LIFE MOBILE APPLICATION

APPLICATION PROGRAM INTERFACE: Research

Colin Man, Malina Jiang

VERSION 1.0

Last Update: 2 February 2015

TABLE OF CONTENTS

1 Document Control

1.1 Change Record

Version	Date	Author	Changes
1.0	02-02-15	Colin Man	Draft following overview + description
1.0	02-02-15	Colin Man	Draft Google Maps API
1.0	02-02-15	Malina Jiang	Draft Yelp and Google Drive API

1.2 Definitions

Term	Definition
LMA	Life Mobile Application
LBS	Location Based Services
API	Application Program Interface

Note: "LMA" is consistently used throughout all documents as the name of the mobile application, although it is understood that another name will be chosen for branding purposes.

2 Introduction

2.1 Scope and Purpose

This document contains a description of third party API's that may potentially be incorporated into LMA, in terms of contribution to the business model as well as technical functionality and constraints.

The API's outlined allow interaction with a variety of third-party products that do not define LMA individually, but come together to form a unique platform that encompasses the totality of the mobile user's needs.

The existence of this document is imperative to the success of the product as it clearly defines the constraints of each interface and the extent to which we can leverage the product to create a novel platform. It is crucial to the development of a business plan with a clearly defined direction as well as a viable and efficient implementation. Without such definitions, it is easy to design a product that is impossible to develop on the technical side or assumes third-party functionality that does not exist.

A description of the API's will also help in defining and outlining a minimal viable product, a subset of LMA's features that form the core functionality and can stand alone if necessary.

As such, each third-party API description contains the following components:

- Overview – Summary of the features and purpose. This includes a description of the functionality encompassed in the API as well as its contribution to the system.

- Structure and API Calls – A technical description of the API and the structure of its interface. Since the way information is obtained varies widely depending on the design of each API, this is vital both in extracting constraints for the business model as well as development of the actual application in the implementation stage.

- Technical Constraints – Detailed summary of constraints in terms of information that can be obtained through the application user interface. Includes information such as maximum volume of requests that is critical to making important design decisions.

3 API: Google Maps

3.1 Summary

The Google Maps API is divided into four main types of interfaces, depending on the technology used to implement the product as well as to access the data.

The four API interfaces are:

- Embed API – The simplest of the four, which simply uses HTML and iframes to embed content into any website.
- Web API – Provides a set of functionality that can be accessed using standard RESTful calls. These calls can be placed from web applications (AJAX) or native applications.
- JavaScript API – Provides extended maps functionality in visualization and map content. Can be integrated into platforms that run mainly JavaScript.
- iOS API – Native integration of functionality that interfaces with the operating system directly. Can only be used by iPhone applications, but has similar functionality to the JavaScript API.

3.2 Embed API

3.2.1 Overview

The Embed API is a simple way to embed maps into an application. It allows developers to embed iframe elements into the web application and specify various features using parameters in the query portion of the src URL. Thus, information is obtained and displayed to the user through purely HTTP requests and no interaction with the API in between the requests is needed. Technical details are outlined in "Structure and API Calls".

The API allows for four different ways to display maps content:

- Place Mode – displays a map pin at a predefined location (landmark, business, geographic feature, or custom pin)
- Directions Mode – displays the path between two different points and shows the time and distance estimated.
- Search Mode – displays a map that contains the results to a search query with a term specified in the attributes of the iframe tag.
- View Mode – Provides a barebones map that contains no visual content apart from the basic map functionality. This can be either a map view or a street view.

Note that the Embed API creates a map that links to Google Maps and provides an interface to work with options that may not be in the code injected through the iframe.

If the user is logged in to their Google account, they will also be able to save the locations from the embedded map into their user account, with custom attribution information that can b specified by the application.

3.2.2 Structure and API Calls

The API call is performed through an HTTP request sent via the "src" attribute of an iframe. The call is structured as follows:

https://www.google.com/maps/embed/v1/MODE?key=API_KEY&parameters

The mode of operation (MODE) is one of place, directions, search, view, or streetview

In order to use the Embed API to obtain maps, the application must first register with Google to obtain an API key (API_KEY) that is included in the url query used to embed the map into the page. This key can be restricted to certain domains to ensure that others do not steal it.

Since the HTML to retrieve the maps is retrieved using a simple HTTP call to the web service, all parameters that have to do with obtaining data must be sent through the query string. Other properties such as width, height, border, etc. can be customized in the iframe and slightly change the rendering of the map.

