Design and Evaluation of a Mobile Photo Gallery in TIP

A thesis submitted in fulfilment
of the requirements for the degree of

Master of Science

in Computer Science
University of Waikato
by

Yi Wang

Supervisor
Dr. Annika Hinze

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2007

Design and Evaluation of a Mobile Photo Gallery in TIP

Yi Wang
Department of Computer Science, University of Waikato
yw178@cs.waikato.ac.nz

28th February 2007

Abstract

As a part of the Tourist Information Provider (TIP) system, this project focuses on creating a photo gallery service in the TIP system, which allows users to share, browse and categorize their photos. The core of this project is to provide users a location-based photo browsing. The system provides photos which are taken in the current user's location. We considered privacy control on photos that users uploaded. A photo owner is able to sign an access level to each of their photos and permit different users to access them. We also considered reusing resources. The system allows a user to use an URL of a photo in the system in stead of uploading the photo from the local computer. The system also provides a URL of each photo in order to use the photo on other web places, e.g., Blogs. We use tags and photo metadata Eixf to categorize photos.

Acknowledgement

It is a pleasant aspect that I have now the opportunity to express my gratitude to those who ever helped me during my study.

First of all, I would like to thank my supervisor Dr. Annika Hinze who gave me ideas and discussions with me about my project. Without her support and encouragement, I would not have been able to finish this thesis.

I would like to thank my parents who continuously gave me financial and spiritual supports. Without my parents, I could not have studied in New Zealand. They always encouraged me when I had a hard time.

I also would like to thank the ISDB research team members for helping me to solve problems related to my project. Especially Jingyu Chen; we had many discussions where we shared ideas about our projects.

Thank you to all academic staff of the Computer Science Department and the administrative staff for your helps and supports.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Taking photos is an important aspect when traveling. Sharing travel photos is the best way to share travel experiences. Nowadays, GPS receiver embedded digital camera is available in the market. This kind of cameras record GPS information into a photo file while the photo is taken which means it records where the photo is taken. Like a time stamp, this is a location stamp on the photo. Many applications could use GPS information with photos. For example, to display photos in map according to the GPS information in the photos.

This project focuses on sharing and reusing photos. It allows a user to view photos that have been taken close to the user's current location. While the user is moving, the displayed photos are updated automatically. In this project, we also consider privacy control to allow users to share their photos with different users. Our project is an extension to the TIP system.

1.1 The TIP System

The Tourist Information Provider (TIP) [9] is a mobile tourist information system. It is a combination of location-based services (LBS) and an event notification service (ENS). Information is delivered to the users based on their current location, sight-related information, their interests and their travel history. The location information is provided by the GPS to the users' mobile devices. The location data are sent to TIP while a user is moving via wireless connection part of the mobile device. The users dynamically get information from the system while they are moving with their mobile devices such as mobile phones or PDAs.

As the system needs to know what are the users' interests for filtering events, users have to register on the system and submit their profiles to the system before they can use TIP. The events are the user's location changes. The profile is adjustable: Users can modify their profile in order to receive different information. After a user registered, the system keeps the user information as a user profile, for example, the sight groups they are interested in e.g. towers and churches as well as the topics the users are interested in e.g. history of the sights. The sights which the user visited are recorded as user's travel history. This information is used to avoid users getting same information many times when they visit the sight again. The existing TIP system is able to provide users information about the current visiting sight. Furthermore, the system can also recommend some nearby sights and current or near future activities in those sights based on the users' profile and the current location.

In order to suit different users' demands, some new TIP components have been developed such as advanced recommendation [11] trust-based recommendation [16], community-based interaction [12] and travel planning [10]. Furthermore, some people worked on improving the current TIP database structure to be more flexible for difference sights' information [2] and connecting to other information sources to get useful information back to TIP such as Greenstone [8]. Travel itinerary [19] allows users to share their travel itineraries with others.

In the current TIP system, users can create and share their travel itineraries. However, they cannot share travel photos.

1.2 The Goal of this Thesis

The goal of this project is to create a photo gallery service in the TIP system. It will allow users to view others photos which are taken around the user's current location in the so-called *living mode*. The photos are updated automatically while the user is moving. Users can share their own photos with other users. The owner of the photos can decide and set who can see their photos. Users can also browse photos in the *browsing mode* by tag, photo capture time, location and photo owner.

1.3 The Structure of this Thesis

In Chapter 2, six scenarios represent user behaviors when they are organizing, sharing, browsing and searching photos. Subsequently, six user requirements have been identified based on the scenarios.

In Chapter 3, we review three mobile photo systems and two photo sharing web sites to see how they achieve our user requirements. We also compare three classification systems and decided which one should be used to classify photos in this project.

In Chapter 4, we describe the whole design process from use case study to database design and user interface design.

Chapter 5 describes implementation techniques in the TIP system and software architecture of the TIP system and this project. The implemented database is discussed using database table definitions. The software implementation is also described in this chapter.

In Chapter 6, we evaluate our implementation on both qualitative and quantitative criteria.

In Chapter 7, we first summarize this thesis. Then we discuss limitations of this project. Finally we outline possible future work.

Chapter 2

Scenarios and Requirements

For most people, taking photos is very important in traveling. In order to find out user requirements of dealing with their photos, we look at six scenarios which illustrate users' behaviors of sharing, browsing their photos.

2.1 Scenarios

The following scenarios represent users' interactions when they are organizing, sharing, browsing and searching photos. The scenarios are based on our and our friends' photo sharing experiences.

2.1.1 Scenario One: Organizing Large Amount of Photos

Mike is an enthusiastic amateur photographer. Anything could interest him to take a photo. When he is out, his camera is always with him. Landscapes, sea, birds, and trees he takes pictures of everything. As a result, he gets hundreds photos every time. Organizing this large amount of photos is not that easy. Sometimes he just too tired to add a name or a comment for each individual photo. Like many other digital camera users, John copies his photos from his camera directly to his computer without changing names or giving comments. It could be very hard if he wants to find out some photos later. The only information that the camera automatically recorded to help him finding out the photos his looking for is the time of the photo taken. Computer can only sort the photos with the taken time and Mike has to manually find out what he is looking for from hundreds photos taken in same day. For other people, let's say that they take tens of photos a day, using time to sort the photo is good enough to find the photos they want. However, in the case of Mike, only sorting by time is not very helpful. He needs a more efficient way to retrieve the photos he wants from his hundreds photos. It will be very helpful for retrieving photos if each photo has a meaningful name. However, in fact, people will even not rename their photos individually because this job is time consuming and boring. How to group photos in common characteristics? Locations, events and people information should be added to photos for later retrieving. For example, selecting photos which were taken in same place and give a place name as an annotation for all those photos once and all photos now have a same place name. Giving a wedding annotation for a bunch of photos related to wedding. One photo can have many annotations. Later, if Mike wants to find out the photos related with wedding, he can search for wedding annotation and all photos have an annotation called wedding is showing up. To narrow down the number of the photos of search results, Mike can also combine more than one annotation. E.g. search for wedding and Hamilton will give more specific results than use wedding only.

2.1.2 Scenario Two: Sharing Different Photos With Different People

Jackie would like to share her photos with others online. For some photos, she would like to share with everyone, even who she does not know. For some special photos, she only wants to them to be seen by all her friends. And some other photos, she does not even want to show to all her friends. She only wants to share these photos with some of her friends. And some very private photos she wants herself can see. Privacy control is very important for sharing photos online, especially for those who want to put all their photos online and do not want all photos to be seen by everyone. Figure 2.1 shows Jackie's requirement for sharing photos. Some online photo sharing web site only gives users limited control for the privacy of their photos. For example, some web site only allows users to set a photo as public, contact or private. Public means everyone can see. Contact means only the users on the owner's contact list can see the photo. Private means no one can see the photo except the owner. Since Jackie requires more access levels, she is not happy with this photo sharing web site. She can only keep the photos as private, which

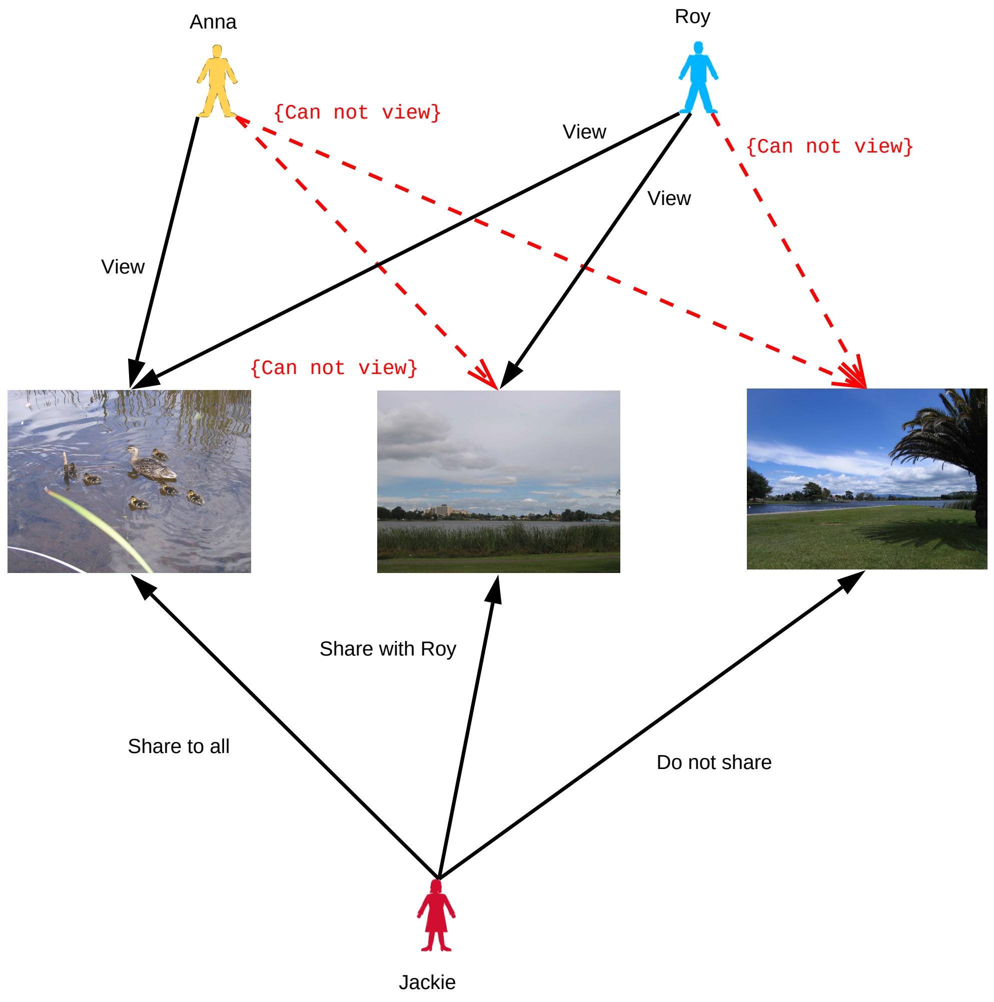


Figure 2.1: Jackie's requirement for sharing photos

she wants to share with some of her friends and does not want her other friends to see.

2.1.3 Scenario Three: Browse Others Photos By Locations

Jason is a travel enthusiast. He has traveled many places. However, there are more places he is still willing to go but he has not got a chance to go because of time and budget. Now he found a new way to travel. He called it virtual travel. He is using an online photo gallery web site. He can easily click on a place on the map and the photos which other users captured in that place will be displayed. Without going there, from viewing others photos, he can see what others saw in that place. From viewing others photos, he gets a general idea about the place which he has not been to. For those places that he has visited, he can also get some new views from others photos. Some good photos are driving him to visit that place. After his travel, he also put his photos online, so that others can also view his photos.

2.1.4 Scenario Four: Search For Photos Related a Particular Event

Carl is a super video game fan. Electronic Entertainment Expo (E3) exhibits the newest video games, consoles, gadgets and video game technologies. Everything happened in the E3 is interested him. Unfortunately, he was not able to attend the 2006 E3 conference. Except browsing the 2006 E3 conference official web site to get the conference information, he also likes to know more about the E3 information from other people who have attended the E3. The photos that other people took in the E3 conference give him more information about the E3 conference. Searching photos by the key word 2006 and E3 will give Carl many photos related to the 2006 E3 conference.

2.1.5 Scenario Five: Reusing Photos Between Online Photo Gallery, Blog And Forum

Peter is an enthusiastic traveler and he likes to keep his journey on his Blog. He also likes to add his journey photos onto his Blog as well. He could use photos in his online photo gallery. The online photo gallery provides him a URL for each photo he has uploaded so that he can reference his photos in his Blog with those URLs. Other way round, the online photo gallery can also take a URL instead of uploading photos from local computer. It is a more efficient way to using his photos. He can also use those URLs to reference photos when he wants to post his photos on other forums. The online photo gallery becomes his personal photo repository. Any time he wants to show his photos to his friend or post his photo some where on the web, he does not need to upload his photos from local computer because his photos are already online somewhere. The only thing he needs to do is get the URL of the photo his wants to show and show the URL to his friend or post it on the web.

2.1.6 Scenario Six: Viewing Others Photos Based on the User's Current Location

Sarah is visiting Hamilton Gardens. She has a GPS receiver embedded PDA. Her current location coordinates, which received by the GPS receiver, will be sent to the server. The server sends back the photos which are taken in the current location and displays in her PDA screen. While she is moving, the photos are updated automatically according to her new position. We call this real mode. She is also able to choose whose photos should be shown. E.g. she could choose only display her friends' photos on the screen. Instead physically moving around to see photos, she could also virtual moving around by changing her position in her PDA which we called *browsing mode*. She will see the photos exactly same as she is in there in the real mode. Since she is visiting gardens, she might want to see the views of different seasons or different times of a day.

2.2 User Requirements

After we studied the six scenarios, we identified the following user requirements:

UR1 Categorize their photos with minimum input information required. Users do not want to input large amount of information about the photos, especially when they upload lots of photos.

UR2 Privacy control: owners set access permission to each of their photos so that they can allow or avoid other users to see their photos.

UR3 Viewing photos by current location.

UR4 Browsing photos by given locations.

UR5 Searching for location or event related photos.

UR6 Reusing photos between the online photo gallery, blog and other places which provide photos as URLs.

2.3 Conclusion

In this chapter, we introduced six scenarios which are based on our and our friends' travel experience. We analyzed each scenario and summarized six user requirements which include categorize photos, privacy control for sharing photos, viewing photos by current location, browsing photos by given location, search for location or event related photos and reusing photos with other places.

Chapter 3

Related Work

In this chapter, we review different kinds of photo systems to see if and how they achieve our user requirements. We have reviewed three mobile photos systems. Considering the web-based photo systems are very popular nowadays, we also considered selected online photo systems. We draw a table to summarize all systems we have reviewed against our user requirements. To take account of categorizing photos, we also looked at the main types of classification systems. We compared predefined classification and folksonomy.

3.1 Mobile Photo Systems

In this section, we review three different mobile photo systems in order to find out how well they achieve our six user requirements. All these mobile photo systems are working with client-server architectures. A comparison of these mobile photo systems against our user requirements is shown in Table 3.1 in Section 3.3.

3.1.1 The Mobile Media Metadata System

The MMM system [1, 3] consists of a client software working on camera phones and a server working on Linux. When a user takes a photo with her/his camera phone, if the user chooses to share this photo, the client software will send the metadata to the server and display a list of suggested sharing recipients. The user can choose one or more recipients. At the meanwhile, the photo is uploaded from the phone to the server in the background for later viewing on the MMM web site. The metadata includes time, date, cell ID, co-present Bluetooth devices and GPS reading if available.

When the server receives the photo and the metadata, it will do a metadata similarity processing. Since cell ID is the nearest cell tower id, the server can find out the location (city block) of the photo taken. If the GPS information is available, it will produce more accuracy location information. Based on this metadata, the similarity processing compares the uploaded photo to a database of previously captured photos and their respective metadata in order to infer the most possibility metadata for the new photo. The server will send metadata guessing results to the user phone's XHTML browser. The guessing results are order by the most probable metadata first. The user can modify the metadata and sent it back to the server. "The photos and metadata in the database are not limited to the user's own photos and metadata, but contain every user's annotated media to leverage the advantages of shared metadata." Sharing and reusing all user's photos and metadata is one of the main purpose of the MMM system design. However, this key point of MMM could be the main issue of the MMM system. The information of location, time and place could be the potential privacy issue.

The MMM system achieved our UR1 every well, it automatically get location, time, date information and increase the efficiency of user annotation input by using metadata similarity processing. The UR4 and UR5 is also achieved by the MMM system.

3.1.2 The MobShare System

MobShare [18] is a mobile picture sharing system which is focusing on immediate and controlled sharing of mobile images. The MobShare system design is based related literature and a use study which interviews mobile camera phone users.

The MobShare system is working as client-server architecture. Client side software is working on users' mobile camera phones. Server side is a web server. Users are able to share their mobile photos to an organized web album and decide who is able to view the photos. Time, date, cell id and MSISDN data are collected by the client side software automatically. The software is also guiding users to annotate and organize the photos, put them in right folder or create a new folder for them. Then, a user could decide who s/he wants to share the photos with, by collection them from her/his mobile phone address book. After the user uploading the photos, the people, who have selected to view the photos, will be notified by SMS. The existing users have an option to receive the new photo notification SMS. If the user does not register on the MobShare system, s/he will get a system generated password. So that s/he can use her/his phone number (MSISDN) and the password to login the MobShare system without register. The password is not temporary. S/he has been registered by their friend's photo sharing notify SMS. Since the mobile phone number is unique. The MobShare system is using the phone number as user ID.

The MobShare system allows a user to browse the photos by horizontal timeline of folders or gallery view and vertical timeline view. Notification of new pictures is used to give users a visual notifier in the photo folders. The MobShare partially achieved our UR2. It provides users an album (folder) level sharing. Users can easily share the photos with the people they want. We are prefer a single photo level sharing which means users are able to set access permission to a single photo.

3.1.3 The Context Navigation System

The context navigation system [15] presents an application which allows users to navigate through a photo collection, using context. The application provides two navigation modes: physical and virtual navigation. Physical navigation permits the user to access to photos that were taken near him/her. The photos are updated while the user's position changed. Virtual navigation permits the user to virtually browse the surroundings by jumping from one photo to another. The virtual navigation mode is very useful. It allows users to virtually access sites which are not able to physically access at the moments. E.g. a site is not opened at the moment.

The virtual navigation also allows users to view photos, taken in different time, of same location. However, we figured out that the article is mixed the site location and the position of a photo taken. In some case, this two positions are could have a quite large distance. Such as landscape photo. Even they are close, they always have a distance. The application in the article is able to show the eight photos surround current location and the eight photos are representing 8 directions of current location. However, the locations they are using here are the locations where the photos are taken. The actual site locations on the photos, which they actually wanted to use for navigation, are different with the location of the photos taken. Due to this concept mixing, their photo navigation is confusing users. Even we are looking at their screen shots; we are still confusing about the direction of the photos showing. This application has achieved our UR4 and UR5.

3.2 Online Photo Systems

In this section, we looked at two popular online photo web sites. They both allows users sharing and browsing photos online. We compared them with our user requirements.

3.2.1 Flickr

Flickr [6] is a popular photo sharing web site. Users have control of their photo privacy: when a user uploads photos to Flickr, s/he can flag the photos as *public* or *private*. *Public* means that the photos can be seen and searched by all people. If the user set the photos as *private*, they get more options. They can set as only themselves can see the photos or their friends or their family can see. Friends and family are the extra information to describe a contact. In the contacts, a contact could be a friend or a family or just a contact. A friend and a family have a higher authority. The authority levels from higher to lower are owner, friends/family, contact and everyone. By using this four authority level structure owners can share their photos to right range of people. The person has a higher authority they can see more photos. However, if a user real wants to share her/his photos to specified users only, Flickr does not directly support it. To solve this problem, Flickr allows users to create groups. The groups could be public or private. Only invited users can be a member of a private group. And only group members can see the photos in the private group.

Flickr allows users to categorize their photos with tags. Users can put one more tags on each of their photos which are helpful for sorting and searching photos. Tags could be place name, event and any relevant words. Flickr also provides sets to allow users categorize their photos. Since a photo can belong to one set or many sets or none at all, it is more flexible than the physical hierarchy.

When users browse photos by tag, they have two ways to sort the photos. One way is choosing most recent which is ordered by uploading date and the latest one displayed first. The other is most interesting which is ordered by the number of people who call this photo a favourite. There is not other way for users to sort photos under a tag. When users browse others' sets, they have no control of the photo order. However, as an owner, s/he can organize the order of their photos in a set. Flickr provides a Macromedia Flash based interface called Organizr to organize photos. Users can easy change the order of the photos in a set by dragging and dropping the photos. Organizr is more powerful than just change the photos' order. Users can use it to create a set and a group, add photos to a set or a group, delete photos from a set or a group, change the set name and set and change the permissions, title, tags of the photos. Flickr has achieved our UR1, UR4 and UR5. Flickr allows users to use the URL of photos elsewhere. However, it does not allow users to use an URL instead of uploading photo from local computer. Therefore, Flickr partially achieved our UR6.

