4-24-2019

Compartments: A To-Do List App for Busy People

James Goodrum Kershner

Compartments: A To-Do List App for Busy People

by

James Goodrum Kershner

An Honors Capstone
submitted in partial fulfillment of the requirements for the Honors Diploma
to

The Honors College

of

The University of Alabama in Huntsville

4/24/19

Honors Capstone Director: Dr. Richard Coleman, Lecturer

James D. Kershner 4/24/19

Student Signature Date

Richard L Coleman 4-24-2019

Director Signature Date

H.L. Marquette/LE 4/24/19

Department Chair Signature Date

4/26/19

Honors College Dean Signature Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

James Goodrum Kershner

Student Name (printed)

James D. Kershner

Student Signature

4/24/19

Date

Table of Contents

Dedication:

This project and paper are dedicated to Bill & Donna Kershner and Bill & Betty Kershner. They enabled me to go to college and gave generously so I could succeed and begin my career with a strong foot forward.

Abstract

This paper will present, describe, and discuss the to-do app Compartments and the entire development process from design to completion. Each step in the process, including developing the initial idea, creating the initial design, and implementing the application, will be analyzed in detail. Furthermore, the author will also present what he learned from the project along with some of the challenges he encountered. Finally, he will address how this project completes and fully satisfies the Honors Capstone requirement for graduation from the Honors College at the University of Alabama in Huntsville.

Introduction

For my Honors Capstone Project, I wanted to do something a little bit different on a couple levels. While many Honors students add additional features or challenges to their degree's required senior project in order to meet the Capstone requirement, I wanted to use my Honors Capstone as a chance to take a second attempt at a personal project I had given up on several years ago. In this paper, I'll explain and expound upon what that personal project was, why it failed, why I wanted to try again, and all the details of making *Compartments: A To-Do List App for Busy People*.

Part 1: The Failed Personal Project

The "List-Everything" Phase

Before understanding the importance and design of my app Compartments and how it was even possible, you have to first understand the history behind the idea for Compartments, both the failed precursor and the unique design of Compartments itself. As a sophomore in Spring 2017, I decided that I wanted to make a list-making app for iPhone. During freshman year and the fall of sophomore year, I used a simple color-coded spreadsheet to keep track of my daily to-dos and my college assignments. While this approach worked well for a while, it gradually became clear that it wasn't quite as good as it could be. I discovered that I wanted more out of my to-do tracking experience. One of the things I wanted was the ability to edit my to-dos on my iPhone while I was out and away from my computer. I usually edited the spreadsheet on my computer because editing it on my phone required a lot of pinching, zooming,, scrolling, and selecting a bunch of buttons. This was one of the biggest reasons I

wanted something different- the experience of editing my spreadsheet on my phone was bad enough that I'd rather try a different app or write my own than continue doing it the way I had been doing it. After coming to that realization, I began looking into other apps to keep track of my to-dos. I honestly don't recall if I found any other apps I liked or even if I searched for other apps for long, because all I remember is being captured by the idea of making my own app to accomplish what my desires. This revelation lead to researching how to get into iOS app development (iOS is Apple's mobile operating system). I quickly found that it was very easy to get into iPhone app development. There are myriads of books, websites, articles, videos, and tutorials on the internet that were accessible to anyone. In addition to that, a year's subscription to the Apple Developer Program was only $100. While membership to the program wasn't necessary to begin developing iOS apps, I went ahead and got the membership and began coding with some simple tutorials published by Apple. However, shortly after this point in the process I began to discover that it wasn't going to be as easy as I had thought. As that point in my college career (sophomore year, remember), I had obtained very little instruction or exposure to the concept of object oriented programming (OOP). That being the case, I really had no idea and no information on how to design an app from scratch all on my own. Instead, I simply followed the Apple tutorials, step by step, and changed a few minor pieces of code in order to make the app look and behave like I wanted. This approach worked well for a short period, but after a certain point I realized I didn't understand what the tutorial code was doing or why it was doing it. Instead, I was just simply copying the code to my app and hoping it would work (sometimes it did, and sometimes it didn't). This lack of knowledge and understanding

lead to some very interesting bugs, and many hours of frustrating work trying to figure out why my code didn't work because I had no idea where to look. After wrestling with the app for a couple months, I set the project aside and moved on to other things as I entered junior year.

