ANDROID DEVELOPMENT FOR HOUSING ANALYTICS-HOMERUN

by

Saurabh Deshpande

A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Computer Engineering

Committee:

[illegible]	Dr. Houman Homayoun, Thesis Director
[illegible]	Dr. Setareh Rafatirad, Committee Member
[illegible]	Dr. Xiang Chen, Committee Member
[illegible]	Dr. Monson H. Hayes, Department Chair
[illegible]	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering
Date: 12/06/2018	Summer Semester 2018 George Mason University Fairfax, VA

Development for Housing Analytics-HomeRun

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

by

Saurabh Deshpande
Bachelor of Science
University of Pune, 2016

Director: Houman Homayoun, Professor
Electrical and Computer Engineering Department

Spring Semester 2018
George Mason University
Fairfax, VA

DEDICATION

This work is dedicated to my parents.

ACKNOWLEDGEMENTS

This thesis could not have been a success without the constant support and guidance of my thesis advisors, Dr. Setareh Rafatirad and Dr. Houman Homayoun. I worked with Dr. Rafatirad continuously dealing with various use case scenarios and strategies to deal with real problems that came during this thesis work. She was constantly encouraging me and provided me her valuable time to discuss the content and design of this HomeRun Application for Android. She was always there when I needed support in UI design and provided resources that helped me to build user-friendly UI. Dr. Homayoun offered me continuous encouragement and guidance. He helped me manage all the modules and guided me for the integration of these modules. He has also helped me in a lot of other academic programs as well. Finally, thanks go out to the Fenwick Library for providing a clean, quiet, and well-equipped repository in which to work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Operating Systems	OS
Application Programming Interface	API
Original Equipment Manufacturer	OEM
Android Runtime	ART
Just In Time	JIT
Ahead Of Time	AOT
User Interface	UI
Uniform Resource Locator	URL
Java SE Development Kit	JDK
Software Development Kit	SDK
Structured Query Language	SQL
Return On Investment	ROI
JavaScript Object Notation	JSON
Extensible Markup Language	XML
Contextual Action Bar	CAB

ABSTRACT

DEVELOPMENT FOR HOUSING ANALYTICS-HOMERUN

Saurabh Deshpande, M.S.

George Mason University, 2018

Thesis Director: Dr. Houman Homayoun

Co-Advisor: Dr. Setareh Rafatirad

Real estate is a huge domain, and people think of selling, renting and purchasing the property which includes huge investments. The most important part comes while dealing with the property is “Searching”. There are millions of homes spread across US states and more and more people are hunting for properties, which suit their needs and pockets. Searching for a property has become easier because of the Internet Revolution. People have started listing the properties online. Now, as smartphones have gained huge popularity as they are handy and has access to the Internet on the go. There is an incredible growth in the operating systems that support mobile devices. Android OS and IOS are the leading operating systems that have millions of applications for their users. More than 75% of the mobile devices support Android. [1]

Although these smartphones can be used to browse the web for searching the property, again one has to go do the tedious job of searching the property. Even if, one finds an interesting property listed they have to go to multiple websites to get all the

information about that property and there is much more you do while considering your next home. Example, Google Maps for the location and navigation, different state websites listing corresponding state properties, list down all the details of the properties, locality inspection.

To ease this process, an Android application ‘HomeRun’ which is a one-stop destination that provides users with all the necessary features integrated. One can search a property based on the location, which may include for a specific state, city or zip code. Users can also use detailed search option where they can feed in the desired number of bedrooms, price, area etc. and get a list of matching properties. Users can compare two or more properties using the feature of spider graph and also look for zones with residential areas with heat maps. Apart from all these features, the user has been provided with the ability to get directions to that property. [1]

The application is developed in native Android development environment using JAVA.

INTRODUCTION

Major OEMS in the world are using Android as the operating systems for their phones, wearable's and other devices. Almost of 190 countries around the world have major market share captured by Android. It is exponentially growing and million and millions of users are using the Android phones and applications developed for this platform.

Every application has a purpose and so does this application "HomeRun for Android Phones" too has a purpose. This helps people to search for nearby properties to their interests. Starting the application users can feed in the required parameters and then this application will search the properties accordingly and will display the result. There are millions of homes across US states. Real estate property listing is a need nowadays. To cope up with this need, we are developing an android application. Which provides affordable home buying opportunities to the user across US states. Users can browse US homes for purchase/rent by comparing their details. [1]

The development of such application requires a lot of planning, considerations and tuning in the following areas: [1]

1. Designing and prototyping the User Interface of different modules.
2. Granting the application with necessary permissions.
3. Database Integration.

4. Implementing the search functionality using query URLs.
5. Populating the results and linking the listed houses with Google Maps.
6. Implementing the filter option.
7. Providing with all Google Maps features like navigation, estimated time etc.
8. Implementing features like heat maps, show on map-nearby hospitals, schools, and restaurants.
9. Displaying the spider graph for comparison.
10. ROI Calculator

This research is oriented in creating an android application which deals with huge database and the application is developed with intensive research on data integration and methods to display information in various formats. The application is developed considering the principles and patterns of mobile interface design.[2]

Need for HomeRun on Android Platform

Handy devices have been around us for last 3 decades. The first popular operating system was the Symbian which dominated the market till 2007 until iPhone was launched with their own iOS. Android OS backed by Google Inc was launched which is an open source platform in 2008.

Android is constantly growing, and many cell phones manufacturing companies have accepted Android as their prime OS for the devices they produce. These devices range from all sizes- smartphones, tablets to TVs. Considering this growth, all the applications existed for the previous OS like Symbian and iOS are developed for Android platform. Now, Android is the leading OS for which most of the applications are built for.

April 2017 report from StatCounter states that android has overtaken the big giant in mobile operating system platform Microsoft considering the internet usage and popularity.[3] Now android has over 3 billion monthly active users growing up from 2 billion in May 2017.[2][3]

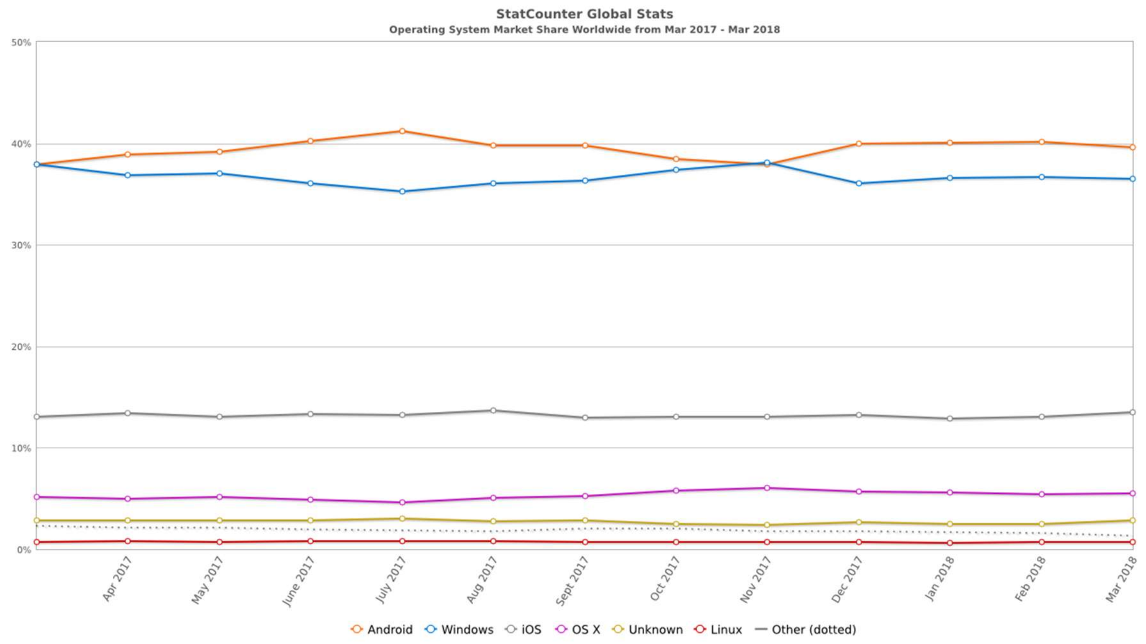


Figure 1: Operating System Market Share Worldwide from Mar 2017 – Mar 2018[3]

According to the Gartner reports, worldwide Q2 smartphones sales were dominated by Android Smartphones devices (see Table 1). Android leads the sales game with 84.8% of the total sales compared to 14.4% of Apple iPhone devices.[2][4]

Table 1: Worldwide Smartphone Sales to End Users by Operating System in 2017 (Thousands of Units)[4]

Operating System	**2017 Units**	**2017 Market Share (%)**	**2016 Units**	**2016 Market Share (%)**
Android	1,320,118.1	85.9	1,268,562.7	84.8
iOS	214,924.4	14.0	216,064.0	14.4
Other OS	1,493.0	0.1	11,332.2	0.8
Total	**1,536,535.5**	**100.0**	**1,495,959.0**	**100.0**

Source: Gartner (February 2018)

Related Work

With the increase in demand for properties for sale and purchase, as discussed above a lot of platforms have been developed using the web and mobile technologies. Few of the established platforms in the real estate domains are Zillow, Trulia, and Realtor.com. The following describes how HomeRun stands against its competitors and how uniquely is it trying to provide a one stop application for property search.

Zillow

Zillow is an online real estate database company which happens to have an android/ios application. Although the Zillow Android application has very unique and useful features there are some differences which make HomeRun stand out.

The first screen of the Zillow application displays map and the dots on the map represents properties near your current location. No filters are applied and all the houses are displayed on a map. The user has to toggle between the list of the houses and the map. Simultaneous viewing of the list and map is missing. Whereas in HomeRun users can directly locate the property listed in the list view and simultaneously the map changes marker position to the location of selected property from the list view. This provides users with a better understanding of locations of different properties in the lists.

As the feature of instant viewing is missing, no nearby schools, restaurants, and hospitals are mapped along with the location marker on the map. Although there is an option to get nearby schools, it only lists 3 nearest schools to that location. This doesn’t provide the user with all information regarding nearby schools. Providing more options allows users to give preferences to those options.

In Zillow, the user has to go through multiple screens before they see the ‘navigate to’ options. Selecting the list expands the list view to provide all the information about that property on the same page. The user has to scroll through the page to get to the directions option. HomeRun provides an elegant yet simple UI and offers easy access to all the features. For navigating to a specific property location, the user just has to select the list item that is the property and map changes instantly to show the location and options such as show on Google Maps and start navigation.

Zillow has almost 5 features which state more information about the property stuffed onto just one screen. Users might require time to get familiarized with all this information at once. On the other hand, HomeRun provides its features in a different way. On long select, on maps etc.

Performance wise, Zillow application has implemented a pagination for next 100 items including the images. HomeRun has a small performance boost in this regard as pagination limit is 10.

Zillow plots the location markers at a specific zoom level on the maps. If the zoom level is decreased the markers are not visible. No clustering is performed while HomeRun plots the marker independent of the zoom level. Also, Zillow fails to provide any comparing options between two or more property details, while HomeRun provides with an option such as Spider graph, a visual tool for comparing house parameters on a spider web-like graph.[5]

Trulia

Trulia is an online residential estate site which lists properties for sale and rent. Similar to HomeRun it also provides tools and information needed in the home search.

Trulia does cover a lot of drawbacks from Zillow but introduces its own. The start screen displays information about houses, rather lists houses from the unrelated area. Example, a user living in Virginia gets feed listed with options from San Francisco. Map and lists are present on different screens.

