Team Chikorita

Alex Aberman – ama4tf

Joshua Angeley – jda5dp

Patrick McGee – pjm5gk

RainyDay

Pitch:

Have you ever checked the weather and realized it was raining, then wondered what to do with your miserable existence of a day? Well fret not! Thanks to RainyDay, you'll never face indecision in your life again\*. Simply start up the app then turn on GPS or if you're worried about the NSA knowing where you are, just input the city you're in, then press “Update Weather”. Instantly you'll be notified the application is finding out the very information you couldn't discover unless you stepped outside. Just press the button one more time and the application will tell you the 5 day forecast and light up your fortunately attached light strip with blue and white for rain or clouds and bright orange for sun. But there's more! If the weather doesn't look too bright, RainyDay will tell you available movies running using RottenTomatoes helpful API, so you can avoid the rain and support your local theater. Have any questions? Forget them and embrace the present!

\*No guarantee of resolving your existential crises

Key Features:

* GPS integration
* OpenWeatherMap for the 5 day forecast
* RottenTomatoes calls for currently running shows
* Magic

Basic Instructions for Usage:

1. Power up the app
2. Select the IP Address input text and change it to the ip you want
3. Turn on GPS or enter the city you want to learn about
4. Press “Update Weather”
5. Wait 5 seconds for processing
6. Press “Update Weather” again
7. Be dazzled by beautiful icons, helpful numbers, and sweet movie recommendations
8. Faint from sheer splendor

How we use the lights:

The lights will shine blue and white if it's raining. They used to flash, but due to the limitations of using the regular python code, the flash looked less like a cool streaming effect and more like a rave in slow motion, so we disabled the flashing. When it's sunny the lights are orange like the sun, because we're original.

How we use the sensors:

We use the GPS to figure out where the user is if they want us to know.

What our third party web service is and how we use it:

We use OpenWeatherMap to get the weather data for the next 5 days. We were going to use Yelp however there is some issue with our tablet in that all Yelp calls had a timecode 18 seconds off from Yelp's clock, so it refused to handle any of our requests. We switched to using RottenTomatoes for current movies being screened since nothing goes together better than movies and rainy days. Except maybe umbrellas and rainy days.

Any special info you need to run the app:

Make sure Google Play Services is enabled on your device before trying to use the GPS

Links to APK:

<http://plato.cs.virginia.edu/~jda5dp/rpi/>

Lessons Learned:

Android programming is hard. Everything that could have proven nigh-impossible made itself so.

1) Android Studio is very slow. We collectively lost about 4 work hours total due to Android studio starting up, compiling, stalling due to running out of processor space. This does not include the time spent for installation, time spent finding print statements in logcat, picking optimal emulators before realizing they wouldn't work with Google Play Services, and fixing failed merges due to Android Studio. We believe the services offered by Android Studio do trump those of Eclipse for mobile development, but the product is not ready for any fulltime development.

2) Researching API's heavily before committing to them will radically improve the experience. When Yelp failed on us, we had to scramble to find an alternative. There were easier API's, but getting them to combine with our Weather usage which was working just fine was the problem. So testing the API usage first, since that's the most likely to break, would be more useful than work on the sensors or wireframes.

3) Android version issues in testing aren't well documented. There were aspects of our app that one member thought worked fine, but it was only after another member downloaded the code and tested it did we discover that it only worked for the first member because their device was on OS 4.1.2 whereas our device was on 4.0.4. And there is no solution that we've found. Which just goes to show, don't count it done until you test it on the device you're presenting with.

4) UI multithreading needs to be taught. This is the biggest fault in our application: You have to trigger the action twice. We had enough troubles with API handling that getting background tasks to update the UI was not at the front of the queue of tasks. And being able to do this is critical for anyone learning Android development, which is a shame we didn't have time to go deep into it, but we understand it's a matter of creating a UiThread.

5) Relative placement of graphic objects is awesome once you figure out how it works. Initially we were hardcoding the location of textViews and editTexts, not really knowing how to use layouts. Around milestone 3 we figured out that organizing things by layouts would make the app a lot more logical in structure and also easier to organize relatively.

6) Lists instead of Tables would be handy. We got really good at using tables, however we learned that most apps use listViews instead, since you can alter the number of elements within easily, whereas tables are restricted in space allocation. This worked fine for the structure of our app, however it was not an extensible solution at all.