University of Limerick

Final year project

Interim Report

GeoTrash

Litter Tracking Smartphone Application

Author:
Patrick Russell -
0734004

Supervisor:
Mr. Michael Coughlan

March 14, 2011

1 Acknowledgements

I am thankful to my supervisor Micheal Coughlan for helping to shape the project by providing ideas on the project structure. I would like to offer my thanks to Gabriella Avram for providing the original project specification, although the project has been changed greatly from this original specification many of the core ideas have been held. I would like to thank my friends for helping me with ideas, advice and support throughout implementation of the project.

Contents

2 Introduction

2.1 General Introduction

A smartphone is a telephone that provides additional information accessing features. Any mobile telephone that combines voice services with e-mail, fax, pager or Internet access is called a smart phone. The majority of smartphones have the ability to install third party applications, the core focus of this project is an application for one of these smartphone devices. The device chosen was an iPhone 3GS with iOS 4.2.

This project is a Geocaching iPhone application targeted at trying to reduce litter. The application works by the user taking a picture of a piece of litter and uploading it to a remote server. This photo can then be uploaded to a remote server along with the location of the device at the time of taking the photo. This data can then be used by other users to identify litter blackspots and help councils perform cleanups in the area.

The project is divided into three areas these are the iPhone application, the web services implementation and the website. The application was developed for the iOS platform for iPhone devices in particular. The application will allow users to take pictures of litter and upload them to a remote database. The geographical location will be stored with this picture on the server and can be displayed on a map by other users. Users can select points on this map and view the photo and details about it. A website will provide a web based representation of the data captured by the application. A map similar to the one on the application itself will be represented with points representing litter location and the image.

The platform chosen for development was the iPhone running iOS 4.2. The development environment chosen was XCode as it is required for iOS development. The language chosen was Objective C which is also a requirement for iOS. The UI design was done using interface builder which is a part of Xcode and is designed for designing interfaces for iOS devices. The project also needed to have a remote database to store photos and GPS locations. For

this a MySQL web server is required along with web services to allow the device to input and retrieve data from this database. Some PHP code was also utilised to handle the web services. The Google Maps API is used on the Website to represent the data in the database on a map for users on a computer to view. The Website will also need to have the function to view and manage photos and account information.

This project is an application for a Smartphone targeted at reducing litter and promoting cleanliness. This application will work by a user locating a piece of litter taking a picture of it. The location of the device when the picture is taken will correspond to the location of this particular piece of litter. This data must be stored in a remote database and can be viewed by other users.

It is a requirement that the application be written in Objective C as this is the language which is used to develop for iOS. iOS is a branch of the C programming language so it bears some resemblance to languages such as C or C++ however it differs in quite a lot of areas. This is a small factor as some time had to be spent to learn this language before beginning. There is also a development environment for Mac OS and iOS applications and this is XCode. The environment also incorporates a UI editor and a simulator to test applications.

A database will need to be implemented to store user/device identies, photos with their geographical location and possibly some other information which would be necessary. I propose to do this by creating an SQL database which the device can connect to to pull and push data as necessary using web services.

The data collected using the application can also be represented on a Website for the users to as with the application view photos and litter spots. Using the location details it will be possible to represent the areas which have high concentration of litter using this Website to highlight key areas so that authorities can take necessary action to reduce the litter.

2.2 Motivation Factors

When choosing my area of focus for my FYP I wanted to focus on an area which interests me while also including the skills which I learned during my degree programme. My programming knowledge was quite good when starting the project however this project brought in a new language which I had not experience with. This is important for future programming work as technology changes quite quickly in the software world so it is important to be versatile. I wanted also to learn about remote data, having data in the cloud is becoming popular in recent times so learning about how this can be done is important. Skills in web development are always useful to have so the website part of the project provides an insight here.

2.3 Objectives of Proposed Work

The objectives of the project are to implement an integrated system based on web an Smartphone application technologies. The project was outlined to have the functionality which was set out in the introduction section but also have a high quality UI and very stable code. Before deciding on the areas of focus on the project I set out to examine similar applications and possible implementation methods. A summary of these findings are outlined in chapter 2. Chapter 3 will outline the design of the system while chapter 4 will outline how this was implemented chapter 5 will present the results of testing the software. Finally chapter 6 will conclude the findings of the project and make recommendations.