3.2.2 MSN Spaces

MSN Spaces [14] is a free blogging service provided by Microsoft. Photo sharing is one of services provided by MSN Spaces. MSN Spaces allows users to set permission for their Spaces, which could be public, MSN allow list and customize. Public means everyone on the internet can access it. MSN allow list means only the owner's MSN contacts can access the owner's MSN Space. Customize means the owner can specify who are able to access her/his MSN Space. However, since the permission is for the whole MSN Space, users cannot set some photos as public and some photos as private. For example, a user wants to share her honey moon photos to her close friends. At the same time, she has shared many photos because she is an amateur photographer. With the current version of MSN Space, she is not able to do those two things at the same time.

MSN Spaces provides albums to allow users to categorize and store their photos. One photo can only be and must be in one album. The albums are flat, which means that users cannot have an album inside another album. Users cannot store their photos with a hierarchy structure with MSN Spaces. When users browse others' MSN Spaces, they are not able to change the order of the photo showing. As the default, the photos are ordered by the upload date. Only the owners can change the order of their photos. When a user open the album edit page, s/he can see the all the photos in one album and they can drag and drop the photos to change the order.

MSN Spaces used albums to help users to organize their photos. However, we believe that is not very flexible way to do it. Sometime it is hard to find a specific photo because MSN Spaces does not provide a search function. We say MSN Spaces partially achieved UR1. MSN Spaces provide user a limited privacy control. Users can only set their whole space to be accessed by other people. They cannot set an access permission on individual photo or even an album. So it is partially achieved UR2.

	UR1	UR2	UR3	UR4	UR5	UR6
The MMM	+	-	-	+	+	-
The MobShare	-	(+)	-	-	-	-
The Navigation	-	-	+	+	-	-
Flickr	+	+	-	-	+	(+)
MSN Spaces	(+)	(+)	-	-	-	-
Current TIP	-	-	-	-	-	-

Table 3.1: Comparison of photo sharing systems and current TIP against URs (Symbols: + achieved, (+) partially achieved, - not achieved)

3.3 Comparison of Photo Sharing Systems

From Table 3.1 we can see that the MMM system achieved our UR1, UR4 and UR5. The MobShare only partially achieved our UR2. The navigation system achieved our UR3 and UR4. Flickr achieved our UR1, UR2 and UR5. Flickr also partially achieved our UR6. MSN Spaces only partially achieved our UR1 and UR2.

From Table 3.1, we can see that none of those systems achieved all six user requirements. No current system fully achieved our UR6 which is photo reusing. Our current TIP system does not allow users to upload or share their photos so that no user requirements are achieved by the current TIP.

3.4 Classification Systems

We compare related classification systems in the section in order to find out a solution for our UR1. Those classification systems are not particular used for photo classification. However they are used to organize some specific area information. We focus on comparing the advantages and disadvantages of predefined classifications, folksonomy and metadata.

3.4.1 Predefined Classifications

The predefined classification schemes are created by the specialists or organizations. It is fixed into the system. Users can use the classification schemes to classify their works or things. For example, ACM Computing Classification System (CCS) is using for classify computing field articles. ACM CCS combined a classification scheme tree and key word which is folksonomy. The classification scheme tree is a four-level tree that has three coded levels and an uncoded level of subject descriptors. The main classification is depending on the classification scheme tree and the key word is only as a supplement. CCS allows user to have more than one node of the tree to index an article. This handles the situation that the article belongs to more than one category. For key word, users can use a word or a phrase.

We can see that CCS has most of advantages of the predefined classification and folksonomy. Such as well defined classification scheme tree and easily creation of key works. However the disadvantages of these two taxonomies are also remained. E.g. the classification scheme tree needs to be revised. The CCS has been revised four times since it was established in 1982. Before users apply the CCS scheme, they have to learn the structure of the scheme tree and find out where it the proper place to put their articles in. Because ACM is mainly involved technical reports, it is more serious than classify personal photos. The author may take few minutes to hours to apply ACM CCS on their report. However, our users do not want to take long time to classify their photos. They are properly looking at a time of few seconds. If they need they need long time to do that, they maybe just ignore to classify photos. There are lot of people even do not want to change the title of their photo after they upload it. What we need for classify photos should be very quick to do and very easy to do. They should not to be asked to learn some classification scheme before classifying their photos.

One thing we found in the ACM CCS is that the authors are more likely to use phrases as key words other than single word. The benefit of doing this is reducing the semantic ambiguity. The drawback is the key words have less chance to be the same. In other words, one key word only related to one or few articles. It is not too bad for CCS because authors are normally using some academically terms as key words which have a better chance to match others. However, if we using more phrase to tag photos we will get more tag fragment which we do not want. Tag fragment means the one tag only related to one or few photos. It is against our purpose of using tag. The idea of using tags on photos is that we want to show and share the common of the photos. We are kind of in the middle of the general and specific. The more general of the tag, the more photos have the tag. On the contrary, the more specific of the tag, the few photos have the tag especially when users use phrase as a tag. It is true that the more specific of the tag the more accuracy of the photos will be returned when we search for the tag. We encourage users to use common tags to tag their photos if the tags are proper for the photos. We will make some special design to help and encourage users to use common tags to tag their photos.

3.4.2 Folksonomy

Folksonomy [17] is a combination of ”folk” and ”taxonomy.” In contrast to professionally developed predefined classification, folksonomies are unsystematic and lower content categorization costs because there is no complicated, hierarchically organized nomenclature to learn. User could simply create and use tags as they want.

Comparing with formal taxonomies, folksonomies are inherently open-ended and can therefore respond quickly to changes and innovations. Perhaps the greatest benefit of folksonomy is it allows user to use any words they want to tag their stuff in stead of picking a most appropriate category from the predefined classifications. After all, folksonomies are generated by people who have spent a great deal of time interacting with the content they tag. Flickr is a good example of using tags to classify photos. Del.icio.us [4] allows users to tag websites and save as online bookmarks.

3.4.3 Metadata

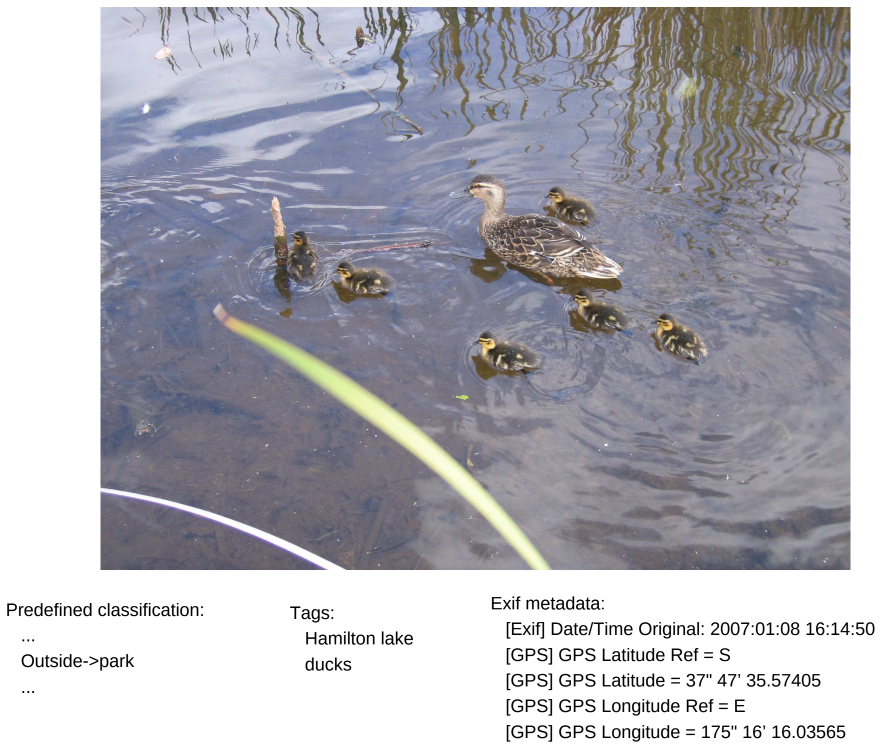
Metadata is data about data. Exif (Exchangeable Image File) is a standard for storing metadata in image files, especially those using JPEG compression. Exif is used by most digital cameras to store metadata. The metadata is created while the photo is taken by the digital camera. The metadata includes digital camera brand, digital camera model, photo capture time, GPS information (for more detail see Exif standard version 2.2 [5]).

Exif metadata can be easily use for classifying photos. For example, a photo capture time can be retrieved from Exif easily and the capture time can be use for organize photos by time. GPS information is most useful information for location base information system such as TIP. Other Exif information has been used for classify photo related things like camera. Flickr has used digital camera brand and model, which are retrieved from photo Exif, to count the popularity of the digital cameras which are used to capture photos in Flickr. This statistic is interesting for those who want to buy a new camera.

Exif is created and stored in a photo by the digital camera while the photo is taken. It is easy for computer to retrieve information from Exif and use it to classify photos.

3.4.4 Comparison of Classification Methods

Predefined classification need revision in a period time due to the new terms are introduced and previous terms are changed or involved into other categories. For example, CCS has been updated four times since it was established in 1982. The newest version is updated in 1998. Folksonomy does not need this kind of revision, because all the tags are created by users and new terms could be added by users themselves. At this point, folksonomy is better than predefined classification for classifying photos.



Predefined classification:
...
Outside->park
...

Tags:
Hamilton lake
ducks

Exif metadata:
[Exif] Date/Time Original: 2007:01:08 16:14:50
[GPS] GPS Latitude Ref = S
[GPS] GPS Latitude = 37" 47’ 35.57405
[GPS] GPS Longitude Ref = E
[GPS] GPS Longitude = 175" 16’ 16.03565

Figure 3.1: A Example of Photo Classification (Predefined Classification, Tag and Exif Metadata)

Figure 3.1 shows an example of using different classifications to classify a photo. The photo is taken in Hamilton lake. Several ducks swim in the water. Using tags, a user can any words to tag the photo. In this example, we have used location name *Hamilton Lake* and ducks to tag the photo. The user can add other word easily. Using predefined classification, here, we assume that we can only find a category called *park* in higher category called *outside* as the most related category to the photo in the classification schema. The Exif metadata of the photo shows the capture time can the GPS information in Figure 3.1. Exif metadata includes other information about the photo and the digital camera which we did not list in the Figure 3.1. All these metadata are created by the digital camera automatically when the photo is captured.

Folksonomy does not require much knowledge about classification scheme, which is easier to use than predefined classification. With folksonomy, users can tag the photos with any words. Predefined classification requires users have some knowledge about the classification scheme. At least users need to follow the scheme structure to find out the most appropriate category for the works or photos.

Folksonomy can handle the situation when a photo belongs to more than one category very well. It just simply adds more than one tags to the photo.

Predefined classification is easier for retrieval information than folksonomy. This is because the predefined classification has a good semantic structure. On the contrary, folksonomy does no have a semantic term. For example, one user could use *apple* to tag his photo of a real apple. And other user may use the same word *apple* to tag a photo of his Mac computer. They are both using the same word apple as a tag. When a user searches for apple, both two photos will show up. This makes semantic ambiguity. It is hard to solve this problem. Even we add more tags to describe the photos. When a user searches for the tag *apple*, both photos will show up with no difference.

We see that a predefined classification follows the structure of the classification scheme. However, the structure of a folksonomy could be badly defined. Tags do not have a hierarchy structure; they are flat. Since tags are created by users, any word could be a tag. Many tags could be only used for few photos or only used by one user. We called it tag fragment. This kind of situation makes tags lose the biggest benefit of the tags, which is the automatic grouping of photos. In order to minimize the occurrence of tag fragment, we could show the popular tags to the users when they are tagging the photos.

Predefined classifications provide users with a classification scheme, where users do not need to input category information, but can select a category from the scheme. This avoids the problem with misspellings. For folksonomy, this could become a large problem [13]: Tags are typed in by users, so it is easy to misspell a tag. The system will create a tag for users as they are typed in. A spell check is necessary and useful when users type in tags.

3.5 Conclusion

In this chapter, we have reviewed three mobile photos sharing systems and two online photo sharing system. As we can see from Table 3.1, none of those systems achieves all six user requirements. No systems fully achieves our UR6 (photo reusing). For UR1, we also compared predefined classification and folksonomy. We decide to use a combination of folksonomy and metadata to classify photos. The reason is that tags are better than predefined classification for classifying photos. They are easy to create and do not need to be revised such as predefined classifications. In the next chapter, we discuss the detail of the design.

Chapter 4

Design

In this chapter, we first discuss how we designed the system to achieve our six user requirements. Secondly, in accordance with the six user requirements, we produced eight use cases to demonstrate user interactions with the system. We will describe each of the use cases in detail with UML diagrams. Thirdly, we will describe the database design of this project which includes database ER diagram and relational schema. Lastly, we illustrate our user interface design by showing our paper-based prototype.

4.1 General Design Consideration

In this section, we describe high level design for each user requirement (as defined in Chapter 2).

4.1.1 Designs for UR1 (Categorize Photos)

In the Chapter 3, we have reviewed related classification systems and we decided to use a combination of tag and metadata to classify photos in our system to achieve UR1.

The advantages of tagging photos are that these are flexible, easy to use, self-extendable and does not need to be revised or maintained.

Tags are very flexible since the users create tags themselves. They can use any word they want to tag their photos. It is more flexible than using predefined classification systems (as discussed in Chapter 3).

It is also easy to use: users do not need to learn about a classification scheme before they can classify their photos. They just need to give words which they think are suitable for describing the photos.

A tag system does not need to be revised as the predefined classifications do. For example, ACM CCS has been revised four times since it was established in 1982. The reason for revising is that the new concepts are showing up which need to be included into the scheme. Sometimes old categories need to be reorganized or combined. According to this reasons, the predefined classification scheme need to be revised in a period time to keep the scheme fresh and suitable for current use.

On the contrary, using tags to classify photos, the new tags are used by the users when the new things or concepts show up. The whole tags system is updated by every time users create new tag or commonly use some particular tags. We encourage users to use common tags so that users can efficiently share their photos each other.

We will implement tag cloud which shows the most popular tags with different font size to visualize the popularity of those tags. Users can start browsing photos by click a tag in tag cloud.

The disadvantages of tagging photos are the possibility of misspelling, and usage of synonym multivocal tags.

We will show the current tag usage to the user while s/he types in a tag. The tag list will include the tag name and the number of the photos which has this tag. It is like a spelling recommendation. Users can easily choose a tag from the tag list. By doing this, it will also reduce the chance of users' misspelling. Synonym and multivocal are related to semantic which is not included in this project. We consider to involve them in our future work.

The most significant advantage of Exif metadata is that it is created automatically by a digital camera when a photo is taken. The information about the photo are store in the Exif. The system can retrieve it without the user's participation, which reduces the time of annotate a picture.

In order to reduce user input, the system automatically retrieves photo metadata from Exif when the photo is uploaded. Exif stands for Exchangeable Image File which is widely used in digital camera to store the photo metadata when the photo is captured. The Exif metadata includes time and location, which are most frequently used in this project. As most currently digital cameras do not have a build-in GPS receiver, we cannot get location (coordinates) information from Exif of the photos. We provide a user interface for users to manually input GPS coordinates for the photo. We also allow users to pick a point on Google Map and the system gets the coordinates of the point automatically.

4.1.2 Designs for UR2 (Privacy control)

UR2 is about privacy control: owners set access permission to each of their photos so that they can allow or avoid other users to see their photos. We defined that each photo can have three access levels, which are public, restricted and private. If a photo is public which means everyone can see the photo. If a photo is restricted only authority users can see. The owner decides and sets the access permission to each of their photos. If a photo is private that means only the owner can see it.

When an owner wants to give several people access permission for her/his photo, s/he could create a group first and then add all the people to give the access permission for the photo. When the group is created, s/he could set access permission to the photos for the group. All users in the group can now see the photo. Groups will be defined for each user individually. User A can not see user B's groups. Groups are used to help owners to give access permission to several users to access a photo and easily to modify the permissions later.

The group is quite flexible since we designed that an owner can put a user into more than one group. For example, a photo owner, user A and user B are university mates so the owner created a group called uni mates and s/he put user A and user B into the uni mates group. However, the photo owner got a photo which s/he only want to be seen by user A. Now the photo owner could directly set access permission to allow user A to see the photo. The better way is s/he creates another group called close uni mates and add user A into the close uni mates. Then set close uni mates to be able to see the photo.

Considering of resource reusing, we allow users to give a URL of a photo instead of actually uploading photos to the server. The only difference is that if a photo is not uploaded to the server, the server is not able to control the photo access. For example, photo owner C gives system a URL of a photo and than owner C set the photo as private. Now when other users search or browse in the system, they will not able to find the photo but if they got the URL, they can see the photo by directly opening the URL. This would not happen if the owner actually uploads the photo to the system. The system will take care of the photos inside the server by using .htaccess. For example, photo owner D uploaded a photo to the system and set it as public. User E visited the photo and saved the photo's URL. Later, D decided not sharing the photo any more so that s/he changes the photo to private. Then E wants to access the photo so s/he open the URL of photo, which s/he stored before. Now access is denied.

4.1.3 Designs for UR3 (Viewing photos by current location) and UR4 (Browsing photos by given locations)

UR3 is viewing photos by current location. When a user has a GPS receiver embedded PDA, s/he is able to see the photos which took in the current location. When the user moves, the new location-related photos will automatically show up. UR4 is browsing photos by given locations, which means a user specify a location and view the photos around that location.

We designed the system to work in two different modes. One we called *living mode* and another we called *browsing mode. Living mode* is used if the user want to see the current location's photos. The *browsing mode* is used for users who want to browse photos follow the time stream, locations or owners.

4.1.4 Designs for UR5 (Searching for location or event related photos)

UR5 is search for photos. We provide tag, time, owner and location search. We also allow users to search photos from a search result. For example, a user may search for a tag called *garden* and the result is quite lot and s/he actually wants to see the gardens in winter so that s/he can search winter again in previous search result of garden.

4.1.5 Designs for UR6 (Reusing photos)

We have designed two ways to reuse photo resources. One way is from our system to the internet. We provide a URL for each of the photos stored in our system. Users who have permission to see the photo will be able to retrieve the URL so that they can use the URL on other place such as their blog. On the other hand, we also allow users to use the URL of a photo in our system in stead of uploading the photo to our system.

4.2 Use Case and UML Activity Diagrams

In order to achieve the six user requirements that we identified in Chapter 2, eight use cases have been developed based on our design consideration in Section 4.1. We will describe all use cases (shown in Figure 4.1) in detail and use UML activity diagrams to show the steps of the action flow.

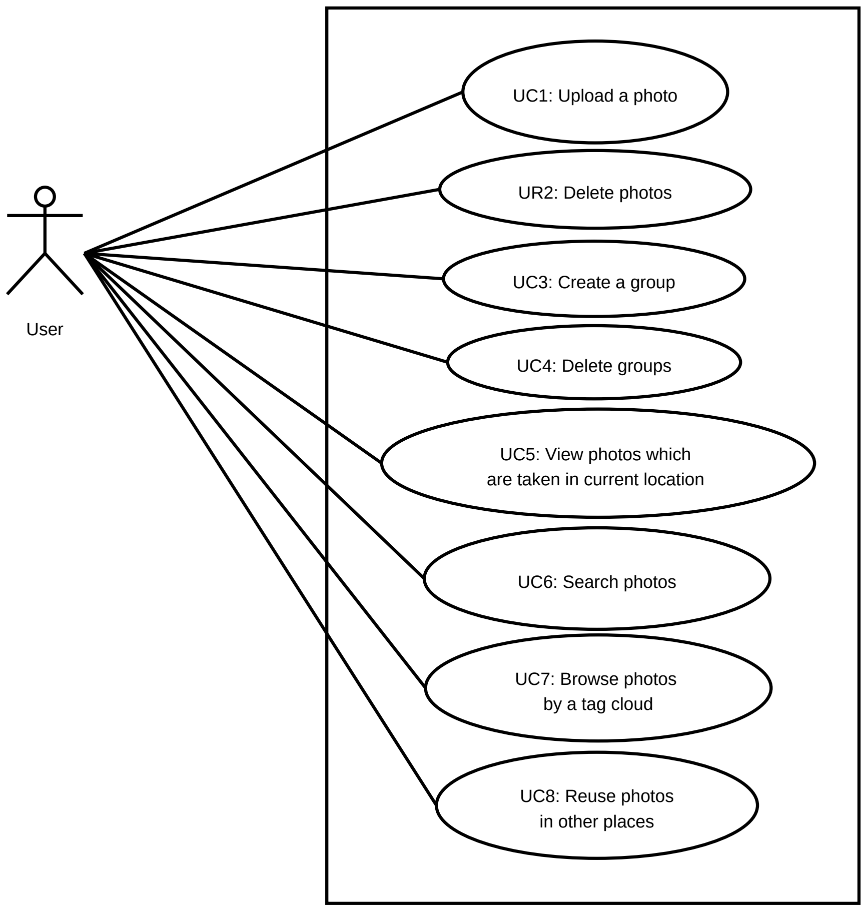

UC1: Upload a photo
UR2: Delete photos
UC3: Create a group
UC4: Delete groups
UC5: View photos which are taken in current location
UC6: Search photos
UC7: Browse photos by a tag cloud
UC8: Reuse photos in other places
User

Figure 4.1: Use Case Diagram

4.2.1 Use Case 1: Upload a Photo

Figure 4.2 shows the activity of the *upload a photo* use case.