The "Studying Competitors" Phase

As I continued in my college career by starting junior year, I still wasn't satisfied with how I was keeping track of my to-dos and assignments. This dissatisfaction and the difficulty of creating my own app led to a quest to see if other developers had already created an app that satisfied my needs. The first app I tried was called Wunderlist. For a while, I was very pleased with Wunderlist. It fit my needs for keeping track of to-dos in a mobile format, and it kept my to-dos up to date across all of my devices. However, my relationship with Wunderlist slowly soured as I discovered more and more bugs in the app. The app began to regularly crash out from under me while I was using it, and I discovered that the more you used the Wunderlist interface, the more confusing and cluttered it became. Both of these things frustrated me, because on such a nice piece of hardware as an iPhone, I felt I shouldn't have to put up with poorly designed apps and should instead be able to find reliable apps that work well. My dissatisfaction with Wunderlist led me to discover Actions by Moleskine, and my relationship with Actions was much better. Actions is beautifully designed and executed, and it has the added bonus of being constantly updated by a dedicated team of developers. However, I also discovered some things I didn't like about this app, just like I did with Wunderlist. While Actions was had rock-solid reliability (unlike

Wunderlist), Actions did include a feature that caused me stress— a “Today” and “This Week” view. Most to-do apps allow you to make multiple parent lists that then include sublists, with each parent list being something like “Groceries”, “Work”, or “School”. Inside of those list would be the sublists, and these sublists would contain things like “tomatoes”, “onions”, and “bread”. However, while Actions did this and did it well, it also gathered all of my to-dos that had corresponding dates and organized them into a list of things I had to do today and this week. This, I discovered, could be very stressful because the “Today” and “This Week” views meant you were faced with a large list of to-dos coming from a variety of different master lists. I felt that this approach made the “one bite at a time” element of sublists as well as the whole point of making unique parent lists completely useless.

Part 2: The Beginnings of Compartments

Taking all of the previously mentioned things into account, I combined them all on a piece of paper. I wrote down things I like about Wunderlist, and I wrote down things I didn’t like about Wunderlist. I wrote down things I liked about Actions, and I wrote down things I didn’t like about Actions. I even wrote down things I liked about my failed to-do app from sophmore year, and I also wrote down things I didn’t like about it. Once this was all done, I was able to look at things I liked and didn’t like about each app’s approach to to-dos. These characteristics began to take shape in my mind, and they began to form into a basic picture of the kind of to-do app I wanted to create and use on my iPhone. Right around when these ideas were coming together, I entered senior year and realized I had gained a lot of the object oriented (OO) knowledge I

needed to be able to properly design and approach building an app of my own. These factors and ideas all combined into me deciding that I'd like to combine the idea of an Honors Capstone Project with my personal project of a to-do list app. That being the case, I began to plan more specifically what I wanted Compartments to look like.

A Clearer Picture of Compartments

First, I wanted my to-do app to "compartmentalize" tasks, hence the "Compartments" title. When a to-do or list-maker app has lots of push notifications or a "today view", the lines between your specific parent lists get blurred. When this happens, the user becomes more and more stressed because they are basically just making one giant list and looking at one giant list at the beginning of the day. That's certainly something that causes me stress, and I know it does the same for other people too. So, with that in mind, I set out to make Compartments different from most other listing apps. I wanted Compartments to be nice and simple, simply having a list of lists. In other words, I wanted Compartments to have a list of parent lists, each of which contained to-dos. While this is pretty common in most apps of the to-do list kind, I wanted to limit my app to only that depth, two levels (lists and to-dos). Other apps, like Trello and Wunderlist, delve 3 or more levels deep with sublists (parent list, child lists, and sublists for each child list item). This also stresses the user, because it's confusing and more difficult to grasp what things they have to do when they're having to mentally keep track of sub-sublists. This is why I chose to design Compartments with only two levels— lists with items— which keeps it simple and makes the amount of to-dos a user has to complete appear fewer.