3 Research

3.1 Geo Caching

Geocaching as described by techtarget.com: "Geocaching, also referred to as GPS stash hunting, is a recreational activity in which someone "buries" something for others to try to find using a Global Positioning System (GPS) receiver." The iPhone application store or App Store contains hundreds of thousands of applications so it was not hard to locate one which provides similar functionality to this project. There are many Geocaching applications available currently available for smartphones. One of these applications is simply called Geocaching . This is a global treasure hunting game where participants hide and seek physical containers called geocashes and they share their adventures online. After observing this application some inspiration was taken from it. First of all the UI was quite good and the layout is something which will be taken into account when it comes to designing the UI . The functionality of this application also provided an insight. Geocaching allows you to place an object on the map in the application and the position of this object is uploaded to their remote server. This is very similar to what needs to be done for this project. The project also has a map where you can view nearby Linuces this functionality would be a useful addition to the project.

3.2 iPhone Development

There are many development tutorials on the Internet which provide sample code for projects which contain some of the functionality which is required for this project. One of these sample applications is one which takes control of the camera on the phone. The application shows a view which is the live feed from the camera. This application utilises the UIImagePickerController class from the iPhone SDK library. With this class you can decide how you want the camera view to appear and choose whether you want controls for taking photos by specifying it in the code. The UIImagePickerController class is defiantly going to be an important part of the project.

Another important factor in developing the application is obtaining a geographical location. Obtaining the location is done using the Core Location Framework. This framework provides access to all of the iPhones location systems. These include integrated GPS, Wifi-based location positioning and tower triangulation. A sample application was located which will simply display the latitude and longitude on the screen. When obtaining a location there are a number of query parameters which need to be considered one which will need to be taken into consideration is the desired accuracy parameter. The reason accuracy will need to be specified is to conserver battery and to avoid unnecessary updates by having only a level of accuracy which is required.

Another important consideration for the application is the persistence of data, it needs to be decided whether a certain amount of state be saved when the application is closed. Data such as user log in data will need to be saved. This data is usually stored in a location which is designated for saving user data. This memory is referred to by calling the NSUserDefaults class where you can store data of different data types.

3.3 User Interface

The user interface is an important design element in the project. As observed from using XCode to develop some sample applications a tool is available for user interface design. This is known as the interface builder which is a part of the Xcode development environment. With this application you can create and edit views. A view is in simplest terms is what is displayed on a screen at a given moment during the execution of an application, an application can have one or multiple views. The iPhone utilises the cocoa touch API. iOS technologies can be seen as a set of layers, cocoa touch is seen as the highest level and the Core OS and Mac Os X kernel at the bottom. The advantage of this layer is that it makes the implementation of applications easier because the developer is not concerned with things such as how to recognise if a button was pressed. The interface builder contains a library of elements which will be discussed in the design section.

3.4 Web Services

There are various ways in which the data can be sent and received from the remote server. To handle the passing of data between phone and database a web service such as SOAP or REST would need to be implemented. The iPhone SDK incorporates the CFNetwork framework this framework makes it possible to communicate across network sockets using either UDP or TCP. TCP sockets will be used for this project as it is important data is not lost in transmission. The CFNetwork framework also provides support for HTTP and FTP which makes the utilisation of web services possible.

Although the CFNetwork CFNetwork provides all the required functionality ASIHTTPRequest is a wrapper which is used around the CFNetwork API. This wrapper can be used to make it easier to do the more tedious tasks associated with communication with the web server. There are some features of this wrapper which may be beneficial to the project.

Features of ASIHTTPRequest:

- A straightforward interface to submitting data to and fetching data from webservers.
- Download data to memory or directly to a file on disk.
- Easy to access request and response HTTP headers.
- Progress delegates to show information about download and upload progress.

Since it is a requirement that data be uploaded to a web server the above functionality would be quite beneficial in that it would take the tedious nature out of completing the task of uploading. It will also be important to retrieve information such as success information for uploaded data and retrieving data from the server using a request. The easy access to request response HTTP headers will be beneficial here. Obtaining data from the remote server will not be an instant process as regards the users interaction. For that reason progress delegates can be used to show the user how long is left to wait before progression.

With web services there also comes the issue of storing the data. The method decided upon is using a MySQL server. The reason for this is that it integrates well with web services but more importantly the database can be accessed by other media such as a Website which is an important factor for this project.

3.5 Data Persistence

This section will disuss the persisesnce of data in the application with focus on the data which is retrieved from the remote database. When data is obtained from the remote database it is possible to just store it within variables in running code. This would be the easiest solution it seems however there are disadvantages to doing this such as updates would have to be done regularly as data may need to be moved from memory if the application was shut for example. When in an area with no data connection this solution would limit the application to areas within wifi coverage or cellular data connections.

