Tour Navigation: A Cloud Based Tourist Navigation System

Junhao Li

A thesis submitted to

Auckland University of Technology

in partial fulfilment of the requirements for the degree

of

Master of Computer and Information Sciences (MCIS)

2014

School of Computer and Mathematical Sciences

Abstract

Google and Apple have provided precise navigation and locating service in outdoor environment. Tourists in New Zealand are able to identify their current location on a map and use navigation services provided by smartphones with Global Positioning Systems (GPS). However Google Map and IOS Map cannot provide navigation and locating service in indoor environments as the GPS signal is blocked within building interior. This thesis proposes a system to provide convenient and efficient tour navigation service to smartphone users no matter they are indoors or outdoors. In order to provide this service, this thesis implements back-end server on the Cloud Computing platform Google App Engine to calculate the navigation path for mobile users when indoors. Elastic cloud computing platform makes the system efficient at handling considerable number of requests while maintenance costs of the system are affordable. In addition, the mobile application Tour Navigation is developed as the interface between users and cloud service. By the clever use of QR codes with Tour Navigation, users are able to know their current location and query the shortest path to destination by sending requests to the cloud server.

This thesis presents the NZ Tour Navigation System which is an affordable and easily-implemented technical solution for indoor navigation. What is more, it proves that the cloud computing platform is able to be the robust server for mobile applications, capable of handling complicated computing tasks.

Contents

List of Figures

List of Tables

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Signature:_____ Date:_____

Acknowledgements

I wish to thank my parents for their love and support. Members of the School of Computer and Mathematical Sciences at AUT University have been significantly helpful in bringing this work to completion. I appreciate all teachers, supervisors and administration staff especially. My supervisors Shoba Tegginmath and Dr. Weiqi Yan have put in a great deal of effort into guiding and supporting me with dedication..

Junhao Li

Auckland

June 2014

Chapter 1 Introduction

In this thesis, a mobile application called Tour Navigation, supported by cloud computing, is designed and implemented for the purpose of guiding tourists in both the indoor and outdoor environments. First of all, this chapter considers the factors that influence New Zealand tourism and goes on to discuss how smartphones could boost New Zealand tourism. Several temporal technical solutions of mobile tour guides and mobile applications are reviewed. After providing a critical analysis of why cloud computing platform is chosen as data server over traditional servers, research questions of this thesis are given. The chapter concludes by detailing the structure of this thesis.

1.1 Background

1.1.1 Travelling in New Zealand

New Zealand still preserves its stunning mountains, rivers forests, beaches and other original landscapes spaces (Alessio, 2010). The fabulous rural landscape appearing on the printed guidebooks explains why a significant number of tourists consider this Oceania country on the top of the list of tour destinations. In addition, buildings have been constructed constantly since the nineteenth century in order to attract enough permanent migrants from Common Wealth countries and short-term visitors from the Anglo American world. Thus, New Zealand's urban centers also have captured national and international visitors' attentions (Alessio, 2010).

However, the tourism market of New Zealand is highly affected by financial factors. Since New Zealand is located far away from major continents, tourists have to pay significant amounts of money on transportation and accommodation. High expense has been the major element that affects whether a tourist would choose New Zealand as a tour destination or not (Schiff & Becken, 2011).

New Zealand has its own inherent advantages that could make it the most popular tour attraction such as well-preserved ancient architectures and tremendous landscapes. However, considering the disadvantages such as high travel expense, some measures need to be taken to attract visitors to New Zealand; measures such as mobile applications that make it easy for foreigners to find their way around tourist attractions. Smartphones are a ubiquitous tool and are the ideal medium to use to boost New Zealand tourism.

1.1.2 Smartphone and Tourism

In contrast to traditional products that could be observed before purchase, tourism is an intangible experience that always comes with uncertainty and risks (Goossens, 1995). In general, tourists would tend to collect significant amount of information to make decisions of where to go before the journey or during the journey, since tourists possess

limited information and experience about the place they intend to travel within. Most tourists hope to maximize their pleasure in the experience under the condition of limited time and financial budget (Oh, Kim, & Jayakrishnan, 2012). Considering that high expense of accommodation and transportation required to travel to and within New Zealand, precise, timely and appropriate tourism information would be a highly valuable asset that helps tourists make wise choices.

According to relevant research on tourism, the use of certain media and images would create the basis for a system for tourists about selecting (Pan, Tsai, & Lee, 2011). Apart from objective data or text descriptions, vivid media content would also make a difference. After considering the characteristics of the smartphone one could easily reach the conclusion that a smartphone could be the platform for a convenient and mobile electronic tour guide providing such information. For example, positioning through GPS is able to provide precise location as well as information about nearby tour attractions. Today, with instant internet connection the information tourists receive can be updated as often as required. Thus, the information they receive would tend to be relatively precise and timely. This can be compared with traditional tourism where tourists have to utilize the combination of map and tour guide book to discover where they are and what places are worth travelling to. In addition, traditional tour guide books could be out-of-date with respect to the contents, and only provide one-way information. Smartphones have the capability to enable tourists to personalize content they are interested in. What is more, smartphones with connection to popular social networks may be the catalyst that triggers tourists' enthusiasm to travel (Kenteris, Gavalas, & Economou, 2009).

1.1.3 Review of Relevant Solutions

Admittedly, this thesis is not the first to discover the significance of the electronic tour guide. There have been several suggested ideas in other theses and implemented smartphone applications (or apps) on various app stores. In this section, these technical solutions are critically reviewed to identify any deficiencies in existing solutions. Based on the drawbacks, motivation of this thesis will be generated.

The first model was designed in 2009 when smartphones were still not pervasive (Kenteris et al., 2009). This model, named myMytileneCity, offered the tour information of Lancaster, UK. The model was implemented through Java and relevant XML technology, see Figure 1.1, as at that time smartphone technology was still far from mature.

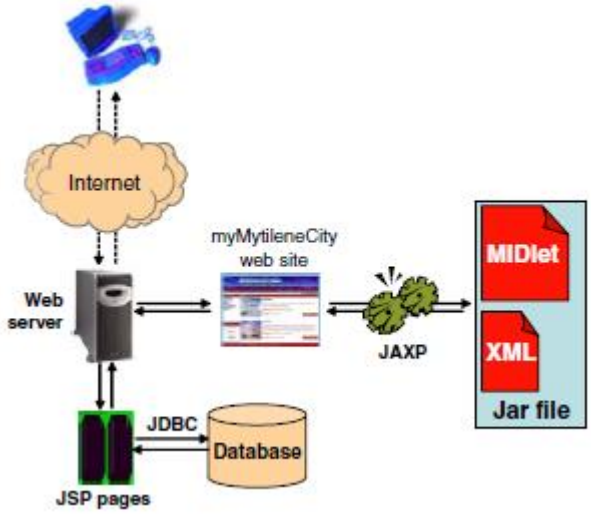


Figure 1.1 The structure of myMytileneCity

The user would first go to myMytileneCity web site to choose content, such as recommended restaurants, activities, hotels. Once the user defined her/his preferred content, the user-specific requirements of content would be defined in an XML file stored on the web server.

The format of the XML file is defined as in Figure 1.2. The server would generate the MIDlet, namely the J2ME application, according to the requirement of the XML file. The user could download the application from the server and install it on the mobile phone for further tour use.

```
<?xml version="1.0" encoding="UTF-8"?>
<city name="Mytilene">
      <content id="2" type="Accommodation">
            <sub id="20" name="Hotels">
            <cname>Hotel Erato</cname>
            <cdescription>Fine view of the Port. Easy access .....</cdescription>
            <caddress>Pavlos Bostani 2, Mytilene</caddress>
            <cphone>22510-41160</cphone>
            <cimage>erato.jpg</cimage>
            </sub>
      </content >
      <content id="2" type="Accommodation">
            <sub id="20" name="Hotels">
            <cname>Blue Sea</cname>
            <cdescription>Situated on the left side of the Port .....</cdescription>
            <caddress>Kountouriotou 22, Mytilene</caddress>
            <cphone>22510-23994-995</cphone>
            <cimage>bluesea.jpg</cimage>
            </sub>
      </content >
</city>
```

Figure 1.2 The format of XML file

When the content that matches user preference has been updated, the server would send a text message to the user to alert her or him to download the updated application from the server and re-install it on the mobile phone again.

There are several advantages of this model. Firstly, since the application has embedded content, it could run in offline state. Constant connection with Internet is not a prerequisite of this model. Secondly, users could personalize the content they are interested in. Time is saved by filtering the content by tourist's preference.

Although the application could run in an offline state, stored content would have requirements on the limited memory space of the mobile phone. Generally, tour information would include pictures or videos rather than plain document description. Whether a mobile phone could store all the multimedia tourism information is doubtful. Apart from this deficiency, updated content cannot be shown simultaneously on the phone; more manual operations from the user are needed to be executed. Last but not least, the efficiency of operating server could not be guaranteed. Lancaster is a fabulous tour city; as a result overwhelming number of connections from tourists' mobile phones is likely to be demanded. Also, the server has to store large quantity of tourism contents. Therefore it is probable that the traditional server may encounter bottlenecks in providing services.

The second thesis (Masoumeh & Mehregan, 2012) suggested that recommendations could be provided to users based on the result of data mining. This system clustered all customers according to their tour location first, so as to reduce the workload of search and create the neighborhood of tourists. In the second step, profiles of tourists were created including behavioral pattern, rating of purchased goods and content of the tours. In the last stage a two-level graph was created consisting of the tour-tour and tourism-tour similarities. Finally, recommendations could be made to tourists in the same category regarding tour attractions and items which are worth purchasing based on the tour behavior of other members and rating of purchased goods from other members in the same group. Data mining is undoubtedly useful for other business analysis such as in supermarkets that email their customers specific discount information based on customers' purchase records. However, the question of whether data mining could be used practically to boost tourism remains doubtful. First of all, the first step of data mining performed was to categorize tourists according to their location. However, unlike customers of supermarkets who would normally frequent certain fixed supermarket branches in months, tourists may change their locations in days. Relatively long-time consumption in data mining may make the result hard to adapt to tourism. Secondly, the authors also admitted the limitation that there were problems of sparsity in the data that could not be ignored. Since data mining is conducted on the basis of large data, the number of tourists and tour attractions are large. In this scenario, even energetic tourists may have travelled less than 1% of the tour attractions. So, the tourist-tour attractions interaction matrix could be sparse. While the intention was to provide users tourism content based on their interest, providing such choices to tourists through data mining has not been shown to be an appropriate choice.

On the other hand, Höpken, Fuchs, et al. (2010) suggest that context-aware data mining may be suitable for tour guide applications as context awareness will help to provide information relevant to tourists' location changes. This would avoid information overload for the tourists and a context adaptive tour guide could provide nearby information according to the tourist's location change. Within context adaptive systems, location is the major criterion to be considered since continuous information update

requires changing information about the location. However, the study only considered the change of outdoor location and ignored indoor locations. In practical scenario, tourists would not only explore outdoor landscape but also indoor buildings.

The fourth study (Rodriguez-Sanchez, Martinez-Romo, Borromeo, & Hernandez-Tamames, 2013) did cover indoor locating. This study offered an idea for electronic tour guide not only covering outdoor location but also being aware of indoor context. In outdoor scenario, as we are all familiar, the location is provided by GPS. In indoor scenario, the smartphone could receive the location feedback from blue tooth signal or reading the QR code. The graph presented in fig 1.3 illustrates the whole process:

Figure 1.3 The whole process to generate the app
Source: (Rodriguez-Sanchez et al., 2013)

In the first step, the user would retrieve the precise current location through the Bluetooth or QR Code. After getting the precise location, users choose what they would like to browse, such as restaurants, museums or theaters. Next the user chooses the smartphone platform her/his smartphone is running on. After all these procedures, the server would generate the app containing chosen context which the user had to download and install. Once the user had downloaded the app from the website and installed it on the phone, the electronic guide could be accessed from various locations.

Compared to ideas from other studies discussed earlier, Rodriguez-Sanchez et al (2013)

advised that indoor locating could be conducted through Bluetooth or QR Code. In practical scenario, QR Code has some advantages; the code can be printed on a piece of paper and can be read by a code reader. Savings on hardware expenses could be significant. However, the major deficiency of this design is lack of constant update of information. Defined contents are included in the application and so, contents cannot be updated immediately on location change of the user. What is worse, due to the limitation of disk and memory of smartphones, storage of downloaded content takes up a good deal of space which is not ideal. Given a smartphone's limitations of disk and memory, what would be ideal would be for the smartphone to be the interface between users and back-end servers.

As to the design of the structure of the tour guide system, the most traditional one is GPS-Server-Mobile model (Chang-Jie & Jin-Yun, 2008), shown in Figure 1.4.

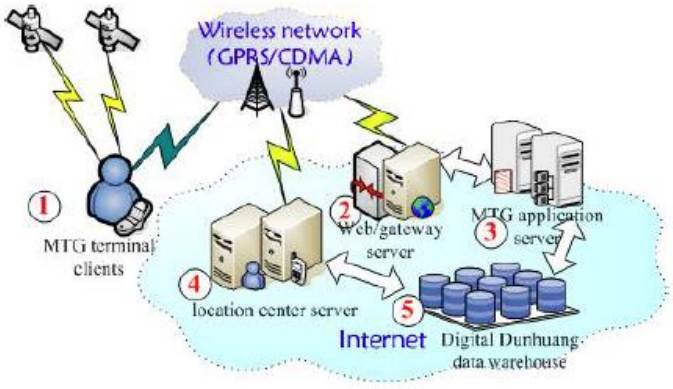


Figure 1.4 The whole structure of the traditional model

This model is deployed at Dunhuang Mogao Grottoes in China for tour guiding. As can be seen from the Figure, the mobile terminal clients firstly receive location information from the GPS. Through wireless network, requests are transferred to several different servers. According to specific location, information of nearby attractions is transferred back to the mobile client.

The major deficiencies of this model are: firstly, the tour guide model would not cover the indoor areas where GPS signal is significantly blocked. Secondly, similar to ideas posted by previous research, this model could encounter bottleneck during times when

there are a great number of requests from clients. Thus there are obstacles to maintaining efficiency in the model.

Another tour guide model mainly serves indoor scenarios such as art gallery or museum (Hsu & Liao, 2011; Kingston et al., 2012). In this model, the total indoor surrounding is described as a tree. The root is the building. The building has floors as branches. Each floor has rooms as branches. Each room has exhibited objects as branches. Each exhibited object has its own RFID tag which contains its location information. Once the RFID reader gets the location information, the reader receives relevant illustration information of the exhibited object.

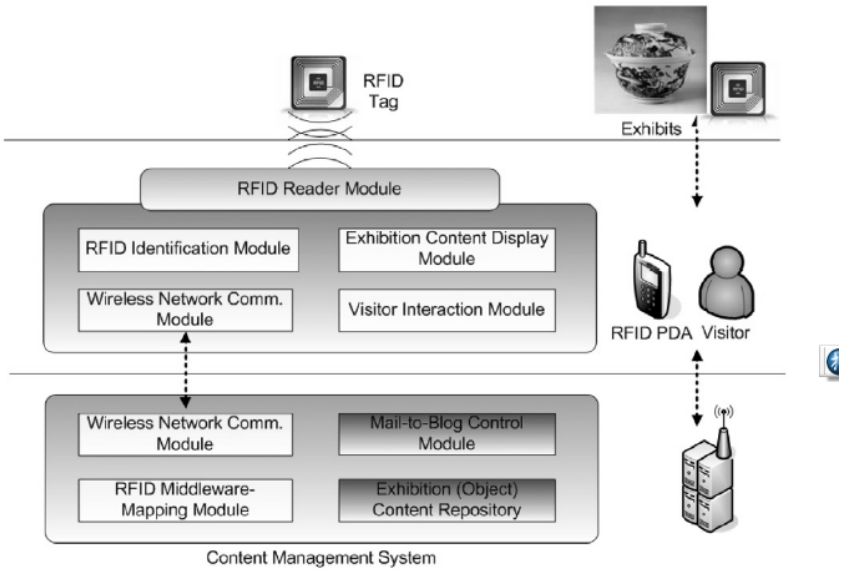


Figure 1.5 The whole structure of the indoor model

As can be seen from the Figure 1.5, there are mainly three parties in this model, namely exhibits, RFID PDA (or smartphone) and the server. Usually the RFID tag is attached on the exhibited object. When the PDA reads the signal, it connects to the server, and sends requests based on the received signal. According to requests, the server would return relevant introduction information back to the PDA.

This thesis provides us inspiration. First of all, the whole building could be divided hierarchically. This method of mapping between practical building and data structure would be of great help in designing algorithms for indoor navigation and indoor locating. Secondly the reader, which could be PDAs or smartphones, receives certain

location information from a tag and subsequently receives tour guide information from a server.

We consider the drawbacks of RFID tags as well: RFID technology is fairly expensive(Ayre, 2012). First of all, RFID tags are required to be purchased and placed with every item in the exhibition. If this thesis adopts this technology, considerable amount of RFID tags are needed to be placed in exhibits and vertices in the building. Secondly, upgrade is expensive. If contents of RFID tags are changed, developers must use specific hardware to rewrite the RFID tags.

After reviewing and analyzing the technical solutions as described above, several major points and weaknesses were identified:

Most smartphone tour guide applications do not cover the indoor exploration due to the blocked GPS signal in indoor scenario.

Some applications prefer to download defined contents to smartphone for future utilization due to the high expense of internet connection. However, with the constantly decreasing charges of internet downloads, contents could be updated if the application is context-aware.

The efficiency of the server needs to be considered. It is theoretically possible for the server to provide detailed and precise tour guide service covering not just one specific area, but the whole city or the whole country. There are mainly two obstacles to be resolved to maintain normal operation, namely great number of requests from clients and storing huge amount of information.