The possible query parameters to the Embed API (parameters) are as follows:

- attribution_source – string that describes the site or app; allows users to see what source is attributed to saving the place on their maps
- attribution_web_url – the url of the site or app (as attribution information)

- attribution_ios_deep_link_id – a URL scheme that links to the iOS application (using iOS internal links)
- center – the center of the map view (latitude and longitude values)
- zoom – (zoom level from 0 to 21)
- maptype – either "roadmap" or "satellite" depending on the view that should be loaded
- language – the language that should be used in the embedded map
- region – the borders and labels that should be displayed

The mode-specific query parameters are as follows:

- Place
    - q – the place to show on the map. This value can either be a location (escaped string) or an address (does not take latitude and longitude since it must be a name location to be highlighted).
- Direction
    - origin – the starting point location (as before, in either escaped address format or place name)
    - destination – the end point location (as before, in either escaped address format or place name)
    - waypoints – a set of intermediary places (as before, in either escaped address format or place name) separated by the pipe character
    - mode – method of travel (choose from driving, walking, bicycling, transit, or flying). Options will be shown on map if none is chosen.
    - avoid – any obstacles that the map should avoid (tolls, ferries, highways). Different items can be separated using the pipe character.
    - units – either metric or imperial depending on the units that should be displayed
- Search
    - q – the search term that should be shown on the map (in escaped URL format – can include restrictions such as "in Danville near 94526")
- Street View
    - pano – the panorama id that should be shown (if not specified, one will be found from the location)
    - heading – the direction that the camera is facing in degrees from due north
    - pitch – the angle of the camera up or down in degrees
    - fov – the horizontal field of view in degrees

3.2.3 Technical Constraints

Using the map modes in the Embed API is not restrictive in terms of the user interaction – the user can move the map around in the iframe and they can switch view modes (Satellite, Maps, etc.). The constraints are mainly on the ease of incorporating dynamic content and on showing or controlling the maps interaction.

It is not possible to dynamically load content into the map since the map is hosted on Google's own servers and we are merely using an iframe to load it. Any attempt to change the "src" of the iframe in order to load the new parameters will cause the entire map to reload, which does not give a good user experience (we want the search to be done within the map interface, not by refreshing and reloading the entire map, which also has a higher latency).

This interface can only be used in a website since there are many limitations with iframes on mobile devices (Safari has compatibility issues, as does the android browser). The iframe served by Google Embed API is also not optimized for mobile, so may not be as easy to use on a phone.

There are no constraints on the volume of API calls that can be made, so the use of this API is essentially unlimited.

3.2.4 Summary

When compatibility and dynamic flexibility is not an issue, using the Google Embed API is the best solution to provide map functionality. It is the easiest to use out of the four API's and provides a very fast integration of maps functionality into a website.

In terms of LMA, the Embed API can be used for various maps services in which the location of a place should remain static for the duration of the webpage load. Examples of this may include: loading directions to a certain place, keeping track of saved routes and places, etc. For additional functionality of Google Maps, we may have to resort to the Web or native API's. See below for a full description.

3.3 Web API

3.3.1 Overview

3.3.2 Structure and API Calls

3.3.3 Technical Constraints

3.3.4 Summary

3.4 JavaScript API

3.4.1 Overview

3.4.2 Structure and API Calls

3.4.3 Technical Constraints

3.4.4 Summary

3.5 iOS API

3.5.1 Overview

3.5.2 Structure and API Calls

3.5.3 Technical Constraints

3.5.4 Summary

4 API: YELP

4.1 OVERVIEW

Yelp offers its users convenience and accessibility to local businesses wherever they go. In addition to business reviews, Yelp also finds events and lists for the user and allows communication between Yelpers. Yelp offers an unbiased look at the businesses within the surrounding area and offers its feedback to users so that they can make the best decisions about which businesses to patronize. The Yelp API offers the following functionality:

- Find the top results in a geographical location in response to a client query.
- Sort results according to the client query (e.g. by highest to lowest ratings, or greatest to least distance from the client location).
- Limit displayed results according to client specifications (e.g. display only businesses that offer Yelp Deals).
- Display detailed information about a Yelp Deal (e.g. savings, purchase URL).
- Identify whether a claimed has already been claimed on Yelp.com

The Yelp API as integrated with the Location-Based Services Module lends its wide database of businesses and various API calls to the task of locating the best deals in the area for clients of the Life Mobile Application. The specific applications of the API calls Yelp offers will be detailed in the section below.

