Fall 2011

AUGMENTED-LIFE PHONE ORGANIZER

Chao-Hsin Shih
San Jose State University

AUGMENTED-LIFE PHONE ORGANIZER

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Chao-Hsin Shih

Dec 2011

ABSTRACT

AUGMENTED-LIFE PHONE ORGANIZER

by Chao-Hsin Shih

Augmented-Life Phone Organizer (ALPO) is a geo-based mobile application for users to search for businesses and make corresponding actions such as calling, navigating, or viewing comments about the business. Users can take image notes and attach them to a virtual wall, so others can view the image at the same GPS coordinates later. Users can also view all business images and augmented notes superimposed on the real world through a mobile device screen and camera. We have measured application usability and feature practicality by conducting function tests on three users, and the results are encouraging.

ACKNOWLEDGEMENT

I would like to express my deep appreciation to my project advisor, Dr. Chris Pollett, for his support, encouragement, and supervising. This project would not have been completed or written without his technical supports, guidance, proofreading, and innovative ideas from our weekly meetings.

I also want to thank to my committee members Dr. Teoh Soon Tee and Dr. Khuri Sami for their directions, and helpful advices along this project.

Finally, I want to appreciate my friend Ming Tsai, who gave me lots of valuable suggestions on database design, helped me to proofread my report, and helped me testing the results. I also want to extend my appreciation to my coworker Joan Andrews, who helped me to proofread my report. Last, I want to thank for my friend Yun-Chieh Lin, and Wayne Tang for helping me collect reference papers and do testing.

Table of Contents

List of Figures

List of Tables

1.0 Introduction to the Project

Augmented Reality has become one of the most popular mobile application topics. Improvements in mobile device hardware allows developers to utilize sensors such as GPS, magnetometer, gyroscope, accelerometer, and cameras to capture information related to the user's surroundings. Augmented reality makes information that cannot be seen by the naked eye visible by overlaying virtual objects in the real world in real-time.

The iPhone is a smartphone designed by Apple that function as a small PC, allowing users to take pictures, surf the Internet, check e-mail, chat via videoconferencing, and play videos, music, and games. The first generation iPhone was released in 2007 and has already become a leading mobile phone because of its user-friendly interface and complete libraries [1]. Today, more and more engineers have devoted effort to mobile device development and have made hundreds of thousands of useful applications. People use mobile devices to get information, such as the best nearby restaurants, restaurant reviews, coupons, directions, and price comparison. Most of social network services such as Facebook, Yelp, and Twitter have ported to mobile platforms so that people can share and view posts anytime and anywhere.

Because of the mobility and portability of mobile devices, popular social networks (e.g., Facebook, Yelp, Twitter) have developed applications for mobile devices. According to the top iPhone and iPad app downloads list since mid-2008, when the App Store opened, Facebook tops the list for free iPhone applications [2].

Augmented-Life Phone Organizer (ALPO) is an application that I wrote for the

iPhone platform. The app will help people find information by using the iPhone to Augment Reality naturally. Users can save a location where they parked their car and easily find it later by navigating to the saved parking space. When users search Points of Interest a list of nearby businesses in the area is displayed with possible options for navigation. Users can call, or view reviews of the businesses. They can also take a picture and share it so that other users in the same area can see the pictures later.

2.0 Definition of Terminology and Libraries

Before we begin digging into the iPhone world, we need to know the meaning of the following terms.

Integrated Development Environment (IDE) – a software application that provides developers with a graphical user interface that integrates several tools that can help maximize a developers' productivity.

Software Development Kit (SDK) – a term for development tool that allows developers to utilize the native library to achieve their goal.

Cocoa Touch – a framework that is a SDK, which share similar patterns with Mac but specializes on touch-based interfaces.

Object-oriented – a programming concept that separates the objects, methods and interactions.

ANSI C – a standard published by the American National Standard Institute that encourages developers to follow requirements so that the code will be more portable.

Objective-C – a programming language based on an object-oriented concept that extends the standard ANSI C language.

iOS – Apple's mobile operating system that was originally developed for iPhone that can be extended to iPod touch, iPad, and Apple TV.

JavaScript Object Notation (JSON) – light-weight text-based data derived from Javascript. JSON is primarily used for serializing and transmitting data between the server and client side.

Document Object Model (DOM) – represents objects in HTML and XML documents, which is cross-platform. Each object is represented as an element that can be manipulated.

2.1 Libraries

In this chapter, we will introduce several libraries that are used in this project. All libraries used are free, and some of them are open sourced. We will explain what these libraries are doing and how are we going to use them.

2.1.1 Facebook

Facebook is a social network, which launched in February 2004, and it has hit 800 million users as of July 2011. Users can connect with friends, exchange messages, organize events, create group chats, post pictures, and share news or links. Facebook launched an iPhone application at August 2007, which has over 350 million users each month. The Facebook platform was then released at May 2007 [3]. The Facebook platform, which supports Single Sign-On (SSO), enables developers to integrate the Facebook social experience into their own mobile applications.

2.1.2 Yahoo Local Search Web Service

This service allows users to search for business, location (can be an address, zip code, or the latitude and longitude), or even route. We will be using this service to provide Points of Interest' information. Since each business has its own ID, we will store

the ID in our database for other purposes.

2.1.3 CloudMade

CloudMade is a company integrating the OSM data and producing APIs for map rendering, route calculating, and other geographic-related services. The style editor allows the user to customize the color and layers of Google and Yahoo maps and export the style numbers for further purposes. The developer can also assign the map style by given style number, which may be generated by the developer or retrieved on the map library. Although CloudMade only supports a few major cities in the United Kingdom, Ukraine, Germany, and the U.S [4]., the routing API is pretty well developed, and was easy to integrate in our project.

3.0 Design and Implementation

People like to share their daily life with others on Facebook. Examples are the food they just had, the places they went. These posts could be a valuable source of information when other users need to know about businesses or places or for businesses to get feedback from users. Yelp provides people business search, users reviews, and ratings. Since Yelp is not focusing on message exchange, users cannot reply to reviews. ALPO's Comment function allows users to leave comments and replies for businesses.

In Facebook, users cannot add location or date information to albums or photos. ALPO's "Image Note" component allows users to upload images from Facebook friends based on their current location, and view them on the real world camera view. We will discuss how to implement the augmented reality techniques in later chapters.

3.1 Design and Implementation Challenges

Since we are building a mobile application that targets iPhone platform, limited screen size is the first factor to be considered when designing functions and user interfaces. For example, on PCs or tablets, we can display all other information on the same page. On mobile phones, we need to deploy a hierarchical view mechanism to present details. User interfaces should be clear, self-explanatory, and easy to interact with; users should be able to start using it without needing explanation**.** In ALPO, we display business names and ratings in a list view, allowing users to view details with a single click.

Another critical issue we have faced is processing speed. Mobile phones are normally equipped with weaker CPU chips, slower memory access, and less memory compared to low-end PCs. Internet bandwidths through iPhone devices are usually lower than that of PCs, which prevents us from sending a large amount of data. In ALPO, we adopt JSON as the server response data type to reduce the bandwidth load.

Heavy computations or device tracking on mobile devices may cause additional power consumption. So we needed to avoid complicated calculation, heavy rendering on a mobile device, due to those limitations. We also need to be careful when dealing with threading, which may cause blocking of UI rendering or even the unexpected resources releasing. In ALPO, HTTP request are running in the background while the loading icon displayed on the main thread.