Actor: User

Pre-condition:

The user has successfully logged into the TIP system.

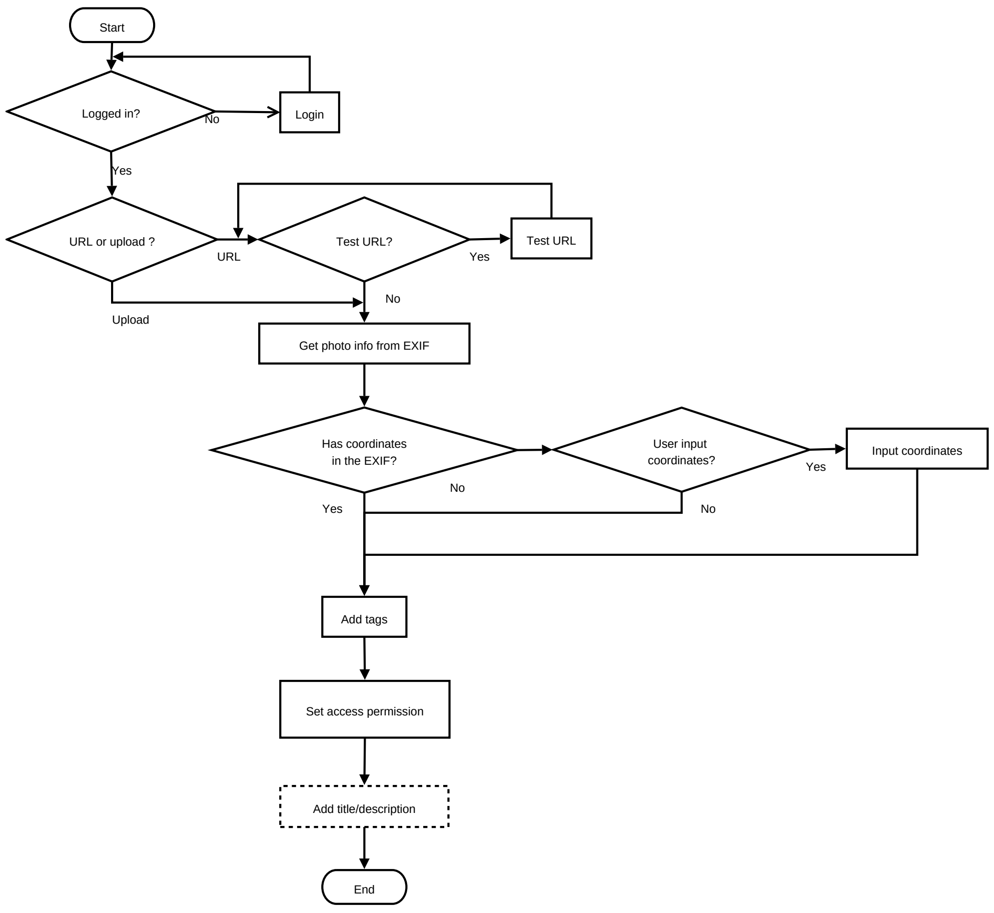


Figure 4.2: Activity Diagram of "Upload a Photo" Use Case

Basic course:

1. The user provides a photo by either local image file or a URL. The user is able to test the URL to know if it works.
2. The system reads the photo information from the Exif of the image file, which include photo capture time and GPS coordinates.
3. The user provides tags to tag the photo.
4. The user sets access permission for the photo. When the user set the permission as *restricted*, s/he is able to select a group to access this photo. Only the selected group members and the owner can access this photo.
5. The user provides title and description for the photo (optional).

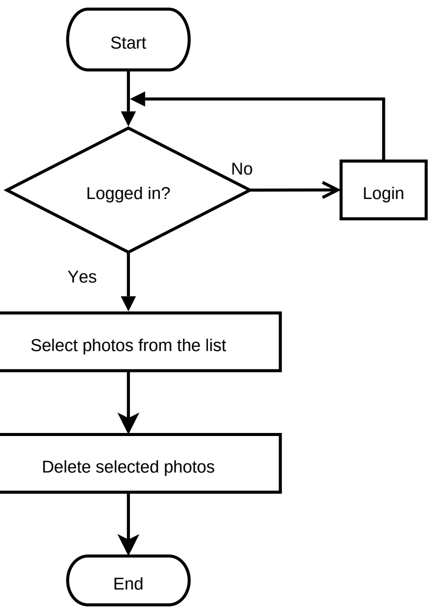


Figure 4.3: Activity Diagram of "Delete Photos" Use Case

Alternative course:

1. If the user has not logged in, the system will require them to login before starting upload a photo.
2. If photo capture time or GPS coordinates is not available, the system will ask the user to provide manually.

4.2.2 Use Case 2: Delete Photos

Figure 4.3 shows the activity of the *delete photos* use case.
Actor: User
Pre-condition:
The user has successfully logged into the TIP system.
Basic course:

1. The system displays a list of all photos of the current logged in user.
2. The user selects photos from the list and confirm to delete.
3. The system deletes the selected photos from the system.

Alternative course:

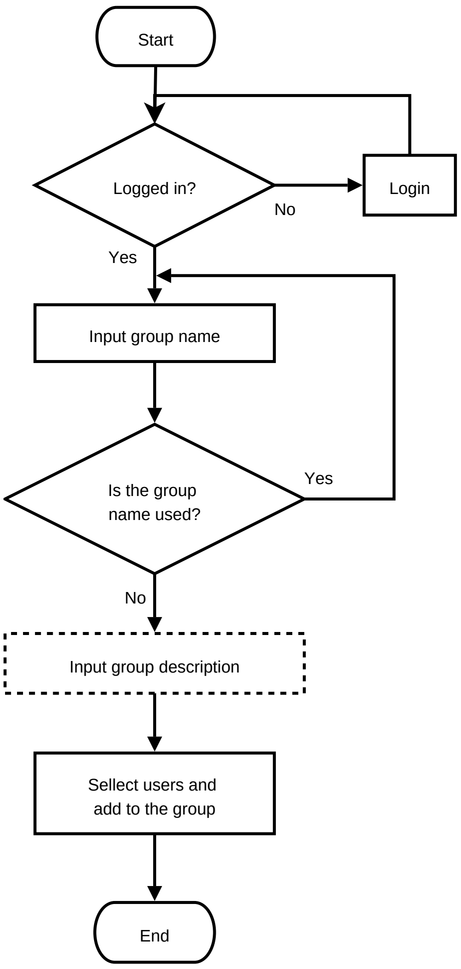


Figure 4.4: Activity Diagram of "Create a Group" Use Case

1. If the user has not logged in, the system will require them to login before selecting photos for deletion.

4.2.3 Use Case 3: Create a Group

Figure 4.4 shows the activity of the *create a group* use case.

Actor: User

Pre-condition:

The user has successfully logged into the TIP system.

Basic course:

1. The user provides a new group *name*.

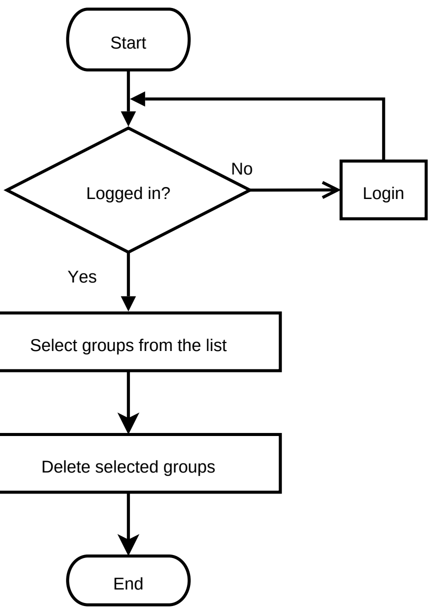


Figure 4.5: Activity Diagram of "Delete Groups" Use Case

2. The user adds other users to this group.

3. The user provides group description (optional).

Alternative course:

1. If the user has not logged in, the system will require them to login before creating a group.

2. If the group *name* exist, the system will ask user to provide another name.

4.2.4 Use Case 4: Delete Groups

Figure 4.5 shows the activity of the *delete groups* use case.
Actor: User
Pre-condition:
The user has successfully logged into the TIP.
Basic course:

1. The system displays a list of all groups of the currently logged in user.

2. The user selects groups from the list and confirm to delete.

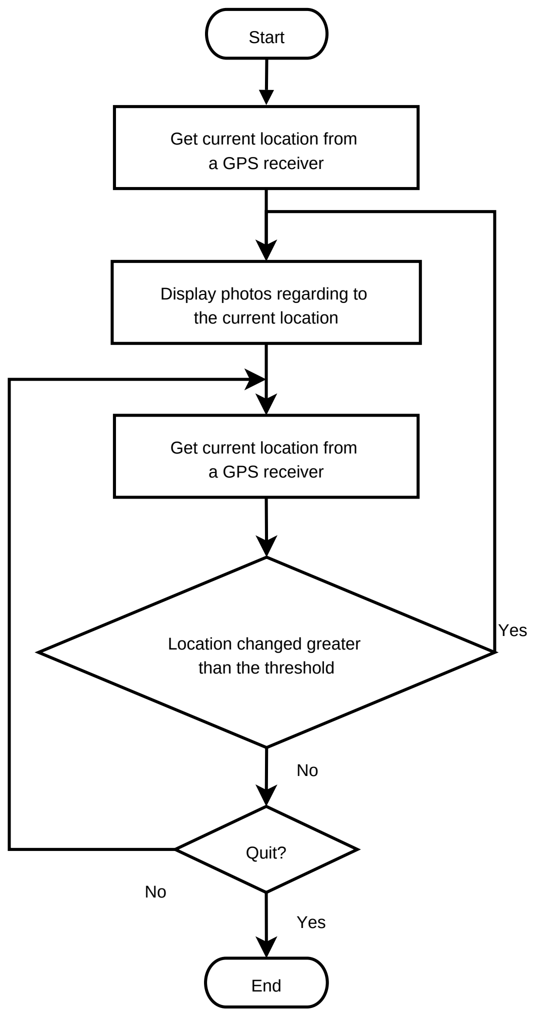

Start
Get current location from a GPS receiver
Display photos regarding to the current location
Get current location from a GPS receiver
Location changed greater than the threshold
Yes
No
Quit?
No
Yes
End

Figure 4.6: Activity Diagram of ”View Current Location Photos” Use Case

3. The system deletes the selected groups from the system.

Alternative course:

1. If the user has not logged in, the system will require them to login before deleting groups.

4.2.5 Use Case 5: View Current Location Photos

Figure 4.6 shows the activity of the *view current location photos* use case.
Actor: User

Basic course:

1. The system gets the current location from a GPS receiver.
2. The system displays photo according to the current location.
3. The system keeps update the current location from the GPS receiver.

Alternative course:

1. If the user has not logged in, s/he can only see *public* photos
2. If the location change is greater than the threshold, the system will update photos related to the new location.

4.2.6 Use Case 6: Search Photos

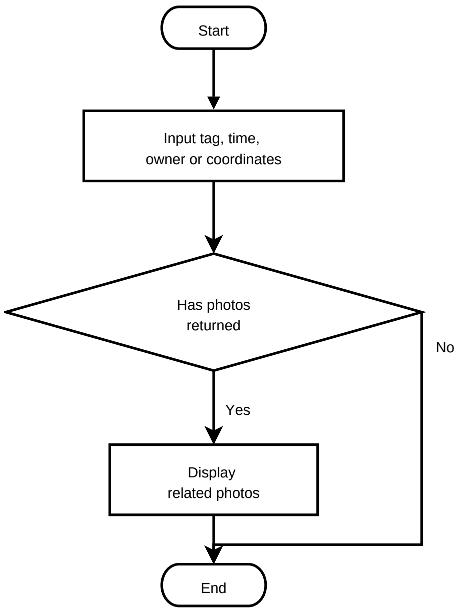


Figure 4.7: Activity Diagram of "Search Photos" Use Case

Figure 4.7 shows the activity of the *search photos* use case.
Actor: User
Basic course:

1. The user provides tag name, owner of the photo, photo capture time or/and location to search photos.

2. The system displays search results.

Alternative course:

1. If the user has not logged in, s/he can only see *public* photos
2. If no photo is found, the system suggests the user to modify search conditions and do another search.

4.2.7 Use Case 7: Browse Photos

Figure 4.8 shows the activity of the *Browse photos* use case.
Actor: User
Basic course:

1. The user choose a start point of browsing from either tag cloud, location or time.
2. The system display related photos.

Alternative course:

1. If the user has not logged in, s/he can only see *public* photos

4.2.8 Use Case 8: Reuse Photos in Other Place

Actor: User
Basic course:

1. The system provides a URL for each photo.
2. The user copies the URL and uses it elsewhere.

Alternative course:

1. If the user has not logged in, s/he can only see *public* photos

4.3 Database Design

In this section, we describe the database design of this project. We describe the ER diagram of this project and introduce the relational schema.

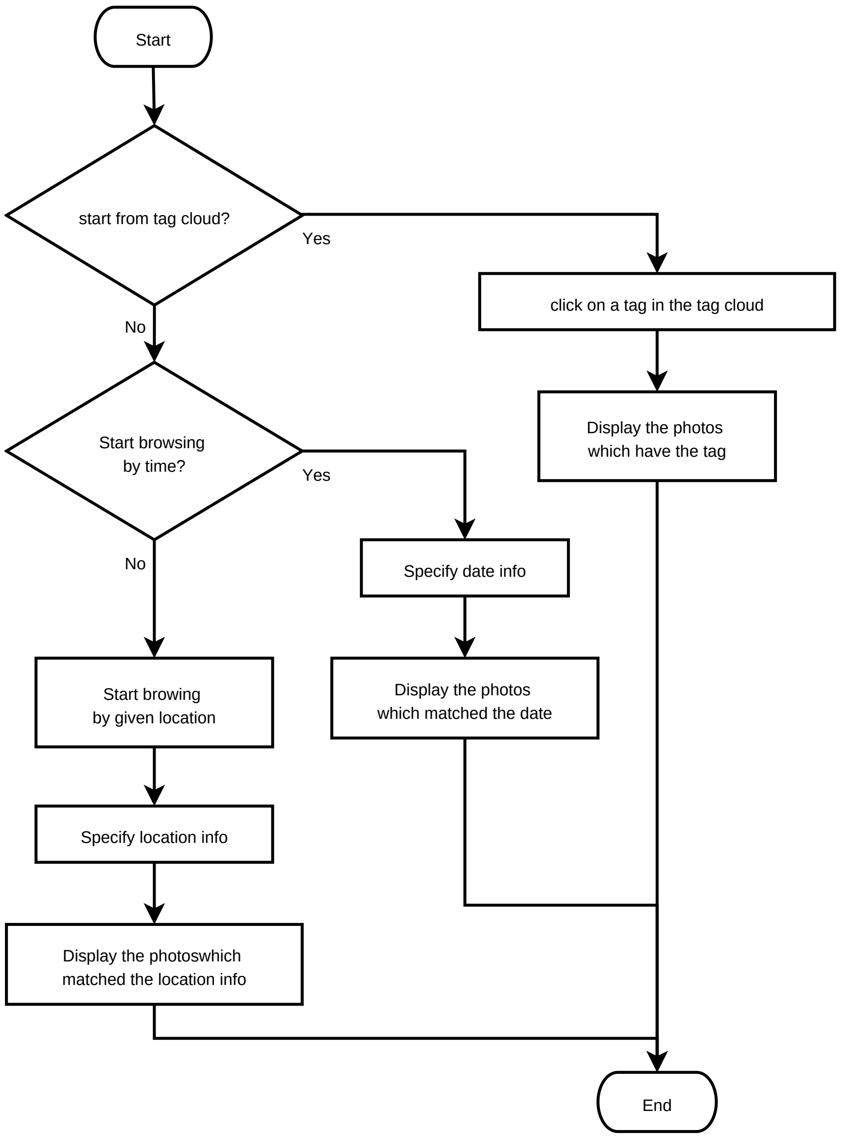

Start
start from tag cloud?
Yes
No
click on a tag in the tag cloud
Display the photos which have the tag
Start browsing by time?
Yes
No
Specify date info
Display the photos which matched the date
Start browing by given location
Specify location info
Display the photoswhich matched the location info
End

Figure 4.8: Activity Diagram of "Browse Photos" Use Case

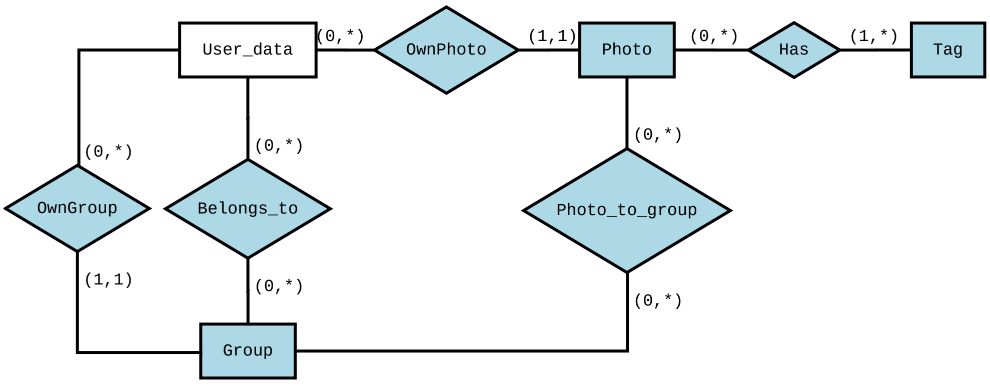


Figure 4.9: The ER Diagram of this Project

4.3.1 ER Diagram

This project is an extension of the current TIP system. In the Figure 4.9, we show only elements that are introduced in this project. For the entire ER diagram of the TIP system, see Appendix A.

As the Figure 4.9 shows, there are four entities and five relations involved in this project. Only entity *user_data* exists in the current TIP system and we did not make any change on it. It represents users and contents users' information. All other entities and relations are created for this project.

Entity *Photo* is for modeling photos. It contents information for each photo. The information includes photo title, description, capture time, upload time, GPS coordinates and access level. Access level allows user to set the photo to be *pubic* (everyone can see), *restricted* (the group members of the selected groups can see) and *private* (only owner can see).

Entity *Tag* is for modeling tags to allow users to tag their photos.

Entity *Group* is for modeling user's groups. A group owner can add/remove other users to/from the group. Groups are used to sign access permission to a photo. Where a photo's access level has been set as *restricted* and a group has been signed a permission to access the photo, all the group member in the group can access the photo. Other users, except the owner of the photo, can not see the photo.

Relationship *Has* is for modeling a photo has a tag. One photo can have zero, one or more than one tags. One tag has to be used by at least one photo. One tag can also be used by many photos.

Relationship *OwnGroup* is for modeling a user own a group. One user can have none, one or more than one groups. One group has to be owned by a user.

Relationship *OwnPhoto* is for modeling a user own a photo. This ownership relationship created when the user has uploaded the photo. One user can have none, one or more than one photos. One photo can only have and must have one owner.

Relationship *Belongs_to* is for modeling a user belongs to a group. One user can belong to none, one or more than one groups. One group can have none, one or more than one group members.

Relationship *Photo_to_group* is for modeling which groups can access a photo. A user can sign her/his group(s) to be able to her/his photo which the access level has been set as *restricted*. Then, all the group(s)' members can see the photo.

4.3.2 Relational Schema

We converted the ER diagram to the relational schema showing in the following.

Photo(pid, url, thumburl, location, title, description, capture_time, upload_time, access_level, internal, ownerid, width, height, thumbwidth, thumbheight): it combines entity *Photo* and relationship *OwnPhoto* in the ER diagram. The relationship *OwnPhoto* become an attribute ownerid in the relational schema of *Photo*. *Pid* is the primary key.

Tag(tid, name, num_of_photos): it converts from entity *Tag* in the ER diagram. Tid is the primary key.

Photo_group(gid, name, description, owner): it combines entity *Group* and relationship *OwnGroup* in the ER diagram. The relationship *OwnGroup* become an attribute *owner* in the relational schema of *Photo*. *Gid* is the primary key.

User_belongs_to_group(udid, gid): it converts from *Belongs_to* in the ER diagram. *Udid* and *gid* are the primary key.

Photo_to_tag(pid, tid): it converts from *Has* in the ER diagram. *Pid* and *id* are the primary key.

Photo_to_group(pid, gid): it converts from *Photo_to_group* in the ER diagram. *Pid* and *gid* are the primary key.

4.4 User Interface Design

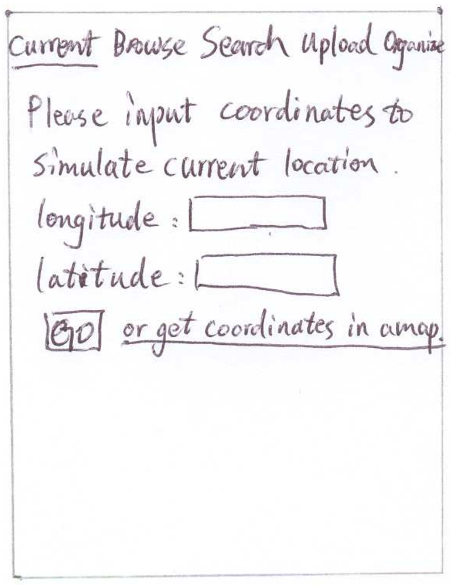
In the section, we describe our user interface design using a paper-based prototype. We show the paper prototype as scans of the images we have drawn in Figure 4.10 to 4.17. In each figure, there is a frame which represents the size of the normal PDA screen (240 pixels * 320 pixels). The words out side the frame means users need to scroll their browser to see all information. In each user interface, there are five taps to allow users to switch between five different functions which are *current*, *browse*, *upload*, *search* and *organize*. We will follow these five functions to describe the user interface design.

