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 SOPA 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.

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.

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.

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.

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 of 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 libraries for wifi and mobile network connection as well as SSL/authentication when connecting with weather 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.

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.

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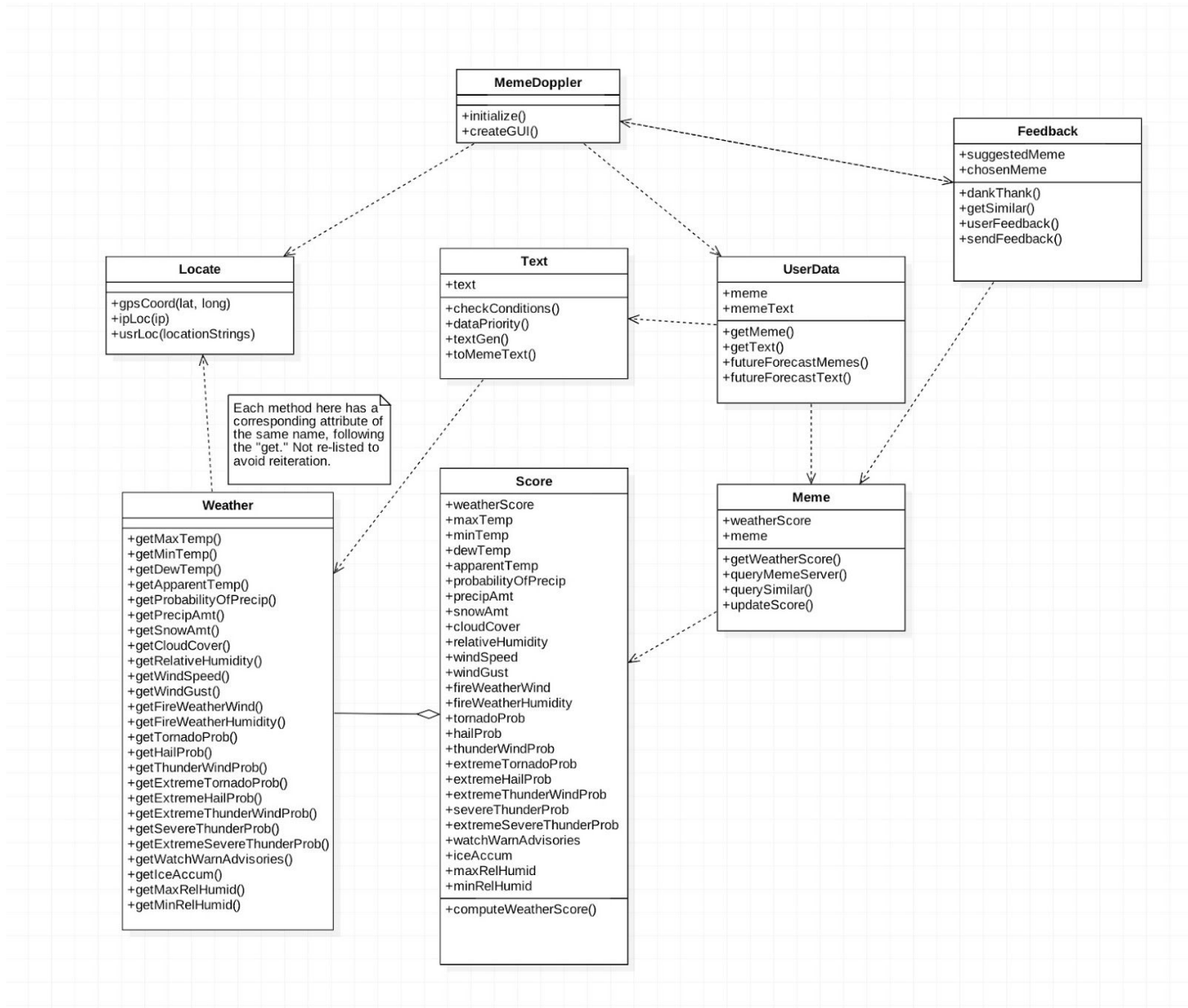
Meme Database Interface - Must take the weather score from the algorithm and fetch a meme corresponding to that score. Must ensure that the same meme is not being used too often.

