Bobby Darian, Curtis, Ronald Manganaro, Paul Karcher

Prof. Chen

Ubiquitous Computing

05/04/2016

Project Report: Perfect Form/Running Buddy

Project Goal:

The purpose of this project was to encourage inexperienced runners and new fitness enthusiasts to run more regularly. In order to achieve this goal we provided feedback on both the runners form as well as the runner's pace. Both of these features helped the target audience to focus on the basics of running and improve on them at their own pace. The point of giving the user the weather at the start of the app is to help the user find a good time to fit in their run. This is important since one of the biggest hurdles to becoming a routine runner is picking a time to run that works for your schedule and will make for a more enjoyable run. There's also a light icon on the app that will tell the runner they should find a better place to run if the lighting is not ideal. This is added since many runners find themselves too into the run to consider simple safety conditions.

The next part that we focused on was determining if the runner was running at a decent pace. Doing some research we found that going too fast can cause injury which is likely to deter the average person from continuing to run routinely. It is also important that you keep a steady pace so that you end up burning more calories. For these reasons we found a pace of between 160 - 200 steps per minute to be the ideal pace. 180 steps per minute is the ideal pace according to many publications including sites like runnersworld.com. This is an achievable pace for new runners and is unlikely to cause injury to the runner.

The last part that we focused on was how to determine if the runner was in good form. We did not use any sort of machine learning for this though it would have been ideal. Unfortunately none of us had any experience in the area. The way that we came up with metrics for “good” versus “bad” form was simply to graph a experienced runners accelerometer data next to a relatively inexperienced runners data. The basic idea was to make separate graphs of the X,Y, and Z data and then determine what the max and min values should be at anytime. By looking at all these graphs we came up with min and max values allowed for x, y, and z data points. The app basically records all the data into separate arraylists and checks the data points every minutes to see if values exceed the max and min allowed. Points are then deducted depending on how many time the use violates the condition over the course of a minute. If the user does not violate any in a minute they will gain 1 point. In the graphs Marc is the experienced runner and I am the inexperienced runner. You can find an example of the data in the repo file DataAnalysis excel document.

Project Features:

Story: Pulling current weather

This component was meant to pull a JSON object from openweatherapi.com. The only thing that this activity would need is a zip code. Once the JSON was obtained it was to be parsed for certain fields.

Story: Schedule run and delay notification

After getting weather data this portion was meant to ask the user when they wanted to run. This activity need to make sure that it built the notification and then set an alarm.