1.1.4 Smartphones and Cloud Computing

In contrast to traditional servers, cloud platform is elastic with changes to requirements. Maintenance and administration of a collection of servers could be easily left to cloud providers such as Google App Engine, Windows Azure or Amazon EC2. Similar to consumption of electricity, the cloud provider would only charge companies when resources are used. For example, cloud providers use measures such as the amount of

data transferred between the cloud and other hosts, and amount of storage utilized as a basis to calculate charges for services. Before cloud computing, small companies had to purchase enough servers to handle sudden avalanches of requests but the servers tended to be idle most of the time. In this scenario, high budgets were required of small companies and they also had to hire staff to maintain the normal operation of servers. However, with the advent of cloud computing, small companies only have to pay more when requests from clients are overwhelming i.e. as at peak demand time (Hansen, Grønli, & Ghinea, 2012).

Since cloud computing is still quite new, there are very few research studies focused on cloud platforms as the server of mobile clients. In New Zealand, there is still no implemented technical solution that cloud computing server is used as the back-end server for electronic tour guides. Nevertheless there is previous overseas research that provides us inspiration. In a study, place tags which contained the location and description of travel information were used (Delev, Gjorgjevik, & Madzarov, 2010). All these tags were transferred from users' smartphone to cloud storage. When other users look for what they could enjoy at a specific location, the web browser could download tags from cloud platforms. This client-cloud model serves as a prototype for the implementation in this thesis. Cloud-based systems could also provide the basis of Location Based Service(LBS) checked-in services (Bisio, Lavagetto, Marchese, & Sciarrone, 2013). With respect to operation efficiency, research has shown that Google App Engine, the cloud platform providing web application hosting services, could be elastic when dealing with considerable amount of mobile requests (Bisio et al., 2013).

In this thesis, cloud computing will be utilized as the back-end server to support data storage and logic calculation of electronic tour guide.

1.2 **Research Questions**

From the introduction to temporary and past electronic tour guide systems discussed in the previous sections, there are a few major deficiencies that have been identified:

- In most cases, indoor locating and navigation are not implemented. In some

specific scenarios, hardware cost is too high for implementation.

- Contents of the apps discussed in previous sections are predefined, and cannot be updated as context changes.
- Servers encounter efficiency problems.
- The cloud computing platform can be a potential competent candidate to be the server; there is no previously implemented tourism navigation model that uses the cloud platform in New Zealand.

Thus, the research questions are:

Research question 1: How can smartphones guide tourists to travel in New Zealand, no matter they are in outdoor or indoor environments?

Research question 2: How can we use the cloud computing platform in a navigation application?

1.3 **Contributions**

In this thesis, a prototype of indoor and outdoor tour guide system named NZ Tour Navigation is proposed and developed.

In this research, first of all, detailed information of several buildings and tour attractions is stored in the cloud computing platform, namely Google App Engine. The information includes the structure of floors, tour attractions on certain floors, and general introduction of buildings and outdoor landscape. The logic of navigation would be implemented in the cloud server.

Once the server side is built, the mobile application NZ Tour Navigation is designed and implemented as well. The smartphone platform used in this thesis is Android. In the app Tour Navigation implemented in this thesis, the tourist will be able to get information including local weather, general introduction of local city, general introduction of nearby tour attractions, and indoor navigation information for tour attractions that can be accessed indoors. The user can also post social network status updates through this app.

1.4 Overall Methodology

Constructive research is the methodology followed in this research. The major objective of this research is to build a model that proves the practicability of our ideas. This is exactly the meaning of constructive research which is concerned with designing frameworks or technical progress (Crnkovic, 2010). Thus, several steps of constructive method are followed:

- Form a question that has research potential.
- Have a general idea and understanding of relevant similar research.
- Post an innovative idea.
- Prove that idea is implementable.
- Analyze the applicability of implemented ideas

In this chapter, the problem has been forged. In the following chapter, the literature review conducted will be discussed. Later, a prototype is designed, developed and evaluated, and recommendations for future research are made.

Chapter 2 Literature Review

This chapter critically reviews theses and technical documents relevant to indoor and outdoor navigation systems. Firstly, we critically review the literature about navigation that helped us draw the research question from the strengths and limitations of relevant literature. Next, the chapter introduces and analyses the utilization of QR Code scanning in daily scenario and how this technique can be implemented through smartphone to provide locating service. Later several theses about the Graph algorithm, to do with path navigation, are considered and reviewed. Following this, we have a look at Google App Engine and how to construct the server on the cloud to provide navigation and data storage service to various clients. Finally, some factors about designing and implementing an Android Phone app will be emphasized while reviewing Android official documents.

2.1 Navigation

2.1.1 Indoor Navigation

Outdoor navigation technology is mature. With the help of GPS most of us have access to exact location service (Montague, 2010). A GPS receiver is able to help outdoor adventurers to reach destinations and return safely no matter they are hiking, fishing, kayaking or mountain biking in the country (Hinch, 2010). Enterprises such as Google and Apple have provided perfect and exact routing services in the outdoor environment. For city residents, smartphone with GPS recommends them nearby living facilities and instant traffic information according to their current location (Ibrahim & Ibrahim, 2010).

Compared with outdoor navigation, indoor navigation enjoys far less popularity (Montague, 2010). GPS signal is mostly blocked within buildings. Exact location navigation service provided by GPS is not available within buildings. In addition, indoor environment is more complicated than the outdoor environment(Montague, 2010). Outdoor environment can be observed by the map, however as to the indoor environment, often there are many storeys. Each storey contains corridors and rooms. For example, in a vast hospital, patients are easily lost and may find it difficult to find exits, entrances and elevators. In the following section, several main articles that are related to indoor navigation are introduced.

2.1.2 Previous Indoor Navigation

In this section previous indoor navigation studies are critically reviewed. Their strengths and drawbacks are identified to reach our research questions.

Micro-Electro-Mechanical Systems (MEMS) accelerometer and gyroscope sensors are able to be used for indoor navigation (Iozan, Collin, Takala, & Rusu, 2011). The initial focus is on how users move. Accelerometer sensors are used to determine how far the user has moved. Peaks of data variation combined with the length of footstep are counted to estimate the distance covered. Gyroscope angular rate data is used to

determine the direction users are heading for. This thesis informs us on how to use low-cost infrastructure to direct users to navigate. However, there is no discussion about how the decision is made to choose the route presented to users. If the user's route is short, exact calculation is not needed to inform users the correct route to navigate. Only the start point and destination point are needed to be shown to the user to inform them whether they are in correct route.

Another study discussed the utilization of magnetic field for navigation and locating (Gozick, Subbu, Dantu, & Maeshiro, 2011). Magnetic field variation can be affected by metal material within the building or human-made sources such as energy power system. This research discovers that magnetic field of corridors and pillars within the building is unique. Due to the unique magnetic field of corridors and pillars, a magnetic map can be designed for users who wear magnetic sensors. However, the article only covers the discussion of navigation of one storey. In the case of two adjacent storeys, interior structure such as pillars and corridors may be similar and the magnetic fields are then similar. However, the storey the user is on cannot be discovered.

Radio signals have also been used for locating and navigation (Yanying, Lo, & Niemegeers, 2009). At first researchers distribute infrastructure such as WIFI hotpots and RFID tags to transmit radio. A set of measurements of these radio signal are then taken at different locations. Map of signal received angle, strength of signal and location are stored. When the user detects signal at some sites, the user is able to know where he or she is by comparing the three parameters angle, strength and location. Once the location is known, shortest path algorithm is performed to calculate the shortest path for the user. However this system needs wide-range distribution of hardware and frequent update of the database of location record.

All the research discussed above require external signals to model the map, such as magnetic field, radio frequency, RFID tag, and WIFI signal. Some researchers believe that traditional map is a more straightforward navigation method (Link, Smith, Viol, & Wehrle, 2012). The software system gains the indoor map of the building. Smartphone

which is held by the user has the accelerometer sensor. According to the variation of accelerometer data, the smartphone calculates how many steps the user has walked and in which direction the user is heading. Based on the original site of the user, the smartphone displays the location the user is currently in. However, accelerometer embedded in smartphones is not accurate enough and mistakes can accumulate but no specific correction methods are discussed in this research for correcting current user location state. As to the user, if they are looking for the path on the map but have the wrong present location information, it cannot result in correct navigation route.

2.1.3 Navigation Problem

Having analyzed the articles discussed in the last section, several that require consideration with relation to navigation are:

1. Infrastructure distribution. When using radio signals for locating users, research suggests that certain hardware be deployed for locating service. The density of the hardware needs to be high enough to locate the user accurately (Yanying, Lo, & Niemegeers, 2009). However, there is a cost to wide-range distribution of hardware.
2. Locating. The articles discussed do not provide an accurate and cheap technical solution for indoor locating. Only wide-range distribution of hardware is considered in these articles (Gozick, Subbu, Dantu, & Maeshiro, 2011).
3. Navigation. The algorithm that provides navigation service is not clearly discussed. Only general ideas are covered in these articles.

With these three major points considered, further review of literature was conducted to discover inspiration and other experiences.

2.2 QR Code

2.2.1 Overview

As discussed in the previous chapter, RFID tags and GPS which are used as technical solutions for indoor locating do have their own disadvantages. High expense of

implementation and inaccuracy make them not appropriate for deployment of navigation system on a large scale.

In this thesis, QR Codes are used to gain location by scanning the QR Code. In this section, we review what a QR Code is, its advantages, how it has been used in previous cases and how these cases inspire our thesis.

2.2.2 Definition

QR code, which stands for Quick Responsive code, is a two-dimensional machine readable barcode that contains information about the object it attaches to (Lin, Luo, & Chen, 2013). Although purely consisting of black and white squares, QR code can contain considerable amount of data and support fast reading. What is more, scanning the QR code can be conducted at various angles. Compared with other techniques such as barcode which requires that barcode should be placed directly over the scanner, QR code can be scanned in various directions in various angles.

The procedures to generate a QR code are not complicated (Lamb & Johnson, 2013). Significant amount of QR code generating tools are able to be accessed. Taking QR code generating website Unitag as an example, users input their text information in the first step. After this, they specify the kind of color, style of QR code, and embellish with photo personalized by the user. In the final step, the picture of QR code can be downloaded.

Since the QR code can be generated with ease and at low cost, it is widely utilized in different scenarios. In a library, QR code is widely deployed to link the printed documents with electronic data (Andrew, 2011)where the QR code provides the link from printed journal articles to electronic journal. For environment friendly use, the QR code provides the electronic alternative PDF format to physical printed books. For further understanding of certain printed books, QR code can provide links to videos uploaded on a website. As can be seen from these examples, the QR code contains valuable information about the object the QR code attaches to.

2.2.3 QR Code for navigation

QR code has not been used for tour navigation before. However, similar cases can inspire our research.

In one education project, enthusiasm for physical exercise is triggered through scanning QR code (Shumack, Reilly, & Chamberlain, 2013). Students were instructed to scan the QR code at the first site to read the navigation information to the second site. By scanning the QR code when they reach the next site, points gained and how to go to the next site could be read. The student who gained the most points in the least time won the trophy.

As can be seen from the research above, the QR code could contain specific information including geo-location. What is most important, compared with RFID Tag and other techniques, QR code is a low expense and available for deployment at large scale. Case mentioned above has indicated that QR code is spa good low-cost candidate for use in a navigation application to indicate location.

In this thesis, the QR code encrypts the vertex which models the indoor structure. How to model the indoor structure using vertex is introduced in the next section.

2.3 Shortest Path

2.3.1 Overview

After reviewing documents of locating, the algorithms to show navigation from one place to another place are reviewed in this section. The algorithm question of shortest path is a heated topic in computer science research area. Since the graph which is the base of shortest path question can simulate various practical scenarios, issues of shortest path have been deeply researched to solve different kinds of practical problems. In this section, other technical implementations based on the graph are reviewed. In the next section, the basic conception of graph which is utilized in this thesis is introduced.

The most widely researched question is the discovery the shortest transportation path from source site to destination site within a city (Joshi, Sridhar, & Chandrasekharan, 1993). In their paper routes of the city were mapped as vertices of the graph. How much time is spent to cover the distance of a route was mapped as the weight of the edge. To calculate the shortest transportation path is to calculate the shortest path from source vertex to destination vertex.

In some scenarios, various types of transportation including tubes, buses and flights are considered as well. As a result, shortest path problem through different layers of transportation is the other research topic (Zhiyuan & Yan, 2009). In the first step, the authors overlaid multiple layers and rebuilt the topological relationship according to the scan line. Based on this relationship, the authors designed the algorithm adopting heap and discovered an innovative algorithm to reduce complexity. In the final stage, it was proved that the algorithm can be efficiently adapted to solve the shortest path problem with multiple layers.

More advanced and complicated situations such as how to direct a robot to navigate through obstacles (Golda, Aridha, & Elakkiya, 2009) is discussed by researchers as well. In a static environment filled with obstacles, a two-dimensional array is built to model the environment. The whole environment is divided into cells. Each cell is detailed whether there is an obstacle such as a wall. To travel from the source cell to the destination cell, the robot only chooses the adjacent cells without obstacles. If it encounters an obstacle on the path, it retreats to the last cell to search for another cell to navigate to until it reaches the destination cell. In a dynamic situation, the robot uses sensors to identify and track changing obstacles, and continuously change its two-dimensional array.

Another practical implementation of shortest path navigation is in directing blind people to go through the indoors of a building (Alghamdi, Van Schyndel, & Alahmadi, 2013). Researchers use RFID tags distributed in the building to identify the location

information. The structure of the building is modelled. Blind people who wear RFID reader receive RFID signals and know their exact location. According to the source RFID tag location and destination RFID tag location, shortest path navigation algorithm is conducted to calculate the shortest path. Thus the blind people are directed.

In the next section, some basic concepts about shortest path question will be introduced. Later, how to simulate the indoor structure is detailed. Finally, to make the algorithm more efficient, some theses are reviewed to inspire our thesis.

2.3.2 Basic Conceptions

Although graph is widely utilized in other comprehensive scenarios, this thesis just adapts the basic concept of graph.Only the concepts that contribute to this thesis are reviewed in this section.

To begin with, basic conceptions of graphs is introduced. A graph G=(V,E) consists of the set V of vertices and the set E of edges (Weiss, 2007). Sometimes weight or cost is used to describe the edge. In practical scenario, the cost of the path sometimes stands for how much time is consumed to cover the distance or how much money has to be paid for a vehicle to travel the route. The path is a series of vertices w1, w2, w3, … wn where (wn, wn+1) belongs to the same edge.

The shortest path question is to discover how to travel from one vertex to another vertex while the least cost is spent. Since in this thesis, the length of corridor does not vary greatly, e.g. from 10 meters to 100 meters, the shortest path of this thesis is considered as unweighted path length from the source site to destination site. The cost of each edge is considered to be the same.

To discover the shortest path from the source vertex to other vertices, there are mainly three attributes of each vertex to be considered, namely Known (whether this vertex is known or not), D (the distance from the source vertex to this vertex), as well as P (the

path from the source vertex to this vertex) (Weiss, 2007).

At the beginning, since all the vertices except for the source vertex are unknown and unreachable, the Known=F (false), D=INFINITY and the P=null. Starting from the start site, this algorithm keeps searching for the adjacent vertices of the vertex which is the last one of the shortest path. The algorithm to discover the shortest path is illustrated as follows:

At first, all the vertices except the start vertex are initialized. Three attributes of every vertex is initialized in preparation for calculating the shortest path. In the second step, starting from the start site, every vertex which is adjacent will be discovered and the path from the start to this vertex is recorded. Starting from these adjacent vertices, the algorithm will be conducted again until all the vertices in the graph are discovered.

```
Void function ( Vertex s) {   //s stands for the source vertex
   For every Vertex v {
   // as to each vertex, initialize the three attributes
   v.D=INTINITY;
   v.Known=F;
   v.P=null;
   }

   s.D=0; // the distance of source vertex to itself is 0
   for (int currDist=0; currDist<number of vertices; currDist++){
     for every Vertex v
      if( v.Known==F&& v.D ==currDist){
      v.Known=T; //the vertex has been discovered
       for every vertex w which is adjacent to v{
          if ( w. D==INIFINITY){
                   w.D=currDist+1;
                   w.Path=v;
          }
       }
      }
    }

    }
```

Figure 2.1 The shortest path algorithm

Source: (Weiss, 2007)

However after observation of this algorithm we find that the time complexity is $O(|V|^2)$. The reason for this situation is that even if a vertex has been discovered it can potentially be scanned again.

Therefore in the improved algorithm, a queue is established to contain the discovered vertices. In this situation, the discovered vertex will not be scanned again:

```
Void function ( Vertex s) {   //s stands for the source vertex
   Establish a Queue q to contain vertices;
   For each Vertex v, set vertex's distance is INFINIT;
   For start vertex s, the distance is 0;
   Put the start vertex s into queue q;


   While(!q.is Empty ( ) ){
   Heap ( ), get the vertex v from the head;
   For each vertex w adjacent to v {
If (w.dist==INFINITY){
   w.dist=v.dist+1;
   w.path=v;
   q.enque(w);
}
}
}


}
```

Figure 2.2 The improved shortest path algorithm (Weiss, 2007)

2.3.3 Modeling the building structure

A typical method to model indoor building structure is to consider each junction as a vertex. The definition of junction includes the junction between a room and a corridor, the junction between two corridors and the junction between a corridor and a staircase (Yu, Li, Liu, & Ning, 2012).

As can be seen from Figure 2.3, S1006, S1002 are vertices, being the junction between room and corridor; s1003 is a vertex, being the junction between two corridors. The other type of junction is the room entrance such as s111 and s110 in Figure 2.4.

