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 3

1.1 Change Record 3

1.2 Definitions 3

2 Introduction 4

2.1 Scope 4

2.2 Functional Overview 4

2.2.1 Discovery 4

2.2.2 Maps 4

2.2.3 Lists 4

2.2.4 Safety 5

2.2.5 Scheduler 5

2.2.6 Entertainment 5

2.2.7 Tools 5

2.3 Assumptions/Constraints 5

2.4 Requirements 6

3 Component: Discovery 7

3.1 Navigation 7

3.2 Overview 7

3.3 Description 8

3.4 Feature: Goods and Services 9

3.5 Feature: Places 13

3.6 Feature: Travel 17

3.7 Feature: Food 19

3.8 Combination Features 22

3.9 Use Cases 23

Figure 3.9.1 – John 23

Figure 3.9.2 23

# Document Control

## Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Changes |
| 1.0 | 01-25-15 | Colin Man | Draft following overview + description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Definitions

|  |  |
| --- | --- |
| Term | Definition |
| LMA | Life Mobile Application |
| LBS | Location Based Services |
| POI | Points of Interest |
|  |  |
|  |  |

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.

# Introduction

## Scope

## Functional Overview­

## Assumptions/Constraints

* This assumes that LMA is run on a system that has a stable Internet connection.
* This assumes that LMA is run on a system that has a stable GPS connection and location data.
* This assumes that users have granted the necessary permissions required to provide services such as:
  + Push notifications
  + Location Based Services
  + Bluetooth LE connections

## Requirements

# API: Yelp

## 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.

## 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.

## 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.

## 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.

# API: Google Drive

## 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.

## Structure and API Calls

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

### 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.

### About

* **Get** – “GET /about”; gets information about the current user and Drive settings.

### Changes

* **Get** – “GET /changes/changeId”; gets specific change.
* **List** – “GET /changes”; lists user changes.
* **Watch** – “POST /changes/watch”; watches changes to user’s Drive.

### 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.

### 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.

### 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.

## Technical Constraints

## Summary