Story: Receive notification and display at appropriate time

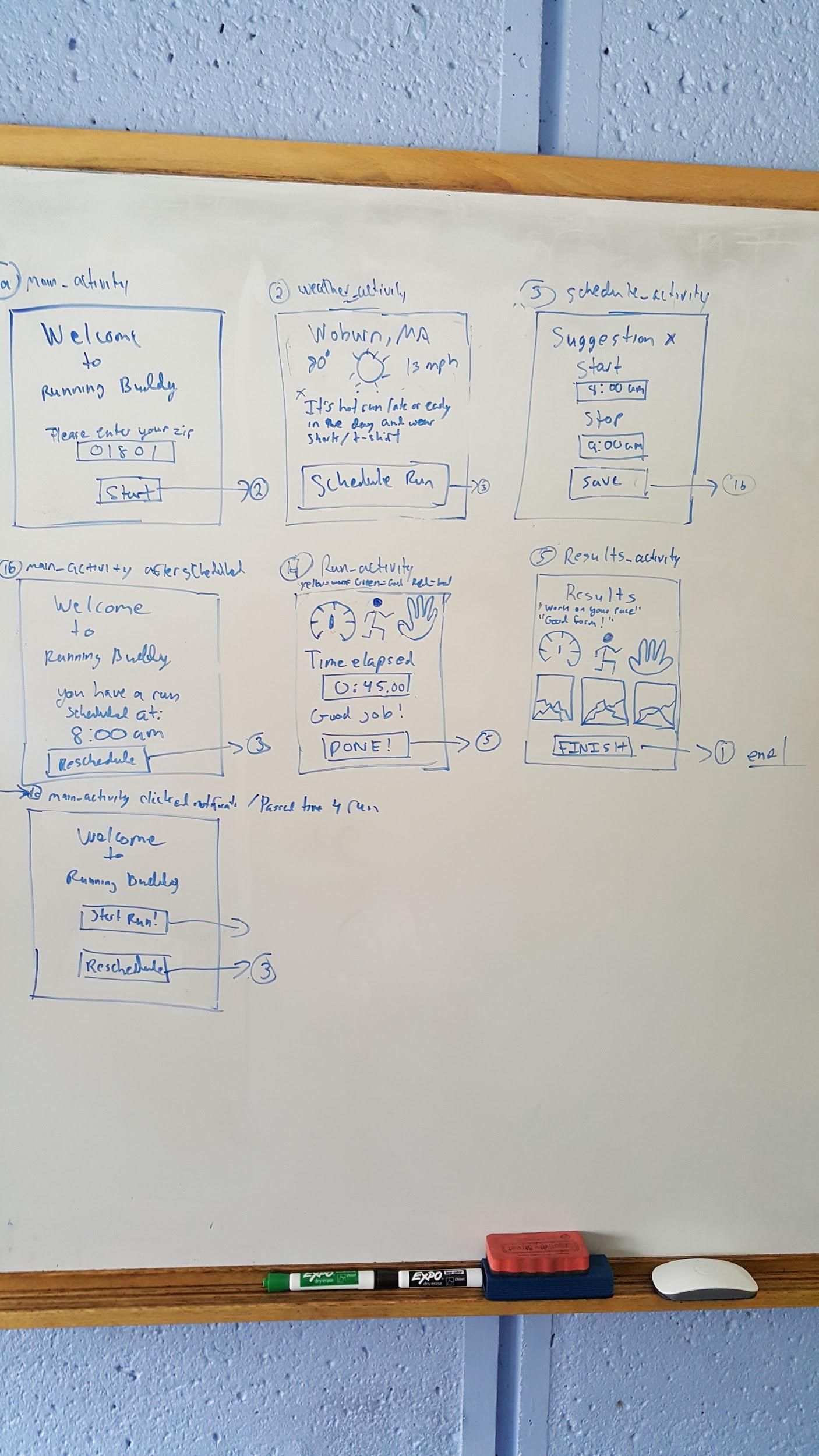
This activity would receive the built notification and display it when the alarm went off.

Story: Collect accelerometer data, light sensor data, and steps/minute and provide feedback

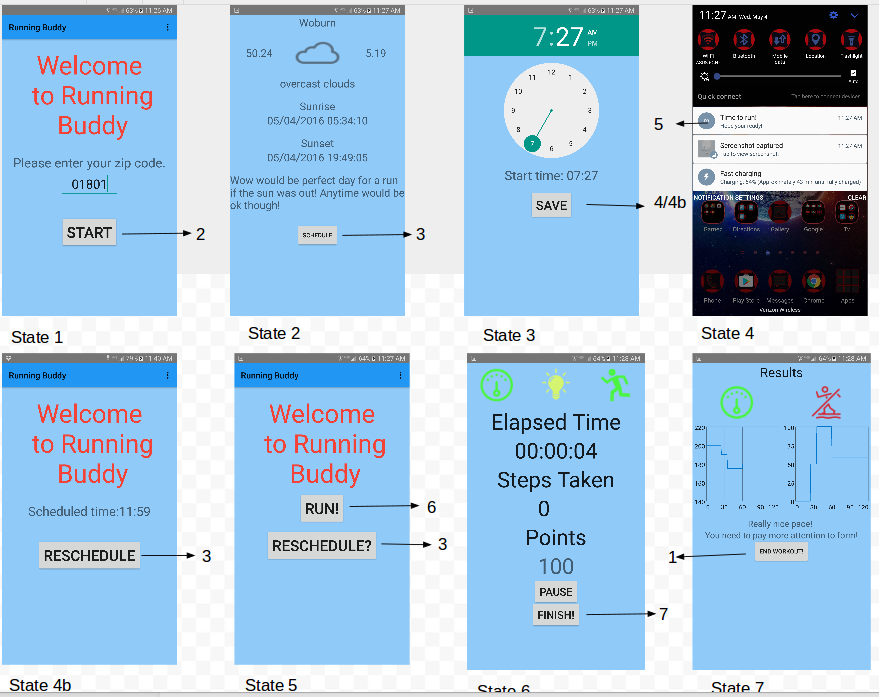
This story ended up being broken down into smaller pieces since we perceived it as more of an epic. We started off making sure we knew how to collect the data and then how to decided what to display the information to the user. At the end it was all put back together into the run activity and extended by the results activity which showed a graph of the results.

