DESIGN AND DEVELOPMENT OF AN INTERACTIVE EVENTS CALENDAR APP PROVIDING EVENT REMINDERS BASED ON USER LOCATION

A graduate project submitted in partial fulfillment of the requirements

For the degree of Master of Science

in Computer Science

By

Deepak Manglani

May 2012
The graduate project of Deepak Manglani is approved:

_______________________________
Professor Taehyung Wang, Ph.D.

_______________________________
Professor Richard Covington, Ph.D.

_______________________________
Professor Robert McIlhenny, Ph.D., Chair

California State University, Northridge
TABLE OF CONTENTS

SIGNATURE PAGE Error! Bookmark not defined.
LIST OF FIGURES .......................................................... iii
ABSTRACT ......................................................................... v
1. Introduction ...................................................................... 6
   1.1 Location Based Services .................................................. 7
   1.2 Calendar Quest (CalQuest) Background ............................ 8
   1.3 Similar Work .................................................................. 9
2. Technical Details .............................................................. 10
   2.1 Android OS .................................................................... 10
   2.2 The Development Environment ......................................... 12
   2.3 Web Services .................................................................. 15
3. Calendar Quest Mobile App ................................................ 18
   3.1 Requirement Analysis ..................................................... 18
   3.2 Planning ......................................................................... 19
   3.3 Implementation .............................................................. 23
   3.4 Testing ............................................................................ 35
4. Review Summary ............................................................... 36
   4.1 Problems Encountered .................................................... 36
   4.2 Degree of Success ......................................................... 37
   4.3 Future work and Improvements ....................................... 38
Bibliography ......................................................................... 39
Appendix ............................................................................. 42
LIST OF FIGURES

Figure 2.1-01: Android Architecture ............................................................................. 10
Figure 2.2-02: Eclipse IDE .......................................................................................... 12
Figure 2.2-03: Android Emulator .................................................................................. 13
Figure 2.2-04: DDMS Eclipse Screenshot ..................................................................... 14
Figure 2.3-01: Google Places Query............................................................................. 16
Figure 2.3-02: Google Places JSON Response ............................................................. 17
Figure 3.2-01: Android Calendar Provider Screenshot .................................................... 20
Figure 3.2-02: SQL-Lite Query ...................................................................................... 21
Figure 3.2-03: SQL-Lite Query Implementation Example ............................................. 22
Figure 3.3-01: Use Case 1 ............................................................................................ 24
Figure 3.3-02: Use Case 2 ............................................................................................ 24
Figure 3.3-03: Use Case 3 ............................................................................................ 25
Figure 3.3-04: Start Screen .......................................................................................... 26
Figure 3.3-05: Main View ............................................................................................. 27
Figure 3.3-06: View event view ....................................................................................... 28
Figure 3.3-07: Edit/Delete Event View .......................................................................... 29
Figure 3.3-08: Activity Diagram .................................................................................... 31
Figure 3.3-09: Sequence Diagram 1 ............................................................................. 33
Figure 3.3-10: Sequence Diagram 2 ............................................................................ 33
Figure 3.3-11: Sequence Diagram 3 ............................................................................ 34
Figure 4.3-01: Android Google Calendar Monthly View ................................................ 38
ABSTRACT

DESIGN AND DEVELOPMENT OF AN INTERACTIVE EVENTS CALENDAR
PROVIDING EVENT REMINDERS BASED ON USER LOCATION

By
Deepak Manglani
Master of Science in Computer Science

Mobile devices today along with voice communication, provide us with a whole host of services such as instant messaging, video streaming, web surfing, social services (Twitter, Facebook) etc, where ever we go, through the use of mobile applications more commonly known as apps.

In the mobile application world, the hottest apps today are the ones that target the users wireless location which are called location based services apps.

A location based service is a software application made for an IP-capable mobile device that requires knowledge of where the device is currently located. They are generally query-based services providing users with answers to queries such as where is the nearest ATM[2].

The focus of this project is to create a location based services app(called CalQuest) using Google's powerful open source Android platform along with Google services such as Google Calendar and Google Maps to create a calendar application that reminds users about events based on their event location.
1. Introduction

This project presents the design and development of an Android application called "CalQuest". This project entails two parts.

Part 1: Design

a) Creating and designing a user interface that gives the user the ability to interact with the calendar.

b) Synchronizing and retrieving events from Google Calendar

c) The basic ability to view, modify and add events in the Google Calendar from the app itself.

d) Providing the ability for the user to add the event in location reminders.

Part 2: Refinement

a) Filtering out expired and deleted events.

b) Determining the users current location that is getting his latitude and longitude coordinates.

c) Retrieving a list of event locations from the events which were added to the location reminders, and checking them against local businesses.

d) Checking to see if a user falls within the vicinity of a particular calendar event location.

e) Sending an alert to the user advising them that they are within the vicinity of a particular event.
The chapters are as follows:

Chapter 1 gives the general overview of the Location based services and various tasks that can be achieved using them, along with a justification of why this application was needed.

Chapter 2 describes the various technologies surveyed and used in the making of this application.

Chapter 3 explores the software development techniques that went into making this application.

Chapter 4 summarizes accomplishments and future Improvements to the application

1.1 Location Based Services

A location based service is a service that needs to know the location of the user asking for the service before it can provide them the service.

It provides a user with contents customized by the users current location.

Typical examples of location-based services are real time turn by-turn directions, the location of the nearest ATM or gas station or particular restaurant, or a social networking service [3].

A service needing to know the user's location defines it as a location based service. It typically consists of five main components [3]. As follows

1) Service providers software application.

2) A mobile network or any network(such as wifi) to transmit data and request for the service (most mobile networks in the US already have this requirement to meet FCC regulations for E-911 Phase II) [11].

3) A content provider to supply the end user with geo-specific information.
4) A GPS enabled device.

5) A software based solution for a device that supports the specific API.

Finally they require the user's permission to track his whereabouts. In most cases this is achieved by installing the application provided by the location-based service.

Although location-based services have been around since 2000, they have mostly been used in commerce with a subscription-based business model. The release of Apple's 3G iPhone and Google's LBS-enabled Android operating system, however, has allowed developers to introduce millions of consumers to LBS. According to the 2008 fourth-quarter report from Nielsen Mobile, a division of The Nielsen Company, location-based services account for 58 percent of the total downloaded application revenue for mobile phones in North America [3].

1.2 CalQuest Background:

Historically calendars in applications like Google Calendar and Microsoft Outlook have provided event reminders to users according to time.

The application designed and developed by this project seeks to enhance event reminder ability by the addition of reminding users about their events based on the event location specified in the events.

A typical daily life example would be a person has put in his events calendar that he needs to buy milk from a shopping complex say "Vons" tomorrow or even the day after tomorrow, but today while eating lunch he goes to a restaurant that has an adjacent Vons next to it. If he has added this event in his location reminders list, the app will continuously scan his surroundings looking for Vons, and will alert him to the presence of a closest Vons nearby (that falls within the user's defined proximity radius) suggesting
him to carry out his event activity right now or allowing him to postpone it for a later
time when he is next to Vons.

This app seeks to optimize one’s schedule, in, that while the user is having lunch, he
receives a reminder about picking up milk at Vons, thus also enhancing productivity for
the user by giving him the ability to finish a task or an activity he has in his calendar
ahead of time. This has other benefits such as saving the user time of going to Vons the
next day and also saves him some money that he would have to spend on gas to go there.
This app automatically synchronizes with the user's Google Calendar thus saving him the
trouble of entering details such as event title, event description, start time, end time etc
about the event again, although he has to go to the apps user interface to add the event to
location reminders by the push of a button.