4.4.1 Current Location Photos

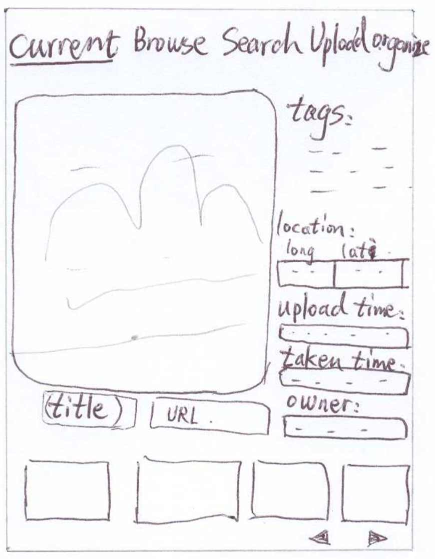
Users can find a link called *Photo Gallery* on the TIP main menu page. We recommend users to log into TIP first. After users logged into TIP, they can also find a link called *Photo Gallery* on the logged in user menu page, which is main enter point of the photo gallery. The difference of logged in access and unlogged in access is that without login, users can only access public photos. However, if a user logged in, s/he can access public photos and her/his authorized photos. We will discuss more detail of privacy control in *upload photos*.

After a user entered the *photo gallery* or clicked on *current* tap, s/he will face the page that is shown in Figure 4.10(a). We use this page to simulate user's current location (GPS coordinates). Users can either type in GPS coordinates manually or click on the link called *or get coordinates in a map*. This map page embeds google map, which gives users better experience to find a location.

When the user input coordinates and click on button *go*, Figure 4.10(b) shows up. In the left top, it shows one of photos, which nearby current location. The user can also click on the photo to open this photo in a new page in full size. All tags which the photo had are displayed beside of the photo. Other information such as longitude, latitude, upload time, taken time, title and URL are also displayed. At the bottom of this page, a list of photo thumbnails display all photos which are taken in current location and the current user is authorized to view. If the user has not logged in, s/he can only see public photos.


Current Browse Search Upload Organize
Please input coordinates to simulate current location.
longitude :
latitude :
GO
or get coordinates in a map

(a) Simulate Current Location



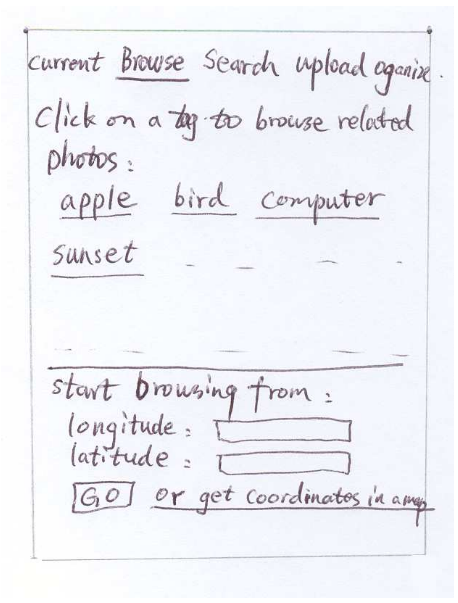
(b) Display Current Location Photos

Figure 4.10: Current Location Photos

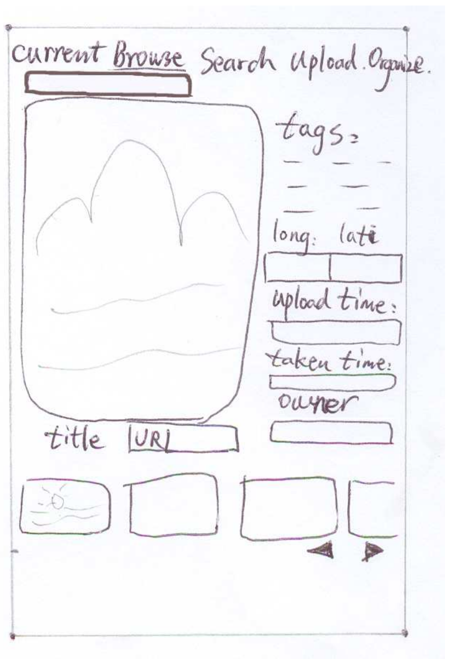
4.4.2 Browsing Photos

When users click on tab *browse* on any page, the *start browsing page* (see Figure 4.11(a)) will show up. This page has two parts. The top part is a tag cloud which represent most popular tags are used in the system. The most popular tags are in the tag cloud. The font size of these tags represent popularity of each tag. In other words, the bigger of the font size, the more popular of the tag. These tags are ordered by alphabet. The bottom part allows user to specify a location (GPS coordinates) as a start point to browsing photos, which is similar to Figure 4.10(a). Users can click on one tag in the tag cloud to browsing photos which have that tag. Related photos will display in the Figure 4.11(b). If the user wants to input coordinates and start browsing photos, the related photos will display in the Figure 4.11(b).

In Figure 4.11(b), there is a small frame under the five tabs. The frame is used for displaying current browsing information called information bar. For example, a user clicked on tag *apple* on the tag cloud in the Figure 4.11(a), so all photos display here have a tag *apple* and the information bar will display as *tag: apple*. If the user input coordinates and browse photos, the information bar will display the coordinates that user provided.


current Browse Search upload oganize.
Click on a tag to browse related photos:
apple bird computer
sunset
start browsing from:
longitude:
latitude:
GO or get coordinates in a map

(a) Start Browsing Page



(b) Displaying Photos Page

Figure 4.11: Browsing Photos

4.4.3 Search Photos

The interface shown in Figure 4.12 provides the search function to users. It allows users to search photos by tag, a time range of the photo taken time, coordinates and owner of the photo.

A user is able to specify a time frame for searching. For example, if the user wants to search for photos taken in 2005, s/he could select 2005 in both *start time* and *end time* and leave months and days unchanged. The search results will be all 2005's photos. The user is also able to only provide *start time* or *end time* for searching. The user can manually input coordinates or pick one in a map, which is similar to location simulation interface Figure 4.10(a).

The search results will display using interface 4.11(b), the same as for browsing.

4.4.4 Upload Photos

Figure 4.13 shows the process of photo uploading which include two steps.

The first step (see Figure 4.13(a)) is upload a photo file to the TIP server

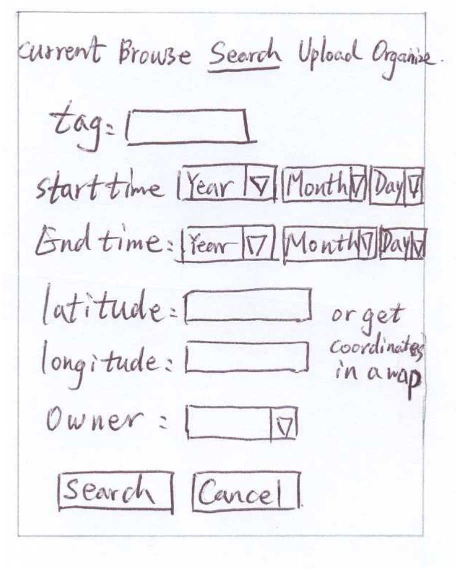
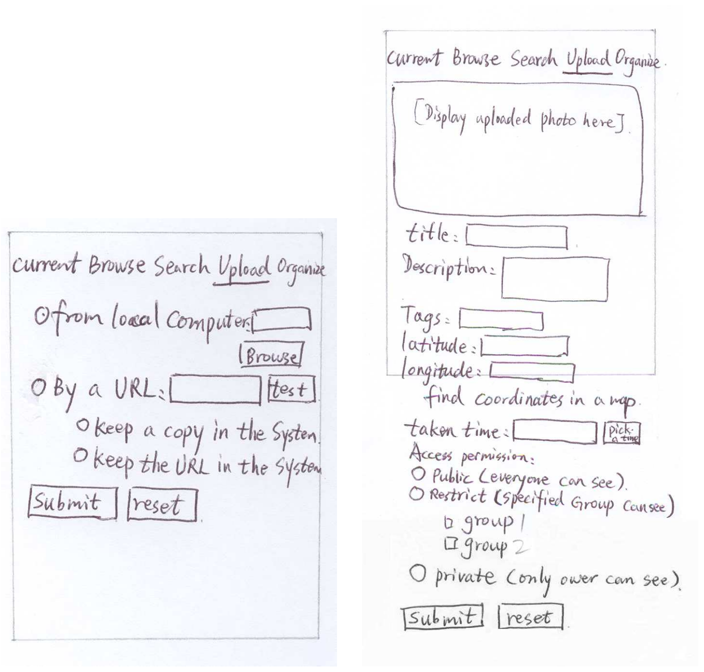

Current Browse Search Upload Organise
tag:
start time Year Month Day
End time: Year Month Day
latitude:
longitude:
or get coordinates in a map
Owner:
Search
Cancel

Figure 4.12: Search for Photos

or specify a URL of the photo. A user can specify a photo in local computer and upload them to the TIP server. Alternatively, the user can indicate a URL of a photo and use it in the TIP server. For example, if a user has uploaded a photo to her/his blog, s/he does not need to upload the photo again. S/he can use the URL of the photo in the TIP. The Figure 4.13(a) also provides a handy button for users to test if the URL is working. For some reason, some users may want to keep a copy of the photo in the TIP system even they have already got a URL. To do this, they can select *keep a copy in the system* in the Figure 4.13(a).

After the user clicked on *submit* button. The system will attempt to retrieve GPS coordinates and taken time from the photo metadata Eixf and move to the second step (see Figure 4.13(b)). The uploaded photo will be displayed in the top. The user is asked to type in title and description. If the user does not want to input, s/he can just leave it. We highly recommended users to input photo content related words as tags, since we are mainly use tags to categorize photos. We provide *a popular tags recommendation* to encourage users to use poplar tags, which pop up a list of tags while a user is typing. Besides of each tag, a number indicates the number of photos have this tag. If the system retrieved GPS coordinates and photo taken time, the system will fill in the input box automatically. Otherwise, the user can type in GPS coordinates manually or find the location in a map and fill in the taken time by click on the *pick a time* button.


Current Browse Search Upload Organize
O from local Computer
Browse
O By a URL:
test
O keep a copy in the System
O keep the URL in the System
Submit
reset
Current Browse Search Upload Organize
[Display uploaded photo here]
title:
Description:
Tags:
latitude:
longitude:
find coordinates in a map
taken time:
pick a time
Access permission:
O Public (everyone can see)
O Restrict (specified Group can see)
group 1
group 2
O private (only ower can see)
Submit
reset

(a) Upload Photo File (b) input information for the photo

Figure 4.13: Upload a Photo

The user can set an access permission on the photo. The access permission includes three levels: *public*, *restricted* and *private*. The default for each photo is *public*. A *public* photo can be seen by everyone even without logged in. When the user set the photo as *restricted*, s/he need to specify which group can access this photo by tick on the group from group list. All ticked groups' members can access the photo. *Private* is used when the user do not want the photo to be seen by others and only himself/herself can see the photo.

4.4.5 Organize Photos and Groups

Figure 4.14 shows the main menu page of the organize. It can be accessed by click tab *organize* in the top of each page. There are three links that users can organize their photos and groups, which are *organize my photos*, *create a group* and *modify groups*.

When a user clicks on link *organize my photos*, a photo list will display in Figure 4.15(a). The user can either click on a photo to start modifying

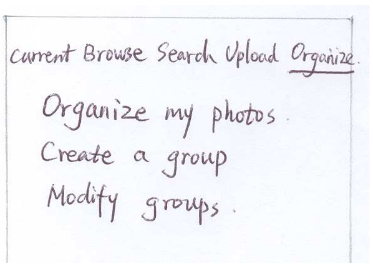


Figure 4.14: Organize Menu

photo's detail (show in Figure 4.15(b)) or click on the delete button beside the photo to delete it.

Figure 4.15(b) shows a page which allows users to modify their photos' details, which include *title*, *description*, *tags*, *latitude*, *longitude*, *taken time* and *access permission*. The page structure is similar to the photo upload page.

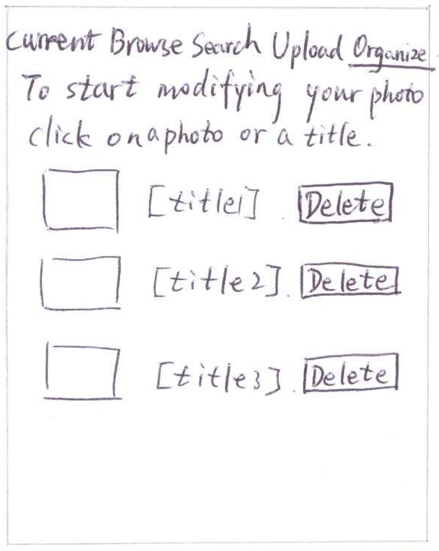
When a user clicks on *create a group* in Figure 4.14, a page (Figure 4.16) will show up. It asks the user to provide group *name* and *description*. It allows the user to search for other users and add them to this group. After successfully creating a group, the user can tick this group to allow this group members to access their restricted photos (see Figure 4.13(b)).

While a user click on *modify groups* in organize menu page (Figure 4.14), a page (Figure 4.17(a)) will show up. The user can either click on a group to start modifying group's detail (show in Figure 4.17(b)) or click on the delete button beside the group to delete it.

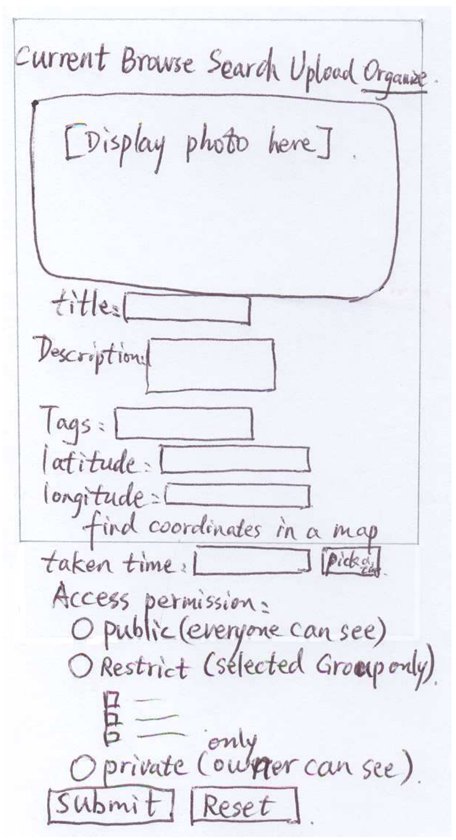
The page *modify a group* (Figure 4.17(b)) allows user to modify group detail which includes group name, description and group member.

4.5 Conclusion

In this chapter, in order to achieve the six user requirements, we analyzed our six user requirements and gave general design considerations to each of the user requirements. Then, we provided eight use cases to represent users' interactions with the system. UML activity diagrams are also given to describe the eight user cases. Base on the eight use cases, we designed database structure which we used ER diagram and relation schema to describe. Base on the eight use cases, we using paper based prototype to describe user interface design. In the following chapter, we will introduce the implementation according to the design.


Current Browse Search Upload Organize
To start modifying your photo click on a photo or a title.
[title1] Delete
[title2] Delete
[title3] Delete

(a) Select a Photo to Modify


Current Browse Search Upload Organize
[Display photo here]
title:
Description:
Tags:
latitude:
longitude:
find coordinates in a map
taken time:
Pick a date
Access permission:
public (everyone can see)
Restrict (selected Group only)
private (owner only can see)
Submit
Reset

(b) Modify Photo Information

Figure 4.15: Organize Photos

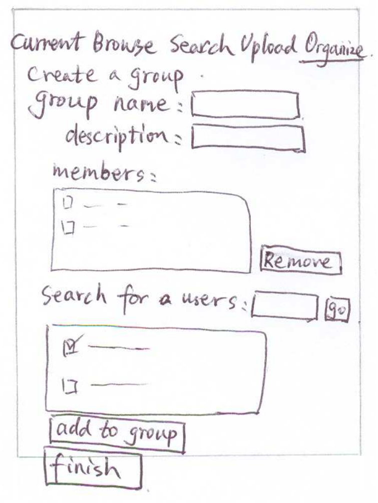
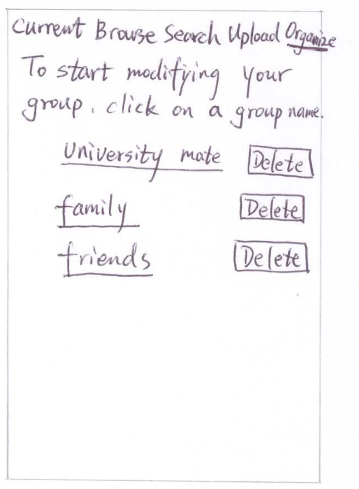
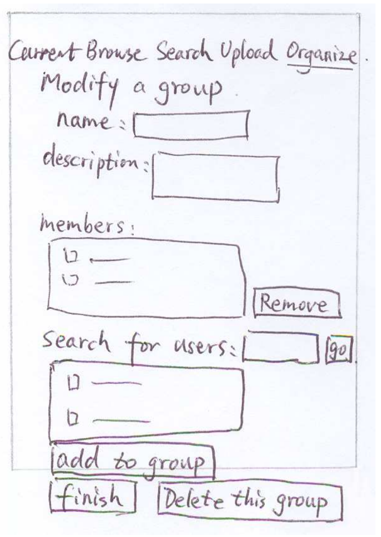


Figure 4.16: Create a Group


Current Browse Search Upload Organize
To start modifying your group, click on a group name.
University mate
Delete
family
Delete
friends
Delete

(a) Choose a Group to Modify



(b) Modify a Group

Figure 4.17: Organize Groups

Chapter 5

Implementation

In this chapter, we describe the implementation in detail. First of all, we introduce the implementation techniques that we have used in the TIP system. Furthermore, we introduce the software architecture of this project. Then, we describe the implemented database by giving the table definitions. Finally, we describe the software implementation by giving the class diagrams.

5.1 Implementation Techniques in the TIP System

The current TIP (TIP2.5) database is created as a central database approach by PostgreSQL 7.4.7 and postgis 0.7.5. PostgreSQL is an open source relational database system. PostGIS adds support for geographic objects to the PostgreSQL object-relational database.

Software part is implemented under the Struts framework which provides an implementation for the MVC structure. The MVC structure separates a web application into three components: controller, model and view. The benefit of the MVC structure is that the business rules and the presentation are independent so that the business rules (Model) and the presentation (View) can be implemented separately. The controller connects those two components. In the TIP system, the model component is implemented by Java Standard Edition 1.5.0_10 and the view component is implemented in JSP. The TIP system server is running on the Jakarta Tomcat 5.5. The implementation of TIP is shown in Figure 5.2.

