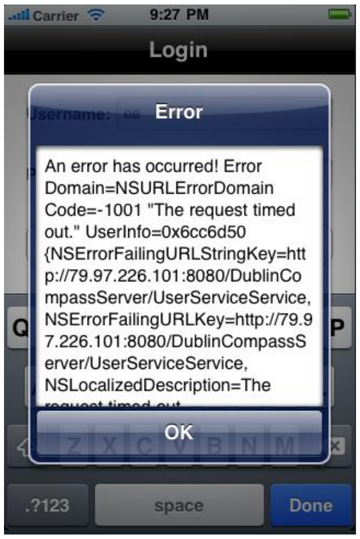
The two images at the top of above figure show the STIS-generated travel plan for a walking mode. This route directs the walker from the starting point, the green pin, to the destination point at the bottom, the red pin. A single mode of transport (walking) is used throughout the whole of this journey.

The two images at the bottom illustrate a bus-based route between the same start and destination points. Specifically, this image shows a multi-modal walker-bus route. The red line on the map represents the walking route, and the blue line is the bus route.

5.3.2 No Internet Environment

In a no-Internet environment, all of the functions based on the online service will be in trouble, so a thorough mobile-based application should notify users that the invoked function can't work without Internet connection. Because the mobile application in this project is just a prototype, it won't display an alert with message "Can't work without Internet connection". But instead, it will give an error list for developer.

The following figure shows the result when an Internet-based function is invoked in a no-Internet environment.



An error has occurred! Error Domain=NSURLErrorDomain Code=-1001 "The request timed out." UserInfo=0x6cc6d50 {NSErrorFailingURLStringKey=http://79.97.226.101:8080/DublinCompassServer/UserServiceService, NSErrorFailingURLKey=http://79.97.226.101:8080/DublinCompassServer/UserServiceService, NSLocalizedDescription=The

Figure 41 – The outcome of evaluating an Internet-based function which is invoked in a no-Internet environment.

6. Conclusion

"Rome was not built in a day" – ***Proverb***

The goals outlined at the start of the project were largely accomplished, and based on the architecture design, some uninvolved online services such as weather service or ticket/hotel booking service can quite easily be involved to produce better and richer travelling services. But due to the time is limited, there are also some works left for future.

The first work should be done in the future is to reduce the size of transfer data. This is because the users of this system can't always stay in a free Wi-Fi environment, or, at least, it is impossible for now. So sometimes, the users have to use the travelling services under some charged network (e.g. 3G). So, the work of reducing the size of transfer data to optimize the flow rate is quite significant.

Next thing should be considered in the future is to improve the security of the system. It will be done from the following aspects:

- Improve the reliability of the password. Because the password is selected by users, sometimes it is not secure enough. For example, some passwords maybe include users' name or birthday. Therefore, when creating a new password, the system needs to check whether this password is "safe" enough. The standard of a secure password will be used in the system is that it can't include users' name and birthday, and must have uppercases, lowercases, numbers, and special symbols (e.g."%" or"*").

- Encrypt the sensitive data. To protect sensitive data in the database, the system will encrypt the data before storing. The algorithm of encryption is proposed to be used is Advanced Encryption Standard (AES). Moreover, when transmitting sensitive data, in order to prevent interception, the system will use Transport Layer Security (TLS) protocol to secure the transfer data.

- Contents restriction. In order to implement the contents restriction, the system will design a component (e.g. filter) that is able to check literal content by using some system-defined keywords. Once finding some information provided by users that includes these keywords, these information will be removed to a “pending area” (e.g. a database table) and won’t be retrieved by the other users directly. Then, the administrators will recheck this pending information to determine whether it should be displayed or not. Moreover, this system will also allow users to report the invalid contents (photos or literal contents). And then these “invalid” contents will be rechecked again by the administrators. Last, if a user who provides invalid contents more than three times, he will be added into a blacklist (a database table).

Reference

[1] S. Brennan and R. Meier, "*STIS: Smart Travel Planning Across Multiple Modes of Transportation*", 2007.

[2] *TomTom UK & Ireland.* [cited March 2010]; Available from: http://www.tomtom.com/products/features.php?ID=940&Lid=1&Category=2

[3] *AA Router Planner.* [cited March 2010]; Available from: http://www.theaa.com/iphone/route-planner-iphone-app.html

[4] *Dublin Buster*. [cited March 2010]; Available from: http://www.shiyunhe.com/

[5] *London Bus*. [cited March 2010]; Available from: http://mbarclay.net/?page_id=193

[6] *Yelp*. [cited March 2010]; Available from: http://www.yelp.com/yelpmobile

[7] Dave Mark & Jeff LaMarche, "*Beginning iPhone 3 Development*", 2009.

[8] *Wsdl2objc*. [cited August 2010]; Available from: http://code.google.com/p/wsdl2objc/

[9] *SudzC*. [cited August 2010]; Available from: http://sudzc.com/

[10] *Google AJAX Language API – Developer Guide*. [cited August 2010]; Available from: http://code.google.com/intl/en/apis/ajaxlanguage/documentation/#fonje

[11] *Craig's Methodology for Drawing Route Line on Map View*. [cited August 2010]; Available from: http://spitzkoff.com/craig/?p=81

[12] Jupiter Research. "*Retail Web Site Performance: Consumer Reaction to a Poor Online Shopping Experience*", June 2006. Survey commissioned by Akamai Technologies.