Second, I wanted Compartments to not have a push notifications feature. When a list maker app has the ability to add reminders and alerts to each list item, it can get out of hand very quickly. These sorts of reminders don't help a user like me at all; instead, they just pile up on my lock screen and notification center, and this makes the amount of to-dos I have seem unbearable. When this happens, the boundaries the user placed by creating separate, topical lists are completely blurred and mean that often the benefit of making those separate lists is completely lost. Instead of being drowned in a stressful tsunami of notifications, I wanted to make Compartments such that in order to see a to-do, the user has to enter their topical list. Similar to the concept of "work staying at work" when an employee leaves at 5pm, keeping the to-dos "compartmentalized" in each topical list means when the user leaves the list, the user mentally is not carrying the responsibility of those to-dos anymore. They can check back to see what items they have under their topical list, but the act of having to enter that list allows them to also leave those items when they exit the list. This can be especially helpful for people who are super busy but still want to keep track of their to-dos. As the old saying goes, "you eat an elephant one bite at a time," and Compartments does this by trapping the to-do items inside their lists. This prevents the user from having to face all their items at the same time and allows them to take a more "elephant-eating" style approach to their daily tasks.

The two features mentioned above, simple lists and no push notifications, are the two main ideas that drove me to create Compartments. Based on my research, which has been broad, albeit not completely comprehensive, most to-do apps use several layers of lists, complicated to-do items, and the ability to make constant

reminders and alerts. I recognize that some users desire the flexibility of complicated to-do items and many layers of lists, but I believe that the average user, keeping track of tasks in categories like work and school, would rather use an app that is as simple and stress-free as possible. Their to-dos are plentiful and stressful enough as it is- why add more stress on top of that with an overwhelming app and categorized lists that immediately blur together?

Part 3: The Process for Making Compartments

After discussing the design of Compartments a little, I'd now like to discuss the process for making Compartments. It has not been an easy experience, but it has been a rewarding and exciting one, one that has given me the desire to continue in the world of iOS app development. To begin, as I mentioned several sections ago, designing and then implementing an app requires the software developer to have at least a working knowledge of object-oriented design. This allows the developer to take an idea and break it down into what "objects" are needed. In order to find the "objects" you need, many software developers underline all the nouns in their app proposal/idea. After finding these nouns, the software developer would then look to see what sort of characteristics those nouns can take on or exhibit. For example, if the designer's idea included a "dog", then he would call his object "Dog" and then lists the characteristics that the object "Dog" can take on, which would include things like weight, size, color, breed, and so on. I learned this approach at UAH, and it is the same approach I used to begin developing Compartments.