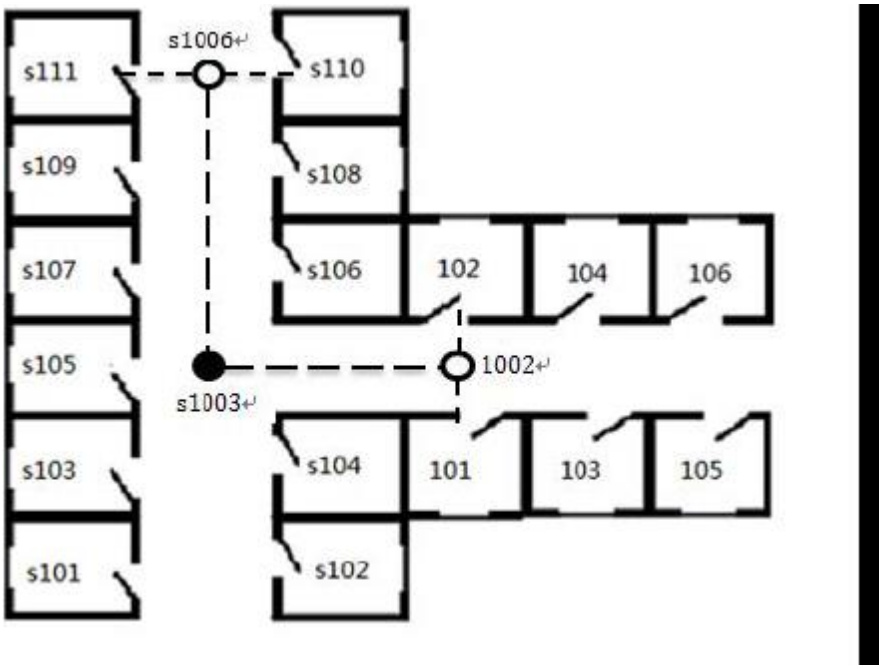


Figure 2.3 Indoor building map (Yu et al., 2012)

The model of this indoor structure is shown in Figure 2.4. Once the graph of vertices is established, the algorithm to discover the shortest path between vertices could be utilized. In this research, as discussed in the last section on QR code, the QR code which encrypts the vertex number is attached to the location. Once the user of the mobile app scans the QR code, the app is informed about which specific vertex the user is at.

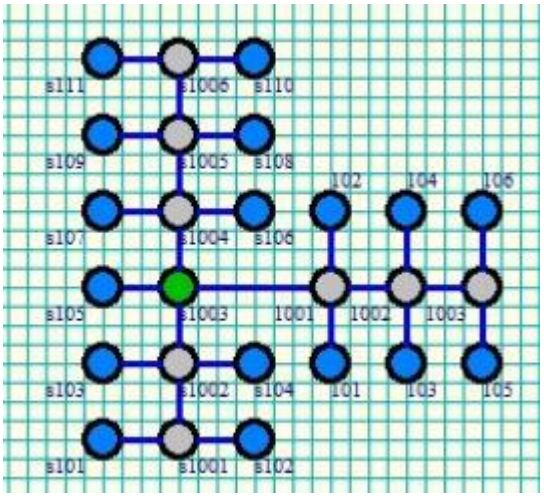


Figure 2.4 Indoor building model

Source:(Yu et al., 2012)

For more efficient computing performance, improved algorithms pay more attention to

vertices that contribute more to path searching, called critical vertices (Yu et al., 2012). Using Figure 2.4 model as example, once the user reaches S1006, it is highly possible that the user will find no difficulty to discover the vertex S1005 and S1004 since these three vertices are in the same corridor. Since S1006 contributes more to find the path, S1006 is referenced as a critical vertex. Critical vertex is the vertex that connects two corridors or two storeys. Shortest path searching is not conducted on normal vertices any more but on the critical vertices. In Figure 2.4, the critical vertices are s1006, s1003 and 1003. Similarly, the shortest path searching is not conducted on normal edges any more, but on the critical edge. Critical edge is the edge to connect two critical vertices. The critical graph is the set (V, E) where V is the set of critical vertices and E is the set of critical edges.

According to the comparison experiment, since the vertices scope has been narrowed down, the algorithm performing on critical graph is much more efficient than the same algorithm performing on the graph using all vertices.

2.3.4 Improved Algorithm

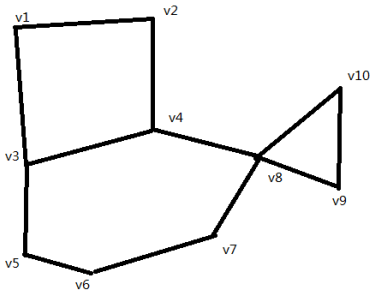
To improve the performance of shortest path searching, not only the model of the graph can be improved, methods can be various according to the practical scenarios. After critically reviewing certain amount of literature, there are mainly two methods to improve the algorithm in this thesis.

First, since the logic to provide shortest path will be located on the server, if significant amount of users request for the shortest path simultaneously, the demand on computational resources would be intense. Thus, the shortest path is pre-computed and stored on the server (Yu et al., 2012). $O(n^2)$ space is consumed but time is saved.

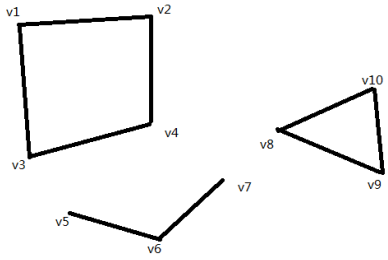
Second, sub-graphs are utilized to improve the efficiency of the shortest path search algorithm, resulting in an improved algorithm (Wu, Xu, Hu, & Yang, 2003). For example, in an urban city, to calculate the shortest path with many streets and motorways would have low efficiency. As a result, streets can be divided into different

zones. First at higher level graph, the shortest path is calculated between source zone and destination zone. In the second step, the shortest path between source vertex and exit vertex within source zone, and the shortest path between entrance vertex and destination vertex within destination zone are calculated. The calculation tasks can be conducted simultaneously.

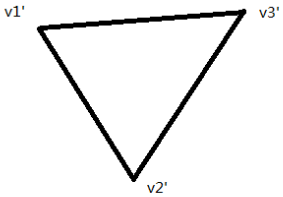
As to the indoor building in this thesis, this idea also inspires us. Computing the shortest path is not necessary to be conducted on the whole graph containing all the storeys.



(a) The Whole Graph



(b) Partition of Graph



(c) Higher level graph

Figure 2.5 Sub-graphs of the whole building

As Figure 2.5 illustrates, each storey could be modeled as a sub-graph. The entrance vertex and exit vertex of each sub-graph could be staircase and lift. The whole storey consists of nine vertices from v1 to v9. Vertices v1, v2, v3 and v4 are in a sub-graph v1'. Vertices v5, v6 and v7 are in a sub-graph v2'. The other three vertices are in the last sub-graph v3'.

This thesis will divide the whole building into storeys. Each storey is modeled as a graph. Within two adjacent storeys, two adjacent graphs are connected through two vertices where each vertex belongs to each graph. This vertex could be the lift or the entrance or the exit of each storey. In each storey, only the critical vertices are modeled.

In summary, this section discussed how critical vertices will provide improved performance for the shortest path algorithm. In addition, it was decided to pre compute and store the shortest path in order to mitigate demand on the server.

2.4 Cloud Server

After reviewing how to provide efficient locating and navigation, in this section we review theses and official documents about the cloud server which contains the locating and navigation service, and stores the location information.

2.4.1 Overview

There are various reasons why cloud computing platforms are increasingly popular as the back-end server of front-end mobile applications. First of all, the overwhelming expense of implementing network devices to cover requests from metropolitans or countries is an important factor for small companies to consider (Bisio et al., 2013). Handling a large number of requests from clients requires setting up robust servers, especially at peak times. However, once requests are not at peak, resources requested of a server are trivial. Cloud computing platform enables computing resources to be used when the resources are needed. Similar to electricity, when devices need more electricity, more electricity is consumed. When more computing resource is needed, cloud computing platform provides more storage and processors to conduct computing tasks.

Secondly, the use of cloud servers with smartphones, which have limited resources, is advantageous. Smartphones lack the battery capacity to support consistent computing when compared with a computer or laptop. In addition, due to insufficient computing capabilities smartphones are not suitable for conducting computing tasks solely (Delev et al., 2010). However, with Cloud computing servers to conduct computing tasks, smartphones can connect with networks and just play the role of interface between servers and users.

As a result, there has been some research into designing the computing model consisting of cloud computing server and smartphone clients (Jarle, Tor-Morten, & Gheorghita, 2012). Push notification is one of the communication methods between servers and Android smartphone. Jarle et al (2012) tested the stability, response time and energy consumption within four most popular push notifications: C2DM, XMPP, Xtify and Urban Airship. Google App Engine played the role of server in the benchmark test to send push notification to smartphones. This previous research has proved that Google App Engine has the potential to be the back-end server for smartphones.

Delev et. al. (2010) used smartphone clients to retrieve and upload certain amount of information for sharing while working to promote tourism. They researched on how to use cloud server to store place tags sent by smartphones. Place tags contain brief introduction and rating about specific sites. In addition, Cloud server has also been used to help control activities at a tourism site. In a study, when tourists carrying smartphones entered certain tour sites, the smartphone application registered temporarily to the cloud server. The server monitors the number of registrations and when the server discovers that the number of registrations is overwhelming, a push notification is sent to the tourists to warn that certain tourist sites are crowed and it would be better for them to visit later.

In another study, the cloud server pushes specific advertisement to smartphone clients based on the analysis of information gathered from the smartphones (Durresi, Luarasi, Baholli, & Durresi, 2013).

As can be seen from the above research, cloud server runs as a data aggregator and service provider. Smartphones run as data provider and service receiver. In this thesis, smartphones provide their location no matter they are indoor or outdoor. Based on this location, the cloud server can provide location information and navigation service.

2.4.2 Google App Engine

Google App Engine is a host which enables programmers to build and run web applications on it (Malawski et al., 2013). With various cloud computing platforms, this research chooses Google App Engine as the backend server. As a cloud computing platform, Google App Engine scales as requests and storage requirements change. Compared with other cloud computing platforms such as Windows Azure and Amazon EC2, there are several major advantages (Prodan, Sperk, & Ostermann, 2012):

1. Free to deploy web applications. While Windows Azure and Amazon EC2 require certain amount of fee to deploy servers on cloud computing platform, it is free to deploy web applications on Google App Engine if requests or data storage is under certain limitation.
2. Convenient to deploy web applications. While Windows Azure and Amazon EC2 require configuring virtual machine to conduct computing tasks, with Google App Engine developers only need to design servlet and certain APIs to design the web server logic.
3. Multiple programming languages. To support developers deploy their logic on the cloud computing server, Google App Engine enables multiple programming languages to design and implement web applications, including PHP, Python, Java and Go.

2.4.3 Setting up the Cloud Server

To launch the server on cloud computing service, there are a few procedures that need to be followed (Jordan & Greyling, 2011).

1. Initialization. The initialization step is to register the domain of the server. On

control panel, the developer registers the domain which enables access to smartphone application.

2. Web project. Once registration is finished, the whole web application is set up ready to be uploaded to the cloud server.
3. Designing the services in the servlets. There is a web.xml file in the project which can be used to configure URL for Http access. After designing the tasks that are finished in the servlet page, such as providing navigation or introduction, certain URL and servlet pages are bound.

Next, design the stored data structure. While designing the data structure, Java Data Objects (JDO) is utilized. As to Google App engine, JDO is a defined typical interface for storing objects in a database. Using this interface, the third-party clients are able to annotate Java objects, retrieve objects from Google Big Table using queries and interact with Google Big table with transactions. In this thesis, JDO is used for object mapping from class to data table. What is more, JDO processes how to set up the table in data store as well.

2.4.4 Servlet

In general, web application designers firstly register a host for a web application. Within the web application, servlets provide the host with the logic for conducting computing tasks and storing data tasks. As to conducting computing tasks, on receiving requests from mobile clients, computing tasks are conducted and results are returned. As to storing data, how to store data through JDO in JSON format is also specified by Servlet.

All the data interaction, request and response are performed through HTTP protocol between mobile client and servlet.

In Google App Engine, it is the servlet's responsibility to provide the service (Google, 2014b). Through Http request, the servlet, which runs the logic to calculate the result, returns the result back to the client. After generating the writer from the Http response, the servlet utilizes the writer to output the result. The model of the servlet is illustrated as follows:

```
    Public class Serviceservlet extends HttpServlet {

      Public void doGet (HttpServletRequest req, HttpServletResponse
resp){

    //conduct the computing tasks

    Writer writer=resp.getWritter ( );//output the result;

    try{writer.write(result);}

    catch(exception ex){ Handle the exception; }
```

Figure 2.6 Servlet model

(Google, 2014b)

2.4.5 Data Structure

The whole system structure design follows Model-View-Controller (MVC) principle. While processing data, for the convenience of modelling from objects in real world to data in Google Big Table, data is firstly modelled as object (Correa & Ricaurte, 2014). Later through JDO mapping from object to table, data is transferred into records of the table (Leone & Chen, 2007). JDO mapping automatically enables object being transferred to rows in Google Big Table. While retrieving records from tables using queries, retrieved records are automatically transferred into object, and the columns in each record are transferred into variables of objects. As to the bridge JDO, the candidate data type should be JSON object.

2.4.6 JSON

For Google App Engine, the format for transferring data to clients is mainly JSON, namely JavaScript Object Notification (DarrelInce, 2010). As to JSON, its syntax does not require the developer to define a method to generate a class. Instead, its syntax enables it to generate a JavaScript Object (Severance, 2012). Thanks to this, JSON object data is easily defined and transferred. Thus, it is widely adopted in internet transfer between clients and servers.

For Example, to create a person object with attributes name and gender:

```
var p={
name:'yeeku',
gender:'male'
};
```

Figure 2.7 JSON format

Compared with JSON's most competent counterpart XML, JSON has three advantages including being light weight, able to support multiple programming languages, and is more understandable (Severance, 2012). Increasingly nowadays, JSON is chosen to transfer data, and not XML, because of these three reasons.

2.5 Android

2.5.1 Overview

As an open source smartphone operating system, Android has attracted considerable attention since its advent. The character of open source makes it compatible for various devices especially smartphones. On one hand, being originally from Google makes it perfectly suitable to use with Google App Engine. Google provides developing toolkit specifically for Android connecting Google App Engine system model. Developing, debugging and testing are convenient. On the other hand, since this operating system could be launched on various devices, fragmentation of devices is a barrier that needs to be overcome (Song, Kim, Kim, Lim, & Kim, 2014). In this section, several key factors of designing and implementing an Android smartphone application will be discussed. These key factors include the http connection between a mobile device and Google App Engine, utilization of google map, and how to download and cache pictures efficiently.

2.5.2 Http Connection

As discussed in the last section, Google App Engine plays the role of data aggregator and service provider. As the party who asks for data and service, the smartphone needs to constantly connect with the data server. However, the data transaction between Google App Engine and smartphones may be highly time-consuming. In this section,

the long-term downloading tasks will be discussed.

Since the http connection with data server may consume a large amount of computing resources and time, it is not a good idea to locate tasks such as the downloading of pictures or videos in the main thread as this may cause the smartphone to be non-responsive (Holleman, 2012). If the downloading task is located in the main thread, considerable amount of time may be spent waiting for responses from the server. , Also, during this time, main thread is not able to interact with any user interface input. Downloading tasks should be conducted within an asynchronous task. Asynchronous task should be utilized for time-consuming tasks. Once the asynchronous task has been finished, it sends a notification to the main thread. After the main thread receives the notification, some change to user interface happens, for example, an alert message will pop up a reminder to user. The whole mechanism is illustrated in Figure 2.8.

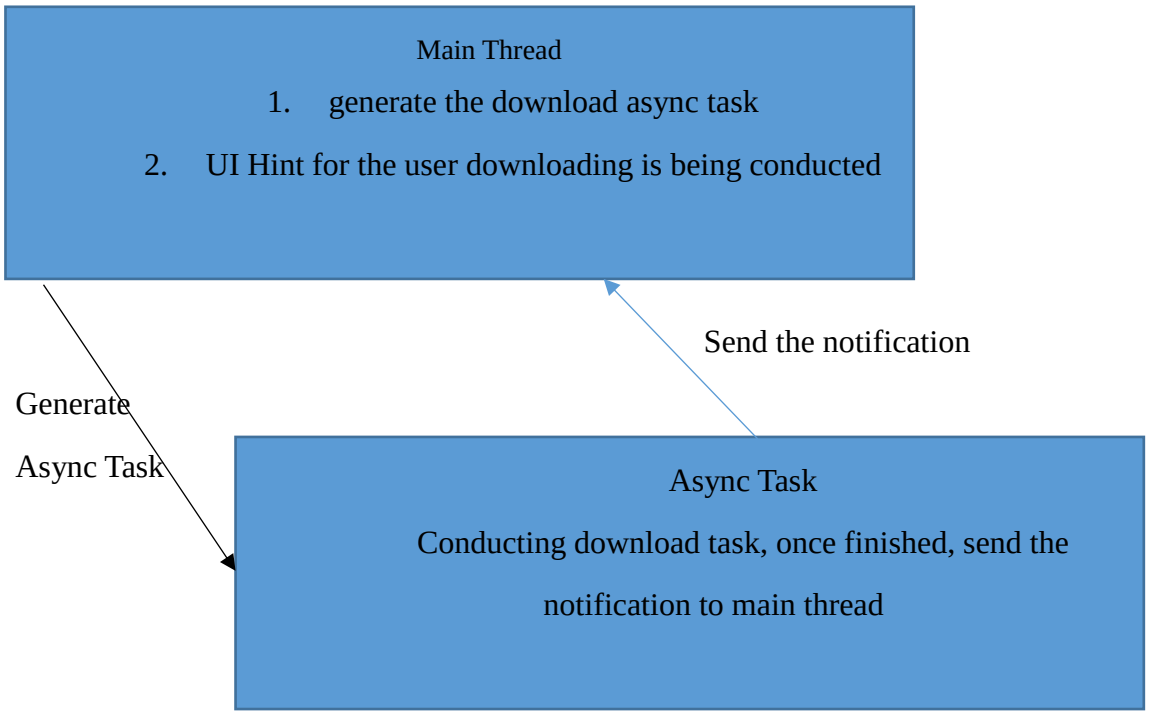


Figure 2.8 Download mechanism

Within the asynchronous task, Apache HttpComponents are used for maintaining internet connection as they performs excellently at creating and maintaining low level Java components concentrated on HTTP and associated protocol. Firstly, the application creates an Http client to generate an Http Get request. If the response is successful, then

the JSON entity which contains the result can be gained and deciphered.

