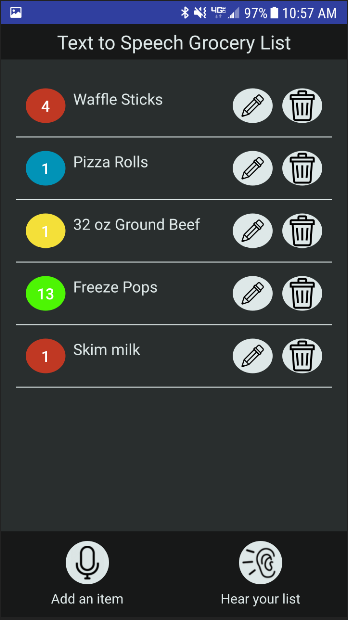
James Rinella / Brandon Andrews

CS 5590 Web/Mobile Computing

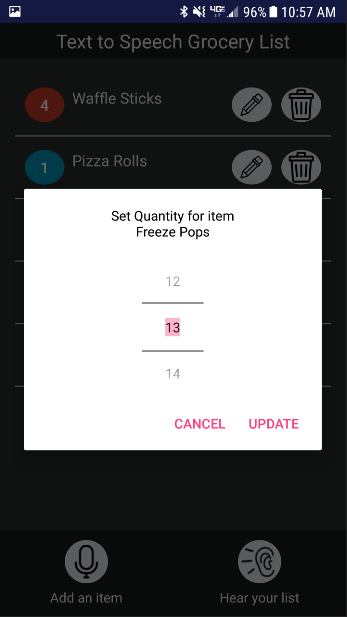
Mobile Lab 6 Report

Grocery Mobile App

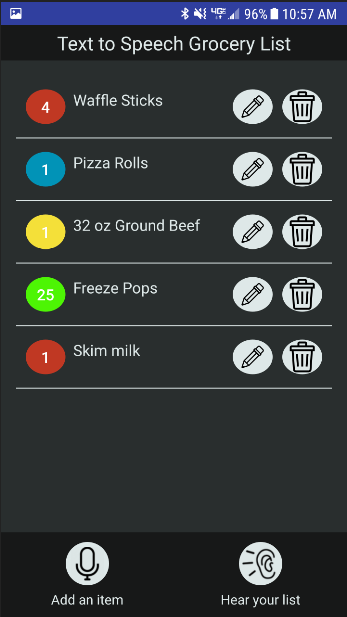
The Grocery mobile App utilizes text-to-speech, speech-to-text and Google firebase backend services to store grocery items in a simple listview. Using the speech-to-text API, the user is able to verbally add items to the grocery list, which will display at the bottom of the listview. Managing grocery items is accomplished by editing and deleting capabilities. We interpreted the “rating system” objective of this lab and instead stored a different piece of data; quantity for each grocery item. A user is able to modify an item’s quantity using a dialog box and can also remove an item from the listview. The text-to-speech API will notify you of a successful edit or delete operation. Last, the app will read your items and their quantities out loud if you don’t want to take the time to scan the list for a specific item.



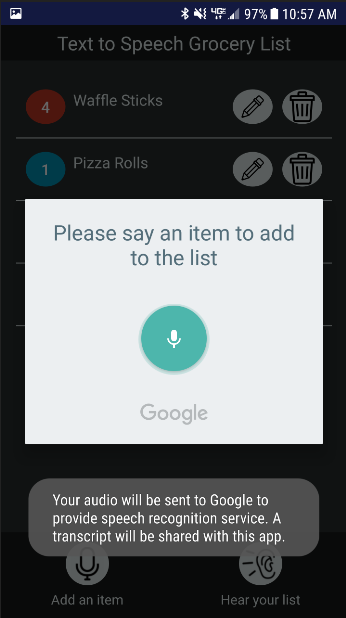
Sample grocery list when application is first launched. Each grocery item has edit and delete operations. The microphone icon allows the user to insert items verbally and the ear icon will read each item name and quantity out loud.



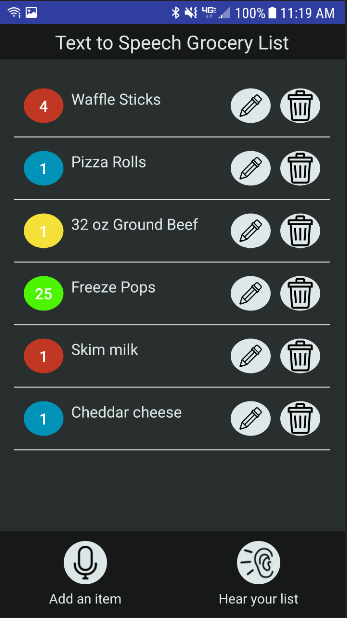
Editing to the quantity for Freeze Pops item using a dialog and a simple number spinner. The spinner is always initialized to the item’s current quantity.