A solution to this problem would be the implementation of a local database which is essentially a copy of the remote one. Some research was done into how this data might be stored. The iOS provides support for sqlite databases. sqlite is a software library that implements a self-contained, serverless, zero-configuration,transactional SQL database engine.

sqlite is an embedded database library that implements a large subset of the SQL 92 standard. Its claim to fame is the combination of both the database engine and the interface (to said engine) within a single library, as well as the ability to store all the data in a single file. I terms of functionality sqlite resides somewhere between MySQL and PostgreSQL. However, when it comes to performance, sqlite is often 2-3 times faster (or even more). This is thanks to a highly tuned internal architecture, and the elimination of server-to-client and client-to-server communication.

All this is combined into a package that is only slightly larger than the MySQL client library, an impressive feat considering you get an entire database system with it. Utilizing a highly efficient memory infrastructure, sqlite maintains its small size in a tiny memory

footprint, far smaller then that of any other database system. This makes sqlite a very handy tool that can efficiently be applied to virtually any task requiring a database.

3.6 Application Map

A map is also a requirement for the application as it is important for the user to be able to browse the map to search for litter. Maps can be added to a view in the application similar to above using the MapKit framework. A map can be added to a view which displays the current location to a specified range. The user can also pinch and drag to move the current position in vision on the map. There is also the issue of adding a marker to the map to display where there is an item of litter. The MapKit framework also provides the function of adding what are called pins, they are simply markers on the map. It is also possible to put code behind one of these pins. This can be used when a user selects a pin a page can load showing the image of the piece of litter and some other basic information. There are various test applications on developer.apple.com/iphone which incorporate MapKit. The application consists of a basic map view which shows the current location. It will also place markers on the screen relative to previous visited locations.

3.7 Website Map

The most popular maps which are used on most websites use the google maps API. Code.google.com provides an explanation of all the different API family items. Google maps API Family:

- Maps JavaScript API
- Maps API for Flash
- Google Earth API
- Static Maps API

A choice must be made here as to which is the best option to implement with. The requirements of the chosen option are that an the map is interactive and is able to display markers

showing litter spots. The option must also enable these points to be selected displaying an image of the litter.

The static maps API can be eliminated here as this option is not interactive.

The javascript option provides all the required functionality mentioned above. The option also includes the option to view the map in Satellite view which would be a useful feature for users of the site.

The Google Earth API allows you to embed three dimensional views of maps. Although it is possible to provide all the functionality with this the three dimensional element is possibly more than is required for this project.

The Flash API is very similar to the javascript version mentioned above in that it provides satellite and basic map functions. The only significant difference to note would be the accessibility of flash v javascript, not all devices support flash but a great deal more support javascript.

4 Design

4.1 Application

4.1.1 User Interface

The user interface for the application required careful consideration before development. Taking into account the examples of other similar applications a user interface design was created. The key requirement of the user interface for this application was to keep it simple. This means limited buttons and menu items. The reasoning for this is that the application serves a simple purpose and it is important not to distract the user from the core functionality of the application.

Figure 1 shows a snap shot of the user interface of the main view. The bar along the bottom is a toolbar. Buttons may be added to this toolbar which perform actions. A simple text view displays an introduction to the application. A bar was chosen as the preferred method of navigation throughout the application as it is simple yet effective.

The map view will simply display a map utilising the google maps API. There are no significant design elements in this view only that it keeps the navigation bar as before.

The photo view which utilises the UIImagePickerController class. This is the only view in the project which the toolbar doesnt follow on the bottom of the screen. The reason for this is so the user will complete taking of the photo before they continue.

The next view will display the image which was taken and ask the user if they wish to upload it. Figure 2 shows an example of how this will appear on screen.

An image view was included in the main section of the application. This view required what is known as a referencing outlet. A referencing outlet is where the view will look for a particular piece of information. Here the theImageView object was referenced for image data. The interface builder also allows editing of colours, shapes and sizes etc. of all UI elements. These tools were used in the final design of the look an feel of the application.

The instructions button will load a screen displaying text based instructions on how to use the application. Again here the toolbar is maintained along the bottom of the screen. Figure 3 shows what this view should look like.

4.2 Maps

The function of maps section of the application is to display a map which zooms to a level close to where the end user is, display a map which shows terrain and roads etc. The map must also display points which correspond to the location of the litter. Once one of these markers are selected an image which corresponds to the point must be displayed.

The iPhone SDK contains an API for Google Maps which is a native application on the phone. Displaying a map was done by loading a new view using the MapKit framework.

There are a number of options which can be set upon loading a new map view. One of these is the showUserLocation method, when set to true this will display a dot corresponding to the users location. It is important to set the accuracy using the desiredAccuracy method. A highly accurate setting on this would cause unnecessary battery drainage due to the requirement that many sensors would have to be used. On the other hand an inaccurate setting would be misleading to the user as to their position so a compromise was made as to the decision. To the nearest ten meters was the setting which was decided upon.

