Stony Brook University

Design, Implementation and Analysis of a Windows Mobile Phone Application

A Thesis Presented

by

Prashanth Krishnan Ranganathan

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Electrical Engineering

Stony Brook University

May 2012

Stony Brook University

The Graduate School

Prashanth Krishnan Ranganathan

We, the thesis committee for the above candidate for the

Master of Science degree, hereby recommend

acceptance of this thesis.

Xin Wang – Thesis Advisor
Associate Professor, Electrical and Computer Engineering

Alex Doboli – Second Reader
Associate Professor, Electrical and Computer Engineering

This thesis is accepted by the Graduate School

Charles Taber
Interim Dean of the Graduate School

Abstract of the Thesis

Design, Implementation and Analysis of a Windows Mobile Phone Application

by

Prashanth Krishnan Ranganathan

Master of Science

in

Electrical Engineering

Stony Brook University

2012

The focus of this research is to design and implement a pioneering photo and time organization application for the Windows Phone 7, that simplifies organization and sharing while expanding the ways of searching and identifying photos, and consequently analyze and explore the future possibilities for the application based on the hardware and software capabilities of the phone and make suitable recommendations to improve the application and the phone. Furthermore, this research is intended to provide valuable input for developers of the Windows Phone 7 by locating and documenting the issues faced. Lastly, we intend the application to serve as a model for future high-end applications that follow the basic theme of photo and time organization on a grander scale.

Table of Contents

List of Figures vii

Chapter 1: Funto Photoshare 1

1.1 Introduction 1

1.1.1 Smartphones and Digital Cameras 1

1.2 Conception 2

1.2.1 Lack of a Photo Management Tool 2

1.2.2 Photosynth 2

1.2.3 Google Landmark Recognition 3

1.3 Functions 3

1.4 Tagging 8

1.5 Comparison Page 10

1.6 Classes and Controls used 11

1.6.1 Canvas Control 11

1.6.2 Stream, WriteableBitmap, decoding JPEG 12

1.7 Search Types 13

1.7.1 Name Search 14

1.7.2 Date Search 15

1.7.3 Location Search 16

1.7.4 Composite Search 17

1.8 Settings and other pages 18
1.8.1 Settings 18
1.8.2 Help 19
1.8.3 Ratings 19
1.8.4 Comments 19
1.8.5 Delete 22
1.8.6 Socket Support 22
1.9 App Bar Buttons 22
Chapter 2: Daily Log App 23
2.1 Lack of a Journal or Diary App 23
2.2 Time Zone Structure 24
2.3 Logging Activities 25
2.4 Search Features 26
Chapter 3: Performance Analysis 27
3.1 Monitoring Device Status 27
3.2 CPU Usage Time 28
3.3 Known Issues 32
Chapter 4: Conclusion 33
4.1 Current Work 33
4.2 Difficulties Faced 33

4.3 Potential Pitfalls 35

4.4 Scope for Improvement 35

References 37

List of Figures

Figure 1 Main Page of Funto.....5
Figure 2 Process Flow Diagram.....7
Figure 3 Comparison Game.....11
Figure 4 Date/Time Search.....15
Figure 5 Location Search.....16
Figure 6 Comments.....20
Figure 7 Ratings.....21
Figure 8 Time Zone Classification (Default).....24
Figure 9 GPS Service Call Time.....29
Figure 10 Cloud Service Call and Photo Display Time.....30
Figure 11 Memory Usage.....31

Acknowledgments

I would like to thank my Thesis Advisor Prof Xin Wang for leading me through some tricky times and for believing in me when I seemed lost. I wouldn't have come this far without her help and am eternally indebted to her. I would also like to thank my parents for being pillars of support and my mother for encouraging me to do a thesis. I must mention my boss, Ms. Marie McCallion, here for providing me with the spark for creating one of the features of this application. This wouldn't be complete if I didn't mention my friends who were understanding during my tough times and gave me proper advice. I would also like to thank Prof. Alex Doboli for kindly agreeing to be one of the readers of this thesis at such short notice.

Chapter 1: Funto Photoshare

1.1 Introduction

1.1.1 Smartphones and Digital Cameras

With the advent of smartphones people have integrated their lives to one device. They communicate using the phone, connect with friends using the internet, listen to music while on the move, check stock market listings and with technologies like Square have replaced the credit card reader.

One of the important changes brought about smartphones is to make digital cameras a luxury. Smartphones can fit into our pockets and take photos with resolution on par with several digital cameras which can be instantly uploaded to the internet for our friends to see and hence digital cameras have become redundant for most users. Only dedicated photographers invest in cameras nowadays. (Potential of Cellphone Photography, 2007)

A 2011 survey by the Pew Research Center on Cell Phone Use in America concluded that one third of all Americans use a smartphone and over 90% of these smartphone users take photos and 80% share them with friends, online or via messaging. (Americans and Their Cell Phones, 2011)

Thus it can be asserted that if a person were given a smartphone today he/she would end up having a few thousand photos within a decade. That contains a lot of information about his/her life but without vital statistics of the photo which are not saved in the photographs, it would be hard to arrange them chronologically or based on any other classification.

1.2 Conception

1.2.1 Lack of a Photo Management Tool

When the Windows Phone 7 first came out, it did not have several features that are the norm of smartphones all over the world right now- multiple applications running in parallel, background task completion etc.

Another important feature missing was a photo locator tool- that is a mechanism to collect and store the GPS coordinates of the phone at the instant of taking each photo. While the hardware for getting GPS coordinates was present and this information was collected automatically in the default photos App of the phone, these coordinates were not made available to the phone's users or to application developers due to security concerns.

This was sidestepped when the GPS hardware was made available to the developers and since the photo locator is a default feature in the competing cellphone brands like iPhone, it served as a motivation to create a similar App with added functionality.

1.2.2 Photosynth

One of the first areas of research interest to be analyzed by us when a photo application had to be created for the Windows Phone 7 was the Photosynth which is Microsoft's own research on generating three-dimensional models from digital photos by analyzing them and creating a point cloud of a photographed object.

Pattern recognition components compare portions of images to create points, which are then compared to convert the image into a model. Users are able to view and generate their own models using a software tool available for download at the Photosynth website (Photosynth) (What is Photosynth?)

However, while a Photosynth App was released for the iPhone in April 2011, Microsoft acknowledged that such an App for the Windows Phone 7 was not feasible due to the then current hardware capabilities of that phone which did not possess “the level of camera” and did not allow “low-level algorithmic hacking” needed to make the Photosynth work. (Photosynth App, 2011)

1.2.3 Google Landmark Recognition

The Google Landmark Recognition project was aimed at “modeling and recognizing landmarks at a world-scale”.

This prompted us to work on creating an App for recognizing day-to-day objects from a library built by analyzing images of certain objects by connecting the phone to an application housed in the cloud was an ambitious task. However, this had to be dropped since duplex communication to the cloud was faulty and was corrected in later updates only, after complaints from many programmers, including us.

With services like “Software as a Service” etc. in the cloud themselves still in their development stage, this project by us for the Windows Phone 7 is still in the pipeline.

1.3 Functions

The posts on social networks are undoubtedly snapshots of the user's mind over the course of his life.

Users of Facebook and similar social networks have bemoaned the lack of a search feature that would seamlessly provide them an insight into their mind process from various periods of time.

Since photos and text are the most uploaded content, any application that allows social connectivity must have the capability to combine searches of both these types of content.

While searching text is straightforward, providing search in photos presents an interesting problem since computer recognition of physical objects in photos is still a nascent science and inaccurate.

