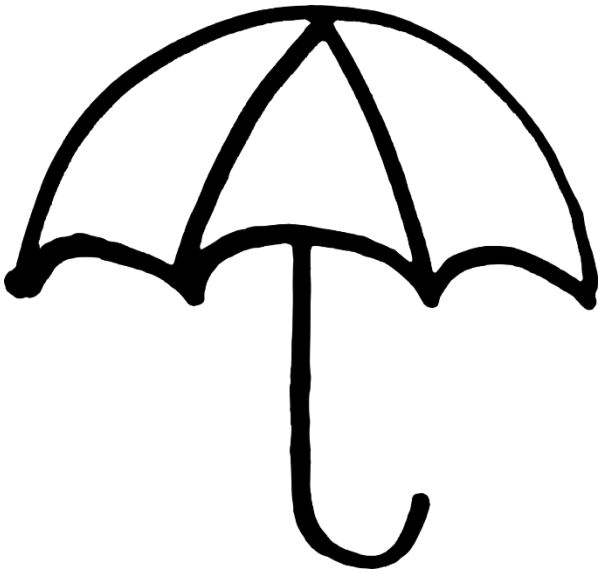
Group 8

BrainStorm Team



MemeDoppler

Process Plan Document

Noah Brayer, Dennis Donoghue, Chad Klinefelter, Kevin Moriarty

Process Plan for Meme Doppler

1. Introduction

The Meme Doppler platform aims to combine the functionality of a weather app with the entertainment of meme culture given the pervasiveness of the meme through the millennial generation.

The primary scope is to provide local weather from reliable sources filtered through the DANK algorithm to produce a memed output as well as allowing users to affect the DANK algorithm to identify the most appropriate meme for a given set of temperature/weather conditions. The software will also alert the user of any emergency weather alerts in their area.

The software will not be a weather database aggregate, that is it does not care about analyzing the actual weather data from its sources, just interpreting it to be output as memes for the end user.

1.1 Definitions and Acronyms

EOF - End Of Finals
DANK - Definitely Appropriate Noteworthy Knowledge
STASH - SToring And Sharing Hoard
~~PEPE - Project Execution and Patching Errors~~
GPS - Global Positioning System
IP - Internet Protocol

2. Process Description

2.1 Project Lifecycle

During the design, implementation, and maintenance of our project, our team will have to sit down and reevaluate the direction, design, and development of our growing product. **This will hopefully happen more during the beginning of our journey**[1], but will continue all the way through to the end of our product's lifestyle. With this in mind, **we will use a lifestyle that will start most closely resembling Iterative, and will later transition to an Agile lifecycle as the project matures**[2] and we enter the maintenance and feature development stage.

1: The majority of our reevaluation in direction and design occurred closer to the end of the development process. Particularly during testing and integration, after initial coding had occurred.

2: We started with a more traditional waterfall style development and transitioned into an Agile lifecycle around the latter portions of the design stage, with the hallmark continuous change and testing being implemented from the architecture sub-stage onwards. Most of the development was implementation and testing, however we did have to reevaluate our architecture at a few points, most notably when it was discovered we could not integrate our SOAP client with our main Android program.

2.2 Process Activities

1. **Locate User**
   Acquire user's location from GPS or IP address.
   We implemented this using the Google Places API

2. **Acquire Relevant Weather Data**
   Query weather database for current data.
   We accomplished this by using SAAJ (SOAP with Attachments API for Java) to make calls to the NWS SOAP server and get back weather data as XML data. We are looking at using a REST API in the future as it is more modern and compatible.

3. **Generate Weather Score**
   From the current weather, an algorithm will determine the weather score.
   From the weather data, we get the temperature and current weather conditions. Several regexes select keywords from the weather conditions (e.g. "snow", "rain", "fog") and summarize them in a maximum of three characters. The temperature is broken down into five main sections: negative temperatures, between 0 and 30 degrees, between 30 and 60 degrees, between 60 and 90 degrees, and 90+ degrees. These sections are summarized with a single character. These two summaries are concatenated and the result is the "score", which is the name of an image file.

4. **Get Meme'd On**
   Select the meme corresponding to the weather score from the meme STASH, ~~as well as the three other closest memes.~~
   Ultimately the activity only pulls the current appropriate meme from the meme database, we did not manage to implement pulling the three closest memes. However memes representing the forecast for the next six hours are pulled and displayed below the meme for the current hour.

5. ~~**Prepare Meme**~~

~~Generate text to be added to the meme. This will be related to current weather conditions.~~

We scrapped this activity as we had no time to implement it being as behind schedule as we were.

6. **Render Meme**

   Send weather data and the selected meme ~~and text~~ to user.

   (Currently, we are fetching a hardcoded set of memes from the database and inserting it into the required panels. Our downloading and inserting code works as it will in the final version, but we will have to hardcode until we can receive unrequested information from the database.)