It will allow the users to view, edit and add events to their Google Calendar from the
application interface itself.

It also allows the user to leave the app as the major work of reminders is done in the
background, thus allowing the user to access other apps on his phone without any issues.

1.3 Similar Work

There are no other known apps available which have the exact same design specs,
although one app (on the Android App store) uses a similar idea and alerts user when he
enters a certain proximity. It is limited in the sense that the user has to tell the app the
exact address. However, our app works with both exact address and business names.
Android Market App is also limited in its ability as it does not synchronize with Google
Calendar[12].
2. Technical Details

2.1 Android OS

Android is a Linux-based operating system for mobile devices such as smart phones. It is developed by the open Handset Alliance led by Google. Google released the Android code as an open source project under the Apache license [9].

Android SDK provides the tools and APIs necessary to start developing applications on the Android platform using the Java programming language.

Android Architecture

![Android Architecture Diagram](image)

**Figure 2.1-01: Android Architecture[8]**

Architectural layers of the Android OS

The above diagram describes the Android operating system architecture.
**Applications layer** it is where all applications reside. It is also where our application CalQuest will reside. It is the topmost layer of the Architecture [8].

**Application framework layer** resides below the Applications layer. This basically provides a set of services and systems such as:

a) Views: A view is a graphical style layout which the user using the application interacts with.

b) Content Providers: Provide content or data from other applications or share their own data.

c) A Resource Manager: Every application uses a some resources such as images, localized strings basically non code elements, a resource manager provides access to them. A typical example would be when making the device compatible for different screen sizes we have two same background images of different sizes, or when targeting different languages.

d) Notification Manager: This enables applications to give alerts to the user in the status bar.

e) Activity manager: This component of the application framework controls activity (a single focused thing a user can do) lifecycle[8].

**Libraries Layer** resides below the application framework layer. These libraries are not exposed to the developers directly but through the applications framework.

Some examples of these core libraries are 3d libraries which enable 3-d graphics, media libraries that support playing of different file formats such as MPEG4 etc[8].
Android Runtime Layer

This is where all applications are run in their own process using an instance of the Dalvik Virtual Machine. Each VM executes a dex format file. Dalvik VM depends on the Linux layer for threading and low level memory management[8].

Linux Kernel Layer

This is the last layer in the architecture and basically provides an abstraction layer between hardware and the software.

2.2 The Development Environment

Figure 2.2-02: Eclipse IDE
The development used by Android is Eclipse which is an open source IDE. It is able to compile and run Android programs using a plug-in which can be downloaded from the Android Google webpage. Code written in Eclipse can be tested on the emulator or can be directly tested on any Android phone able to support the version of Android you are targeting.

Figure 2.2-03: Android Emulator

The ADT plug-in for Eclipse along with installing the development settings in Eclipse also installs DDMS which stands for Dalvik debug monitor service. This service has a bunch of tools of which I will discuss few that I had the first hand experience of using.

a) LogCat: This tool provides logging facilities to the developer which he can use to debug his code as the traditional debugging facility of adding breakpoints is not available.
b) Emulator Control: This tool enables the user to provide mock data to the emulator to test out our code like in the case of an app using the gps functionality of the device one can test out the device by providing mock latitude and longitude to the emulator.

c) Memory allocation tracker tells the developer how much memory is being consumed by his application components.

Below is a screen shot of the Eclipse environment showing the various tools.

![Figure 2.2-04: DDMS Eclipse Screenshot](image)

DDMS also provides the developer the ability to take screen shots of his device screen, which is convenient when one wants to ask questions on community forms [13].
The Android SDK

Along with installing the ADT plug-in for Eclipse before a developer can actually start coding for Android apps he needs to download the Android SDK for the version of Android he is specifically targeting. The SDK for various Android versions can be downloaded from Google manually or one can use Eclipse to download the SDK. Once the SDK is downloaded, the developer needs to set up an AVD (Android virtual device) so that he can run his applications on it. This can be done easily again using the Eclipse environment[1].

2.3 Web Services:

Two web-services have been used while developing this app, namely:

1) **Google's Geocoding API** which has been used indirectly by using the Geocoding class available in the Android operating system libraries.

This library is basically used to convert addresses into latitude and longitude coordinates and latitude and longitude coordinates into addresses.

This API has a limit of 2500 request/day per /application which is sufficient in most cases for a client side application. It can take in both addresses and co-ordinates as input parameters [6].

2) **Google Places API**

This is a external web service in which the only way to access this web service is by making HTTPS calls to the Google places web service. It has a usage limit of 1000 requests/day for developing and testing purposes, but can be increased at no cost to 100,000 per day upon verifying user identity through Google console.
Calls to this web-service require co-ordinates of the user, the radius of search and the name of the business, along with a sensor parameter with a typical query string as shown below[4]

https://maps.googleapis.com/maps/api/place/search/json?location=-33.8670522,151.1957362&radius=500&types=food&name=harbour&sensor=false&key=AddYourOwnKeyHere