Also, if searching in photos must be enabled, extra information must be saved for every photo than the photo itself.

Searching with text requires timestamps and the context of posting/speech., as in if it's a comment then to which photo is it related to, etc. Also, the comment must be connected with the profile that posted it so this info must also be stored with every comment that is posted.

Furthermore, another issue that raises here is whether the comment must be stored only alongside the photo or in the other profile of the user as well. That is, whether the comment by itself has significance or whether it has context and significance only if placed with the photo.

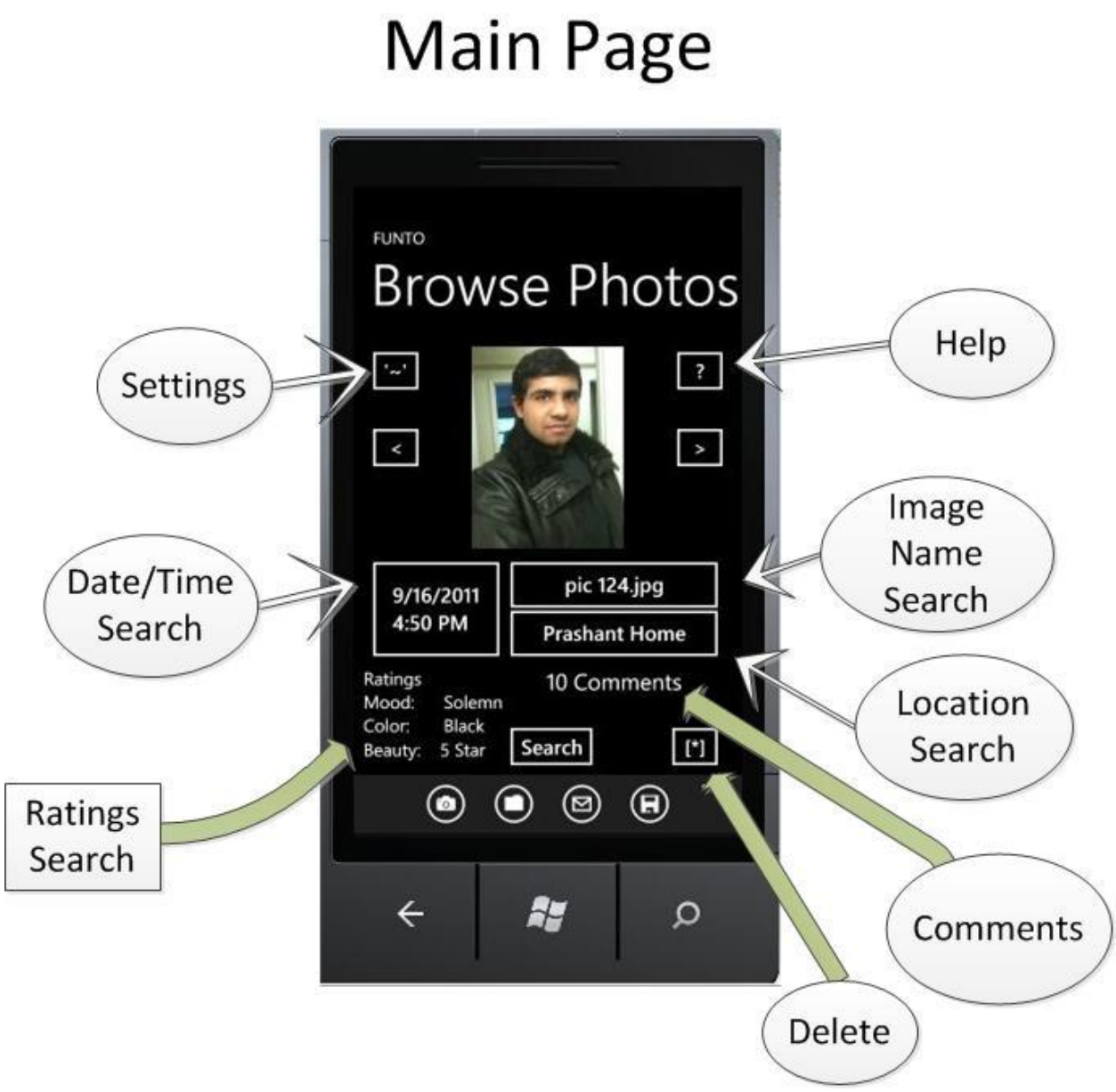

Figure 1 Main Page of Funto

Funto's basic functions are aimed at improving organization and sharing of the photos.

For this purpose, photos can be organized based on name, date/time, location or other factors like rating. Photos are not required by the user to be stored in albums.

For sharing it with friends, the user has to sign in to their Azure account. Once signed in, a one-click post option is available. Currently photos are available for

sharing only through Azure. In the future this will be expanded to the sharing photos via popular social networks like Facebook, Twitter, Instagram etc.

Friends can also comment on photos and rank photos as well based on categories. Users are also given the option of making only certain photos (taken using the application) public while all photos are available to viewed by them.

The composite search allows users to look for photos based on a specific name between two dates and in a specific location. It gives the users enormous flexibility.

The application does not interfere with the normal photo capturing process. Users merely have to take the photo and the application does all the work in the background. Any information that the user has to enter (like marking locations or rating photos or commenting) can be done any time afterwards without any loss of information with regards to the photo or the moment.

The application stores the date/time and the location of the capture as soon as the user captures and decides to save a picture. Unmarked locations can be tagged by the user at a later time. By looking at the photo and at the location on the map, users will be able to easily identify and tag.

A location once marked will automatically be updated in all other photos in proximate locations. If a friend has already tagged the location, the application will suggest this tag and once accepted it will be automatically used thereafter.

One of the other advantages to the application is that it eliminates the need to arrange photos into albums. Just like how the advent of Gmail transformed organization of web hosted email, by introducing labels, flags, stars and extensive search, we would like to look at this application as a start that would revolutionize the way photos are stored and used in everyday life.

