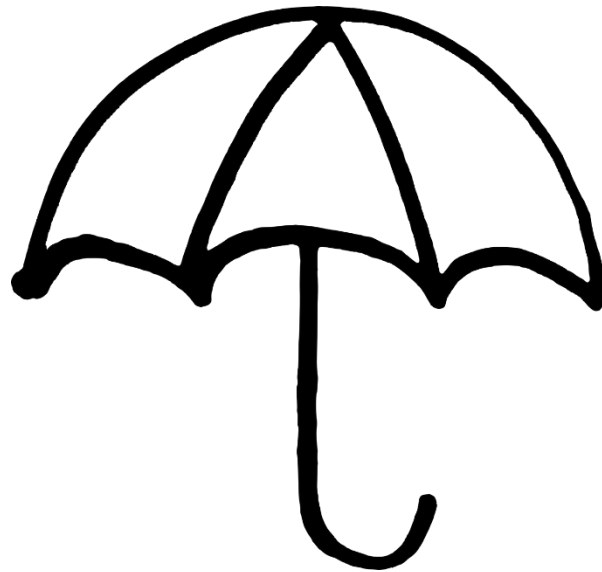


Group 8  
BrainStorm Team



## Software Design Document

*Noah Brayer, Dennis Donoghue, Chad Klinefelter, Kevin Moriarty*

# 1. Overview

The goal of this project is to develop an app for checking the local weather combined with entertainment value through the power of memes. A given meme will be displayed based on location input that is referenced through the NWS SOAP servers for weather input. These weather conditions are fed to the DANK algorithm which references the meme database and returns a meme to our ~~front-end servers to be sent to the user for display. The DANK algorithm is then adjusted using user feedback.~~ This document contains the design plans for the system and its interactions.

The memes are returned from the database server directly to the user's android phone. We have also eliminated user feedback in the final design.

# 2. Design Considerations

The issues we have addressed, some resolved and some ongoing, while completing this design have been plentiful. They mainly stem from a sort of Mother-issue, being our collective lack of experience, which tends to birthing multiple issues that would be simple for an experienced team. This is by far the largest influence upon our design. It pushed us into developing for Android as we are more familiar with Java as a language and will be able to communicate issues among ourselves more readily in addition to knowing where to go online for answers.

How true this section was, our very assumption that Android ran Java as opposed to having its own specialized SDK that was not necessarily backwards compatible with pure Java. We also branched into programming with PHP for our database and server-side functionality as well.

We had to decide on a front-end service and decided on PARSE.com. Issues that may affect the design process going on are those encountered when attempting to interface all of our individual parts. Between the user's device, front-end, weather services, and our database. We may have to use a different front-end service for example.

All front-end development and service occurred/occurs on the Android device. We ultimately chose Namecheap.com for our hosting server, a standard PHP web server with cPanel access. Interfacing the individual parts of the application did prove to be an issue that has affected the design process, as we were unable to integrate our Java SOAP Client with the Android SDK even when provided with custom libraries.

The major constraints on the system are the ~~hardware and~~ software capabilities of Android capable devices, our time window for the project, and our lack of experience.

The hardware capabilities of Android capable devices was not an issue here. We did not design anything that would test the limits of any Android phone, tablet, or TV. Our lack of experience was most certainly a design issue, and the software capabilities of Android devices proved a design constraint as described above.

As such we cannot program a particularly complex system, one from the ground up, or one that is as efficient as a professional design and we must use a design that relies on already available services with our work relying on parsing the data and linking everything together.

The system will be designed particularly with modern mobile phones in mind and as such will be modeled using design criteria for android applications. The application must also interface with networking/wifi software, in addition to the previously mentioned NWS SOPA servers. As such the app will use existing **Java Android** libraries for wifi and mobile network connection as well as SSL/authentication when connecting with weather services.

**Authentication is limited to our back-end database server. Weather services are public and do not require any form of Authentication to make SOAP requests. We are using a Google API key specific to our app, for location-related services.**

### 3. Architectural Design

#### 3.1 Overview

User Interface - Must clearly and concisely communicate the weather for the hour ~~and the forecast for the day~~ with a meme.

