# SOFTWARE DEVELOPMENT FOR MOBILE DEVICES

# COS80019

ASSIGNMENT 11.1D – Experience Report

# Accessibility Map in Melbourne

# Student name: Duy Thuc Pham

# Student number: 101767225

# INTRODUCTION

Due to the rapid development of infrastructure in Melbourne recently, the disabled people, especially the wheelchair people, are difficult to go around the city. Therefore, this application aims to create a map including all accessibility information so that disabled are able to check the accessibility level of the building or places that they intend to go. The application utilizes the Google Map to display accessibility places within the specific radius as markers. It also supports users to look up the place they want to travel efficiently. Moreover, the user can bookmark places and find the route to go to these places.

This report will discuss the design communication perspectives which is one of the three perspectives of experience report. It will first introduce the user interface and workflow of the interaction between activities in the application. Then, the RESTful APIs, which have been used in the application to retrieve the data from Melbourne Open Data, will be presented. Thereafter, the report will illustrate the Google Map APIs that are using in this application to load the map and also to create a cluster marker to improve the user experience. Last but not least, the database schema and related libraries will be introduced.

# SUMMARY

## App Flow

The application follows the sketches that have been proposed in task 7.3 to develop (Appendix). There are five activities represent five screens: MapActivities, BookMarkActivity, DetailActivity, FilterActivity and SearchActivity. Screenshots of the application are in Appendix.

The application commences with the MapsActivity. It gets the current location of the user via GPS and use it to load accessibility places around the user location. After loading successfully data, the map creates cluster markers and also markers with respect to the accessibility places. In the map screen, the user can go to the Bookmark screen – uses to bookmark place, Filter screen (uses to manage the user preferences), Search screen (uses to search accessibility places), or Detail Place screen (displays detail information) by clicking to relevant buttons. Furthermore, the user can go to Detail Place screen from Search screen as well as Bookmark screen. The application navigation model is in the Appendix.

## REST APIs

This application uses the APIs which is developed by City of Melbourne Census of Land Use and Employment (CLUE) (Socrata Developer Portal | Socrata). To request the APIs from Open Data, this link is used <https://data.melbourne.vic.gov.au/resource/q8hp-qgps.json> along with the X-App-Token in the header field.

### Request API

The APIs in Melbourne Open Data are quite easy to use and customize to retrieve the expected data. Nevertheless, the data is not pretty good containing many duplicate information and thus filter parameters have been applied to improve the quality of data. To illustrate, the sample request API of Melbourne Open Data looks like this

https://data.melbourne.vic.gov.au/resource/q8hp-qgps.json?$limit=10&$offset=0&$where=building\_name!='' AND accessibility\_type!='' AND census\_year = 2016 &$select=block\_id, count(block\_id), accessibility\_rating, accessibility\_type, accessibility\_type\_description, lower(building\_name), location, street\_address,suburb,x\_coordinate,y\_coordinate&$group=block\_id, accessibility\_rating, accessibility\_type, accessibility\_type\_description, lower(building\_name), location, street\_address,suburb,x\_coordinate,y\_coordinate&$order=block\_id

The SODA API that Melbourne Open Data uses is technically a MySQL base and thus these parameters are applied to select, filter or order the request data in this application (Simple Filtering 2018).

|  |  |
| --- | --- |
| Parameters | Description |
| $select | Select fields that necessary for the application |
| $where | Filter query to get the expected data |
| $limit | Limit the number of record for the request |
| $offset | The starting index of the record for the request |
| $group | Group data for the request |
| $order | Order the data |
| $start\_with | Find the starting character of the specific record |
| $within\_circle | Find the data within the radius |

### Response API

The API response has the following structure

[

{

"accessibility\_rating": "3",

"accessibility\_type": "High level of accessibility",

"accessibility\_type\_description": "Main Entrance is at grade and has no steps or ramp",

"block\_id": "1101",

"location": {

"type": "Point",

"coordinates": [

144.9541587,

-37.82020559

]

},

"lower\_building\_name": "grand central apartments",

"street\_address": "33-71C Spencer Street",

"suburb": "Docklands",

"x\_coordinate": "144.9541587",

"y\_coordinate": "-37.82020559"

}

]

### Integrate and Parse API

In order to improve the effectiveness for the application, on the one hand Retrofit is applied to retrieve data and handle the error for the request (i.e. network issues, bad request, etc.). Instead of handling the response API manually, Retrofit implements two callback functions: onResponse() and onFailure() so that we easily detect whether the data is successfully requested or not.