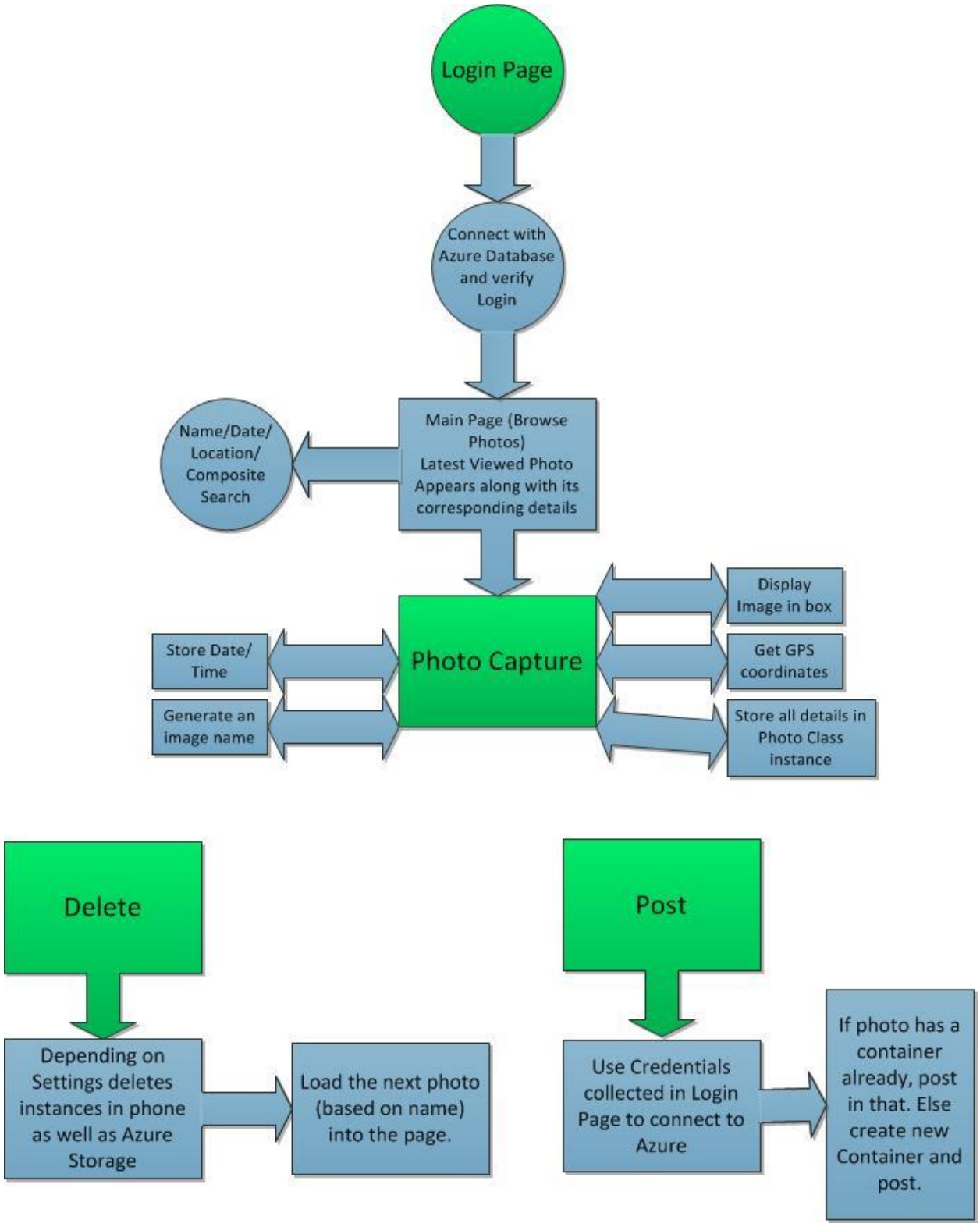

Figure 2 Process Flow Diagram

1.4 Tagging

We use the system to store the location and the time. The user has to provide information the context for the location and the occasion for the time. That is, the location coordinates obtained by the phone's GPS hardware are just a point on the map. However, most points on the map are not noteworthy and only the user will be able to provide meaning by identifying them as places of importance in his/her life. This is done by tagging locations.

Without this, photos cannot be related to events, which is the point of the application- to provide snapshots of the user's life.

The advantage of using this application compared to any similar Apps is that this allows the user to come back to the photos at a later date and tag them. The location and time information, combined with the actual photos themselves, would help the users identify the events successfully and relate to them.

Furthermore, their friends can also tag the photos of the location, occasion, ratings etc.
Tagging a location is merely the process of entering a name for the coordinates. The user does not see the coordinate values but rather sees a point on the map. Once it is named, all photos taken at the approximate location are tagged under the same location name and this is added to the list of locations.

Location coordinates are unique. No two items of the same coordinates are present on the list. Location names are not unique though. If the same location has been given different tags by different users, then both names will pertain to the same location.

If a contingency arises when two or more different locations have the same name, then depending on the search input, both items will be shown or one or more will be eliminated using more input and a narrower search.

However, such a contingency will arise only if the "Automatically Copy Friends' Tags" option is selected and if a friend has tagged locations incorrectly. For example, the friend or the user himself may tag photos taken at the geographical "*Brooklyn Bridge*" as "*Manhattan Bridge*". This error is probable since both bridges run almost parallel to each other.

Since the photos were taken at "*Brooklyn Bridge*", they will not pertain to the exact location coordinates of the "*Manhattan Bridge*". However, since the system does not know one location from the other this problem will not be detected automatically. Unless the user notices this discrepancy and corrects it, it would remain the same.

Since location search takes as input the name tag of the location rather than the coordinates, which cannot be provided by the user, this name tag error will continue to exist.

As of now, we have not arrived at a solution for this. We have rules in place as precautions but cannot completely prevent its occurrence.

These include precedence rules regarding tags whereby:

a. User-created tags are first precedence. So if same location coordinates have two different names, then both names will be used as tags for that location but the user created tag will be searched first. This method will only be the case if the "Automatically Copy Friends' Tags" option is selected.
b. If two different location coordinates have the same name after copying friends' tags, then the conflict will be resolved by the precedence rule stated above.
c. The application will not allow the user to accidentally give the same name tag to two different coordinates since, as previously stated, these are cardinal

rules of the application and hence will prompt the user to give a different name before proceeding.

1.5 Comparison Page

This is a jovial feature of the application wherein the user can pick a category and choose two images to compare for comic relief or even to document their weight loss or, as in this case, difference in makeup.

In this case, the category picked is the same person. Likewise, photos can be picked on the other categories, viz., location, mood and other ratings.

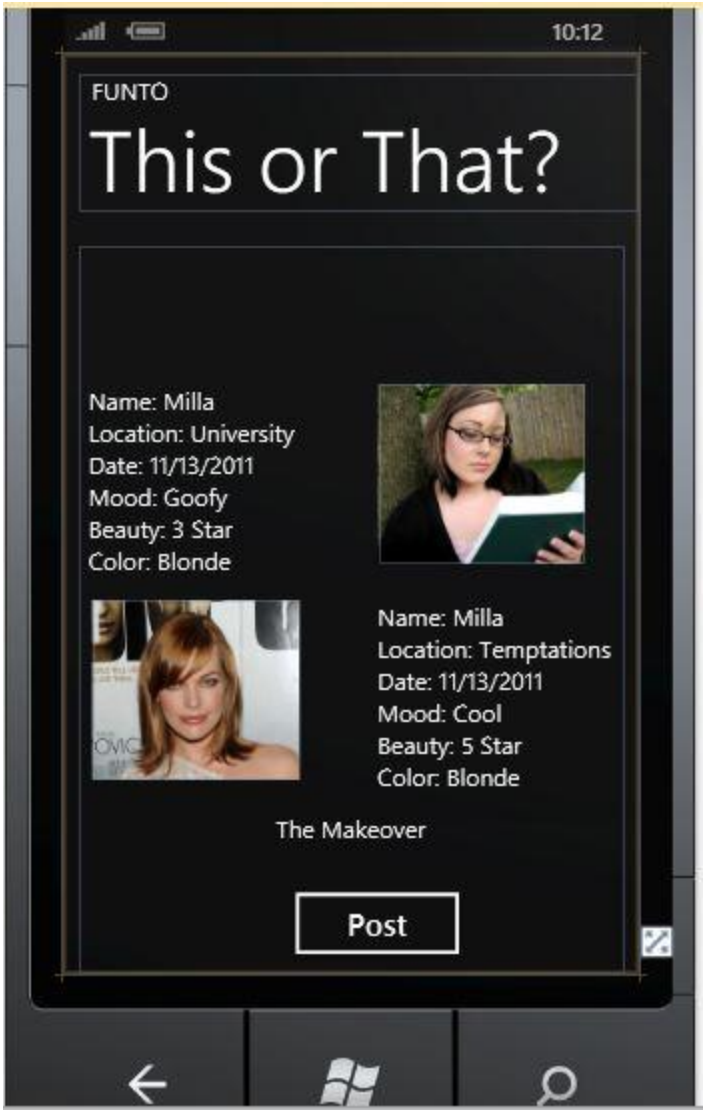

Figure 3 Comparison Game

1.6 Classes and Controls used

1.6.1 Canvas Control

For pop-ups we use the Canvas object since this allows us to restrict input from any other portion of the screen. So the user is forced to give an answer/make a decision to the prompt in the pop-up before proceeding with the application.