Zoom level matters while displaying the markers on the maps. A problem similar to Zillow exists, where decreasing the zoom levels makes the markers invisible on the map.

All these limitations have been taken care in HomeRun, where simple UI is designed to provide important information to the users with a couple of taps.

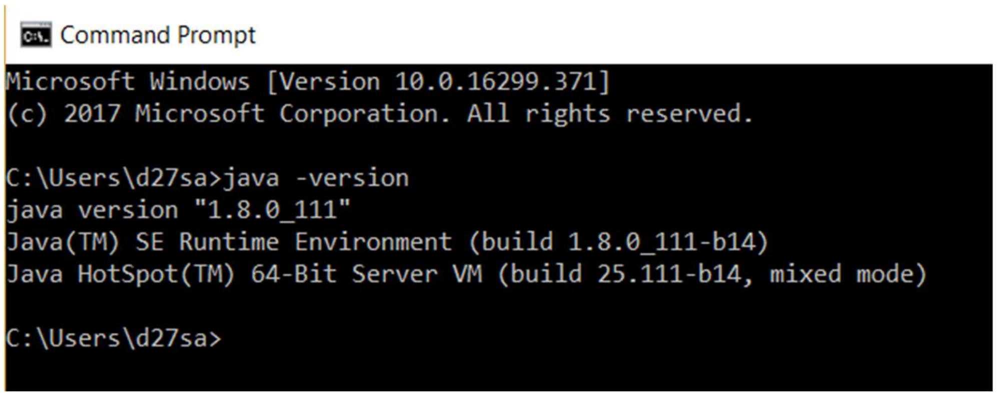
Other

There are other applications like Realtor.com, Homes.com, Apartments.com etc. which serve a similar purpose to HomeRun but fails to identify the above-mentioned drawbacks.

ANDROID ENVIRONMENT SETUP

To start with the Android application, we need the Java Development Kit. Android studio is like the workshop but JDK is the toolbox we need for Android programming. JDK is a software that allows the computer to read and work with Java Programming Language. This is sort of fundamentally different from a software program because it's more like an upgrade to your computer.[6]

Checking the availability of Java on your machine: Using the terminal, we can find we can check whether the machine has the Java Developer Kit, version 7 or greater. The command used to check this is java –version. This gives the current java version.



```
Microsoft Windows [Version 10.0.16299.371]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\d27sa>java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)

C:\Users\d27sa>
```

Figure 2: Java Version

If the JDK is not available, or the version is lower than 7, then download the JDK from the Oracle site.

We also need the IDE that is the Android Studio (the workplace). It can be downloaded from the Android Studio website. Android SDK stands for software development kit that helps developer to create applications for Android platforms. Android SDK includes application development tools, sample projects with source codes and required libraries to build an Android application. A virtual mobile device running on the computer is the Android emulator in which multiple existing mobile models are loaded. One can chose any of the specific model. [6]

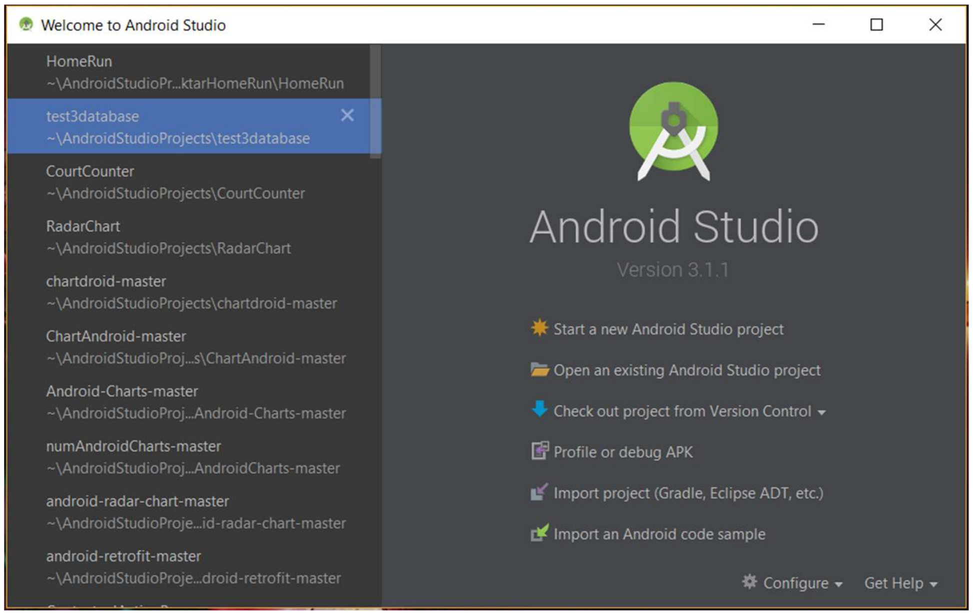
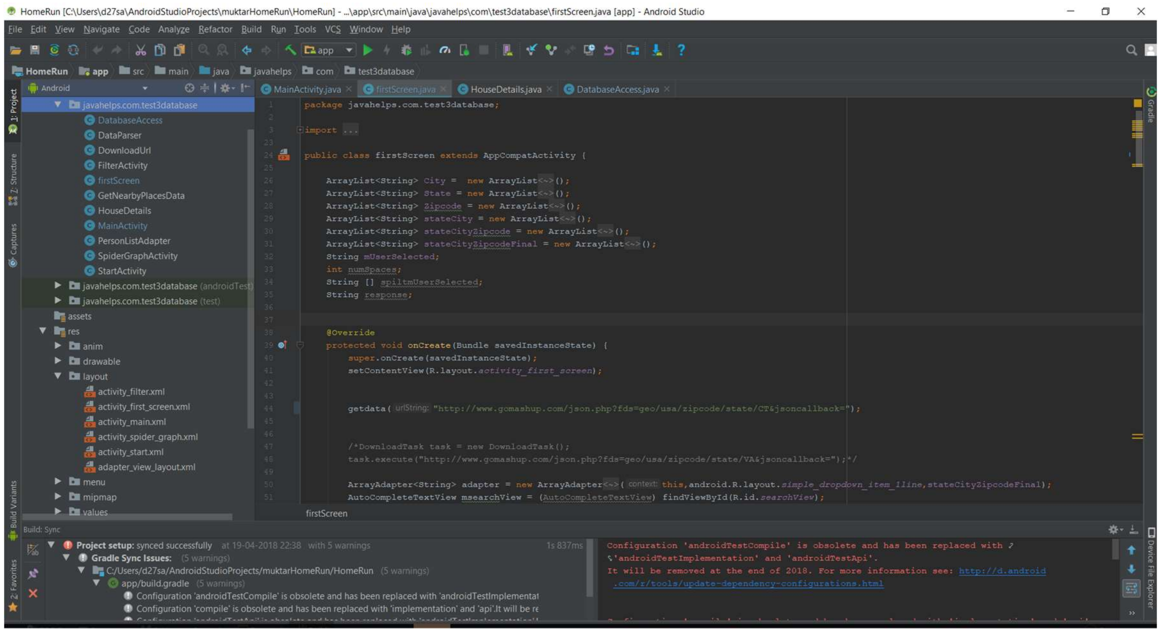

Welcome to Android Studio
HomeRun
~\AndroidStudioPr...ktarHomeRun\HomeRun
test3database
~\AndroidStudioProjects\test3database
CourtCounter
~\AndroidStudioProjects\CourtCounter
RadarChart
~\AndroidStudioProjects\RadarChart
chartdroid-master
~\AndroidStudioProjects\chartdroid-master
ChartAndroid-master
~\AndroidStudioProj...s\ChartAndroid-master
Android-Charts-master
~\AndroidStudioProj...Android-Charts-master
numAndroidCharts-master
~\AndroidStudioProj...AndroidCharts-master
android-radar-chart-master
~\AndroidStudioProje...id-radar-chart-master
android-retrofit-master
~\AndroidStudioProje...droid-retrofit-master
Android Studio
Version 3.1.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

Figure 3: Android Studio



HomeRun [C:\Users\d27sa\AndroidStudioProjects\muktarHomeRun\HomeRun] - ...\app\src\main\java\javahelps\com\test3database\firstScreen.java [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HomeRun app src main java javahelps com test3database
Android
MainActivity.java firstScreen.java HouseDetails.java DatabaseAccess.java
javahelps.com.test3database
DatabaseAccess
DataParser
DownloadUrl
FilterActivity
firstScreen
GetNearbyPlacesData
HouseDetails
MainActivity
PersonListAdapter
SpiderGraphActivity
StartActivity
javahelps.com.test3database (androidTest)
javahelps.com.test3database (test)
assets
res
anim
drawable
layout
activity_filter.xml
activity_first_screen.xml
activity_main.xml
activity_spider_graph.xml
activity_start.xml
adapter_view_layout.xml
menu
mipmap
values
firstScreen
Build: Sync

```
package javahelps.com.test3database;

import ...

public class firstScreen extends AppCompatActivity {

    ArrayList<String> City =  new ArrayList<~>();
    ArrayList<String> State = new ArrayList<~>();
    ArrayList<String> Zipcode = new ArrayList<~>();
    ArrayList<String> stateCity = new ArrayList<~>();
    ArrayList<String> stateCityZipcode = new ArrayList<~>();
    ArrayList<String> stateCityZipcodeFinal = new ArrayList<~>();
    String mUserSelected;
    int numSpaces;
    String [] spiltmUserSelected;
    String response;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_first_screen);


        getdata( urlString: "http://www.gomashup.com/json.php?fds=geo/usa/zipcode/state/CT&jsoncallback=");


        /*DownloadTask task = new DownloadTask();
        task.execute("http://www.gomashup.com/json.php?fds=geo/usa/zipcode/state/VA&jsoncallback=");*/

        ArrayAdapter<String> adapter = new ArrayAdapter<~>( context: this,android.R.layout.simple_dropdown_item_1line,stateCityZipcodeFinal);
        AutoCompleteTextView msearchView = (AutoCompleteTextView) findViewById(R.id.searchView);
```

```
Project setup: synced successfully   at 19-04-2018 22:38   with 5 warnings   1s 837ms
  Gradle Sync Issues:  (5 warnings)
    C:/Users/d27sa/AndroidStudioProjects/muktarHomeRun/HomeRun  (5 warnings)
      app/build.gradle  (5 warnings)
        Configuration 'androidTestCompile' is obsolete and has been replaced with 'androidTestImplementat
        Configuration 'compile' is obsolete and has been replaced with 'implementation' and 'api'.It will be re
```

```
Configuration 'androidTestCompile' is obsolete and has been replaced with
'androidTestImplementation' and 'androidTestApi'.
It will be removed at the end of 2018. For more information see: http://d.android
.com/r/tools/update-dependency-configurations.html
```

Figure 4: Android Studio with HomeRun Application

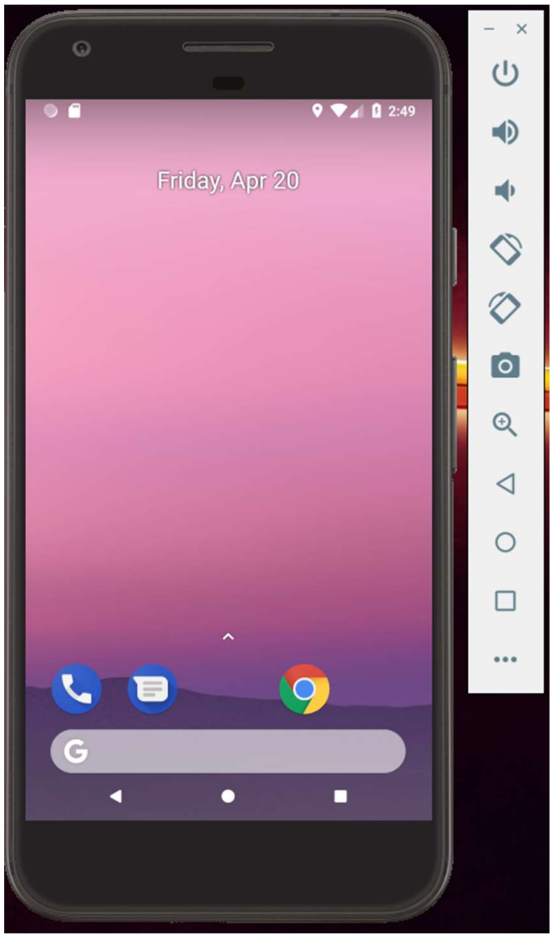


Figure 5: Emulator (Pixel XL API 27)

DESIGN SPECIFICATION

To standardize the development of applications, there are established patterns that the developers can use to accelerate and provide user with a better UI. Abiding by these patterns both the users and developers are benefitted from the structure of the application.