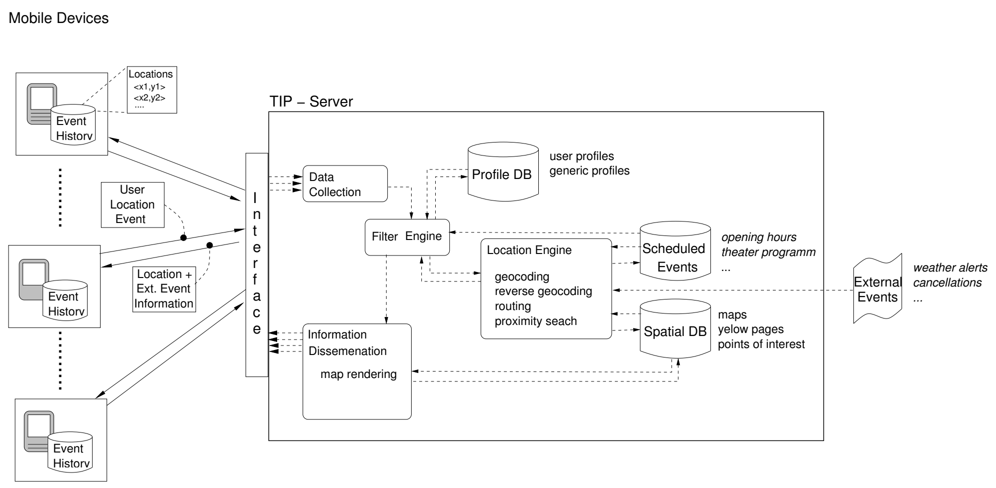


Figure 5.1: TIP Architecture [9]

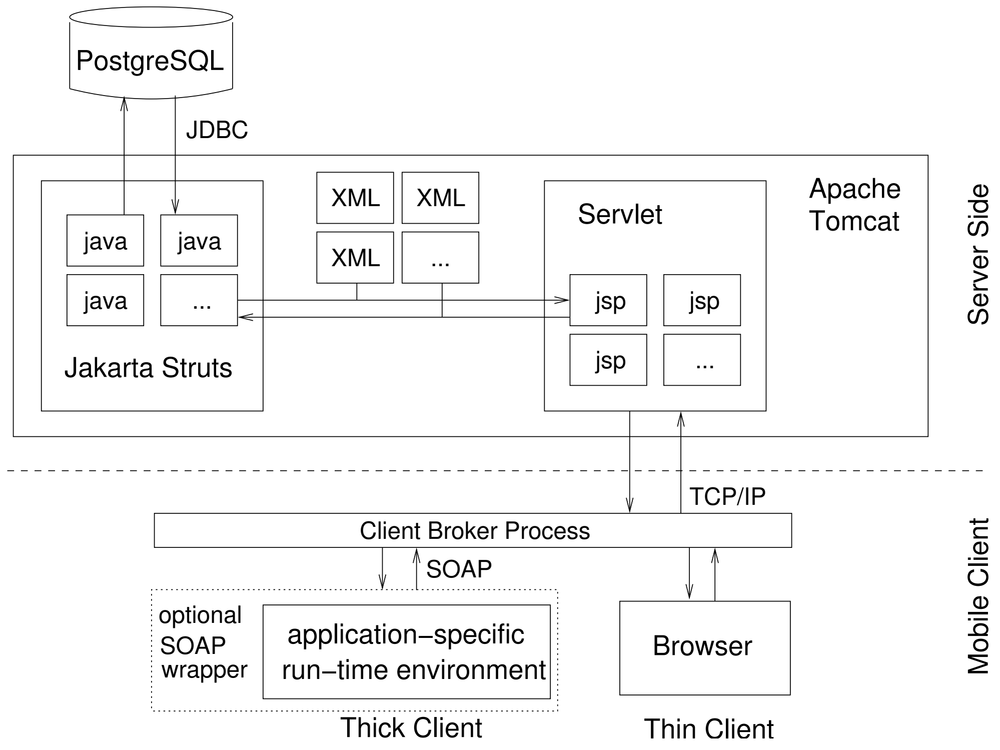


Figure 5.2: Implementation of TIP [7]

On the client side, JavaScript has been used to reduce server access and increase interaction with users. A suggestion of tag names is given to the user while a user is typing a tag, which implemented by JavaScript. A time picker is also implemented by JavaScript.

5.2 Software Architecture

Figure 5.3 is the architecture of the TIP component model. The architecture of the photo gallery component is highlighted in the dashed frame. The photo gallery component involves six elements which are *photo gallery manager*, *data input/update*, *search engine*, *location engine*, *privacy controller* and *photo repository*.

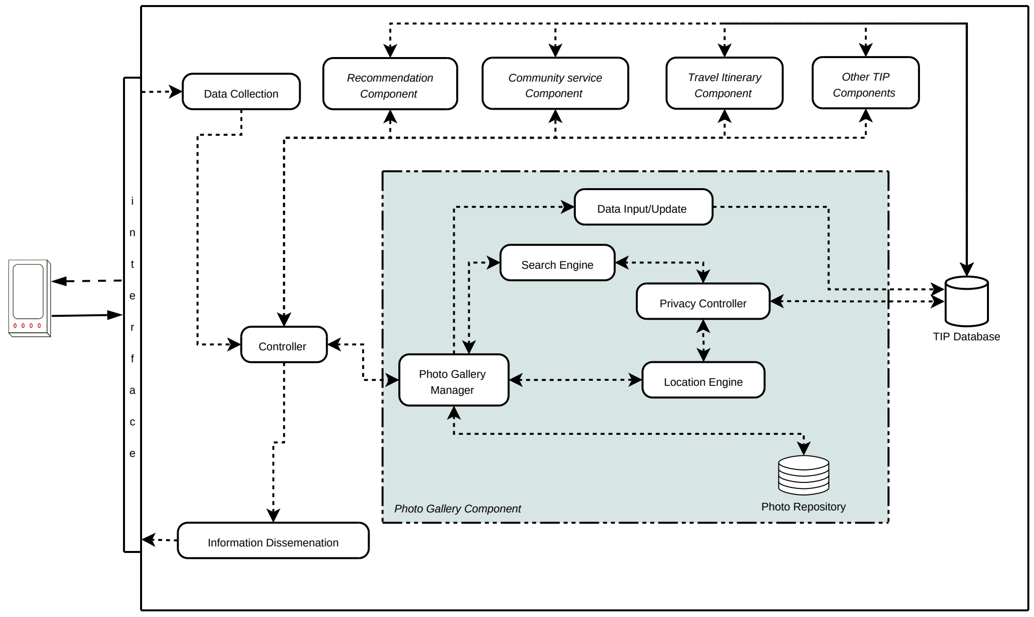


Figure 5.3: Architecture of the TIP Component Model

Photo gallery manager element works as a manager in photo gallery component. It receives a user's request which come from the user's web browser through *interface*, *data collection* and *controller*. *Photo gallery manager* analyzes the user's request and signs different tasks to different elements depends on the request type.

If the user is on *living mode*, the *manager* will pass the user's request to the *location engine*. The *location engine* retrieves user's current location and user id from the request and generate a database query to retrieve nearby photos. In order to make sure the user only see the photos which s/he is allowed to view, the *location engine* passes the database query to the *privacy controller*. The *privacy controller* modifies the database query statement and ensure the query statement only queries the photos which the user has a permission to access. After the query executed successfully, the result photos' information or a no photo found message will pass back to the *manger* via *location engine*.

If the user is on *browsing mode* or doing a search, the *manager* will pass the user's request to the *search engine*. The *search engine* will depend on the content of the request to generate different database queries statement to retrieve related photos. For privacy reasons, the search engine passes the query statement to the *privacy controller*. The *privacy controller* modifies the query in order to ensure the current user only get photos which s/he is allowed to view. After the query successfully executed, the query results photos' information or a no photo found message will be passed to the *manger* via *search engine*.

If the user is uploading a photo, the *manager* will store the uploaded photo into the *photo repository*. The *photo repository* is a fold, which contains all the photos that user uploaded, in the TIP web server. The TIP web server provides an URL for each photo stored in the *photo repository*. The manager passes the photo URL and user request to *data input/update* element. The *data input/upload* element stores the information into TIP database.

If the user is creating or modifying a group, modifying a photo or other data inputting or modifying, the manager will pass the user's request direct to the *data input/upload* element. The *data input/upload* element accesses TIP database and complete the work.

When the *manager* receives photos' information or a no photos found message from *location engine* or *search engine*, it will pass the photos' information or the no photo found message to the *controller*. The controller passes them to *information dissemination* and the *information dissemination* send them to the user's browser via *interface*.

5.3 Implemented Database: Table Definition

In this section, we will describe the tables that we have created in the database. All tables are according to the relational schema which we discussed in Chapter 4.

Definition of Table *Photo*:

Column	Type	Modifiers
pid	integer	not null default nextval('pid_seq'::text)
url	text	
thumburl	text	
location	geometry	
title	text	
description	text	
capture_time	timestamp without time zone	
upload_time	timestamp without time zone	
access_level	smallint	not null default 0
internal	boolean	default true
ownerid	integer	
width	integer	default 240
height	integer	default 160
thumbwidth	integer	default 20
thumbheight	integer	default 20

Table 5.1: Photo Table

Indexes:
"photo_PK" primary key, btree (pid)
"location_idx" gist (location)
"owner_id_idx" btree (ownerid)
"capture_time_idx" btree (capture_time)
Foreignkey constraints:
"owner_FK" FOREIGN KEY (ownerid) REFERENCES user_data(ud_id)

Definition of Table *Tag*:

Column	Type	Modifiers
tid	integer	not null default nextval('tid_seq'::text)
name	text	not null
num_of_photos	integer	default 0

Table 5.2: Tag Table

Indexes:
"PK_tag" primary key, btree (tid)

Definition of Table *Photo_group*:

Column	Type	Modifiers
gid	integer	not null default nextval('gid_seq'::text)
name	text	not null
description	text	
owner	integer	not null

Table 5.3: Photo_group Table

Indexes:
"pid_pk" primary key, btree (gid)
Foreign-key constraints:
"FK_group_owner" FOREIGN KEY ("owner") REFERENCES user_data(ud_id)

Definition of Table *User_belongs_to_group*:

Indexes:
"user_belongs_to_group_PK" primary key, btree (udid, gid)

Column	Type	Modifiers
udid	integer	not null
gid	integer	not null

Table 5.4: User_belongs_to_group Table

Foreign-key constraints:
"user_data_FK" FOREIGN KEY (udid) REFERENCES user_data(ud_id)
"group_FK" FOREIGN KEY (gid) REFERENCES photo_group(gid)

Definition of Table *Photo_to_group*:

Column	Type	Modifiers
gid	integer	not null
pid	integer	not null

Table 5.5: Photo_to_group Table

Indexes:
"photo_to_group_PK" primary key, btree (gid, pid)
Foreign-key constraints:
"photo_FK" FOREIGN KEY (pid) REFERENCES photo(pid)
"group_FK" FOREIGN KEY (gid) REFERENCES photo_group(gid)

Definition of Table *Photo_to_tag*:

Column	Type	Modifiers
pid	integer	not null
tid	integer	not null

Table 5.6: Photo_to_tag Table

Indexes:
"photo_to_tag_pk" primary key, btree (pid, tid)
Foreign-key constraints:
"tid_fk" FOREIGN KEY (tid) REFERENCES tag(tid)
"pid_fk" FOREIGN KEY (pid) REFERENCES photo(pid)

5.4 Software Implementation

In this section, we first discuss an overview class diagram of the photo gallery component (showing in Figure 5.4). Then we use *search photos class diagram* as an example to describe the detail of implementation (shows in Figure 5.5). We also use *search photos sequence diagram* (shows in Figure 5.6) to describe action flow in the implementation.

Figure 5.4 is the overview of the photo gallery component class diagram. It shows *Action* classes, *Logic* classes, *data* classes and *container* classes that we have implemented in the photo gallery component. All *Action* classes are inherited from *org.apache.struts.action.Action*. *Logic* classes: *PhotoLogic*, *GroupLogic* and *TagLogic* are responsible for accessing database. All data retrieved from database are stored in *data* classes. *Container* classes are the containers of the *data* classes.

According to the functions that *photo gallery* component provided, all *action* classes can be divided into four groups: *browsing*, *searching*, *uploading* and *organizing*. *BrowsePhotoAction*, *BrowsePhotosStartAction* and *DisplayPhotoAction* are responsible for *browsing*. *Searching* involves action classes: *SearchPhotosAction*, *DisplayPhotoAction* and *SetupPhotoAction*. Both *browsing* and *searching* are sharing *DisplayPhotoAction* to display photos to users. *Uploading* involves *action* classes: *SetupUploadPhotoAction*, *UploadPhotoAction* and *UploadPhoto2Action*. Organizing involves six action classes: *OrganizePhotosAction*, *SetupModifyPhotoAction*, *ModifyPhotoAction*, *ModifyGroupAction*, *CreateGroupAction* and *ModifyGroupAction*.

In order to describe the implementation in detail, we choose *search photos* as an example to explain the implementation. Figure 5.5 is the *search photos* class diagram which shows all classes involved in the *search photos* function. The elements inside the dash line frame are *struts controller*. SearchPhotoForm class inherits ActionForm class. It contains search conditions which submitted by a user. SearchPhotosAction inherits *Action* class. *PhotoVO* class is a data class which contains photo information. *PhotoVOContainer*

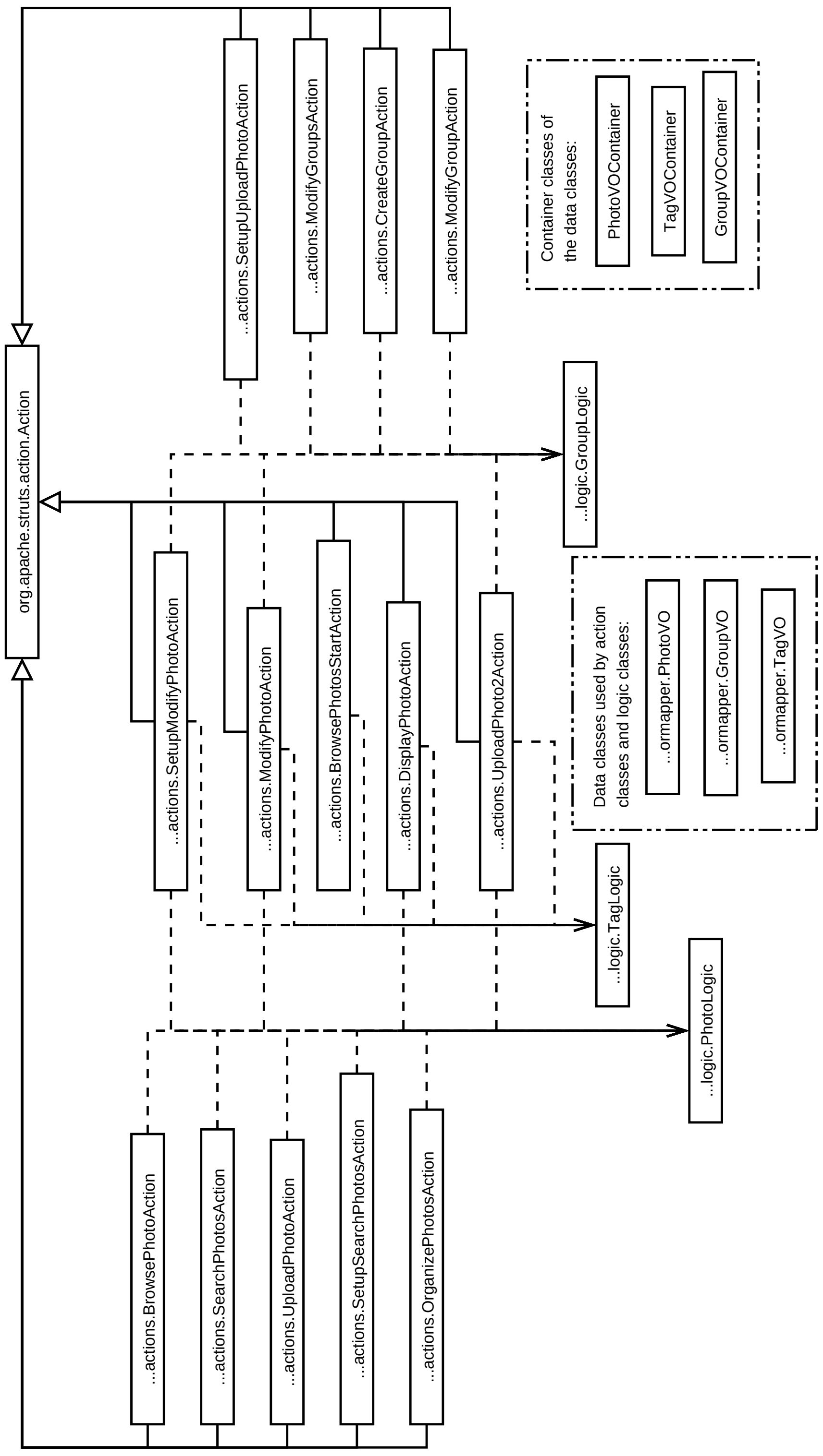

org.apache.struts.action.Action
...actions.SetupUploadPhotoAction
...actions.ModifyGroupsAction
...actions.CreateGroupAction
...actions.ModifyGroupAction
Container classes of the data classes:
PhotoVOContainer
TagVOContainer
GroupVOContainer
...logic.GroupLogic
...actions.SetupModifyPhotoAction
...actions.ModifyPhotoAction
...actions.BrowsePhotosStartAction
...actions.DisplayPhotoAction
...actions.UploadPhoto2Action
Data classes used by action classes and logic classes:
...ormapper.PhotoVO
...ormapper.GroupVO
...ormapper.TagVO
...logic.TagLogic
...logic.PhotoLogic
...actions.BrowsePhotoAction
...actions.SearchPhotosAction
...actions.UploadPhotoAction
...actions.SetupSearchPhotosAction
...actions.OrganizePhotosAction

Figure 5.4: Overview of the Photo Gallery Component Class Diagram

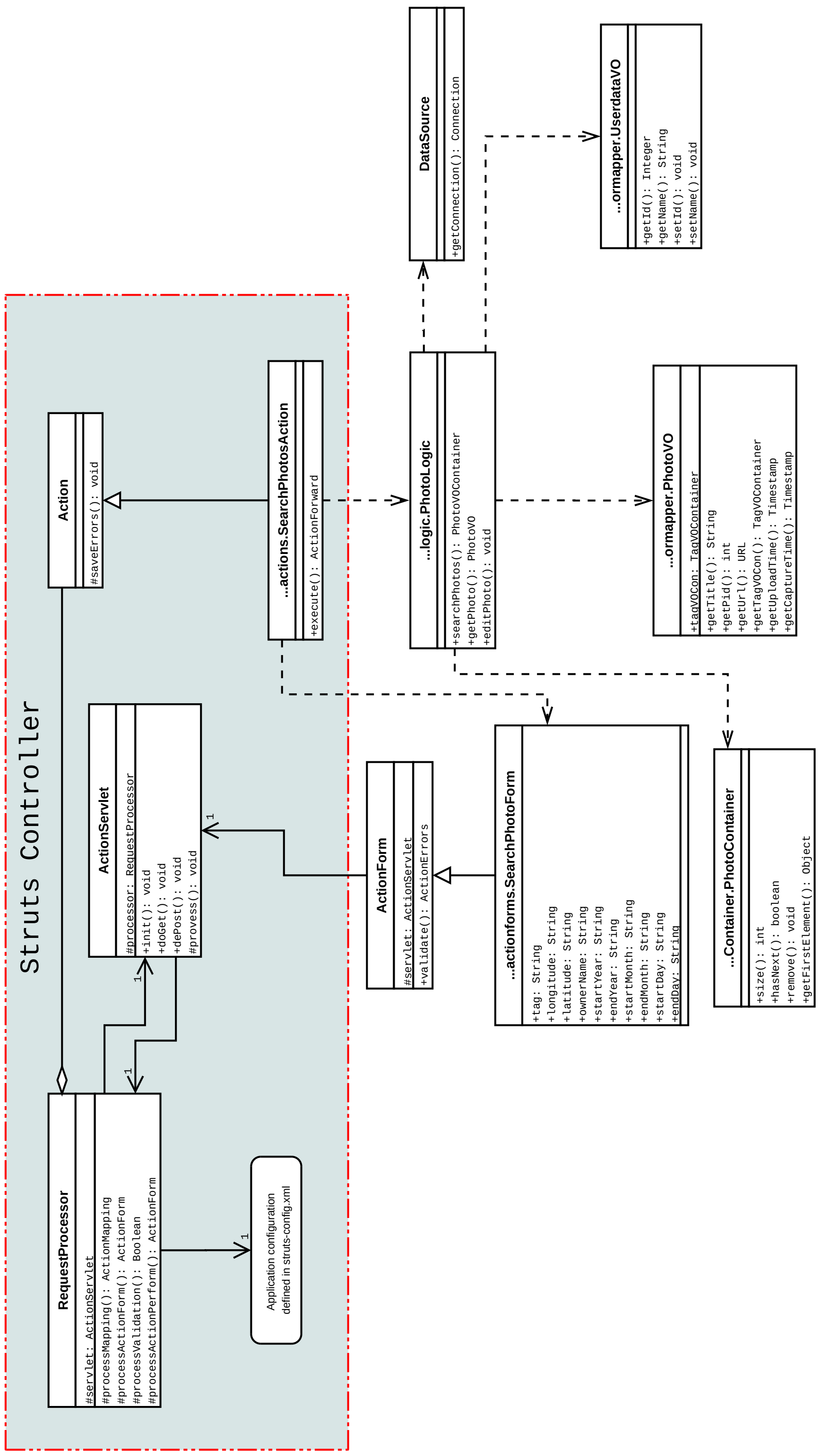


Figure 5.5: Detail of the *Search Photos* Class Diagram

class is a container class which contains *photoVO* classes. *DataSource* class provides database connections. *UserdataVO* class contains the user's information.