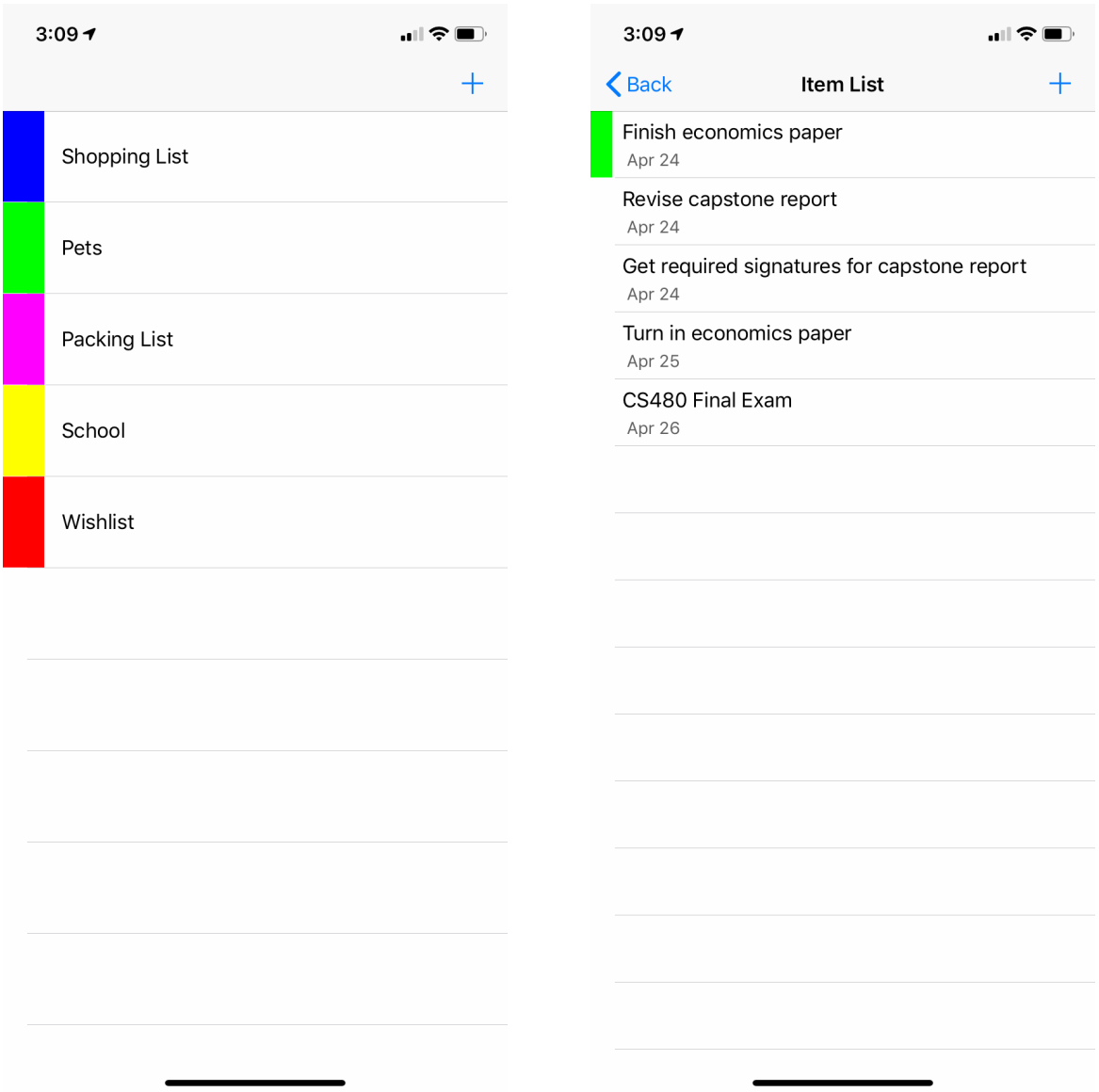

Figure 1: Compartment Lists and Item List

For Compartments, I looked at my idea, which was simply a list app with a list of lists. I realized that I would need two lists to accomplish this. One list would contain the lists, and another list would contain the contents of each list. I separated the two types of things that would be contained in those two lists. One list would contain lists, and the other list would contain to-dos. I dubbed these parent lists “compartments”,

because they held the actual to-dos, and I dubbed the to-dos “items”. Put simply, compartments are like a drawer where you put items. You open a compartment to view and edit the items, and then you close the compartment. So, after deciding that I needed two objects, compartments and items, I then considered what characteristics I wanted these two objects to have. As previously mentioned, I did not want items to have a reminder or notification feature, but I did want them to have a date that could be displayed to visually show when the user planned on completing the to-do. However, due to the constantly changing nature of life and to-dos, I chose to make this date optional and easily changeable. In addition to a date, I gave each item a name and a complete/incomplete state (see Figure 1). For the compartment object, I gave each compartment a name, a color (to visually distinguish it from other compartments), and a collection of items (see Figure 1). After doing these two things, I was now able to configure and design my Compartments app.

When making an iPhone app, you have to use a specific computer language called Swift. While I had used Swift a little bit in the past when I attempted to create that app in sophomore year, I was still not super familiar with the language and certainly not completely familiar with the program Apple uses to create its apps: Xcode. In taking on this task to build my Compartments app, I was also taking on the parallel task of learning how to code in Swift as I went along. I felt I was up for the challenge and plunged in. Many times I ran into troubles or questions, but when I did I quickly searched for articles, videos, tutorials, forum posts, and any other resources I thought might help me along. Many of the resources I found were very helpful and some were less than helpful. Through it all, I realized that as a software developer, you never really

know everything. Developing software is a constant process of learning, using what you learned, running into something you don't know, and learning again. This was definitely the case with designing and implementing Compartments and I know it's something that will continue for the rest of my career in computer science and software development.