Design Pattern Architecture

The question here is why it is necessary to have a design pattern architecture. Basically, having a proper architecture helps to achieve simplicity, testability and low-cost maintenance. Defining a clear and simple role for each component in the app helps to maintain simplicity. Implementing this will help to know what the class does and what is inside it. To make sure that the code is written is Unit Testable, we have to write testable codes.[7]

Following are few of the popular patterns:

1. MVC (Model – View - Controller)
2. MVP (Model –View - Presenter)
3. MVVM (Model – View - ViewModel)
4. Clean Architecture

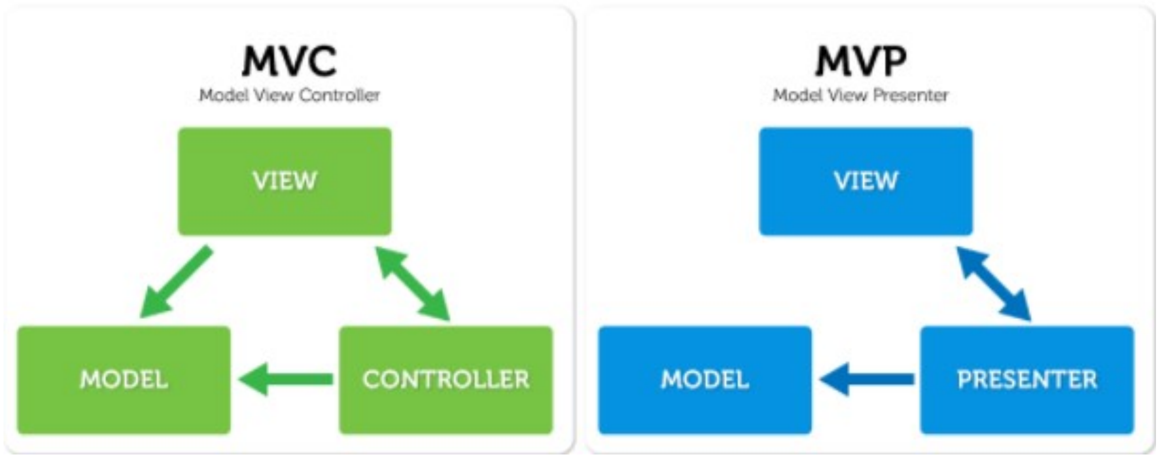


Figure 6: MVC vs MVP

MVP model for HomeRun

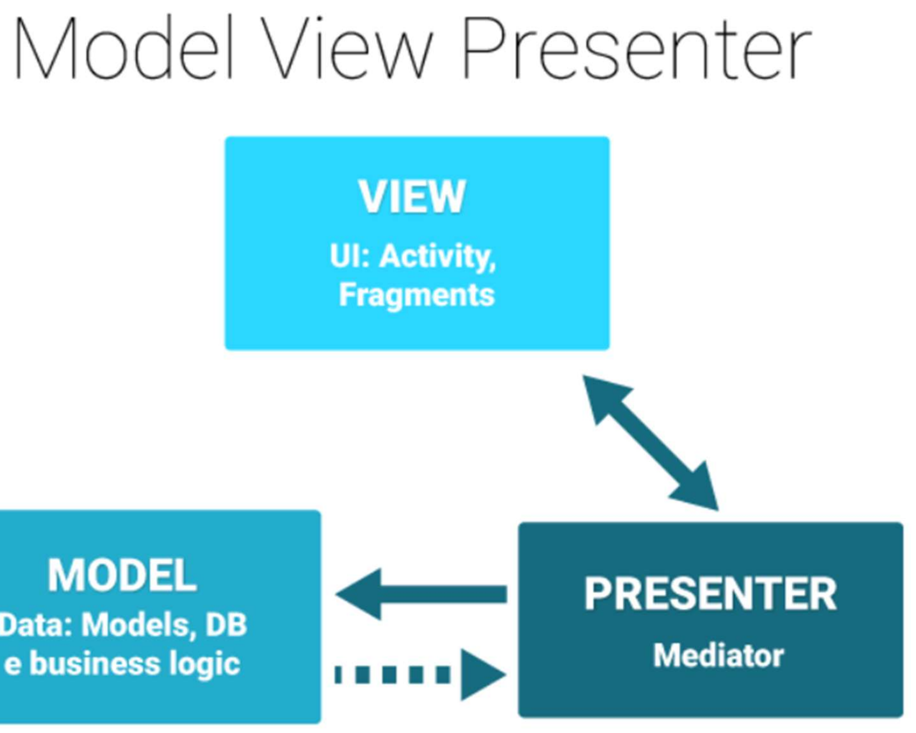


Figure 7: Model View Presenter

View = a passive interface that displays data to the Presenter. It also routes the user actions to the presenter. Activity, Fragment, or View does the job of View module.

Model = this layers has the logic or key to data creation, modification or storage. Model can be represented in android by remote server or by database API.

Presenter = It can be considered as a middleman whose role is to transfer the data from the Model and display the same in the view. [7]

Important points of MVP are:

- The view does not have access Model.
- The presenter is attached to single View.
- The view can be considered as transporter where it gets data from the action of user and supplies the same to presenter. It has no concerns with the actions operated by the presenter on the data. [7]

Component Overview

There are four major components included in the infrastructure of the application. They are:

1. Database
2. Interactive Visualization Tools
3. Data Exploration
4. Prediction Model

Figure 11 shows how these modules are linked with each other.

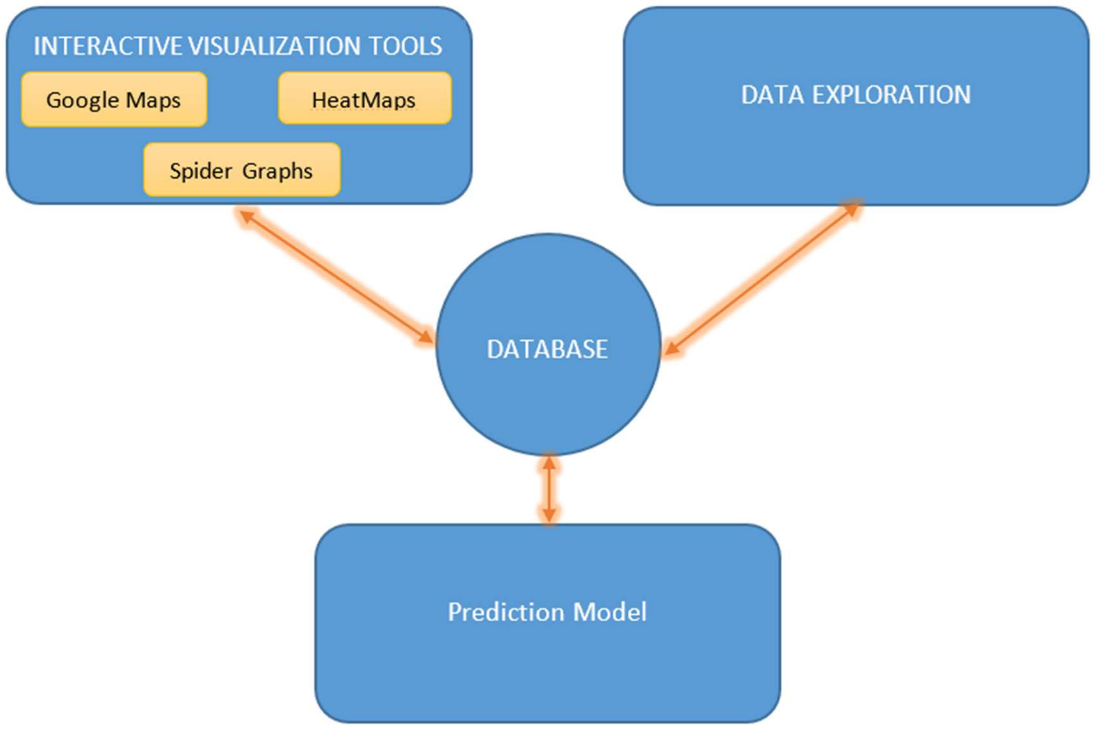


Figure 8: Model View

High-level components are very essential for any project to get a clear understanding of the purpose and the various high-level modules involved. The design of high-level components forms the backbone of the project upon which the detail components stem out.

Database

This is the core module for the application. All the data that is circulated in the application is stored in the database module. This database includes all the house details for all the states in the US. There are many ways of storing the data in an Android application. SQLite is the database supported by android. The structured query language is the database in which the data is structured into columns and stored in a format.

We can create multiple tables and relate them using the keys. For HomeRun, the initial development was done in SQL database but considering the data set size the database was migrated to MongoDB server.

SQL Database: SQL stands for Structured Query Language, is a domain specific language which is designed for managing data held in a relational database management system. Compared to old methods of read/write APIs, SQL offers the concept of accessing many records with a single command and eliminates the need to specify each record with an index.

The housing data was stored in the application. Sample data of size less than 3mb was used. The data was acquired from the Zillow database in CSV format. This CSV format was changed to db which is adaptable with the android studio. External dataset files were integrated with the application under the assets folder. This included address, state, city, zip code, price, latitude, longitude, number of beds, number of baths, property details columns. All the required queries were done on this dataset.

MongoDB: It is a NoSQL database program which uses JSON like documents with schemas. MongoDB was used to set up the database on university local server named HH-

Xeon2. The data was stored in collections based on the states. API was developed to access the data.

Interactive Visualization Tools

HomeRun needed to present a large amount of data to the user. Dealing with huge chunks of data in a plain old fashion way doesn't abide by the design patterns of android application. There are many ways of presenting the data in a systematic manner. HomeRun uses ListViews, Google Maps, HeatMaps and Spider Graph.

ListView: According to the android developer guide: ListView is a view group that displays a list of scrollable items.[8] The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database query and converts each item result into a view that's placed into the list.

The adapter used is configured to show multiple items from the database like the address of the house, state, city, zip code, and price. This provides the user with sufficient information about the house in a proper list manner.

Google Maps: Google provide Google Maps Android API to add maps using the Google Maps data in the android application. This Google Maps API is designed to take care of Maps servers, managing data transfers and downloads, displaying items on maps and responding to the gestures done on maps.[9]

In the HomeRun, markers were added to the Google Maps in the application, where these markers represent the location of the houses in the list view. It provides a great way of representing the location of the property. Additional functionality is integrated into the Google Maps where the user can see the nearby places on the selected location. Nearby places include the hospitals, schools, and restaurants.

