**Problem Statement:**

1. Person travelling in train/bus after a long day at work and wants to take a nap but could not because she does not want to miss the stop
2. Person in a new town is using public transport to reach to her destination. She is continuously tracking GPS so that she does not miss her stop. She is not able to enjoy the town

There is a requirement for an app that will detect when the user reaches destination and inform user that she has reached the destination.

**Solution:**

There are a few applications in the app store, which tries to solve the above stated problem. They all work well to solve the problem but the problem lies either with their UI or they are paid apps. The UI of these apps is cluttered and not intuitive. The aim of this project was to build a Location Based Alarm, which removes all the cluttering of the existing apps and presents users with simple and easy to use app.

Application starts with a home screen, which has text field to enter the destination address, and buttons to look up the input address or show the history page. Home screen also has a map, which shows current user location. User can also provide destination by long pressing on the map. When the destination has provided by the user by any of the three means, the map will update the location and will be annotated at the destination location. Map will also show a circle with radius of 500 meters with center as the destination address. User can now press set alarm button to set the alarm for that destination and the app would show “status” screen.

In the status screen, there is a map view which show and update the current location of the user continuously. Destination location will be annotated and overlaid but that will be visible only when user is about to reach destination. As the user is at predetermined distance from destination, app will notify they user.

Apart from the above two main scenes, app also has two scenes for history and configuration each. History scene shows all the previous locations, user has searched for. History could be deleted by swiping to the left. Configuration scene helps user to configure the app. It gives option to the user such as choice of unit system, distance from destination when the alarm should sound and choice just to vibrate or vibrate with sound.

Apart from this, app also supports background location updates and local push notifications. App uses MapKit, CoreLocation and AVFoundation frameworks to perform desired actions. Also, app uses NSUserDefaults instead of CoreData to store user location history. App supports any iPhone sizes and both portrait and landscape mode.

**Project Description:**

Location Alarm is a great app for anyone who wants to do anything without the fear of missing the stop. The application is designed to be intuitive and very simple to use. App supports all iPhones of different sizes in both landscape and portrait mode for all the scenes in app.

**App Features:**

Following features have been implement in the app:

1. *MapKit* → To display users location with annotations and overlays
2. *CoreLocation* → To get current location as well as the destination location. This framework is also used to calculate distance between two locations.
3. *AVFoundation* → To play audio
4. *Auto Layout* → To make the app work perfectly on all iPhones
5. *NSUserDefaults* → To stores users location search history. Given that size of the data would not be large, I decided not to go for Core Data
6. *Background Location Updates* → Location data will be updated even if the application is in background or iPhone is locked
7. *Push Notification* → To send notification to user that he has reached the stop when either the app is in background or iPhone is locked.

**Challenges:**

For this project, I was using features, which I have not used before so I strategized to develop mini apps for all the features and explore possible problems and solutions before I integrate it to final product. This approach helped me to modularize the issues and work on specific problems and fixing rather than working on overlapping issues which would have been difficult to debug and find solutions. By using this approach, I faced some problem during integration but they were easy to resolve.

1. Google Maps API or MapKit/CoreLocation

As Google Maps API is more evolved and provides better features, I decided to develop a basic map application using Google Maps API. Personally, I found it difficult to implement. I found Apple’s MapKit easy to use as it has its own view controller and it integrates seamlessly with the code. Though Google Maps API might be better but I decided not to go with it because of the following reasons:

* Needed time to explore working with API keys
* Google Maps API cannot be directly integrated. I have to use CocoaPods. Though CocoaPods is a great repository of some cool iOS features but it needed time to explore and implement.
* There was not view available for Google Maps API

Given these challenges and time constraints, I preferred to go with MapKit. I would definitely explore CocoaPods and Google Maps API after the session to better compare both the APIs.

1. Background Updates and Push Notifications

When I started working on the app, I did not think about this background update and how the user will be informed about the location but when I was testing the app, I found this to be a necessary feature. Though this was beyond the scope of proposed application, I decided to implement it. I found background location updates are easy to implement by just adding one line of code to existing code I had already written. Also, I had to update info.plist file to request user for background location updates.

As for the Push Notifications, initially I found it difficult to explore solutions, which were for iOS9. Finally, I was able to implement it by using old swift code and modifying it as per current development environment.

1. Auto Layout

I started the design of this application with a single aim – To have UI as simple as possible. This design idea made me use minimum widgets on each of the scenes. Though there were few widgets to work with, I still faced challenges to implement Auto Layout. Finally, I was able to pin all the widgets and got Auto Layout to work.

