# Cloudy

## An Android Weather Application by Logan Wright

Cloudy is a web-enabled application built for the Android operating system. On the most basic level, it queries forecast.weather.gov (NOAA) for weather data and displays it to the user.

The architecture allows for a list of locations (referred to as MetaLocations) to be saved and displayed. This data persists between instances by using the SharedPreferences public interface to read and write data to internal storage.

The program listens for a coarse location from the mobile device to find and (if necessary) add the current location to the list, in addition to showing saved locations. The nearby location is marked with a pinpoint icon next to the location text.

Below is a brief description of some of the noteworthy components in the application.

### CloudyUtil

This is a utility class used primarily for creating and receiving HTTP requests, handling MetaLocation list conversions, and handling weather AsyncTasks.

When saving instance state, this class converts MetaLocations to JSON strings. When restoring a state, it expands the JSON back into an ArrayList.

When a weather data request is made, this class processes the request and, upon completion, makes use of the observer pattern to notify listeners that the MetaLocation has been changed.

One important aspect of this static class handling the AsyncTasks is with concurrency. The tasks are directed to run parallel to enable faster returns. This class caches current tasks to prevent duplicate tasks from running concurrently.

### MainActivity

This is the launch activity and entry point for the application. It shows the city, state, temperature, and current weather conditions for MetaLocations in a ListView that makes use of a custom adapter to make the information more organized.

The user has the opportunity to do a hard-refresh of data by pushing the refresh icon, and can delete MetaLocations by pushing the trash icon.

The custom adapter is coded to automatically handle MetaLocation weather data refreshes from within the getView() method. Forecast.weather.gov is presumed to feed new data to the program on a 1-hour delay, and the program will check against this first when determining whether to call CloudyUtil for a refresh. The last refresh attempt date is saved, and an automatic-refresh will not occur for the next 15 minutes.

A user is able to select a MetaLocation by pushing it. Doing so will launch the MetaLocation in a new DetailActivity by attaching the location to the Intent.

### DetailActivity

This activity handles showing details for a MetaLocation. It also uses a ListView with a custom adapter. The list shows future conditions per weather period (daily and nightly) for approximately a week in advance. The user still has the option to hard-refresh the data and even delete the location from within this view.

This activity has an alternate view layout for landscape mode where the list orients to the right side of the screen with current conditions on the left. This enables better use of the unique screen dimensions in this orientation.

### SearchResultsActivity

This activity is created when the user performs a search from the ActionBar by selecting the plus icon. The search query is passed along with the Intent, and passed to a separate AsyncTask that calls maps.googleapis.com for geolocation data. Because forecast.weather.gov works with latitude and longitude, the primary purpose of the LocationTask is to grab the name and location of the search query.

When a request is answered, the location matches are converted into MetaLocations and populated in a ListView. When a user selects a location, it is automatically added to the adapter (accessed statically) in MainActivity. The program has rudimentary handling of duplicate locations, and will not allow it to occur by indicating to the user, through a Toast, that the location is already in the list should they attempt to add it.

### Animations

Data changes are sudden, and to help alleviate this and perform more in-line with the Google design specifications, a refresh container is placed above layouts that present weather data. When a refresh is called for a MetaLocation, a refresh notification fades into view and darkens the background elements.

When the weather task is complete, the corresponding views fade the refresh container back out and restore the content view to full opacity. This enables the refresh state to occur as a smooth transition.