Michael Rzepka – 34

Haritha Rao – 14

# Grocery Buddy

An Android application which helps users manage their grocery shopping needs, including directions and information about nearby grocery stores (using the Android frameworks GPS functionality) and curating recipes to track and maintain them within a local SQLite DB.

## 1) Design

### Architecture Diagram



### Class Diagram

## 2) Features Implemented

Two primary features drive the application. The first being a remote web service which takes in the Android Framework’s GPS functionality to find the latitude and longitude of the users current position and returns information about any nearby grocery stores (hours, directions, websites, and other general information). The second is a recipe curator which helps users manage their recipes by using a local SQLite Database through the Android Framework. The recipes persist between application uses and allow users to enter both the detailed ingredients needed as well as the directions to follow.

### Web Services

An external Google Maps Web Service which takes in the desired business along with the users current latitude and longitude to reverse geo-locate nearby stores. Information is then returned including directions from the current location, information about the establishments, and further contact information.

### Database

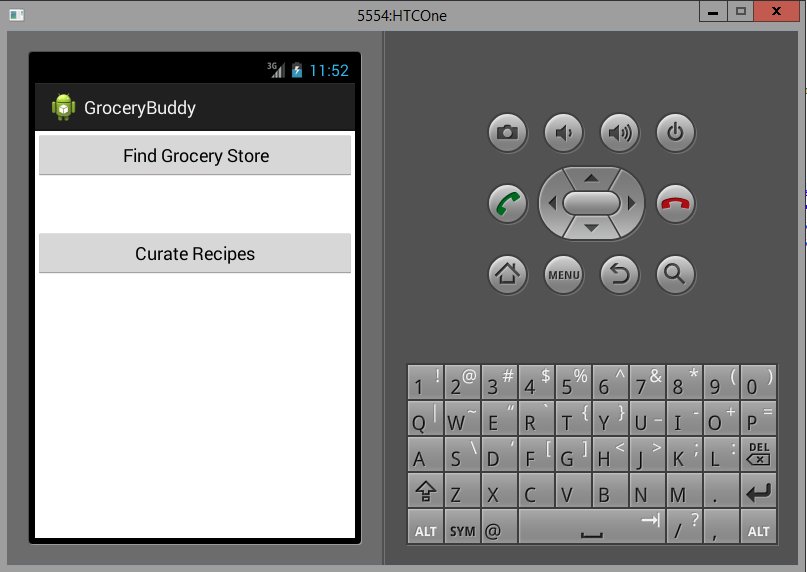
A local SQLite Database built within the Android framework to exist locally on the device which will maintain recipes between application uses. It helps users curate the recipes they maintain using a “recipes” table with uniquely identifiable items based on the primary key \_ID field, a unique title, detailed list ingredients, and step by step directions.

### Mobile User Interface

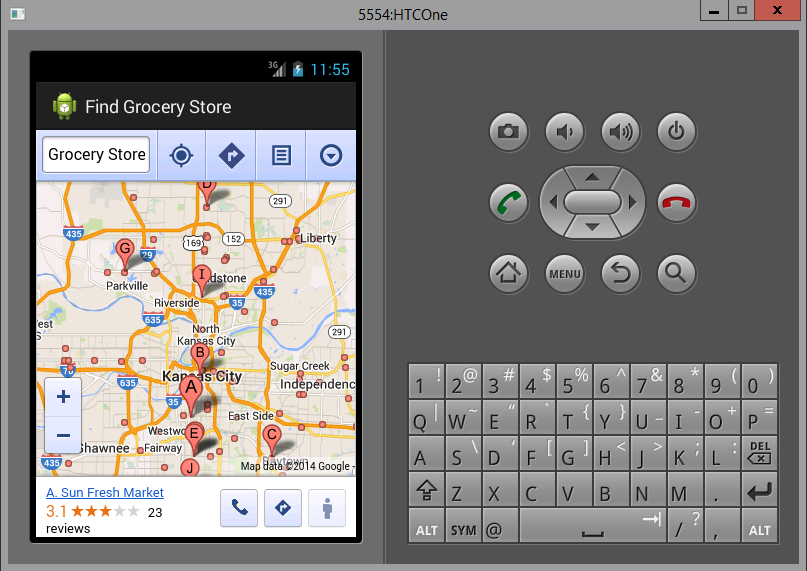
The interface uses standard Android layout widgets with a custom design. It’s based on a primarily landing page which directs users to each of the features available. If the users desires to find a local grocery store, they are prompted with a WebView of the information using their GPS location to populate the data. If the user would like to save, retrieve, or manage their recipes, they can use the curate recipe functionality to persist those changes to a local device database which maintains them across uses of the application.