|  |
| --- |
| RequestDataInterface service = RetrofitClientInstance.*getRetrofitInstance*().create(RequestDataInterface.class);  Call<List<Building>> call = service.getBuildingInRange(data);  call.enqueue(new Callback<List<Building>>() {  @Override  public void onResponse(Call<List<Building>> call, Response<List<Building>> response) {  if (response.isSuccessful()) {  callback.onResponse(response.body());  } else {  String errorMessage = ErrorHandlingClass.*handlingRequestError*(response.code());  callback.onFailure(errorMessage);  }  }  @Override  public void onFailure(Call<List<Building>> call, Throwable t) {  Log.*e*("Error", "network failure :( inform the user and possibly retry");  callback.onFailure("Network failure");  }  }); } |

Figure : Code snippet for performing API request with Retrofit

On the other hand, GSON is used to parse the JSON into object. By defining the @SerializeName(JSON key), the JSON response can automatically parse to the defined object.

|  |
| --- |
| public class Building implements Parcelable {  @SerializedName("street\_address")  private String address;  @SerializedName("lower\_building\_name")  private String name;  @SerializedName("block\_id")  private String blockId;  @SerializedName("x\_coordinate")  private Double longitude;  @SerializedName("y\_coordinate")  private Double latitude;  @SerializedName("suburb")  private String suburb;  @SerializedName("accessibility\_rating")  private int rating;  @SerializedName("accessibility\_type")  private String type;  @SerializedName("accessibility\_type\_description")  private String accessibilityDes;  } |

Figure : Parse JSON to object with GSON

## Map API

The map takes an essential role in this application and hence, in this part the report will demonstrate how the map, markers and get user location are developed. For the implementation of map, the application has used the Google map. Furthermore, to improve the user experience, the application applies '*com.google.maps.android:android-maps-utils:0.5'* to implement the cluster marker which reduce the number of marker displays on the map Figure 4. Additionally, with the aim to get the location more accurate, the application uses ACCESS\_FINE\_LOCATION to get the location from GPS provider (Location strategies 2018*)*. However, it may cause the crash if the user does not approve for this permission. Therefore, the user is asked to allow the permission before requesting the location from the GPS provider Figure 5.

|  |
| --- |
| if (ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*) {  ActivityCompat.*requestPermissions*(this, new String[]{Manifest.permission.*ACCESS\_FINE\_LOCATION*}, 8); } else {  requestLocationService(); } |

Figure : Requesting user permission

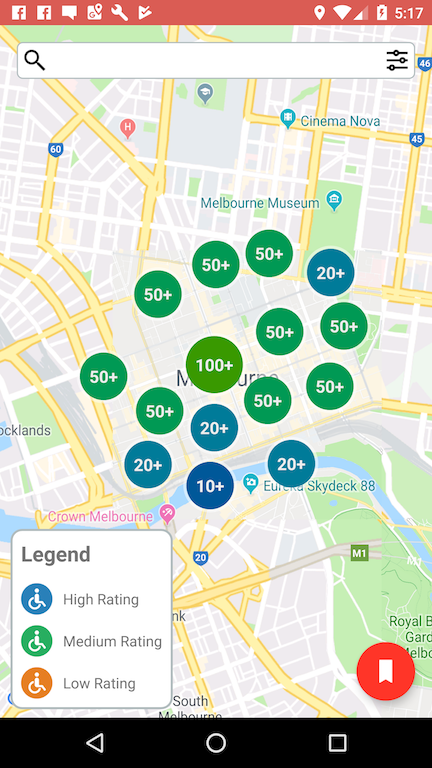


Figure : Google map with cluster view

## Database

The application uses Active Android ORM to replace basic SQLite function as well manages the local database effectively. In this application, the accessibility places are stored local for the user so that they can see these places without searching it again.

### Data Schema

|  |
| --- |
| @Table(name = "BuildingModel") public class BuildingDA extends Model {  @Column(name = "address")  public String address;   @Column(name = "name" , unique = true, onUniqueConflict = Column.ConflictAction.*REPLACE*)  public String name;   @Column(name = "blockID")  public int blockId;   @Column(name = "longitude")  public Double longitude;   @Column(name = "latitude")  public Double latitude;   @Column(name = "suburb")  public String suburb;   @Column(name = "rating")  public int rating;   @Column(name = "type")  public String type;   @Column(name = "accessibilityDes")  public String accessibilityDes;   @Column(name = "imageURL")  public String imageURL;  } |

Figure : BuildingModel table with ActiveAndroid ORM

In this application, I create a *BuildingModel* table which uses to store bookmark places. To manage the query and operation for the database, the DatabaseManager class is created. To illustrate, below is the example of how the building is retrieved from database.

|  |
| --- |
| public static List<BuildingDA> getListBuilding() {  return new Select()  .from(BuildingDA.class)  .execute(); } |

Figure : Query List Building in database with ActiveAndroi

# CONCLUSION

This application has completely developed to support the disabled people are able to get to expected places effectively without worrying much about the accessible level of that place. By applying Retrofit along with GSON, the application can parse the JSON and manage the response API effectively in terms of reusability, maintenance and quality of the code. Furthermore, the cluster helps reduce many markers displaying on the screen and hence it improves the user experience in this application. Also, with the assistance of Active Android, the application manages the database effectively with regards to data migration and query. Nevertheless, the data of Melbourne Open Data does not include images for these places which are not adequate for the user to have the visual information of this place. In the future, to improve the application, the application should have its own server which includes the data from Melbourne Open Data and images for these places. On top of that, by implementing the Restful API, the query for the web services will not be complicated like the APIs from Melbourne Open Data.