The opacity of the Canvas is changed from 0 to 1 based on the requirement.

1.6.2 Stream, WriteableBitmap, decoding JPEG

In this the captured photo is stored as a Stream and this allows for better handling of it, especially allowing it to be passed between functions.

It is important to decode photos to be able to modify them in the application.
For this purpose, the Stream Object is created for the JPEG and then the JPEG is decoded in a WriteableBitmap object. (How to: Decode a JPEG for Windows Phone, 2012)

Once the photo is captured and accepted by the user, then it is saved by invoking a constructor of the class Picture and the location, date/time is automatically stored while a unique name (based on index and date) is generated for the Picture. Other attributes like ratings are all initialized to default values.

This is the complete identity of the photo. And wherever it is stored all this information is attached with the actual photo itself.

The *BitMap* image class is used to store the captured photo in the Picture (user created class) while the Stream class is used to create an entity that can be sent as arguments via functions.

The detailed process is as follows: (insert figure here)

- The *CameraCaptureTask* class is used to access the Camera hardware of the phone and bring up the camera feature. Control is transferred to this camera feature until the user takes the photo and clicks accept.
- The function *OnCameraCaptureCompleted* takes the captured photo as a Stream and saves it as a *BitMap* image in the phone and then transfers control back to the application.
- Then the application invokes the GPS service by using an instance of the *GeocoordinateWatcher* class and sets the accuracy level (high in our case).

- After that the *GeocoordinateWatcher* class object scans for GPS signals and waits for a period of time. If a signal is obtained within that time, it is a success else its sends a message that location data is unavailable which means the user will have to wait for the *GeocoordinateWatcher* to initialize again and note the location.

  The application will first check if GPS service is available. If so, it will keep trying to get the coordinates until successful. However, if the service is not available, then it will immediately return this message and stop checking.

  If the user is walking, then he can continue walking- the distance traveled by the user will not affect the readings (between the point where he took the photo and point where the watcher got the data in the 2nd attempt). However, if the user is in motion like in a car or bicycle then it would be a problem since the location of the photo will not be accurate.

  This can be remedied only by the user- he has to wait until the location reading is successfully taken. This is a major drawback but as long as the GPS reading works- which it does in most regions - this will not be a regular problem.

For the folder/gallery option in the App Bar we use the *PhotoChooserTask* class. This allows the user to select a photo from a list of photos saved in the phone by showing their thumbnails.

The Azure Storage Part uses the *WindowsPhoneCloud.StorageClient* assembly file. The downloading portion is done using Azure Storage Explorer which provides the absolute URI of the files saved and hence the source of the image control with the absoluteURI will bring the image onto screen and since the absoluteURI contains details of the container- the other information like location, date/time etc which are stored in the same container as a .txt file can also be extracted.

1.7 Search Types

The actual search is not done on the database in the azure account. The search is done in the phone which contains the information regarding name, date, location and ratings in the application storage. Since this is all text data, the memory usage is nominal. This

is done to ensure that the data usage is reduced to look up the azure account for each search.

The search returns a list of results. The results are ordered differently based on the type of search used. When the user clicks an item on the results list, that photo is retrieved from the database and saved in the phone memory and a smaller version displayed in a popup on the same page. The user either clicks the popup to view the larger photo on the main page, along with the other details, or closes the popup to return to the search results.

1.7.1 Name Search

A substring search is used to look for the names faster. The names are entered into a search array.

The first n characters of the search term are taken as a substring and compared with the first n characters of the names in the array. The number n which is the length of the substring is varied from 1 to the length of the search term.

Any element not matching with the substring is deleted from the array and it is reordered. The search ends when the number of elements in the array is just 1. It is possible that this element might not be an exact match for the search string- however the user will get at least one result so that he can refine his search term in case he mistyped it.

Figure 4 Date/Time Search

1.7.2 Date Search

The calendar displays the dates with photos/notes in yellow and the rest in white. The present day is highlighted and the days yet to come are grayed out.

It follows the same principle as the Name Search only in this case we search based on the number of seconds from the start of 0 CE. Since this is a large number, we do not save the date/time information in this format unless this search is called for.

So this way, although it takes time to convert the date format to the seconds' format, it hastens the search. Furthermore, the simplicity of the search once the conversion is done adds to the advantage.

For storing Date/Time values, the System.DateTime class was used. The DateTime.now returns the Date and Time at the moment of accessing the variable. This can be manipulated into displaying just the year or exact time or month etc. Date and time are stored as one but displayed separately.

Figure 5 Location Search

1.7.3 Location Search

Although in principle this is much simpler (with just comparison of coordinates), there is also the feature of tagging locations which makes it slightly more complicated than a number search. First of all, the name of the location that the user enters must be

matched with the coordinate and then photos corresponding to the coordinates must be listed.

GPS Data is taken using the *System.Device* and *Geocoordinatewatcher* class. It is displayed using *BingMaps*.

In the location page, below the map there are two Listboxes on either side. One of them contains the location names and the other contains the names of the photos corresponding to the location- that is if the user clicks a location he can see the photos in that location on the other list box. If the user wishes to tag the location- he can click the text box and write the name of the location and then it will be added to the database.

Any location not tagged by the user will remain unmarked and there will be no way for the user to search for this location using location name.
He can only click on its name (unmarked) in the list box and then the point will show on the screen and the corresponding photo list will come up on the other list box. After that the user must try to remember the name of the place using the point on the map and the photos and tag the location for ability to search.

Location tagging is among many people and hence even if different names are entered for the same location, all names pertaining to a certain location are combined in one search string and stored. For example, if a user tags a location as “Liberty Statue” while another tags it as “Liberty NYC” then the name of the location will be entered in the database as “Liberty Statue NYC” and any of the 3 search terms will return this location. However, this will work only if both differently tagged locations have the same coordinates. This will be verified by the application- since although the search takes for input the names of locations, the mapping methods works only with coordinates.

1.7.4 Composite Search

The composite search involves combining all three search criterion. It also differs from the other pages in display since Date/Time looks has a calendar, the location page has

a map etc. but this merely contains inputs for all criterion and starts searching based on that.

Order of precedence for search:

1. Ratings (Just one condition- higher or lower or equal. So N photos=N condition checks) Complexity is lower.
2. Date/Time first (The execution time required for conversion to seconds makes this delayed compared to Ratings)
3. Location (Must be searched for matching coordinate pairs. Hence this takes longer than a single condition check like the ones above)
4. Name (The toughest search criterion since a wide net must be cast- so the final search will be this)

1.8 Settings and other pages

1.8.1 Settings

A variety of settings can be managed in this page.

1. The Photo Tab contains settings for the present photo being viewed in the Main Page. If the owner of the photo is the user, then it contains options to rename, delete or change privacy information. If not, these features are disabled (although they appear in the screen but appear as disabled)
2. Delete Settings: When Delete button is pressed, it can be chosen whether Photo stored in the phone is deleted or the one in Azure storage is deleted or both. Same option available for comments.
3. Reset Button- Erases all information on the phone- deleting all photos, comments, ratings etc. and the information about user storage accounts and corresponding friends.
4. Accounts information can be edited here. Also other defaults like maximum photo size that can be uploaded to Cloud etc. can be changed.
5. Extra Tabs can be added depending on additional features.