The next item for consideration was the points on the map which correspond to the litters location. The API also contained a class for displaying these points and they are referred to as pins. An Annotation or pin may be added to the current map view by specifying a latitude and longitude. A number of annotations had to be loaded when the application started these points were loaded from the locally stored database then represented on the map once loaded.

4.3 Remote Database

A decision had to be made as regards how long data stay present in the database. It was decided that the data remain in the database as it is not expected that there would be millions of records inserted. The application itself will be designed so that the application is not overloaded with data by limiting the amount of records displayed. The database needs to contain data pertaining to the location of the litter, timestamp the URL of the uploaded photo and the identiy of the device. The Database will be outlined as follows: GeoTrash ID Timestamp Latitude Longitude PhotoURL DeviceID

4.4 Web Services

Sudocode for inserting:

```
int timestamp = currentTimestamp
int latitude = POST(lat)
```

```
int longitude = POST(lon)
int deviceID = POST(devid)

connectToDatabase()
INSERT INTO GeoTrash Values (timestamp, latitude, longitude, deviceID)
CloseDatabase()
```

Sudocode for selecting:

```
result = SELECT * FROM GeoTrash WHERE 1=1
echo result;
```

Deleting:

```
id = POST(ID)
sql = DELETE FROM GeoTrash WHERE ID = id
echo success
```

4.5 Application Back-End

This section will discuss the implementation of the coding of the application. Figure 4 shows a class diagram of the application.

4.5.1 Description Of Structure

Image Picker: The purpose of this class is to manage the initiation of a new photo view and carry out the photo taking operations.

Annotation: This will handle the points on the map referring to pieces of trash and the images connected with them.

Cacher: This will handle updating of the local database.

GeoTrashViewController: This will handle the UI elements of most of the application.

ImagePicker: This will manage the taking of photos.

TrashPiece: This will be an object which contains info for each trash piece which is currently in memory. Holding things like latitude, longitude etc.

The pink interfaces above show interfaces which are provided by the API which are required for functionality by classes mentioned.

4.6 Web Services

For the data to move between the web server and the application and the website web services had to be implemented.

5 Application

5.1 User Interface

The design of the API has been mentioned in a previous section this section will discuss the inner workings of the user interface and the issues which occurred during development.

Interface builder objects such as buttons have a predefined list of events which can occur on the object. These events can be linked up with classes in the code to be invoked by a chosen event. For button items the most commonly used event is touch up inside which causes an event when a button is pressed. The event can be linked up to classes with the IBAction identifier in the project.

Once the application is launched the first thing which happens is the GeoTrashViewController interface file is loaded to the screen. If the a photo option is selected the photo view is presented. Here is the code which causes this to happen:

```
-(IBAction) getPhoto:(id) sender{
// load the image picker view
```

```
UIImagePickerController * picker = [[UIImagePickerController alloc] init];
picker.delegate = self;

if((UIButton *) sender == sentPhoto) {
picker.sourceType = UIImagePickerControllerSourceTypeSavedPhotosAlbum;
} else {
picker.sourceType = UIImagePickerControllerSourceTypeCamera;
}
//make visable
[self presentModalViewController:picker animated:YES];
}
```

The significance of IBAction here is that this piece of code is called when a specific event occurs in the interface. The first line is the allocation of memory and initialisation of the data for the picker object. The options are set for the source type for camera. The view is then set to appear above the main view by setting animated to YES. When this view is no longer needed or the user selects another menu option removeMapFromSuperview is called on the view object, this removes it from vision. Since it is the same implementation for adding and removing other views this document will not continue to explain how the other views were implemented. theImageView refers to the image view object created in the interface builder. This view is invisible when there is no image connected with the view but after the user takes a photo this image is set in this line of code: self.theImageView.image = [info objectForKey:@"UIImagePickerControllerOriginalImage"]; The UIImagePickerControllerOriginalImage key refers to the image which was taken using the image picker.

5.2 Uploading Images

This section will discuss uploading the image which was obtained from the image picker. imageUploader is the operation which handles this.

5.3 Building Local Database

The local database contains the same rows as the remote database however when the application is started it contains no records. Once the application loads the following method is called:

```
- (void)populateLocationList:(id)sender
{
cacher = [[Cacher alloc]init];
id number;

//[number locationsArray];

number = self.cacher;
    [number buildDatabaseFromRemoteData];

}
```

This method creates a new cacher object and calls buildDatabaseFromRemoteData on the object.

```
-(void)buildDatabaseFromRemoteData{


// Starts the asynchronous request which calls for all the rubbish items from the remote

locCache = [[NSMutableArray alloc] init];

    NSArray *documentPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NS
    NSString *documentsDir = [documentPaths objectAtIndex:0];
    databasePath = [documentsDir stringByAppendingPathComponent:databaseName];

NSURL *url = [NSURL URLWithString:@"http://www.skynet.ie/~paruss/iPhone/getLocations.php
```

```
ASIHTTPRequest *request = [ASIHTTPRequest requestWithURL:url];
[request setDelegate:self];
[request setDidFinishSelector:@selector(requestFinished:)];
[request startAsynchronous];

//check and create the database

[self checkAndCreateDatabase];

}
```