2.5.3 Google Map

The smartphone application being researched in this thesis is required to present tour attractions on a map. In this section, a review of how to display this information on the google map is discussed.

Map view of smartphone applications uses polygons to represent buildings or other objects on the map. In Google map, IOS map or other map tool, a site can be displayed as polygon, such as a pin. A few steps need to be executed before Google Map can be utilized for displaying polygons (Google, 2014a):

1. Download and configure the Google Service SDK.
2. Obtain an API Key specific for the application
3. Include the required setting in application
4. Include the map view in the application

Once these steps have been completed, the data can be presented on the map view through polygon. Polygon is the painted object which is drawn on the map view.

2.5.4 Image Download

As downloading images from Google App Engine is required to show the path navigation, there are some measures that should be considered to ensure images are downloaded efficiently (Android, 2014).

The first consideration is to load bitmap efficiently. Image sizes and shapes vary. In a lot of scenarios they are bigger than the restriction for a traditional application user interface. For instance, in a gallery which shows many photos the images are in lower resolution while the camera is in higher resolution. To load the images from Google App Engine Server, firstly the app has to be informed about what the resolution of images is. Secondly, there are three major measure standards to decide whether the bitmap should

be compressed to load in memory. The first measure standard is how much memory is needed to store the image in memory. The second one is to calculate how much memory is available to be committed, in order to avoid memory exceed exceptions. The last one is user interface's screen size and resolution. After deciding whether an image should be compressed, it can be decided whether the image is to be decoded or not.

The second consideration is whether to launch another thread to download images. Just as discussed in the last section, since downloading images consumes time and computing resources, locating this task in the main thread will undoubtedly block the main thread. Thus, this task will be put into an asynchronous task.

The third consideration is whether to cache the image. Loading one image in an image view is relatively straight forward since one UI component would not create a huge effect on memory. However, we need to consider how to make the application perform efficiently when loading an array of pictures that are needed to indicate the navigation path. While the scroll view is scrolled down and the image in the image view vanishes from the screen, the smartphone operating system automatically collects the image and UI component as garbage. However, when the user scrolls down and up the images, images are recreated again and again. To make the whole application operate smoothly, memory cache and disk cache should be utilized.

2.6 Summary

In summary, the literature review is the basis for implementing the navigation system. QR code, attached on vertices in a building, ciphers the location information. Users use the smartphone to scan QR codes - to know where they are exactly and where they wish to go to certain locations. The smartphone gains the location through the scanning of the QR code. QR code reader within the application deciphers the location information of QR code. In the second step, the application sends requests to Google App Engine through HTTP connection. Once a servlet in Google App Engine receives the request, logic within it utilizes the shortest path algorithm to calculate the shortest path, retrieves the relevant data such as URLs of pictures to show the shortest navigation path from the

Google big table, and return the result in JSON format. Once the application receives the result, it downloads certain images for efficient navigation.

Chapter 3 Methodology

According to constructive research method, the system construction starts with certain specific needs which are illustrated as requirements. In the first chapter, a research question is detailed. Later through literature review of relevant theses and official technical documents, we gain a more sophisticated understanding of the thesis topic and about how to construct the whole system. In the last step, a prototype is developed. To finish these procedures, the whole software lifecycle is run through, including gaining system requirements, designing the structure of system, implementing the whole system and software system evaluation.

3.1 Gaining Software Requirement

Illustrating and discovering major requirements of the system is one of the major challenges in this thesis. In this thesis, the requirements are illustrated through human natural language and use case (Binkley, Feild, Lawrie, & Pighin, 2009).

Since human natural language is utilized by normal human beings to communicate with each other every day, there are few barriers for a human to understand the contents of natural language (Santiago Júnior & Vijaykumar, 2012), compared with other communication channels. To illustrate the software requirements through natural human language is the best alternative solution especially for those without software development background. To narrow down the impreciseness, usually software requirements are illustrated in the following format:

Table 3.1 Requirement Format

Entity	Should	Do
Parties involved in the system	Could / Will	Actions

The ‘Entity’ stands for any parties that are involved in the whole software system. Since requirements illustrate what functions should be finished, ’Do’ stands for what actions the entity will conduct.

Once the format for the software requirement is defined, then requirements of a software system can be defined. A software system has three main requirement categories (Souza, Lapouchnian, Angelopoulos, & Mylopoulos, 2013) which are requirements on engineering, qualification and delivery. The kind of requirements in the first category, requirements on engineering are closely related with software itself. For example, what the user interface should represent and what functions the software should have. These are also called functional requirements. As to the other two types of requirements namely requirements on qualification and delivery, they are related more to product standard. Such requirements are non-functional. To meet requirements and be effective, quality of the software should be guaranteed and bugs of software should be

controlled to a limited amount. For example, a quality requirement may be - software is required to respond to the user within ten seconds.

Although human natural language is much more understandable to normal human beings, there are deficiencies that cannot be neglected. First of all, natural language is not precise enough. Different from statistics or graph, text description does not describe factors in detail. Secondly, misunderstanding is another problem. With the same natural language description of the software requirements, understanding about requirements vary from person to person. Due to its inherent disadvantages, natural human language is only utilized for defining the high-level and general requirements.

To improve deficiencies of natural human language, use case analysis is adopted (Kanyaru & Phalp, 2009). Use case analysis defines the whole system as an isolated entity. Use case analysis can be constructed through several major steps.

First of all, the boundary of the whole software system is defined. The software requirements can be illustrated as interaction between external entities and the software system. Thus, the boundary needs to be detailed to tell whether a party is involved in the software system or is part of external parties.

Secondly, the actors that link external parties and software system are defined. Actors represent interaction between external entities and software system. For example as in Figure 3.1, if the user asks for the navigation path from specific start site to destination site, the actor in this scenario is the request.

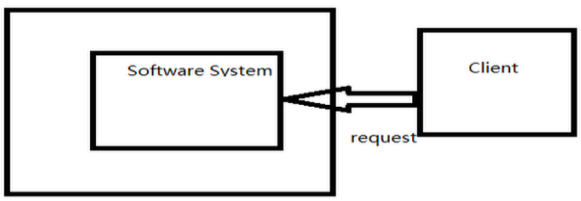


Figure 3.1 Actor

Thirdly, specify each actor in use case scenario. The actor just describes the software requirement in general and abstraction to a certain extent. The use case scenario details each procedure and background for an actor.

Finally all these factors mentioned above are included in the UML graph.

In general, use case analysis requires readers to have software development background to illustrate the software requirements. However, the use case descriptions are more precise and detailed. As to human natural language, even normal human beings without software development background are able to understand the content. Nevertheless, due to the inherent characters of the natural language, sometimes the meaning of software requirements cannot be expressed precisely.

3.2 Methodology for System Design

Our navigation system consists of two major parts. The client party is the Android smartphone application which can be accessed by the user. The server party is located on the Google App Engine cloud server. In this thesis, MVC (Model-View-Controller) is adopted to design the whole software system. With regard to deploying servers on the Google App Engine, there is no interface to present the data. Only data models, business models and controllers are stored on the server. The smartphone application plays the role of view in the whole system.

The data model in this system stores the data of building structure. All the data is stored in Google Big Table. Apart from the statistics of building structure, the information of general description of cities and buildings within cities are stored in Google Big Table as well.

The business model is located in the project repository. At first it takes all the data of vertices in every storey. After calculating the shortest path consisting of different

vertices, it returns the result. Apart from returning shortest navigation path within indoor building, the business model also returns the general introduction data of nearby buildings and city where the current user is, according to specific requests from user's mobile application.

The controller is located in servlets. Once the mobile application sends Http request to the server containing the parameters of vertex of start point and the destination point, the controller executes the business model. Once the business model returns the result, the controller transfers the result back to smartphone application, namely the view in the whole structure.

As mentioned earlier, the view in the whole structure is the smartphone application. The smartphone application sends requests to the server's servlet, namely the party of controller. Once the controller gains the result of a group of navigation pictures' URL, it returns these results to the view namely the smartphone application.

3.3 Method of Data Acquisition and Storage

There are mainly four major types of data which are designed and implemented in the Google Data Big Table. The first type of data is City General Description which contains the general introduction of city, including information about economy, history, geography and people. The second type of data is the Building General Description, which contains the latitude, longitude, building name, general description, images of the building and video of the building. Building data represent actual buildings within a city. The third type of data is the vertices within the building. The fourth type is the exhibited objects within the building, which are near vertices.

City General Description provides the first glimpse of a city to visitors and makes them familiar with the city. In the software system developed in this thesis, the app Tour Navigation will be informed of the current location first, and then the location is sent to the server. Later once the google app engine namely the server gets the location, it returns the description information back to the app Tour Navigation. In this thesis, firstly,

information is collected about economy, history, geography and people of the city. The method of collecting information includes searching through Wikipedia, related tourism journals and books. After getting related information, all this information is stored in the table of Google App Engine. To make presentation of information to app users more vivid, not only text, but pictures and videos are included as well. When talking about pictures, only URLs of pictures are stored in the table. The picture itself is stored in the source code repository as resources. When talking about videos, a similar approach is used. Due to the reason that Google App Engine does not support live stream video, only URLs of videos are stored in the table. Videos are uploaded to YouTube or stored in the data source code repository as resources.

As to how to construct the second type of data, General Building description, information about the building is collected through reliable channels. Once the information is collected, it is stored in the table. Information includes pictures, video and text description. Apart from these, longitude and latitude are stored as well.

The third type of data is the most important type of data for the Tour Navigation software system. The vertices of each building are the basis of providing navigation service. For each vertex, there is an ID which can indicate its universal identity. Apart from the universal identity, there are the identities of eight adjacent vertices as well, which indicates relation between this vertex and the eight vertices adjacent to itself. There are eight because there are eight directions, namely south, north, east, west and northeast, southwest, southeast and northwest. In addition to eight identities, to show how to go through certain paths, pictures of paths are stored as well. Eight picture URLs are stored. These eight pictures show the first point view of eight directions while on the vertex.

The fourth type of data is the information about the exhibited objects. Exhibited objects are objects worth seeing within a building. Similar with the first and second type of data, text description and URLs of videos and pictures are stored. Apart from this information, in order for users to navigate from one place in the building to where the exhibited

objects are located, the exhibited object has a location ID to specify its vertex.

3.4 Method of Testing

Testing on the system is to ensure whether the system is performing on requests and performing efficiently. Its mission is to discover bugs which may potentially cause the system to crash or end with an unexpected result. To discover the bugs there are mainly three types of testing method utilized in this thesis to ensure the whole software system performs correctly.

The first type of testing is method testing (Walter & Just, 2013). Method testing is the basis of testing the whole system since the whole software system consists of various methods. To detect the methods, black box or white box testing is utilized to find three major errors (Krishna Mohan, Verma, Srividya, & Papic, 2010):

1. Syntax Error. If with IDE, these kinds of errors can be detected during coding.
2. Function Error. The method may seem well designed at first glance, but bugs that cause errors may be present. For example, when input parameters have values near the boundary or out of boundary.
3. Design Fault. The initial idea on which the method is planned for implementation is wrong. This kind of error is mostly hard to detect compared with two previous errors. This kind of error does not come from syntax error or inherited error but from initial design. Thus, this type of error has to be checked through general prospect. After implementing functions, functions are required to be checked whether they have met requirements or not.

The second type of testing is module testing(Walter & Just, 2013). After ensuring each method performs correctly, the next procedure is to ensure each module performs correctly. Ensuring each method to perform correctly cannot guarantee each module perform correctly. Sometimes although the method can perform correctly by itself, the conflict created by input or output parameters within two methods may leads to the collapse of the module. The module has to be tested as a whole.

The last type is to system testing (Walter & Just, 2013). System testing always involves testing for bottleneck during extreme workload. For example, HP LoadRunner can record scripts of action conducted at the website. HP LoadRunner launches hundreds or thousands of requests simultaneously to test whether the system will crash or not. In this thesis, since the system is hosted in the Google App Engine, simultaneous requests could be launched through pressure testing tools. At the same time, observation is conducted on the console of Google App Engine to check whether there is a bottleneck within the software system.

The objective of this thesis is to design a software model and implement it. We will focus on module testing in this thesis to ensure the system can run smoothly without crucial defects or bugs. Since hundreds of methods are programmed to support this thesis, method testing is not conducted given that it consumes considerable amount of time while time to finish this thesis is limited. System testing to find the bottleneck of the software system, if one exists, also requires programming scripts and huge experiment time. In the future, these two types of testing will need to be done to discover whether there are potential issues in this software system. At this stage, we admit the lack of these two types of testing as a limitation of this thesis.

The next chapter discusses software requirements for the Tour Navigation system and application.

Chapter 4 Software Requirements Detail

Detailing software requirements is the initial step of constructing the whole software system. Without requirements, the goal is unclear and the construction may be off track. In this chapter, requirements of the software system are defined through human natural language and use cases. As mentioned in the previous chapter; these two methods do have their own deficiencies but are able to complement each other to produce an effective system.

4.1 Requirements detailed by Natural Language

The general requirement of the software is:

By using Tour Navigation provided by this system, a user should be able to discover her or his current location within a building by scanning the QR code attached within the building, or through GPS signal when outdoors. In addition, a user should be able to get related tour information according to her or his current location as well.

As mentioned in chapter three, requirements can be defined in the format of "who should do something". Requirements, defined in this format should detail contents in following categories:

1. Requirements of user interface
2. Requirements of capability
3. Requirements of data
4. Requirements of quality
5. Requirements of constrains

With regard to requirements of user interface, requirements are listed as following:

1. The mobile app should display the weather information of current city.
2. The mobile app should indicate which city the user is currently in.
3. The mobile app should display the general introduction of the city using a user interface similar to that of Flipboard[1] to be attractive to users. Contents of introduction include text, pictures and videos.
4. The mobile app should display the nearby tour attraction buildings on the map.
5. The mobile app should display the introduction of the tour attraction buildings through gallery, video and text documents.

5a. Gallery should presents four pictures of the building.

5b. Video should present video introduction.

5c. Text should present brief text introduction.

6. Once the user is in the building, the mobile app should be able to display

[1] https://flipboard.com

pictures for navigation path.

7. The mobile app should provide the camera interface for scanning the QR code.

With regard to requirements of software system capabilities, requirements are listed as following:

1. The software system should be capable of updating weather information through Yahoo API.
2. The software system should be able to store introduction information of cities and provide the introduction information back to the mobile app.
3. The software system should be able to store the location and general introduction of the tour attractions. The contents of introduction include pictures, videos and text. These contents should be delivered back to the phone.
4. The software system should be able to store the structure of storeys of buildings and provide navigation service within these buildings.
5. The software system should be able to provide the introduction to the exhibited objects within buildings.

With regard to requirements of data format, requirements are listed as follows:

1. Images should be in JPG format.
2. Images should be stored in the project source repository. Once the source repository is uploaded to the cloud server, images can be reached through URL. URLs of pictures are stored in Google App Engine's Big Table.
3. Videos should be uploaded on to YouTube. URL of videos should be stored in Google App Engine's Big Table.
4. URL of contents of general introduction, including images and videos, and text should be stored in Google App Engine's Big Table.

With regard to requirements of design constraints, requirements are listed as follows:

1. The development should be conducted using the free cloud computing platform Google App Engine.
2. The testing should be conducted through inexpensive android devices.

With regard to requirements on quality, requirements are listed as follows:

1. The time duration between sending requests from mobile app and getting the response from cloud server should be within ten seconds.

4.2 Use Case Analysis

In the last chapter, use case analysis was introduced in the context of obtaining the software requirements. In this chapter, use case analysis is implemented and functional requirements are gained.

In this section, three main primary tasks conducted by the whole software system are defined as use cases for further use case analysis:

1. Users who just arrive in the city and hope to discover the general information of the city.
2. Users looking for nearby tour attractions
3. Users who have identified tour attractions which are worth sightseeing and hope to travel within the tour attraction, are introduced to the tour site and helped with navigation within the site's buildings

According to these three primary tasks defined in the above section, UML diagrams for the software system could be defined, as in Figure 4.1.

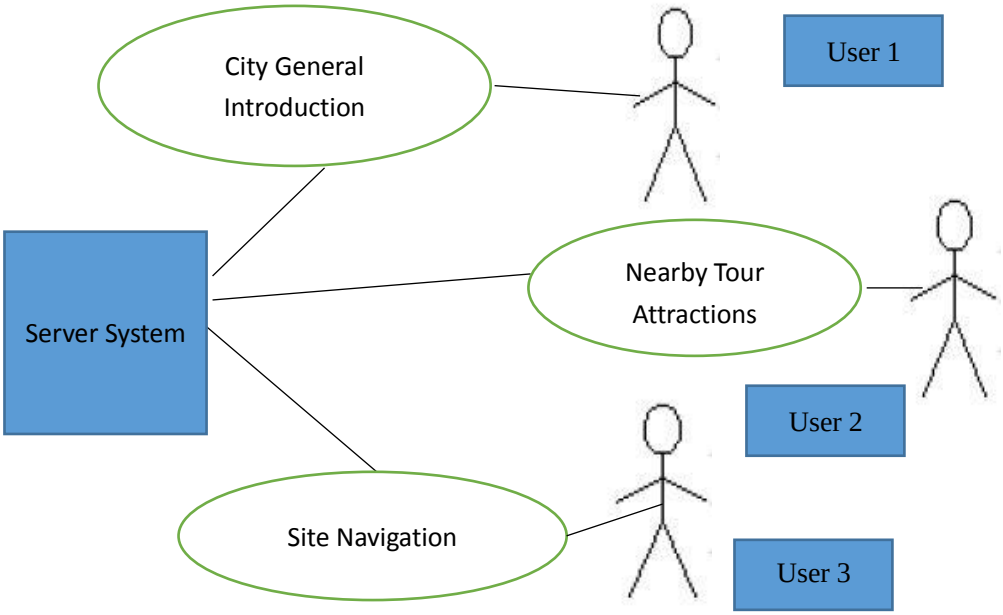


Figure 4.1 Overall system UML use case

The figure 4.1 shows the overall use cases of the system. Each use case can be detailed through other UML diagrams. In the following section, Figure 4.1 is divided into three sub UML diagrams.