The current version of the Yelp API is 2.0. Yelp v1.0 is deprecated but will but Yelp does not have any plans to turn it off as of now. Yelp currently limits API Calls to 25,000 calls, but accommodates requests for more calls. The API uses the standard, secure authorization protocol OAuth 1.0a, xAuth and offers two main sub-API's – the Search API and the Business API. As LMA focuses mainly on consumer users instead of business users, use of the Yelp API will mainly be constrained to the Business API.

4.2 STRUCTURE AND API CALLS

The Yelp API is limited to one type of API call, namely the GET request. Yelp authenticates each request using OAuth per the usual standard.

GET – Allows the client to search for local businesses. The API request takes in the following optional parameters:

- Term – Search term inputted by the client. If not included, the API will search everything.
- Limit – Number of search results to return.
- Offset – Offset the list of return businesses by this amount.
- Sort – Determines what the data will be sorted by. Options include 0 (for best matched), 1 (by distance), or 2 (most highly rated). When returning results, the business rating is adjusted for the number of ratings to give a more comprehensive view of the business quality.
- Category_filter – Applies filter to search results.
- Radius_filter – Determines the search radius of the query, within 25 miles.
- Deals_filter – Decides whether to search only for businesses with Yelp Deals.

As Yelp is also a location-based service, Yelp API requests have one required parameter:

- Location – Specifies the location to be used when conducting a search. Locations can specified with an address, neighborhood, city, state, zip, or county. They can additionally be specified with geographical coordinates:

    - Cll Parameters – These parameters are formatted as "cll = latitude, longitude".
    - Bound Parameters – These parameters form a geographical bounding box and are formatted as "bounds = sw_latitude, sw_longitude | ne_latitude, ne_longitude".
    - Ll Parameters –These parameters are formatted as "ll = latitude, longitude, accuracy, altitude, altitude_accuracy".

In response to the API request, the client receives the following:

- Region – Suggested bounds for map display.
- Total – Number of businesses in search result.
- Businesses – List of business entries that fulfill the search parameters. Each business entry has the following attributes.

    - Id – Yelp Business ID.

- Is_claimed – Whether the business has already been claimed.
- Is_closed – Whether the business has been permanently closed.
- Name – Business name.
- Image_url – Business photo URL.
- Url – Business Yelp page URL.
- Mobile_url – Mobile business Yelp page URL.
- Phone – Business phone number.
- Display_phone – Phone number formatted for display.
- Review_count – Number of reviews on the business.
- Categories – List of category names the business is associated with.
- Distance – Distance from search location in meters.
- Rating – Business rating. Includes rating image associated with the business.
- Snippet_text – Snippet text associated with the business.
- Snippet_image_url – URL of snippet image associated with the business.
- Location – Location data of the business. Includes address, city, state, zip, country, neighborhood and other location information of interest.
- Deals – Deals offered by the business. Includes deal ID, name, URL, image URL, start and end times, popularity, restrictions, and additional details.
- Gift_certificates – Gift certificate information for the business if the business offers them. Includes gift certificate ID, URL, image URL, balance, price, and other information.
- Menu_provider – Provider of business menu.
- Menu_date_updated – Timestamp of last update.

4.3 Technical Constraints

The main constraint on the Yelp API is the limit of 25,000 API calls. However, this constraint is relatively flexible, as the Yelp API describes the process for appealing for a greater API call allotment. An increase in the volume of calls will not incur additional costs, making for greater flexibility in the LMA application design.

Additionally, since all requests must be authenticated with an OAuth token, LMA will need a centralized account with the Yelp API. The account will allow LMA to manage access to the Yelp API and is used by the API to keep track of the number of API calls.

In terms of constraints for the API call (GET) offered by the Yelp API, most of the search parameters that the API allows are optional, and are only used to pare down the search results. The one required parameter is the location of the client and/or the location that the client wishes to research. The location is passed into the search call either in the usual form as an address, city, state, and zip, or as a set of geographical coordinates. In either case, the location associated with the search query must be determined, whether automatically or manually inputted by the client.

Finally, in terms of mobile platforms, Yelp's iPhone application (yelp for iPhones >= 2.0.0 and yelp4 for iPhones >= 4.0.0) enables search, view, and check-in functions. Yelp does not have the same support for Android phones or other smart phones, which may influence the direction of the app in the future.

