

Creating Data-Driven Mobile Applications

COS80019

ASSIGNMENT 11 – 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-qgpps.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-  
qgpps.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 =
ErrorHandlerClass.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 1: Code snippet for performing API request with Retrofit

On the other hand, GSON is used to parse the JSON into object. By defining the `@SerializedName`(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 2: 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 3: Requesting user permission

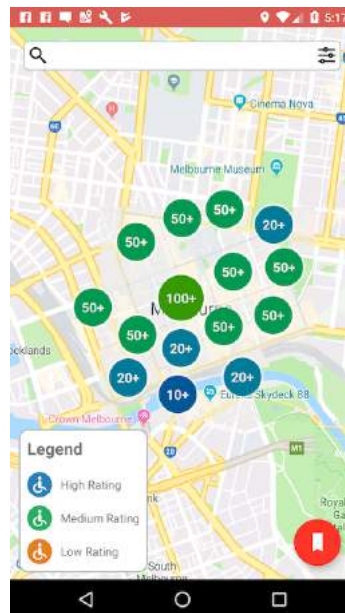


Figure 4: Google map with cluster view

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 5: 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 6: Query List Building in database with ActiveAndroid

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


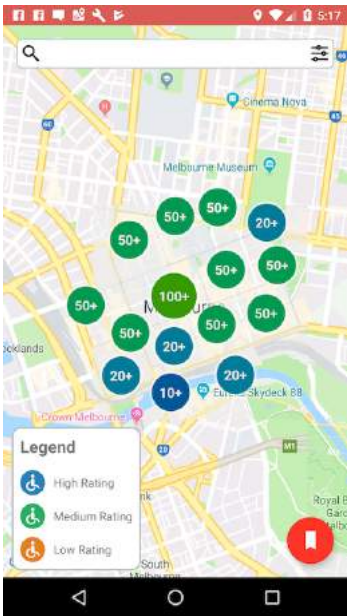
Socrata Developer Portal | *Socrata*, Dev.socrata.com, viewed 29 May 2018,
<<https://dev.socrata.com/foundry/data.melbourne.vic.gov.au/q8hp-qgps>>.

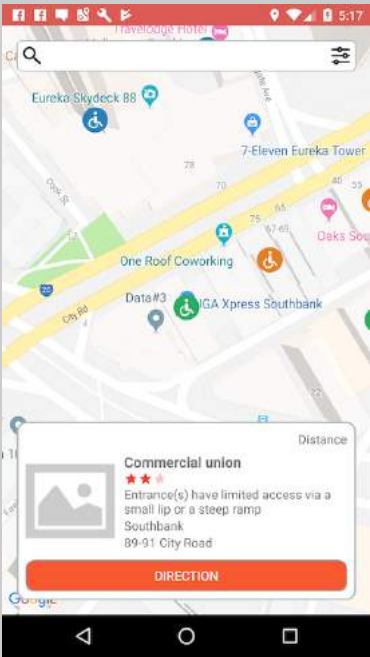
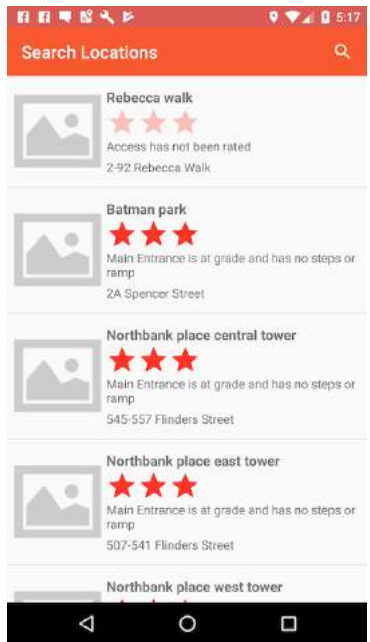
Simple Filtering | *Socrata*, Dev.socrata.com, viewed 29 May 2018,
<<https://dev.socrata.com/docs/filtering.html>>.