The code above starts an asynchronous request for the data in the remote database. The URL points to a PHP file which obtains data from the SQL server. The significance of it being an asynchrous request is so that the user does not have to wait while this is taking place. Once the request has finished requestFinished is called. This method will firstly check if the local database exists, if it does it will delete everything from it as this data may be old.

```
NSString *responseString = [request responseString];
NSLog(@"Response: %@", responseString);
NSArray *chunks = [responseString componentsSeparatedByString: @" "];
```

The response from the request is loaded into a string. The data must then be split up by the spaces to separate the string into chunks containing individual records to be re-assembled to insert into the local database. The following is a snippet which performs the operation of iterating through the list of items and inserting them into the local database:

```
int count = [chunks count];
//iterate through each item
int i = 0;

while (count >= 3)
{
```

```
const char *sqlStatement = "insert into cache(id, timestamp, latitude, longitude) VALUES

if(sqlite3_prepare_v2(database, sqlStatement, -1, &addStmt, NULL) != SQLITE_OK)
NSAssert1(0, @"Error while creating add statement. '%s'", sqlite3_errmsg(database));



        int ID = [[chunks objectAtIndex:i] intValue];
int ts = [[chunks objectAtIndex:i + 1] intValue];
double lat = [[chunks objectAtIndex:i + 2] doubleValue];
double lon = [[chunks objectAtIndex:i +  3] doubleValue];

sqlite3_bind_int (addStmt, 1, ID);
sqlite3_bind_int (addStmt, 2, ts);
sqlite3_bind_double(addStmt, 3, lat);
sqlite3_bind_double(addStmt, 4, lon);
```

Once a photo has been taken the Upload method is called and is outlined here:

```
- (void)Update:(id)sender
{
NSString *latPost = self.lat;
NSString *lonPost = self.lon;
NSURL *url = [NSURL URLWithString:@"http://www.skynet.ie/~paruss/iPhone/Uploader.php"];
ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:url];
[request setPostValue:lonPost forKey:@"lon"];
[request setPostValue:latPost forKey:@"lat"];
[request start];
NSError *error = [request error];
if (!error) {
```

```
NSString *response = [request responseString];
NSLog(@"Output", response);
}
}
```

From the code above we see that the URL is set for the request. The POST values are then set for the request. This code snippet uses the ASIHTTPRequest wrapper. The request is then started.

The map is loaded by loading a new map view similar to the photo views mentioned above. Once the map is loaded a dot will appear on screen displaying the current position of the device. The desiredAccuracy variable of the map view must be set to define how accurate the updating of the current position must be. The chosen value was to within ten metres. This is a value which is accurate enough for the user to identify nearby litter but not too accurate that it drains the battery in the device. Once the map is loaded all the values in the local database are loaded and the Annotations of the litter are placed on the screen.

5.4 Google Maps API

The Google Maps API was chosen to represent litter data on a map for the website. The javascript API was chosen for embedding on the website. The data is retrieved from the server using web services which are implemented in PHP. A request will be made to retrieve location coordinates. Once returned they are loaded into the javascript code to be displayed on the map.

The Google Maps API allows you add what is known as an overlay. Many different kind of overlay items can be implemented but the one being concentrated on for this project is markers. Markers are quite similar to annotation pins which were mentioned in the application.

Markers put simply identify a location on a map and can be specified using geographical coordinates.

6 Testing

6.1 Application

XCode provides support for test cases using OCUnit. Test cases help ensure health of the code. Below is a sample test case which tests the x.

6.2 Web Services

The web services were tested using web forms to insert different types of data. The PHP code is quite simple for this project so testing was not a large requirement.