1. Storing Data using NSUserDefaults

I have used NSUserDefaults before to persist String data type but never used to store any other data type. I assumed that I can persist any data in NSUserDefaults and I could not be more wrong. I was not able to save CLLocation data. Later I discovered that I could encode CLLocation data to NSData to save data in NSUserDefaults. Similarly it can be decoded to original form for retrieval.

App Limitations:

1. App requires user to add proper address to get desired location. Currently, it chooses the first location information that is provided by the server and ignores the rest. This can be resolved by populating all the options provided by the user and let user choose the desired location from the data. This can be implement in future releases of the application.
2. Currently I have uploaded a single audio file in application bundle but It would be great to add some more audio files and have an option to choose from.
3. As of now, app just stores the location data – address string and coordinates. There could be a feature where user could save the data using some kind of annotation such as home, work, etc.

**SDK and Xcode Limitations:**

Xcode is a powerful tool and it provides large scale of features that are developer friendly and makes app development fast. I would not categorize my comments as limitations but more as suggestions for future releases.

1. I found error description to be cryptic often times. This could be because I am new to this tool.
2. I have been using Eclipse for Java development and I really like how eclipse suggests various options when there is a compilation error. I found that Xcode could improve suggested solutions making developer’s life easier.
3. Suggestions provided during Auto Layout can mess the layout completely. I found sometimes following Xcode’s advice on Auto Layout is more like inviting trouble

Overall, it is great tool for app development

**Overall Experience:**

I enjoyed the whole process of becoming an iOS developer. I found Xcode development environment to be one of the best. I tried my hand on Android Studio for Android development for a month or so and found that the time to run the application on Android Emulator was really discouraging. But I found iPhone simulator to be very fast and responsive. This notion of Android Emulator might be incorrect and is totally based on my initial experience. From a beginner’s perspective, I found interface builder of Xcode to be more powerful than provided by Android Studio.

Overall, I enjoyed learning and developing iOS Apps.