Story: Make suggestions based off data collected

This was custom built and integrated into the weather story and the data analysis story in order to provide more than just icon cues as to what the data was telling the user to do.

Project design: visual diagram of project components/flow

Followed the original blueprint very closely



File structure: what does each file do and who wrote that file

Note: Some files included in this project were used for debugging as well we to work on features that we wanted to implement for this reason the files in use are used in the final prototype, but the others were essential to getting the app up and running.

In use:

Ronald Manganaro

MainActivity

This is the entry point of the whole project. The point of this activity is to wait for the user to enter a zip code.

NotificationPublisher

The notification publisher is used to receive the notification built in the scheduleactivity file then notify the user at the correct time.

ReadyToRunActivity

This is a modified version of the mainactivity that will allow the user to click run or reschedule. This activity is what opens when the notification that your run is ready will take you to.

Results

The last screen of the app that will show you a graph of your progress. This includes the pace over time as well as the how your points fluctuated over time.

Run

This is the screen that monitors your accelerometer readings, step count, and the amount of light in the area. The stepcounter will check every minute how many steps have been taken and based on that adjust the speedometer icon to show how you have been doing. The form icon will turn red if you went over a certain number or under a certain number for any of the x,y, and z marks we determined to be out of bounds of good form.

ScheduleActivity

This activity is based off the files Bobby submitted. Was able to take the vales from the time picker and set an alarm based off that time and then display a delayed notification. Bobby submitted a more robust alarm clock that included picking a certain song to play, but there was an issue integrating it.

WaitingActivity

This is another modified version of the mainactivity screen which was displayed anytime the user was waiting on the runtime to occur. They were also given the option to reschedule the run if need be.

Paul Karcher

WeatherActivity

This activity dealt with pulling the appropriate JSON file depending on the zip passed to it. In order to do this the program used a function SendfeedbackJob in order to asynchronously download and grab the data so that the program would not be in a state of hanging while we waited for data to come in, exception handling was also provided in the event of an incorrect zip code or server error. The activity then parses the JSON by using the built-in JSONObject library provided by Google and pulls out temperature, weather condition, wind speed, name of the town, sunrise, and sunset time. The activity then chooses a particular icon depending on the weather condition and will give a suggestion about the best time to run based on this information. This was one of the larger code files for the project and provided a great learning experience to someone completely new to Android programming.

Unused:

Bobby Darian

Alarm\_Reciever

This is what the notification publisher was based off. The file has some other functionality such as dealing with what song the user picked to play.

ScheduleAlarm

I based my schedule activity on this file. This took the time from the time picker in order to set the alarm for the users run. The activity also created an pendingintent for the notification that it built.

Ronald Manganaro

GyroscopeSensor

I made this in order to see what the data from the gyroscope looked like while running but ended up not using it since could not find a good use for it.

LightSensor

Simple test to make sure that the light sensor worked before implementing it into the run activity.

SensorAccelerometer

This was used as the original tool we used to collect data about the users run. It is the only way to export the data collected to a computer. I had a difficult time getting the data from the phone and was only able to get the data by connecting to to android studio and using adb pull.I could not figure out how to store it locally for the user to access.