Other than equipment limitations, since iOS is a well-supported framework, it has many resources, third party APIs, libraries, and frameworks that allow developers build their applications easily. Although there are many third party libraries, we need to estimate carefully before deciding on one that fits out needs. There are some factors we need to consider before adopting a new library. For example, does it use any deprecated APIs? Does it support only newer versions? Performance? Is there any conflict between the selected library and others? Is it flexible enough to customize and fit into our goals? We need to make sure each library pass those tests before integrate into our project. In the experiments chapter, we will show and explain the result of libraries we have experimented with and how did we decide which libraries to adopt.

3.2 Architecture

The architect of ALPO is based on a server-client model. The mobile device acts as a client that interacts with users and the web server that retrieves data from a MySQL database, which is hosted at the same domain as the web server (See Figure 3.1).

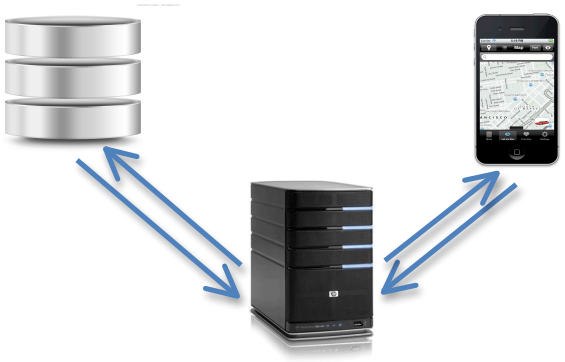

Figure 3.1. Server-client architecture

The server side is running on Apache/2.2.21 (Unix), MySQL 5.1.60, and PHP 5.2.1. On the client side, we developed our app using Xcode, which has the iOS development environment including SDKs, on Mac OS X. Once we finish setting up the working environment on both server side and client side, we are ready to go to the next step.

3.3 The MVC Design Pattern

iPhone programs are written in Objective-C. Objective-C encourages developers to adopt the Model-View-Controller (MVC) design pattern, which is a software

architecture that focuses on isolating data from user interfaces and connecting them via controller. The Model manages the data in the application. The View renders the user interface. The Controller receives user input and performs actions based on instructions. Figure 3.2 shows the relationships between those three patterns.

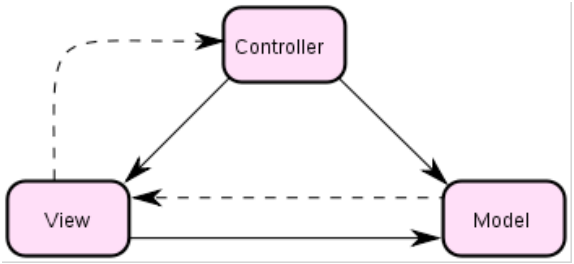

Figure 3.2. Model-View-Controller concept. Solid lines represent direct association while dashed lines represent indirect association.
Source: Model-View-Controller (n.d.). Retrieved December 01, 2009, from Wikipedia site: http://en.wikipedia.org/wiki/Model–View–Controller

Since iOS is based on object-oriented Objective-C, we adopted the MVC design pattern to meet the object-oriented programming language design principle. The complete MVC development design diagram is shown in Figure 3.3.

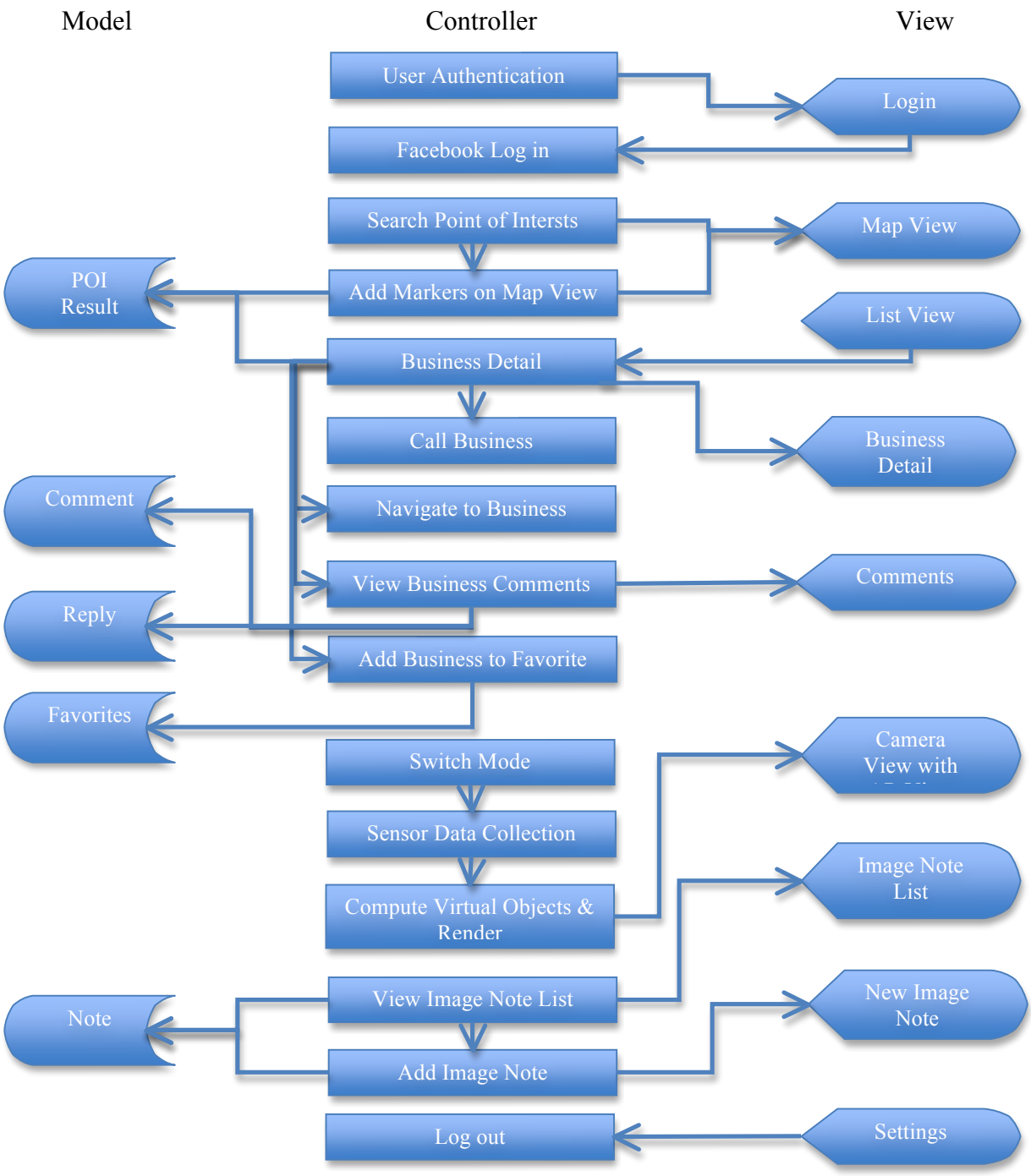

Figure 3.3. MVC class diagram

3.3.1 The Data Model

ALPO's Model layer structure consists of data objects that are managed by the application, such as business details, business reviews and replies, and image notes. Three customized data models serve as intermediate data stores for storing remote third-party API data retrieved.

