Team member list:
John Zhang, jwz2kn@virginia.edu
Jeffery Cui, jyc2ra@virginia.edu

Device Name:
Serial Number CCQPR0DFGGK5

Project/App Title:
IOS Reminder List

Basic instructions on usage:
1) Open up the app from the apps page
2) Click the add button in the top right corner to add an item
    a) You will need to fill out a title, description, and due date in order to submit
    b) Press submit to add to list
3) To edit an item, press a list item and it will take you to the info page where you can edit info
4) When the due date passes there will be a reminder popup that has two choices
    a) Postpone an hour
    b) Delete the item from list
5) Also can delete items by sliding left on any item

Any special info we need to run the app:
N/A. If any errors occur, please don't hesitate to contact us.

Lessons learned:

We primarily learned three lessons of iOS app building in this project. We built had to use NSArchiver to hold the reminder list items, learned how to deal with important UI components in AutoLayout, and learned the differences between Swift syntax and Java or Objective-C syntax.

One added benefit of using NSUserDefaults was that our reminder list was stored in between app runs. That is, our reminders don't disappear from the list once we close the app! We thought it was important to learn how to do this as early as possible, because more professional apps will have that kind of continuity. Because we decided to use NSUserDefaults, we had to store our custom objects and data as NSData in order to save the object array into memory. This required us to add NSCoding to the custom object class and then use NSArchiver inside the tableview class. This combination required us to package the data to store and then unpack when we used it.

Using Xcode's AutoLayout along with ScrollView was a difficult step for us. When the app was turn horizontal during the enterinfo screen, the contents would go past the bottom of the screen. To solve this we tried to do this with ScrollView and ran into issues. We kept running into an issue where the size of the ScrollView content was ambiguous. We tried many different things like putting an extra View to act as a content holder for all the other views. In the end we realized that all we had to do was to set constraints on every view in relation to each other as well as the screen. This gave the ScrollView a definite size and solved our problem.

One of the things in Swift that took some getting used to was the required “!” and “?” after many lines of code. The “!”, when used in an initialization, means that a variable will not be nil, and the “?” means that a variable could be nil. Problems occurred during coding where we often had to insert additional “!” or “?” when dereferencing a variable, or trying to pass in parameters, or trying to cast variables of different types. That took some getting used to, but we did learn how to use “!” and “?” properly to write sensible code. We also had a few problems with the keywords “let” and “var”, which were the difference between mutable and immutable variables. We initially had some problems with this, but our solution, more often than not, was to use “var”, because it allowed us to change a variable’s value when necessary.