I went into my project hoping that it would force me to learn how to do more complex programming and problem solving than I had been used to and that I would be able to create a product I could be really happy about. I found that the notion that ideas are sometimes harder to create than the actual product based off those ideas rings true for me sometimes. I originally wanted to create an app that would automate formulas, removing the need to make a spreadsheet or run the numbers manually for things like physics equations and the like (Original App Proposal). The app seemed fairly plausible but it didn't seem all that appealing to people I talked to. It would only be useful to certain groups of people such as students. Even more importantly, however, my app idea didn't really drum up the sort of interest, the "wow factor" or "sex appeal" or whatever you call it, to really gain interest among people (Record of Thinking 1-3). I decided to have some preset formulas to make my app a unit converter of sorts that included the ability to make custom conversion formulas as well as the ability to used the iOS device's hardware, such as the accelerometer for measurements (Record of Thinking 1-3). However, doing that didn't help to make the app feel more interesting. When I talked to actual people some more, I found people were more interested in simple information delivery apps. Lindsey showed me the "Do I need a coat?" app and Brianna suggested an "Inspirational quote of the day" app (Interview Notes). I tried to incorporate these into a comprehensive "information center" with unit converters, weather and other live-updating informational items. I then remembered how I though windows 8's updating tiles were a good idea and an advantage over the iOS idea of separate apps. I decided to incorporate the idea of placing multiple bits of info into one consolidated view to make my app more unique among information apps (Record of Thinking 4). However, I found that I had overloaded on features and create what I called a "747 cockpit" of an app and made writing it within 5 weeks nearly impossible (Record of Thinking 4, Record of Thinking 12/14/12). I decided to significantly trim down the concept to just a clock, calendar, reminders and weather module in order to simplify development and make a cleaner, more minimalist app (Record of Thinking 4). It was here that I finally settled on the idea that the app should almost never require user input and that all the information could be consumed passively as opposed to the more active consumption warranted by the opening of individual apps in iOS (Record of Thinking 4). I finally had an app concept that worked for me and another advantage was that the features could be built up incrementally, which would allow me to remove some of them if I changed the feature list. However, by this point I as already around two weeks into my app development period, putting a strain on how much time I had (October 29th Work Log-Week 2 Work Log).

Once I finalized my proposal, I started working on the clock. I already knew the basics of how to create an NSDate from my labs. The immediate problem that came up was converting the current NSDate into plain english (Record of Thinking 4). I tried something with NSDateFormatter that involved using a string to provide the basis for the formatting of the NSDate (ex. @"MMDDYY"). It didn't work, so I tried something different that, instead of using strings of values to format the date, would use preset styles such as NSDateFormatterLongStyle. I then created a string that would have the date and time in

"plain English" (Record of Thinking 4). I then could feed all the values into an array by separating them based on the spaces in between them. I now had the array with all the time info I needed. It allowed me to use the appending string method to create strings with the time and the date. I then moved on to the calendar. I was expecting a relatively simple fix for the calendar initially, because it seemed like such a logical thing to make available for easy use like the "cal" command in terminal (Record of Thinking 5). However, I found myself manually writing the calendar. I used NSDateFormatter again to get the days of the week. I then had to find a way to enumerate through the next several days and append them into a string (Record of Thinking 4-5). I had to create an array with just the number of days in a month to make sure that the days on the calendar would start from one at the beginning of the month, so that, for example, if it is Dec 31, tomorrow is reported as Jan 1 rather than Dec 32. I also had to feed the days of the week into an array and enumerate through them. This came out relatively well and I was able to tweak the spacing to work properly (New Years Record of Thinking). I still couldn't get the clock and other info to self update. Simply having the method repeat itself gave me a blank view. While I was presenting, Lindsey suggested using an NSTimer, which I was able to read up on and implement fairly quickly (Record of Thinking 12/7). However, I found that as the clock would update, it would update more and more slowly. I fixed this by moving the update method to it's own method that could be called by the NSTimer rather than putting them in the same method. I later found that this was the main fix as the problem was that putting the timer and the code for the clock together would cause the NSTimer to compound upon itself, increasing the delay with each update (New Years Record of Thinking). I also created a background that would change with the time and, aside from a few errors regarding PM/AM and the fact that 12AM/PM is BEFORE 1AM/PM, it worked out fine and proved to be a nice touch and was one of the issues I ironed out towards the end (New Years Record of Thinking). Overall, the clock and calendar portion took much longer and was more challenging than I expected. I think that's because there were not as many "easy methods" for NSDate that would do most of the grunt work for me and so I had to do a lot of things manually. However, I did have the benefit of NSDate and clocks being relatively common so that there were a few tutorials from which I could pull various methods.

I then turned my attention to the calendar events. I found out about EKEventKit pretty quickly from the apple documentation. However, this did have some issues as EKEventKit was not talked about all that often. It wasn't mentioned in our textbook and generally, I had to rely on the apple documentation and Stack overflow forums. Even then, often times the forums would conflict with each other and the documentation, while helpful, didn't have the in-depth explanation that many of the tutorials had (Final Record of Thinking). I had to copy-paste the code, learn what each method did and then use those methods to my advantage. This problem was made worse by the fact that iOS 6 required that the app request permission to use calendar data from the user. I had to rely on stack overflow to get the methods to request permission from the user. (Final Textual Record of Thinking) Eventually, I was able to get everything to work in proper order. I had a few issues with the SIGABRT error because I messed up the numbering system for calling in the values of the array, so I would call for values at indexes that didn't exist (Final Textual

Record of Thinking). Overall, the EKEventKit didn't take as long as I though it would (I expected it to be the bulk of my work). It turned out to be a fairly easy solution. However, the comparative lack of documentation made it more difficult. I found the value of picking apart sample code for methods and help.

Overall, I feel that the overall product turned out quite well. I did receive a lot of help from Michael and Mr. Kiang in the troubleshooting of various errors I encountered. I also relied on my classmates for help with testing. Aside from Lindsey's help with the NSTimer, I found myself mostly relying on the internet just as a product of the fact that my project utilized different parts of iOS from other people's projects. Nobody I knew was using EKEventKit. Even then, they served a great moral support and as beta testers (Record of Thinking 12/14/12). I want to eventually add more features and customizability to my app. I did have to omit the reminders and weather modules, the first for space and the second because of time issues (New Years Record of Thinking). When I looked at similar apps on the App Store, I saw that many of them had lots of features like stocks and weather. Even then, I want the main focus of my app to stay on being simple and minimal to keep it easy to use and to reduce the need for user input and keep it visually pleasing (Final Record of Thinking). Next time, I would try to take into account my audience more when planning my app so that I would be able to formulate a concept faster. I would also try to do things in larger blocks. I found that I would often do research (Work Log Week 5) and program later (Winter Break Work Log), which would delay completion more than if I just did the coding right after the research. My computer also broke around halfway through, although I was able to recover my work fairly quickly (Work Log 12/1-12/8) and only had a few provisioning profile issues (New Years Work Log). I also feel that maybe a partner might have been helpful in keeping me more punctual (All my gant charts). I originally had fears that my project was too sparse, but that simplicity later came to be an advantage. In this project I had to learn how to get by on less structure and guidance than before and how to manage my time and build an app without a prompt. This was one of my first mostly unstructured projects and I did have some procrastination issues I had worked out by the end. Overall, I think it went pretty well. I did start a bit late, but I was able to get a working product out that I feel worked nicely and performed as I wished.