3.3.2 The Interface View

The Interface View layer handles the task of presenting information to users and getting input from users. In our case, this layer contains the main windows, tabs, all UI elements, tables, map view, and Augmented Reality view in the client application. In addition, the interface view elements need to bind with controllers that can take user input and provide responses.

3.3.3 The Controller

The Controller layer acts as a bridge between the Model layer and Interface View layer. The Controller receives user input from the View layer and makes corresponding changes to the Data Model. For example, when users type words on the search bar and click the "search" button, the Interface View generates notifications and sends them to the Controller after receiving input from users. Once the Controller receives these updates, it sends the request to the server to query the business data from the Yahoo library regarding user input keywords. The Controller then parses the response data, updates the list view, and overlays the markers on the map view.

3.4 Supported Features

In this section, we will introduce the main components and the corresponding functions for this project. Figure 3.4 shows the user case diagram. According to our goals and requirements in section 3.1, there are two main components in ALPO: the Points of Interest and Image Notes. For Points of Interest, ALPO implemented following functions:

- Search businesses
- Get business details
- Add business as a favorite
- Call businesses
- View or leave comments for businesses
- Plan routes and direct users to businesses

For Image Note, ALPO separates viewing privileges of different users in the following way:

- Everyone nearby – includes Facebook and non-Facebook users
- Facebook Friends
- Specific Facebook friend group
- No anyone else

In ALPO's Augmented Reality view, both the searched business within the vicinity and the Image Notes are displayed and scaled by distance in the Augmented Reality view when the camera points towards the targeted Points of Interest. This

Augmented Reality view provides an easy one-look experience that presents the users' desired set of items. Figure 3.4 shows the user case diagram.

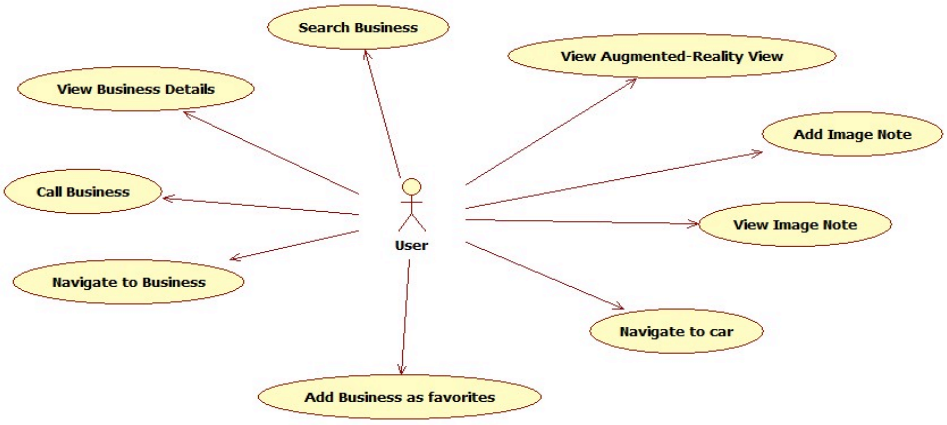

Figure 3.4. User Cases diagram

3.4.1 Points of Interest

When users switch to the main view, "Let me see" tab, a navigation bar is displayed on top, and then a search bar and a map view covers the lower half screen as shown in Figure 3.5. In the navigation bar, a segmented control is on the left side of the bar title and two buttons on the right side. The segmented control allows switching between map view and list view mode.

The "Park" button allows users save their parked car location. The eye image button on the far right allows the user to switch to the Augmented Reality view.

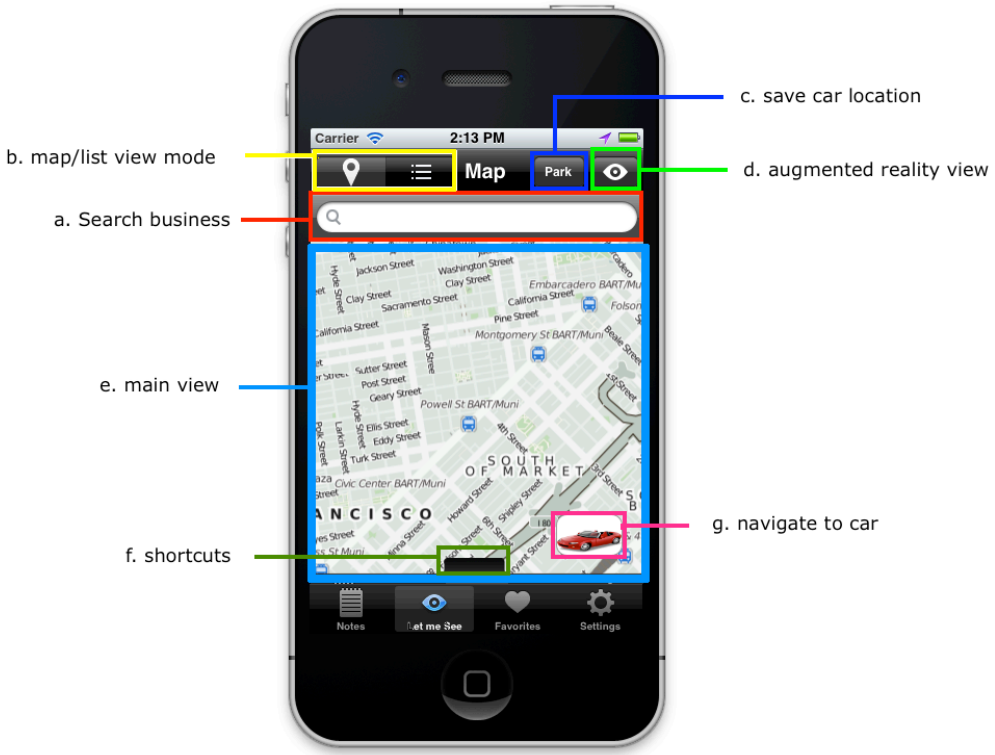

Figure 3.5 Business search screen

3.4.1.1 Search Business

Users can search nearby Points of Interest by either using a partial or full name of the business, category, or a combination of the name and category. For example, if users need to find the nearest fast food burgers, they can type “fast food”, “burger”, or “fast food burger” for keywords in the search bar to get all businesses based on users’ current location. Once the user finishes the search, the map view displays updated pins or markers and the list view is updated. Figure 3.6 (a)(b) shows the result in the list and map view mode of the searched keyword “burger”. Frequently used keywords such

as nearest gas station, ATM, and parking space are added to the bottom bar. Tapping on the pull-tab can hide or extend the bar.

