Orkun Krand

Dr. Shubham Jain

CS541 - App Development for Smart Devices

February 4, 2018

TO-DO LIST

I was able to complete all the requirements of this assignment. Firstly, I removed the default top action bar and put the general title as a text in its place. It looks cleaner this way as I think the top action bar is somewhat ugly. Below the title are two EditTexts for the new item title and the new item description. Below the EditTexsts is a ListView to hold the to-do items entered by the user. That is followed by the add button, separated from both the ListView and the bottom of the screen using `android:layout_marginTop` and `android:layout_marginBottom`. I used an adapter to successfully integrate the contents of the arraylist with the ListView.

The main layout is stored in a vertical LinearLayout. The template for the elements of the ListView are stored in a separate file called `list_item.xml`. This file has a main horizontal LinearLayout to separate the TextViews from the checkbox. The TextViews are stored in a vertical LinearLayout (within the horizontal LinearLayout). All text is centralized using `android:gravity` and `android:layout_gravity`. I put limits on how many characters a user can enter for each TextEdits as you mentioned that there should be a "short" description for each item.

I created a custom class to hold my list items. This custom class is called `ListItem` and has two strings for title and description. To be honest, I was expecting the Java part of this application to be simpler. This is partly my fault as I didn't read the part of the assignment that said we should write our to-do list into a file. Initially, I decided to use `onSaveInstanceState` but later realized I can't use it for arraylists of custom type. While playing with the instance state, I realized I can use it to keep what the user has typed into the textboxes when the user presses home, or rotates the device. So I left that part of the implementation for better usability. After, in order to keep the list after the user terminates the

application, I went for the shared preferences route. In order to keep its structure, I turned my list items into a JSON array and stored them that way (after trying many different methods). It was after successfully sending them using shared preferences that I re-read the assignment instructions and realized that I should've written the data into a file. So in order to keep some of the code I'd already written, I decided to write the JSON array as a string into the file and read from it.

The checkboxes were a little bit frustrating to deal with mostly because Android is trying to recycle them. I would get a checked checkbox after deleting one. So I decided to change the checked property of the box to false, which called the `setOnCheckedListener` twice, causing me to delete two items at a time. After fixing that, I then decided to make it fancy by putting a delay so the user can see the checkbox being checked. Setting the checked property to false also initiates the uncheck animation which I couldn't get rid of no matter what I tried.

Overall, I think this was a good assignment to start learning Android as it pushed me to experiment with both instance states and shared preferences. Plus, I have actually built something I can put on Github. I know you suggested we turn off the auto-complete feature of Android Studio but I don't know what I would've done without it. It was extremely helpful.