The first use case is City General Introduction; it is broken down into two parts, namely Weather Forecast and Introduction.

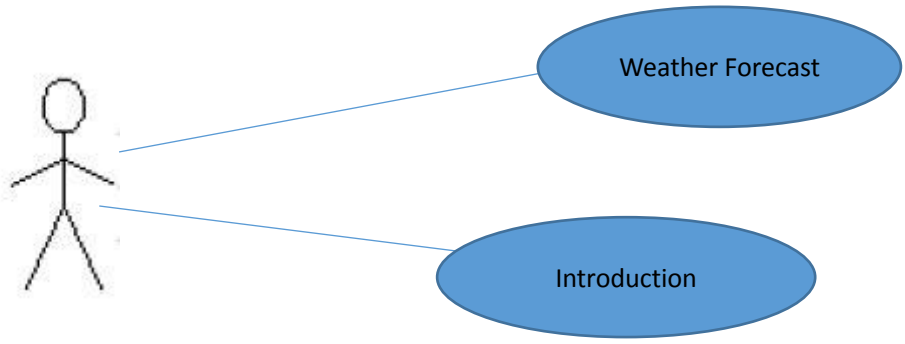


Figure 4.2 Use Case 1: City general introduction

As to the use case Weather Forecast shown in Figure 4.2, once the user launches the app Tour Navigation, the first page of the mobile app will indicate what the weather will be

of the city where the user is currently in. As to the use case Introduction, the information including pictures and text will be presented as flipped page. The link on the page could be clicked to reach videos uploaded on YouTube as well. The use case index of use case one is detailed as below:

Table 4.1 Use case index (Use Case One)

Use case ID	Use case name	Primary actor	Scope	Complexity	Priority
1	Weather Forecast	Generic user	In	Medium	1
2	Introduction	Generic user	In	Medium	1

The second use case Nearby Tour Attractions is broken down into two parts, namely Displaying Pins and Introduction of Sites.

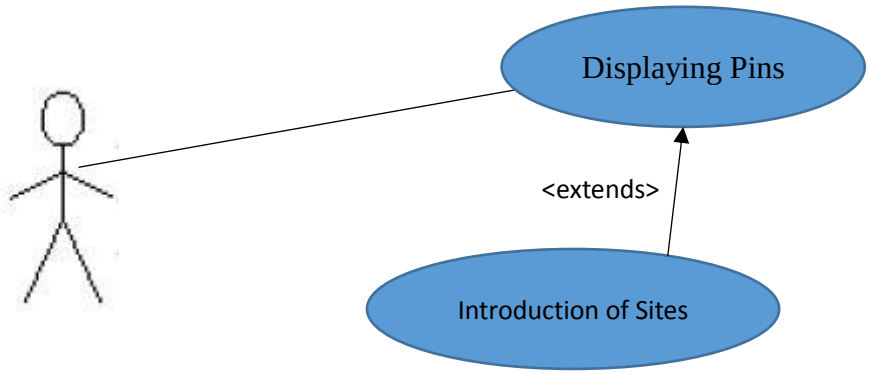


Figure 4.3 Use case 2: Nearby tour attractions

As to the use case Displaying Pins, once the app Tour Navigation gets the current location of the app users, it displays all pins of nearby tour attractions, which are stored in Google App Engine Big Table. As to the use case Introduction of Sites, once the app user clicks pins on the map, the app Tour Navigation will display contents of general introduction of the site. The use case index of use case 2 is detailed in Table 4.2.

Table 4.2 Use case index (Use Case Two)

Use case ID	Use case name	Primary actor	Scope	Complexity	Priority
1	Displaying Pins	Generic user	In	Medium	1
2	Introduction of Sites	Generic user	In	Medium	1

The third use case Building Navigation is broken down into three parts, namely Indoor

Navigation, Locating and Introduction of Exhibited Objects.

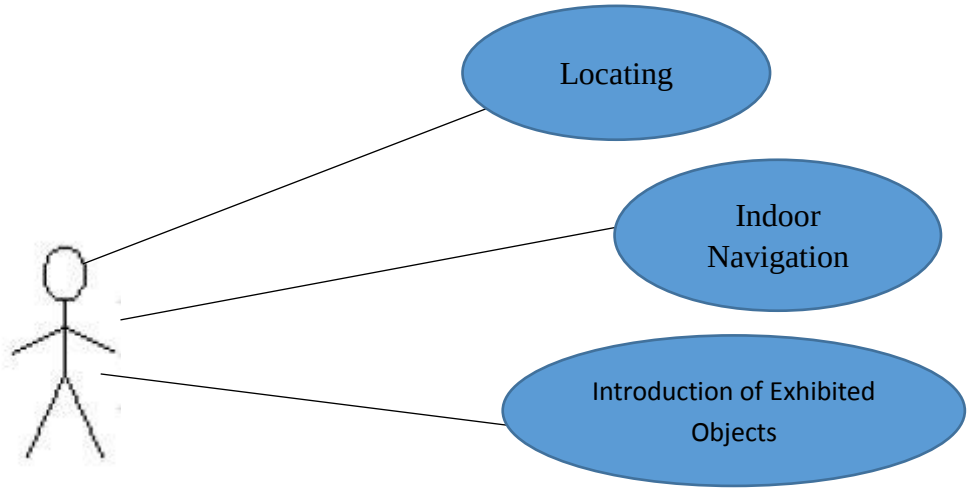


Figure 4.4 Use case 3：Site navigation

As to the use case Locating, once the app user goes into the building, the user can scan QR codes attached in the building, then the app Tour Navigation will report on the exact location the user currently is in. As to the use case Indoor Navigation, once the user chooses a certain destination which is stored in cloud server and returned for chosing they hope to reach, the application will show pictures to indicate the path to get to the destination. As to the use case Introduction of Exhibited Objects, once the user choose certain exhibited objects they wish to look at, the application will show pictures to indicate the path how to get to that object. In addition, introduction in the content of pictures, videos and text will be displayed to the user.

Table 4.3 Use case index (Use Case Two)

Use case ID	Use case name	Primary actor	Scope	Complexity	Priority
1	Indoor Navigation	Generic user	In	High	1
2	Introduction of Exhibited Objects	Generic user	In	High	1
3	Locating	Generic user	In	Medium	1

From above use case analysis, the following functional requirements are gained:

1. Weather forecast. Once the app user is in a certain city, the app gets the GPS signal to calculate which city the current user is in. The app Tour Navigation sends a request with city name as parameter through Yahoo API. It gets the response and displays the weather forecast on the app.

2. General Introduction of the City. Once the app gets the current city name, it sends a request to Google App Engine with the parameter of city name. The response contains the introduction of pictures, videos and text. As to the pictures, the server does not return the picture itself but the URLs. As to the videos, since the app engine server does not support live stream protocol, the server returns the URLs.
3. Displaying nearby tour attractions. Once the user is in a certain location, the app Tour Navigation sends a request containing current longitude and latitude as parameters to the server. The Google App Engine Big Table which stores the information of tour attractions searches nearby tour attractions and returns results. The app Tour Navigation generates pins of each tour attraction to display them on the map.
4. General Introduction of the tour attraction. Once the user clicks the pin on the map, the app Tour Navigation sends a request containing the tour attraction ID as parameter. According to this ID, the Google App Engine Server searches the Google Big Table and responds with general introduction, containing picture URLs, video URL and text information. Once this information is retrieved, the app displays them.
5. Navigation and Locating. Once the user enters the tour attraction site, the user scans the QR code. While the app Tour Navigation deciphers the QR code containing the vertex ID, the app Tour Navigation is informed of exact location the user is in. When the user chooses the destination site, the vertex ID of start and destination are sent to the Google App Engine server. The server calculates the shortest path and returns the URLs of pictures indicating the shortest path.
6. Introduction of Exhibited Objects. This function is quite similar to function 5. The only difference is when the user goes to the destination vertex, the introduction of exhibited object will be displayed.

Chapter 5 System Design

In this chapter, the system design is implemented according to MVC (Model-View-Controller) principle. To begin with, the architecture of the whole software system is detailed. In later section, the user interface of the whole software system and the structure of database are discussed.

5.1 MVC of the Software System

MVC stands for Model-View-Controller model. In this section, the system can be defined through these three aspects.

Views define each user interface component on the mobile app screen. In addition, the trigger events of certain UI components are also defined. Controllers are located in the Servlets, receiving requests from mobiles. Once certain events are triggered, for example, a button on the user interface is clicked, certain methods are executed by the controller to dispatch computing tasks to models for further processing. There are mainly two types of models in this thesis, namely business model and data model. Data models mainly define how to map between objects and each record in the Google Big Table. Business models are mainly responsible for receiving requests, taking data from data model, processing data and returning the result back to controllers. Figure 5.1 details the relationship between these three parties:

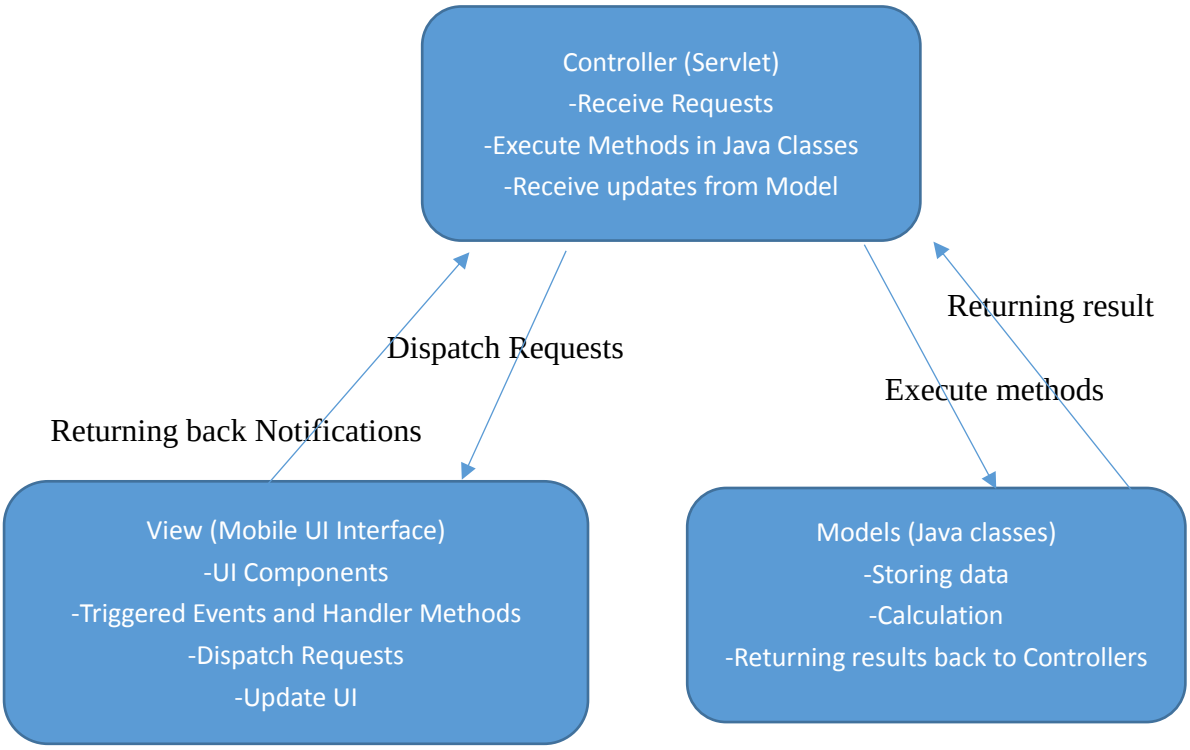


Figure 5.1 Relation between Model, View and Controller

Figure 5.2 illustrates the relationship between mobile and the Google App Engine server in a more general perspective.

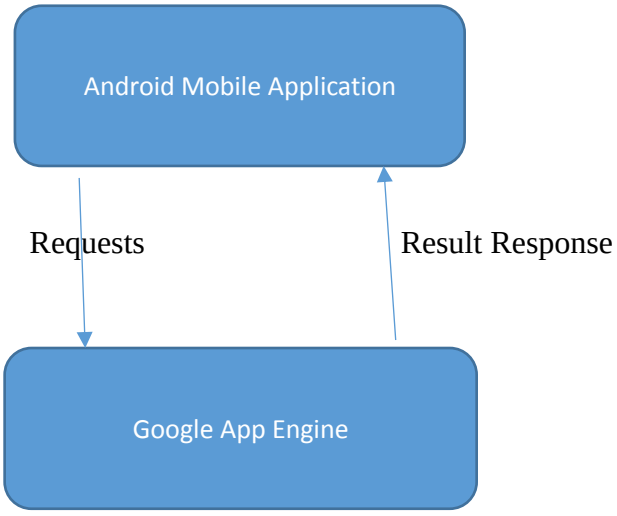


Figure 5.2 Relation between mobile and Google App Engine

The whole system could be divided into two parts namely the client and the server. The client side is run by Android application and the server side is run by Google App Engine. The data interaction between these two parties is run on the protocol of Http.

5.2 System Architecture

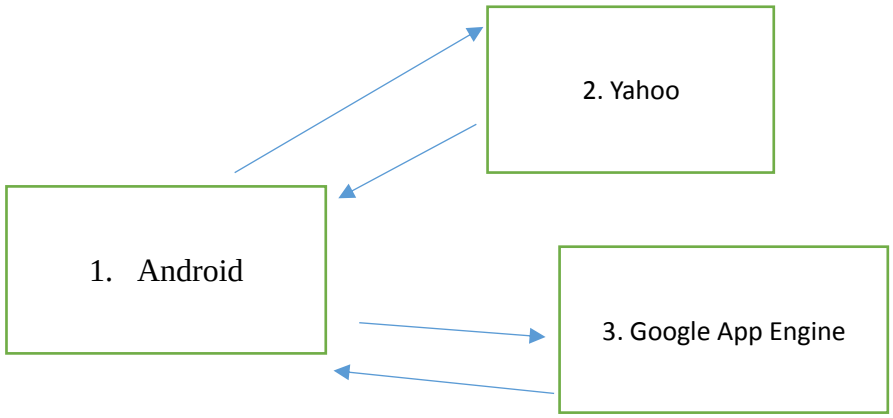


Figure 5.3 Modules relation

The whole system architecture is as detailed in the Figure 5.3: Modules are located in three major parts. The first part is modules in Android, for dispatching requests and handling responses. The second part is modules in Yahoo, for returning weather information. The last part is modules in Google App Engine, for navigation and

introduction.

Function Weather Forecast

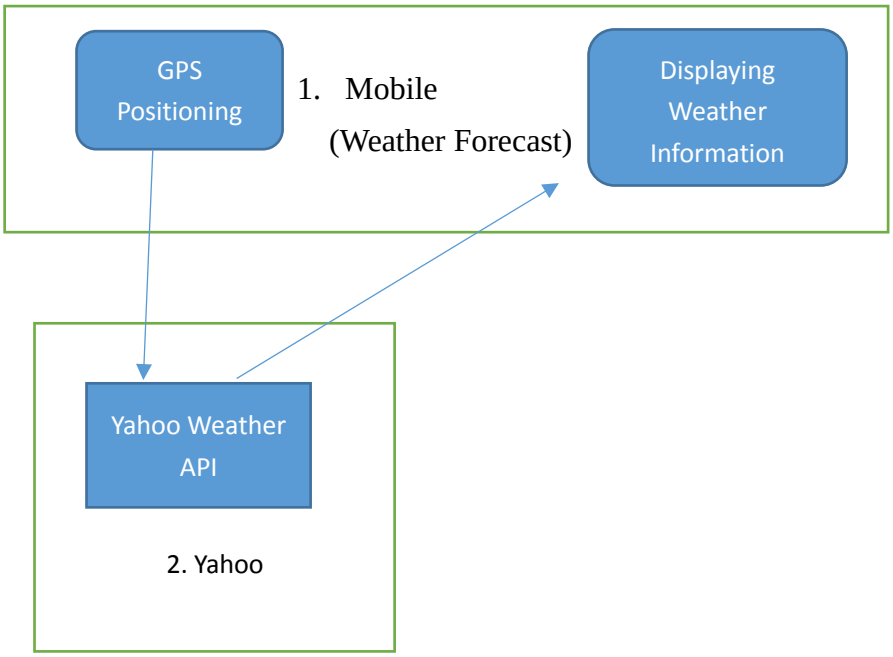


Figure 5.4 Modules in Weather Forecast

The modules that make up weather forecast part are shown in Figure 5.4. Following sections detail how the modules interact with each other.

In the GPS positioning module, when the GPS sensor is triggered temporary location information including latitude and longitude are received. The Location Listener attached with GPS is updated and the location information can be gained. Then the module dispatches requests containing location information to Yahoo Weather API module in Yahoo server. This module in Yahoo retrieves latest weather information wrapped in JSON Format. The information in JSON format is sent back to module Displaying Weather Information as response. Once the module Displaying Weather Information receives the data, it will display it on the screen.

As shown in Figure 5.5, modules in other functions of the whole software system, are tightly linked with Google App Engine.

Function General Introduction of City

The mobile phone, or mobile for short, utilizes the module GPS Positioning to locate where the mobile is then dispatches requests containing location information as parameters to Google App Engine's Big Table. Google Big Table takes two parameters namely longitude and latitude of current location of user. On receiving such a request this module queries the cloud database Google Big Table about city general introduction

according to the current location. Once general introduction information including picture URLs, video URLs and text introduction are received from Big Table, all this information wrapped in JSON format is sent to module Displaying City Information. This module will display this information in eye-pleasing UI style.

