SOFE4640 Mobile Application Development Assignment 2 - Luke Hruda - 100654666

https://github.com/LukeHruda/LocationFinderApp

Introduction

This report will cover how I used Geocoding in conjunction with Latitude and Longitude grid coordinates to populate an SQLite database on the Android application. I was then able to use intents and RecyclerViews to modularly display the data from the database.

Application Install / Database Populating

To fulfil the first two requirements of the assignment, I used the CSV file provided publicly by the Canadian government that contains the list of Latitude and Longitude coordinates for the courthouses of Ontario. The file can be found at:

https://open.canada.ca/data/en/dataset/849b516b-e355-48e9-89f0-9190598814c9

I slightly modified the file to only be 50 rows of lat and long coordinates. This file is saved in the res/raw directory. The main activity immediately check the Shared Preferences of the app for a boolean variable called “FrstTime”. If the boolean is false or returns null, the app will read the CSV file using the custom class object and add all 50 entries to the database. This works by looping through the pairs of latitude and longitude coordinates and parsing them through the GeoCoder Object to get the Street Address, it then adds them to the SQLite Database. When this runs, it then creates the “FrstTime” variable and sets it to true, adding it to the Shared Preferences ensuring this code only runs once on app initial start up after install.

The database is set up with the four required columns of: ID, which is set to auto increment as an integer; Address, which is a String; Latitude, which is a Real data type and Longitude which is also a Real data type.

Main Page

The main page / main activity of the application is a simple UI. It contains a Linear Layout, which has a text input field and the search button, along with another button for viewing and modifying all the entries in the database. These elements are layered in a constraint view. To better understand how the Query Functionality works, I will first discuss the view all and modify functionality of the application.

View All Locations / Add, Delete, Update Locations

Upon selecting the “View and Update Locations” button on the main activity, the application switches to the ViewAll activity. This activity layout contains a recycler view and a button for adding new locations to the database. The RecyclerView populates by getting all the

data entries in the database passed to it through the arraylists initialized in the ViewAll activity, it then loops through these creating a cardview named “view_row.xml” that contains the cardview element, and three text boxes for the Address, Lat and Long. This allows for scrolling through the list of addresses found in the Database.

The add location button, brings up a simple input menu via intent. The menu allows the user to input Lat and Long Coordinates into the menu. Latitude must be between -90 and 90 while Longitude is between -180 and 180. Once both values are entered and the user hits save, the application will take the lat and long coordinates and parse them into the Geocoder to determine the address and then add them to the database.

To update or delete an entry, each unique item in the database has an on click listener attached to it. So clicking on an entry will bring up the Update Location page that looks very similar to the Add Location page, except here the user will be able to modify the actual address determined by the GeoCoder in the Add Location functionality. So they can change the address from “1349 Simcoe St N.” to “Simcoe St. McDonald’s” if they so desire. Once the user hits submit, the database will run the update query that updates the database entry based on its ID value. Next to the update button is also the delete button, that will initialize an AlertBox window to confirm if the user wants to infact delete that address.

Lastly, on the view all page, there is a menu item for deleting all the entries from the database. When the user selects this an alert box asks for confirmation then deletes all upon yes.

Query Location

The search function works on the same level as the view and update one does with a slight difference. The user can input any string into the Address box above the “Search For Location” button. When the user presses the button for Search, the application gets the value entered in the input box and adds it as an extra to the intent. The activity queries the SQLite database for all entries where the Query String is found as a substring of the address. From here, the Activity works almost identically to the View All activity. Where a RecyclerView is used to display all the entries found by creating a layout full of cardviews. Each Cardview can be clicked to bring up the Update functionality again. The only key differences are that on this page, there is no Add Note Option, nor is there a delete all option.

Conclusion

In conclusion I was able to use RecyclerViews, SQLite Databases and Geocders in conjunction with Views, Activities and Intents to create an Application that can determine Addresses based on their Latitude and Longitude and store that information in the Database. These entries can then be queries, deleted, updated or new ones can be created.