**Appendix:**

**Screenshots**:

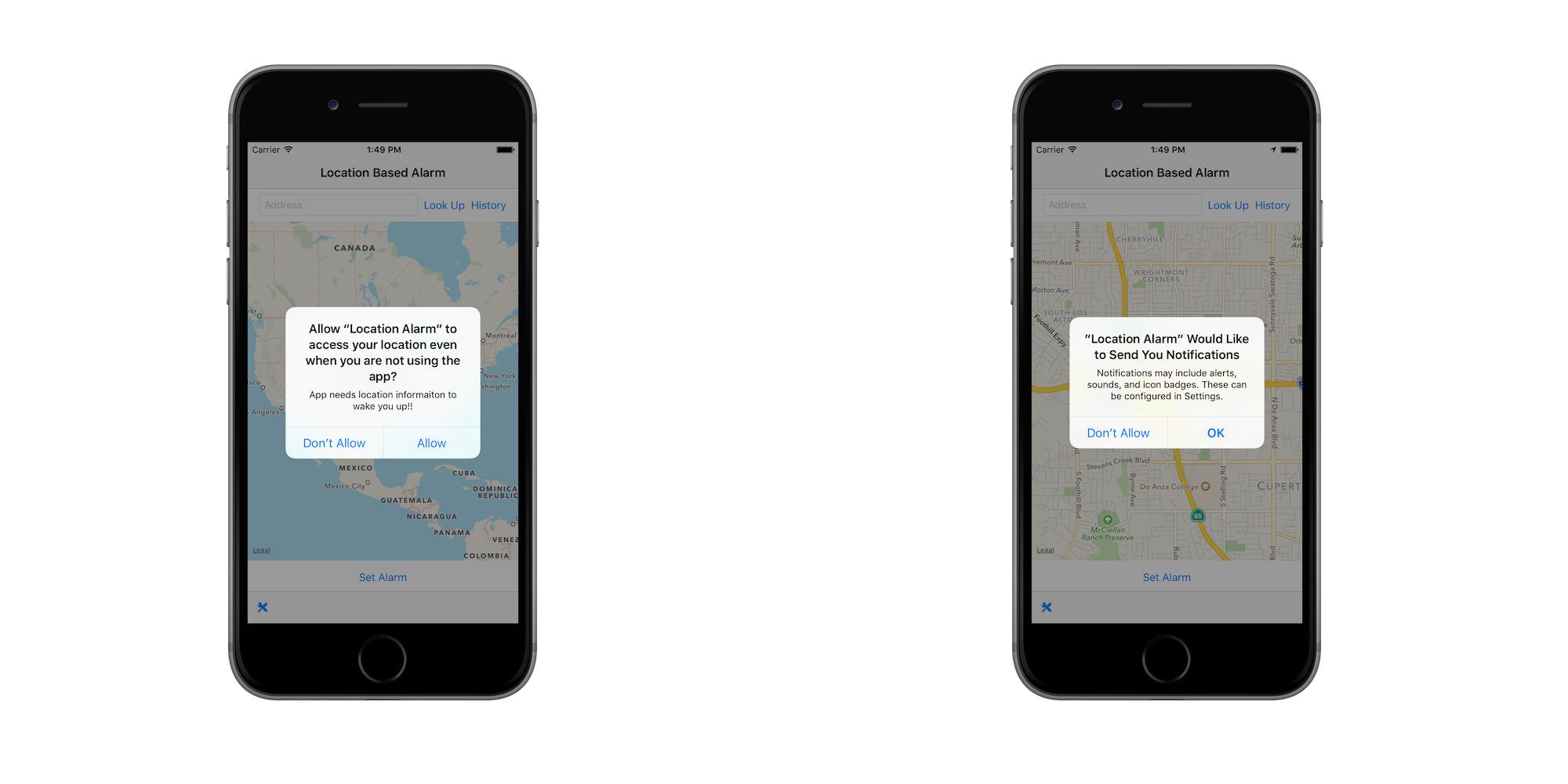


Figure Permission to Access Location and Send Push Notification