7 Appendices

7.1 Glossary

A cocoa touch is a very useful addition to any technical document, althoug TCP

7.2 References

Glossary

3GS Third generation of the iPhone . 4

API Set of rules or specifications which a software program can follow. . 5, 8, 9, 11–14, 16, 21

App Store An application used on iPhone devices to download native and third party applications . 7

ASIHTTPRequest an easy to use wrapper around the CFNetwork API that makes some of the more tedious aspects of communicating with web servers easier. . 9

CFNetwork A framework in the Core Services framework that provides a library of abstractions for network protocols.. . 9

cocoa touch An API for developing user interfaces for iOS devices. 8, 23

Core Location The Core Location framework lets you determine the current location or heading associated with a device. . 8

FTP File Transfer Protocol. Network protocol used to transfer files . 9

Geocaching A pastime in which objects are hidden at secret locations for participants to find using GPS positions posted on the Internet. . 4, 7

Google Maps A web mapping service provided by Google . 5, 13, 21

GPS Global positioning system . 4, 7, 8

HTTP Hyper Text Transfer Protocol. 9

IBAction Identifier used to qualify an instance-variable declaration so that Interface Builder can synchronize the display and connection of outlets with Xcode. . 16
iOS Stands for iPhone Operating System is a mobile operating system for use with Apple mobile devices. . 4, 5, 8, 10
iPhone A line of internet and multimedia enabled smartphones designed and marketed by Apple inc. . 4, 7, 8, 13

Mac Os X Unix based operating system developed by Apple. 8
MapKit Framework for developing map applications for iOS devices. 11, 13
MySQL Relational Database management system . 5, 10

NSUserDefaults Provides access to the default settings in the device.. 8

Objective C Object oriented programming language . 4, 5
OCUnit Request to remote server where further code cannot be executed until the request has finished. 22

PHP Server side scripting language which is HTML embedded. . 5, 21, 22

REST Representational state transfer. 9

SDK Software development kit. When spoken about in this document referring the the iPhone OS SDK. This is a software development kit which provides tools to develop iPhone applications. The development kit is available to third party developers.. 7, 9, 13
Smartphone A telephone that provides additional information accessing features. Any mobile telephone that combines voice services with e-mail, fax, pager or Internet access is called a smart phone. . 5, 6
SOAP Simple object access protocol. 9
sqlite a software library that implements a self-contained, serverless, zero-configuration,transactional SQL database engine. . 10, 11

TCP Transmission control protocol. 9, 23

UDP user datagram protocol. 9
UI User interface . 5–7, 13

UIImagePickerController class manages customisable, system-supplied user interfaces for taking pictures and movies on supported devices, and for choosing saved images and movies for use in your app. An image picker controller manages user interactions and delivers the results of those interactions to a delegate object.. 7, 13