**Figure 2.3-01: Google Places Query[4]**

This API also requires the user to register for a key which can be easily obtained through the Google console. This API generally gives back a reply in two formats, namely JSON or XML. A typical JSON object is given below for reference.

```json
{
    "html_attributions": ["Listings by Yellow Pages\n"],
    "results": [
        {
            "geometry": {
                "location": {
                    "lat": -33.8719830,
                    "lng": 151.1990860
                }
            },
            "icon": "http://maps.gstatic.com/mapfiles/place_api/icons/restaurant-71.png",
            "id": "677679a4e58049a7e8e079e0890897eb953d79b",
            "name": "Zaaffran Restaurant - BBQ and GRILL, Darling Harbour",
            "rating": 3.90,
            "reference": "CpBjAAAhHsHslmUQATR6g6fWm6zKsDmUKq_p46BpSzPQCk1m9glTKkiAHH_G4xGttuOSj35WJJDAV90dApnNzTKZ2OxuMgogdeHKQhlehdh6UuFrW53wtxXg7UfpAzcCgIzYNI0UQcCj38cr_DE56RH4W9d2bWbbIaRydXtsx2Fmk2EeqOzJVJ-oqZYSu6l75rnxJQfsnWUVVHUp9jvc517DKkuU6dNP0qQZZqgiXGXSwGQ1wq9gtc",
            "types": ["restaurant", "food", "establishment"],
            "vicinity": "Harbourside Centre 10 Darling Drive, Darling Harbour, Sydney"
        },
        {
            "geometry": {
                "location": {
                    "lat": -33.8722580,
                    "lng": 151.1986550
                }
            },
            "icon": "http://maps.gstatic.com/mapfiles/place_api/icons/restaurant-71.png",
            "id": "27ea39c8fed1c0437069066b8dccf958a2d06f19",
            "name": "Criniti's Darling Harbour",
            "rating": 3.60,
            "reference": "CnRwAAAAAAAhHsHsHslmUQATR6g6fWm6zKsDmUKq_p46BpSzPQCk1m9glTKkiAHH_G4xGttuOSj35WJJDAV90dApnNzTKZ2OxuMgogdeHKQhlehdh6UuFrW53wtxXg7UfpAzcCgIzYNI0UQcCj38cr_DE56RH4W9d2bWbbIaRydXtsx2Fmk2EeqOzJVJ-oqZYSu6l75rnxJQfsnWUVVHUp9jvc517DKkuU6dNP0qQZZqgiXGXSwGQ1wq9gtc",
            "types": ["restaurant", "food", "establishment"],
            "vicinity": "Harbourside Centre 10 Darling Drive, Darling Harbour, Sydney"
        }
    ]
}``
"vicinity": "Shop 461, 2-10 Darling Drive, Harbourside Shopping Centre, DARLING HARBOUR",
"events": [
  {
    "event_id": "7IgK1GphU",
    "summary": "Google Maps Developer Meetup: Rockin' out with the Places API",
    "url": "https://developers.Google.com/maps/documentation/places"
  }
],
...additional results...
"status": "OK"

**Figure 2.3-02: Google Places JSON Response[4]**

This API can be used to obtain details about the place. The request URI will have to be slightly modified to get the details query [4].
3. Calendar Quest Mobile App

This chapter provides a brief explanation of the software development lifecycle that was carried out.

3.1 Requirement Analysis

1) The app should remind the user of their events by their location

2) The app should be simple to use

3) Calendar events should be able to be synchronized with popular calendars such as Google Calendar and Outlook calendar.

4) Basic editing and viewing of events on calendars should be easily done.

5) Finding a place within the vicinity to also provide directions as to how to go there

6) Weather conditions based on their current location.

7) The app should be free.

Other requirements I obtained from Google’s Android website were based on the response that mobile phones are great devices but they are limited by two main things

1) Battery power: Any application designed for mobile phones has to try and use the battery as minimally as possible[1].

2) Processing power: Processing power of smart phones is limited due to limited amount of RAM and processing power of their CPU [1].

Another aspect to keep in mind since this is an App for the Android platform, and since this platform is available on a number of phones with varying screen sizes, is that the app should be able to work optimally on each phone.

Last but not the least is the user's experience. Apps that crash or are just unresponsive, do not receive a good rating among users [1].
Of all the above requirements **numbers 1 to 4** were are definitely going to be implemented, numbers 5 and 6 will be added if time permits. By default all recommendations recommended by Google are going to be implemented.

### 3.2 Planning

#### 3.2.1 Calendar Choice

Among a variety of calendars, Google Calendar was chosen for ease of synchronization. Another calendar like the one available with Microsoft "Outlook" would require finding a way to synchronize it with the user's phone, which would requiring having an external server that pings the user's Microsoft Outlook calendar for changes in events and then pings the user's phone to update it. Since the calendar could contain user's private data, it would require encrypting and decrypting the data for security.

Android phones already have a Google Calendar app built in that does the synchronizing and maintains security by encryption. Also Google Calendar easily synchronizes with Microsoft Outlook so updating Google Calendar will update other applications as well.

One problem at the start of this project Android Google Calendar API was never made public to users. As can be seen from the below picture it was released only recently and hence has the "new" tag on it.
The lack of information regarding the implementation required installing the entire Android Operating System platform available through Github, and researching the Calendar API code.

The most attractive feature of Android is its reusability of interfaces and activities. This enabled harnessing Android's Google Calendar content providers and activities which was used to sync the Calendar to Google Calendar and also to retrieve the list of events.

### 3.2.2 Choosing the view

The selected view should add the user's event-to-location reminders by the click of a button. This view should help segregate the data obtained from Google Calendar. The

---

**Figure 3.2-01: Android Calendar Provider Screenshot[1]**
most convenient view that provided this design was the list view which is why it was chosen.

3.2.3 Storing Data

Android has several ways to store data:

1) Internal Storage: This stores files on the internal phone memory, and the files are only private to the application. However this requires the opening and closing a file, with the risk of accidentally overwriting critical data[14].

2) External storage: In this type of storage Android saves the files on an external storage system such as an SD card. However if the user does not have an SD card, it also requires opening and closing files [14].

3) Storing on an external network and then making Http calls is not a very convenient as user's want instant results, although you could cache some data in the internal storage but then why not just use the internal storage.

Finally My choice came down to the last 2 ways namely a SQL-Lite database and something called shared preferences[14].

4) SQL-Lite is a relational database management system (RDBMS) that can run independently and does not need to be hosted on a server. This makes it very suitable for mobile phones. Data is stored in tables, and all tables have a primary key which can be used to retrieve data. For an Android device, one can query the database in various ways including the standard practice of making a class for each table and returning an object, or querying the database directly using an SQL query such as

```sql
"Cursor cursor = getReadableDatabase().
    rawQuery("select * from todo where _id = ?", new String[] { id });"
```

Figure 3.2-02: SQL-Lite Query[7]
where getReadableDatabase() is a method provided by the SQLiteOpenHelper class provided by Android. A query returns a Cursor object. A Cursor represents the result of a query and basically points to one row of the query result. This makes Android handle the query efficiently and does not have to load everything in memory. The database can also be queried using the Query method in the ContentProvider class provided by Android, as shown below.

```java
query (Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)
```

Example implementation:

```java
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder) {
    SQLiteDatabase db = getWritableDatabase();
    String where = selection;
    if (projection == null) 
        projection = ContentContract.Picker.PROJECTION_ALL;
    return db.query(DATABASE_TABLE_NAME, projection, where, selectionArgs, null, null, sortOrder);
}
```

Parameters:

- `uri`: The URI to query. This will be the full URI sent by the client, if the client is requesting a specific record, the URI will end in a record number that the implementation should parse and add to a WHERE or HAVING clause, specifying the _id value.
- `projection`: The list of columns to put into the cursor. If null all columns are included.
- `selection`: A selection criteria to apply when filtering rows. If null then all rows are included.
- `selectionArgs`: You may include ?s in selection, which will be replaced by the values from selectionArgs, in order that they appear in the selection. The values will be bound as Strings.
- `sortOrder`: How the rows in the cursor should be sorted. If null then the provider is free to define the sort order.

Returns:

- a Cursor or null.

**Figure 3.2-03: SQL lite Query Implementation Example[10]**

The problems with using SQL lite is one has to open and close connections to the
database, also accessing the database in Android means accessing the file system which can be slow[7].

5) Shared Preferences is a class that provides a general framework to save and retrieve persistent key-value pairs to save primitive data such as ints, longs, floats, double, strings. One does not need to open any kind of connection strings and files to retrieve data, and being key value pairs data retrieval is extremely fast thus making it a very attractive option to be used for CalQuest.

3.3 Implementation

There were several aspects for implementation.

- The lifespan of an event. The decision was made to remove an event when the event time expired.
- Editing an event: if the user edits an event, Google Calendar assigns the event a new event Id which makes tracking difficult. To handle this, the current event (if previously added) in location reminders was removed, and the user being must add it again to the location reminders list.
- Syncing the calendar: If an upper limit time period is not specified, the list view can become long. An upper limit of one week was set, based on the realization that most people go about their business on a weekly basis.

3.3.1 Use Cases

Below are high level Use Case diagrams of my application

Use Case 1:

A start screen was added as in common in most apps.
Figure 3.3-01: Use Case 1

The use case shown above basically depicts the user clicking on the CalQuest icon and receiving the start screen (on the Android it is called Splash Activity) and the start screen redirects the user to the main app, namely the view controller.

Use Case 2:

Figure 3.3-02: Use Case 2

Use case 2 depicts the view controller class. It shows the different relationships between the view controller class and the other classes. The view controller is how the user interacts with when using the app.
Use case 3:

Use Case 3 shows the calendar background service that performs the major functionality in the background.

Figure 3.3-03: Use Case 3
3.3.2 Activity Diagram Explanation

The app starts with the user clicking on the cal-quest icon which then takes the user to the start screen. The timer for this screen can be set by the user from 0 to 5 seconds. The start screen then automatically directs the user to the applications UI (User Interface). The UI (represented by the View Controller) gathers data from the calendar database and loads it into the list-view.

Figure 3.3-04: Start Screen
The each item in the list view consist a text view, two radio buttons and a submit button. Before adding the event to location reminders the user has to tell the app whether the location is a business name such as "Vons" shown in the example below or a physical address such as "9500 xxxx Blvd Northridge CA 91324" as shown in figure 3.3-4.

![Main View](image)

**Figure 3.3-05: Main View**

Depending upon which radio button the user selects the app adds data to the key value pairs that are saved in the shared preferences file.
The Text view when pressed leads to a detailed view event as shown in Figure 3.3-5. The user can also edit the view add reminders from here. This is a view generated using Androids great feature of re-usability.

Figure 3.3-06: View event view
Holding down the text view takes the user directly to edit and delete events. Here the user can edit his event details such as time, location etc, as shown in the figure below.

![Edit/Delete Event View](image)

**Figure 3.3-07: Edit/Delete Event View**

If the Address radio button is selected the app first performs a reverse geo-code that is it converts the address into latitude and longitude co ordinates this is done to check validity of the address, if it is able to do it successfully it proceeds to add the address' co-ordinates obtained from reverse geo-coding to shared preferences. Otherwise it displays a message to the user that it cannot add the address to location reminders.

If the Business radio button is selected the app only checks to see whether the
event location field is populated as there is no way to determine whether a business name is valid or not since this requires a very large database of business names to verify against. It is assumed that the user has entered a valid business name, in the event that the user has not entered a valid Business name, when the query is sent to Google Places API will come back with zero results but it may also come back with zero results if it does not find the place within 50 km of the user's location. As the API only searches up-to 50 km, so Business name verification cannot be enforced.

On another note the user will still be reminded about the event by time reminders from Google Calendar.
Figure 3.3-08: Activity Diagram
Once the event has been added to shared preferences, it activates the Calendar Background Service which runs in the background and locates the user’s coordinates. Once the coordinates are found, it will filter out the deleted and expired events from the location reminders list. Then for each event, it will check based on the event key if it is an address event or if it is a business event. If it is a business event, it will call the Google Places web-service. Once it receives the response from the service, if it determines that it has valid result it parses the response to get the coordinates of the closest business and then checks them with the user’s current coordinates. If it finds it within the range, it displays a notification to the user.

If however the Address button is clicked, it parses the key value pair in the shared preferences and retrieves the coordinates. Then it checks whether the user is within the specified radius and issues a notification if so.
3.3.3 Sequence Diagrams

Figure 3.3-8 shows the sequence in which parts of the app interact with each other.

![Sequence Diagram 1](image1)

**Figure 3.3-09: Sequence Diagram 1**

![Sequence Diagram 2](image2)

**Figure 3.3-10: Sequence Diagram 2**
Figure 3.3-11: Sequence Diagram 3
3.4 Testing

When making the app the following types of testing were conducted

Unit testing

The app's various functions were tested independent of the system. The following types of unit test were conducted:

- The navigation from one view to another
- Synchronization of data between Google Calendar and the app
- Adding of events to location reminders
- Removal of deleted and expired events
- Reverse geo-coding of address
- Calls to web-service
- Parsing of response data from the web-service
- Finally firing of notifications when user is within radius

Integration Testing

Integration testing was done after all the modules of the app were completed.

This was done to test the bugs after all the modules were combined.

Examples of Integration test cases are:

- All functionality as specified should work correctly
- Navigation between views should not crash
- Data synchronization between Google Calendar and the app should be correct
- User should get a notification when within radius of a specified event location
• Storing and removal of events from shared preferences

4. Review Summary

4.1 Problems Encountered

The first problem encountered when starting the Android application was how to access the Google Calendar and keep it updated and synchronized. As at that time, Android had not released support for Android Google Calendar. This was overcome by installing the whole Android operating system project from Git Hub onto a Linux machine, and studying it to get the content provider for Google Calendar.