# REFERENCES

*Socrata Developer Portal | Socrata*, Dev.socrata.com, viewed 29 May 2018, <https://dev.socrata.com/foundry/data.melbourne.vic.gov.au/q8hp-qgps>.

*Simple Filtering | Socrata*, *Dev.socrata.com*, viewed 29 May 2018, <https://dev.socrata.com/docs/filtering.html>.

*Location strategies*, viewed 3 November 2018, <https://developer.android.com/guide/topics/location/strategies>.

# APPENDIX

## App Screen

|  |  |
| --- | --- |
| Screen |  |
| Launch Screen | ../../../../../Sem%202/Creating_Data_Driven/Assignment/AccessibilityMap/AccessibilityMap/AccessibilityMap/Shared/Image/Assets.xcassets/launch_screen.imageset/1430144371545-2015-04-26%2015.52.57.png |
| Map Screen | ../../../../../../../../Desktop/Google%20Nexus%206%20-%207.0.0%20-%20API%2024%20-%201440x2560/screensho |
| Detail Window Screen | ../../../../../../../../Desktop/Google%20Nexus%206%20-%207.0.0%20-%20API%2024%20-%201440x2560/screensho |
| Search Screen | ../../../../../../../../Desktop/Google%20Nexus%206%20-%207.0.0%20-%20API%2024%20-%201440x2560/screensh |
| Bookmark Screen | ../../../../../../../../Desktop/Google%20Nexus%206%20-%207.0.0%20-%20API%2024%20-%201440x2560/screensho |
| Detail Location Screen | ../../../../../../../../Desktop/Google%20Nexus%206%20-%207.0.0%20-%20API%2024%20-%201440x2560/screensho |
| Filter Screen | ../../../../../../../../Desktop/Google%20Nexus%206%20-%207.0.0%20-%20API%2024%20-%201440x2560/screensho |

## Navigation Model

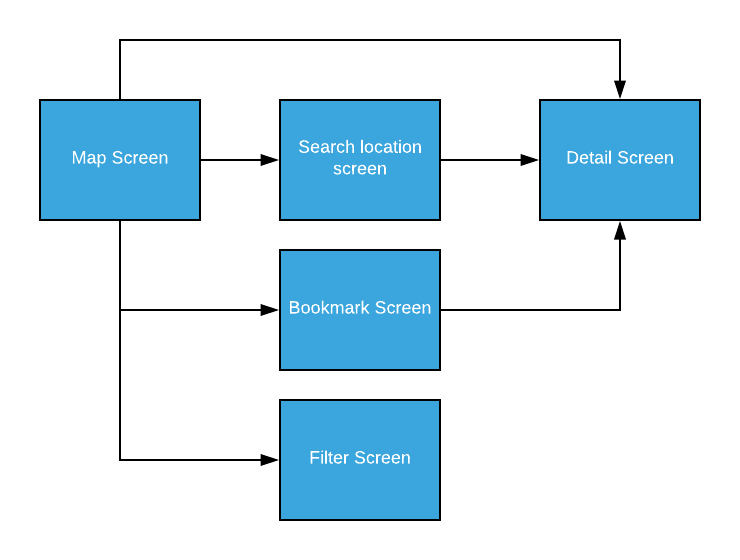


Figure : Navigation Model

## Class Diagram



Figure : Class Diagram

## Sketches

|  |  |  |
| --- | --- | --- |
| Screen Name | Screen | Comment |
| Home-Screen |  | This screen is used to display accessibility places.  The user can search for the location by touching on the search bar |
| Selected Home Screen | ../../../../../../../../Downloads/New%20Wirefr | When the user selects the marker, the info window will display to illustrate the briefly information of the accessibility places |
| Search Location-Screen | ../../../../../../../../Downloads/New%20Wirefr | This screen is used to search accessibility places based on the name in pagination API. |
| Bookmark Location-Screen | ../../../../../../../../Downloads/New%20Wirefr | This screen is used to search accessibility places based on the name in sqlite database. |
| Detail Location Screen | ../../../../../../../../Downloads/New%20Wirefr | This screen is used to display an information of accessibility locations in detail. |
| Filter Location Screen | ../../../../../../../../Downloads/New%20Wirefr |  |

## App Source Code

The source code of the application

<https://github.com/duythuc28/AccessMap_Android->