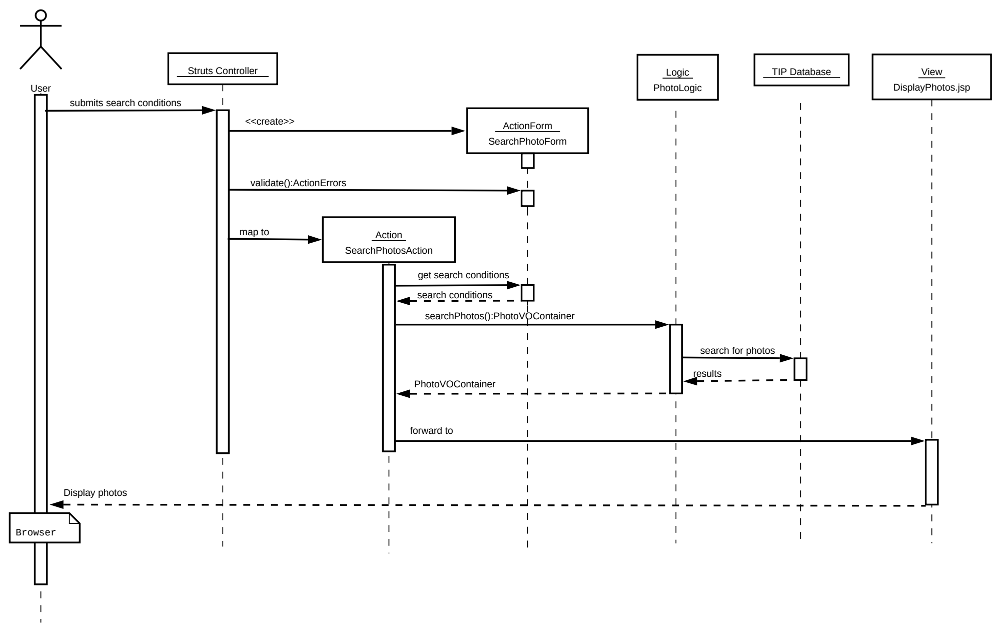


Figure 5.6: *Search Photos* Sequence Diagram

Figure 5.6 shows the sequence of the *search photos* interaction. The *search photos* process runs as follows:

1. A user submits a photo search request from her/his browser to the *struts controller.*

2. The *struts controller* creates a *ActionForm* called *SearchPhotoForm* to store users' search conditions.

3. *Struts controller* validates *SearchPhotoForm* to ensure user's search conditions are valid. If the any search condition is invalid, a *ActionErrors* object is returned.

4. According to *struts-config.xml* config file, *struts controller* maps to *Action* called *SearchPhotoAction.*

5. *SearchPhotoAction* retrieves search conditions from action form *SearchPhotoForm.*

6. *SearchPhotoAction* sends a message to PhotoLogic and asks it to search photos from TIP database.
7. *PhotoLogic* query TIP database with search conditions and return query results as a *PhotoVOContainer* back to *SearchPhotoAction*.
8. *SearchPhotoAction* forwards to *DisplayPhotos.jsp* page to display the search results.
9. *DisplayPhotos.jsp* page display the search results in the user's browser.

5.5 Conclusion

To sum up, in this chapter, we introduced the implementation techniques of the current TIP(TIP2.5) used. TIP database is created as a central database approach by PostgreSQL 7.4.7 and postgis 0.7.5 spatial database extension for PostgreSQL. Software part is implemented under the Struts framework which provides an implementation for the MVC structure. The TIP system server is running on the Jakarta Tomcat 5.5.

We used Figure 5.3 to describe the architecture of the TIP component model and the photo gallery component. The photo gallery component involves six elements which are *photo gallery manager*, *data input/update*, *search engine*, *location engine*, *privacy controller* and *photo repository*.

We described implemented database by given table definition. We also described software implementation by given the overview of the photo gallery component class diagram showed in Figure 5.4. In order to explain the implementation in detail, we chose *search photos* interactions a representative and used to explain the implementation.

In the next chapter, we will evaluate how we achieved each user requirement in our implementation.

Chapter 6

Evaluation

In this chapter, we will evaluate our implementation. According to our user requirements that we discussed in Chapter 2, we consider both qualitative evaluation and quantitative evaluation.

6.1 Qualitative Evaluation

In this section, we evaluate functionality of the photo gallery. Eight test cases are defined to match the six user requirements which we discussed in Section 2.2. The user requirements are given in the following list:

UR1 Categorize photos.

UR2 Privacy control.

UR3 Viewing photos by current location.

UR4 Browsing photos by given locations.

UR5 Searching for location or event related photos.

UR6 Reusing photos between the online photo gallery, Blog and other places which provide photos as URLs.

The following eight test cases are defined base on user requirements we listed above. We will use the screen shots taken in the TIP system to show the process of each test case.

Test case 1: Evaluate if a user can categorize photos.

Test case 2: Evaluate if a private photo can be seen by other users.

Test case 3: Evaluate if only the user can see a photo when s/he is the owner of the photo or belongs to a group which have signed to the photo and the photo has an *access level* of *restricted*

Test case 4: Evaluate if the system can display the current location photos.

Test case 5: Evaluate if a user can browse photos by given a location.

Test case 6: Evaluate if a user can search for photos by tags or a location.

Test case 7: Evaluate if a user can get the URL of the photo easily in order to reuse it in other place.

Test case 8: Evaluate if the system allow a user to upload a photo with an URL.

6.1.1 Test Setting

Photo	Location	Latitude	Longitude	Tags
p1	Hamilton lake	-37.793217	175.271113	Hamilton, lake
p2	Hamilton lake	-37.793223	175.271109	Hamilton, lake
p3	Hamilton lake	-37.793231	175.271124	Hamilton, lake
p4	Hamilton lake	-37.793215	175.271121	Hamilton, lake, ducks
p5	Hamilton lake	-37.793238	175.271116	Hamilton, lake, ducks
p6	Hamilton gardens	-37.803659	175.308281	Hamilton, garden
p7	Hamilton gardens	-37.803662	175.308278	Hamilton, garden
p8	Hamilton gardens	-37.803663	175.308280	Hamilton, garden

Table 6.1: Test Photos

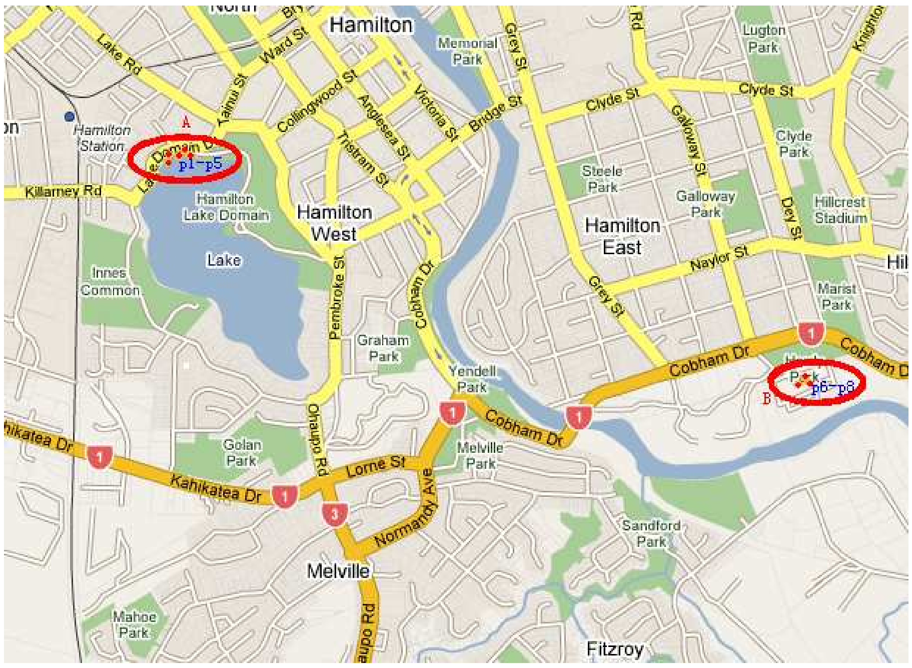


Figure 6.1: Location of photos

Figure 6.1 shows the location of the eight photos (p1-p8) we used in the evaluation. Five dots in oval A, which represent photo p1 to p5 that have been taken around *Hamilton lake*. Three dots in oval B, which represent photo p6, p7 and p8, are taken in *Hamilton gardens*. The information about these eight photos are shown in Table 6.1. All these photos have an *access level* of *public* (everyone can see). Table 6.1 shows the information of the photo p1 to p8.

Photo	Photo owner	Tags	Access level	Signed to group
p9	u1	secret	private	-
p10	u1	secret	restricted	g1

Table 6.2: Secret Photos

For privacy control test, we create three users u1, u2, u3 and we use u1 to create group g1. We add u2 to g1. We use u1 to upload a photo called p9 and set an *access level* of *private* to it. We use u1 to upload another photo called p10 and set an *access level* of *restricted* to p10 and signed p10

Group	Owner	Group member
g1	u1	u2

Table 6.3: Group Information

to g1. A *restricted* photo can only be accessed by the photo owner and the group members, which the group is signed a permission to the photo. A *private* photo can only be accessed by the photo owner. Table 6.2 shows the information of the secret photos and Table 6.3 shows the group information. Since location information does not influence privacy control test, we ignored location information of photo p9 and p10.

6.1.2 Hypotheses and Results

In this section, we give a hypothesis for each test case and then give screen shots we taken in the TIP system to represent test results.

Test case 1: Evaluate if a user can categorize photos. Since we are using tags to categorize photos, categorizing process depends on the tags what users used. Photos have same tag are grouped together. In our test data (see Table 6.1), there are ten photos and five tags are involved. As we can see from Table 6.1 and Table 6.2, eight photos have a tag *Hamilton*. It is the most popular tag.

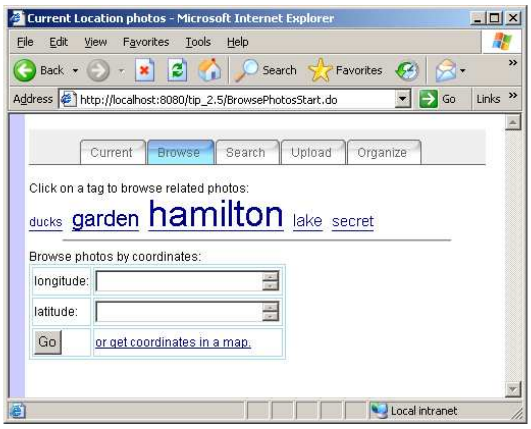
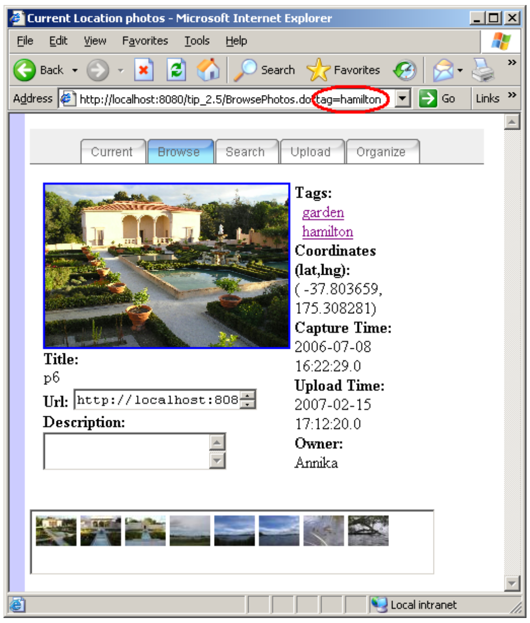

Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/tip_2.5/BrowsePhotosStart.do Go Links
Current Browse Search Upload Organize
Click on a tag to browse related photos:
ducks garden hamilton lake secret
Browse photos by coordinates:
longitude:
latitude:
Go or get coordinates in a map.
Local intranet

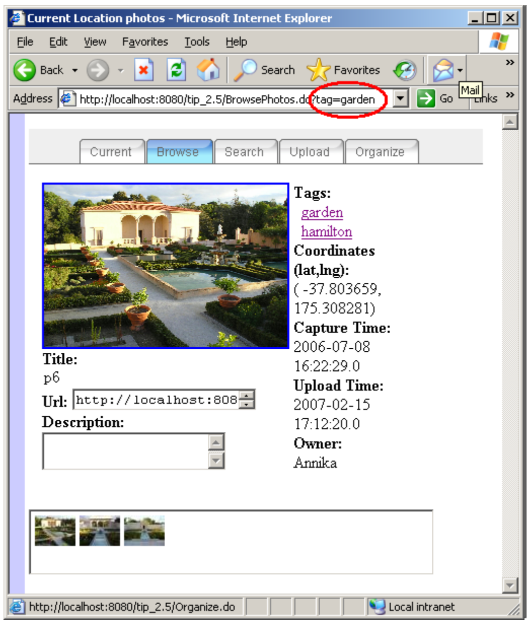
Figure 6.2: Start Browsing Page


Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/tip_2.5/BrowsePhotos.do?tag=hamilton
Go Links
Current Browse Search Upload Organize
Local intranet

Title:
p6
Url: http://localhost:808
Description:

Tags:
garden
hamilton
Coordinates (lat,lng):
(-37.803659, 175.308281)
Capture Time:
2006-07-08 16:22:29.0
Upload Time:
2007-02-15 17:12:20.0
Owner:
Annika

Figure 6.3: Photos with a Tag *Hamilton*


Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Mail
Address http://localhost:8080/tip_2.5/BrowsePhotos.do?tag=garden
Go Links
Current Browse Search Upload Organize
http://localhost:8080/tip_2.5/Organize.do
Local intranet

Title:
p6
Url: http://localhost:808
Description:

Tags:
garden
hamilton
Coordinates (lat,lng):
(-37.803659, 175.308281)
Capture Time:
2006-07-08 16:22:29.0
Upload Time:
2007-02-15 17:12:20.0
Owner:
Annika

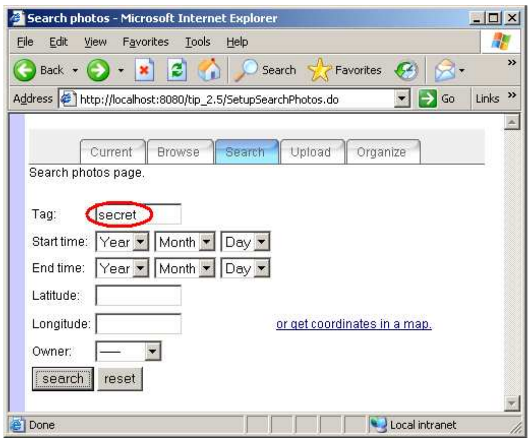
Figure 6.4: Photos with a Tag *garden*

Results: In Figure 6.2, we can see all five tags have been displayed in different font size. The font size represents the popularity of the tag used. The higher popularity, the bigger of the font size. It shows the tag *Hamilton* in the biggest font size. By clicking on the tag *Hamilton*, all photos with a tag *Hamilton* are display in *photo display* page (see Figure 6.3). A user can also click on any one of current displaying photo's tags to browse that tag related photos. For example, we click on tag *garden* on the page showing in Figure 6.3. Then the tag garden related photos are displayed (see Figure 6.4).

Test case 2: Evaluate if a private photo can be seen by other users. We use user u3 to log into the TIP system and search for tag *secret*. Since we have only uploaded two photos p9 and p10 which have a tag *secret* and

Hypotheses: User u3 found no photos after s/he searched for tag *secret*.

Results: Figure 6.5 shows the search interface and Figure 6.6 shows no photos that the search returned. The result shows that we have successfully disallowed unauthorized users to access *private* and *restricted* photos.


Search photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/tip_2.5/SetupSearchPhotos.do Go Links
Current Browse Search Upload Organize
search reset
or get coordinates in a map.
Done Local intranet

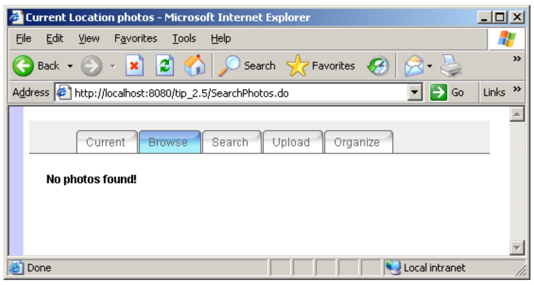
Search photos page.

Tag: secret
Start time: Year Month Day
End time: Year Month Day
Latitude:
Longitude:
Owner: —

Figure 6.5: Search for *secret*

Test case 3: Evaluate if only the user can see a photo when s/he is the owner of the photo or belongs to a group which have signed to the photo and the photo has an *access level* of *restricted*. Since test case 2 has approved that we have successfully disallowed unauthorized users to access *private* and *restricted* photos, we still need to approve that an authorized user can access *private* and *restricted* photos. We will log into the TIP system as u1 and u2 respectively and search for photos with a *secret* tag.

Hypothesis: User u1 can see photo p9 and p10. User u2 can only see

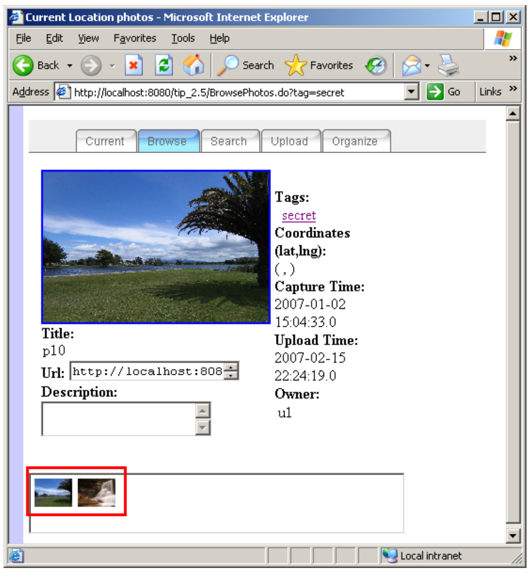

Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/tip_2.5/SearchPhotos.do Go Links
Current Browse Search Upload Organize
Done
Local intranet

No photos found!

Figure 6.6: No Photos Found

photo p10.

Results: Figure 6.7 shows the search results of the user u1 logged into the TIP system and search for tag *secret*. User u1 can see both photo p9 and p10 because s/he is the owner of photo p9 and p10. Figure 6.8 shows the search results of the user u2 logged into the TIP system and search for tag *secret*. S/he can only see photo p9 since s/he is a group member of group g1 and p9 has signed a access permission to g1. User u2 cannot see p10 because s/he does not have an authority to do so. Photo p10 is a *private* photo only the owner can see and the owner is u1.

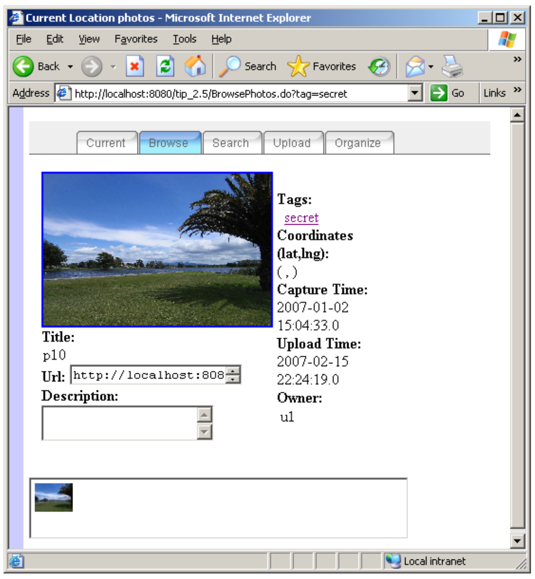

Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/tip_2.5/BrowsePhotos.do?tag=secret Go Links
Current Browse Search Upload Organize
Local intranet

Tags:
secret
Coordinates (lat,lng):
(,)
Capture Time:
2007-01-02 15:04:33.0
Upload Time:
2007-02-15 22:24:19.0
Owner:
u1

Title:
p10
Url: http://localhost:808
Description:

Figure 6.7: Private Photos: Two Photos Found

Test case 4: Evaluate if the system can display the current location photos.


Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
Go
Links
Current
Browse
Search
Upload
Organize
Local intranet

http://localhost:8080/tip_2.5/BrowsePhotos.do?tag=secret

Tags:
secret
Coordinates (lat,lng):
(,)
Capture Time:
2007-01-02 15:04:33.0
Upload Time:
2007-02-15 22:24:19.0
Owner:
u1

Title:
p10
Url: http://localhost:808
Description:

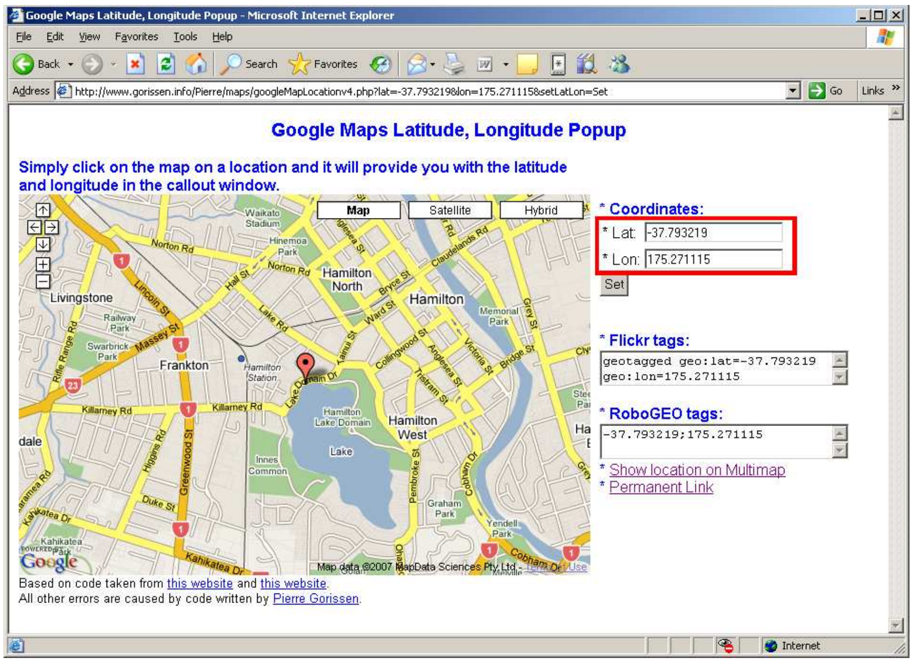
Figure 6.8: Restricted Photos: One Photo Found

Since we are using a desktop computer to do the evaluation, we pick a point from the map (shows in Figure 6.9) and use the page (shows in Figure 6.10) to simulate the current location. We picked a point around *Hamilton lake*, so that we can retrieve five photos (p1-p5) which are taken around *Hamilton lake* in section 6.1.1.