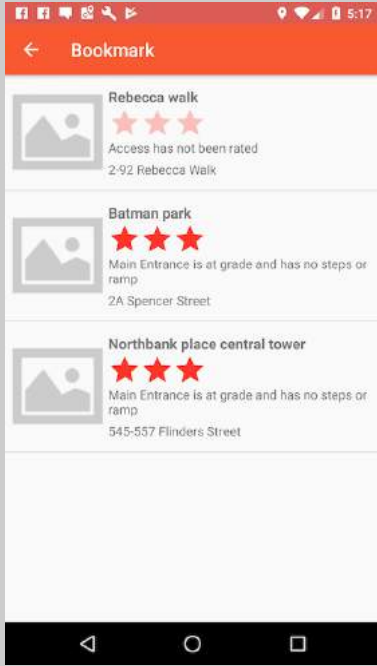
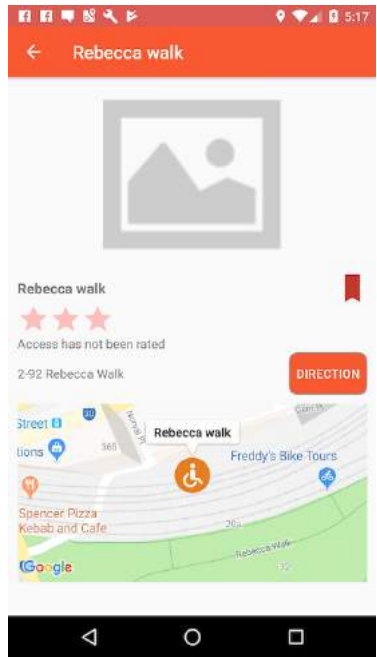
Location strategies, viewed 3 November 2018,
<<https://developer.android.com/guide/topics/location/strategies>>.

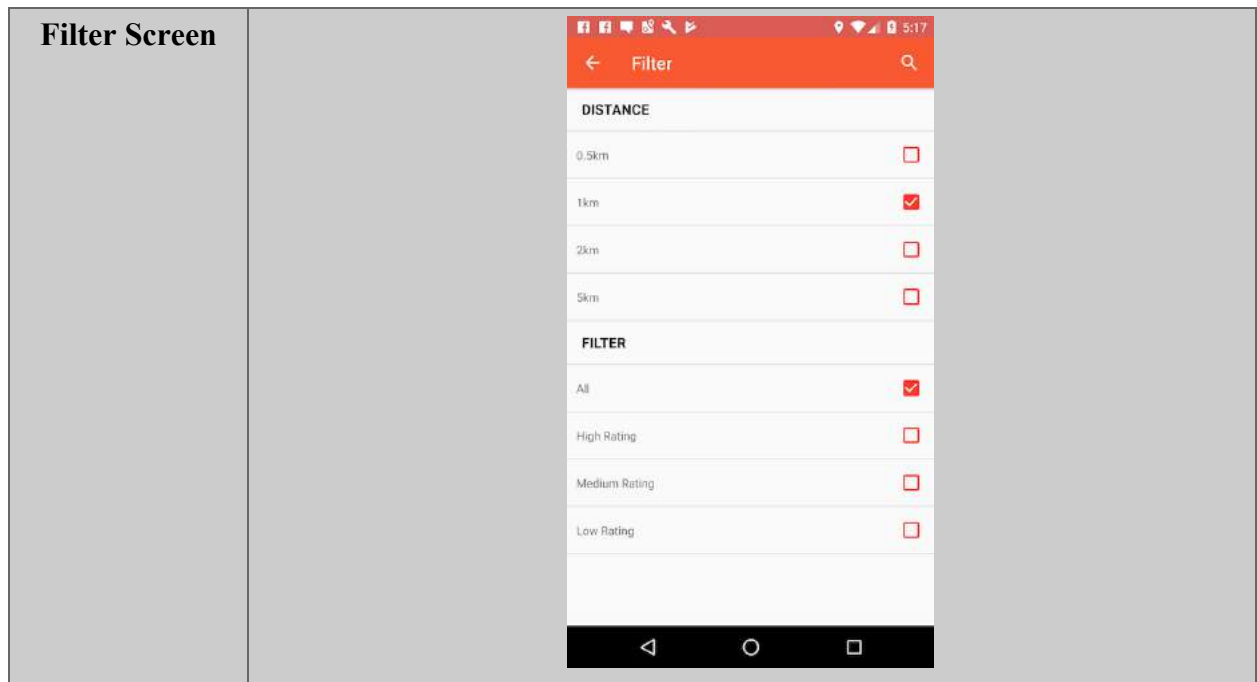
APPENDIX

App Screen

Screen	
Launch Screen	
Map Screen	

Detail Window Screen	 <p>The screenshot shows a mobile application interface for a map. At the top, there's a search bar and a status bar with the time 5:17. The map displays various locations including Eureka Skydeck 88, 7-Eleven Eureka Tower, One Roof Coworking, Data#3, and GA Xpress Southbank. A pop-up window for 'Commercial union' is visible, showing a star rating of 3 stars, a description of limited access, and a 'DIRECTION' button.</p>
Search Screen	 <p>The screenshot shows a mobile application interface for a search screen. At the top, there's a search bar with the text 'Search Locations' and a magnifying glass icon. Below the search bar, there's a list of locations with their respective star ratings and descriptions:</p> <ul style="list-style-type: none">Rebecca walk: Access has not been rated, 2-92 Rebecca WalkBatman park: Main Entrance is at grade and has no steps or ramp, 2A Spencer StreetNorthbank place central tower: Main Entrance is at grade and has no steps or ramp, 545-557 Flinders StreetNorthbank place east tower: Main Entrance is at grade and has no steps or ramp, 507-541 Flinders StreetNorthbank place west tower