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.
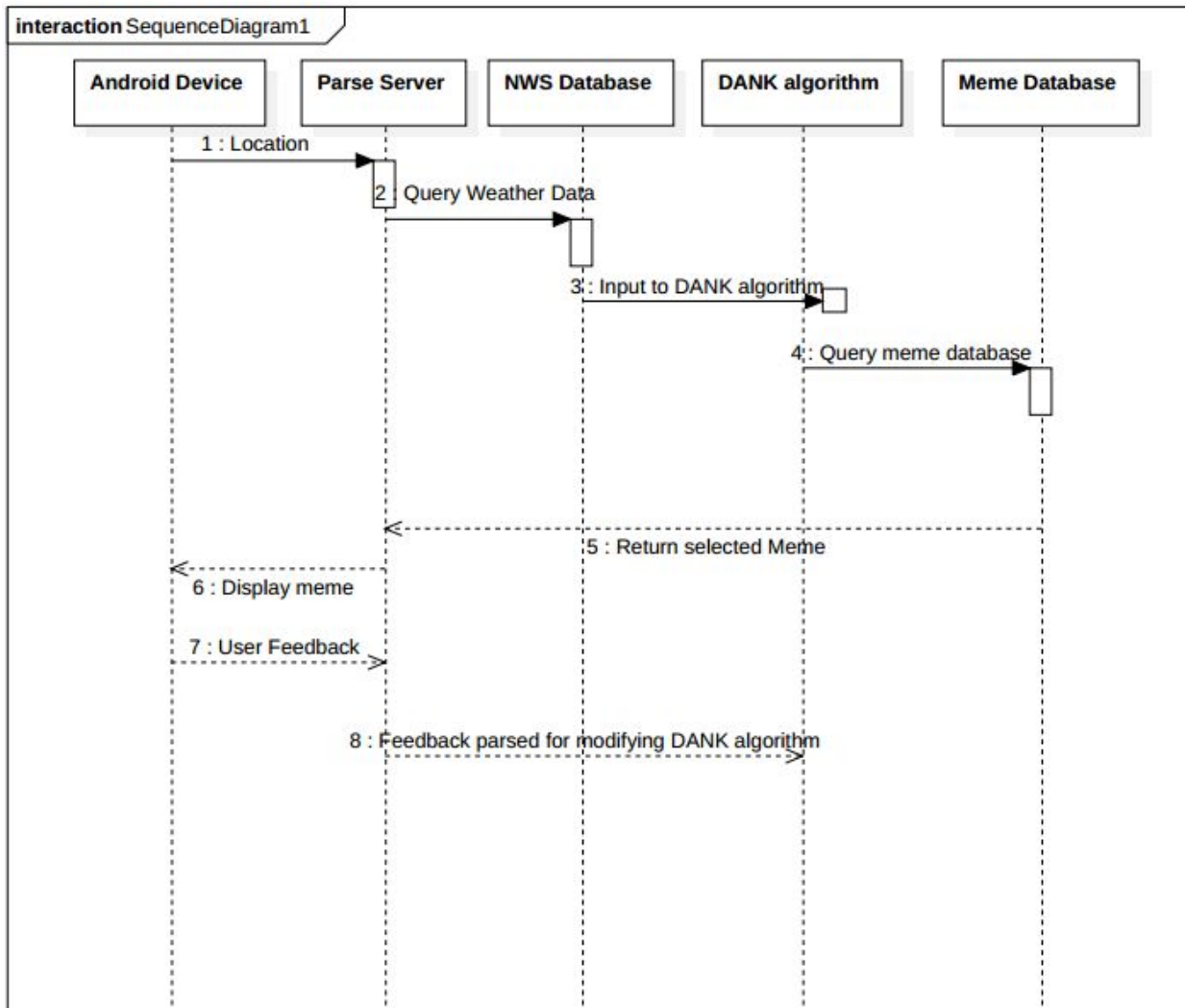
4. Low Level Design

4.1 Class Diagram

MemeDoppler

+initialize()
+createGUI()

Feedback

+suggestedMeme
+chosenMeme

+dankThank()
+getSimilar()
+userFeedback()
+sendFeedback()

Locate

+gpsCoord(lat, long)
+ipLoc(ip)
+usrLoc(locationStrings)

Text

+text

+checkConditions()
+dataPriority()
+textGen()
+toMemeText()

UserData

+meme
+memeText

+getMeme()
+getText()
+futureForecastMemes()
+futureForecastText()

Each method here has a corresponding attribute of the same name, following the "get." Not re-listed to avoid reiteration.

Weather

+getMaxTemp()
+getMinTemp()
+getDewTemp()
+getApparentTemp()
+getProbabilityOfPrecip()
+getPrecipAmt()
+getSnowAmt()
+getCloudCover()
+getRelativeHumidity()
+getWindSpeed()
+getWindGust()
+getFireWeatherWind()
+getFireWeatherHumidity()
+getTornadoProb()
+getHailProb()
+getThunderWindProb()
+getExtremeTornadoProb()
+getExtremeHailProb()
+getExtremeThunderWindProb()
+getSevereThunderProb()
+getExtremeSevereThunderProb()
+getWatchWarnAdvisories()
+getIceAccum()
+getMaxRelHumid()
+getMinRelHumid()

Score

+weatherScore
+maxTemp
+minTemp
+dewTemp
+apparentTemp
+probabilityOfPrecip
+precipAmt
+snowAmt
+cloudCover
+relativeHumidity
+windSpeed
+windGust
+fireWeatherWind
+fireWeatherHumidity
+tornadoProb
+hailProb
+thunderWindProb
+extremeTornadoProb
+extremeHailProb
+extremeThunderWindProb
+severeThunderProb
+extremeSevereThunderProb
+watchWarnAdvisories
+iceAccum
+maxRelHumid
+minRelHumid

+computeWeatherScore()

Meme

+weatherScore
+meme

+getWeatherScore()
+queryMemeServer()
+querySimilar()
+updateScore()

Class Diagram outlining relationships between all components of the application.