Hypothesis: The browser display five photos (p1-p5).

Results: Figure 6.11 shows that the browser displays five photos (p1-p5) thumbnails in the bottom of the page and displays the photo p1 in main display frame with full information around it. A user can click on any photo thumbnail to display the photo in main display frame and the photo information. As an example, we clicked on photo p4 thumbnail and p4 is displaying in main display frame (shows in Figure 6.12).

Test case 5: Evaluate if a user can browse photos by given a location. We start from *start browsing* page (shows in Figure 6.2), which provides a *tag cloud* in the middle of the page to allow users to choose a tag to start browsing the tag related photos. In the bottom of the *start browsing* page, it provides coordinates input fields to allow users start browsing photo from a particular location. In order to help user to find a location efficiently, it also provides a link called *or get coordinate from a map* which users can get coordinates by clicking on a map (shows in Figure 6.13). We get the Hamilton Gardens coordinates from the map and start browsing from *start*


Google Maps Latitude, Longitude Popup - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.gorissen.info/Pierre/maps/googleMapLocationv4.php?lat=-37.793219&lon=175.271115&setLatLon=Set
Go Links
Map Satellite Hybrid
Set
Internet

Google Maps Latitude, Longitude Popup

Simply click on the map on a location and it will provide you with the latitude and longitude in the callout window.

* Coordinates:
* Lat: -37.793219
* Lon: 175.271115

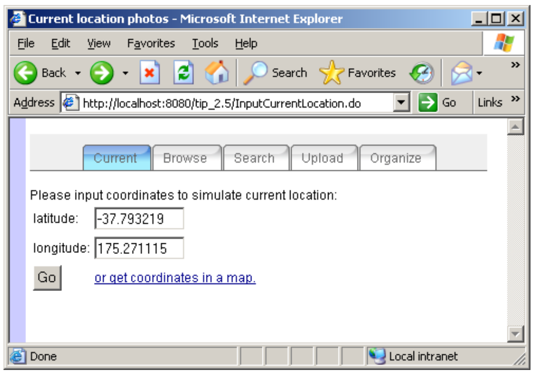
* Flickr tags:
geotagged geo:lat=-37.793219 geo:lon=175.271115

* RoboGEO tags:
-37.793219;175.271115

* Show location on Multimap
* Permanent Link

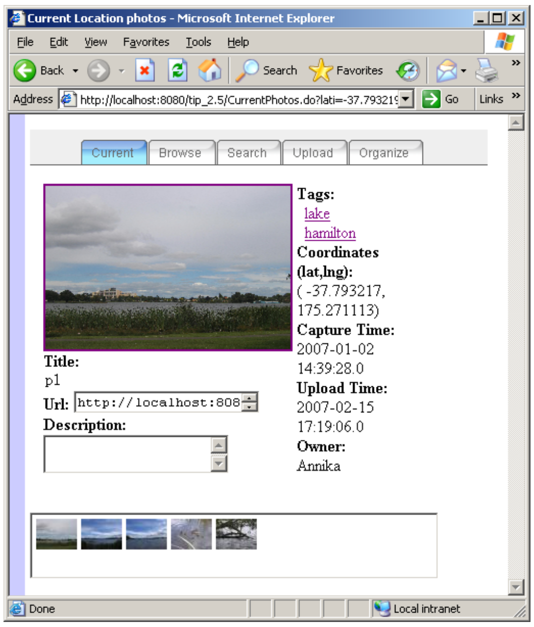
Based on code taken from this website and this website.
All other errors are caused by code written by Pierre Gorissen.

Figure 6.9: Get Hamilton Lake Coordinates from a Map


Current location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/tip_2.5/InputCurrentLocation.do
Go Links
Current
Browse
Search
Upload
Organize
Done
Local intranet

Please input coordinates to simulate current location:
latitude: -37.793219
longitude: 175.271115
Go
or get coordinates in a map.

Figure 6.10: Simulate Current Location

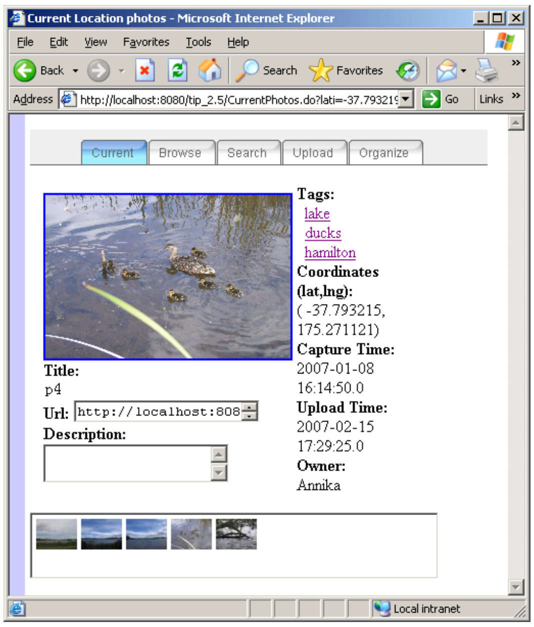

Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
Go
Links
Current Browse Search Upload Organize
Done
Local intranet

http://localhost:8080/tip_2.5/CurrentPhotos.do?lati=-37.79321

Tags:
lake
hamilton
Coordinates (lat,lng):
(-37.793217, 175.271113)
Capture Time:
2007-01-02 14:39:28.0
Upload Time:
2007-02-15 17:19:06.0
Owner:
Annika

Title:
p1
Url: http://localhost:808
Description:

Figure 6.11: Photo P1 Displays in Main Display Frame


Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
Go
Links
Current Browse Search Upload Organize
Local intranet

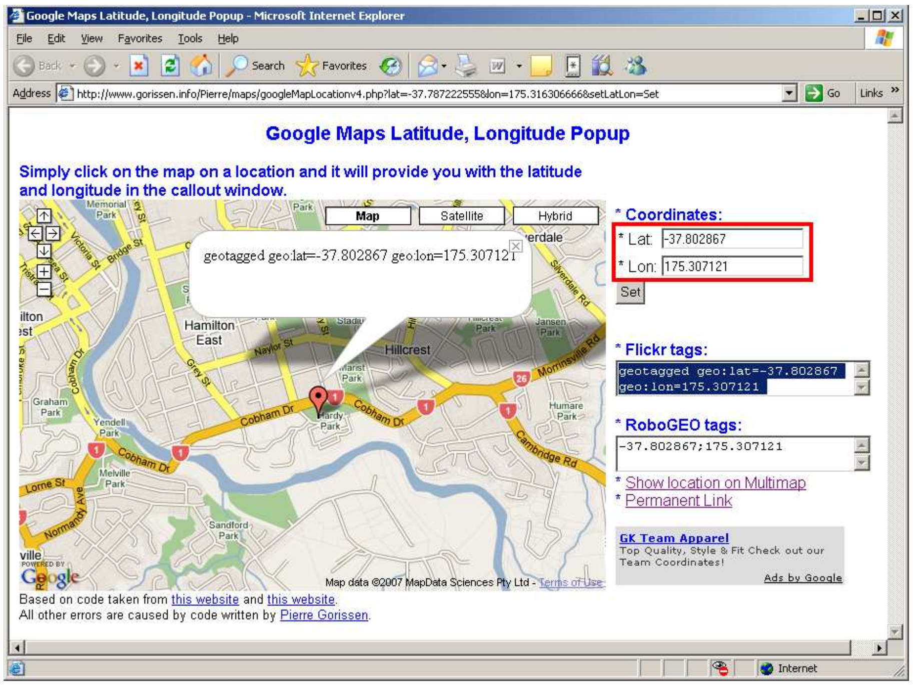
http://localhost:8080/tip_2.5/CurrentPhotos.do?lati=-37.79321

Tags:
lake
ducks
hamilton
Coordinates (lat,lng):
(-37.793215, 175.271121)
Capture Time:
2007-01-08 16:14:50.0
Upload Time:
2007-02-15 17:29:25.0
Owner:
Annika

Title:
p4
Url: http://localhost:808
Description:

Figure 6.12: Photo P4 Displays in Main Display Frame

browsing page by indicating the coordinates.


Google Maps Latitude, Longitude Popup - Microsoft Internet Explorer

Address http://www.gorissen.info/Pierre/maps/googleMapLocationv4.php?lat=-37.787222555&lon=175.316306666&setLatLon=Set

Google Maps Latitude, Longitude Popup

Simply click on the map on a location and it will provide you with the latitude and longitude in the callout window.

geotagged geo:lat=-37.802867 geo:lon=175.307121

* Coordinates:
* Lat: -37.802867
* Lon: 175.307121

* Flickr tags:
geotagged geo:lat=-37.802867 geo:lon=175.307121

* RoboGEO tags:
-37.802867;175.307121

* Show location on Multimap
* Permanent Link

Based on code taken from this website and this website.
All other errors are caused by code written by Pierre Gorissen.

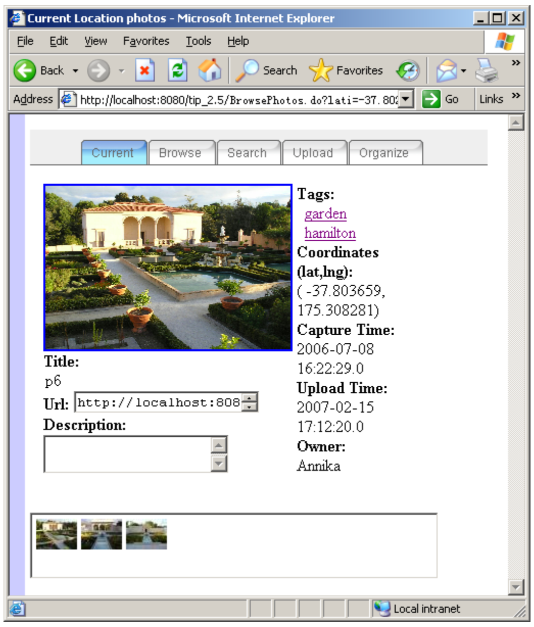
Figure 6.13: Hamilton Gardens in a Map

Hypothesis: The browser display three photos (p6, p7 and p8).

Results: Figure 6.14 shows that the browser displays three photos (p6, p7 and p8) in thumbnails and displays the photo p6 in main display frame with full information around it.

Test case 6: Evaluate if a user can search for photos by tags or a location. Our system provides users a search function to find photos. The search function (shows in Figure 6.15) allows users search photos by tags, capture time, locations (coordinates) and owner.The users may combine different conditions to retrieve more specific photos. We encourage users to add location name to their photos as tags. Meanwhile we allow users to search photos by their physical location (GPS coordinates). Since we have used and showed screen shots for searching tags in test case 2 and 3, we do not do that again here. We use coordinates to search photos. On the *search page* (shows in Figure 6.15), We open the *map page* by click on the link *or get coordinates in a map* find the coordinates of the *Hamilton Lake* from the map. Then search for photo around the coordinates.

Hypothesis: According to our test data (in section 6.1.1), the search

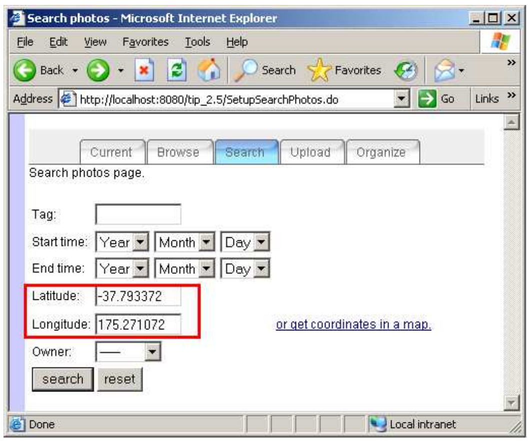

Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
Go
Links
Current Browse Search Upload Organize
Local intranet

http://localhost:8080/tip_2.5/BrowsePhotos.do?lati=-37.80

Tags:
garden
hamilton
Coordinates (lat,lng):
(-37.803659, 175.308281)
Capture Time:
2006-07-08 16:22:29.0
Upload Time:
2007-02-15 17:12:20.0
Owner:
Annika

Title:
p6
Url: http://localhost:808
Description:

Figure 6.14: Photos in Hamilton Gardens

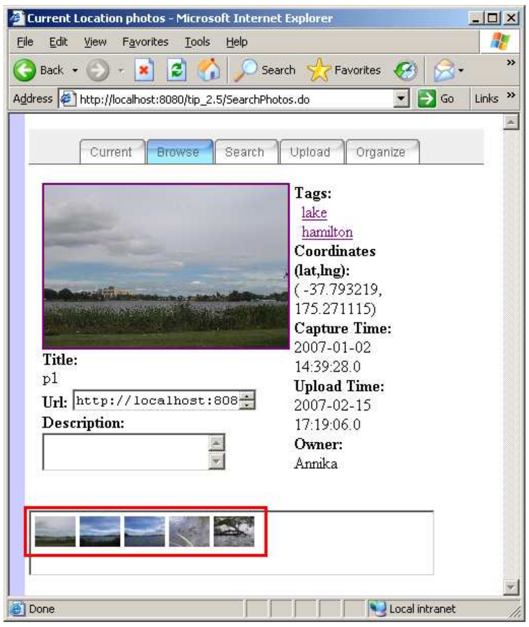

Search photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
Go
Links
Current Browse Search Upload Organize
search
reset
Done
Local intranet

http://localhost:8080/tip_2.5/SetupSearchPhotos.do

Search photos page.

Tag:
Start time: Year Month Day
End time: Year Month Day
Latitude: -37.793372
Longitude: 175.271072
or get coordinates in a map.
Owner: ——

Figure 6.15: Search Photo Page


Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/tip_2.5/SearchPhotos.do Go Links
Current Browse Search Upload Organize
Done
Local intranet

Tags:
lake
hamilton
Coordinates (lat,lng):
(-37.793219, 175.271115)
Capture Time:
2007-01-02 14:39:28.0
Upload Time:
2007-02-15 17:19:06.0
Owner:
Annika
Title:
p1
Url: http://localhost:808
Description:

Figure 6.16: Location Search Results

should retrieve five photos (p1-p5).

Results: Figure 6.16 shows the results of the search. Five photos are found as we hypothesized.

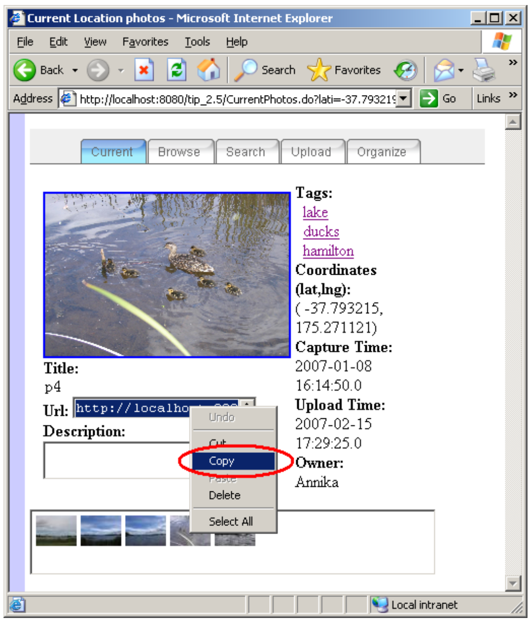
Test case 7: Evaluate if a user can get the URL of the photo easily in order to reuse it in other place. *Photo display page* provides an URL for each photo that displayed in the *main display frame*. It uses a textbox to display the URL of the current browsing photo. When a user click on the textbox, the URL will be selected and ready for copying to the clipboard. Then, the user can use the URL in other place.

Hypothesis: The URL can be easily copied to clipboard and use it in other place.

Results: Figure 6.17 shows we retrieve the URL of the photo p4 in the *photo display* page. The process is simple: 1. right click on the URL textbox and the content is selected automatically; 2. select *copy* item from the pop up list and the URL is copied into the clipboard. 3. copy to other place as users want.

Test case 8: Evaluate if the system allow a user to upload a photo with an URL.

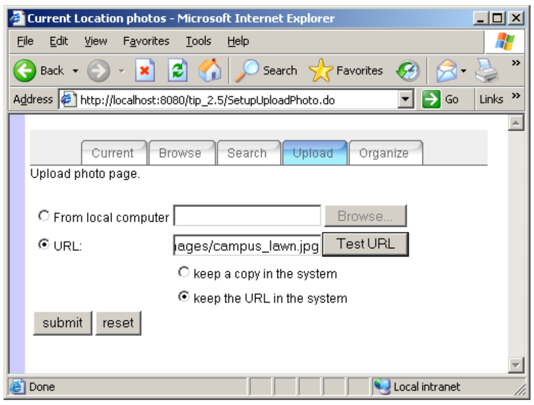
Hypothesis: A user successfully used the URL of a photo which from


Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/tip_2.5/CurrentPhotos.do?lati=-37.79321
Go Links
Current Browse Search Upload Organize
Undo
Cut
Copy
Paste
Delete
Select All
Local intranet

Tags:
lake
ducks
hamilton
Coordinates (lat,lng):
(-37.793215, 175.271121)
Capture Time:
2007-01-08 16:14:50.0
Upload Time:
2007-02-15 17:29:25.0
Owner:
Annika
Title:
p4
Url: http://localho
Description:

Figure 6.17: Retrieve the URL of a Photo

web instead upload photo from local computer.


Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/tip_2.5/SetupUploadPhoto.do
Go Links
Current Browse Search Upload Organize
Browse...
Test URL
submit
reset
Done
Local intranet

Upload photo page.

From local computer
[✓ URL:] ages/campus_lawn.jpg

keep a copy in the system
[✓ keep the URL in the system]

Figure 6.18: Upload a Photo with an URL

Results: We open the *upload photo* page (shows in Figure 6.18). We select radio button *URL* and copy the photo URL into the textbox next to it. Then we click on the button *Test URL* to pop up a window and display the photo with the URL (shows in Figure 6.19). In *upload photo* page, We click on button *submit*. The photo is display in the page (shows in Figure 6.20) successfully, which means the system received the URL. Then

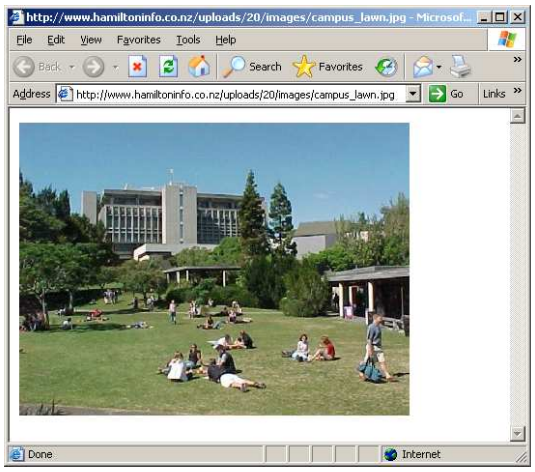

http://www.hamiltoninfo.co.nz/uploads/20/images/campus_lawn.jpg - Microsof...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.hamiltoninfo.co.nz/uploads/20/images/campus_lawn.jpg Go Links
Done Internet

Figure 6.19: Display the photo of the URL

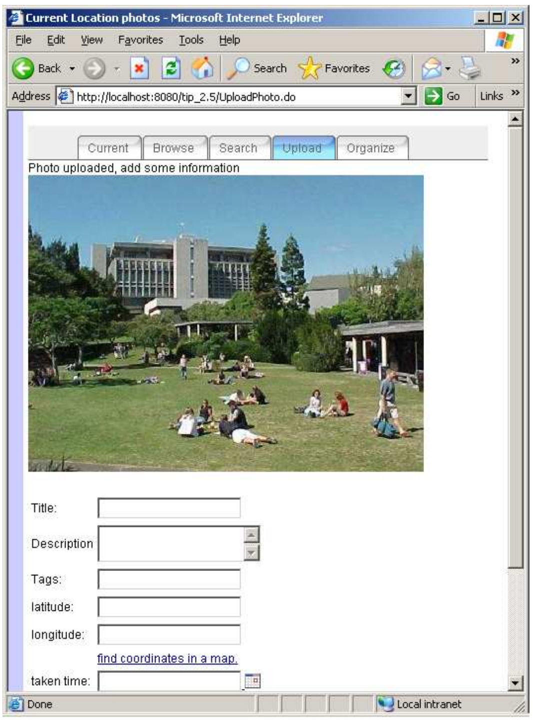

Current Location photos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/tip_2.5/UploadPhoto.do Go Links
Current
Browse
Search
Upload
Organize
Done Local intranet

Photo uploaded, add some information

Title:

Description

Tags:

latitude:

longitude:

find coordinates in a map.

taken time:

Figure 6.20: Input photo information as local photo uploading

we input photo information and signed access permission to the photo which is same as upload photo from local computer.

6.2 Quantitative Evaluation

In this section, we carry out the quantitative evaluation of our implementation. We measure response time of retrieving photos when different parameters are used. First, we introduce the experiment setting we use for the evaluation. Second, we explain experimental setting. Then, we describe hypotheses and experiment results for each experiment. Last, we analyze and explain the experiment results.