Bookmark Screen	
Detail Location Screen	



Navigation Model

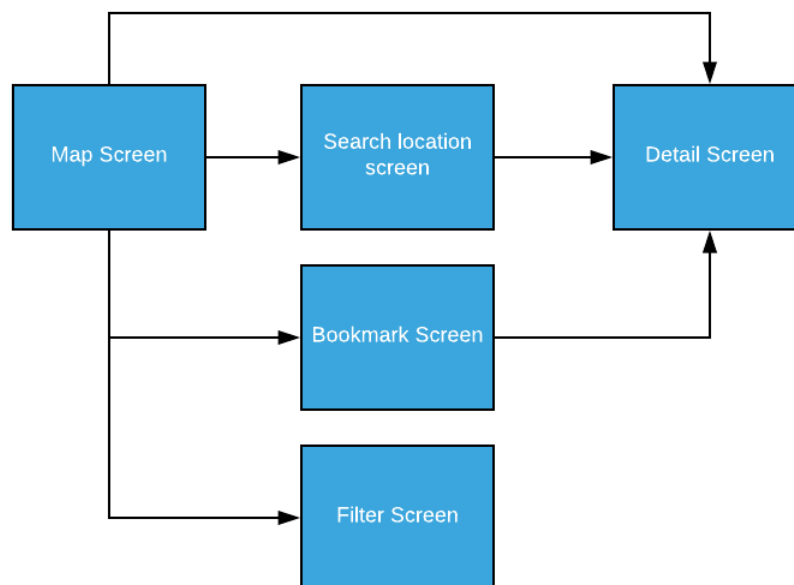


Figure 7: Navigation Model

Class Diagram

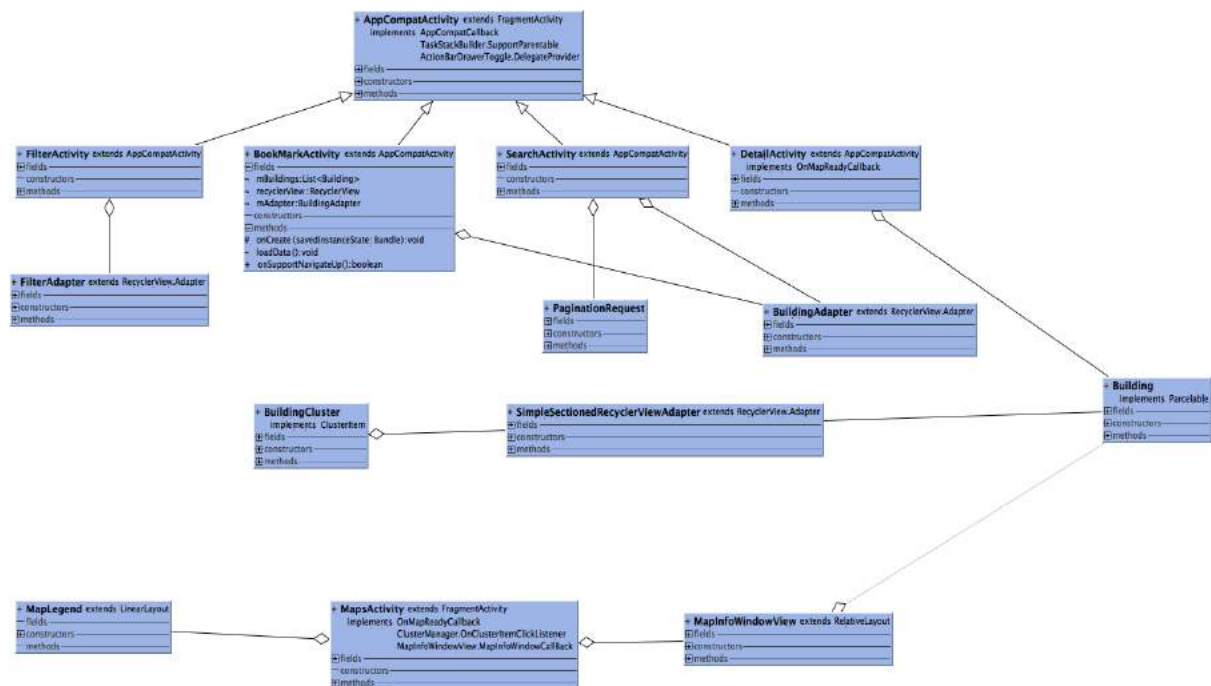
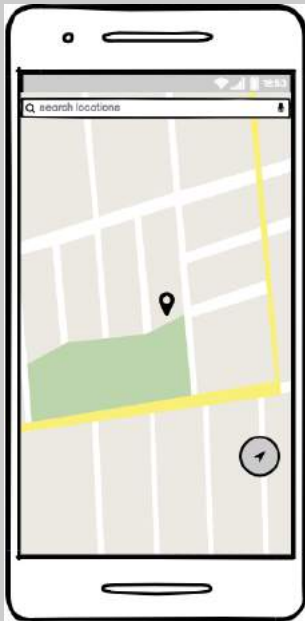