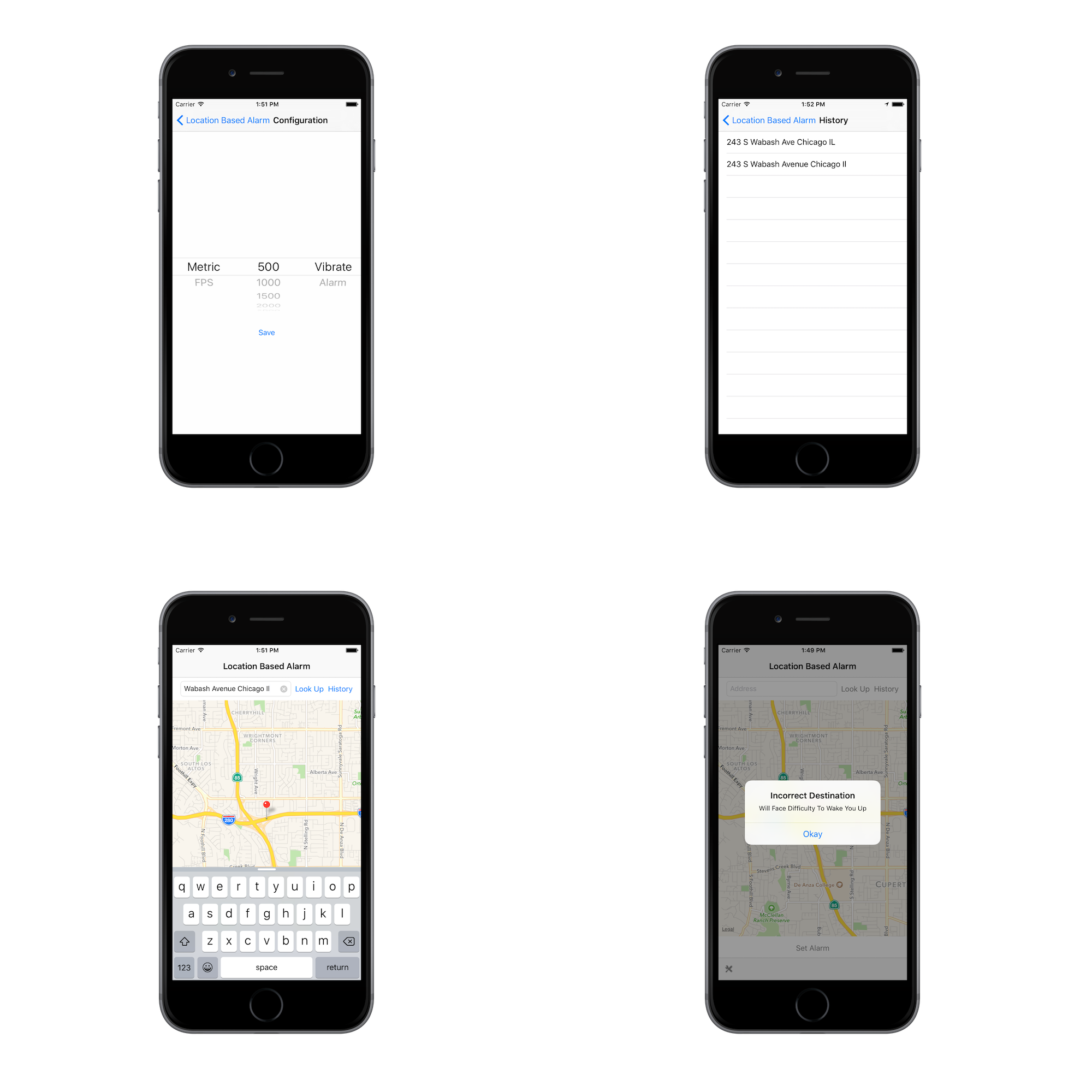


Figure Scenes in the App

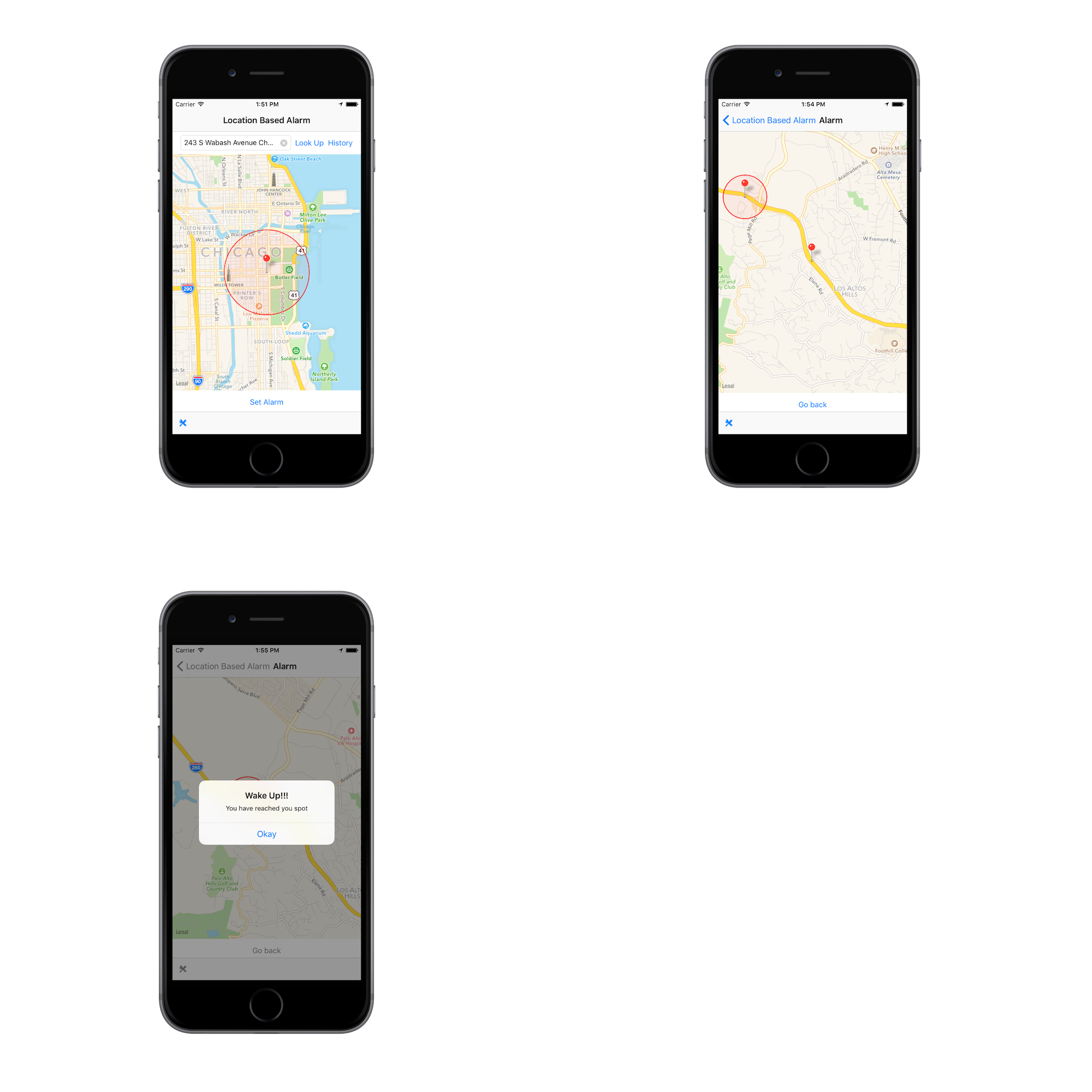