XCode Development environment for Mac OS applications . 4, 5, 8, 22

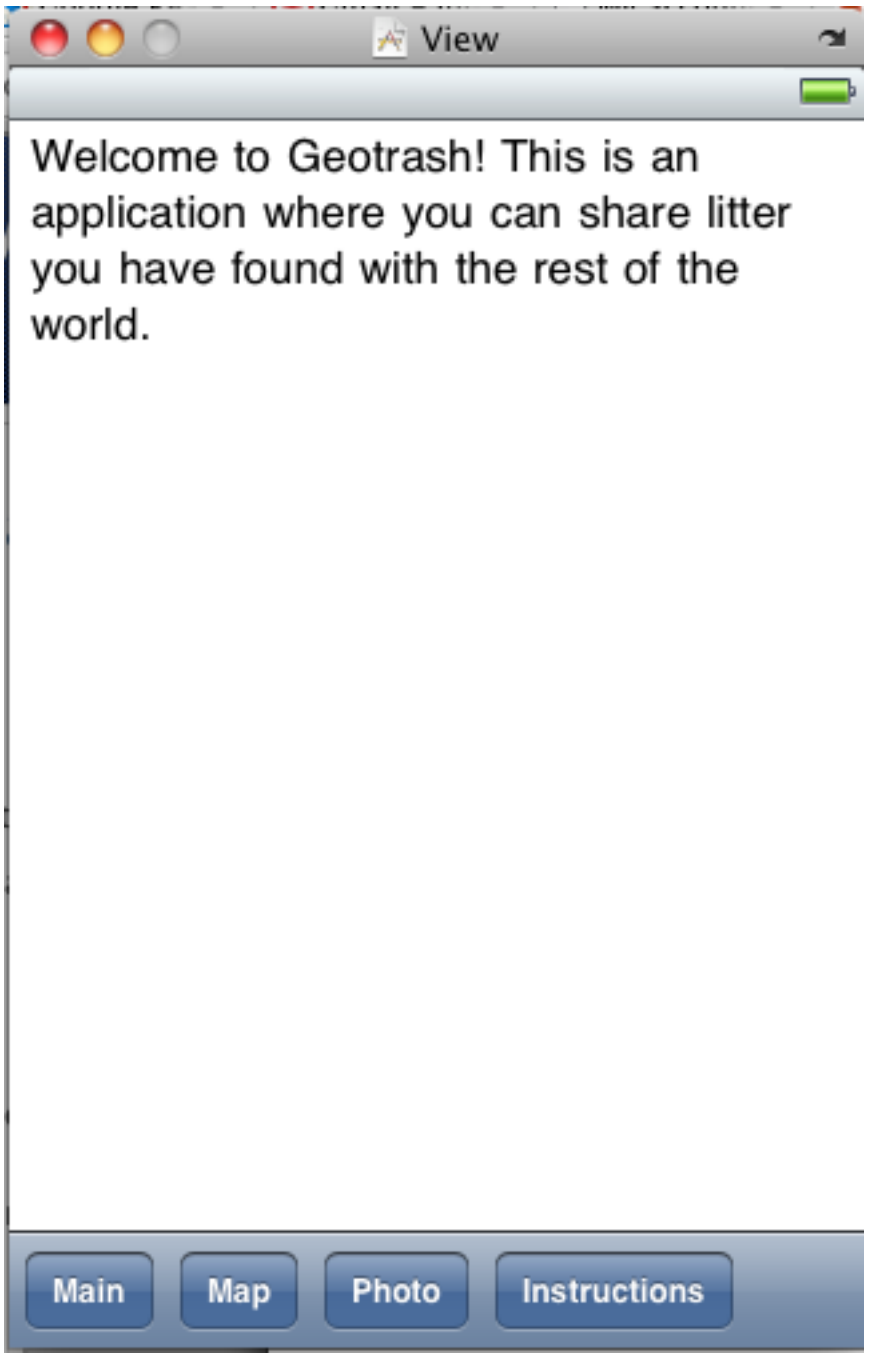

Figure 1: Main View

Figure 2: Upload Image View