HeatMaps: Google Maps Android HeatMap Utility provides HeatMaps that are useful for representing the distribution and density of data points on a map. Heat Maps providers users with a better understanding of the distribution and depicts the intensity (relative) of the data points on the Google Maps. It uses color variations TO SHOW THE data points instead of using the markers.[10]

HomeRun uses to depict the areas with higher residential availability. The number of available properties depicts the intensity of heat of that region.

Spider Graph: Radar Chart/Spider Graph is basically a graph to display data involving three or more variable qualities in form of a 2-d chart where variables are represented on the axes of the graph starting from the same point. Spider graph is implemented in the HomeRun application so that users can compare house details on a graph to get a better understanding of the ROI i.e return on investment.

Data Exploration

It will be a tedious task for the user if he has to go through huge lists of houses in a specific state or zip code. Options to filter out these lists according to user requirement should be available. Data Exploration model allows users to specify their filters on the houses they are interested, like price, area, number of baths, number of beds, state, zip code, city etc.

Predictive Model

A model is developed which aims for predicting the rent and price estimation of the houses. This feature is facilitated in the Mortgage calculator. Models are developed to predict the prices and estimated rent of the houses in a particular zip code. This provides the user with a better understanding of trends in the real estate world.

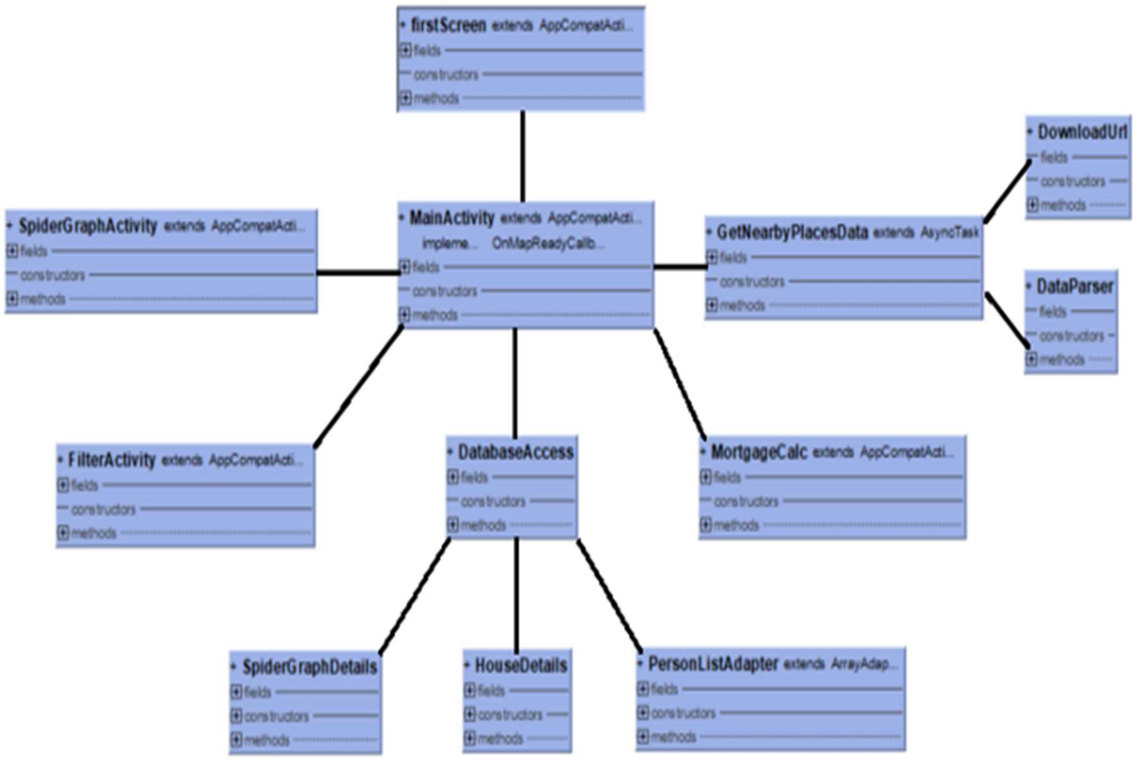
Features

Following are the features of the HomeRun Android application:

1. Filtering option: Houses are filtered according to the requirement specified by the user.
2. Map View: Listed houses are displayed on Google Maps.
3. Navigation: Navigation from the current location to selected house location.
4. HeatMap: A heat map depicts the intensity of residential properties in an area.
5. Graph: A spider graph to compare house details.
6. Nearby Places: For each house selected,
7. Nearby schools, hospitals, and restaurants are marked.
8. Mortgage Calculator

IMPLEMENTATION

Application Workflow



firstScreen extends AppCompatActi...
MainActivity extends AppCompatActi... impleme... OnMapReadyCallb...
SpiderGraphActivity extends AppCompatActi...
GetNearbyPlacesData extends AsyncTask
DownloadUrl
DataParser
FilterActivity extends AppCompatActi...
DatabaseAccess
MortgageCalc extends AppCompatActi...
SpiderGraphDetails
HouseDetails
PersonListAdapter extends ArrayAdap...
fields
constructors
methods

Figure 9: Application WorkFlow Diagram

Implementation Details

First Screen Search

The first screen of the application provides users with a facility of searching the database of the houses across all US states. The User Interface of the first screen contains a TextView which states the user to enter state, city or zip code. The element in the UI is an Autocomplete TextView. This AutoComplete TextView is where user types in the keyword associated with the area they want the property to search. The keyword can be a state, zip code or a city. Entering any keyword will trigger the AutoComplete TextView with suggestion related to the keyword entered.[11]

Example 1: Entering State

User Entered Keyword: CT

Suggestions: CT

CT STAMFORD

CT EAST BERLIN

And so on...

Example 2: Entering City

User Entered Keyword: Sta

Suggestions: CT STAMFORD

CT STAFFORD

CT STAFFORD SPRINGS

CT STAFFORDVILLE

CT STAFFORD 06075

And so on...

Example 3: Entering Zipcode

User Entered Keyword: 060

Suggestions: CT AVON 06001

CT BLOOMFIELD 06002

CT WINDSOR 06006

And so on...

The above-mentioned examples indicate the pattern implemented in the suggestions. And the suggestions works same for all 50 states in the US. The pattern tried to achieve here is State or State-City or State-City-Zip code. According to the keyword entered, the results are populated in any of the three patterns described. The last element in the UI of the first screen is the Search button which queries the database according to user input and displays the information in the next activity.

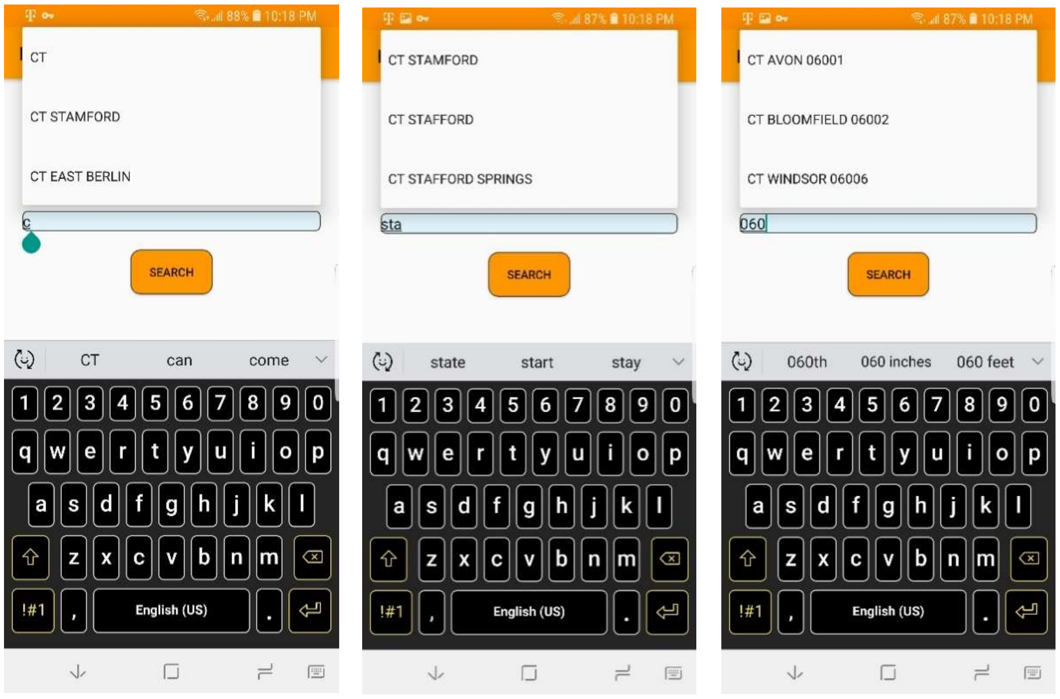
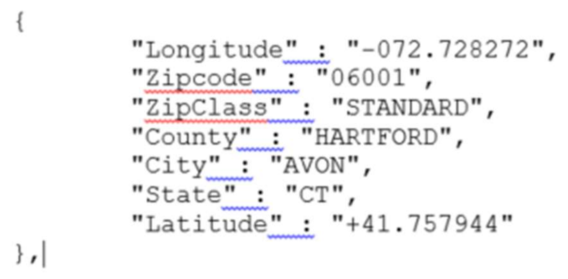


Figure 10: AutoComplete Suggestions for state city and zipcode

Logic For AutoCompletion

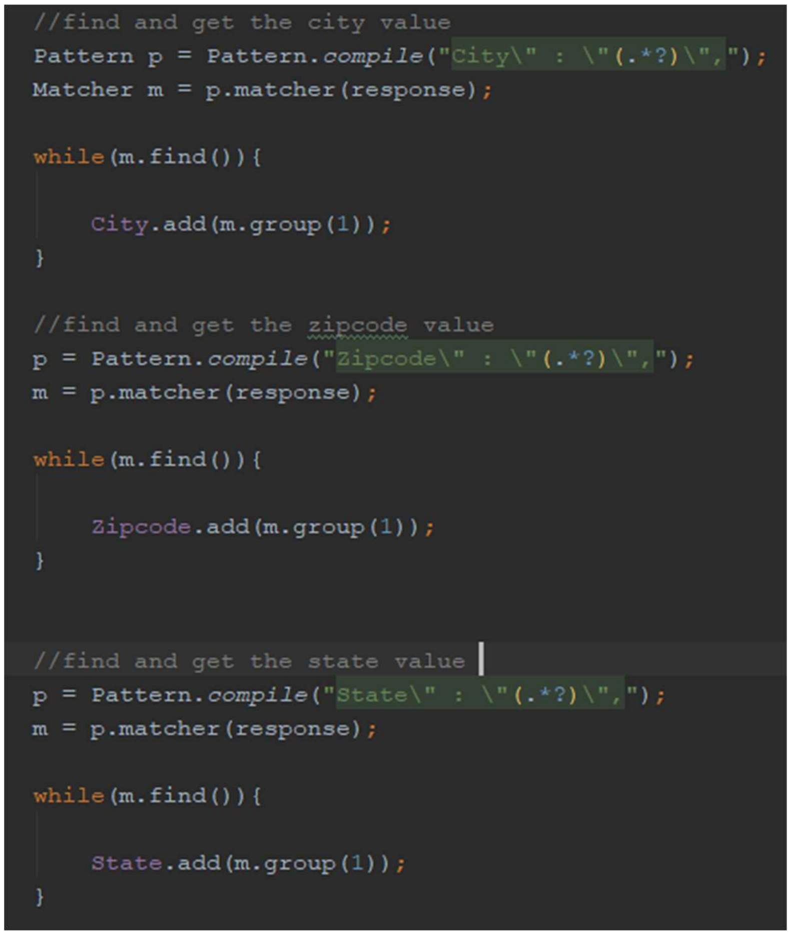
A database which has all the states, cities and zip codes information is stored within the application. This database is created when the application is started for the first time. This database is derived from the URL: "http://www.gomashup.com/json.php?fds=geo/usa/zipcode/state/<state>&jsoncallback=" . An array of string which has all postal codes for all 50 states is passed to this URL to get the response. The response contains information about all the cities and zip codes included in that state. One of many JSON objects looks like:



```
{
        "Longitude" : "-072.728272",
        "Zipcode" : "06001",
        "ZipClass" : "STANDARD",
        "County" : "HARTFORD",
        "City" : "AVON",
        "State" : "CT",
        "Latitude" : "+41.757944"
},
```