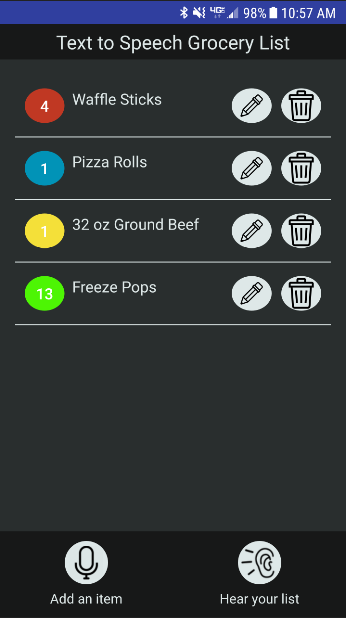
After editing Freeze Pops through the dialog, notice the quantity is updated on the listview



Upon tapping ‘add an item’, the user is prompted to say what item they want inserted into the list. Google will do some processing for the app and return what it thinks the user said. As an example, I added ‘cheddar cheese’ via the speak dialog

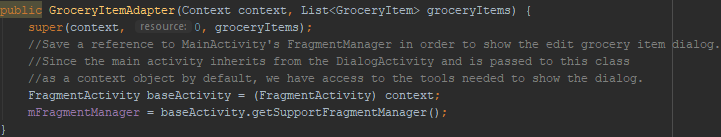


Cheddar Cheese is now added to the list and initialized to a value of 1 for the quantity.

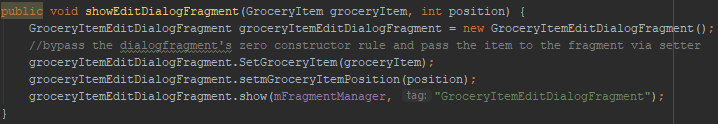


Deleting a couple items via the trash icon.

One technical challenge posed by this application was launching a DialogFragment layout from a listview item. Dialogs are simple to launch from activities since a main activity inherits from an Activity base class that comes bundled with a FragmentManager. However, since listviews are controlled by an ArrayAdapter class, the FragmentManager is not available in this class. Luckily, we can pass the main activity to the ArrayAdapter as a context then re-cast the context back to an activity and retrieve the FragmentManager for use in the ArrayAdapter class. Now dialog boxes can be used within a listview.



Constructor for the listview’s custom adapter GroceryItemAdapter. Here, we retrieve the fragment manager from the main activity and save a reference to it so that the dialog box can be launched when the edit icon is tapped for a list item.

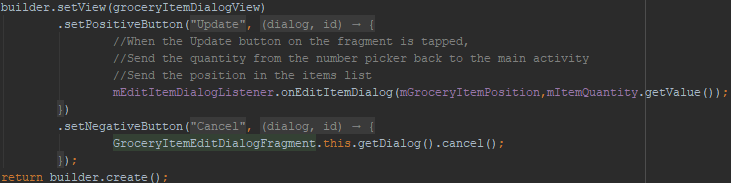


Launching the dialog from a listview item is now possible since a reference to FragmentManager was saved in the last code snippet.

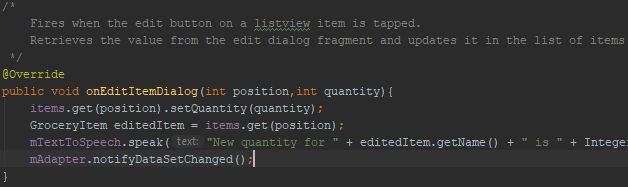
Another technical challenge was ensuring the application knew which grocery item to modify when the user taps edit and updates the quantity on an item. The issue was that the main activity, which houses the list of grocery items globally, has no way of knowing which item was modified in the dialog box. The solution was to modify the interface responsible for sending item modifications back to the main activity from the dialog to include a callback method with the item’s quantity and position in the list. So when an update is made on the dialog box, it sends the new quantity and the position of the item in the grocery list so that the main activity knows which item to modify. The dialog knows the position of the item already since it was launched from the GroceryItemAdapter.



A simple interface in the EditGroceryItemsDialogFragment that the main activity is required to implement.



The GroceryItemEditDialogFragment has a reference to the main activity through this listener and is able to call onEditItemDialog() with the modifications made through the dialog. The new quantity is sent back to the main activity.



The main activity’s implementation of EditDialogListener’s method. Since the position was passed back in this callback method, the main activity knows which grocery item to update.

In Firebase, the application stores all of the grocery items along with their quantities. The core piece of code that identifies changes is the ValueEventListener (screenshot below) which triggers each time a change is made to the database. When this occurs, the local list that stores the grocery items is cleared and repopulated. The main view displays data from this list, so it is integral for our application that it is updated. The database is located here: <https://console.firebase.google.com/project/csee5590lab6/database/csee5590lab6/data/groceries>

groceryDatabase.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(DataSnapshot dataSnapshot) {  
  
 items.clear();  
  
 for (DataSnapshot grocerySnapshot : dataSnapshot.getChildren()) {  
 String key = grocerySnapshot.getKey();  
 Number tempvalue = (Number) grocerySnapshot.getValue();  
 Integer value = tempvalue.intValue();  
 GroceryItem groceryItem = new GroceryItem(key, value);  
  
 items.add(groceryItem);  
 }  
  
 mAdapter.notifyDataSetChanged();  
 }  
  
 @Override  
 public void onCancelled(DatabaseError databaseError) {  
// new Log(databaseError.getMessage());  
 }  
 });