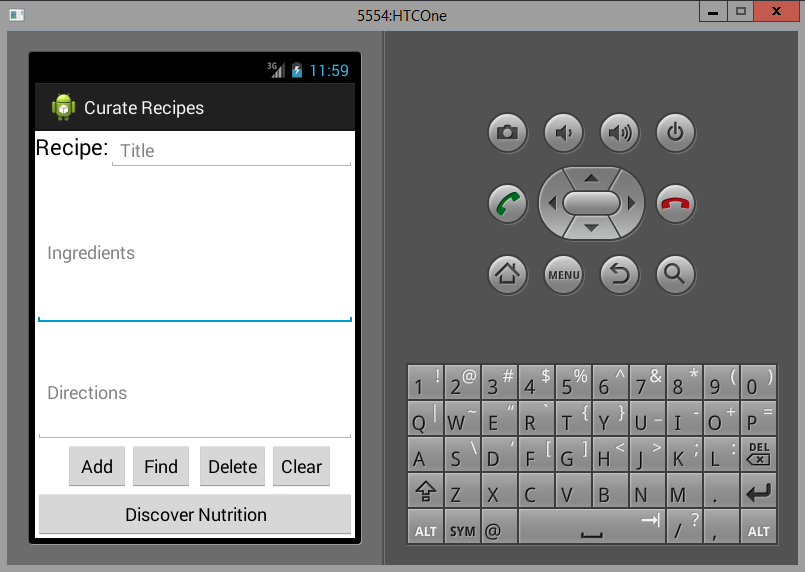
## 3) Screenshots

The basic application landing page (MainActivity):

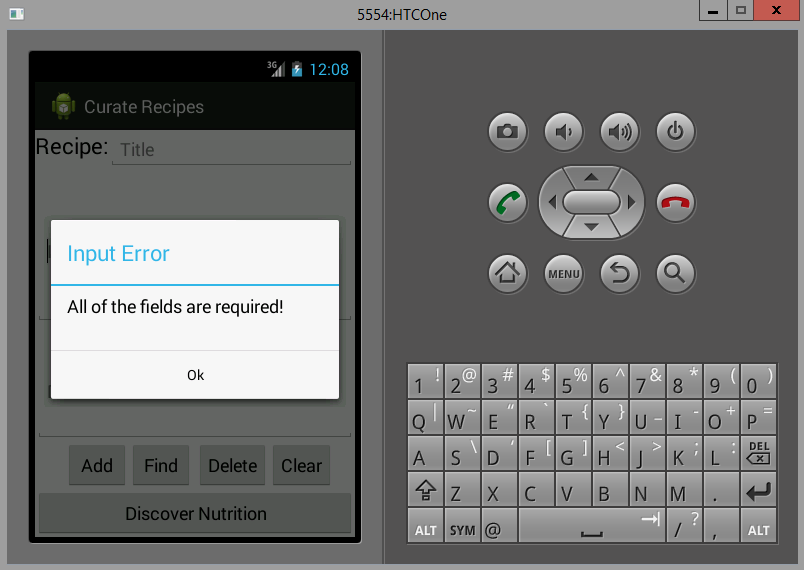


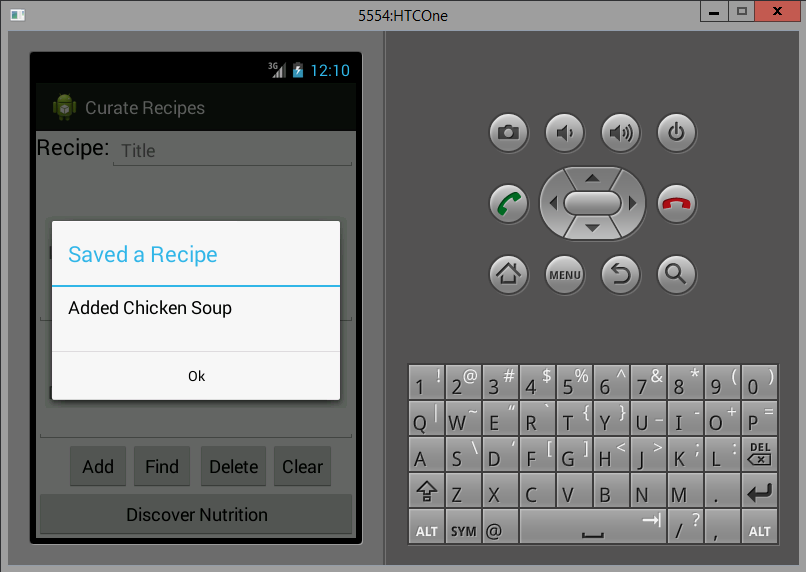
The find grocery store feature:



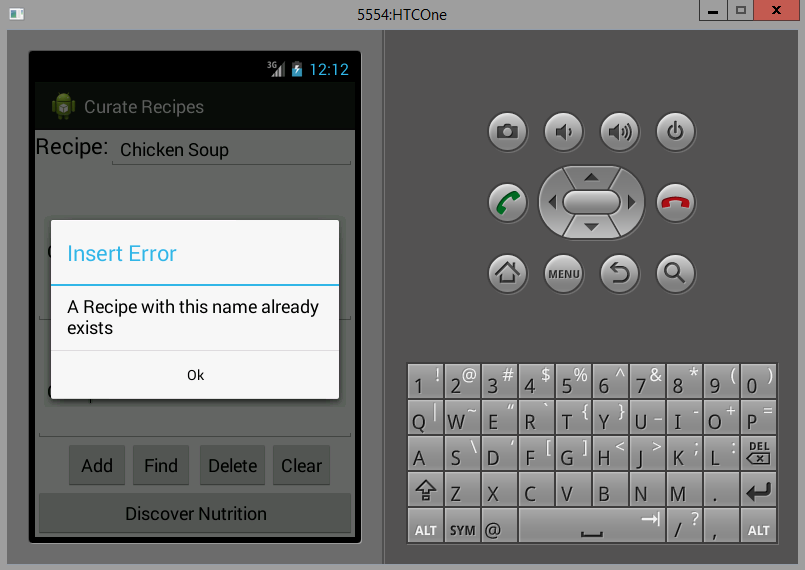
The curate recipes feature:

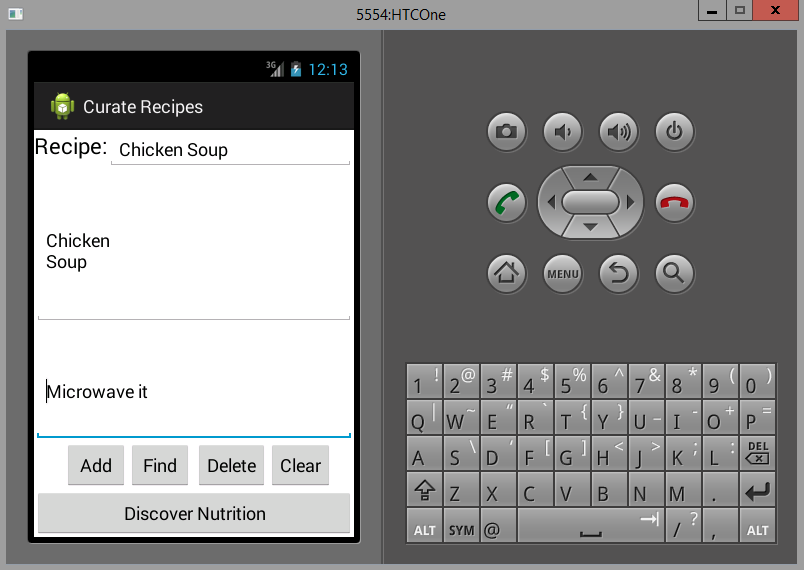
The curate recipes input validation:



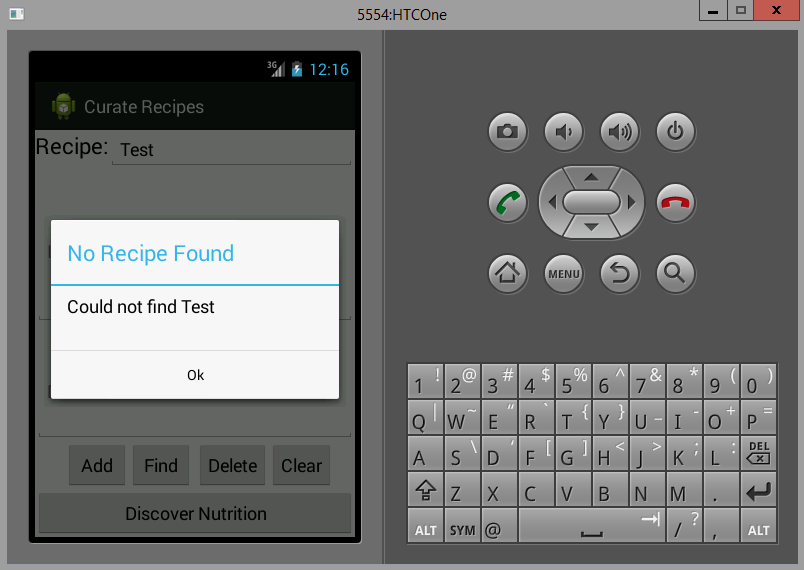
The curate recipes save feature:

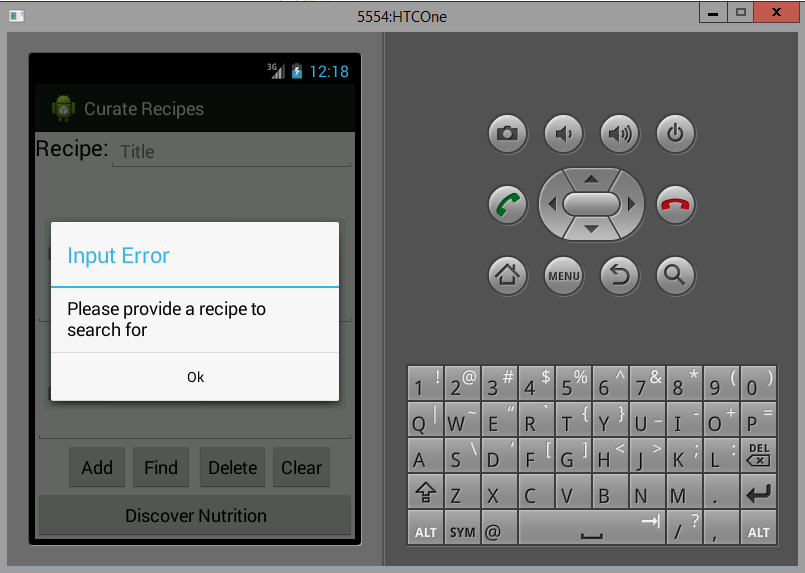
The curate recipes exists validation:



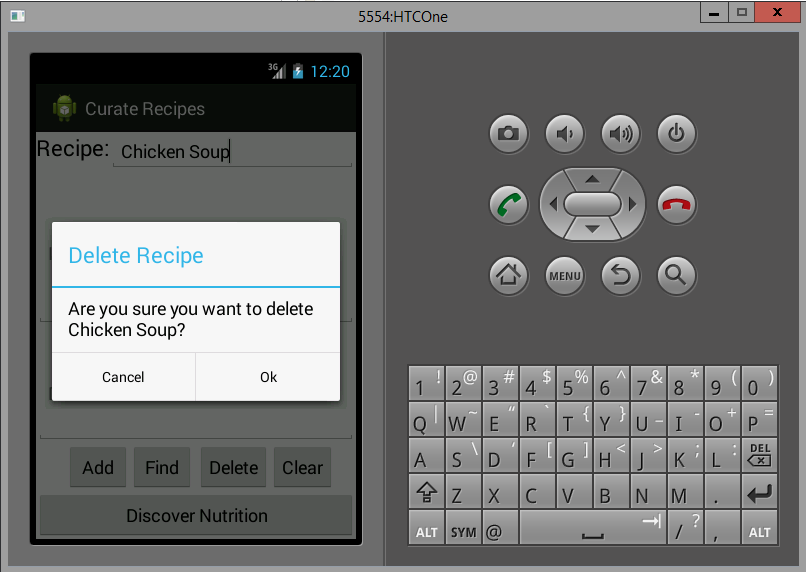
The curate recipes find feature:

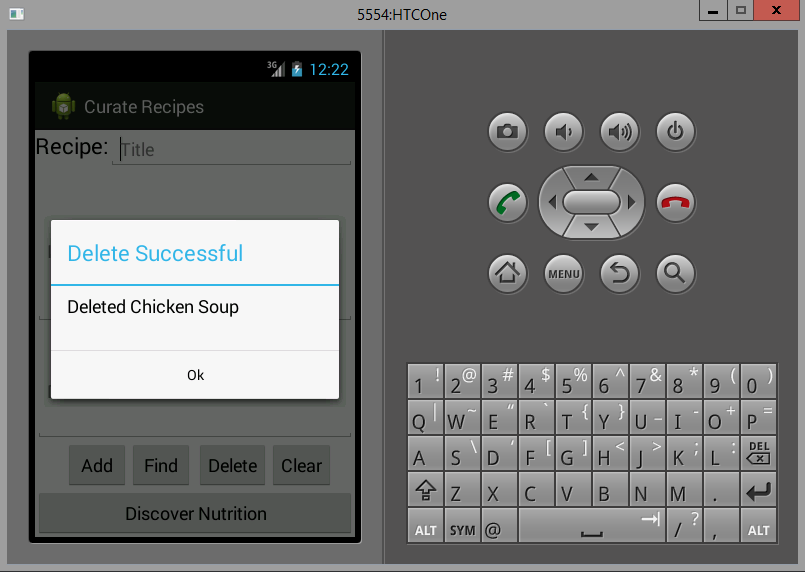
The curate recipes find validation:



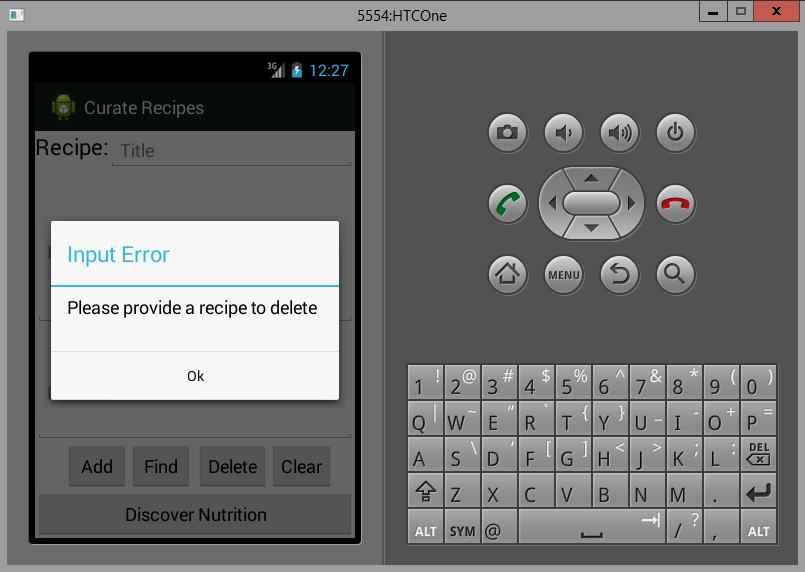
The curate recipes find requires item:

The curate recipes delete verification:



The curate recipes delete successful:

The curate recipes delete requires item:



## 4) Web Service URLs

"https://maps.google.com/?q=Grocery+Store&center=" + lat + "," + lng

## 5) Challenge #1 Github URL

<https://github.com/mtrzepka/ase-challenge1>

## 6) Limitations

There are some limitations with the application above. One with the curate recipes feature is that there is not delete validation that first does a find check (to determine if the entry is already in the database prior to attempting to delete it). The current functionality will not throw an exception and fail the application in its current state (a dialog will appear stating that there was a database delete error), but this validation should be included to make the application work in a more natural fashion. This could be added by abstracting the find logic out of the CurateRecipeActivity findRecipe method into a separate method that could be called by both findRecipe and deleteRecipe. From there, an AlertDialog could be created similar to the ones shown above which would validate if the entry did not exist (and therefore should not actually try to delete it from the database).

Another limitation faced was the Discover Nutrition flow. This feature was not implemented, but was planned and could possibly be implemented in the future. This would be another Web Service call which would use the Nutrition API (<https://developer.edamam.com/docs>). It would convert the entries made in the local SQLite DB, convert them into a JSON format, make a POST call to the REST Service, and retrieve the nutritional information about the recipe back to the user in a GUI format. This would not be very difficult to implement from this point forward, as the framework is composed in a generic enough manner to accommodate.

## 7) References

<https://developer.android.com/training/index.html>

<http://developer.android.com/guide/topics/ui/dialogs.html>

<https://developer.edamam.com/docs>

<http://www.vogella.com/tutorials/AndroidLocationAPI/article.html>

<http://www.vogella.com/tutorials/AndroidSQLite/article.html>

<http://www.w3schools.com/sql>