**Provides the user with the forecast for the next 6 hours instead of the day**

Weather Database Interface - Must quickly and accurately get the weather data from the weather databases. Must translate it into data that is usable by the Algorithm.

**Interface ultimately determined to be an inappropriate description, final design is not an interface but rather a method that retrieves weather data based on the ZIP code inputted. As integration has not yet been achieved currently the ZIP is hardcoded.**

Algorithm - Must take the data from the Weather Database Interface and calculate the weather score for the Meme Database Interface.

**This is less an algorithm and more a means to select an image from a database that accounts for any weather conditions. It simply tests for certain keywords in the forecast and combines that result with the current temperature.**

Meme Database Interface - Must take the weather score from the algorithm and fetch a meme corresponding to that score. ~~Must ensure that the same meme is not being used too often.~~

**The weather score is simply the name of an image file, so it is trivial to select that image file from the database. The app just requests the image with the URL that the server gives it.**

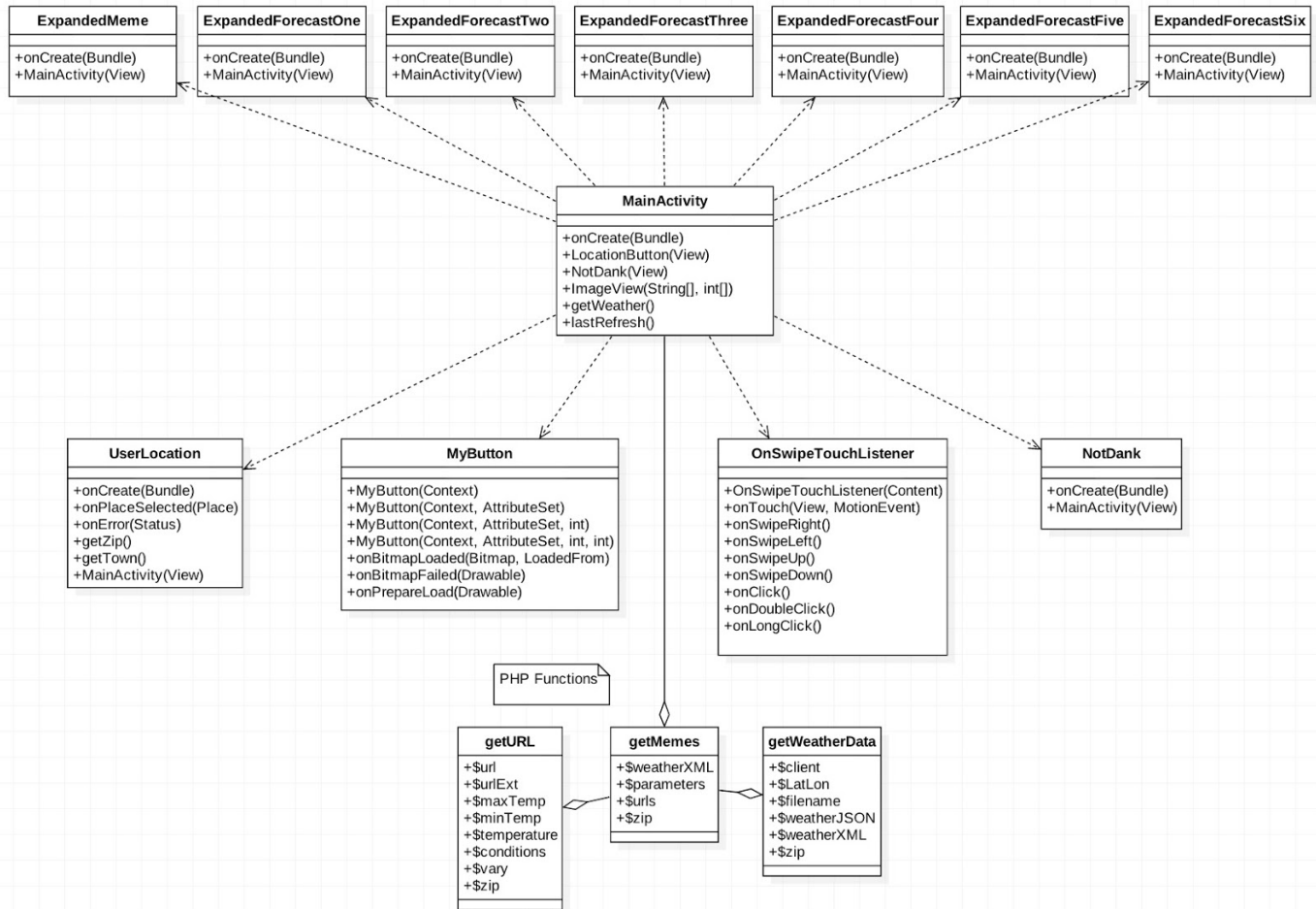
#### 3.2 Rationale

The driving force behind our decision to develop in Android is its ubiquity and ease of use. Android phones represent the largest market share of any single phone architecture. Android Studio, the official IDE for android development, is free and compatible with programming in java. Android Studio will allow us to develop faster and will allow us to respond more flexibly to changes in user specs.

**Our assertion that Android is compatible with programming in Java was slightly incorrect as this is not always the case, as described earlier in this document. Android Studio was ultimately a fast and flexible method of development however not as much as we had originally thought. The IDE does not have its ins and outs documented with the same extensivity that Eclipse or IntelliJ does, with the latter's paid version reputedly being better for Android development, as it is the platform Android Studio is designed from.**