Part 4: Challenges

In making Compartments, I ran into several challenges. Some of these challenges were related to me still learning the Swift language, others were related to me learning about the Xcode development environment, and still others were because it was a new challenge to apply my knowledge gained from college to an entire app development process. A couple particular challenges come to mind and I'd like to take a moment to discuss these.

Unknown Errors

First, I ran into errors I had no idea how to fix, which was one of my major challenges. There was one particular error that, based on my research, usually meant that the programmer (me) had forgotten to connect a few pieces of his code. I scoured my entire codebase for those missing connections, deleting and rewriting several sections of code multiple times, but never found the source of the error. Missing connections proved to not be the problem, but I only discovered that after many hours of frustration searching through my code. After examining the error message more carefully (something I should have done at the very beginning), I discovered that I had

simply forgotten to name a part of my code. This challenge was a clear and constant reminder to always examine the error codes- you'll save yourself hours of difficulty and frustration.

Overestimating My Skills

Second, I overestimated my skills as a software developer. While I mentioned above that I overestimated my skills with Swift and Xcode, I also overestimated my skills in another way. I came into the project confident that I had everything I needed to design and develop my app, and I was also sure that when I did come across a part of Swift or Xcode that I didn't understand, I would be able to quickly find a solution with either my current skills or a quick Google search. However, as I progressed in the project, I quickly discovered that my knowledge of design and execution of the entire development of a program was not as large or all-encompassing as I thought. As I got deeper into the project, I discovered that my knowledge of Swift was limited enough that as I followed tutorials and got parts of my app set up, I was not able to do certain things I thought I could because Swift isn't designed that way. For example, after following some tutorials (both from Apple and from Youtube), I found that the code I was following along with was unable to accomplish what I needed. This code passed a certain piece of data in a certain way, but when I tried to pass a different piece of data in the same way, it crashed. After examining the error messages and my code, I discovered that instead of just a coding mistake, I needed to redesign that entire piece of Compartments. This took a large amount of time, an amount that I would have preferred to spend on other things like assignments for other classes or adding

additional features to Compartments. More knowledge up front would have allowed me to design Compartments with the programming language in mind, and this would have allowed my program to be more organized and stylistically better as far as coding goes. The lessons I learned from overestimating my capabilities up front (which happened more than once) taught me three important lessons.

The first lesson I learned was that when I'm designing an entire program, I need to have a better knowledge of the language in which I'm programming AND the development environment. In most of my college career, I've written programs that have been relatively simple graphically. Most of these programs have been focused on completing a certain task or satisfying a certain requirement instead of looking nice and pretty. As such, I've written most of these programs for use on the command line, and therefore have not really had to worry about designing my code to work with my graphics, other than printing lines of code to the terminal now and then. However, with Swift, I also had to understand the graphical concepts and how each graphical piece communicates with the other graphical pieces. Unfortunately, I wasn't fully familiar with how these graphical pieces worked, so by the time I discovered my code wasn't working the way I thought it should, I had already very thoroughly confused myself. This meant the time it took to fix the error was even longer because I didn't always know what I was looking for. In these situations, the Apple Developer forums were a lifesaver for me. I would typically get a quick response whenever I posted a question and I was able to post my code inside my question so the other users of the forums could see it and understand the problem I was having. These moments were very helpful, but it took a lot of time to get frustrated enough with something to finally ask a

question on the Developer forums, partly due to stubbornness and partly due to simple lack of knowledge. That's unfortunate because in a working environment, I often have a limited amount of time to wrestle with a problem.