Figure 11: JSON object from gomashup URL

As for populating the AutoComplete TextView, the fields required are state, city and zip code, other fields such as longitude, latitude are eliminated. This is achieved by using the java.util.regex.Pattern and java.util.regex.Matcher libraries.



```
//find and get the city value
Pattern p = Pattern.compile("City\" : \"(.*?)\",");
Matcher m = p.matcher(response);

while(m.find()){

    City.add(m.group(1));
}

//find and get the zipcode value
p = Pattern.compile("Zipcode\" : \"(.*?)\",");
m = p.matcher(response);

while(m.find()){

    Zipcode.add(m.group(1));
}


//find and get the state value
p = Pattern.compile("State\" : \"(.*?)\",");
m = p.matcher(response);

while(m.find()){

    State.add(m.group(1));
}
```

Figure 12: Code Snippet-Use of Pattern and Matcher

ListView with custom Adapter

Querying onto the database retrieves thousands of house details. An efficient way should be used to display the information in proper manner. If we chose to use to display this retrieved information using Linear Layout a lot of memory is used up to create specific objects for all the retrieved items. We have another option that can be used to display large amounts of information memory efficiently.

As the memory is a limited resource ListView helps in managing memory efficiently. When there is a lot of information to be shown in a list form, ListViews are used with ArrayAdapters. Shown below is a comparison of memory usage in listing numbers 1 to 1000 using Linear Layout and with ListView. [8][12]

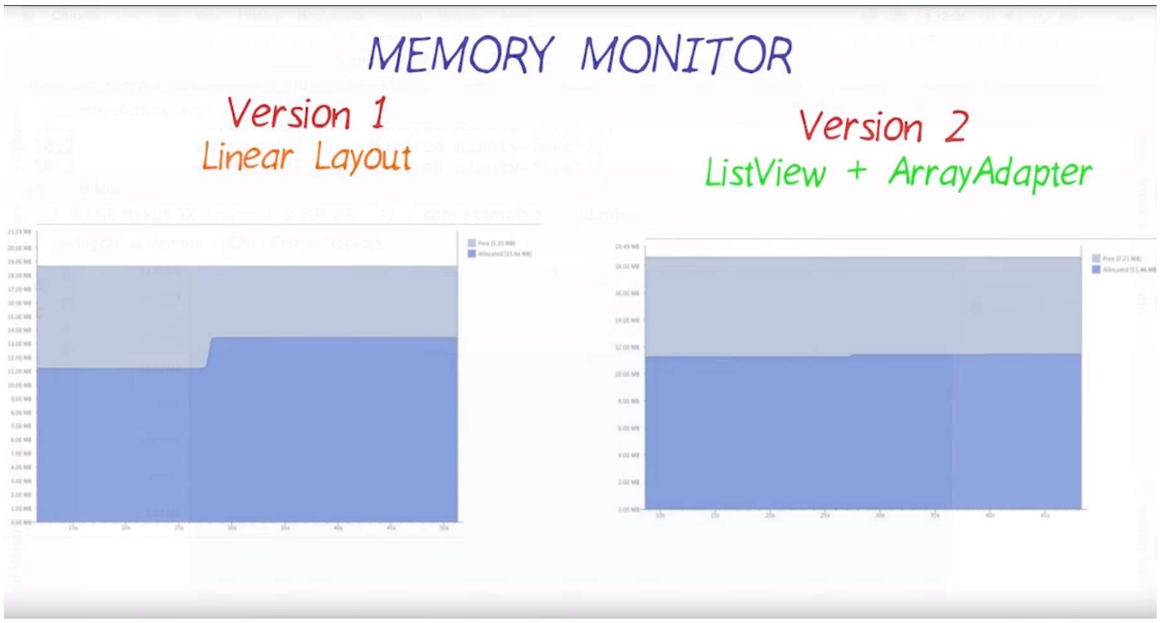


Figure 13: Memory Monitor (Linear Layout vs. ListView with ArrayAdapter)[12]

It is clear from the above figure that memory consumption using Linear Layout is more that compared to the ListView with ArrayAdapters.

ListView and ArrayAdapter

Although there are a lot of list items that have to be displayed in the ListView all the items don’t have to be fetched at the same time as only a fixed limited number of item views are displayed in the ListView. This is achieved by using the ArrayAdapter with the ListViews.

The high-level interaction between the ListView and ArrayAdapter is as follows:

ListView is powered by the ArrayAdapter. Without the adapter, the ListView can be considered as empty container. ArrayAdapter holds on to the set of data that should be shown on the screen. When the ArrayAdapter is associated with the ListView, the ListView will ask how many data items it expects to display and the ArrayAdapter knows this information. In the called method by listView on ArrayAdapter, the input is passed which has the updated position of the list that the user is viewing currently(can be position 0 or position 100). Knowing this position information the ArrayAdapter looks for the internally present data. And then it has the instructions to convert that source of data into a list item view and returns the same.[12]

After scrolling, some of the item views in the ListView are not visible. Those item views are reused by passing them to the adapter. Now the adapter just has to put the data value from the internal data source and don't have to worry about converting into view as it is already an item view. This method of reusing the scrap views is called Recycling. Recycling can be done with ListViews, GridViews etc.[12]

Querying onto database

The user selected input from the first screen is stored as a String and then passed to next activity of the application. The second screen of the application is where the data is displayed in ListView and Google Maps are implemented. Many other features like filtering, spider graph generation, mortgage calculator and heatmaps can be accessed from the second screen.

The user input string is retrieved in this activity from the Bundle and then this information is used to query the database. To make an URL query from the input string, this string is manipulated to extract the state, city and zip code value from it. This done on the basis of the number of spaces present in the input string.

If number of spaces are zero, this suggests the user has only selected state. If the number of spaces is 1, that means the user input string has state and city. If the number of spaces is 2 and the last substring is not an Integer, the user has selected state and city in which the city comprises of two words. And if the last substring is integer along with number of spaces equal to three then the user has selected state, city and zip code.

Using this information in a switch case, query URL is created. The default URL is http://129.174.126.235:5959/api/housing/ and the extracted information from the input string is attached to default URL. This query URL is then passed onto the JsonArrayRequest and details such as Address, state, city, zipcode, and price is extracted from the response.

Custom ArrayAdapters

The ArrayAdapter class available in the android can only hold one data element in the list item view. As the retrieved details from the database have more than one field, custom ArrayAdapter are required to display multiple data elements like Address, State, City, Zipcode, Price in one list view item.

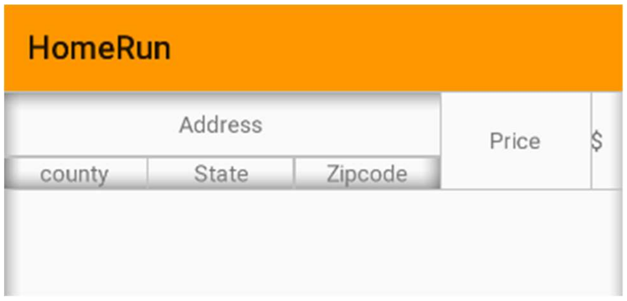
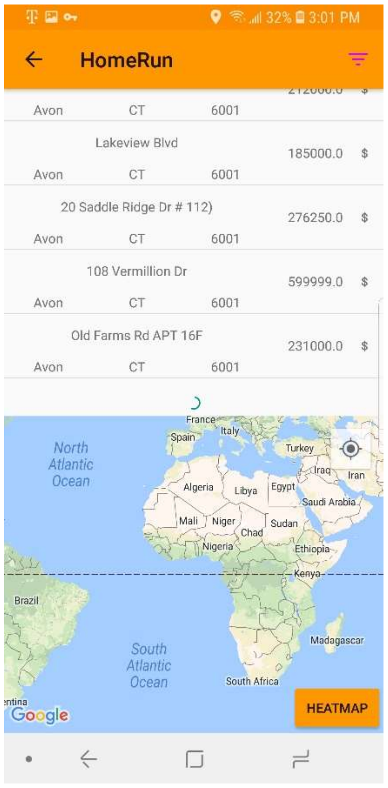


Figure 14: Custom Adapter Blueprint

Pagination

Every time we query the database, thousands and thousands of records are fetched. This records not only contains just the 5 fields (state, city, zip code, address, and price) but also has all the other information. We have to parse through this response to get these 5 fields. Doing this for a large number of records introduces an overhead which introduces a delay to display the information in the ListView. This problem is solved by Pagination.[13]

Pagination is also known as Endless Scrolling or Infinite Scrolling. This feature is common in the content heavy application as it breaks down a list of content into smaller equal pieces, loaded one at a time. Pagination is used in many applications like Facebook, Instagram and many more other. Users continue to scroll and the page at the end of scroll loads more data to the feed. A similar concept is implemented in the HomeRun application where first 10 entries of the query result are loaded and fed to the ListView. [13]



Avon CT 6001

Lakeview Blvd 185000.0 $
Avon CT 6001

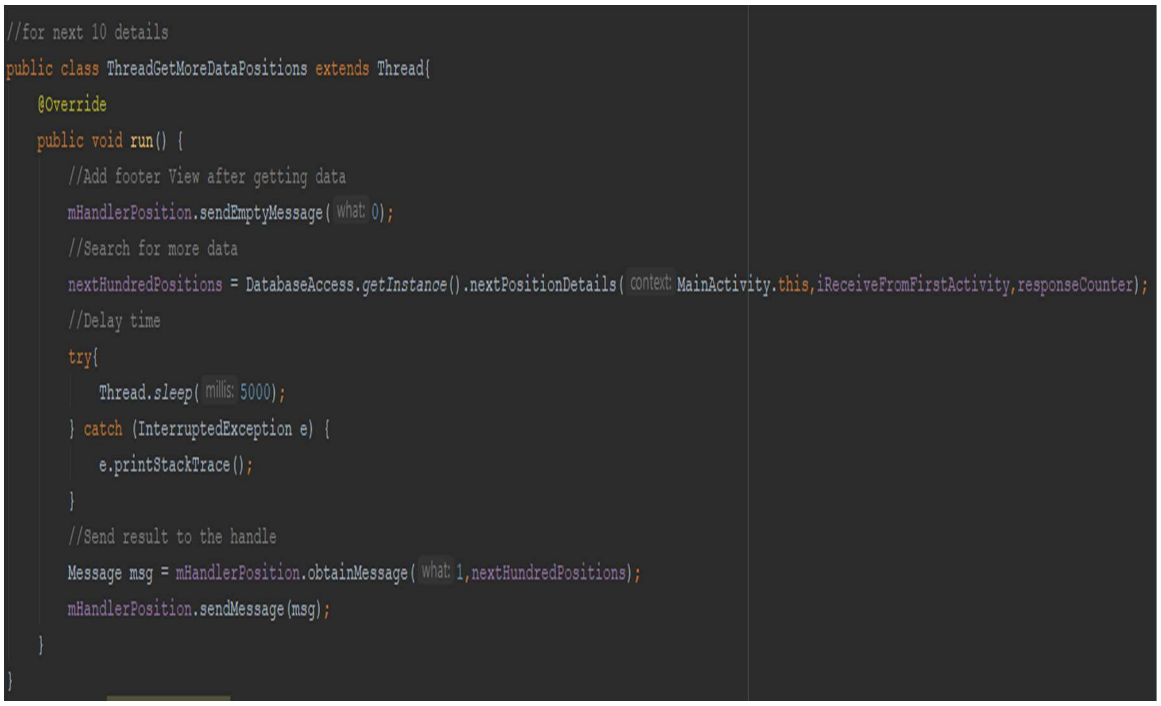
20 Saddle Ridge Dr # 112) 276250.0 $
Avon CT 6001