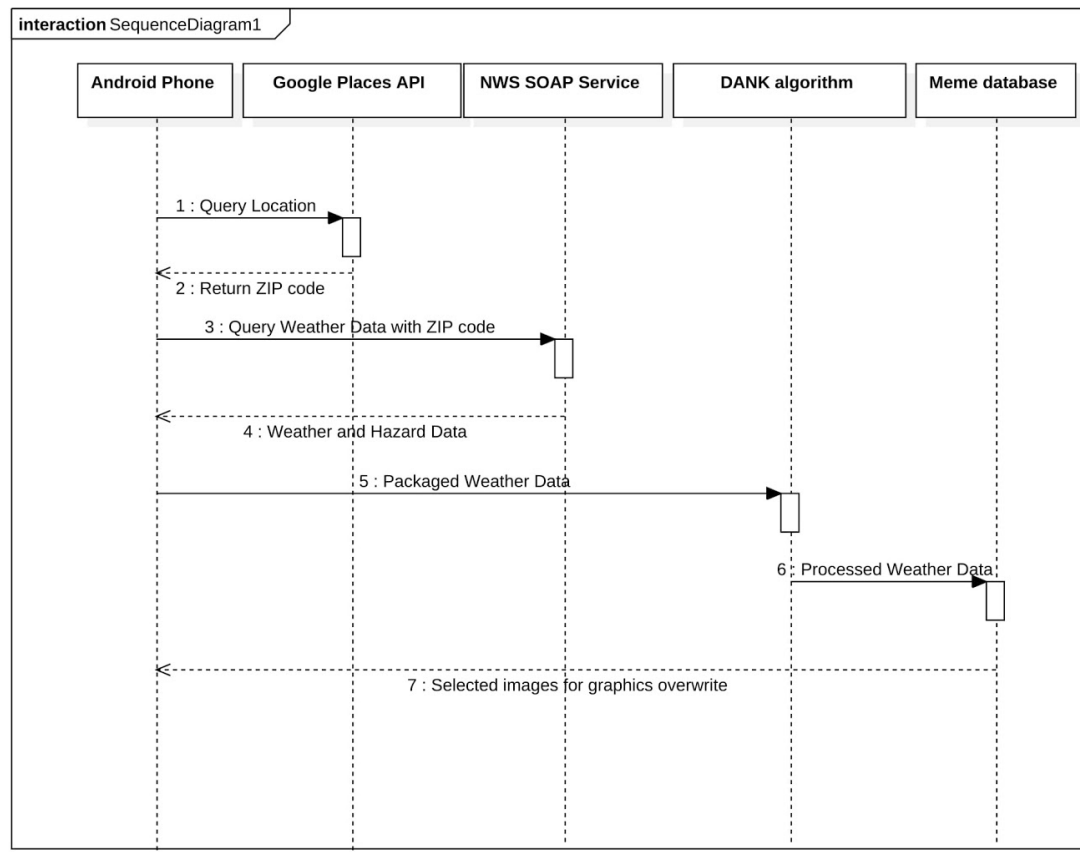
## 4. Low Level Design

### 4.1 Class Diagram



Class Diagram outlining relationships between all components