1.8.2 Help

The help feature at this point only contains basic steps for each of the operations. Those operations (for example, some of the search features) that are self-explanatory are not included in this page.

It can be expanded in later versions.

1.8.3 Ratings

The values in the page of each photo can be changed and the save button is pressed to store those. And when post is clicked they are uploaded to the azure account.

1.8.4 Comments

Contains TextBlocks that display the comments and a textbox wherein a new comment can be entered and posted. When the post button is clicked it will be saved in the Azure account in the comments text file in the container corresponding to the photo.

Each photo contains in its container a text file for comments and another for its vital details.

Figure 6 Comments

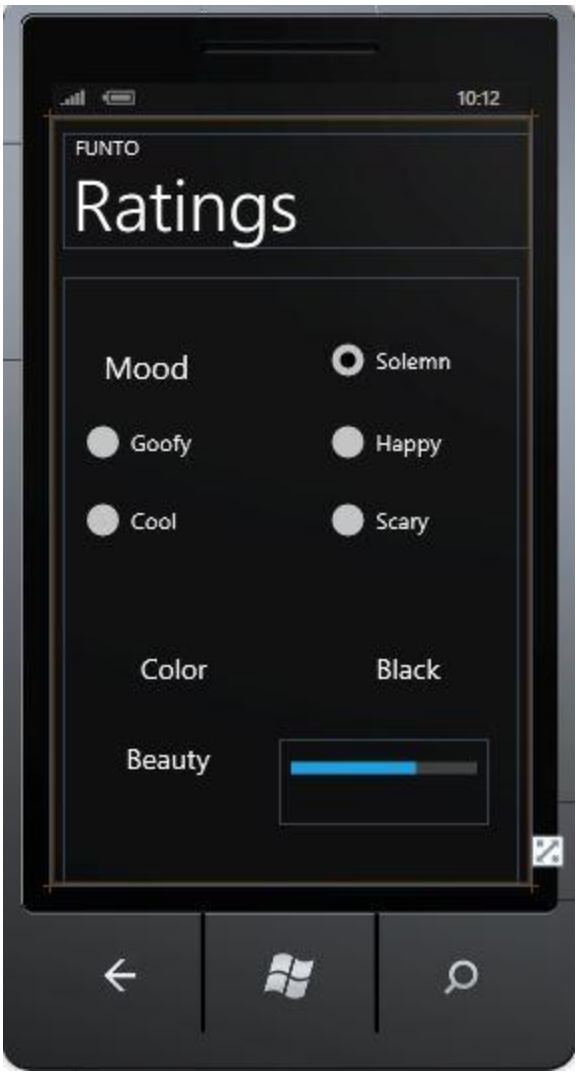

Figure 7 Ratings

1.8.5 Delete

It deletes the presently viewed photo and loads the next photo (in chronological order). Depending on the settings chosen in the settings page it deletes from phone or azure account or both.

1.8.6 Socket Support

The new windows phone 7 Mango Update allows devices to connect using TCP or UDP sockets. Socket is a mechanism for delivering data packets between applications. (Sockets Overview for Windows Phone, 2012)

Since most of our application involves file transfer, we use the connection-oriented stream based TCP sockets.

1.9 App Bar Buttons

The Windows Phone 7 does not allow more than 4 App Bar buttons. However, these can be interchanged depending on the page. The Photoshare App uses a standard four button format and these are either hidden or visible or inactive based on the page or input. New buttons are not added. The four buttons are:

1. Photo Capture
2. Choose Photo
3. One-click Post
4. Save

The Photo Capture button opens the photo capture tool and lets you take the photo. While this tool can be reached in many ways, if the user has to avail of the benefits of this application, then it is imperative that all photos must be taken via this route. So this button is very important. For this reason, this button is visible and active in all pages of the application.

And once the photo is captured it appears on the screen with the other information pertaining to it (like GPS coordinates etc. explained in the Process Flow Diagram).

The Choose Photo is the only option that displays thumbnails of all the photos taken- in the default manner of the other photo albums in the Windows Phone 7. All other options (like Date/Name/Location) only display one photo at a time. Therefore this feature is useful to just glance at the photos in a short time. And once the photo is chosen from the thumbnails it appears on the main page with all its details.

If the user has entered the login details for the Azure account already, then using these, the One-click Post button will upload the presently viewed photo along with its details to the Azure account. A new container will be created if this is the first time the photo is uploaded to the cloud.

The Save button is used when renaming photos, adding ratings, tagging locations- any operation that modifies the properties of the photo. If the Save button is not pressed, any unsaved changes are revoked. The user is not prompted to save any unsaved changes and is also not asked to confirm the changes made. This is present to avoid frequent function calls for updating the Picture object. It also prevents frequent prompts which are usually annoying to the user.

Chapter 2: Daily Log App

2.1 Lack of a Journal or Diary App

1. There was a lack of a diary style application to store notes or occurrences and attach the related photos with them.
2. While there have been Apps like Easy Diary or Hush Hush Diary, they have not served the purpose of a time log and have rather followed the model of Facebook by requiring the user to enter all information (like captions, location info, date/time etc.) about each of the posts/photos that they upload. (Hush Hush Diary App) (Easy Diary : journal app for Windows Phone 7)
3. Furthermore, a swift and easy time management App has not been made available for the phone.

4. Also, a generic calendar App to store reminders or notes was not available.

The absence of the Calendar was remedied with the Mango update but the time management feature was still absent.

2.2 Time Zone Structure

The Time Zones page displays the classification. By default, I have used 7 time zones, viz. Yellow, Red, Purple, Light Green, Dark Green, Light Blue and Dark Blue

Time Zone	Activity
Yellow	Unavoidable Events, Usual activities like Eating, Sleep, Travel, Ablutions etc.
Red	Recreation
Purple	Wasted Time
Dark Green	Useful activities
Light Green	Constructive activities that stuck to a plan
Light Blue	Important Events
Dark Blue	Life Changing events/ Epiphanies

Figure 8 Time Zone Classification (Default)

Most of our daily activities are repetitive so they can be easily grouped into such zones. While zones have been decided by default, they can be altered. Also, activities have not been entered or allotted to zones. Just like the process of

location tagging, the user can add activities and allot them to time zones and these will be added to the library.

2.3 Logging Activities

The process of entering an activity involves the following steps:

1. User chooses day and a row of time slots appear. User scrolls sideways and chooses the timeslot. Three buttons will appear on the right:
    a. Time zone
    b. Activities from Library
    c. Enter new activity
2. Buttons b & c will be inactive until a time zone is selected by clicking on that button.
3. When time zone button is clicked, it opens up a new page with a list of time zones and their classification schemes. This also includes user-created zones.
4. Once a zone is chosen, the control returns to the previous page and the user has the choice to enter a new activity and choose from the list.
5. Once one of them is chosen and the activity entered, it appears on the time slot. The time slot remains selected and if the user would like to add notes then the corresponding button is pressed and the text is added. Several notes can be added.

The display screen consists of a row of cells indicating the activities in 30-minute slots.

The background color indicates the type of time zone and the text indicates the activity done. If there are any additional explanative notes associated with this time slot, then a small arrow icon will appear on the bottom right corner of the cell.

If this icon is clicked it will show a pop-up with the list of associated notes.

