Team Member List: Benjamin Fuhrman, Mark Manahan

Device Name: Nexus 6 API 24 (Android 7.0, API 24)

Project/App Title: An Android Bucket List Application

Basic Instructions on Usage: On launch, the application will present a pre-populated bucket list of activities that still have to be done, which are sorted in order from due soonest to latest.

The floating action button marked with the plus sign serves to add a new item to do to the list, bringing up an entirely new interface. On the add new item interface, there are four text fields, name, description, latitude, and longitude, which can be filled to describe the activity to be done. Additionally, a calendar is displayed below the fields which is used to select the due date for the new item. Pressing the back-arrow button cancels an addition to the list and returns the user to the main screen with the whole bucket list. Pressing the “save new item” button creates the activity to be done according to what is stored in its descriptor fields and adds it to the overall list and finally sorted among the items that are already on the list.

Tapping on the name of an item on the list brings up the edit-item interface. Editing an item that is already on the list is similar to adding a new activity in that editing has the same interface but the descriptor fields are pre-populated with the descriptors of the item that was tapped. Any of the descriptor fields can be filled to change the components of an item. The back-arrow can be used to cancel the edit and return to the whole bucket list; the “save changes” button saves the information that was entered into the fields for the item and returns to the whole bucket list, reflecting any changes made.

Checking off a checkbox next to an item marks it as complete. On completion, the completed item is re-sorted among the other completed items, if any; incomplete items are kept sorted separately from completed items, incomplete items being sorted “on top” of completed items in the whole bucket list.

Lessons Learned: We were able to learn about what formally constitutes an Activity in a mobile application and how it is fundamentally a commonplace for some sort of task to do or for several tasks to be done. Activities work much like regular Java classes in that they, too, have their own attributes and forms of functionality about them as an “instance,” itself. Instances of activities are initialized through the onCreate method which lays out the skeleton of the activity, re-using a previous instance of the same activity, if necessary, and adding functionality to buttons that may appear on the activity itself. Previous instances are managed through the onSaveInstanceState and onRestoreInstanceState methods in the case where certain attributes of an instance are necessary to be “remembered,” such as when the application rotates itself, which re-initializes an instance of the activity. We also learned that there is a lot of communication between an activity .java file and its corresponding .xml file, mainly for identifying buttons on the activity’s layout and specifying particular methods that should be run should they be interacted with. We also learned a lot on the semantics of activity-to-activity communication, which is mainly done through intents. Activities can start, or call the onCreate method of, other activities from within their own context while sending information on what to do in the called activity through Intents. Intents are crux of communication; data can be “attached” to intents which are sent over to other activities and extracted from those other activities to be used in some way. We also learned that adapter classes are useful for dynamically adding functional UI interfaces to an activity.