of the application.

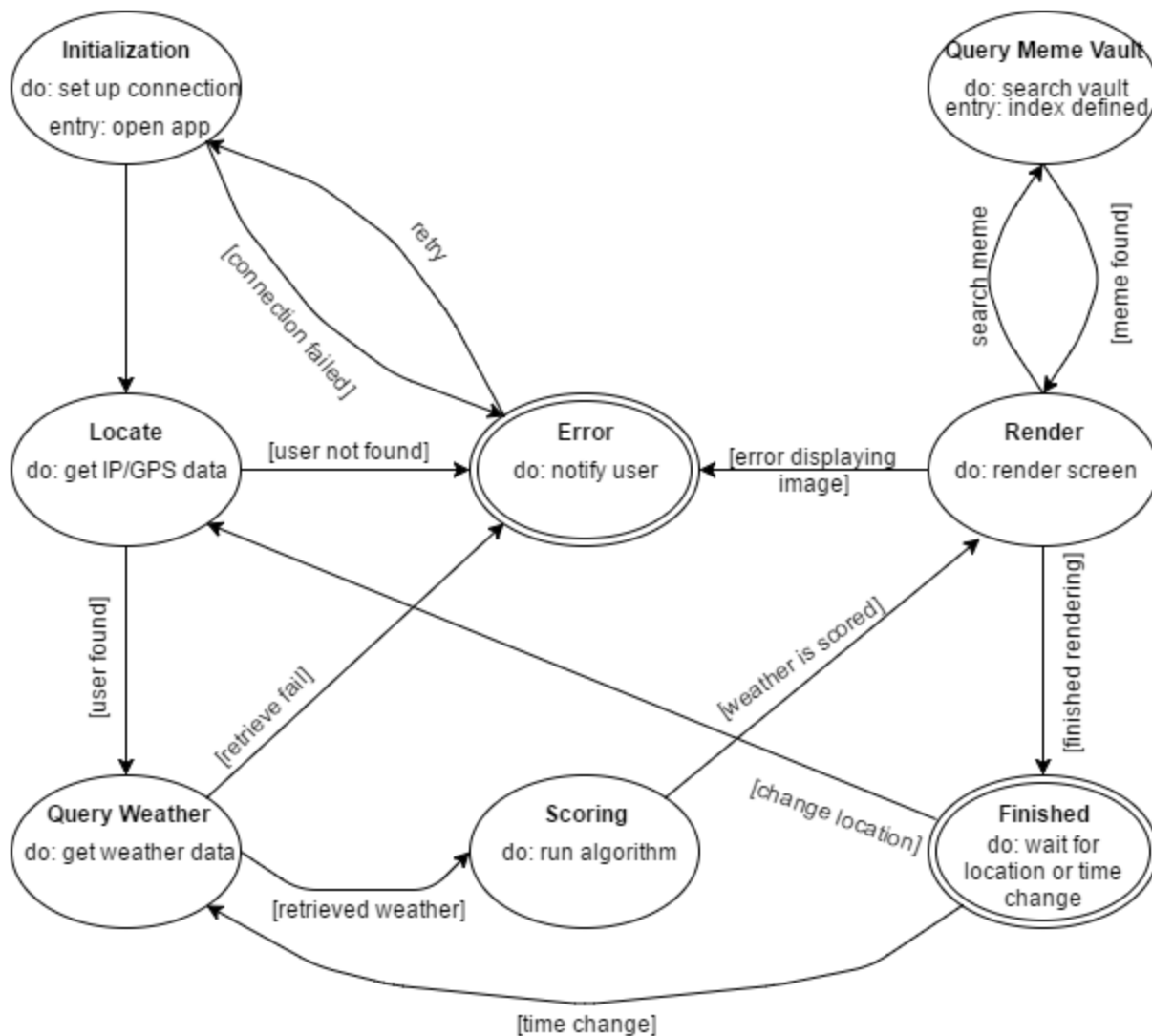
## 4.2 Sequence Diagram



Sequence Diagram for standard weather functionality.