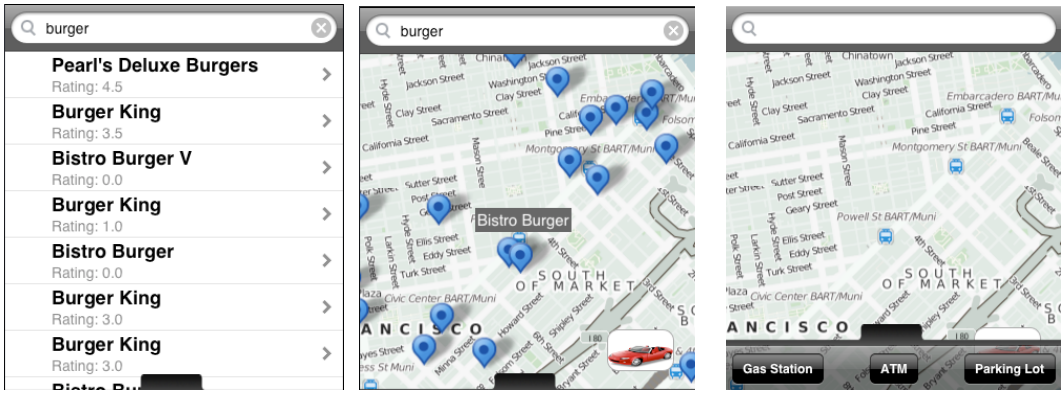

Figure 3.6(a)(b). Search result in list/map view mode (c) shortcut menu bar

3.4.1.2 Get Business Detail

If users click any of the list items, a details page will display (See Figure 3.7). The business detail page includes the address, distance, phone number and rating of the business. Users can see whether the business is a favorite. Users can also view comments by clicking the “View Comments” button.

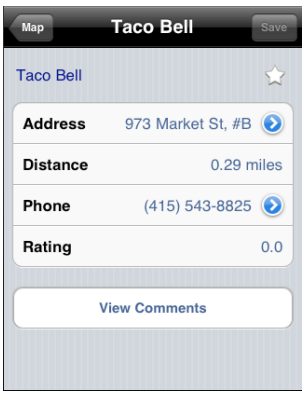

Figure 3.7. Business detail screen

3.4.1.3 Add Business to Favorite

Users can add the restaurant as a favorite by clicking the star icon as shown in Figure 3.7. When the star is yellow, the business is recorded as the user's favorite. Users can reach added favorite businesses in the Business tab (See Figure 3.8). The search starts with the selected business name and ALPO switches the view to the main view (See Figure 3.5 (a)(b)) after the user taps on the favorites.

Figure 3.8 Favorites View

3.4.1.4 Business Navigation

When the user presses the address row as shown in Figure 3.7, the view switches back to map view with routing details (See Figure 3.9).

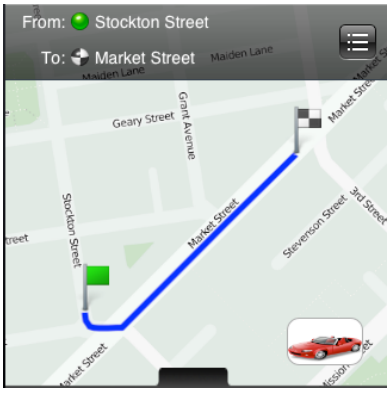

Figure 3.9 Navigation view

The gray information bar at the top indicates the starting and ending points. When the user clicks the button with a bulletin image at the top right side, the turn-by-turn list will be displayed with total travel distance and time (See Figure 3.10).

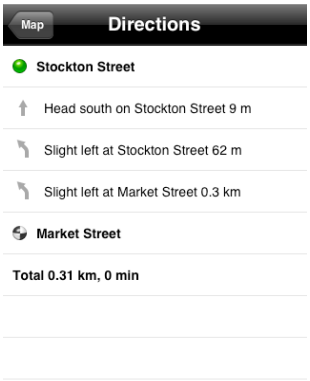

Figure 3.10 Turn-by-turn list

3.4.1.5 Call Business

When users need to make a call to the business to make a reservation, they can click on the phone number row as shown in Figure 3.7. The application will make a call to the business.

3.4.1.6 Navigate to Car

Users can store the current location of the car by clicking the “Park” button on the navigation bar (See Figure 3.5). They can hit the red Cabriolet car on the right-bottom corner, and directions to their car will show in the same way as search business does.

3.4.1.7 Comments and Replies

Sometimes the ratings stars are not reliable enough to determine whether a restaurant is worth trying. Users can view comments by clicking the “View Comments” button as displayed in Figure 3.7. This also applies when users need to see the restaurants’ most popular dishes. Users can also post a new comment or reply to existing comments (See Figure 3.11). Comments, name of poster, date posted, message, and numbers of replies are displayed in descending order by creation date. Replies are inserted below comments when clicking on the originating comment. Each reply includes the message and the name of message owner.

Figure 3.11. Comments and Replies view

3.4.2 Image Note

Image Note is the second component ALPO has implemented. Users can view the Image Notes list or add new Image Note.

3.4.2.1 Image Note List

Users can view the note list in the “Notes” tab as shown in Figure 3.12(a). For power consumption, the list does not automatically refresh. Users can manually update the list by dragging the list down. The loading bar displays on top when an update takes place. Releasing the list shows the loading in the progress bar as shown in Figure 3.12(b). The pull-to-refresh function is provided by EGOTableViewPullRefresh library, which is hosted on github.

Figure 3.12(a)(b). Note list

3.4.2.2 Add Image Note

Users can add Image Notes by either taking a picture from a mobile device's camera or picking an image from the photo library. Users can then enter corresponding information that is related to the image to be saved, such as title or comment. Users can also determine which Facebook friend group can see the note by selecting the "Share with", and when the user want to show/hide the image note. (See Figure 3.13 (a)(b)).

Figure 3.13. (a) Uploading image note (b) select Facebook group

3.4.3 Augmented Reality View

Users can view both Points of Interest and Image Notes in the Augmented Reality view by clicking the eye image button shown in Figure 3.5. The Camera View gets brought up and corresponding virtual objects overlay on the top of real world as shown in Figure 3.14. The virtual objects get displayed when the user direct the camera to the right direction. Users can get details of objects by double tapping on the floating objects. The segmented control bar at the right top is used to filter the results. The left control displays all searched Point of Interest and the right control displays all image notes.

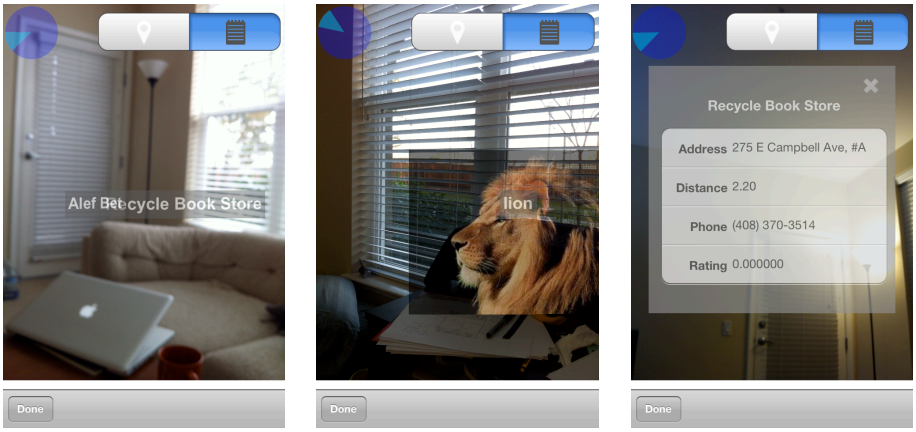

Figure 3.14 (a)(b)(c) Augmented Reality View

We have introduced the designs, user interfaces, and functions of ALPO. In the next section, we will introduce the details of implementations on the both client-side and the server side.