This pop-up will also contain an icon to close it (since all pop-ups will prevent input to be passed to the application to any other portion of the screen).

Furthermore, another arrow icon will appear on the top right if any photos associated with this slot are available.
If clicked, it will show a pop-up with the list of photos and the thumbnail of the photos.

2.4 Search Features

Time management App allows you to calculate the time between the last two activities, the frequency of each activity within a time period and the amount of time spent on any activity over a period of time.

These searches are not restricted to activity alone but overall time zones can also be given as input for these searches.

The searches are done by straightforward scanning of every day's activity from start to finish and the based on the number of iterations spanning from the starting day to the end day, the number of days between two dates is found and used to determine the frequency.

The Algorithm to determine Frequency between two occurrences of a specific Time Zone is O(n). Using a better method of populating the slots with activities (in the form of a search tree) would lead to a better response time. This is unrelated to chronological order of displaying the activities to the user.
This option is currently being explored and will be implemented in future versions.
A hash table may also be suitable. However, it may not be as efficient as a search tree in this case since there will be frequent deletions and insertions and regenerating the table each time would be time consuming.

Chapter 3: Performance Analysis

3.1 Monitoring Device Status

According to a research conducted by Greystripe for its US iPhone customer base, the average iPhone App has an engagement time of 9.6 minutes per session. (Greystripe Consumer Insights Report, 2009) (Average iPhone App Sees..., 2009)

Hence it is fair to assume that a person might use our application for a period of 5 to 10 minutes and hence it would be sensible to monitor the status in 60 or 90 second intervals.

If the application has been tested for crashes and if there are existing issues during certain circumstances, then it would be prudent to check the status more frequently when dealing with such situations. For example, if the application is known to crash during service calls to the Cloud or when rotating the image from landscape to portrait, although frequent checks might slow down the system further, it would be useful in writing crash reports later on which will help in eliminating such bugs.
Currently there are no such issues and the application runs smoothly and has not crashed.
The application has been tested for crashes or freezing on an extensive scale with tests on the following:

1. Multiple-button presses triggering events that will be initiated in different pages.
2. Tested for freezing due to fast typing and the consequent delay in the characters typed to appear on the screen.

The *DeviceStatus* class contains methods to obtain information about various properties of the device like Current Memory Usage, Memory Usage Limit, Total Memory available, Power Source being used- i.e. battery or external power- whether onscreen keyboard is being used or not, hardware and firmware version etc.

When the onscreen keyboard is deployed, it prevents input from all other portions of the screen unless it is closed. In this aspect it is similar to the Canvas control which also forces the user to make a decision on the pop-up before proceeding but it also differs from the fact that while a Canvas control cannot be side-stepped- that is closed without making a decision, the onscreen keyboard can be deployed or closed at will without having to make decisions.

The PowerSource method is very useful in allowing the application to determine if the phone has enough battery power to perform certain operations that may require intensive processing or large data transfers (in the order of 10 MB or more).
This method only returns us the information whether the phone is connected to battery power or to an external source.

3.2 CPU Usage Time

The CPU Usage Time of the vital tasks have been measured and used for optimizing performance.

- It takes approximately 1 second to call the GPS service and store the coordinates. High Resolution GPS is used and this is the best case result. In this case, the GPS service is readily available. If the phone was in a region with a poorer GPS signal, it would take more calls and consequently more CPU Usage

Time.

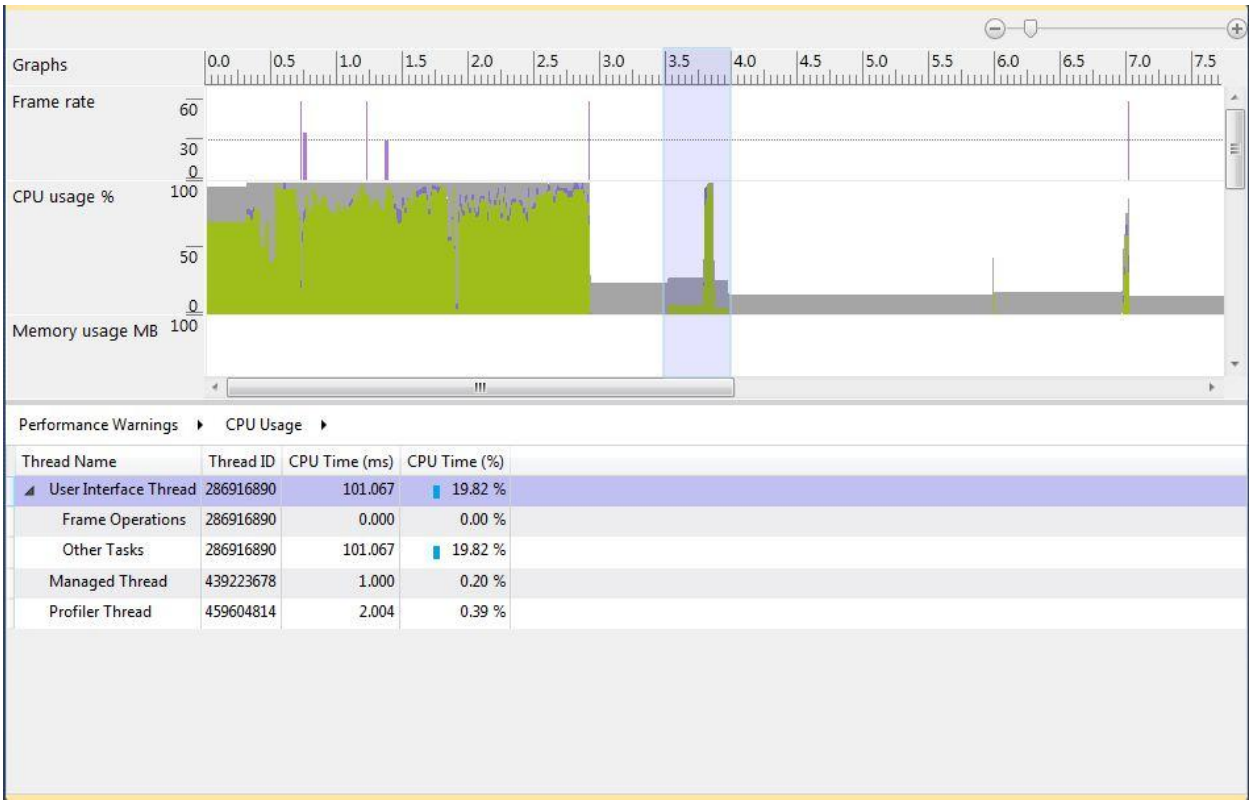

Figure 9 GPS Service Call Time

- The Cloud Service Call and the Framing operations to display the photo take about 0.9 s and use 30% of the CPU time.
- Other tasks including search and result display take up 2.4 s, which is 50% of CPU time.

