Group 8

BrainStorm Team

Test Plan Document

Noah Brayer, Dennis Donoghue, Chad Klinefelter, Kevin Moriarty

1. Introduction

This document outlines testing procedures for the Meme Doppler application. The testing implemented is with respect to getting data from our server. We are doing so via PHPUnit. **At the time of this documentation (12/14), our project was set up in Travis CI, but test cases were not fully functional.**

2. Quality Control

2.1 Test Plan Quality

Quality is ensured by testing multiple conditions on our core weather and meme functionality. This extends to our three main server functions: getMemes, getWeatherData, and getURL. When the app calls getMemes, this function gets the weather data based on zip code, returns this, and then gets the URL for the respective image. Our test cases must reflect that every weather combination we have accounted for will be able to generate an image. Our test plan is successful if this is possible.

2.2 Adequacy Criterion

Define the termination criteria for your testing activity. In other words, define how to decide when testing is done. Typically, this involves some form of branch, statement, or path coverage.

3. Test Strategy

There are eleven different weather criteria that we evaluate to choose the image. Additionally, we have five different temperature ranges. All of these different conditions must be tested to ensure functionality for all weather parameters.

- Ice/hail/freezing all invoke the ‘i’ criteria.
- Snow invokes the ‘s’ criteria.
- Thunderstorms invoke the ‘t’ criteria.
- Rain/showers invoke the ‘r’ criteria. This is only invoked if i, s, or t are not invoked.
- Cloudiness/overcast invokes the ‘c’ criteria. This is only invoked if i, s, t, or r are not invoked.
- Haze/fog invokes the ‘h’ criteria. This is only invoked if i, s, t, or r are not invoked.
- If none of these conditions are invoked, the implicit condition is ‘clear.’
- Temperature ranges are >90°F, between 90°F and 60°F, between 60°F and 30°F, between 30°F and 0°F, and <0°F.

Effectively, the best way to test this in real time is to pick a varying set of locations, query their weather conditions, and check that the server outputs the same conditions for those locations.

4. Test Cases

Case Number	Case Purpose	Actual Result
1	**Data for Portland, ME 04101**	**http://memedoppler.com/androidServer.php/?zip=04101**

2	**Data for New York, NY 10001**	**http://memedoppler.com/androidServer.php/?zip=10001**
3	**Data for Tallahassee, FL 32301**	**http://memedoppler.com/androidServer.php/?zip=32301**
4	**Data for Fargo, ND 58102**	**http://memedoppler.com/androidServer.php/?zip=58102**
5	**Data for St. Louis, MO 63101**	**http://memedoppler.com/androidServer.php/?zip=63101**
6	**Data for Kansas City, MO 64101**	**http://memedoppler.com/androidServer.php/?zip=64101**
7	**Data for Austin, TX 73301**	**http://memedoppler.com/androidServer.php/?zip=73301**
8	**Data for Los Angeles, CA 90001**	**http://memedoppler.com/androidServer.php/?zip=90001**
9	**Data for San Francisco, CA 94101**	**http://memedoppler.com/androidServer.php/?zip=94101**
10	**Data for Portland, OR 97201**	**http://memedoppler.com/androidServer.php/?zip=97201**

After meticulously reviewing the xml files we receive from the NWS and comparing it to the results of the tests, we can confirm that our server and the DANK algorithm functions correctly.