### 4.3 State Diagram

## State Diagram



State Diagram showing lifecycle of object classes.

## 5. User Interface Design - Mockup Below (v0.1)

We will be delivering Meme Doppler as an Android App. This in mind, we will be using Android Studio, the official IDE for Android platform Development. Android studio supports UI changes based on screen size and pixel density, and we plan to use these features, but we will be developing with a specific goal. We are

developing for our application to run best in portrait orientation on smartphones. We do not currently plan to support landscape or in app orientation shifts. We also do not currently plan to develop with tablets in mind.

Example: First Time User: Panes are on the last page, and are read left to right, with the final pane being all the way right and the last step having no pane. Higher resolution panes are highly recommended and are available here [https://drive.google.com/open?id=0B2Si\\_b9bP8g4bmFTRk4ya2dXUkk](https://drive.google.com/open?id=0B2Si_b9bP8g4bmFTRk4ya2dXUkk).

1. Having Downloaded and Installed our App, the user opens up MemeDoppler and is immediately send a request to use location data. ~~If they accept, we will use their GPS location data. If they do not accept and have an internet connection, we will use their IP data to determine location, which is less accurate. If for some reason we cannot acquire any location data, the User will remain in this state, minus the request pop-up, until they manually enter location data (see step and pane 6).~~
2. Having received the Location data through IP or GPS, the Central Image, current weather meme, has loaded, as well as first ~~two~~ six forecast memes. The current weather is the first meme processed and downloaded in our code, and loads in once it is ready, filling an empty pane (loading image possible in the future).
3. The user tapped the central image, which triggers it to expand to fill the screen. Previously queried weather data is displayed here to give the user a more informative experience. A single tap returns to the former screen.
4. In the ~~background~~ six panes below the main pane, the rest of the forecast images have loaded.
5. ~~After an hour, we have an automatic refresh.~~ If the queried weather data differs from the previous forecast, new memes are generated and delivered. If not, the memes shuffle forward and a single new meme is generated.

Automatic refresh is not enabled, weather data is queried when the app is accessed to ensure constant updating.

6. The user then decides to manually enter a location, in the form of a zip code. He has entered this screen by tapping ~~anywhere on the search bar~~ the current displayed location.
7. It turns out the user entered the same location delivered by their GPS location. After querying the weather server for data, the scored data is checked against the current data and have been no appreciable changes in score.
8. ~~The user looks outside, and notices the current meme is very appropriate for the weather at his location. He also finds the meme entertaining, and taps on the "Dank" green area or swipes right. His user feedback is recorded and the data is sent about the current meme to the meme database.~~
9. ~~After a sudden and severe change in weather with no corresponding change in forecasting, the user is frustrated and swipes left or clicks on the "Not Dank" red area. This loads the User Feedback Decision Pane. The user can select the meme they find appropriate for the current weather.~~