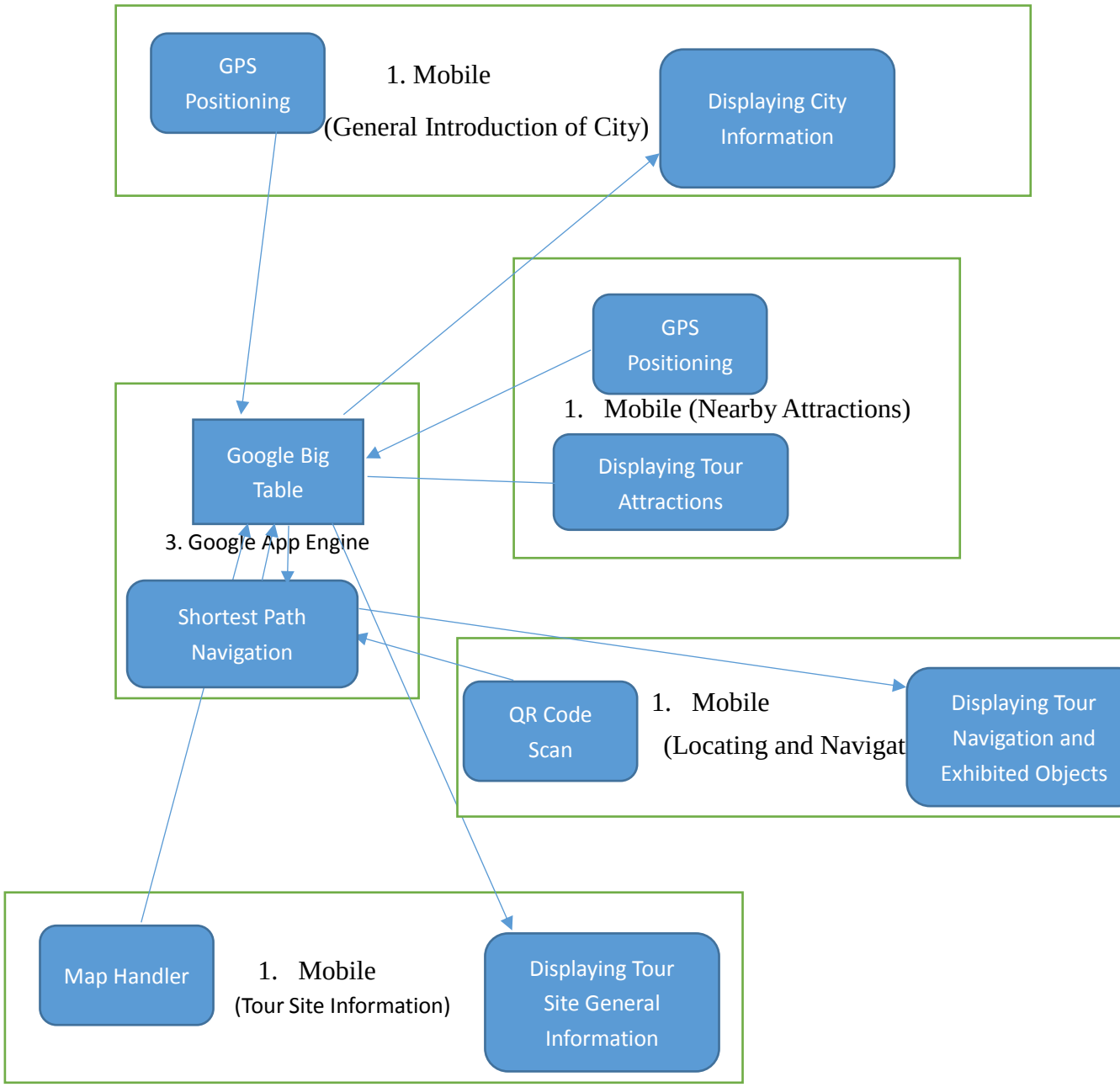


Figure 5.5 Other modules

Function Nearby Attractions

As shown in Figure 5.5, design pattern of Nearby Attractions is similar to the function Weather Forecast. At first module GPS Positioning gains the current location and the module dispatches the current location information as a request to Google Big Table.

The Google Big Table searches records about tourist attractions which are near the location provided by the request. Once results are calculated, the result will be dispatched back to the Displaying Tourist Attractions Module. In module Displaying Tourist Attractions, each record of one tourist attraction site is represented as one pin attached on Google Map for display to user.

Function Tour Site Information

If the user is looking for information about a certain tourist attraction site, what they need to do is to click the relevant pin on the map. Tour Site Information function takes over and the Map Handler, attached to the Google Map View, receives the touch gestures from users. Once the touch gesture is received, the Map Handler dispatches requests to Google Big Table to request general information of specific tour site including picture and video URLs and text description. Once the results are returned, the information is displayed on the user interface screen.

Function Locating and Navigation

In the function we are most interested in, that of locating and navigation, scanning of QR code is utilized. QR codes are attached in the building vertices to indicate current location. Each QR code contains the ID of the vertex. Mobile is used to scan the QR code. In module QR Code Scan, and a picture is taken of the QR Code. An open-source library[2] is utilized to decipher what ID the QR Code ciphers. Once the ID is known, this ID is sent to the module Shortest Path Navigation as a parameter. In the Module Shortest Path Navigation, all vertices are queried in Google Big Table. Each vertex has eight attributes to indicate whether there are adjacent vertices. Due to the existence of these attributes, the relations of vertices can be known. The shortest path algorithm is conducted to calculate the shortest path. In the final stage, the URL of pictures showing the path from start site to destination site are returned back as result. The module Displaying Tour Navigation and Exhibited Objects downloads pictures from returned URLs and displays the pictures to help the user navigate to destination.

[2] https://github.com/phishman3579/android-quick-response-code

As to exhibited objects, the module works in a similar manner. When the QR code near an exhibit is scanned, the URLs of pictures, videos and text description about the exhibited object are returned as a result, and displayed to user.

5.3 GUI Design

In this software system, the mobile screen is the only interface between human users and the whole software system. In this thesis, the mobile application design strictly follows the latest user interface design principle defined by Google, avoiding fragmentation of user interface design on Android apps (Morris, 2011). Before the guidance as suggested by Google, developers designed and implemented their application's user interface according to their own styles. Thus fragmentation of user interface design style became a barrier for the Android platform in its competition with IOS. Since the launch of Android 3.0, Google suggested certain conventions be followed such as the tool bar should be posted on the top and the navigation button should be at the top-left corner of the screen. The rest, i.e. the main content, should be placed on the major part of the screen (Jain, Bose, & Arif, 2013).

Figure 5.6 Weather Forecast

Figure 5.6 illustrates what the interface for weather forecast looks like. As the Android app UI design principle suggests, the main content should occupy the most proportion

of the screen, while other functions should be listed in the tool bar. Users are able to choose refresh, search for nearby tourist attraction sites, and get the latest general introduction in the tool bar. In the major component of the screen, detailed weather information can be seen by users, including what the current weather is, the highest temperature, and the lowest temperature.

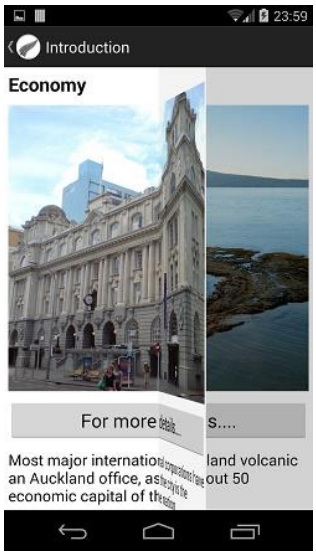


Figure 5.7 General Introduction of City

Figure 5.7 illustrates the interface of general introduction of city. The interface is in the shape of pages. The user swipes the screen from right to left in order to flip the page.

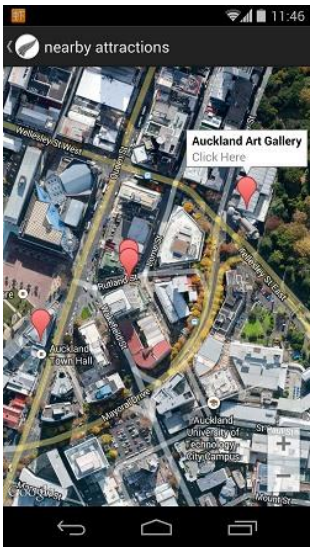


Figure 5.8 Tourist Attractions

Figure 5.8 illustrates the nearby tourist attractions. Users are able to click red pins to get

access to general introduction of nearby tourist attractions.

Figure 5.9 General Introduction of Tourist Attraction

Figure 5.9 illustrates the interface of general introduction of a tour attraction. Videos, pictures as well as text description are included.

Figure 5.10 QR Code Scan

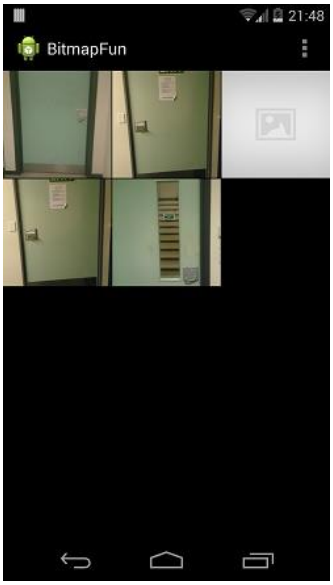


Figure 5.11 Path Navigation

Figure 5.10 illustrates that there is an image view for mobile phone's camera to scan the QR code. Once the QR code is scanned, the current location is deciphered and the location is identified. Later as Figure 5.11 illustrates, pictures are loaded showing the path to the destination.

5.4 Database Design

In this thesis, database Google Big Table is implemented on the Google App Engine. Nearly all data are stored in Big Table. Since there are limitations on capacity of stored objects and duration of connection with Google App Engine, big files such as pictures and videos are not stored directly in the Big Table. Only their ID and URL are stored as string while the picture and video files are stored in source code repository.

The table called City stores a city's ID, name, introduction text and picture URLs.

The table Tour Attractions stores the attraction name, ID, longitude, latitude, text description, URLs of four pictures and URL of Video.

The table Vertex stores Universal ID, ID, Story, Tour Attraction's ID and the URLs of pictures of eight directions of each vertex.

5.5 Design Decisions and Development Environment

Stored data format.

There are several reasons affecting what kind of format data is stored in: First of all, Google Big Table limits the time of connection. Therefore it is not possible to download large-size media files. Secondly, the Big Table limits the capacity of storage. Therefore, media file will not be stored directly in Big Table. Photos and videos are uploaded to source code repository or video website for access. Only the URL of these is stored in the Big Table as keys to access.

Development Environment.

The prototype is developed mainly using Java and Servlet. As to the server side, all the logic handlers are coded in Java and called by Servlet. All the Servlets are uploaded to the Google App Engine to handle requests. As to the mobile side, the app is developed in Java to send requests to Google App Engine and process responses from Google App Engine.

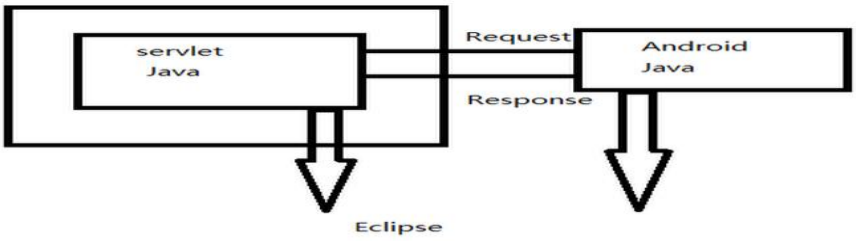


Figure 5.12 Development Environment

Hardware configuration and development tools.

The prototype is to be developed mainly on the ThinkPad R400— Windows 8, 64bit.

The specification of the hardware is:

CPU: Intel (R), Core (TM)2 Duo, CPU P8700, @2.53GHz

RAM: 2.00GB

The main development IDE is Eclipse.

Chapter 6 System implementation

In this chapter, details on how each module was implemented will be provided. Contents include what methods are implemented in each module, what data are transferred within these modules and how data is structured and stored.

6.1 Weather Forecast

This module implements the designed function Weather Forecast. In the module weather forecast, data transfer is only between mobile app Tour Navigation and Yahoo weather API.

In the first step, since information is delivered through internet, internet access permission needs to be claimed. Permission claims of INTERNET permission and ACCESS_NETWORK_STATE permission are both claimed in Manifest.xml as depicted in Figure 6.1 below:

```
<uses-permission  android:name="android.permission.INTERNET"
/>

<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Figure 6.1 Permission Claims

In the second step, methods of sending requests are defined. Three types of parameters can be sent, namely location detail string which can be deciphered by Yahoo Server, exact longitude and latitude as well as the GPS object of the smartphone.

In the third step, the method of how to handle returned weather information is defined. As previously discussed, since the internet traffic situation is unpredictable, the main thread is not the place for intensive time-consuming tasks as the user interface response and interaction may be blocked. Thus, sending requests and handling responses are asynchronised. Loading bar is loaded to indicate that weather information is being gained. Detailed weather information is returned from Yahoo Weather in XML format. Details include weather, humidity and the wind speed. Next, method of handling the returned weather information is called once the detailed weather information responses are received. In the last stage of loading, loading bar vanishes and the returned weather information is displayed.

6.2 City General Introduction

This module implements the designed function General Introduction of City. This module represents general model of how mobile app interacts with Google Cloud Server.

In this module, the very first step is to populate the Google Big Table with collected data. In this thesis, data only covers three major tour cities in New Zealand, namely Auckland, Wellington and Christchurch. The data structure of table design is detailed in Table 6.1:

Table 6.1 City general introduction

Attribute Name	**Data Type**	**Info**
ID	String	The Universal ID in Google Big Table
cityName	String	The Name of City
economyDescriptions	String	General Introduction of the Economy of the City
economyPicURL	String	The URL of Picture Used to Illustrate Economy of the City
economyURL	String	The URL of Text Description of Economy of the City
generalDescriptions	String	General Introduction of the City
generalPicURL	String	The URL of Picture Used to Illustrate the City
generalURL	String	The URL of Text Description of the City
geographyDescriptions	String	General Introduction of the History of the City
geographyPicURL	String	The URL of Picture Used to Illustrate the Geography of the City
geographyURL	String	The URL of Text Description of the Geography of the City
historyDescriptions	String	General Introduction of the History of the City
historyPicURL	String	The URL of Picture Used to Illustrate the History of the City
historyURL	String	The URL of Text Description of the History of the City
peopleDescriptions	String	General Introduction of the population of the City
populationPicURL	String	The URL of Picture Used to Illustrate the Population of the City
populationURL	String	The URL of Text Description of the Population of the City

As mentioned in chapter 5, Data models mainly define how to map between objects and each record in the Google Big Table. In the second step, the data model is designed to

transfer data in JSON format returned from the server to Java class object. To inject data into Google Big Table and for data management convenience, data is encrypted into parameters contained in a link in a web page first. When a link in the management page has been clicked, data in parameters can be transferred into JSON object and later transferred into Java class for storage in Google Big Table.

Once the data has been stored in Google Big Table, the data can be accessed through Http query. Again, since the Internet traffic situation is unpredictable, Asynchronous task is adopted to conduct this time-consuming task. In the Asynchronous Task, the Apache HTTP Component is utilized to maintain the low-level Java component focusing on Http connection. Once the connection is set up and data is returned, the data can be transferred from JSON format into the group of attributes that are needed. The attributes are the information the app needs.

A business model is needed to handle requests from the app. In this business model, requests are gained with parameters. Parameters which are taken as filters for the query play the role of identifying the records that should be taken from the Big Table.

In the app, the returned data is displayed in the page, in a format similar to Flipboard. The Flipboard interface is a pleasing one with pages that can be flipped/swiped. Flipboard attracts considerable number of users because of this reason(Crnkovic, 2010). An open-source project named android-flip in GitHub makes this user interface style available; this style is adopted in this thesis.

6.3 Nearby Tourist Attractions

This module implements the designed function Nearby Attractions. With regard to the module Nearby Tourist Attractions, the process of populating data in the Google Big Table and retrieving data from the Google Big Table is quite similar to the last module discussed. The difference is explained in this section. To begin with, the data structure design is different. The data table is designed as in Table 6.2.

Table 6.2 Nearby tour attraction

Attribute Name	Data Type	Info
ID	String	Universal Identification of Google Big Table
latitude	String	Latitude
longitude	String	Longitude
buildingId	String	The ID to Identify Tourist Site
buildingName	String	The Name of Tourist Site
description	String	General Description of Tourist Site
enterOrNot	Bool	Indicator to show whether Indoor Navigation Service will be Provided
imageURL1	String	URL of Illustration Picture One
imageURL2	String	URL of Illustration Picture Two
imageURL3	String	URL of Illustration Picture Three
imageURL4	String	URL of Illustration Picture Four
videoURL	String	URL of Illustration Video

The app sends requests, in the form of URLs, to servlets in Google App Engine to gain information of nearby tourist attraction sites, The URL contains parameters of latitude and longitude. For example, the following URL stands for request sent to query_map servlet with parameters of longitude and latitude:

```
query_map?latitude=-36.85302&longtitude=174.76345
```

Tourist attractions that are within a 1-kilometer radius of the location will be searched. In the servlet, -36.85302 and 174.76345 are taken as parameters. These two parameters are added to filters for search.

Once data of tourist attractions is gained, the data is wrapped as polygon shown on the map view of the app. The polygon is displayed as a pin. Once the user clicks a pin, a click event is triggered. A method handling this click event will display tourist attraction information.

6.4 Tourist Attraction Site Information

This module implements the designed function Tour Site Information. This module is

designed and implemented following the same principles as the previous modules. Firstly, Google Big Table is populated with introduction data including text introduction, picture and video URLs. Once the app sends requests with the ID of specific tourist attractions to handling servlets in Google App Engine, servlets conduct search in Google Big Table. Results are retrieved from Google Big Table and wrapped in JSON Format to transfer back to the app. The app translates these data first and displays them all on the screen.

6.5 Locating and Navigation

This module implements the designed function Locating and Navigation. Navigation and locating the current position of user in the building are the major focus of this thesis.

As to the navigation, the following steps are conducted to make this service available:

1. Populate the Google Big Table with data of storeys within a building.
2. Design and implement navigation algorithm in business model, located in Google App Engine's servlet. This is discussed later in this section

As to the first step, data structure design is listed in Table 6.3. Vertices were explained in chapter 4.2.

Table 6.3 Vertices

Attribute	**Data Type**	**Description**
ID/Name	String	Universal ID of Google Big Table
buildingId	String	ID of Building Vertex Belongs to
counjunctionId	String	With Prefix of buildingId, ID of Vertex
directionOneUrl directionTwoUrl directionThreeUrl directionFourUrl directionFiveUrl directionSixUrl directionSevenUrl directionEightUrl	String	The URL of Pictures showing Eight Directions when Users Standing at that Vertex
neighbourOneId neighbourTwoId	String	The ID of eight adjacent vertices.

neighbourThreeId neighbourFourId neighbourFiveId neighbourSixId neighbourSevenId neighbourEightId		

How vertices are linked with each other is discovered from the interior structure of the building. In this thesis, only critical vertices that contribute significantly to searching path are included. According to these relationships, vertices are linked with each other to knit the map of the whole storey. There are IDs of eight directions of each vertex to indicate which vertices are adjacent to the current vertex. The ID of start vertex and destination vertex are the parameters for calculating the shortest path between these vertices.

Eight photos of eight directions are taken for each vertex. These photos are stored in the project repository but not the Google Big Table, given the problem of limited-time connection with Google Big Table and limited storage capacity of the server. Since URLs are strings, URLs are stored in the Google Big Table for further access.

Calculating the shortest path is conducted in the business model. To calculate the shortest path, first of all, an adjacency list is constructed for describing vertex relationship in each storey. Adjacency list consists of vertices; all vertices that are adjacent to a vertex are listed in the adjacency list. Each vertex has its own adjacency list. Thus, how vertices are linked with each other is described.

In the business model, breadth first search is utilized to search for the shortest path. At first, the start vertex and its adjacent vertices from the adjacency list are put into the queue. When the very first vertex which enters the queue is taken out, it is judged whether it is the destination. If not, this vertex is recorded down as one part of shortest path and all its all adjacent vertices are pushed, until the destination vertex is discovered. If it is destination, the shortest path to this vertex is the shortest path to the destination.

After the shortest path has been calculated, each vertex on the route can be gained. The

relationship, namely which vertex is located adjacent to other vertex is gained. According to the direction one vertex compared to another vertex, URLs of pictures are returned. Once these URLs are gained, pictures showing navigation path are shown.

Since a considerable number of pictures are transferred from Google App Engine to the app and each picture occupies certain amount of memory, it can be a challenge for the app to display them efficiently. In this thesis, techniques utilized are memory caching and disk caching. For the memory cache, a class named LruCache is utilized. In this thesis, 12.5% of total memory is used as cache. However, there is still the possibility that the memory is exceeded. For such cases, disk cache is utilized for storing pictures. Another class, diskLruCache is utilized for caching pictures on disk.

The technique of scanning QR code is adopted to locate current position of user. To gain the ID of the start vertex, the module in the app translates the QR Code to gain the deciphered information of the QR code. In this thesis, the open-source library QRCodeView[3] in GitHub is adopted. In the app, the QR code scanning activity implements the predefined methods, including method handling the event when QR Code is successfully deciphered, handling the exception event when camera is not found or when QR Code is not found in the image which is taken. Relevant code is triggered when relevant events occur. In this scenario, when QR Code is successfully deciphered, vertex ID is taken to be sent to business model for calculating the shortest path between this vertex and chosen destination vertex ID.

6.6 Exhibited Objects

Exhibited objects are introduced based on the vertex. Since exhibited objects are located near vertices, when users navigate to certain vertices the introduction to specific exhibited objects will be displayed.

Data interaction between the app and Google App Engine is similar to the last module

[3] https://github.com/SkillCollege/QrCodeScan

discussed. The data structure of Exhibited Object is listed in Table 6.4:

Table 6.4 Exhibited Object

Variable	**Data Type**	**Information**
ID/Name	String	The Universal Identifier of Google Big Table
buildingId	String	Which Tourist Site this Exhibited Object is located
imageURL1	String	The URL of Illustration Picture One
imageURL2	String	The URL of Illustration Picture Two
imageURL3	String	The URL of Illustration Picture Three
imageURL4	String	The URL of Illustration Picture Four
objectDescription	String	Text Introduction
objectId	String	The ID of object, with prefix of ID of Vertex
objectIntroductionVideoURL	String	The URL of Illustration Video
objectLocationId	String	The ID of Related Vertex
objectName	String	Name

Chapter 7 System Testing and Demonstration

After designing and implementing the software system, in this chapter we look at testing the whole system, by testing the modules discussed in last chapter. Each module is tested to ascertain whether it is performing correctly or not. In addition, screenshots of the mobile app and Google App Engine administration web pages are provided to demonstrate the availability of the whole system.

7.1 Weather Forecast

When the user clicks the app icon, the app presents the welcome page which indicates what this app is generally aiming for.

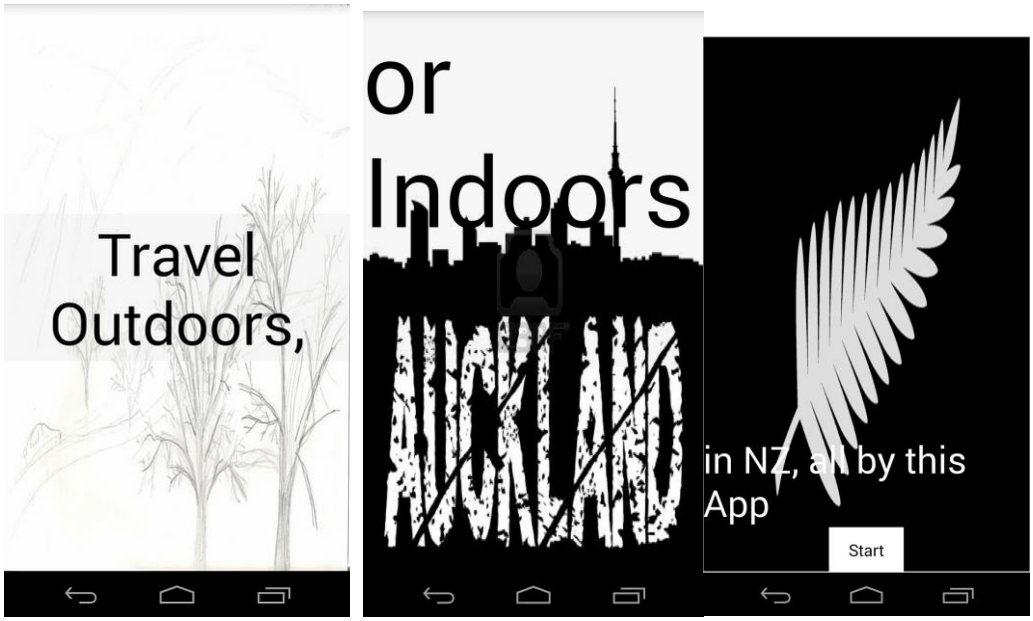


Figure 7.1 Welcoming page

After the welcoming page, the app shows the loading bar (Figure 7.2) while the app downloads the data from the Yahoo Weather API.

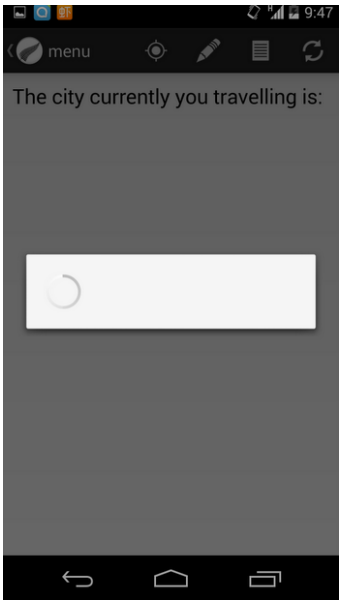


Figure 7.2 Loading bar

After successfully downloading the data, the data is presented to the user, as in Figure 7.3.

Figure 7.3 Weather forecast

7.2 City General Introduction

While testing the module General Introduction of the City, testing is conducted in two parts. In the first part, the test to see whether raw data is returned successfully and the second part to test the resulting user interface display.

After a request has been sent to Google App Engine with the parameter ‘City name’, the returned raw data contains general information of the city which is queried from Google Big Table. All the result is written on the servlet page and can be used for further analysis.

The request URL is sent:

http://tournavigationlijunhao.appspot.com/query_citygeneral?cityName=Auckland

The returned data information is as expected and is shown in Figure 7.4.

1. Most major international corporations have an Auckland office, as the city is the economic capital of the nation.
2. http://upload.wikimedia.org/wikipedia/commons/7/7e/Britomart_Outside_Facade.jpg
3. http://en.wikipedia.org/wiki/Auckland#Economy
4. Auckland straddles the Auckland volcanic field, which has produced about 50 volcanoes.
5. http://upload.wikimedia.org/wikipedia/commons/5/5f/Rangitoto_from_Achilles_Point.jpg
6. http://en.wikipedia.org/wiki/Auckland#Geography_and_climate
7. The isthmus was settled by Mori around 1350 and was valued for its rich and fertile land.
8. http://upload.wikimedia.org/wikipedia/commons/2/20/Lower_Queen_Street.jpg
9. http://en.wikipedia.org/wiki/Auckland#History
10. Auckland is home to many cultures.
11. http://upload.wikimedia.org/wikipedia/commons/b/b2/Helen_Clark_welcomed_to_Hoani_Waititi_Marae_2006-02-06.jpg
12. http://en.wikipedia.org/wiki/Auckland#People

Figure 7.4 Returned general introduction data

The raw data is returned successfully.

The first record of the raw data is description of the City's economy.

The second record of the raw data is the URL of illustration picture of the economy. The third record of the raw data is URL of long text description of Auckland's economy.

The fourth record of raw data is description of Auckland's geography.

The fifth record of raw data is URL of description picture of geography.

The sixth record of raw data is URL of long text description of geography.

The seventh record of raw data is description of history in Auckland.

The eighth record is the URL of illustration picture of history of Auckland.

The ninth record is the URL of long text description of history.

The tenth record of raw data is culture description of Auckland.

The eleventh record is the URL of illustration picture of culture of Auckland.

The twelfth record is the URL of long text description of culture.

City general introduction data covers three cities, namely Auckland, Wellington and Christchurch in Google Big Table. The test was conducted and the testing result is presented in the following table. All data about the three cities are successfully returned after dispatching the request.

Table 7.1 City general introduction testing result

City	**Return raw data successful?**
Auckland	Yes
Wellington	Yes
Christchurch	Yes

The second part testing is to test whether data returned is successfully represented in Flipboard similar page in the mobile application. After testing, all raw data is successfully loaded in the mobile page. Screenshots of the mobile application pages that can be flipped are shown in Figure 7.5.

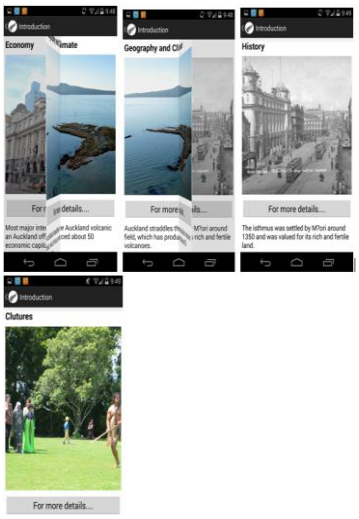


Figure 7.5 City general description

When the user clicks the ‘For more details’ button, the user is successfully directed to a Wikipedia webpage for further detailed information, as shown in Figure 7.6

Figure 7.6 Long description

7.3 Nearby Tourist Attraction

The same approach as adopted in section 7.2 is adopted testing this module as well. At first, we tets whether Google App Engine can return raw data about nearby tourist attraction sites, accordingly, a request is sent with the parameter of specific location:

http://tournavigationlijunhao.appspot.com/query_map?latitude=-36.85302&longtitude=174.76345

The returned data is:

```
Auckland Town Hall

2

-36.8529

174.7634

AUT Tower

1

-36.8521

174.7646
```

Figure 7.7 Nearby tour attraction data

The raw data contains the name, ID, longitude and latitude of the tourist information.

The server is able to return the data successfully.

Various sets of combinations of latitude and longitude are also input to be tested:

Table 7.2 Nearby tour attraction testing result

Longitude	Latitude	Return raw data successful?
-36.85302	174.76345	Yes
-36.86302	174.77345	Yes
-36.87302	174.78345	Yes
-36.88302	174.79345	Yes
-36.89302	174.80345	Yes

The second part is to test whether raw data is successfully presented on the mobile application.

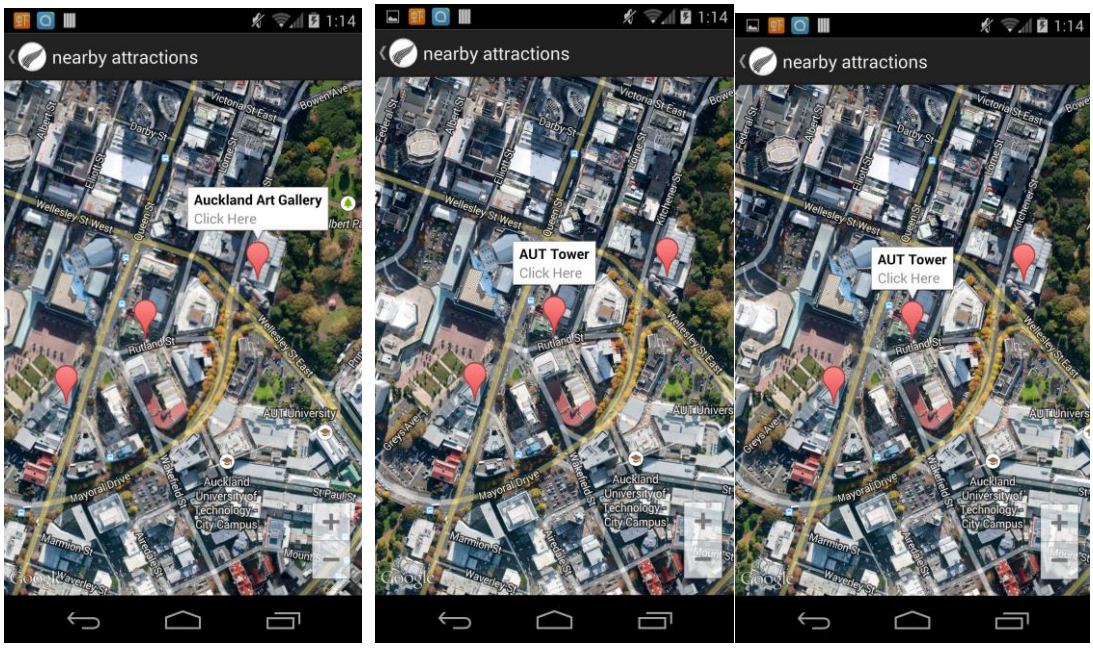


Figure 7.8 Polygons

The data is displayed in the format of polygon successfully.

7.4 Building Description

Similar to the last two modules, testing is conducted in two parts. In the first part, Google App Engine receives the request being part of the test of whether it is able to return the building description data or not. The request url is sent:

http://tournavigationlijunhao.appspot.com/query_specificbuilding?buildingId=1

The raw data is returned successfully, as shown in Figure 7.9.

```
http://tournavigationlijunhao.appspot.com/images/buildingDescr
iption/1/1_1.jpg
http://tournavigationlijunhao.appspot.com/images/buildingDescr
iption/1/1_2.jpg
http://tournavigationlijunhao.appspot.com/images/buildingDescr
iption/1/1_3.jpg
http://tournavigationlijunhao.appspot.com/images/buildingDescr
iption/1/1_4.jpg
http://www.youtube.com/embed/BkmrTS3DZE8
true
```

Figure 7.9 Building Description Data

The first four URLs in Figure 7.9 are tourist site's illustration pictures' URLs. The Fifth url is the URL for a video. The last bool value indicates whether the tourist site provides indoor navigation service or not.

Three building Ids are input as part of the test:

Table 7.3 Building introduction testing result

Building Id	**Return raw data successful?**
1	Yes
2	Yes
3	Yes

In the second part of testing, the raw data is succesfully presented in the app, as shown in Figure 7.10.

Figure 7.10 Building description

7.5 Exhibited object introduction

Similar with last modules, exhibited object introduction is tested in two modules. In the first module, the Google App Engine is tested whether the server can return the description raw data of exhibited object or not.

The request URL is sent:

http://tournavigationlijunhao.appspot.com/query_exhibitedobject?buisdldingId=2

The raw data is returned successfully:

```
http://tournavigationlijunhao.appspot.com/images/exhibitedObjects/1/4/4-1.jpg
http://tournavigationlijunhao.appspot.com/images/exhibitedObjects/1/4/4-2.jpg
http://tournavigationlijunhao.appspot.com/images/exhibitedObjects/1/4/4-3.jpg
http://tournavigationlijunhao.appspot.com/images/exhibitedObjects/1/4/4-4.jpg
this is wt 406. Which specific for master thesis room
http://www.youtube.com/watch?v=4FgwvximJw4&list=UUAhC-ufgtoTQHTjSuYrGc9w&feature=c4-overview
wt406
1-4-4-1
```

Figure 7.11 Exhibited object description data

The first line of raw data represents display picture one of the exhibited object. The second line of raw data represents display picture two of the exhibited object. The third line of raw data represents display picture three of the exhibited object. The fourth line of raw data represents display picture four of the exhibited object. The fifth line of raw data stands for text description of the exhibited object. The sixth line of raw data stands for the URL of introduction video. The seventh line of raw data stands for the name of the object. The last line of the raw data stands for the Id of the object.

The second part of testing is to test whether the raw data is displayed on the mobile application.

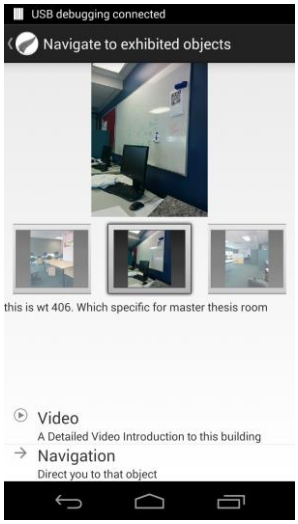


Figure 7.12 Exhibited object

The raw data is successfully displayed on the mobile application.

7.6 Shortest path navigation

Similar to previously explained modules, shortest path navigation is tested in two parts. In the first module, the Google App Engine is tested whether the server can return shortest path pictures' URLs.

To show the path from vertex of ID 1_5_9 to vertex of ID 1_4_0, the request URL is：

http://tournavigationlijunhao.appspot.com/query_conjunction?start=1_5_9&destination=1_4_0

The returned raw data is：