6.2.1 Experiments and Test Setting

We consider that users retrieve photos mainly by tag, capture time and location. We defined three experiments to evaluation our implementation performance. We employ the number of users of 10,000 and each of users owns the number of photos of 1,000. In total, the number of photos is up to 10,000,0000 in the TIP database. We use an average response time of 100 times tests for each test. In the following, we describe experiments and test setting individually.

Experiment 1: *Varying numbers of photos have a tag that is the same to the given tag.*

In this experiment, we focus on the number of matched photos which have the given tag as a tag. Based on the number of the photos in the TIP database, we defined the number of matched photos starts from 0 to 100,000 photos and increasing with 10,000 photos. We measure the average response time, which starts from a user sending the request and ends with the TIP system sending back the photos as a result of the user request. The average response time depend on the number of matched photos.

Experiment 2: *Varying numbers of photos near-by the given location*

In this experiment, we focus the number of matched photos which close by the given location. As a default, the close by locations are defined in TIP system as: the close by location's latitude (*cl_lat*), the close by location's longitude (*cl_lng*), the given location's latitude (*gl_lat*) and the given location's longitude (*gl_lng*) have relationships of $(gl\_lat - 0.001) \leq cl\_lat \leq (gl\_lat + 0.001)$ and $(gl\_lng - 0.001) \leq cl\_lng \leq (gl\_lng + 0.001)$. The unit is in degree.

Based on the number of the photos in the TIP database, we defined the number of matched photos range in 0 to 100,000 photos in an increment of 10,000 photos. We measure the average response time which start from a user send the request and end with the TIP system send back the photos as a result of the user request.

Experiment 3: *Varying numbers of photos taken in the given time range*

In the experiment, we employ the number of the matched photos which the photo *capture time* is in the given time range. Based on the number of the photos in the TIP database, we defined the number of matched photos starts from 0 to 100,000 photos and increasing with 10,000 photos. We measure the average response time which count starting from a user send the request and ending with the TIP system sent back the photos as a result of the user request.

6.2.2 Hypotheses and Results Analyze

Experiment 1: *Varying numbers of photos have a tag that is the same as the given tag.*

Hypothesis: The average response time increases linearly while the number of photos that have a tag same as the given tag increases.

Results and analyze: Figure 6.21 shows the experiment results. From the figure we can see that increment of response time vary with the number of the matched photos linearly. We can also found that the average response time is about doubled while the number of the matched photos is doubled.

We have used b-tree index on tag id and photo id on table Photo_to_tag and b-tree on photo id on table photo, the response time mainly depends on the retrieve time, which means the more photos we found, the more retrieve time we need.

Experiment 2: *Varying numbers of the photos near-by the given location*

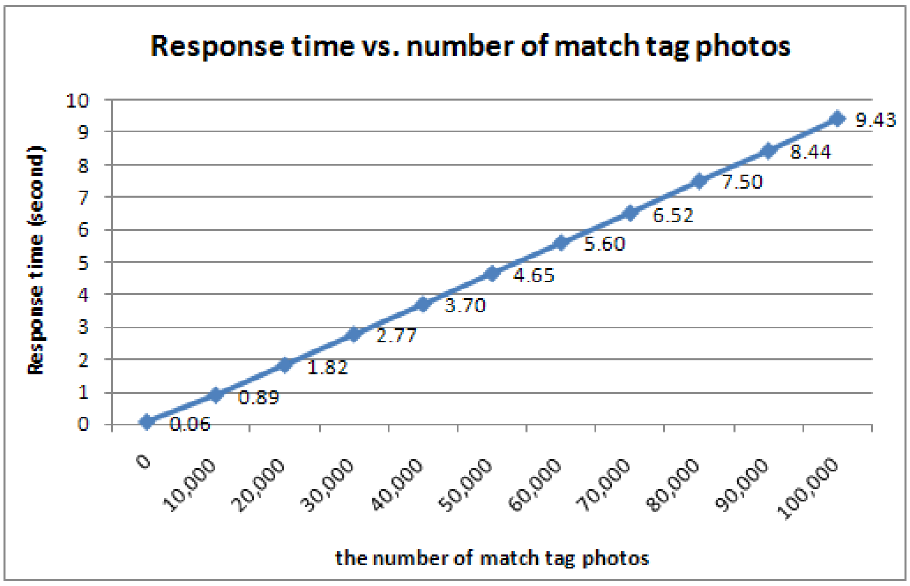


Figure 6.21: Response Time vs. Number of Matched Photos by tag (10,000,000 Photos in Total)

Hypothesis: The average response time increases linearly while the number of near-by photos of the given location increases.

Results and analyze: The experiment result shows in Figure 6.22. The figure shows that response time increases linearly while the number of near-by photos of the given location increases. We can see from the Figure 6.22, while the no photos are near-by the given location, the response time is 0.06 second which means the index is working efficiently.

Since B-tree is not suitable for indexing spacial information but GiST does, we used GiST index on attribute *location* on table *photo* to index coordinates. As the result, the response time increases linearly while the number of near-by photos of the given location increases.

Experiment 3: *Varying numbers of the photos taken in the given time range*

Hypothesis: The average response time increases linearly while the number of photos in the given time range increases.

Results and analyze: The experiment result shows in Figure 6.23. The figure shows that increment of response time vary

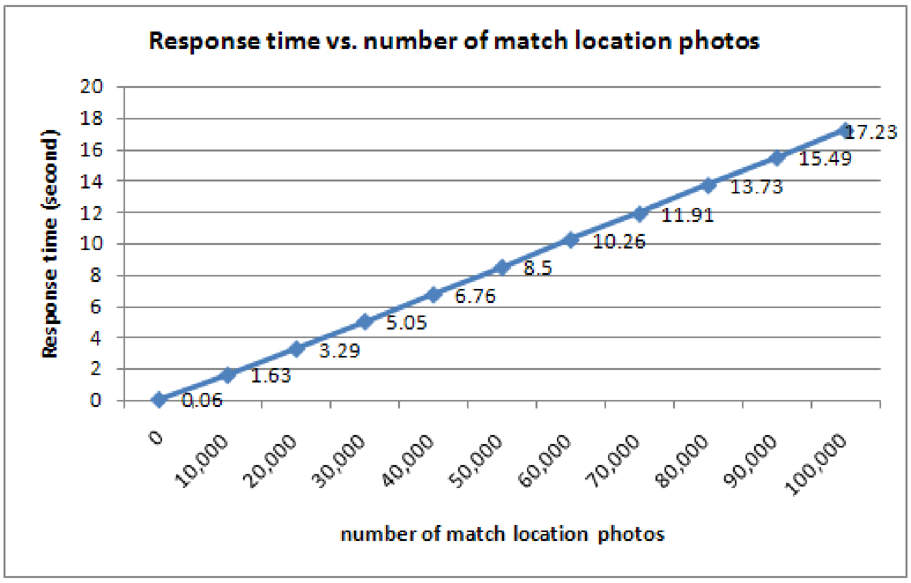


Figure 6.22: Response Time vs. Number of Near-by Photos of Given Location (10,000,000 Photos in Total)

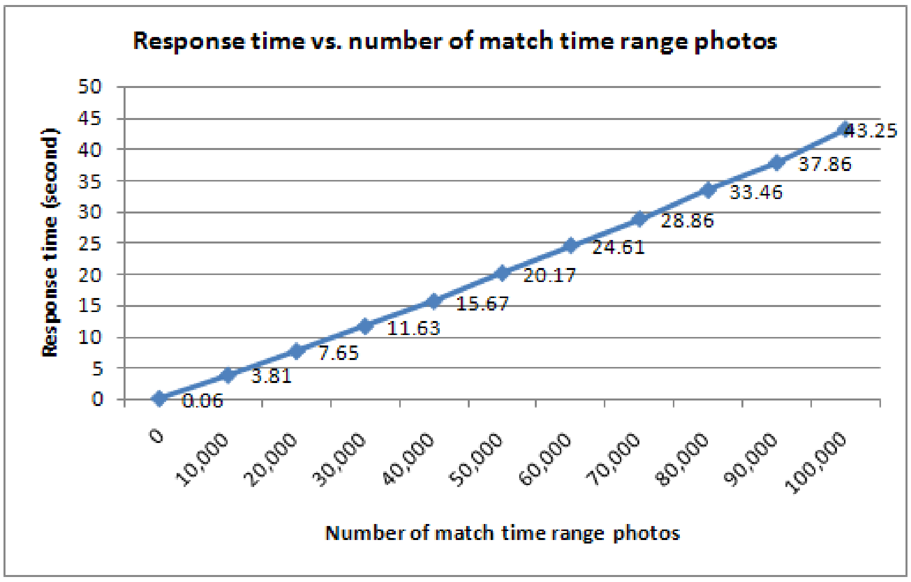


Figure 6.23: Response Time vs. Number of Photos in the Given Time Range (10,000,000 Photos in Total)

linearly with the number of the photos are nearby the given location.

We use a B-tree index on attribute *capture_time* on table *photo*. As shown in Figure 6.23, when no photos are taken in the given time range the response time is only 0.06 second which means the index works effectively. As the result, the response time depends on the number of photos in the given time range.

6.3 Conclusion

In this chapter, we evaluated our implementation for both qualitative and quantitative criteria.

For the qualitative evaluation, we carried out eight function tests covering the six user requirements we discussed in Section 2.2. The all eight function tests performed successfully. We used screen shots to indicate the processes and the results of the tests. The tests confirmed approved that our project has successfully achieved all six user requirements.

For quantitative evaluation, we performed three experiments to evaluate the response time when a user uses tags, locations and capture time to request related photos. All tests results shows the response time increases linearly with the increment of the numbers of matched photos. We noticed that the response time is quite long when the number of the matched photos is large, which shows a limitation of our implementation. The current implementation is not suitable for large amount of photos matched users request. We will discuss this limitation in Section 7.2.

Chapter 7

Conclusion

In this chapter, we first summarize what we accomplished in this project. Then we briefly outline possible future work.

7.1 Summary

As a part of the Tourist Information Provider (TIP) system, this project focuses on creating a photo gallery service in the TIP system, which allows users to share, browse and categorize their photos.

We introduced six scenarios to represent the travelers' behaviors when they are sharing, categorizing and browsing they photos. We have identified six user requirements from the scenarios:

UR1 Categorize photos.

UR2 Privacy control.

UR3 Viewing photos by current location.

UR4 Browsing photos by given locations.

UR5 Searching for location or event related photos.

UR6 Reusing photos between different web resources.

In order to find if and how other related photo systems achieved our user requirements, we have reviewed three related mobile photo sharing systems and two online photo sharing systems. We found that none of those systems achieved all six user requirements. There is no current system fully achieved our UR6 which is photo reusing. For UR1, we also compared predefined classifications and folksonomies. We decided to use tags and photo metadata Exif to classify photos. The reason is that tags is better than predefined classification for classifying photos: tags are easy to create and do not need to be revised as predefined classifications do.

To fulfil the six user requirements in our implementation, we first gave general design consideration for each of the user requirements. Then we drew eight use cases to represent users' interactions with the system. UML activity diagrams are also given to describe the eight user cases. Based on the eight use cases, we described the database design with an ER diagram and relational schema. We illustrated paper-based prototype to describe our user interface design.

We described the architecture of the TIP component model and the photo gallery component (see Figure 5.3). The photo gallery component involves six elements

1. Photo gallery manager
2. Data input/update
3. search engine
4. location engine
5. privacy controller
6. photo repository

We gave details of the table definitions to describe the implemented database. We also described the software implementation by giving an overview of the photo gallery using a component class diagram (see Figure 5.4). To explain the implementation in detail, we chose *search photos* interactions as a representative and used it to explain the implementation.

We evaluated our implementation in both qualitative and quantitative. For qualitative evaluation, we carried out eight function tests to cover the six user requirements which we discussed in Section 2.2. The eight function tests performed successfully. We used screen shots to indicate the processes and the results of the tests. It also confirmed that our project has successfully achieved all six user requirements. Table 7.1 shows the result of a comparison of this project and other photo sharing systems, which we reviewed in Chapter 3, against our six user requirements.

	UR1	UR2	UR3	UR4	UR5	UR6
The MMM	+	-	-	+	+	-
The MobShare	-	(+)	-	-	-	-
The Navigation	-	-	+	+	-	-
Flickr	+	+	-	-	+	(+)
MSN Spaces	(+)	(+)	-	-	-	-
Photo Gallery in TIP	+	+	+	+	+	+

Table 7.1: Comparison of Photo Gallery and other photo sharing systems against URs (Symbols: + achieved, (+) partially achieved, - not achieved)

We performed three experiments as quantitative evaluation to evaluate the response time when a user uses tags, locations and capture time to retrieve related photos. All tests results show that the response time increases linearly with the increment of the numbers of matched photos. The response time is high when the number of the matched photos is large. We identify this as a limitation for this system. We will discuss this more in the following section.

7.2 Limitations

As we discovered from the quantitative evaluation experiments, the response time is very long when the number of matched photos is large. For example, in experiment 3, when the number matched photos is 100,000, the average response time is up to 43.25 seconds. To reduce the response time of retrieving photos, especially when retrieving a large amount of photos, we may use paging technology. We separate the matched photos into pages and each page contains a certain number of photos. The system only retrieves one page's photos on each user's request. One page may contain 50 photos or even less. According to the three experiments results, we assume that the response time will be dropped to less than 0.5 second.

Another limitation is that the inherent limitation of tags. As we mentioned in Section 4.1.1, using tags has the three disadvantages of the possibility of misspelling, synonyms and multivocal tags used. We have not considered a solution for synonyms and multivocal tags in this project. They remain in our system as an unsolved limitation.

7.3 Future Work

There are serval things we would to do in the future:

1. First, we would like to solve the efficiency problem as we mentioned in *Limitations* Section 7.2 in our next version of the photo gallery implementation. The paging technology will be used to help reduce photo retrieve time and performance statable retrieve time which does not related to the amount of the photos matched.

2. Second, a semantic search on tags would be helpful to improve or solve the tag limitation (synonyms and multivocal tags) to make the tag becomes a more powerful and reliable classification technique.

3. HCI evaluation techniques will be helpful to identity any user interaction problems in the system. For this, test data and participants are needed in order to get real users feedback about the user interaction. This would include longer test as that would require travels and feedback. These further tests are planed for the user participation and will involve all TIP system UI components.

Bibliography

[1] S. Ahern, S. King, and M. Davis. Mmm2: mobile media metadata for photo sharing. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 790–791, New York, NY, USA, 2005. ACM Press.

[2] J. Chen and A. Hinze. Importing external data sources into TIP. PGDip Project report, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 2006.

[3] M. Davis, S. King, N. Good, and R. Sarvas. From context to content: leveraging context to infer media metadata. In *MULTIMEDIA'04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 188–195, New York, NY, USA, 2004. ACM Press.

[4] Del.icio.us: a social bookmarking website. available at `http://del.icio.us` Accessed on 22/01/2007 4:05 p.m.

[5] Exchangeable image file format for digital still cameras: Exif version 2.2. available at `http://www.exif.org/Exif2-2.PDF` Accessed on 02/01/2006 11:00 a.m.

[6] Flickr: The best way to store, search, sort and share your photos. available at `http://www.flickr.com` Accessed on 02/04/2006 10:30 a.m.

[7] A. Hinze and G. Buchanan. Cooperating services in a mobile tourist information system. In *Proceedings of the 13th International Conference on Cooperative Information Systems (CoopIS 2005)*, Agia Napa, Cyprus, 31 October-4 November 2005.

[8] A. Hinze, X. Gao, and D. Bainbridge. The TIP/Greenstone bridge: A service for mobile location-based access to digital libraries. In *European conference on research and advanced technology for digital libraries (ECDL 2006)*, pages 99–110, Alicante, Spain, 2006.

[9] A. Hinze and A. Voisard. Location- and timebased information delivery in tourism. In *Advances in Spatial and Temporal Databases (SSTD 2003) Vol. 2750 of LNCS*, Satorini Island, Greece, 2003.

[10] X. Huang and A. Hinze. Travel planning component in the tourist information provider system (TIP). Master's thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 2007, Work in progress.

[11] S. Junmanee and A. Hinze. Advanced recommendations in a mobile tourist information system. Master's thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 2006.

[12] W. Lin and A. Hinze. Community-based interaction for TIP. PGDip Project report, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 2006.

[13] A. Mathes. Folksonomies - cooperative classification and communication through shared metadata. December 2004.

[14] Msn space: Blogging and online photos sharing services. available at `http://spaces.msn.com` Accessed on 03/04/2006 11:25 a.m.

[15] J. Pauty, P. Couderc, and M. Banatre. Using context to navigate through a photo collection. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 145–152, New York, NY, USA, 2005. ACM Press.

[16] Q. Qiu and A. Hinze. Trust-based recommendations in TIP. Master's thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 2006.

[17] P. Riddle. Tags: what are they good for? available at `http://prentissriddle.com/school/papers/385q/riddle-2005-tags.pdf` Accessed on 18/4/2006 2:38 p.m.

[18] R. Sarvas, M. Viikari, J. Pesonen, and H. Nevanlinna. Mobshare: controlled and immediate sharing of mobile images. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 724–731, New York, NY, USA, 2004. ACM Press.

[19] Y. Wang and A. Hinze. Travel itinerary in TIP. PGDip Project report, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 2006.

Appendix A

Database Structure of TIP

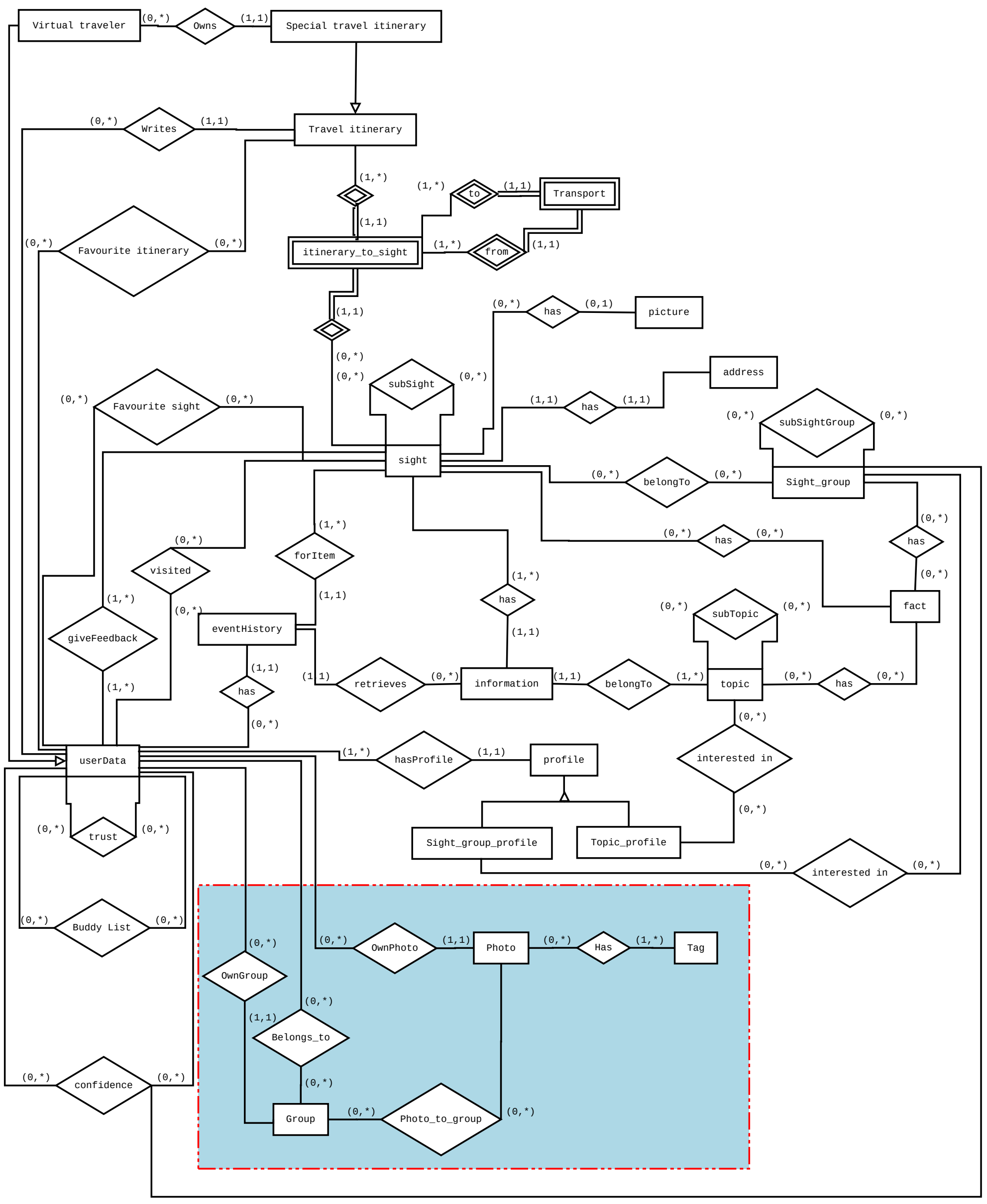


Figure A.1: Entity relationship diagram of the TIP database (Dash line frame shows the ER diagram of this project)