108 Vermillion Dr 599999.0 $
Avon CT 6001

Old Farms Rd APT 16F 231000.0 $
Avon CT 6001

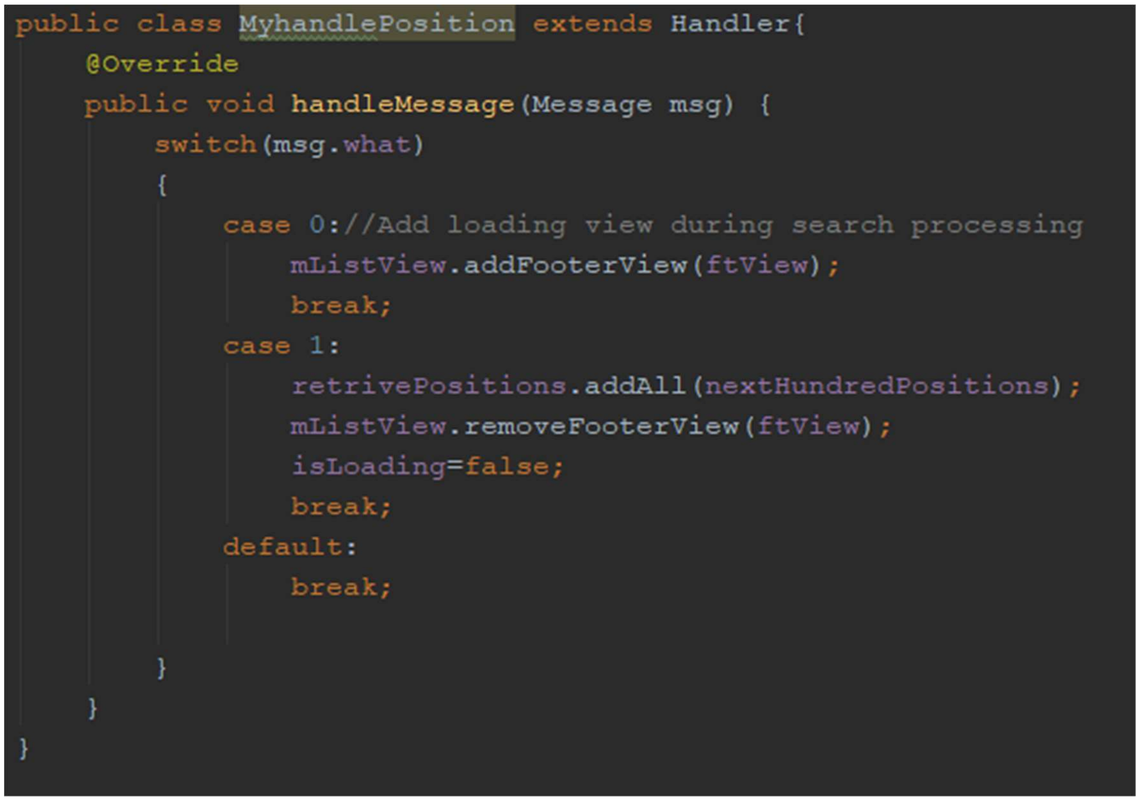
Figure 15: Pagination

When the user scrolls down the list and at the end of these 10 list items, next 10 records are loaded and fed to the ListView. Loading animation is displayed to show the user that more data is loading. A different thread is created which displays the progress dialog and checks with the loading of next 10 entries.



```
//for next 10 details
public class ThreadGetMoreDataPositions extends Thread{
    @Override
    public void run() {
        //Add footer View after getting data
        mHandlerPosition.sendEmptyMessage( what: 0);
        //Search for more data
        nextHundredPositions = DatabaseAccess.getInstance().nextPositionDetails( context: MainActivity.this,iReceiveFromFirstActivity,responseCounter);
        //Delay time
        try{
            Thread.sleep( millis: 5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        //Send result to the handle
        Message msg = mHandlerPosition.obtainMessage( what: 1,nextHundredPositions);
        mHandlerPosition.sendMessage(msg);
    }
}
```

Figure 16: Pagination Implementation-Thread



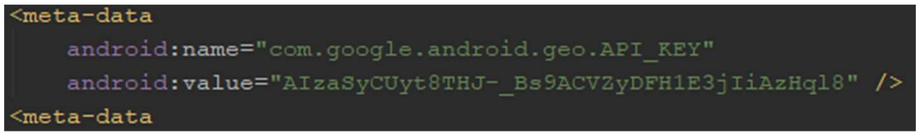
```java
public class MyhandlePosition extends Handler{
    @Override
    public void handleMessage(Message msg) {
        switch(msg.what)
        {
            case 0://Add loading view during search processing
                mListView.addFooterView(ftView);
                break;
            case 1:
                retrivePositions.addAll(nextHundredPositions);
                mListView.removeFooterView(ftView);
                isLoading=false;
                break;
            default:
                break;

        }
    }
}
```

Figure 17: Pagination Implementation-Handler

Google Maps

With the help of the Maps SDK for Android, Google Maps has been integrated into the application. The API is also useful for adding markers to specific locations, adding overlays to maps, and manage the user gestures with the map.[9]

To integrate Google Maps into the application, a key has to be obtained from the Google API Console. A unique Android-restricted key is created for this project. This API key is then added as a child of the <application> element, by inserting it just before the closing </application tag> and to the google_maps_api.xml[9]



```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyCUyt8THJ-_Bs9ACVZyDFH1E3jIiAzHql8" />
<meta-data
```

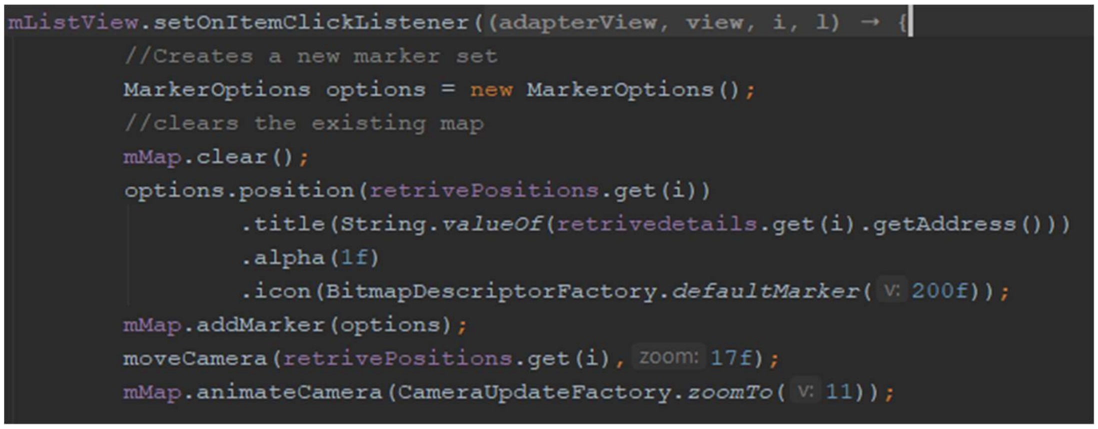
Figure 18: Code Snippet for Google Maps API key

The layout of the second screen is divided into two parts. The first half of the screen is allocated to the ListView while the second half is dedicated to the Google Maps. This facilitates users with checking location information of the selected item from the list. The Google Maps is added as a fragment in the layout resource file.

Markers

Markers provide accurate information about the location of the property.[10] Whenever the user selects one of the items from the ListView, a marker is added to the Google Maps. The zoom and focus are set accordingly to the location. To achieve this, along with retrieving the property details its location details are also retrieved and stored in a LatLng ArrayList. This contains Latitude and Longitude parameters of the properties. The setOnItemClickListener is triggered whenever the user selects one of the items and following steps are executed to add a marker on Maps:

- The markeroptions object is created
- The existing map is cleared to make sure the map displays the currently selected item location only.
- MarkerOption object calls position method which takes in the latitude and longitude of the location.
- Title method which sets the current address to the marker.
- Icon method to get the marker color.
- Created object is then passed onto the maps by the addMarker method.
- The zoom level of the maps is set to 17f and the camera is animated to the location.



```
mListView.setOnItemClickListener((adapterView, view, i, l) → {
        //Creates a new marker set
        MarkerOptions options = new MarkerOptions();
        //clears the existing map
        mMap.clear();
        options.position(retrivePositions.get(i))
                .title(String.valueOf(retrivedetails.get(i).getAddress()))
                .alpha(1f)
                .icon(BitmapDescriptorFactory.defaultMarker( v: 200f));
        mMap.addMarker(options);
        moveCamera(retrivePositions.get(i), zoom: 17f);
        mMap.animateCamera(CameraUpdateFactory.zoomTo( v: 11));
```

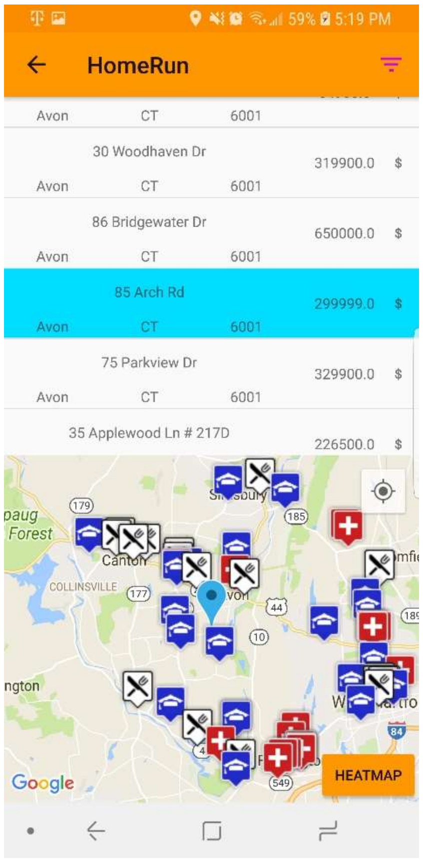
Figure 19: Code Snippet for adding a marker

Nearby Hospitals, Restaurants, and Schools

To give additional information about the selected property, the nearby schools, restaurants, and hospitals are shown on the map along with location marker. A new class is created named GetNearbyPlaces which extends from AsyncTask. Here, a class named DownloadURl helps getting the data from the mentioned URL. This can be done using the HttpURLConnection. [14]

The retrieved data will be in the JSON format, and as we don't need all the retrieved information it should be parsed. The resultant obtained after parsing is stored in the list.