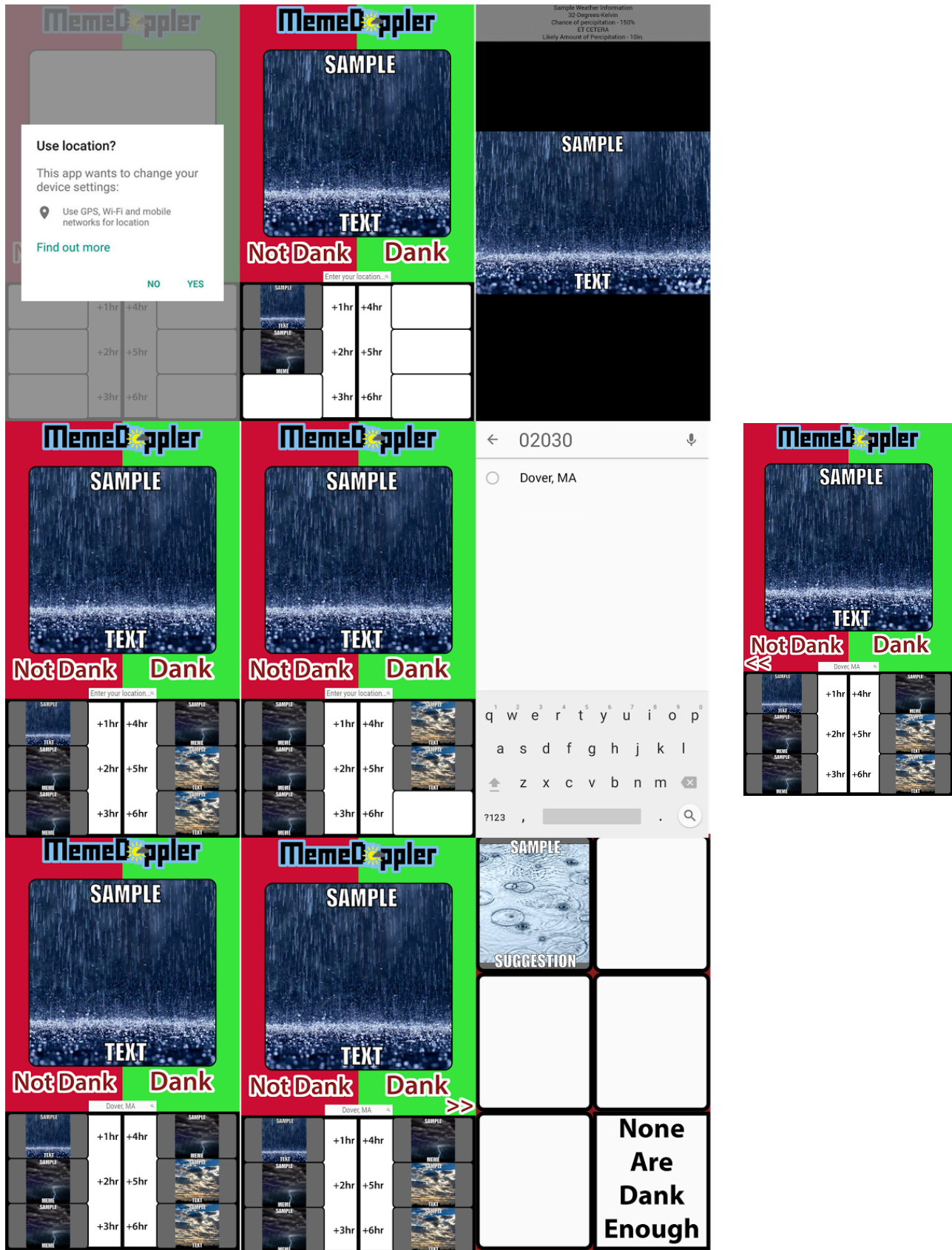


- ~~10. Having finished giving feedback, the user is returned main screen, which has been updated to reflect the given rating.~~

User feedback has been removed as a feature with the final design. It may be implemented in future versions

11. This has showcased the planned features of this app. EVERYTHING is subject to change and revision, but the general gist of the flow of the app should remain the same.

Ultimately, this statement has proved accurate, and we have experienced change and revision since our first design. However, the final design still bears semblance to it.



## 6. User Interface Design - Mockup Below (v1.0)

Again, left to right and top to bottom. Not all described within is as of yet implemented.