4.2 Sequence Diagram

interaction SequenceDiagram1

Android Device	Parse Server	NWS Database	DANK algorithm	Meme Database

1 : Location

2 : Query Weather Data

3 : Input to DANK algorithm

4 : Query meme database

5 : Return selected Meme

6 : Display meme

7 : User Feedback

8 : Feedback parsed for modifying DANK algorithm

Sequence Diagram for standard weather functionality.

4.3 State Diagram
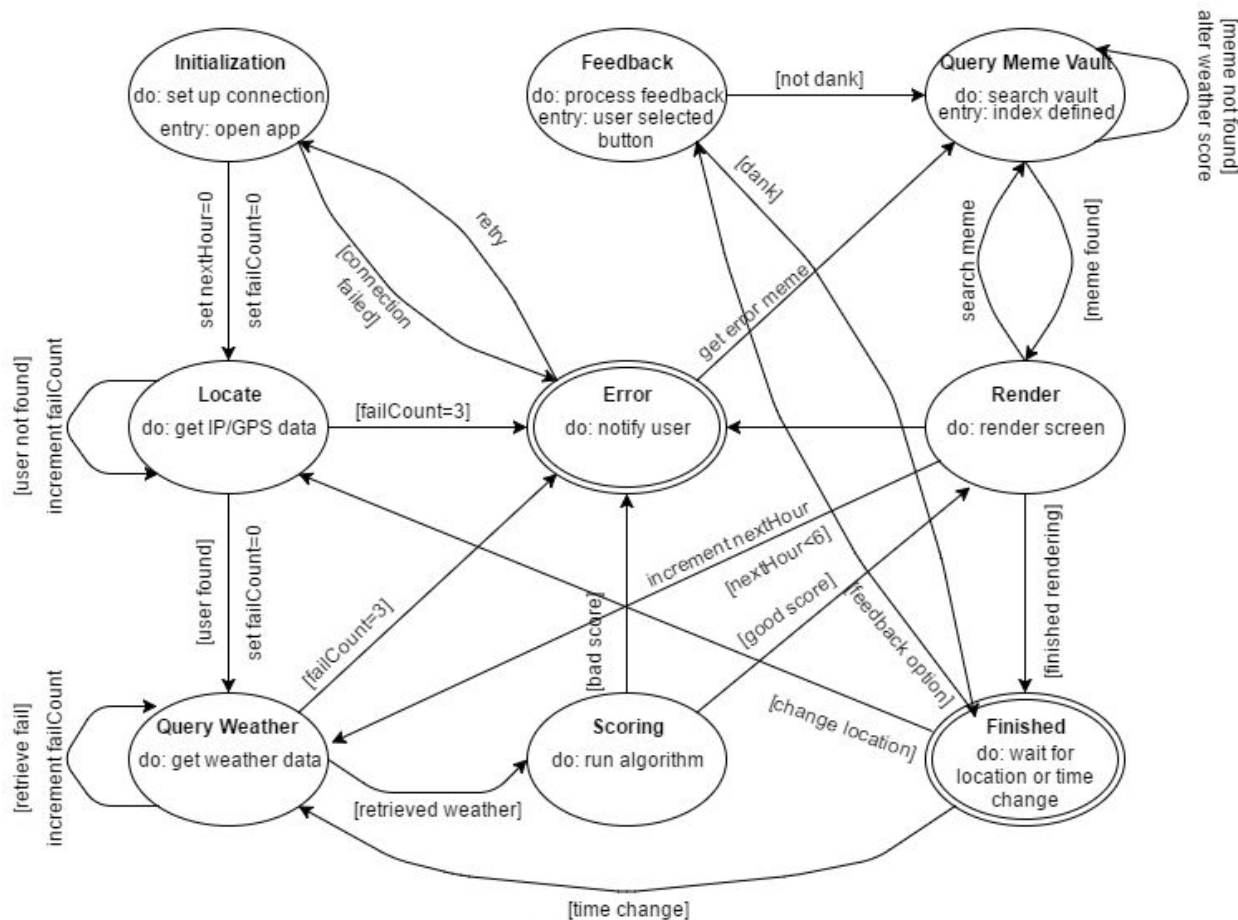
State Diagram

State Diagram showing lifecycle of object classes.

5. User Interface Design – Mockup Below

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.

1. Having Downloaded and Installed our App, the user opens up MemeDoppler and is immediately send a request to use location data. If they accent, we will use their GPS location data. If they do not accent 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 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, 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.

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.

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.

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.