3.5 Implementation

In this section, we will first introduce implementations and technologies that are used to perform the functions of the client application and then introduce the work on the server-side in details.

3.5.1 Single Sign-On

To support Single Sign-On, ALPO asks the first-time user to log in with existing Facebook account as we discussed in previous sections. ALPO checks whether the FBTokenAccessKey and FBExpirationDateKey are existed. If both keys are existed, then ALPO checks whether the FBTokenAccessKey is still valid by comparing the current date and FBExpirationDateKey. If the session key is valid, then users can pass the Log-In screen. If the session key is expired or not exists, users will be asked to log in with Facebook account and authorize ALPO mobile application permissions such as personal information and users' friends list to access ALPO services. Once users successfully log in, the FBTokenAccessKey and FBExpirationDateKey will be generated and permanently stored in the application. The following flowchart (Figure 3.15) shows the Log-In flow.

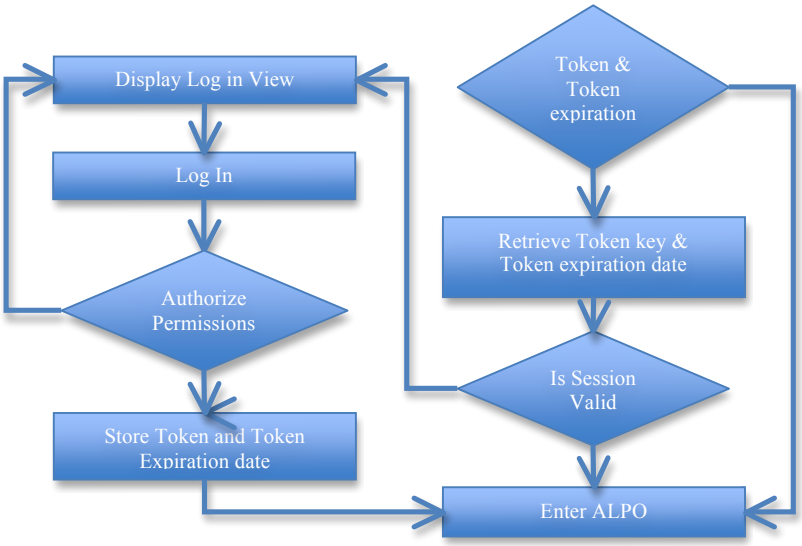

Figure 3.15 Single Sign-On flowchart

Users can sign out the Facebook account by switching to Settings tab. Then both FBTokenAccessKey and FBExpirationDateKey will be removed.

3.5.2 GPS Location

In order to obtain the current user's location or compass information, we imported a framework called Core Location. Using the Core Location framework, ALPO can retrieve data in three ways: Global Positioning System (GPS), Wi-Fi positioning Service (WPS), and cell tower triangulation [5]. The framework determines which technology will be used, but the level of accuracy used by the framework is manually set in ALPO. We can communicate with the Core Location framework via Location Manager. With the GPS information, ALPO can locate the user and provide associated business information such as directions of parking lots, restaurants, and other points of

interests.

3.5.3 Map View

The MapKit API in iOS is based on Google maps. Since this natively comes with Xcode SDK, it is well developed but also have some limitations. The MapKit API provides map view, coordination plotting, and pin/annotation placing. Since we are building an application that helps people get Points of Interest information, we should not only provide them details but also show them how to get to the business. Compared to MapKit, the CloudMade library allows customized map tiles and routing/driver directions, both of which are particularly useful in this project. The three variables that are needed to make a routing API call in CloudMade are current location coordination, destination coordination, and vehicle mode. The vehicle mode can be walking, bike, or car; in this project, we only support car mode in ALPO. The take-users-to-car function uses the same principle by storing the current locations permanently in device using NSUserDefaults class. The data will not be removed even if users leave the app or restart the device; this only get deleted when user remove the application.

3.5.4 Augmented Reality View

The camera view is the foundation of the Augmented Reality (AR) view. The camera view gets brought up, and the listing business info and Image Note will be overlaid on the top of camera view when users switch to augmented reality view mode. The capability of computing on mobile devices has been largely improved in past years

that allowed AR techniques to be ported on mobile devices. There are two ways to achieve AR technique on mobile devices; one is using object recognition (OR) technology, and the other one is using input sensor pose such as GPS and gyroscope. The OR technique has higher accuracy due to it is rendering the object according to the real world view on camera, but it only works if the target is already stored in the database. The second method is better than the previous method since most smart phones now incorporate GPS and a gyroscope sensor. It is also easier to build the data in the database using the second method since all we need are the latitude and longitude information of the target. ALPO adopts the second approach. By combining the info acquired from embedded camera, GPS sensor, and magnetometers, ALPO can
obtain users' location and device directions easily and accurately. There are several steps to make the virtual object layout accurately on the real-world view:

1. Attain device sensor pose
2. Calibrate target
3. Check whether the object is contained in camera view
4. Render virtual objects

We first capture the device latitude, longitude, and heading, and then calculate the distance, azimuth, and inclination of the device and
the target. iPhone has built-in equipment enabling the sensing of six-degree-of-freedom (6DOF), which means the movement of device of x, y, and z value in three-dimensional

coordination system plus orientation in pitch, yaw, and roll, of the camera lens(See Figure 3.10 (b)).

We first use the LocationManager API to retrieve the current location and camera heading information. Secondly, we calibrate each object using current location; we can get degrees of angle between two points with North Pole as pole and radial coordinate by calculating the difference of latitude and longitude between two points. Thirdly, we use the azimuth of the current location and the field of view of the camera to calculate the viewport and further decide objects to render. Lastly, ALPO scales the size of an object according to distance and render the object.

3.5.5 User Interface Render

Although the iPhone has very complete UI libraries, there are still some UI elements we need to customize such as the loading dialog, refreshing table header view, or the badge view for indicating replies of comments. It would be too much work if we needed to customize those UI elements from the existing iPhone SDK UI libraries. Fortunately, there are some third-party libraries that achieve what we required.
We adopted the TapKu Library, which is an open source framework for iOS. The library includes customized coverflow view, calendar view, line chart view, progress alert view, and image, label, and button in table cells. We utilized the progress alert view and badge view in ALPO. The progress alert view displays after the HTTP request is sent, and before data parsing is finished. The badge view displays in the Comments view that indicates the numbers of replies. ALPO displays refresh status and last-update timestamp.

3.5.6 Yahoo Local Search Web Services

After discussing the concept of how Augmented Reality view works and how AR is implemented through ALPO on iPhone, we are going to discuss the implementations of functions we mentioned in 3.5 section. When the user executes a search, the controller sends a request to the Yahoo API for the correspondingly searched businesses. The Yahoo local web services API allows us to customize queries and the response data format. For example, we can assign the number of results to be returned, radius distance, latitude and longitude, route, and category. We can also sort the results, omit category, format the output as xml, JSON, or php. Since JSON is data-oriented while xml is document-oriented, JSON is more suitable for data-interchange. JSON matches the data model of most object-oriented programming languages.