```
http://tournavigationlijunhao.appspot.com/images/1/5/9-7.jpg,
http://tournavigationlijunhao.appspot.com/images/1/5/8-1.jpg,
http://tournavigationlijunhao.appspot.com/images/placeholder.jpg,
http://tournavigationlijunhao.appspot.com/images/1/4/0-7.jpg,
http://tournavigationlijunhao.appspot.com/images/1/4/1-3.jpg
```

Figure 7.13 URLs of shortest path pictures

The URL's array stands for the navigation path pictures' URLs. Different combination of start conjunction and destination conjunction are tested.

Table 7.4 Navigation path testing result

Start	Destination	Returned shortest path successfully?
1_5_9	1_4_0	Yes
1_4_0	1_5_9	Yes
1_5_9	1_4_4	Yes
1_4_4	1_5_9	Yes
1_5_9	1_4_5	Yes
1_4_5	1_5_9	Yes
1_5_9	1_4_7	Yes
1_4_7	1_5_9	Yes

As to testing the display on the app, the picture is successfully shown, as in Figure 7.14.

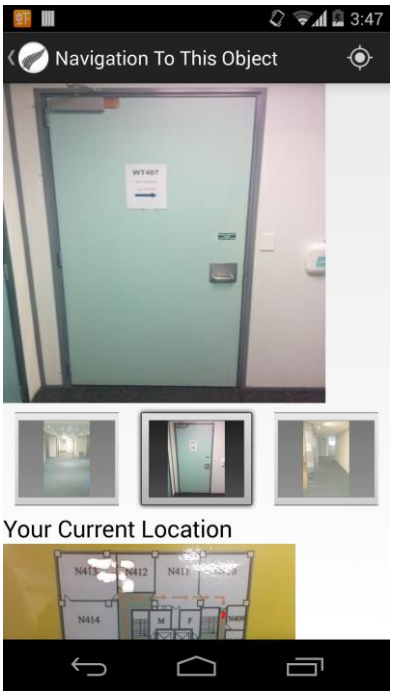


Figure 7.14 Navigation pictures

Chapter 8 Conclusion

This thesis gives a critical thought of what tourist navigation system should be, and how the system was designed and what steps were conducted to implement the whole system, according to the constructive methodology. In the final stage, demonstration is conducted to show the availability of the whole system. In this final chapter, we draw the conclusion of this thesis. Limitations and what work can be done in the future are detailed.

Conclusion

This research thesis displays an innovative cloud based tourist navigation system, following the principles of software development. It implements the objects of this thesis research, which include setting up the navigation server on cloud computing platform and providing navigation service to users through the smartphone application. Recalling the original thesis questions, they are: How smartphones guide tourists to travel in New Zealand with the help of smartphones or tablets, no matter they are in outdoor or indoor environments? And how can smartphones be supported by the cloud computing platform? Since the system has been implemented, these thesis questions have been addressed successfully.

In addition, it has also successfully implemented the requirements detailed by natural language and use cases. After completing the whole procedures of implementing the whole software system, the main contribution of this thesis includes:

1. The prototype software system is constructed on the cloud platform, providing navigation services to tourists steadily. Cloud based tour navigation system has never been constructed before. This thesis illustrates the supportability of this proposal, has implemented such a system, tested it and demonstrated it..
2. Data introducing tourist sites, navigation and locating user position are stored in the cloud.
3. The prototype enables users to get tourism information through the smartphone connecting to cloud platform. Previous tour-related mobile applications are mostly content-fixed. Now contents of the navigation and introduction can be updated according to the environment the user is in.
4. The prototype directs users no matter they are indoors or outdoors. Previous navigation systems are limited either to the indoor environment or the outdoor environment.
5. The prototype enables users to locate themselves by scanning QR code, which saves the expense of additional hardware and improves the accuracy of locating position.

However, this prototype is far from perfect, although it proves the possibility of constructing cloud service to provide indoor navigation service.

1. The service is only available on the Android platform, the service on other smartphone platforms is not available yet.
2. Tourism raw data collected was for the purposes of proof of concept and is not sufficient and correct enough. Since all data is collected by an individual, the objectivity cannot be guaranteed and the data may be biased. City general introduction only covers three cities. Building introduction only covers three sites in Auckland. Indoor navigation only covers two floors in one specific site.
3. The whole system is constructed on a trial, prototype version but not commercial version. Whether it can handle overwhelming data and perform servicing efficiently has not been tested.
4. The system is not fully tested. As mentioned earlier, due to time constraints, only module testing of the software system is conducted.

Based on the limitations mentioned here, some points to be considered for future work are:

1. The mobile app can be constructed using Phonegap technology; multiple platforms are covered with only one-time development. In addition, with respect to mobile devices, mobile web is another alternative technical solution that can be used to cover different smartphone platforms.
2. Data could be filtered and collected by tourism specialists. In future, more data could possibly be collected to cover tourism of the whole of New Zealand to make this system applicable to real life.
3. Upgrade the system to financial version. Financial versions of apps deployed on the Google App Engine are built to handle huge number of requests more efficiently. What is more, capacity for storing data is large. Load testing can be conducted on the system in the future to test whether it is robust enough to handle huge number of requests simultaneously.

4. Other types of testing are also needed to be performed. In addition to module

testing, method testing, system testing and other types of testing need to be carried out to ensure there are no crucial defects.

References

Alessio, D. (2010). Travel, tourism and booster literature: New Zealand's cities and towns at the turn of the twentieth century. *Studies in Travel Writing, 14*(4), 383-396. doi: 10.1080/13645145.2010.522811

Alghamdi, S., Van Schyndel, R., & Alahmadi, A. (2013, 2-5 April 2013). *Indoor navigational aid using active RFID and QR-code for sighted and blind people.* Paper presented at the Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on.

Andrew, W. (2011). Blurring the boundaries between our physical and electronic libraries. *The Electronic Library, 29*(4), 429-437. doi: 10.1108/02640471111156713

Android. (2014). Displaying Bitmaps Efficiently. from https://developer.android.com/training/displaying-bitmaps/index.html

Ayre, L. B. (2012). RFID Costs, Benefits, and ROI. *Library Technology Reports, 48*(5), 17-19.

Binkley, D., Feild, H., Lawrie, D., & Pighin, M. (2009). Increasing diversity: Natural language measures for software fault prediction. *Journal of Systems and Software, 82*(11), 1793-1803. doi: http://dx.doi.org/10.1016/j.jss.2009.06.036

Bisio, I., Lavagetto, F., Marchese, M., & Sciarrone, A. (2013). GPS/HPS-and Wi-Fi Fingerprint-Based Location Recognition for Check-In Applications Over Smartphones in Cloud-Based LBSs. *Multimedia, IEEE Transactions on, 15*(4), 858-869. doi: 10.1109/TMM.2013.2239631

Chang-Jie, M., & Jin-Yun, F. (2008). Location-based mobile tour guide services towards digital dunhuang. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 37*(B4), 949-953.

Correa, J. D. Y., & Ricaurte, J. A. B. (2014). Web Application Deveplopment Technologies Using Google Web Toolkit And Google App Engine-Java. *Latin America Transactions, IEEE (Revista IEEE America Latina), 12*(2), 372-377. doi: 10.1109/TLA.2014.6749559

Crnkovic, G. (2010). Constructive Research and Info-computational Knowledge Generation. In L. Magnani, W. Carnielli & C. Pizzi (Eds.), *Model-Based Reasoning in Science and Technology* (Vol. 314, pp. 359-380): Springer Berlin Heidelberg.

DarrelInce. (2010). *JSON*: 'Oxford University Press'.

Delev, T., Gjorgjevik, D., & Madzarov, G. (2010, 21-24 June 2010). *Place-Tags, discovering and promoting places through mobile phones and collaborative filtering.* Paper presented at the Information Technology Interfaces (ITI), 2010 32nd International Conference on.

Durresi, M., Luarasi, T., Baholli, I., & Durresi, A. (2013, 4-6 Sept. 2013). *Targeted Advertisement Using Smartphones and Cloud Computing.* Paper presented at the Network-Based Information Systems (NBiS), 2013 16th International Conference on.

Golda, A. F., Aridha, S., & Elakkiya, D. (2009, 22-24 July 2009). *Algorithmic agent for effective mobile robot navigation in an unknown environment.* Paper presented at the Intelligent Agent & Multi-Agent Systems, 2009. IAMA 2009. International Conference on.

Google. (2014a). Getting Started. *Google Map API.* from https://developers.google.com/maps/documentation/android/start#getting_the_google_maps_android_api_v2

Google. (2014b). Using JSPs. 2014, from https://developers.google.com/appengine/docs/java/gettingstarted/usingjsps

Goossens, C. F. (1995). External Information Search. *Journal of Travel & Tourism Marketing, 3*(3), 89-107. doi: 10.1300/J073v03n03_06

Gozick, B., Subbu, K. P., Dantu, R., & Maeshiro, T. (2011). Magnetic Maps for Indoor Navigation.

Instrumentation and Measurement, IEEE Transactions on, 60(12), 3883-3891. doi: 10.1109/TIM.2011.2147690

Hansen, J., Grønli, T.-M., & Ghinea, G. (2012). Towards Cloud to Device Push Messaging on Android: Technologies, Possibilities and Challenges. *International Journal of Communications, Network and System Sciences, 5*(12), 839-849.

Hinch, S. W. (2010). *Outdoor navigation with GPS*. Birmingham, Ala U6 - ctx_ver=Z39.88-2004&ctx_enc=info%3Aofi%2Fenc%3AUTF-8&rfr_id=info:sid/summon.serialssolutions.com&rft_val_fmt=info:ofi/fmt:kev:mtx:book&rft.genre=book&rft.title=Outdoor+navigation+with+GPS&rft.au=Hinch%2C+Stephen+W&rft.date=2010-01-01&rft.pub=Wilderness+Press&rft.externalDocID=1549198&paramdict=en-US U7 - eBook U8 - FETCH-aut_catalog_15491983: Wilderness Press.

Holleman, P. J. (2012). Android Parking... Allensworth, Alder, " do you know...?. Parking Solutions", PN 2012 Feb; 66(2): 44-5. *PN, 66*(4), 11-11.

Hsu, H.-H., & Liao, H.-T. (2011). A mobile RFID-based tour system with instant microblogging. *Journal of Computer and System Sciences, 77*(4), 720-727. doi: http://dx.doi.org/10.1016/j.jcss.2010.02.011

Ibrahim, D., & Ibrahim, A. (2010). Real-time GPS based outdoor WiFi localization system with map display. *Advances in Engineering Software, 41*(9), 1080-1086. doi: 10.1016/j.advengsoft.2010.06.005

Iozan, L. I., Collin, J., Takala, J., & Rusu, C. (2011). INERTIAL INDOOR NAVIGATION SYSTEM. *Acta Technica Napocensis, 52*(1), 47-50.

Jain, R., Bose, J., & Arif, T. (2013, 2013). *Contextual adaptive user interface for Android devices*.

Jarle, H., Tor-Morten, G., & Gheorghita, G. (2012). Towards Cloud to Device Push Messaging on Android: Technologies, Possibilities and Challenges. *International Journal of Communications, Network and System Sciences, 5*(12), 839-849.

Jordan, L., & Greyling, P. (2011). Using the Google App Engine with Android *Practical Android Projects* (pp. 275-310): Apress.

Joshi, D. S., Sridhar, R., & Chandrasekharan, N. (1993, 27-29 May 1993). *Efficient algorithms for all-pairs shortest path problem on interval, directed path, and circular-arc graphs.* Paper presented at the Computing and Information, 1993. Proceedings ICCI '93., Fifth International Conference on.

Kanyaru, J., & Phalp, K. (2009). Validating software requirements with enactable use case descriptions. *Requirements Engineering, 14*(1), 1-14. doi: 10.1007/s00766-008-0070-8

Kenteris, M., Gavalas, D., & Economou, D. (2009). An innovative mobile electronic tourist guide application. *Personal and Ubiquitous Computing, 13*(2), 103-118. doi: 10.1007/s00779-007-0191-y

Kingston, D. G., Eastwood, W. J., Jones, P. I., Johnson, R., Marshall, S., & Hannah, D. M. (2012). Experiences of using mobile technologies and virtual field tours in Physical Geography: implications for hydrology education. *Hydrol. Earth Syst. Sci., 16*(5), 1281-1286. doi: 10.5194/hess-16-1281-2012

Krishna Mohan, K., Verma, A. K., Srividya, A., & Papic, L. (2010). INTEGRATION OF BLACK-BOX AND WHITE-BOX MODELING APPROACHES FOR SOFTWARE RELIABILITY ESTIMATION. *International Journal of Reliability, Quality and Safety Engineering, 17*(03), 261-273. doi: doi:10.1142/S0218539310003792

Lamb, A., & Johnson, L. (2013). QR Codes in the School Library: A Dozen Practical Uses. *Teacher Librarian, 40*(3), 63-67,71.

Leone, A., & Chen, D. (2007). Implementation of an object oriented data model in an information system for water catchment management: Java JDO and Db4o Object Database. *Environmental*

Modelling & Software, 22(12), 1805-1810. doi: http://dx.doi.org/10.1016/j.envsoft.2007.05.016

Lin, Y.-S., Luo, S.-J., & Chen, B.-Y. (2013). Artistic QR Code Embellishment. *Computer Graphics Forum, 32*(7), 137-146. doi: 10.1111/cgf.12221

Link, J. Á. B., Smith, P., Viol, N., & Wehrle, K. (2012). Accurate map-based indoor navigation on the mobile. *Journal of Location Based Services, 7*(1), 23-43. doi: 10.1080/17489725.2012.692620

Malawski, M., Kuz, x, niar, M., Wo, x, . . . Bubak, M. (2013). How to Use Google App Engine for Free Computing. *Internet Computing, IEEE, 17*(1), 50-59. doi: 10.1109/MIC.2011.143

Masoumeh, M., & Mehregan, M. (2012). An Effective Model for Improving the Quality of Recommender Systems in Mobile E-Tourism. *International Journal of Computer Science & Information Technology, 4*(1), 83-92.

Montague, K. (2010). *Accessible indoor navigation*. Paper presented at the Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility, Orlando, Florida, USA.

Morris, J. (2011). *Android User Interface Development Beginner's Guide* Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=948578

Oh, J.-S., Kim, H., & Jayakrishnan, R. (2012). Tourist Activity Simulation Model for Assessing Real-Time Tour Information Systems. *Journal of Intelligent Transportation Systems, 16*(3), 118-131. doi: 10.1080/15472450.2012.688388

Pan, S., Tsai, H., & Lee, J. (2011). Framing New Zealand: Understanding tourism TV commercials. *Tourism Management, 32*(3), 596-603. doi: http://dx.doi.org/10.1016/j.tourman.2010.05.009

Prodan, R., Sperk, M., & Ostermann, S. (2012). Evaluating High-Performance Computing on Google App Engine. *Software, IEEE, 29*(2), 52-58. doi: 10.1109/MS.2011.131

Rodriguez-Sanchez, M. C., Martinez-Romo, J., Borromeo, S., & Hernandez-Tamames, J. A. (2013). GAT: Platform for automatic context-aware mobile services for m-tourism. *Expert Systems with Applications, 40*(10), 4154-4163. doi: http://dx.doi.org/10.1016/j.eswa.2013.01.031

Santiago Júnior, V. d., & Vijaykumar, N. (2012). Generating model-based test cases from natural language requirements for space application software. *Software Quality Journal, 20*(1), 77-143. doi: 10.1007/s11219-011-9155-6

Schiff, A., & Becken, S. (2011). Demand elasticity estimates for New Zealand tourism. *Tourism Management, 32*(3), 564-575. doi: http://dx.doi.org/10.1016/j.tourman.2010.05.004

Severance, C. (2012). Discovering JavaScript Object Notation. *Computer, 45*(4), 6-8. doi: 10.1109/MC.2012.132

Shumack, K. A., Reilly, E., & Chamberlain, N. (2013). QR Code Mania! *Strategies, 26*(3), 9-12.

Song, W., Kim, Y., Kim, H., Lim, J., & Kim, J. (2014). Personalized optimization for android smartphones. *ACM Trans. Embed. Comput. Syst., 13*(2s), 1-25. doi: 10.1145/2544375.2544380

Souza, V. S., Lapouchnian, A., Angelopoulos, K., & Mylopoulos, J. (2013). Requirements-driven software evolution. *Computer Science - Research and Development, 28*(4), 311-329. doi: 10.1007/s00450-012-0232-2

Walter, U., & Just, T. (2013). Combining Testing Methods (Vol. 110, pp. 345-345): Deutscher Aerzte-Verlag GmbH.

Weiss, M. A. (2007). *Data structures and algorithm analysis in Java*. Boston: Pearson/Addison-Wesley.

Wu, Y., Xu, J., Hu, Y., & Yang, Q. (2003, 12-15 Oct. 2003). *A shortest path algorithm based on hierarchical graph model.* Paper presented at the Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE.

Yanying, G., Lo, A., & Niemegeers, I. (2009). A survey of indoor positioning systems for wireless personal networks. *Communications Surveys & Tutorials, IEEE, 11*(1), 13-32. doi: 10.1109/SURV.2009.090103

Yu, H., Li, M., Liu, T., & Ning, Z. (2012, 26-28 Sept. 2012). *Use Critical Sub-graph to Optimize the In-building Shortest Path Algorithm.* Paper presented at the Innovations in Bio-Inspired Computing and Applications (IBICA), 2012 Third International Conference on.

Zhiyuan, L., & Yan, L. (2009, 18-20 Jan. 2009). *A Novel Shortest Path Approach for Multiple Layers of Graphs.* Paper presented at the Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on.