Figure Alarm Status Screens

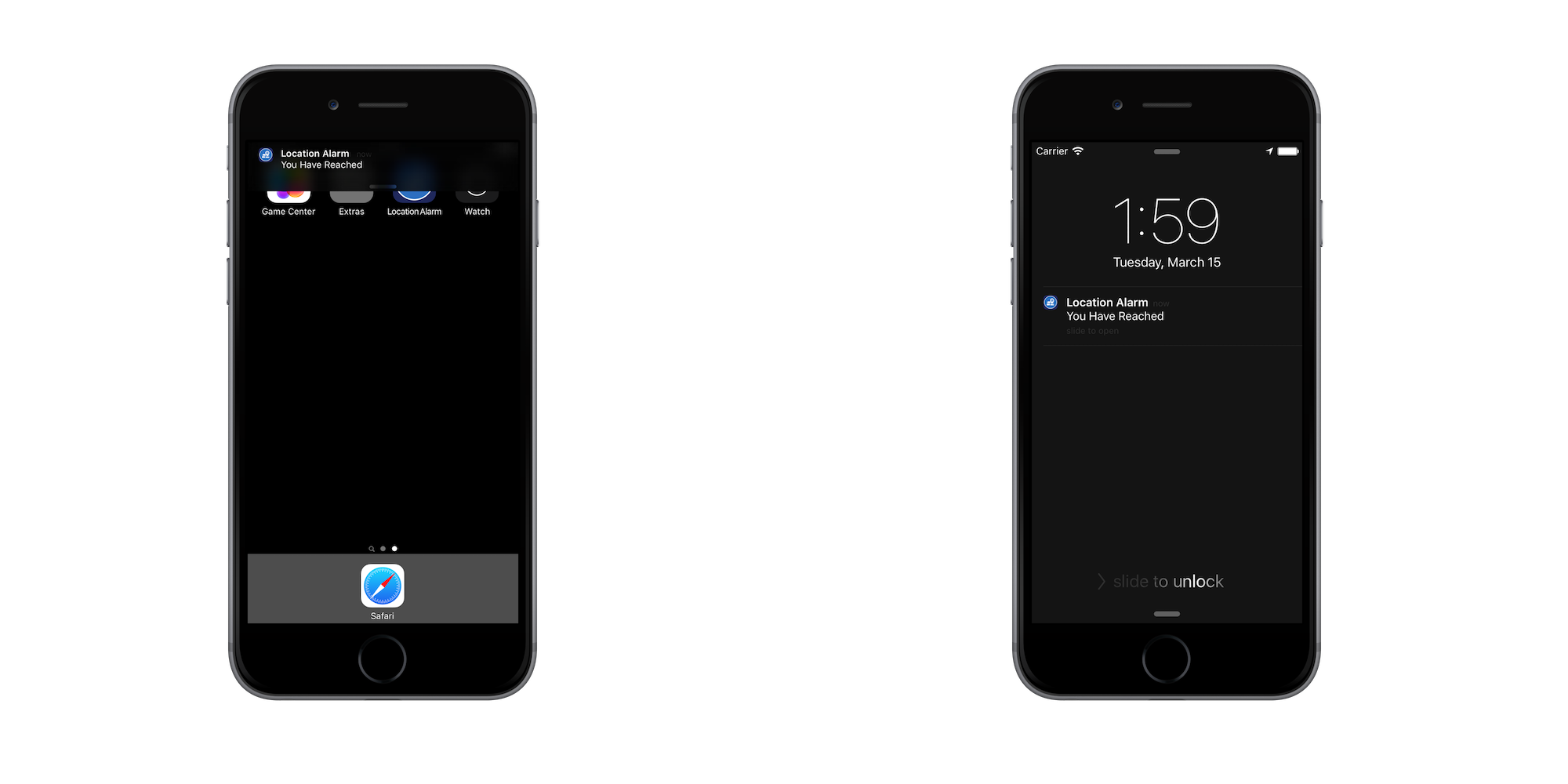


Figure Push Notification - Background and Locked Screen