Criteria	Meets Specifications	Exceeds Specifications (Completely Udacious)
README file		
Does the app contain a README file that describes the intended user experience in the app?	**• The app contains a README that fully describes the intended user experience. After reading the document, a user can easily use the app.** Yes, see README.pdf file in project directory.	*Not Applicable*
Does the README detail all steps needed for the reviewer to build, run, and access the app?	**• The README provides all the necessary information to enable the reviewer to build, run, and access the app.** Yes, see README.pdf file in project directory.	*Not Applicable*
User Interface		
Does the app contain a user interface with multiple pages of content?	**• The app contains multiple pages of interface in a navigation controller or tab controller or a single view controller with a view that shows and hides significant new content.** Yes, the app has a map view standard, satellite, hybrid view, a heat map, heat map animation, time window sliders (customized UIControl), 9 configuration pages (including forms, table views, navigation controller), different annotations and callout views.	*Not Applicable*
Does the app use more than one type of control in its user interface?	**• The user interface includes more than one type of control.** The map view has various controls for loading events, animate events, open callout views, open webcam page, open detailed information, setting different map views (standard, satellite, hybrid),	**• The app contains a customized subclass of UIControl.** Yes, the map view has a time window slider to select from when the events shall be displayed.The left thumb defines “hours ago” / start of time

	setting map content (pins, gaetmap, hybrid), enabled/disabled display of different catageories of events.	window and the right thumb defines the length o fthe time window (in hours).
Networking		
Does the app interact with a networked API that provides the app with data?	**• The app includes data from a networked source.** Yes, events are polled on a configurable schedule from Reuters News, Twitter, USGS Earthquake info, GVP volcanic activity, Apple/MapQuest Geo decoder.	**• The app uses data from more than one networked API.** Yes, events are polled on a configurable schedule from Reuters News, Twitter, USGS Earthquake info, GVP volcanic activity, Apple/MapQuest Geo decoder.
Is the networking code in the app encapsulated into networking classes?	**• The networking code is encapsulated in its own classes.** See the sub groups in the client group. Networking could is encapsulated in client classes.	*Not Applicable*
Does the user interface clearly indicate network activity, particularly in situations where the user is waiting for data?	**• The app clearly indicates network activity, displaying activity indicators and/or progress bars when appropriate, and an alert in case of connection failures.** Activity Indicator are shown when loading the annotations the in the map, the map itself, seeding the world city database info and seeding the volcanic locations. Scheduled data poll, database seeding and data truncation send start/end info notifications to the notification center. If there is any DB error, file error, format error or network error, then an error notification is sent to the notification center. Network error do not interrupt the flow of the application. The data poll is repeated based on the configured schedule. In case of an error also an alert dialog is shown. Only one alert dialog is shown at the same time. Duplicate error messages are eliminated if thrown within 10 seconds to reduce the number of notifications.	*Not Applicable*

Core Data		
Does the app store persistent state in Core Data?	**• The app has a persistent state that is stored using Core Data.** The event, Points-of-interest and location / geo decoding data are stored in the SQLite database via CoreData.	**• The app has a sophisticated data model with three or more entities, and more than one relationship between the entities.** Yes there are 5 entities (Event, DataSource, MapCategory, PointOfInterest, Location) and there are 3 relationships (Event to MapCategory and DataSource, PointOfInterest to MapCategory).
App Functionality		
Does the app function as intended?	**• The app functions as described in the README, without crashes or other runtime errors.** Yes the app does the intended functionality without crashes and runtime errors.	*Not Applicable*

App Specification: Capstone App

iOS Developer Nanodegree

[Note that this is an informal app description. It will give you an idea how the app should work, but when you submit your app it will be rated based on the Rubric.]

The fifth portfolio app is open-ended. Let your own interests guide you. You will invent the app, then design and build it. We hope that you choose an idea that is motivating and challenging.

The app should showcase your iOS skillset. Your app should have a complexity similar to the On The Map app and the Virtual Tourist apps, and should include code from the following areas:
Complexity is similar if not more

User Interface

Your app should demonstrate that you can combine the essential UIKit components in effective ways. It should include the following common UI features:

More than one view controller
yes
A table or collection view
yes (table views for setting options, keywords (add and remove), news topic selection
Navigation and modal presentation
yes (beside the map there is a navigation to 9 different configuration screens and the callout view has a dialog and links to a web browser for the volcano webcam).
Image assets in 1x, 2x, and 3x formats. Or in vector format.
yes (application icons, launch screen logo, annotation icons for volcanoes and 10 magnitudes of earthquakes, webcam icon in callout view) and all in vector or 1x, 2x, and 3x formats.

Networking

Your app should incorporate data from a networked source:

Choose an API and integrate downloaded data into the app
5 different APIs are used (all HTTP based: 3xREST/JSON, 1xCAP/XML, 1xRSS/XML)
Give users feedback around network activity, displaying activity indicators and/or progress bars when appropriate, and an alert in case of connection failures
Activity Indicator are shown when loading the annotations the in the map, the map itself, seeding the world city database info and
seeding the volcanic locations.

Scheduled data poll, database seeding and data truncation send start/end info notifications to the notification center.

If there is any DB error, file error, format error or network error, then an error notification is sent to the notification center. Network error do not interrupt the flow of the application.
The data poll is repeated based on the configured schedule. In case of an error also an alert dialog is shown. Only one alert dialog is shown at the same time.
Duplicate error messages are eliminated if thrown within 10 seconds to reduce the number of notifications.

Encapsulate networking code in a class to reduce detail in View Controllers
See the sub groups in the client group. Networking could is encapsulated in client classes.

Persistence

Your app should incorporate data that needs to be persisted between runs of the app.

Include an object graph that can be persisted in Core Data
Yes (object grap of 5 entities and 3 relationships)
Manage the Core Data Stack outside of your view controllers, either in the App Delegate or in a separate Core Data Stack manager class

The CoreData/DB operations are encapsulated the Repository classes and there is a CoreDataStackManager class setting up the database. For performance reason (3.5million city to coordinate mappings and 1500 volcanic locations with coordinates, image links, webcam links and detailed description) a pre-filled SQLite database is copied from the main bundle directory to the Documents directory (only initially if there is no SQLite database).

Aside from your primary app state, you should find some additional state that can be stored outside of Core Data, either in NSUserDefaults, or in the documents directory using an NSKeyedArchiver

The configuration of the map, the data polling/scheduler, keyword search, news topics … are stored outside of CoreData in NSKeyedArchivers (serialized in files when the application terminates and unserialized when the application starts). An initial default configuration is created.

README

The app should also include a README file within the project's directory. The README should accomplish two goals:

Describe the intended user experience

yes: all the functionality and every page is described.

Include all specific actions and/or commands necessary for the reviewer to compile, run, and access any aspect of the project

Yes: those instructions are include (use Cocoapods), third party libraries are described, default API key for MapQuest is included and can be also configured via UI (see README.pdf file), default Twitter consumer key/secret is provider and can be configured via UI (see README.pdf file). The key will be invalidated after the App review.

To learn more about creating README files, check out Udacity's course on Writing READMEs.

App Functionality

The app should function as described in the README. No crashes or other runtime errors should be evident.

Yes it does.