7. ~~**Gather User Feedback**~~

   ~~Offer two buttons with "DANK" or "Not DANK" for users to choose. If DANK is chosen, then the meme is appropriate. If Not DANK is chosen, then users will be offered several options as to why the meme was not appropriate, such as inaccuracy, offensiveness, or outdatedness. After selecting their option, they will be offered the three other closest memes to select which would be more appropriate, as well as a "None of the above" option.~~

   Feedback has been effectively scrapped for the final design as we do not have the time to implement it.

3. Roles

3.1 Roles Table

Role	Responsibility
Project Manager	Coordinate communications between leads. Coordinate meetings and set goals for individual leads.
Requirements Lead	Set reasonable goals for development of entire project based upon current progress.
Quality Assurance Lead	Ensure that requirements are being met without errors. Test real code from development engineers. In charge of PEPE.
Design Lead	Brainstorm ideas to be implemented and organize layout of the interactive application. Analyze current product needs and plan further development. Responsible for design of new features.
Implementation Lead	Take input from requirements and design branches and feedback from QA to form pseudocode framework.
Full Stack Developer	Take pseudocode from implementation lead and translate

	into real code.

3.2 Role Assignment Table

Team Member	Role
Noah Brayer	Requirements Lead / Full Stack Developer **(PHP)**
Dennis Donoghue	Implementation Lead / Full Stack Developer **(Android)**
Chad Klinefelter	Project Manager / QA Lead / Full Stack Developer **(Android)**
Kevin Moriarty	Design Lead / Full Stack Developer **(Java)**

4. Estimates

Estimated lines of code: Approximately 8000; our software will use a bevy of existing resources for the actual gathering of weather data, doppler map, etc. Therefore our code will not be as robust as that of a program which independently provides these functions.

Actual lines of code: **Approximately 2000; we did use existing resources for the actual gathering of weather data and location services, however the doppler map was not included in the final program. The actual code for the main app and its various activities was written by Chad and Dennis. As such our code was still robust, especially since we did not utilize a remote self-contained separate API to gather weather data but rather the code implementation of a Java library for SOAP requests.**

Estimated defects: We expect a total of a few (2-3) hundred defects with the majority being small errors that are easily rectified with several larger defects which take multiple tens of man-hours to rewrite or work around.

Actual defects: **We were very nail-on-the-head with this estimation. Ultimately the individual small bugs for each group member were on the 50-100 scale, and we had/have a few larger bugs particularly during integration that have taken multiple tens of man-hours to rewrite or work around. This may have been the most realistic estimation in our entire project.**

Task	Estimated Effort (person-days)	Start Date (DD/MMM/YY)	End Date (DD/MMM/YY)	Person	Actual Effort (person-days)
System Design	8	25SEP16	20OCT16	D, N, K, C	15
Detailed Design	14	20OCT16	20NOV16	D, N, K, C	15
Code Input Module	8	20NOV16	05DEC16	C,K	6

Code Output Module	8	20NOV16	05DEC16	D, N	6
~~Code Map Module~~	~~12~~	~~01NOV16~~	~~30NOV16~~	~~N, K~~	
Code Framework	10	10NOV16	05DEC16	D, N, K, C	10
Test Planning	3	25NOV16	05DEC16	D, N, K, C	3
Testing and Integration	5	05DEC16	EOF	D, N, K, C	10
Rework and Final	3	12DEC16	EOF	D, N, K, C	5

Ultimately the process of creating our app was complicated and we learned a lot from it. Looking back on what we would do differently, we would have gone with a lot more pre-packaged code from the start in the form of APIs as we ended up doing for location services. Particularly in the case of obtaining weather data, we would have used a REST web API from the start that got the data for us which we would then have piped directly to our server and algorithm. Although we discovered that Java and Android are not the same thing, we would still choose to develop for Android, as it was a challenge and a rewarding experience.

We would have scrapped the idea of a front-end server from the start, since each Android device handled it's own front-end tasks, with everything back-end on our servers or those of the NWS.As mentioned we would have gone with REST over SOAP for data aquisition as it is more modern compared to SOAP and would be more compatible with modern code. We would have still attempted to implement feedback since with a better direction from the beginning there would be time to achieve full functionality with the design. Implementation of text being added to the memes however would have been scrapped were we to do it all over again, as this would have been very hard to integrate with feedback without a large amount of manual curation.

As for more general aspects of the design process we most certainly would have begun coding earlier. As it was the general intimidation of a task this size and lack of experience led to a partial paralysis that was not broken until we actually sat down and coded. While we would have ultimately taken around the same amount of time, starting earlier would have been of great benefit and very feasible knowing what we know now, that the task is not as insurmountable as we once pictured it.