The second lesson I learned was that I shouldn't take coding an entire app on my own lightly. At work, I can always get up and ask one of my coworkers a question or for help. It's great and it's straightforward- no struggling through pages upon pages of Google links only to still not find anything. The coworker I ask either knows the answer or doesn't, and if they know someone who knows the answer, they'll point me in the direction of that person. However, when I take on an entire project— start to finish— on my own, one that is in a language that many people aren't familiar with, I have no coworkers that I can quickly ask for help. Instead, my only option is to read books or search the web, which can sometimes be hit-or-miss depending on how unique my need is. If my need is specific or if I'm doing something in a certain way that not many people have done, I end up having a very hard time finding answers or help. This makes the development process longer, my stress levels higher, and the deadline approach quicker than I would like.

The third and final lesson I've learned in this category is how making an iPhone app is far more difficult that one might expect. We all read articles or news headlines about how one person made a great app and it took off like wildfire, but that's not very common and exceedingly difficult to pull off. If it was as easy as everyone makes it out to be, everyone would have a successful app and be rolling in money from millions of sales. However, as I've come to find out, apps are very difficult and very involved to make. They require a lot of knowledge to even begin, and they also require a lot of

knowledge to plan out properly. The less you know up front, the poorer a job you'll do planning and designing your app. The poorer job you do planning and designing your app, the more issues you run into during development. The more issues you run into during development, the longer development takes and the more you get stressed if you have a certain deadline to meet. These were all things that I discovered the hard way, and they all make me want to study Swift and Xcode more before I dive into another project involving iPhone app development or add any additional features to Compartments.

Part 5: A Self Assessment of My Performance

I am usually extremely critical of myself, and that makes a self assessment something that takes a little more thought than it might for other people. As I reflect back over the course of designing, developing, and creating Compartments, I see moments where I didn't know what I was doing and periods of time spent trying to figure out why my code didn't work the way I thought it should. I also see changes to my original plan and even some features that I wasn't able to implement. However, these things mentioned above are very good things, and actually are far more normal than you might think. Therefore, I truly feel that I did a good job with Compartments and a job worthy of the Honors Capstone requirement.

Towards the end of my Honors Capstone journey, I took a few minutes to sit down with my faculty advisor, Dr. Richard Coleman, and discuss my project, what I had accomplished, and also what features I discovered were too much for me. As we chatted, Dr. Coleman reminded me of something that I had learned myself over my four

years at UAH as a computer science student: very rarely do computer programs go as planned. In the instance of Compartments, I originally designed it with multiple types of to-dos (“items”, remember). However, as I progressed through designing and developing Compartments, I realized that the concept of different kinds of items was both a little too complicated for someone who was just learning iOS app development and also a concept that went opposite of my purpose for Compartments, making a to-do app that kept things simple and helped the user to compartmentalize their to-dos. Having multiple types of items added complication to the app that was not countered by added usability. Therefore, I decided to drop the multiple types of items in favor of a one type of item style. This is a fine example of how computer programs never go as originally planned, and they shouldn’t either. As many professors have taught me and as I’ve learned in many a difficult programming assignment, software development is a living process that is constantly changing as you navigate through it.

Connecting that back to my self evaluation, I realized that Compartments changing from the initial plan is something that is completely normal, and therefore is not something I should count as a “less than stellar” part of my Capstone journey. There were several other related instances where I wanted to add more features than was feasible in the time that I had for my Honors Capstone, and therefore I chose to remove them from the project. However, like Dr. Coleman mentioned, this is something that is totally normal in the software development process. By encountering this on my own, with a project entirely of my own, I now have a better idea of what I am capable of and what I am not capable of.

Overall, and using the examples above, I truly feel that I created a piece of software that exemplifies the characteristics and qualities of the Honors College. While yes, I accomplished my goal of creating a simple and usable list-maker app, I also learned a vast number of lessons along the way, spanning everywhere from learning what I'm capable of to learning how to pass data between ViewControllers in Xcode. The lessons and skills I've developed and refined in creating Compartments will serve me well in the months and years to come as I embark on my post-college journey into the industry of software development.

Conclusion

In conclusion, for my Honors Capstone project, I created a simple list making app called Compartments. Compartments is a simple and easy to use app designed to help people compartmentalize their to-do lists and lower stress in their everyday lives. In the months to follow beyond graduation, I hope to continue adding features to Compartments and develop it into an even more powerful application for the organization of people's lives.