2) Google's Android API has useful functions, but lacks examples. This was overcome through experimentation and online help forums such as stackoverflow.com.

3) Another problem faced was debugging, since the Android emulator provided by Google does not have Google Calendar built in. Building the app had to be done entirely directly on the phone. The traditional method of putting break points and debugging code could not be done, as it was not supported. This was overcome by using the "Logcat" tool provided through the adt plug-in which basically gives text output of what is happening.

4) Another problem initially was an older version of Eclipse installed on the computer which was not compatible with the ADT plug-in provided by Google. This resulted in Eclipse crashing. Upgrading to a newer version of Eclipse fixed the issue.

5) Battery life is a very important feature to keep in mind when developing mobile applications, and continuous usage of GPS can drain a battery quite fast. Measures taken to reduce battery usage were to automatically kill the Background Service when
no more location reminders were left. If the Android phone's screen has been in active for some time, the application automatically increases the firing interval of the GPS so as to minimize battery drainage, and comes back to lower value when the screen becomes active again.

In reality increasing the firing interval of GPS scans is the only way to reduce battery drainage and this will vary as to how frequently users want the GPS location scans to fire and what kind of phone and battery (new or old) they are using. Thus the decision was made to let the user finally decide the firing interval.

4.2 Degree of Success

The application is working according to the requirements finalized in section 3.1. The Calendar is able to synchronize with Google Calendar. The user has basic viewing and editing event capabilities. The app reminds the user about event per location. The interface is very simple to use. At the touch of two clicks, one can easily add the event to location reminders.

The response time is quick, as most of the main calculation is done by the service in the background away from the main thread.

4.3 Future Work or Improvements

1) The user receives a notification from the app when in the vicinity of the business place. A future addition to the project could be to provide a list of coupons and deals at the place of business, as well as enabling reviews of the place of business which could be then posted to online forums for others to access.

2) Google Places API also supports what it calls a "Place Details Request" which gives us a more comprehensive list of information about a particular place, such as
address, phone number, user rating. These could all be provided to the user, which the user could use to either call the place in advance or read reviews about it and decide whether or not to go there.

3) At present the app has a plain simple list view it would be interesting to add different views to the app like monthly view like the one designed by Google for the Android Google Calendar as shown below.

![Android Google Calendar Monthly View](image)

**Figure 4.3-01: Android Google Calendar Monthly View**
Bibliography


doi: 10.1109/IMSAA.2009.5439442


<http://www.gsmarena.com/t_mobile_g1-2533.php>.
Appendix

Hardware used

Phone: T-mobile G1 specs as follows [15]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GENERAL</strong></td>
<td></td>
</tr>
<tr>
<td>2G Network</td>
<td>GSM 850 / 900 / 1800 / 1900</td>
</tr>
<tr>
<td>3G Network</td>
<td>HSDPA 2100</td>
</tr>
<tr>
<td>Announced</td>
<td>2008, September</td>
</tr>
<tr>
<td>Status</td>
<td>Available, Released 2008, October</td>
</tr>
<tr>
<td><strong>BODY</strong></td>
<td></td>
</tr>
<tr>
<td>Dimensions</td>
<td>117 x 55.7 x 17.1 mm</td>
</tr>
<tr>
<td>Weight</td>
<td>158 g</td>
</tr>
<tr>
<td>Keyboard</td>
<td>QWERTY</td>
</tr>
<tr>
<td><strong>DISPLAY</strong></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>TFT capacitive touchscreen, 55K colors</td>
</tr>
<tr>
<td>Size</td>
<td>320 x 480 pixels, 3.2 inches (~180 ppi pixel density)</td>
</tr>
<tr>
<td>Protection</td>
<td>Corning Gorilla Glass</td>
</tr>
<tr>
<td></td>
<td>- Trackball</td>
</tr>
<tr>
<td><strong>SOUND</strong></td>
<td></td>
</tr>
<tr>
<td>Alert types</td>
<td>Vibration; Downloadable polyphonic, MP3, WAV ringtones</td>
</tr>
<tr>
<td>Loudspeaker</td>
<td>Yes</td>
</tr>
<tr>
<td>3.5mm jack</td>
<td>No</td>
</tr>
<tr>
<td><strong>MEMORY</strong></td>
<td></td>
</tr>
<tr>
<td>Card slot</td>
<td>microSD</td>
</tr>
<tr>
<td>Internal</td>
<td>192 MB RAM, 256 MB ROM</td>
</tr>
<tr>
<td><strong>DATA</strong></td>
<td></td>
</tr>
<tr>
<td>GPRS</td>
<td>Class 10 (4+1/3+2 slots), 32 - 48 kbps</td>
</tr>
<tr>
<td>EDGE</td>
<td>Class 10, 236.8 kbps</td>
</tr>
<tr>
<td>Speed</td>
<td>HSDPA 7.2 Mbps; HSUPA, 2 Mbps</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wi-Fi 802.11 b/g</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Yes v2.0 with A2DP, headset support only</td>
</tr>
<tr>
<td>USB</td>
<td>Yes, miniUSB</td>
</tr>
<tr>
<td><strong>CAMERA</strong></td>
<td></td>
</tr>
<tr>
<td>Primary</td>
<td>3.15 MP, 2048x1536 pixels, autofocus</td>
</tr>
<tr>
<td>Video</td>
<td>Yes</td>
</tr>
<tr>
<td>Secondary</td>
<td>No</td>
</tr>
<tr>
<td><strong>FEATURES</strong></td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>Android OS, v1.6 (Donut)</td>
</tr>
<tr>
<td>Chipset</td>
<td>Qualcomm MSM7201A</td>
</tr>
<tr>
<td>CPU</td>
<td>528 MHz ARM 11</td>
</tr>
<tr>
<td>GPU</td>
<td>Adreno 130</td>
</tr>
<tr>
<td>Sensors</td>
<td>Accelerometer, compass</td>
</tr>
<tr>
<td>Messaging</td>
<td>SNS (threaded view), MMS, Email, Instant Messaging</td>
</tr>
<tr>
<td>Browser</td>
<td>HTML</td>
</tr>
<tr>
<td>Radio</td>
<td>No</td>
</tr>
</tbody>
</table>
Developed on a HP Laptop with the following specifications running windows 7 professional 64 bit operating system.
Source Code:

package com.Android.LocationCalendar;

import java.io.IOException;

import java.util.*;
import Android.app.Activity;
import Android.app.ActivityManager;
import Android.app.ActivityManager.RunningServiceInfo;
import Android.app.AlertDialog;
import Android.app.ListActivity;
import Android.content.ContentResolver;
import Android.content.ContentUris;
import Android.content.Context;
import Android.content.Intent;
import Android.content.SharedPreferences;
import Android.database.Cursor;
import Android.graphics.Color;
import Android.location.Address;
import Android.location.Geocoder;
import Android.net.Uri;
import Android.os.Bundle;
import Android.text.format.DateUtils;
import Android.util.Log;
import Android.view.LayoutInflater;
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.view.ViewGroup;
import Android.widget.BaseAdapter;
import Android.widget.Button;
import Android.widget.CheckBox;
import Android.widget.CompoundButton;
import Android.widget.CompoundButton.OnCheckedChangeListener;
import Android.widget.ListView;
import Android.widget.RadioButton;
import Android.widget.RadioGroup;
import Android.widget.TextView;
import Android.widget.Toast;

@SuppressWarnings("unused")
public class viewController extends ListActivity {
    private List<userEvents> myEvents = new ArrayList<userEvents>();
    private MyCustomAdapter calAdapter;
    public static String sharedPrefrenceFile = "calQuestPrefFile";
    SharedPreferences prefData;
    private static boolean firstRun = true;
    
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        
    }
    
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
    
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
firstRun = false;

prefData = getSharedPreferences(sharedPrefrenceFile,0);

createlistView();

protected void onListItemClick(ListView l, View v, int position, long id) {
    // launchEvent(Intent.ACTION_VIEW, position);
}

private void createlistView(){
    myEvents = calQuery(this);
    if(myEvents.size()==0){
        setContentView(R.layout.main);
        TextView maintxtview = (TextView)findViewById(R.id.maintxtview);
        maintxtview.setText("You do not have any events in the coming week");
    }
    else{
        calAdapter = new MyCustomAdapter();
        for(userEvents myevent : myEvents){
            calAdapter.addItem(myevent);
        }
        setListAdapter(calAdapter);
    }
}

protected void onResume(){

super.onResume();
if(firstRun)
createmListView();
}

private void launchEvent(String action, int position) {
    userEvents event = myEvents.get(position);