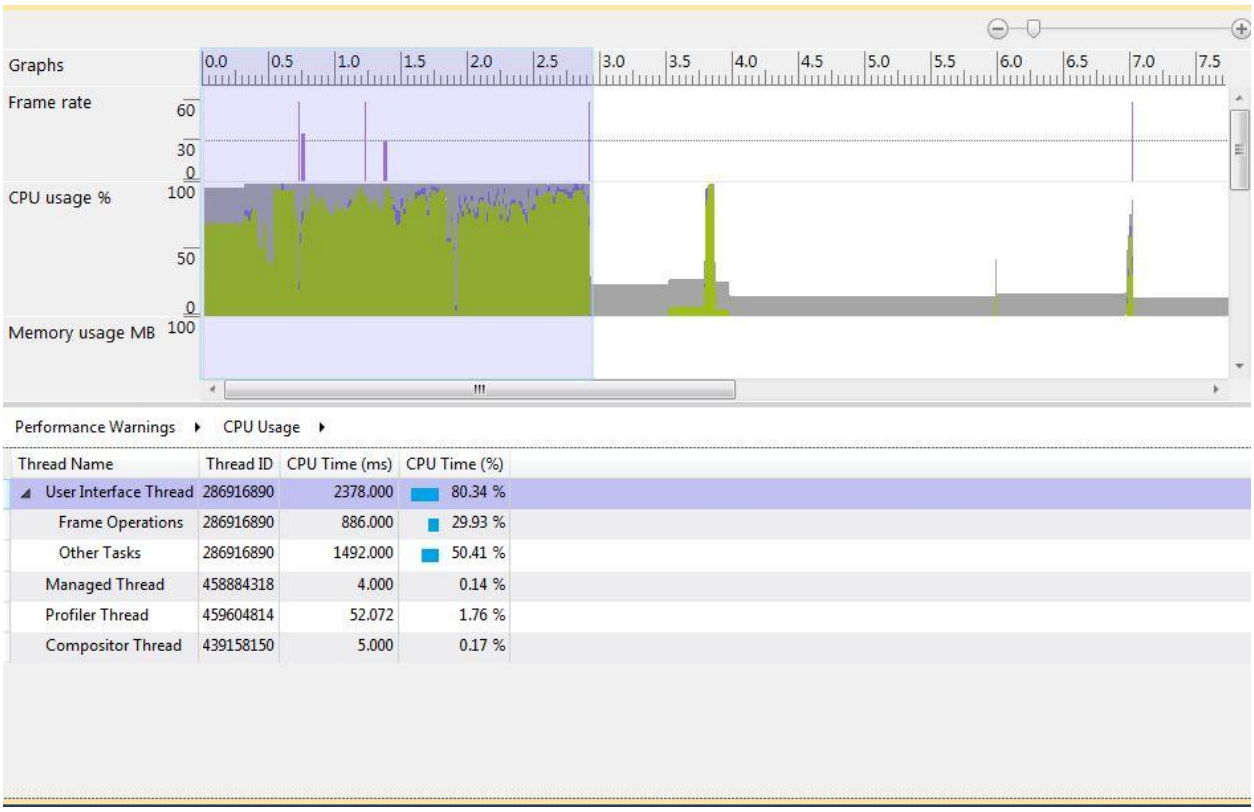

Figure 10 Cloud Service Call and Photo Display Time

The Memory usage graph shows nominal memory usage. However, better memory management measures must be implemented since memory usage increases over time and does not reduce.

Right now, memory is not cleared regularly. It is cleared only when the application is closed or when the user signs out. At all other times, the phone stores in its storage all the photos downloaded from the Cloud. When the application closes or is signed out, all photos, except the one on display at the Main page, are deleted from the phone memory by default.

This option can be disabled from the settings page so that downloaded photos remain in the phone storage even after the application closes.

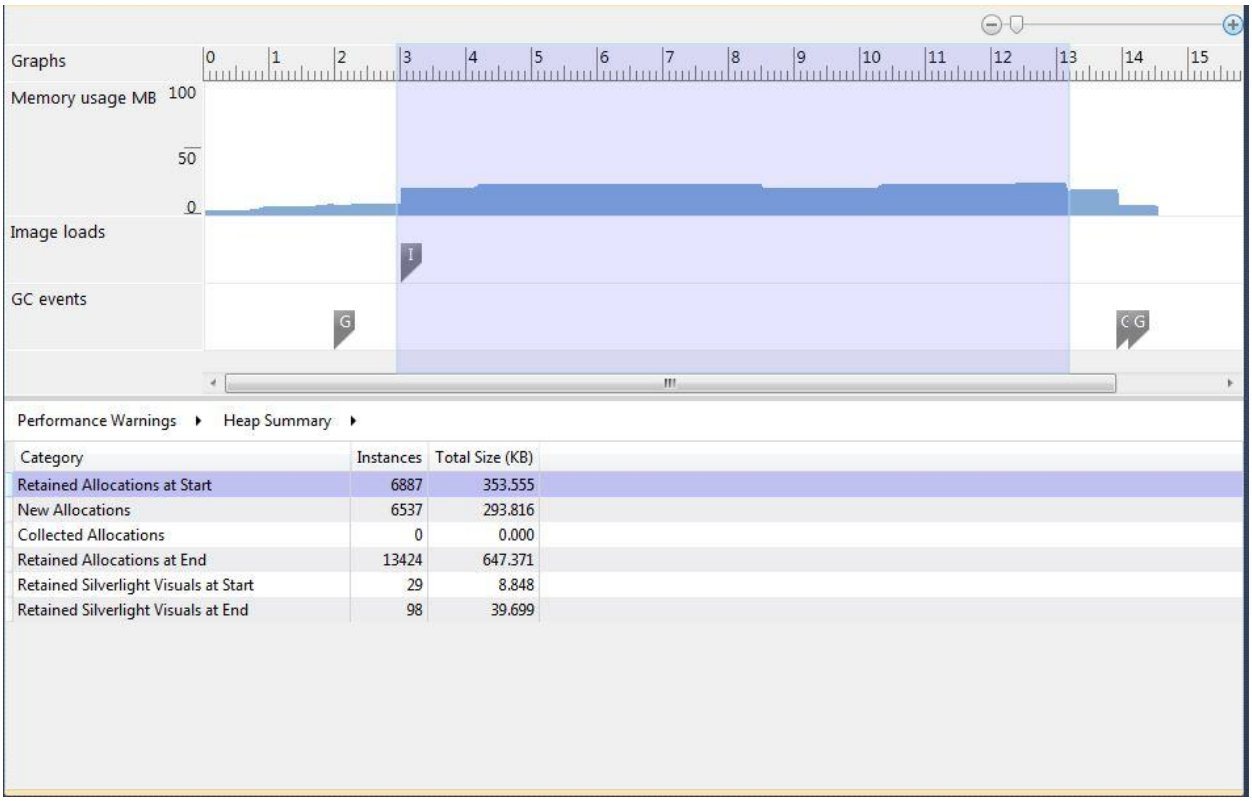

Figure 11 Memory Usage

- Although the time taken for operations is at an acceptable level, the CPU Usage Percentage is very high and this calls for better thread management. This will create problems when background tasks are many.
- The Performance Data for Photo Capture Process is unavailable since the Performance Analysis Tool requires the phone to be plugged into the CPU but in such cases the Camera Capture Task crashes fails.

The application can be divided into three states:

1. Data Acquisition State
2. Data Access State
3. Processing/Idle State

Data Acquisition State

In this state, the user enters input and creates new data. This differs from the Data Access State, wherein, in this state his actions pave the way for new data by way of capturing, deleting, saving or posting photos. This is the state wherein the application also accesses the cloud and the GPS and this state uses the most power.

Data Access State

This refers to the times when the user searches for photos in the database. The user does not initiate new data into the phone through service calls to the cloud. The phone searches in its database text file and returns matching results hence all operations are restricted to within the phone.

Processing/Idle State

The application triggers processes when the user initiates events like capturing a photo or using a search feature.