3.5.7 HTTP Request

To make HTTP request, we adopt ASIHttpRequest library in this project. ASIHttpRequest wraps the Objective-C framework CFNetwork API, a framework of Objective-C that helps perform network tasks such as sockets, SSL, FTP, and Apple Bonjour services that make the connection with web server easily and efficiently. Background-threading is supported in iOS4, which is the first version that supports multi-tasking, so the UI can be rendered while the network tasks are processing in the background. Since we are targeting an iOS that have higher version than iPhone4, we send the request asynchronously; while HTTP request and the response parsing is being

performed in a background thread, ALPO shows a spinning wheel with the word "Loading…" to present the application is processing the query. Once the parsing has done, the Controller adds pins and annotations on the map view, updates the list view, and removes the loading animation when finished.

3.5.8 JSON Parsing

We use JSONKit library to parse JSON format data. JSONKit library is customized for Objective-C on iOS platforms. The library decodes JSON format and maps to Objective-C foundation class. For example, null map to NSNull, boolean value and number to NSNumber, string to NSString, array to NSArray, and Object to NSDictionary. The object contains key value hash tables, associative arrays, and dictionaries, of which the dictionary is used the most in ALPO. Users can view the details of businesses by clicking items on the list view. The business data is stored in POIResult Data Model introduced in Figure 3.3 and passed to business detail View Controller.

3.5.9 Image Loader

While we are doing the query for retrieving an Image Note, the response data includes an image URL. We need to load those images in table view without blocking the main thread, and cache the images data for future rendering. JSImageLoader solves the loading time problem by returning DOM image objects from given URLs; with image dynamically loaded, the client side can preload and render additional

information at the mean while. It also speeds up the loading time by caching the image data in a local file for future use.

3.6 Server Side

After seeing the functional design of the application, let's look at the database design schema. We will also explain how we implemented the request processing and database query on the server side.

3.6.1 Member

Since ALPO uses Facebook accounts to identify users, we collect the Facebook ID for primary key in the member table and foreign key for other tables that we will discuss in later sections. We retrieved the Facebook ID and personal info using FQL [6], which is a SQL-style interface to query the Facebook data. After authentication, we retrieve personal info, collect user's information such as name and profile picture URL along with the ID in the member table.

3.6.2 Points of Interest

For the Points of Interest feature, we do not keep the data in the database since it is coming from Yahoo local search database. ALPO stored the IDs of businesses for the purpose of saving and loading comments.

3.6.3 Comments and Replies

Comment and Replies tables are for storing all messages from users. It has columns for message owner, creation time, message content, and comment id for replies.
To retrieve the comments and replies, we first need to get the business_id, which is the column name y_id in our tables. In this case, we will need information from three tables: the user name from member table, all comments belonging to this business, and all replies belong to selected comments. The query is as following:

```
SELECT c.y_id, c.c_id, c.comment, c.m_id,    m.name AS CommentOwner,
c.comment_created, r.r_id, r.comment AS reply, r.created AS replyCreated,
mr.name AS replyOwner FROM comment c LEFT JOIN replies r ON c.c_id =
r.comment_id LEFT JOIN member m ON c.m_id = m.uid LEFT JOIN member mr ON
r.replies_owner = mr.uid WHERE c.y_id="'.$yid."' ORDER BY c.comment_created
DESC
```

We first join the replies table to the comment table on Comment ID, and the Member table left joins the generated table by Member ID. Then we order the results descended by Comment's creation date.

3.6.4 Favorites

For the favorite function, to get better performance, we pre-load all favorites associated with the current user so that we can filter the needed information in a short

time. The favorite table only has two attributes: member_id and yahoo id. Since we can always do a query to Yahoo database to retrieve business information, we only store Yahoo ID.

3.6.5 Image Notes

The second main component in ALPO is the Image Note. There are several ways to store images in MySQL those are listed below:

- Using *blob* column type that images stores in binary mode
- Using longtext column type that images encode with base64 schema and stores as string
- Store the image locally in filesystem and store URL in image_url column

We used the third method after experimenting with all methods. The first two methods are not efficient in this project. Binary Large Object (BLOB) type is a field type for storing binary data and UIImage class is used to capture images stored on iPhone. To convert the UIImage to BLOB, we first need to compress the UIImage into JPEG or PNG file format using API, which can take a few seconds. We then cast the file to NSData class type, which can be posted via HTTP request method. After we get the image data on the server side, we format the binary data for insertion. When we need to retrieve the image data, we use the reverse steps to get images displayed on the client side. The upload process takes up to a minute but not vice versa. When the data

volume goes up, then the entire processing time to pull all image data grows painfully long too. For the second method, the steps for uploading/downloading are similar to the previous one we discussed except the image is encoded with base64 schema and then stored in the table as a longtext type. This takes longer for both uploading and downloading images due to data transmission in string type**.** The downloading transmission time statistics comparison is presented in the experiments section. After considering the timing factor, we decided to upload the image to server and then save the URL in the database to speed-up the data transmission. We have a folder on the server-side to keep all uploaded images, the client-side only post values and images with MIME type. The PHP code on the server side stores the image information along with stored image URL. The Note table stores all details corresponding to the image, beginning date, expiration date, shared group, and note URL.

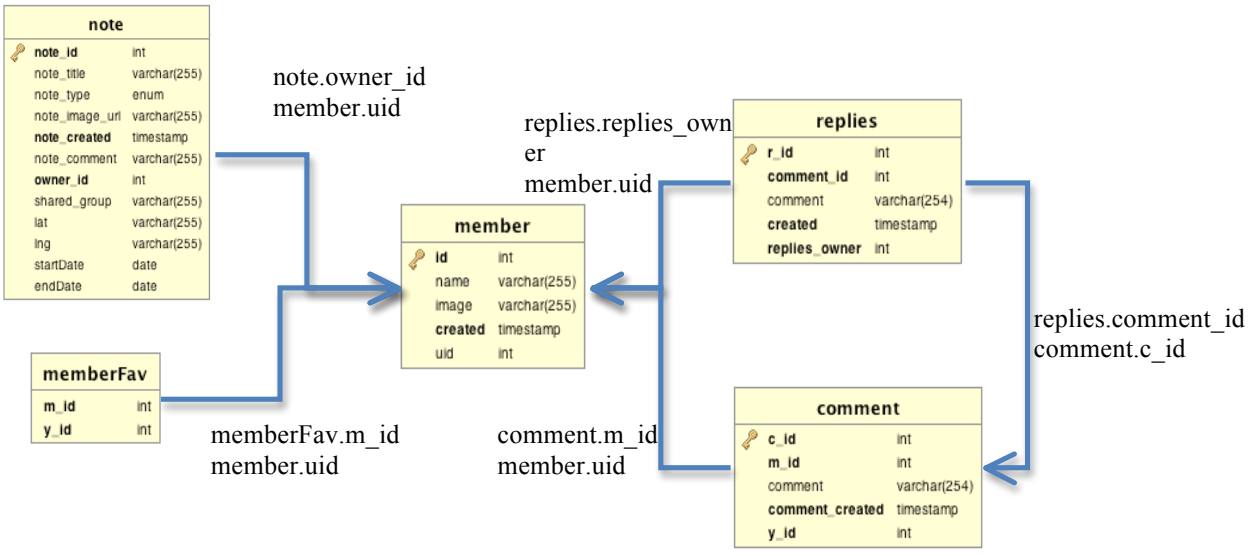

Figure 3.16. Database schema

3.6.6 Mobile Phone Server Application

The ALPO project is separated into three parts: database, server, and client. On the server side, we use PHP to act as a bridge between the database and mobile device (client). The client-side uses POST request to submit data to the server, the server then collect inputs in $_POST variables and execute corresponding queries. The response data from server side strips off all unnecessary information to lighten the load on clients, due to the hardware limitation on the client side. Multi-table queries are used to achieve this goal. After the query results are obtained, PHP encodes the result using json_encode function and returns it to the client side. There are seven PHP classes listed below, on the server side to handle the request:

- Uploader - takes the image data and corresponding information and store in Note table
- AddComment - takes user ID, message, and Business ID as inputs and insert record into Comment table
- AddReply - takes user ID, message, comment ID as inputs and insert record into Replies table
- AddFavorite – takes user ID and Yahoo ID as inputs and insert record into memberFav table
- GetFavorites – takes userID as input and returns favorites encoded results in JSON type

- GetNotes – takes user ID as the only input and returns notes encoded results in JSON type
- RegisterMember – takes user ID, name, and picture URL as inputs and insert ignore record into RegisterMember

4.0 Experiments

In this section, we will conduct experiments for libraries selections and Augmented Reality View computation.

4.1. Libraries Selection

We have mentioned earlier in design and implementation challenge section that there are many open-source and public third-party APIs for iOS development. Many of them have similar functions and targets with different approaches. Each of them may suit different situations, such as iOS version, device family, or data type.

4.1.1 Point of Interest Local Search Library

There are several public Points of Interest data libraries, such as Open Street Map (OSM), Yelp, and Yahoo. OSM is an open-source map whose most data comes from GPS devices, aerial photography, and user uploads. OSM has the richest database among three libraries, and it consists of global Points of Interest [7]. The disadvantage of OSM is the information is insufficient. Many business phone numbers ae missing, business categories are limited. Yelp and Yahoo local search web services allow users to search detailed complete business data in real-time by giving name or location information (this can be an address, zip code, or the latitude and longitude), but both databases are limited to United States businesses. Yelp has its own iOS SDK, which is easy to integrate, but it has strict guidelines to follow, such as request amount limits, user interface design-style, link requirement to the Yelp website, and users must log in to

leave comments [8]. ALPO targets users in the United States and fully controls the layouts design style and allow users to share comments without needing to create another Yelp account. ALPO adopted the Yahoo local search web services after evaluation.

For testing the Open Street Map library, we first needed to find an existing APIs for querying OSM data. For example, Xapi, MapScript, and CloudMade. CloudMade was the most complete API and had the best documentation among them. Further,
free form address and POI search features will be supported soon. The CloudMade website provides different data types created and extracted from OSM map data. We chose GPX files for testing. GPX, means the GPS Exchange Format. It is XML data format for exchange GPS data such as waypoints and routes between applications and web services. To store GPX data in out database, we first parsed the node element and then converted to SQL insert statements. There was two reasons that we ended up with doing it this way: The first one is the data only includes the name, category, latitude, and longitude of the business, which was not enough for our purpose. Secondly, the data itself was scattered and incomplete in that some states had extremely limited data in both quality and quantity.

The Yelp developer API is well-documented, and the input and output data types are adaptable. We can pass word, numbers, filters, bound box, and locations for input parameters. Each business also has precise detail information. There are two reasons why we decide not to Yelp API. First, Yelp requires branding and linking to Yelp that restricts the way user interface displayed. Secondly, Yelp requires users need to register to leave comments that against our goal for Single Sign-On.

4.1.2 JSON Parse Library

There are several options for JSON parsing frameworks, such as YAJL, json-framework, Touch JSON, and JSONKit. We only tested the json-framework and JSONKit for parsing time and import complexity. The json-framework is the oldest and follows the JSON protocol strictly. It also has the biggest (and so slowest to download) library. The json-frameowork, which is adopted by the Facebook iOS API, provides a parsing interface that allows one to specify the configuration parsing depth. This allows developers to parse large JSON structures without going too deep. In ALPO, the JSON structures are fairly straightforward –they only have two levels. We adopted JSONKit for ALPO. JSONKit is a lightweight library that has an easy to use API. To import the JSONKit library, developers only need to add JSONKit.h and JSONKit.m to do the project. The parsing speed is faster than json-framework in our case since the JSON structure is straightforward.

4.1.3 Image Recognition and Visual Search

At some point in the ALPO project, we were planning to have an image recognition feature. Using image recognition, a user can enter the details themselves or allow the image recognition tool to identify the image object and details for them. IQEngines provides free limited API for developers, and they offer an easy way for mobile developers to integrate visual search into mobile applications. There are two ways to identify objects in IQEngine, one is using computer vision and the other one uses human Crowdsourcing. When the engine receives a request, the Computer Vision module

identifies the key visual content in the scene. This takes a few seconds. Since this is based on existing data, there is the possibility that the image is not in the database. If the system cannot get an accurate result, the engine sends the image to human CrowdSourcing modules that using real people to identify images [9]. This may take minutes to complete. We experimented on several objects, such as flyers, mugs, furniture, books, and electronic products. It took one to two minutes to get the result back, and a few of them were not accurate. The reason for the lengthy processing time and inaccurate results was because IQEngine had just started their service and its dataset was small. After many experiments, we decided to make ALPO simpler, and users will need to enter the title and comments manually.

4.2 Image Storing Method

We have mentioned earlier in Section 3.6.5 that there are three ways to store images in the database. We have compared the download time for each method by recording time intervals between requests sent and responses received. Looking at Figure 4.1 and Table 1, it is evident that the transmission time for Blob and Longtext is considerably longer than the time of URL.

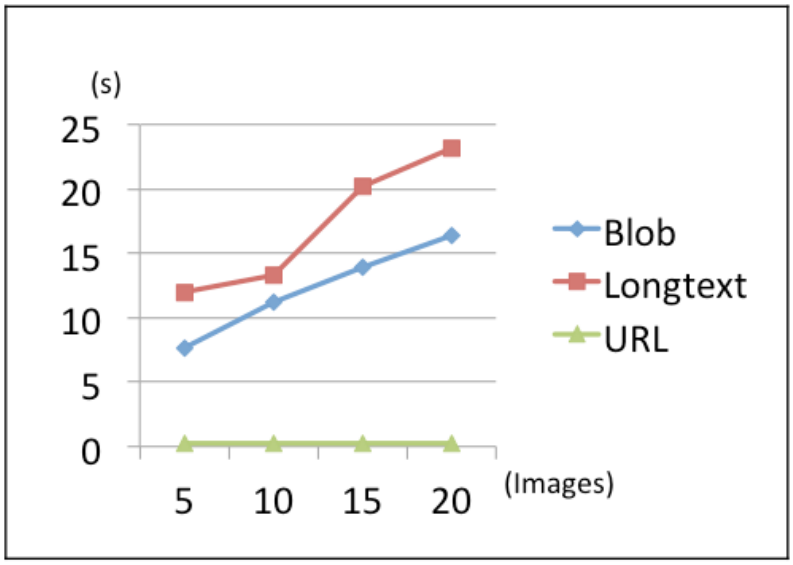

Figure 4.1 Image transmission time chart

	5	10	15	20(Images)
Blob	7.68	11.183	13.937	16.359(s)
LongText	12.014	13.338	20.233	23.169(s)
URL	0.253	0.266	0.216	0.272(s)

Table 1 Image transmission time comparisons

4.3 Testing

We will now describe the usability and functionality tests we conducted for our app. We designed several scenarios and asked three users to complete them without any help. The scenarios are listed in the first column of Table 3. We recorded their Facebook, Yelp, and Google map use habits (See Table 2).