    Intent intent = new Intent(action);
    Uri eventUri =
        ContentUris.withAppendedId(Uri.parse("content://calendar/events/"),
                                  event.getUserEventId());
    intent.setData(eventUri);
    intent.putExtra("beginTime", event.getStartTimeUTC().getTime());
    intent.putExtra("endTime", event.getEndTimeUTC().getTime());
    startActivity(intent);
}

private HashSet<String> getCalendarIds(Context context) {
    ContentResolver contentResolve = context.getContentResolver();
    final Cursor cursor =
        contentResolve.query(Uri.parse("content://calendar/calendars"),
                             (new String[] { "_id", "displayName", "selected" }), null, null, null);
    HashSet<String> calendarIds = new HashSet<String>();
    while (cursor.moveToNext()) {
        final String _id = cursor.getString(0);
        calendarIds.add(_id);
    }
    return calendarIds;
}
final String displayName = cursor.getString(1);

final Boolean selected = !cursor.getString(2).equals("0");
calendarIds.add(_id);
}
return calendarIds;

private List<userEvents> calQuery(Context context){

    String title;
    Date begin;
    Date end;
    Boolean allDay;
    int eventId;
    String eventLocation;
    String eventDescription;
    int calId;

    ContentResolver contentResolve = context.getContentResolver();
    List<userEvents> Events = new ArrayList<userEvents>();
    HashSet<String> calendarIds = getCalendarIds(context);
    final String LOG_TAG = "Tag";

    for (String id : calendarIds) {

        Uri.Builder builder =

        Uri.parse("content://calendar/instances/when").buildUpon();
        long now = new Date().getTime();
        }
ContentUris.appendId(builder, now);
ContentUris.appendId(builder, now + DateUtils.WEEK_IN_MILLIS);
Cursor eventCursor = contentResolve.query(builder.build(),
new String[] { "title", "begin", "end", "allDay","event_id", "color",
"_id","eventLocation", "startDay",
"startMinute","description","calendar_id" }, "Calendars._id=" + id,
null, "startDay ASC, startMinute ASC");
while (eventCursor.moveToNext()) {
    title = eventCursor.getString(0);
    begin = new Date(eventCursor.getLong(1));
    end = new Date(eventCursor.getLong(2));
    allDay = !eventCursor.getString(3).equals("0");
    eventId = eventCursor.getInt(4);
    eventLocation = eventCursor.getString(7);
    eventDescription = eventCursor.getString(10);
    calId = eventCursor.getInt(11);
    Events.add(new
    userEvents(title,begin,end,allDay,eventId,eventLocation,eventDescription,calId));
    Log.v(LOG_TAG,"Title: " + title + " Begin: " + begin + " End: " + end
    " All Day: " + allDay+ " id: " + eventId + " calendar_id "+
    calId+ "_id:"+eventCursor.getInt(6));
}
}
return Events;

private class MyCustomAdapter extends BaseAdapter {
    private ArrayList<userEvents> myEventList = new ArrayList<userEvents>();
    private LayoutInflater layoutInflater;
    public MyCustomAdapter() {
        layoutInflater = (LayoutInflater)getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }
    public void addItem(userEvents event) {
        myEventList.add(event);
        notifyDataSetChanged();
    }
    @Override
    public int getCount() {
        return myEventList.size();
    }
    @Override
    public userEvents getItem(int position) {
        return myEventList.get(position);
    }
    @Override
    public long getItemId(int position) {

return position;

}

@Override

public View getView(final int position, View convertView, ViewGroup parent)
{

String tempkey = null;
userEvents currentEvent = myEventList.get(position);
final String value = currentEvent.getUserEventLocation() + "^" +
currentEvent.getEventTitle();

final SharedPreferences.Editor editor = prefData.edit();
Map<String, ?> eventKeys = prefData.getAll();
boolean allDay = currentEvent.getAllDay();
viewContainer container = null;
if (convertView == null) {
    container = new viewContainer();
    convertView = layoutInflater.inflate(R.layout.listview,null);
    container.txtView = (TextView)convertView.findViewById(R.id.text);
    convertView.setTag(container);
} else {
    container = (viewContainer)convertView.getTag();
}
else {
    container = (viewContainer)convertView.getTag();
}

TextView calTextView = container.txtView;
String loc = currentEvent.getUserEventLocation();
if(loc == null || loc.length() == 0) {loc = "Location not specified";}

if(!allDay){
    calTextView.setText("Title: " + currentEvent.getEventTitle() + \\
    "Location: " + currentEvent.getUserEventLocation() + \\
    "Start Time: " + currentEvent.getStartTimeUTC().toLocaleString() + \\
    "End Time: " + currentEvent.getEndTimeUTC().toLocaleString();
    tempkey = "CalQuest" + "^" + currentEvent.getCalId() + "^" + \\
    currentEvent.getUserEventId() + "^" + currentEvent.getEndTimeUTC();
}
else {
    calTextView.setText("Title: " + \\
    currentEvent.getEventTitle() + \\
    "Location: " + loc + " All Day Event");
    tempkey = "CalQuest" + "^" + currentEvent.getCalId() + "^" + \\
    currentEvent.getUserEventId() + "^" + "All Day Event";
}
final String key = tempkey;

calTextView.setBackgroundResource(R.drawable.bg_color_states);

calTextView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        launchEvent(Intent.ACTION_VIEW, position);
    }
});
calTextView.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {
        launchEvent(Intent.ACTION_EDIT, position);
        return true;
    }
});

container.radioButton1 = (RadioButton)convertView.findViewById(R.id.rad1);
container.radioButton2 = (RadioButton)convertView.findViewById(R.id.rad2);
container.radgrp = (RadioGroup)convertView.findViewById(R.id.radgroup);
final RadioButton radio1 = container.radioButton1;
final RadioButton radio2 = container.radioButton2;
container.button = (Button)convertView.findViewById(R.id.button1);
final Button locAdder = container.button;
if(eventKeys.containsKey(key)){
    locAdder.setText("Remove from location reminder");
}
else  locAdder.setText("Add to location reminder");
locAdder.setOnClickListener(new OnClickListener(){
    @Override
    public void onClick(View v) {

    }
});
public void onClick(View v) {
    if(locAdder.getText().equals("Add to location reminder")){
        if(radio2.isChecked()){
            locAdder.setText("Adding Please Wait");
            String coordinates = addressToCoordinates(value);
            if(coordinates!=null){
                editor.putString(key, "GeoCodeApi^"+value+coordinates);
                editor.commit();
                toastNotification("Event Successfully Added to Location Reminders By address");
                locAdder.setText("Remove from location reminder");
                startService(new Intent("com.Android.locationcalendar.CALENDARBACKGROUNDSERVICE"));
            } else{toastNotification("Cannot add to location Reminders:Address is invalid or is too vague or event location is empty
Please read app usage instructions in menu");
                locAdder.setText("Add to location reminder");
            }
        }
    }
    if(radio1.isChecked()){
    }
}

if(radio1.isChecke
if(checkPlaceValid(value)) {
    locAdder.setText("Adding Please Wait");
    editor.putString(key, "PlacesApi^"+value);
    editor.commit();
    toastNotification("Event Successfully Added to 
    Location Reminders By Place
    Please remember there is no way for the app to know 
    whether the place name you have entered is correct 
    for Instructions on what is a correct name read usage instructions");
    locAdder.setText("Remove from location 
    reminder");
    startService(new 
    Intent("com.Android.locationcalendar.CALENDARBACKGROUNDSERVICE"));
}
else {toastNotification("No event location specified,
    Please read app usage instructions in menu ");}

if(!radio1.isChecked() && !radio2.isChecked()){
    toastNotification("Please specify whether the Event Location 
is a Business Name or an Address");
}

else {
    editor.remove(key);
}
editor.commit();

locAdder.setText("Add to location reminder");
toastNotification("Event Removed From Location Reminders");

startService(new Intent("com.Android.locationcalendar.CALENDARBACKGROUNDSERVICE"));

private String addressToCoordinates(String value){
    if(value == null)return null;
    Geocoder coder = new Geocoder(viewController.this);
    String eventHighlights [] = value.split("\^\^");
    String eventlocation = eventHighlights[0];
    if(!eventlocation.equals("null") && eventlocation.length()!=0){
        List<Address> address;
        try {
            address = coder.getFromLocationName(eventlocation, 1);
            if(!address.isEmpty())
                return "^"+address.get(0).getLatitude()+"^"+address.get(0).getLongitude();