4.4 Summary

The Yelp API will be integrated into the LBS Module of LMA. LMA offers the user as its features the convenience of pulling up information about local businesses in order to make an informative consumer decisions. The Yelp API enables this LMA function by allowing users to search for local businesses or services and viewing the associated information that Yelp provides, such as the location, rating, and any deals that the business offers.

In addition to allowing users to proactively search for services to meet their needs, the Yelp API enables LMA to recommend services to the user based on LMA's profile of the user. In this way, LMA extends the functionality of the Yelp API far beyond its current offerings. Since LMA also has information on the user's wish lists or to-do lists, LMA can make better recommendations for the user by matching the user's needs to local businesses nearby and calculating the optimal distribution of the user's patronage.

There are a few limitations to the Yelp API, namely the constrained volume of API calls and the need to collect information about the user's location. However, as described earlier, if the volume of API calls exceeds the initial quota, it is possible to appeal for a greater API call allowance. Additionally, since the LBS Module will need the user's location for many more applications, it will be relatively easy to gain access to such information.

5 API: Google Drive

5.1 Overview

Google Drive is a file storage and synchronization service based in the cloud that allows users to store, share, and edit files. Google Drive makes collaboration between users convenient by allowing modification of the same file by different users at the same time possible. It also makes files easily accessible no matter the location of the user. Among its many offerings, the Google Drive API offers the following functionality:

- Create and open files in the UI.
- Search for files.
- Distribute and market web applications.
- Share and collaborate on files by modifying the permissions of files.
- Export and convert Google docs.

The Google Drive API as it is integrated with LMA will provide users with cloud-based storage for their documents, pictures, and other files. It will also allow users to organize their files and bring them wherever they go.

The current version of the Google Drive API is Google Drive API v2. Google Drive API v2 is mostly compatible with v1 and Document List API v3, except for two changes: the parameter id from files.update and files.get renamed to fileId, and the parentsCollection field in files listing parents renamed to parents. However, Google Drive v2 supports migrating from these older versions. The API uses OAuth 2.0 protocol to authenticate a Google account and authorize access to user data.

5.2 Structure and API Calls

The Google Drive API is composed of many resource types, which subsequently are composed of various API calls.

5.2.1 Files

- **Get** – "Get /files/fileId"; gets a file's metadata by ID.
- **Insert** – "POST /files"; inserts a new file.
- **Patch** – "PATCH /files/fileId"; updates file metadata.
- **Update** – "Put /files/fileId"; updates file metadata and/or content.
- **Copy –** "POST /files/fileId/copy"; creates a copy of the specified file.
- **Delete** – "DELETE /files/fileId"; permanently deletes a file, skipping the trash.

- List – "GET /files"; lists user files.
- Touch – "POST /files/fileId/touch"; set's file's updated time to the current server time.
- Trash – "POST /files/fileId/trash"; moves file to trash.
- Untrash – "POST /files/fileId/untrash"; restores file from the trash.
- Watch – "POST /files/fileId/watch"; watches file for any changes.
- EmptyTrash – "DELETE /files/trash"; permanently clears user's trashed files.

5.2.2 About

- Get – "GET /about"; gets information about the current user and Drive settings.

5.2.3 Changes

- Get – "GET /changes/changeId"; gets specific change.
- List – "GET /changes"; lists user changes.
- Watch – "POST /changes/watch"; watches changes to user's Drive.

5.2.4 Children

- Delete – "DELETE /files/folderId/children/childId"; removes child from folder.
- Get – "GET /files/folderId/children/childId"; gets specific child reference.
- Insert – "POST /files/folderId/children"; inserts file into folder.
- List – "GET /files/folderId/children"; lists a folder's children.

5.2.5 Parents

- Delete – "DELETE /files/folderId/parents/parentId"; removes parent from folder.
- Get – "GET /files/folderId/parents/parentId"; gets specific parent reference.
- Insert – "POST /files/folderId/parents"; adds parents folder for a file.
- List – "GET /files/folderId/parents"; lists a folder's parents.

5.2.6 Permissions

- Delete – "DELETE /files/fileId/permissions/permissionId"; deletes a permission from the file.
- Get – "GET /files/fileId/permissions/permissionId"; gets a permission by file ID.
- Insert – "POST /files/fileId/permissions"; inserts a permission for the specified file.
- List – "GET /files/fileId/permissions"; lists a file's permissions.

5.3 Technical Constraints

5.4 Summary