Now as this list has all the information about the locations of the restaurants/hospitals/ schools, markers are made from the location data and plotted on the maps. [14]



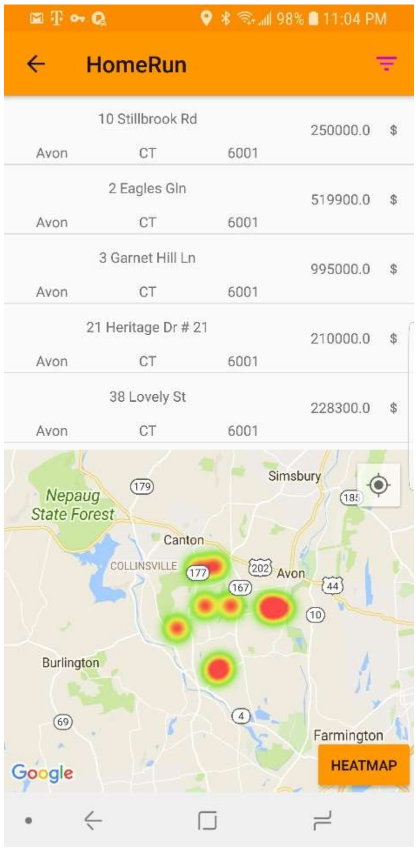
Avon	CT	6001		
30 Woodhaven Dr			319900.0	$
Avon	CT	6001		
86 Bridgewater Dr			650000.0	$
Avon	CT	6001		
85 Arch Rd			299999.0	$
Avon	CT	6001		
75 Parkview Dr			329900.0	$
Avon	CT	6001		
35 Applewood Ln # 217D			226500.0	$

Figure 20: Nearby Schools, Hospitals, and Restaurants

HeatMaps

A dataset consisting of the coordinates for each location of interest is created which helps to add heatmap to Google Maps. A HeatmapTileProvider is created, passing it the collection of LatLng objects. A fresh TileOverlay accepts the heatmap tile provider which is then used to display on Google Maps.[10]

HeatmaptileProvider accepts the latlng objects as arguments and according to the zoom levels and the radius, gradient and opacity different tile images are created. Here the heatmap shows the number of available houses in an area. The intensity of the heatmap varies according to the zoom level.[10]



10 Stillbrook Rd			250000.0	$
Avon	CT	6001		
2 Eagles Gln			519900.0	$
Avon	CT	6001		
3 Garnet Hill Ln			995000.0	$
Avon	CT	6001		
21 Heritage Dr # 21			210000.0	$
Avon	CT	6001		
38 Lovely St			228300.0	$
Avon	CT	6001		

Figure 21: HeatMaps

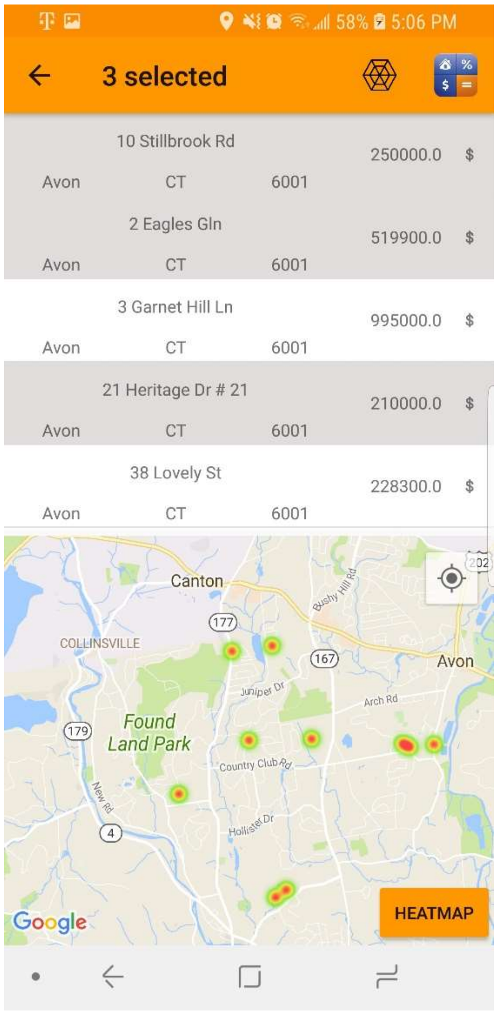
Contextual Action Bar (CAB)

Every application has an action bar, which is the top section of the screen where usually the name of the application or the purpose of the activity is stated. For a specific time period the app's original action is replaced with a custom action bar called as contextual action bar. This overlay is temporary and will be transitioned to default when the sub-task is finished.[15]

Examples can be found in day to day application like Gmail application. On long pressing the emails or by clicking the images to the right of every email, the action bar changes to gray and provides options such as deletion, archive and displays the number of emails selected.[15]

Similar functionality is implemented in the HomeRun application. When a long press is detected on any of the items in the ListView, the contextual mode of the action bar is activated and provides multiple options for the selected items. Once these subtasks are done, the back arrow key to the left in the CAB gets back to the default action bar.

The option provided in the CAB are spider graph and mortgage calculator. A number of items selected by the user are displayed onto the CAB. If the user wants to compare details on the graph, minimum of two or maximum of three items should be selected to compare the details on the spider graph. For the mortgage calculator, only one item needs to be selected. Activation of the CAB changes the selection of the item and action bar color to grey. And the icons for the spider graph and the mortgage calculator are displayed.



	10 Stillbrook Rd		250000.0	$
Avon	CT	6001		
	2 Eagles Gln		519900.0	$
Avon	CT	6001		
	3 Garnet Hill Ln		995000.0	$
Avon	CT	6001		
	21 Heritage Dr # 21		210000.0	$
Avon	CT	6001		
	38 Lovely St		228300.0	$
Avon	CT	6001		

Figure 22: Contextual Action Bar

CAB is implemented by calling the setMultiChoiceModeListener method on the ListView object. It has four override methods:

1. onItemChangeStateChanged
2. onCreateActionMode
3. onPrepareActinMode
4. onActionItemClicked
5. onDestroyActionMode

onItemChangeStateChanged

Here, the number of selected/deselected items are determined and the title in the CAB is updated. [15]

onCreateActionMode

On creating the CAB, inflate the spider graph and mortgage calculator using the menuInflater method. These icons are pre-defined in the menu resource file. [15]

onPrepareActinMode

Updates to the CAB can be done due to an invalidate() request. [15]

onActionItemClicked

This method responds to the clicks on the menu icons like spider graph and mortgage calculator icons. Explicit intent to corresponding activities is created. Spider graph icon opens up the Spidergraph activity and the mortgage calculator icon opens up the MortgageCalc activity. [15]

The logic to ensure that minimum of two and maximum of three items are selected is implemented here.

onDestroyActionMode

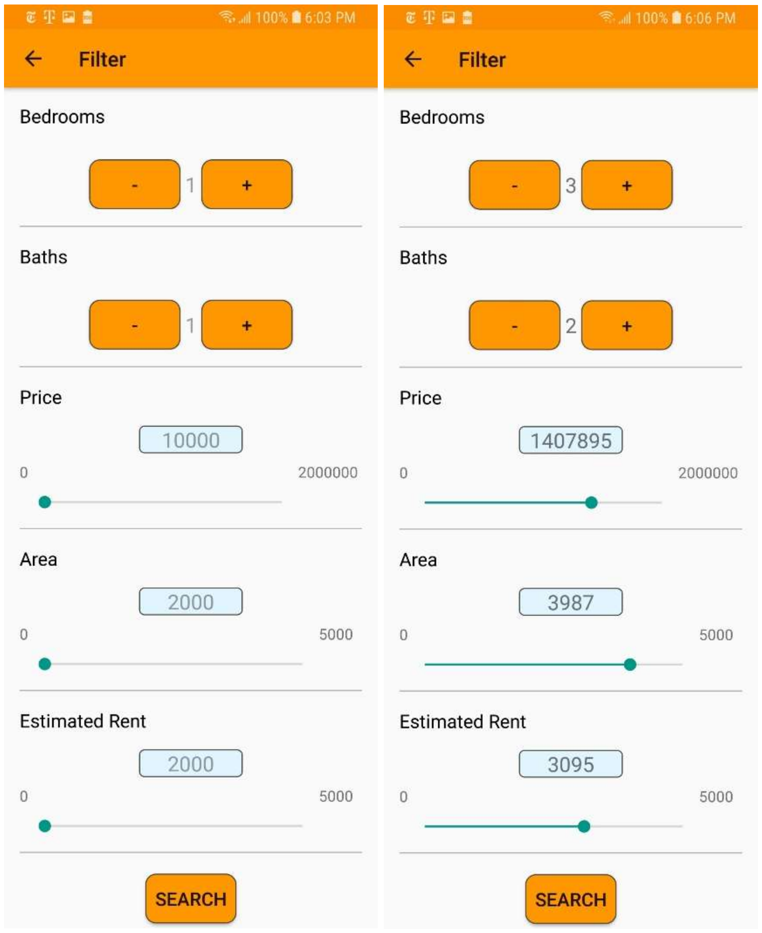
Necessary updates to the activity are done when the CAB is removed. The selected items are deselected/unchecked. The adapter is cleared using the clearSelection method. [15]

Filter

The items in the ListView are fetched according to the user input parameters such as state, city and zip code. To provide more options to users so that they can imply more restrictions on the properties they are looking for, filter option is provided. The filter icon is placed to the top right of the second screen. The action bar is inflated with the filter icon, for which an onClickListener is set.

Users can filter the listed house details according to rent, price, area, number of bathrooms and number of bedrooms. Seekbar is implemented for price, rent and area fields, which gives users an easy way to choose the range. Minimum and maximum values are set according to the min/max values from the database. Change in seekbar changes the value in the TextView present on the top of each seekbar. This change is spontaneous with sliding seekbars.

For bedrooms and bathrooms, quantity is selected with the plus and minus buttons. Bounds are checked and made sure that the value doesn’t fall below 1 and may rise up to the maximum value present for corresponding fields in the database.


100% 6:03 PM

Filter

Bedrooms

- 1 +

Baths

- 1 +

Price

10000

0 2000000

Area

2000

0 5000

Estimated Rent

2000

0 5000

SEARCH

100% 6:06 PM

Filter

Bedrooms

- 3 +

Baths

- 2 +

Price

1407895

0 2000000

Area

3987

0 5000

Estimated Rent

3095

0 5000

SEARCH

Figure 23: Filter

Querying onto database

All the filter information is combined with the user input string. Again the logic to determine the type of user input is applied and a query URL is created which is then passed to the MainActivty of the application. This data is received by the MainActivity in onResume() lifecycle method. Response from the JsonArrayRequest is then sorted to extract the desired JSON objects. These objects are stored in the ArrayList which is fed to the ArrayAdapter of the listView. The adapter is notified of the change in the dataset and ListView is updated.

Spider Graph

Graphs are one of the best ways to compare information about two or more objects. And Homerun provides users with a unique way of comparing house details in form of Spider Graph. Spider Graph as the name suggests looks like a spider web. Users can select multiple items from the ListView and the spider graph plots area, rent and price on the graph's axis. As these three parameters play a crucial role in the financial calculation, it gives a better understanding to the users about the property valuation.