	User A	User B	User C
Facebook	✗	✓	✓
Yelp	✓	✓	✗
Google Map	✓	✓	✗
Total	**2**	**3**	**1**

Table 2 Users' knowledge for mobile applications

We also recorded the number of attempts it took for the users to complete the task (See Table 3).

Tasks	User A		User B		User C		**Total**	
	Fail	Useful	Fail	Useful	Fail	Useful	**Failed**	**Useful**
Image Note List	0	✓	0	✓	0	✓	**0**	**3**
Note Details	0	✓	0	✓	0	Δ	**0**	**2**
Add Note	0	✓	0	✓	0	✓	**0**	**3**
Assign Note Privilege	1	✓	0	✓	0	Δ	**3**	**2**
Assign Existence time	0	✓	0	Δ	0	✓	**0**	**2**
Search Business	0	✓	0	✓	0	✓	**0**	**3**
View Business Detail	0	✓	0	✓	0	✓	**0**	**3**
Navigate to Business	1	✓	0	✓	0	✓	**6**	**3**
Turn-by-turn lists	2	✓	1	✓	1	✓	**3**	**3**
Call Business	0	✓	0	✓	0	✓	**1**	**3**
View Comments	0	✓	0	✓	0	✓	**0**	**3**
Leave Comments	0	✓	0	✓	0	✓	**0**	**3**
Reply Comments	2	Δ	1	✓	1	✓	**3**	**2**
Add/Delete Favorites	1	✓	0	✓	0	✓	**1**	**3**
Record Car Location	1	✓	1	✓	1	✓	**2**	**3**
Navigate to Car	1	✓	0	✓	2	✓	**3**	**3**
Shortcut Search	2	✓	1	✓	3	Δ	**6**	**2**
Augmented Reality View	0	✓	0	Δ	1	✓	**2**	**2**
Virtual Object details	2	✓	2	Δ	3	✓	**7**	**2**
Total	**13**	**15**	**6**	**16**	**12**	**13**		51/57

Table 3 Learning times and usability questionnaire

From the two tables, we can see that if the users were already familiar with the activity provided by applications from Table 2, then they did better on corresponding tasks in our application. Users with more experience on social networks and map tools seemed comfortable with our application. We combined the failure rate with the usefulness rating of each task to arrive at the final score.

In the “Useful” column, we add 1 point if the user thinks the function is useful (with ✓ symbol), 0 points if the function is optional (with ∆ symbol), and -1 point if the user thinks the function is unnecessary (with ✗ symbol). Overall satisfaction was 89% or 51/57 points, which proves that ALPO is a very useful application. From this testing we learned how users felt and what they expect from ALPO, so that we can make future improvements.

5.0 Conclusion

ALPO is an iPhone application that implements useful functionality from existing social networks to help people get information to help them with their daily life such as business search, navigation, comments and replies, favorites, image notes, and Augmented-Reality view. By combining Yelp's business list info feature and social network information from Facebook in our service, users can share daily life with friends and indirectly help friends with suggestions on business.

To create ALPO, we overcame the following challenges: hardware limitations, low bandwidth, threading, misuse library, and lower performance.

There are still some features that are not implemented presently. To make this application more attractive and functional, I will add the following features in the future:

People rely on post-it notes to remind them of things like buying diapers, and depositing checks. This same reminder can be integrated into the business search function. By doing a search, the items in the reminder list that relate to the search item will show in an alert. For example, if a user puts "buy milk" in their reminder list, when they search for stores like "Walmart", an alert will pop up reminding them to "buy mill".

For the Map View part, the ratings are not visible. For future enhancements, we can scale the pins according to the ratings or use different colors to represent the reputation of the business. We should incorporate the walking and biking mode to fit different needs, since it supports only car mode. The CloudMade API includes geo-based advertisement services that can display a coupon banner on the bottom of Map View. For the directions,

we can add multi-routing functions so that users can add multiple destinations without changing the destination over and over. The last function we want to add to ALPO is just for fun. The icon for taking users to the car button should be customizable so that users can upload a photo of their own car.

References

[1] *iphones*. (2011, 11 10). Retrieved from http://en.wikipedia.org/wiki/IPhone

[2] *Apple reveals most downloaded iphone, ipad apps of all time*. (2011, 01 09). Retrieved from http://www.networkworld.com/news/2011/011911-apple-appstore-iphone-ipad-downloads.html

[3] Wales, J. (n.d.). *http://en.wikipedia.org/wiki/facebook*. Retrieved from http://en.wikipedia.org/wiki/Facebook

[4] *Geocoding and geosearch*. (n.d.). Retrieved from http://developers.cloudmade.com/projects/show/geocoding-http-api

[5] *ios 5: Understanding location services*. (2011, 10 25). Retrieved from http://support.apple.com/kb/HT4995

[6] *Facebook query language (fql)*. (n.d.). Retrieved from http://developers.facebook.com/docs/reference/fql/

[7] *Openstreetmap*. (n.d.). Retrieved from http://www.openstreetmap.org/

[8] *Api 1.0: Overview*. (n.d.). Retrieved from http://www.yelp.com/developers/documentation

[9] *Iq engines: Image recognition platform*. (n.d.). Retrieved from http://www.iqengines.com/

The following papers provide thoughtful insights for augmented reality:

Ebling, Maria R.; Cáceres, Ramón; , "Gaming and Augmented Reality Come to Location-Based Services," *Pervasive Computing, IEEE* , vol.9, no.1, pp.5-6, Jan.-March 2010 doi: 10.1109/MPRV.2010.5
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5370800&isnumber=5370796

Jongbae Kim; Heesung Jun; , "Vision-based location positioning using augmented reality for indoor navigation," *Consumer Electronics, IEEE Transactions on* , vol.54, no.3, pp.954-962, August 2008
doi: 10.1109/TCE.2008.4637573
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4637573&isnumber=4637567

Chang, W.; Qing Tan; , "Augmented Reality System Design and Scenario Study for Location-Based Adaptive Mobile Learning," *Computational Science and Engineering (CSE), 2010 IEEE 13th International Conference on* , vol., no., pp.20-27, 11-13 Dec. 2010
doi: 10.1109/CSE.2010.66
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5692452&isnumber=5692448

Paucher, R.; Turk, M.; , "Location-based augmented reality on mobile phones," *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on* , vol., no., pp.9-16, 13-18 June 2010
doi: 10.1109/CVPRW.2010.5543249
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5543249&isnumber=5543135

Chang, W.; Qing Tan; Fang Wei Tao; , "Multi-Object Oriented Augmented Reality for Location-Based Adaptive Mobile Learning," *Advanced Learning Technologies (ICALT), 2010 IEEE 10th International Conference on* , vol., no., pp.450-451, 5-7 July 2010
doi: 10.1109/ICALT.2010.130
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5573236&isnumber=5571093