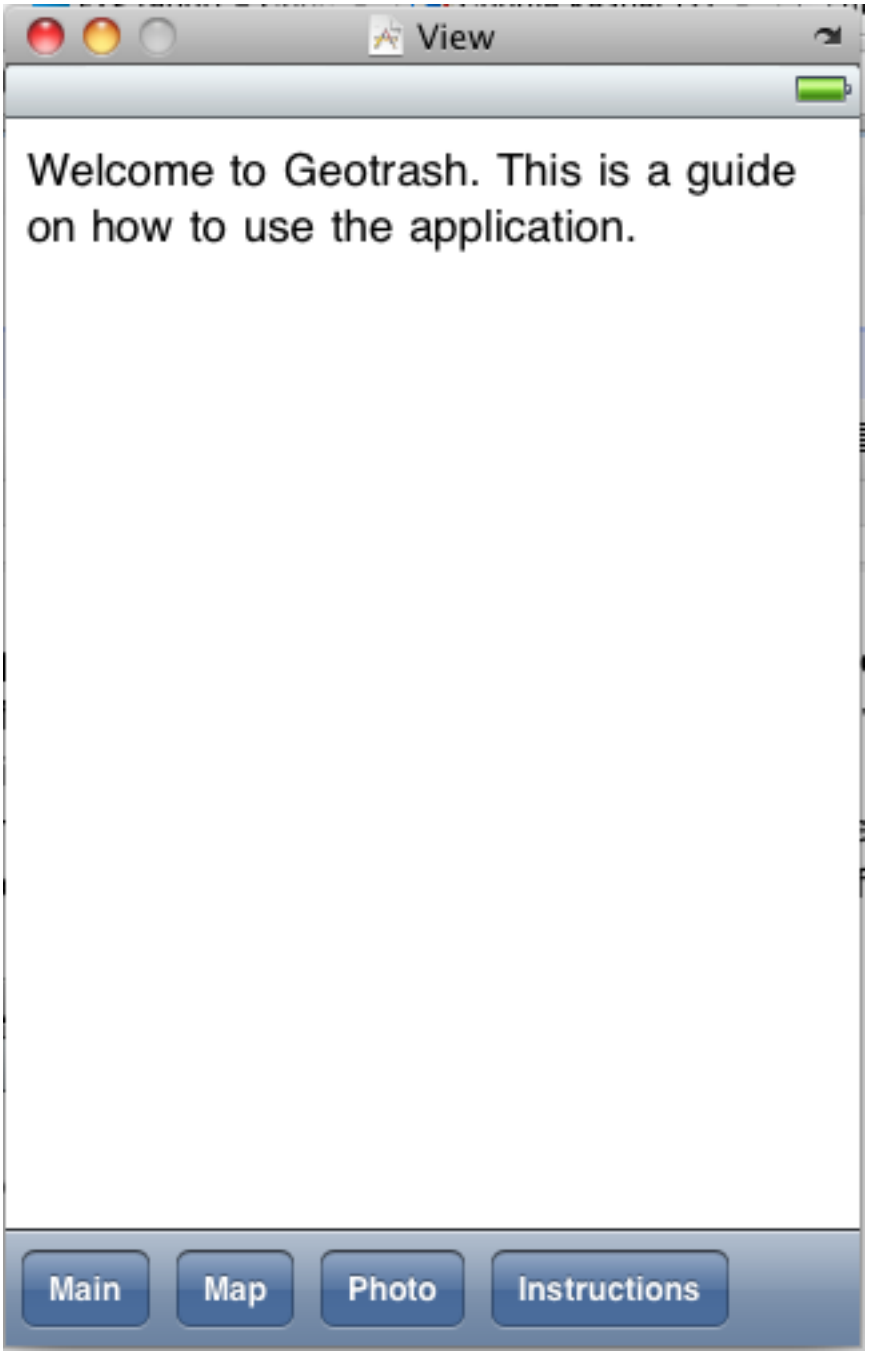

Figure 3: Instructions View

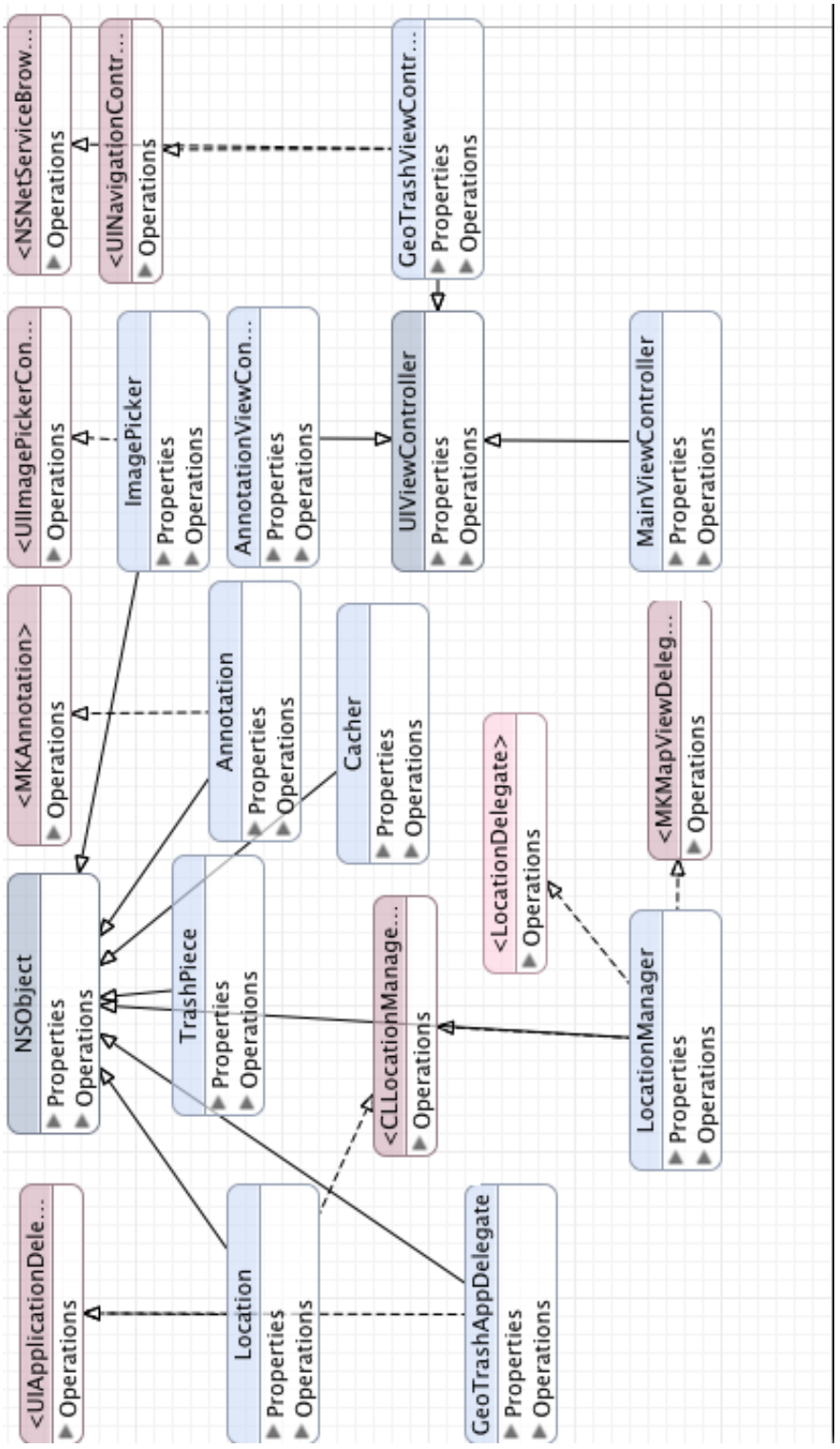

Figure 4: Design Class Diagram