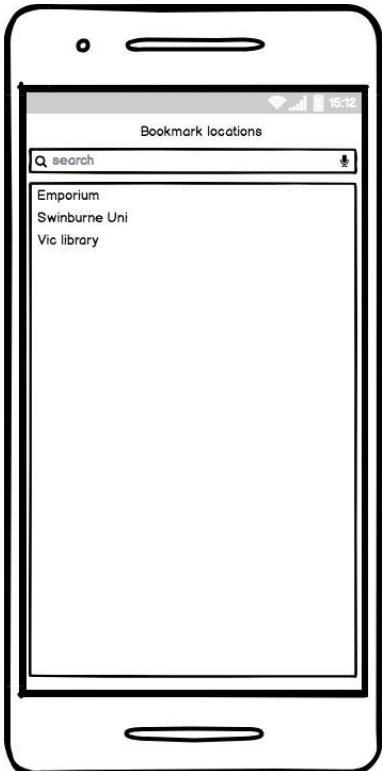
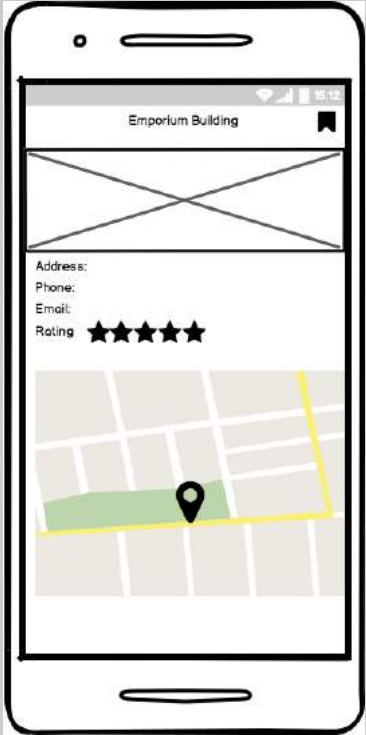


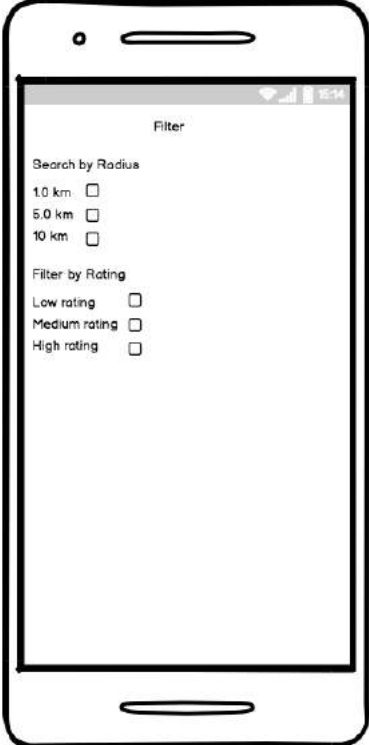
Figure 8: Class Diagram

Sketches

Screen Name	Screen	Comment
Home-Screen		<p>This screen is used to display accessibility places.</p> <p>The user can search for the location by touching on the search bar</p>

Selected Home Screen	 A smartphone screen displaying a map application. A location pin is placed on a green area representing a park. An information window is open at the bottom of the screen, showing a checkmark icon, the name 'Emporium Building', the address 'Address: 123 Bourke Street', a rating of five stars, and a 'Go to this place' button.	<p>When the user selects the marker, the info window will display to illustrate the briefly information of the accessibility places</p>
Search Location-Screen	 A smartphone screen showing a search interface. At the top, there is a search bar with the placeholder text 'search locations'. Below the search bar, a list of search results is displayed: 'Emporium', 'Swinburne Uni', and 'Vic library'.	<p>This screen is used to search accessibility places based on the name in pagination API.</p>

Bookmark Location-Screen		This screen is used to search accessibility places based on the name in sqlite database.
Detail Location Screen		This screen is used to display an information of accessibility locations in detail.

Filter Location Screen		
-----------------------------------	---	--

App Source Code

The source code of the application

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