        }
    }
    firstRun = true;
    return convertView;
}
else return null;
}

} catch (IOException e) {
    e.printStackTrace();
    return null;
}
} else return null;

private boolean checkPlaceValid(String value) {
    if(value == null){return false;}
    boolean isplaceValid = false;
    String eventHighlights [] = value.split("\^");
    String eventlocation = eventHighlights[0];
    if(!eventlocation.equals("null") && eventlocation.length()!=0)
    isplaceValid = true;

    return isplaceValid;
}

private void toastNotification(CharSequence text){
    Context context = getApplicationContext();
    int duration = Toast.LENGTH_LONG;
    Toast toast = Toast.makeText(context, text, duration);
    toast.show();
}
private boolean isMyServiceRunning() {
    ActivityManager manager = (ActivityManager)
    getSystemService(ACTIVITY_SERVICE);
    for (RunningServiceInfo service :
    manager.getRunningServices(Integer.MAX_VALUE)) {
        if
        ("com.Android.locationcalendar.CALENDARBACKGROUNDSERVICE".equals(service
        .service.getClassName())) {
            return true;
        }
    }
    return false;
}

private static class viewContainer {
    public TextView txtView;
    public Button button;
    public RadioButton radioButton1;
    public RadioButton radioButton2;
    public RadioGroup radgrp;
}

private void deleteAllReminders(){
    prefData = getSharedPreferences(sharedPrefrenceFile,0);
Map<String, ?> items = prefData.getAll();

SharedPreferences.Editor editor = prefData.edit();

for(String s : items.keySet()){
    editor.remove(s);
    editor.commit();
}

package com.Android.LocationCalendar;

import java.io.IOException;
import java.io.InputStream;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONObject;
import com.google.android.maps.GeoPoint;
import Android.app.Service;
import Android.content.ContentResolver;
import Android.content.ContentUris;
import Android.content.Context;
import Android.content.Intent;
import Android.content.SharedPreferences;
import Android.database.Cursor;
import Android.location.Address;
import Android.location.Criteria;
import Android.location.Geocoder;
import Android.location.Location;
import Android.location.LocationListener;
import Android.location.LocationManager;
import Android.location.LocationProvider;
import Android.net.Uri;
import Android.os.Bundle;
import Android.os.IBinder;
import Android.text.format.DateUtils;
import Android.util.Log;
import Android.widget.Toast;

@ SuppressWarnings("unused")

public class calendarBackgroundService extends Service implements LocationListener {
    private static final String TAG = calendarBackgroundService.class.getSimpleName();
    private LocationManager locationManager;
    public static String sharedPrefrenceFile = "calQuestPrefFile";
    SharedPreferences prefData;
    private static Boolean isFirstStart = false;
    private String provider;
    private boolean isLmRunning = false;
    private ArrayList<BasicNameValuePair> currLocRemList = new ArrayList<BasicNameValuePair>();

    @Override
    public IBinder onBind(Intent arg0) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d(TAG, "Service created");
    }

    @Override
    public void onLocationChanged(Location location) {
        // TODO Auto-generated method stub
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        // TODO Auto-generated method stub
    }

    @Override
    public void onProviderDisabled(String provider) {
        // TODO Auto-generated method stub
    }

    @Override
    public void onProviderEnabled(String provider) {
        // TODO Auto-generated method stub
    }

    @Override
    public boolean onQueryLocation(LocationRequest locationRequest) {
        // TODO Auto-generated method stub
        return false;
    }

    @Override
    public void onProviderPrepareFailed(String provider) {
        // TODO Auto-generated method stub
    }

    @Override
    public void onLocationNotAvailable() {
        // TODO Auto-generated method stub
    }

    @Override
    public void onLocationRemoved() {
        // TODO Auto-generated method stub
    }

    @Override
    public void onLocationProviderEnabled(String provider) {
        // TODO Auto-generated method stub
    }

    @Override
    public void onLocationProviderDisabled(String provider) {
        // TODO Auto-generated method stub
    }
}

61
isFirstStart = true;
startLocationManager();

@Override
public void onDestroy() {
    super.onDestroy();
    locationManager.removeUpdates(this);
    Log.d(TAG, "Service ended");
}

@Override
public void onStart(Intent intent, int startId) {
    // TODO Auto-generated method stub
    super.onStart(intent, startId);
    Log.d(TAG, "Service started");
    if(!isFirstStart){
        //reminderAdder();
    }
    isFirstStart = false;
}

public HashSet<String> getCalendarIds(Context context){
    ContentResolver contentResolve = context.getContentResolver();
    final Cursor cursor =
    contentResolve.query(Uri.parse("content://calendar/calendars"),
HashSet<String> calendarIds = new HashSet<String>();

while (cursor.moveToNext()) {
    final String _id = cursor.getString(0);
    //final String displayName = cursor.getString(1);
    //final Boolean selected = !cursor.getString(2).equals("0");
    calendarIds.add(_id);
}

return calendarIds;
}

private ArrayList<String> calQuery(Context context) {
    ContentResolver contentResolve = context.getContentResolver();
    ArrayList<String> myCurrentEvents = new ArrayList<String>();
    HashSet<String> calendarIds = getCalendarIds(context);
    String key = null;
    for (String id : calendarIds) {
        Uri.Builder builder =
        Uri.parse("content://calendar/instances/when").buildUpon();
        long now = new Date().getTime();
        ContentUris.appendId(builder, now);
        ContentUris.appendId(builder, now + DateUtils.WEEK_IN_MILLIS);
        Cursor eventCursor = contentResolve.query(builder.build(),
        (new String[] { "_id", "displayName", "selected" }));
        null, null, null);
    }
new String[] { "end", "allDay","event_id", "event_id","calendar_id" },"Calendars._id=" + id, null, "startDay ASC, startMinute ASC" );

while (eventCursor.moveToNext()) {
    if (eventCursor.getString(1).equals("0")) {
        key = "CalQuest"+"^"+eventCursor.getInt(4)+"^"+eventCursor.getInt(3)+"^"+new Date(eventCursor.getLong(0));
    } else {
        key = "CalQuest"+"^"+eventCursor.getInt(4)+"^"+eventCursor.getInt(3)+"^"+"All Day Event";
    }
    myCurrentEvents.add(key);
}

private void eventFilter(){
    //this will delete all deleted events and also expired events
    currLocRemList.clear();
    prefData = getSharedPreferences(sharedPrefrenceFile,0);
    ArrayList<String> eventsInCalendar = calQuery(this);
    SharedPreferences.Editor editor = prefData.edit();

Map<String, ?> eventKeys = prefData.getAll();
for(String s:eventKeys.keySet()){  
    if(s.startsWith("CalQuest", 0)){  
        if(eventsInCalendar.contains(s)) { 
            currLocRemList.add(new 
            BasicNameValuePair(s,prefData.getString(s,"null"))); 
        } 
        else { 
            editor.remove(s);  
            editor.commit(); 
        } 
    } 
} 
private boolean startLocationManager(){
    locationManager = (LocationManager) 
    getSystemService(Context.LOCATION_SERVICE); 
    Criteria criteria = new Criteria();
    criteria.setAccuracy(Criteria.ACCURACY_FINE); 
    criteria.setAltitudeRequired(false);
    criteria.setBearingRequired(false); 
    criteria.setCostAllowed(false); 
    criteria.setPowerRequirement(Criteria.POWER_LOW); 
}
this.provider = locationManager.getBestProvider(criteria, false);
System.out.println("Provider = "+provider);
locationManager.requestLocationUpdates(provider,1000 * 60,200,this);
return isLmRunning = true;
}

private void toastNotification(CharSequence text){
Context context = getApplicationContext();
int duration = Toast.LENGTH_LONG;
Toast toast = Toast.makeText(context, text, duration);
toast.show();
}

@Override
public void onLocationChanged(Location location) {
    // TODO Auto-generated method stub
toastNotification("location changed fired");
eventFilter();
if(currLocRemList.size()==0){
    stopSelf();
}
prefData = getSharedPreferences(sharedPrefrenceFile,0);
String [] keyComponents,valueComponents = null;
Location eventLocation;
float distance = 0;
for(BasicNameValuePair keyValue: currLocRemList) {
    keyComponents = keyValue.getName().split("\\^\^");
    valueComponents = keyValue.getValue().split("\\^\^");
    if(!keyValue.getValue().startsWith("GeoCodeApi^")) {
        try {
            StringBuilder responseString = new StringBuilder();