As the user long presses one of the items in the ListView, CAB is activated where the user can select spider graph option on selecting two or three items to compare. The Spider Graph has three axes, area, price, and rent. The selected items address is stored in an ArrayList and passed to the spiderGraph activity. Using this information, a query URL is created which when passed on the JsonArrayRequest retrieves a response. This response has Json objects containing information about the area, price and rent values for corresponding addresses. These values are used to create data sets which are used to plot on the Spider graph.

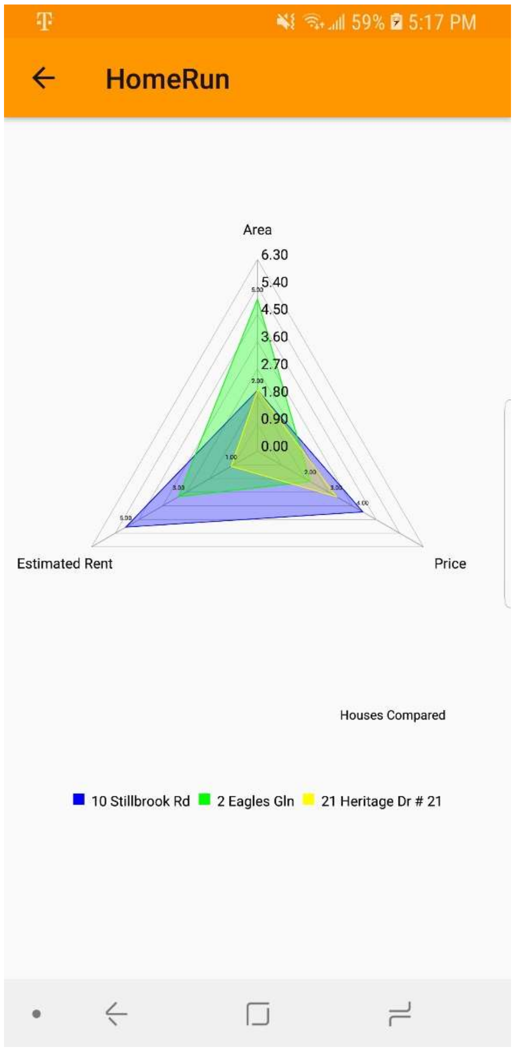
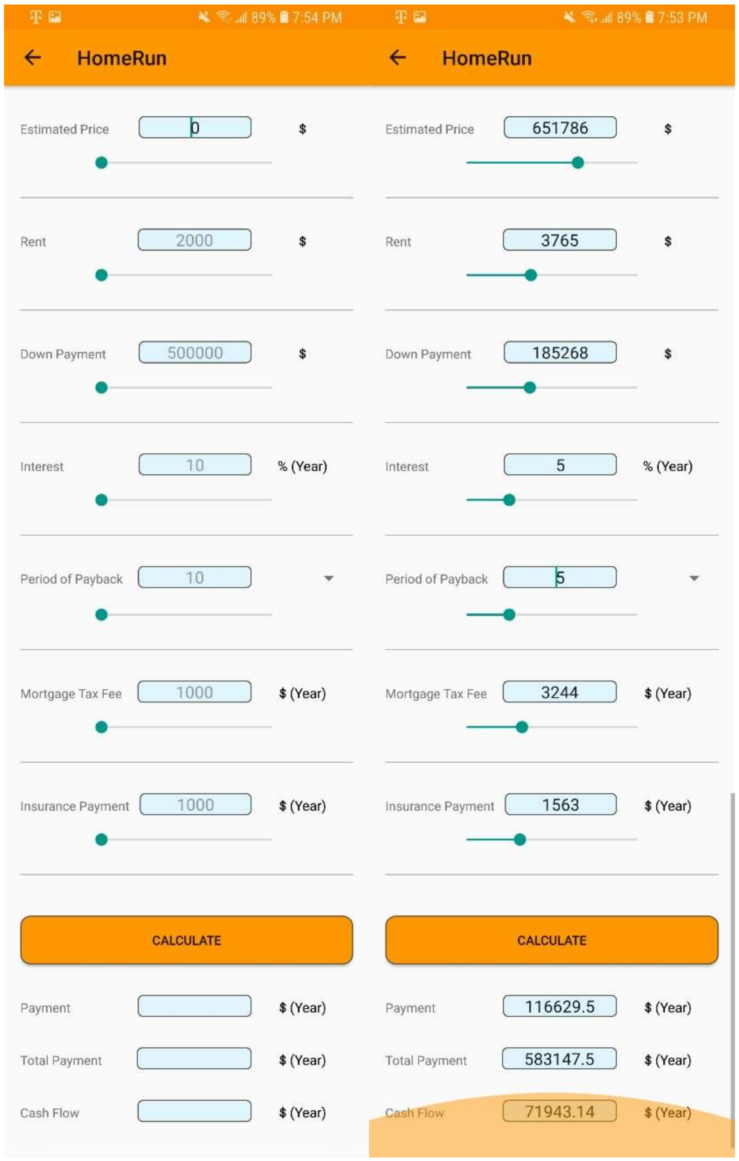


Figure 24: Spider Graph

Mortgage Calculator

HomeRun provides with the built-in Mortgage Calculator. This feature provides users with an estimate of how the new property stand up against expenses. Users enter information such as price, down payment, interest, a period of payback, mortgage tax fee and insurance fee to calculate the Payment and Cash flow.

Users have to select at least one item from the ListView so that the mortgage calculator can retrieve available information such as price and rent from the database for the selected house. The UI allows users to either directly enter the values or use seek bar. In either of the case when calculate is pressed, the related text view and the seek bar are updated. On pressing calculate, the output text views for the cash flow and payment are bought into focus by scrolling down to the bottom of the screen.



Estimated Price	0	$
Rent	2000	$
Down Payment	500000	$
Interest	10	% (Year)
Period of Payback	10	
Mortgage Tax Fee	1000	$ (Year)
Insurance Payment	1000	$ (Year)

Payment		$ (Year)
Total Payment		$ (Year)
Cash Flow		$ (Year)

Estimated Price	651786	$
Rent	3765	$
Down Payment	185268	$
Interest	5	% (Year)
Period of Payback	5	
Mortgage Tax Fee	3244	$ (Year)
Insurance Payment	1563	$ (Year)

Payment	116629.5	$ (Year)
Total Payment	583147.5	$ (Year)
Cash Flow	71943.14	$ (Year)

Figure 25: Mortgage Calculator

FUTURE WORK

Future work depends on eliminating the limitations for HomeRun for this stage and then extending the application with few another features.

1. **Mortgage:** For now the mortgage calculation is not accurate. It does not consider the rules and regulations are different for different states. How the interest rate varies across different banks. How the loan amount, actual price has effect on the rate of interest. These are couple of other measures to be taken care of before calculating the mortgage.[1]

2. **Settings with login management:** Having a dedicated user account on the HomeRun will help in providing the flexibility and freedom to the user. Users can save the results they are interested in and it will be helpful to administer user activity. Thus, it adds to the security aspect of the application as well.[1]

3. **Dynamic location display:** Starting navigation to one property location, on the way to that location, the map can show all the properties passing by. This provides users with various options.[1]

4. **Photos:** For now the details does not include the photos of the application. Adding pictures of properties is a must needed feature. This brings satisfaction to the user.[1]

5. **Draw on Map:** Allowing users to draw on map a custom boundary in which all the properties falling in will be displayed. This feature is really helpful if the user wants to live in a specific area that is not already been marked by the Google Maps.[1]

CONCLUSION

At the end, I would like to conclude that HomeRun has various useful features and makes a good run against its competitors being a one-stop property search application. A lot of different approaches while implementing features, incorporating database and designing features have been considered and tested. All the development progress can be seen at the Github repository of the HomeRun application.

Existing features of the HomeRun are very interesting and will satisfy the basic demands of the users. But, in future, working on limitation and users reviews an improved version of this application will be launched.

APPENDIX

Github: https://github.com/d27saurabh/HomeRun2

YouTube: https://www.youtube.com/watch?v=gwS6UpSmg78

REFERENCES

[1] Gupta Rohit, “Mobile Real Estate Agent For Android,” M.S. thesis, Dept. of Computer Science, SDSU, CA, USA, 2011. [Online]. Available: https://sdsu-dspace.calstate.edu/bitstream/handle/10211.10/1712/Gupta_Rohit.pdf?sequence=1

[2] Siddharta Sreenivasa Reddy, “Trip Tracker Application on Android,” M.S. thesis, Dept. of Computer Science, SDSU, CA, USA, 2011. [Online]. Available: http://sdsudspace.calstate.edu/bitstream/handle/10211.10/1303/Sreenivasa.pdf?sequence=1

[3] “Operating Systme Market Share Worldwide”, StatCounter Global Stats. Accessed April 16. [Online]. Available: http://gs.statcounter.com/os-market-share

[4] Egham, “Gartner Newsroom,” Gartner,February 22,2018. [Online]. Available: https://www.gartner.com/newsroom/id/3859963

[5] K. Abrosimova, “How to Develop a Real Estate App Like Zillow,” YALANTIS, Accessed June 10,2018.[Online]. Available: https://yalantis.com/blog/mobile-real-estate-app-development-usa-zillow-trulia-apps-technology/

[6] Shereen Messi, “Install Android Studio,” Github – Shereen Messi, November 26, 2017. [Online]. Available: https://github.com/ShereenMessi/Install-android-studio

[7] Hay Zohar, “What are the most popular design patterns for Android?,” Quora, November 11, 2017. [Online]. Available: https://www.quora.com/What-are-the-most-popular-design-patterns-for-Android

[8] “List View,” Google Developers, April 17, 2018. [Online]. Available: https://developer.android.com/guide/topics/ui/layout/listview

[9] “Android > Maps SDK for Android: Overview,” Google Developers, July 16, 2018. [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/intro

[10] “Google Maps Android Heatmap Utility,” Google Developers, July 3, 2018. [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/utility/heatmap

[11] “AutoCompleteTextView,” Google Developers, June 6, 2018. [Online]. Available: https://developer.android.com/reference/android/widget/AutoCompleteTextView

[12] “ListView And ArrayAdapter” Udacity- Android Basics, Accessed date. March 15, 2018. [Online]. Available: https://classroom.udacity.com/courses/ud839/lessons/7709673667/concepts/78832374790923

[13] Suleiman, “Pagination Android Tutorial with RecyclerView: getting Started,” GRAFIX ARTIST, November 1,2016. [Online]. Available: https://blog.iamsuleiman.com/android-pagination-tutorial-getting-started-recyclerview

[14] Navneet, “Google Maps Search Nearby : Displaying Nearby Places using Google Places API and Google Maps API V2,” ANDROID TUTORIAL POINT, July 27, 2016. [Online]. Available: https://www.androidtutorialpoint.com/intermediate/google-maps-search-nearby-displaying-nearby-places-using-google-places-api-google-maps-api-v2/

[15] Paresh Mayani, “Contextual Action Bar(CAB) in Android,” DZone Mobile Zone, Oct. 24, 2013. [Online]. Available: https://dzone.com/articles/contextual-action-bar-cab

BIOGRAPHY

Saurabh Deshpande grew up in India. He attended the University of Pune, where he received his Bachelor of Science in Electronics and Telecommunication in 2016. He went on to receive his Master of Science in Computer Engineering in 2018 from George Mason University. He will be joining GoCanvas solutions in Fall 2018.