In the above Process Flow Diagram, the blocks in Green are the times the user gives some input. (1st state) All other blocks are background operations performed by the application (3rd state). That is at all other times the user merely searches the database for existing data. (2nd state)

3.3 Known Issues

1. Application is unable to access the Camera hardware when it is plugged to a PC. Hence the *CameraCaptureTask* returns a blank photo stream when the “Capture Photo” button is pressed. However, all other features of the application work when plugged in- that is the GPS coordinates are collected, the time is noted etc.

   We have been unable to explain this and will be working to rectify this in the future.

   On certain occasions this has also known to forcibly close the application. It is not identified as a crash since the application safely exits

Chapter 4: Conclusion

4.1 Current Work

This documents the current work being done to improve the project.

1. Automatic assigning of photos to time slots and thereby if a user clicks a time slot the corresponding photos can be accessed with one click.
    a. The reason why this has not been implemented is that it would require changing the class structure by making one of the classes (corresponding to either function- Photo-Storage or Time-Management) a base class and the other function as its subclass. This contingency arose due to the fact that each of these applications was developed separately.
2. Location coordinates are not accurately noted. There is no mechanism to detect altitude. So two floors of the same building will contain same location coordinates.
3. If a building is large enough, then two different coordinates can apply to the same building. As stated earlier, one of the rules of locating tagging is that no two coordinates can have the same name tag. However, since this is an exception to that rule, we must account for it in later versions.
4. An undo button to revert changes made to the photo properties. A stack has to be put in place for this.
5. A mechanism to store timestamps for all notes made to the Daily Log App.
6. Mass selection of Photos:
   In order to implement this, a page must contain thumbnails of several photos and must mimic the ctrl+click action of the PC.

4.2 Difficulties Faced

1. Calendar function was not available in Windows Phone 7. I had to modify a user created Calendar, found online, for the date/time search option. Since it wasn't the primary objective of the project (to have a multi-purpose Calendar) it lacked sophisticated functionality. This was remedied in Windows Phone SDK 7.1. (Additions in the Windows Phone SDK 7.1, 2012)

2. Photos taken are always shown in landscape mode in the phone- even those taken using portrait mode. Although this is a major talking point in the developer community the problem has not been solved satisfactorily. This can be annoying for the user.

   As of now developers are discussing a way to rotate all the pixels in the photo using a mapping function but this method has its faults since it has to use a user-created assembly file (which is non-standard, hence might cause other problems, like crashes for instance) and might also tamper with the original photo for just a rotation- if the mapping goes wrong (if the dimensions in the program do not match the dimensions of the photo etc.). While this has worked for the authors of the aforementioned blog, we have not used it in this application.

   Hence this application does not rotate the photo and will always display it in landscape mode.

3. Connectivity to Windows Azure was a major roadblock in the early stages of this project. The assembly *WindowsPhoneCloud.StorageClient.dll* containing the functions for connecting to Azure was not compiled specifically for Silverlight and since such assemblies cannot be used in Silverlight programs, it was a dead-end. (How can I use non-Silverlight assemblies in a Silverlight app?)

   There was a Code Sample for creating a Console application for posting files into Azure and this assembly worked successfully and using Azure Storage Explorer I was able to retrieve these files into the phone.

   So it was only one-way traffic with regards to the phone and the cloud. I was unable to upload files to Cloud while retrieval was possible using the *absoluteURI* property of the file.

   Microsoft acknowledged this error and it was corrected in the Windows 7.5 Mango Update.

   This method had security lapses since using the *absoluteURI* is unencrypted and does not require a password or any form of authentication since anyone can access the files. Hence care was taken not to reveal any code to the end user.

   This is not a suitable solution since containing the login details, the *absoluteURI,* and similar sensitive details in the code makes it vulnerable to hackers.

Using Microsoft's provided Azure toolkit however requires the user to install several certificates and makes it complicated. However, this is a secure solution, although not elegant.

4.3 Potential Pitfalls

1. Any image saved must be have the *BuildAction* property as "Content" to prevent build errors since we deploy a XAP file to the phone and images will be included to XAP only if they are "Content".
   However, the default *BuildAction* for all images is set to "Resource". So it takes one extra statement to be executed for each image to just modify this property before the image can be used. (Content Vs. Resource Build Action, 2011)

4.4 Scope for Improvement

1. Spelling Suggestions
   If the user enters an unmarked location, then there are two options: Either an error statement is displayed or the user must be given possible alternative names of the location (based on spelling error and such). However implementing the second option would require more complexity and hence this is left for the future.
2. Friends Profiles:
   Profile and Stats for friends like number of highly rated photos, most commented photos etc. This would bring the application closer to the interface of the popular social networking websites/applications.
3. Extend Search to Friends' Photos
4. Search Feature in Help Section:
   Users can type in a question or keywords and get relevant responses.
5. Using Hawaii Text-to-Speech for allowing users to post comments by just speaking.
6. Using Hawaii OCR to take photos of printed documents and converting them to text and using them for keyword- searching. (This is not a direct extension of the present application but since it also deals with photos this can be included here).

7. Searching time can be reduced by maintaining a quick search database in the cloud/server.

References

Potential of Cellphone Photography. (2007, February 7). Retrieved from Imaging Resource for Schneider Kreuznach: http://www.imaging-resource.com/NEWS/1170873932.html

Average iPhone App Sees... (2009, May 14). Retrieved from Marketing Charts: http://www.marketingcharts.com/interactive/average-iphone-app-sees-199-uses-and-96-minutes-per-play-9086/

Greystripe Consumer Insights Report. (2009, April). Retrieved from Slide Share: http://www.slideshare.net/dkman/greystripe-consumer-insights-report-q1

Americans and Their Cell Phones. (2011, August 15). Retrieved from Pew Research Center: http://www.pewinternet.org/Reports/2011/Cell-Phones.aspx?src=prc-headline

Content Vs. Resource Build Action. (2011, February 2). Retrieved from Windows Phone Geek: http://www.windowsphonegeek.com/tips/wp7-working-with-images-content-vs-resource-build-action

Photosynth App. (2011, April 18). Retrieved from Style is Violence: http://styleisviolence.com/photosynth-app/

Additions in the Windows Phone SDK 7.1. (2012, March 23). Retrieved from MSDN Library: http://msdn.microsoft.com/en-us/library/ff637516%28v=VS.92%29.aspx

How to: Decode a JPEG for Windows Phone. (2012, March 22). Retrieved from MSDN Library: http://msdn.microsoft.com/en-us/library/ff769559%28v=VS.92%29.aspx

Sockets Overview for Windows Phone. (2012, March 23). Retrieved from MSDN Library: http://msdn.microsoft.com/en-us/library/hh202874%28v=vs.92%29.aspx

Easy Diary : journal app for Windows Phone 7. (n.d.). Retrieved July 2011, from 1800 Pocket PC: http://www.1800pocketpc.com/easy-diary-journal-app-for-windows-phone-7/16500/

How can I use non-Silverlight assemblies in a Silverlight app? (n.d.). Retrieved July 2011, from Stack Over Flow: http://stackoverflow.com/questions/721375/how-can-i-use-non-silverlight-assemblies-in-a-silverlight-app

Hush Hush Diary App. (n.d.). Retrieved April 2012, from Smart Device Central: http://www.smartdevicecentral.com/slide/Windows+Phone+7+ThirdParty+Apps/249189_249195_9_0.aspx

Photosynth. (n.d.). Retrieved May 2011, from Wikipedia: http://en.wikipedia.org/wiki/Photosynth

What is Photosynth? (n.d.). Retrieved May 2011, from Photosynth: http://photosynth.net/about.aspx