            String eventLoc = valueComponents[1].replaceAll(" ", "\%20");
            System.out.println(location.getLatitude() + " coordinates service" + location.getLongitude());
            System.out.println("eventLoc :" + eventLoc);
            System.out.println("value :" + keyValue.getValue());
            System.out.println("value :" + keyValue.getName());
            HttpPost uri = new HttpPost("https://maps.googleapis.com/maps/api/place/search/json?location=" + location.getLatitude() +"," + location.getLongitude() + "&name=" + eventLoc + "&sensor=true&rankb" + "=distance&key=AIzaSyBA0rzSebA6DxRbFCu2lzqIQfIWiFneQU8");
            HttpClient client = new DefaultHttpClient();
            HttpResponse response = client.execute(uri);
            InputStream responseStream = response.getEntity().getContent();
            int reader;
            while((reader = responseStream.read()) != -1) {
                responseString.append((char)reader);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
JSONObject jsonObject = (JSONObject)new JSONObject(responseString.toString());

String status = jsonObject.getString("status");
System.out.println("status :"+ status);
if(status.equals("OK")){
    JSONArray results = jsonObject.getJSONArray("results");
    String vicinity = results.getJSONObject(0).getString("vicinity");
    System.out.println("vicinity :"+vicinity);
    JSONObject closestLocation = results.getJSONObject(0).getJSONObject("geometry").getJSONObject("location");
    eventLocation = new Location("Event_Location");
    eventLocation.setLatitude(closestLocation.getDouble("lat"));
    eventLocation.setLongitude(closestLocation.getDouble("lng"));
    distance = location.distanceTo(eventLocation);
    System.out.println("distance :" + distance);
    if(distance<5000){
        toastNotification("eventLocation within range");
    }
}
catch(Exception e){
    continue;
}
} else{
    eventLocation = new Location("Event_Location");
    eventLocation.setLatitude(new Double(valueComponents[3]));
    eventLocation.setLongitude(new Double(valueComponents[4]));
    distance = location.distanceTo(eventLocation);
    System.out.println("distance : " + distance);
    System.out.println("Co-ordinates " + valueComponents[3] + " +valueComponents[4]);
    if(distance<5000){
        toastNotification("eventLocation within range");
    }
} }

@Override
public void onProviderDisabled(String provider) {
    toastNotification("Please enable your GPS for location reminders to work");
    locationManager.removeUpdates(this);
    stopSelf();
}

@Override
public void onProviderEnabled(String provider) {
toastNotification("Please restart service in CalQuest App to resume location reminders");

}

@Override

public void onStatusChanged(String provider, int status, Bundle extras) {
    // TODO Auto-generated method stub
    switch(status) {
        case LocationProvider.OUT_OF_SERVICE:
            toastNotification("Location reminder service is temporarily OUT_OF_SERVICE");
            break;

        case LocationProvider.TEMPORARILY_UNAVAILABLE:
            isLmRunning = false;
            toastNotification("Location reminder service is TEMPORARILY UNAVAILABLE");
            break;

        case LocationProvider.AVAILABLE:
            toastNotification("Location reminder service has resumed again");
            startLocationManager();
            break;
    }
}
package com.Android.LocationCalendar;

import java.util.Date;

public class userEvents {

    String userEventTitle;
    String userEventLocation;
    int userEventId;
    Date userEventStartTimeUTC;
    Date userEventEndTimeUTC;
    int userEventStartMinute;
    long userEventBeginday;
    boolean userEventAllDay;
    int userEventColor;
    String userEventDescription;
    int calId;

    public userEvents(String eventtitle,Date startutc,Date endutc,boolean allday,int usereventId,String usrEventLocation
            ,String usereventdescription,int calid)
    {
        userEventTitle = eventtitle;
        userEventStartTimeUTC = startutc;
        userEventEndTimeUTC = endutc;
        userEventAllDay = allday;
        userEventId= usereventId;
    }
}
userEventLocation = usrEventLocation;
userEventDescription = usereventdescription;
calId = calid;

} 
public int getcalId(){ return calId; }

public String getEventTitle(){return userEventTitle; }
public Date getStartTimeUTC(){return userEventStartDateTimeUTC; }
public Date getEndTimeUTC(){return userEventEndTimeUTC; }
public boolean getAllDay(){return userEventAllDay; }
public int getuserEventId(){return userEventId; }
public String getuserEventLocation(){return userEventLocation; }
public String getuserEventDescription(){return userEventDescription; }

}

package com.Android.LocationCalendar;
import Android.app.Activity;
import Android.content.Context;
import Android.content.Intent;
import Android.os.Bundle;
import Android.widget.Toast;

public class splash extends Activity { 
    
@Override
    
protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub

    }

    }
super.onCreate(savedInstanceState);
setContentView(R.layout.splash);
toastNotification("Starting Application Please Wait");

Thread timer = new Thread()
{
    public void run()
    {
        try{
            sleep(1000);
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
    }
}

Intent openMainClass = new Intent("com.Android.locationcalendar.MAINCLASS");
startActivity(openMainClass);

@override
protected void onPause() {
    // TODO Auto-generated method stub
    super.onPause();
    finish(); // kills splash activity
private void toastNotification(CharSequence text) {
    Context context = getApplicationContext();
    int duration = Toast.LENGTH_SHORT;
    Toast toast = Toast.makeText(context, text, duration);
    toast.show();
}

<?xml version="1.0" encoding="utf-8"?>
<selector
    xmlns:Android="http://schemas.android.com/apk/res/Android">
    <item
        Android:state_selected="false"
        Android:state_pressed="false"
        Android:drawable="@color/ltgrey" />
    <item Android:state_pressed="true"
        Android:drawable="@color/dkgrey" />
    <item Android:state_selected="true"
        Android:state_pressed="false"
        Android:drawable="@color/medgrey" />
</selector>

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
<TextView Android:visibility="visible"
    Android:layout_height="wrap_content"
    Android:gravity="center_vertical"
    Android:textColor="#FF000000"
    Android:text="text"
    Android:background="@drawable/bg_color_states"
    Android:layout_width="fill_parent"
    Android:id="@+id/text">
</TextView>

<RadioButton Android:id="@+id/rad1"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:text="Business Name"
<RadioButton
    Android:id="@+id/rad2"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:text="Address"
/>

</RadioGroup>

<Button
    Android:layout_gravity="bottom"
    Android:layout_height="wrap_content"
    Android:layout_width="fill_parent"
    Android:id="@+id/button1"/>

</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>

    Android:orientation="vertical"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    >

<TextView
    Android:id="@+id/maintxtview"
    Android:layout_width="fill_parent"
    Android:layout_height="wrap_content"
    Android:text="@string/app_name"
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:background="@drawable/introimage">
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.android.LocationCalendar"
    android:versionCode="1"
    android:versionName="1.0">
<uses-sdk android:minSdkVersion="4" />
<uses-permission android:name="android.permission.READ_CALENDAR" />
<uses-permission android:name="android.permission.WRITE_CALENDAR"/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<application android:icon="@drawable/icon" android:label="@string/app_name">
<uses-library android:name="com.google.android.maps" />

<activity android:name=".splash">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name=".viewController"
    android:label="@string/app_name">
    <intent-filter>
        <action android:name="com.android.locationcalendar.MAINCLASS" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

<activity android:name=".Maps"
    android:label="@string/app_name">
    <intent-filter>
        <action android:name="com.android.locationcalendar.MAPS" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

<service android:name=".calendarBackgroundService"
    android:process=":remote" >
<intent-filter>

<action
android:name="com.android.locationcalendar.CALENDARBACKGROUNDSERVICE"
/>

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

</service>

</application>

</manifest>