1. If your internet connection or device is slow, you will see the loading images.
2. Then, based on the current location, images load in from the database.
3. This user is entering a new location after clicking the location button.
4. The images have been updated to reflect the new location and forecast.
5. Having clicked on the main image, the user sees the image expanded and is given a small amount of specific weather information. This works for all the images on the main screen.
6. If the user swipes left or clicks the button on the left, the Not Dank user feedback screen pops up.
7. After giving user feedback, the user is thanked.
8. The user then swipes up or down, refreshing the images. Notice the change in last updated.
9. Having clicked on the 4 hour forecast, the image has expanded and specific information is given.

MemeDoppler

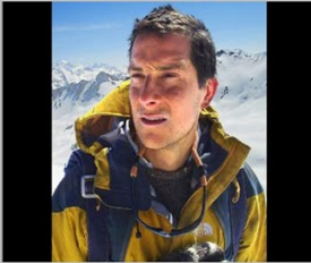
MemeDoppler

Loading Images

current date and time

TEST

MemeDoppler



Last updated: 2016/12/14 23:20:26

AMHERST, MA

MemeDoppler

MemeDoppler

MemeDoppler

1hr

2hr

3hr

MemeDoppler

MemeDoppler

MemeDoppler

4hr

5hr

6hr

MemeDoppler

MemeDoppler

MemeDoppler

1hr

2hr

3hr

MemeDoppler

MemeDoppler

MemeDoppler


4hr

5hr

6hr

MemeDoppler

Sample Weather Information  
32-Degrees-Celcius  
Chance of Percipiation - 98%



Last updated: 2016/12/14 23:24:08

BOSTON, MA

MemeDoppler

MemeDoppler

MemeDoppler

1hr

2hr

3hr

MemeDoppler

MemeDoppler

MemeDoppler


4hr

5hr







6hr

MemeDoppler

Sample Weather Information  
32-Degrees-Celcius  
Chance of Percipiation - 98%




Replacement Memes



NOTHING DANK

MemeDoppler

Sample Weather Information  
32-Degrees-Celcius  
Chance of Percipiation - 98%



Last updated: 2016/12/14 23:36:09

AMHERST, MA

MemeDoppler

MemeDoppler

MemeDoppler

1hr

2hr

3hr

MemeDoppler

MemeDoppler

MemeDoppler


4hr

5hr

6hr

MemeDoppler

Sample Weather Information  
32-Degrees-Celcius  
Chance of Percipiation - 98%



Last updated: 2016/12/14 23:38:09

AMHERST, MA

MemeDoppler

MemeDoppler

MemeDoppler

1hr

2hr

3hr

MemeDoppler

MemeDoppler

MemeDoppler

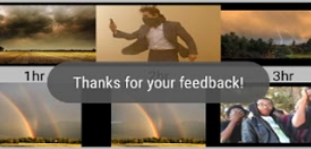
4hr

5hr

6hr

MemeDoppler

Sample Weather Information  
32-Degrees-Celcius  
Chance of Percipiation - 98%



Thanks for your feedback!

MemeDoppler

MemeDoppler

MemeDoppler

1hr

2hr

3hr

MemeDoppler

MemeDoppler

MemeDoppler

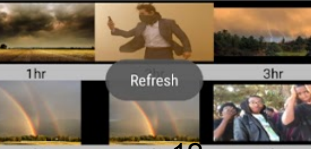
4hr

5hr

6hr

MemeDoppler

Sample Weather Information  
32-Degrees-Celcius  
Chance of Percipiation - 98%



Refresh

MemeDoppler

MemeDoppler

MemeDoppler

1hr

2hr

3hr

MemeDoppler

MemeDoppler

MemeDoppler


4hr

5hr

6hr

MemeDoppler

Sample Weather Information  
32-Degrees-Celcius  
Chance of Percipiation - 98%



MemeDoppler

MemeDoppler

MemeDoppler

1hr

2hr

3hr

MemeDoppler

MemeDoppler

MemeDoppler


4hr

5hr


6hr

MemeDoppler

Sample Weather Information  
32-Degrees-Celcius  
Chance of Percipiation - 98%



Replacement Memes



Bo

Boston  
MA, United States

Bolton  
MA, United States

Boylston  
MA, United States

Boxborough

q w e r t y u i o p

a s d f g h j k l

z x c v b n m

7123 , .