**How to Use this Template**

1. Make a copy [ File → Make a copy... ]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

**Submission Instructions**

1. After you've completed all the sections, download this document as a PDF [ File → Download as PDF ]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

---



**GitHub Username**: 9mat

# ParkMyCar

## Description

ParkMyCar aims at providing car drivers in Singapore with real-time information about nearby car-park locations and vacancies.

**Problem**
Drivers in Singapore spend a large portion of their time driving from car-parks to car-parks just to look for a vacant parking lot, especially during peak hours when parking lots are in great demand. Much of this time and effort could have been saved had the drivers known beforehand the locations and vacancies of the car-parks without the need of actually being at the car-parks to collect those information.

**Proposed Solution**

This app is designed to pull the data from the Singapore Land and Transport Authority's real-time data service on car park availability and display it to the users. After receiving the data, the app will display a list of nearby car parks together with their numbers of vacant parking lots. The app can read the user's current location and sort the list according to the distance of the car parks to the user. The user can click on each item on the list and the app will show the actual location of the chosen car park on the map, together with the direction how to get there from the user's current location,

## Intended User

The app is intended for car drivers in Singapore, especially those who frequent the centre business district during peak hours.

## Features

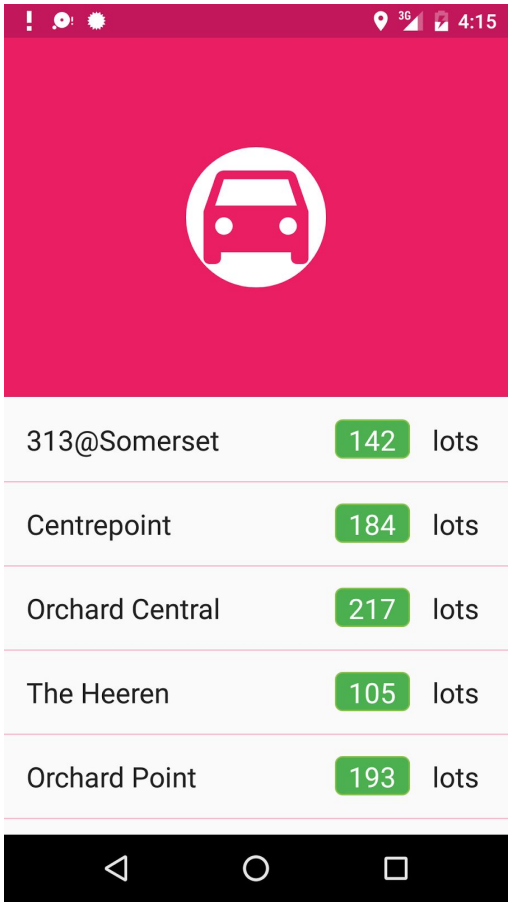
Main feature of the app:

- Display a list of nearby car parks
- Display real-time information on car-park availability of the nearby car parks
- Display the location of a chosen car-park on Map
- Show the direction how to drive to a chosen car-park

- Allow users to add widgets to the home screen that display the vacancies of a particular car park

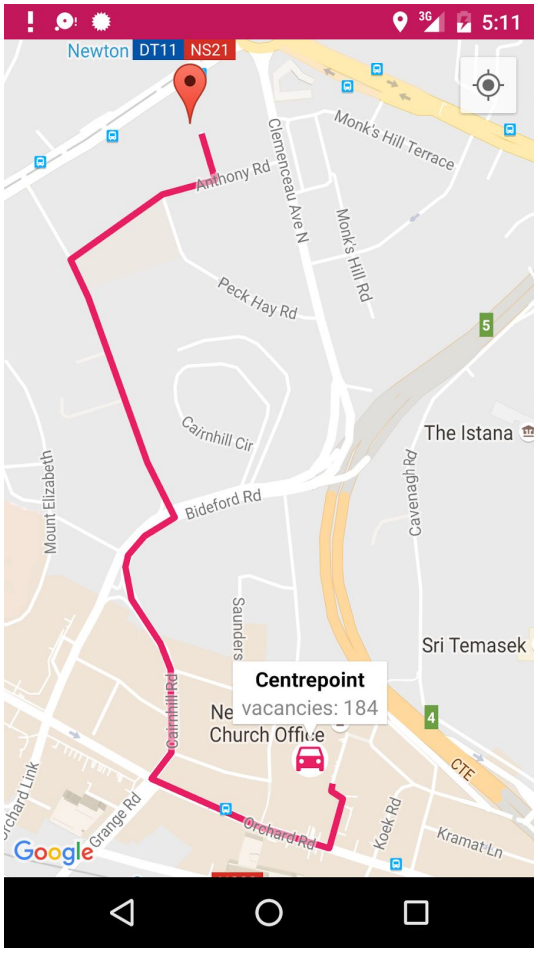
## User Interface Mocks

### Screen 1



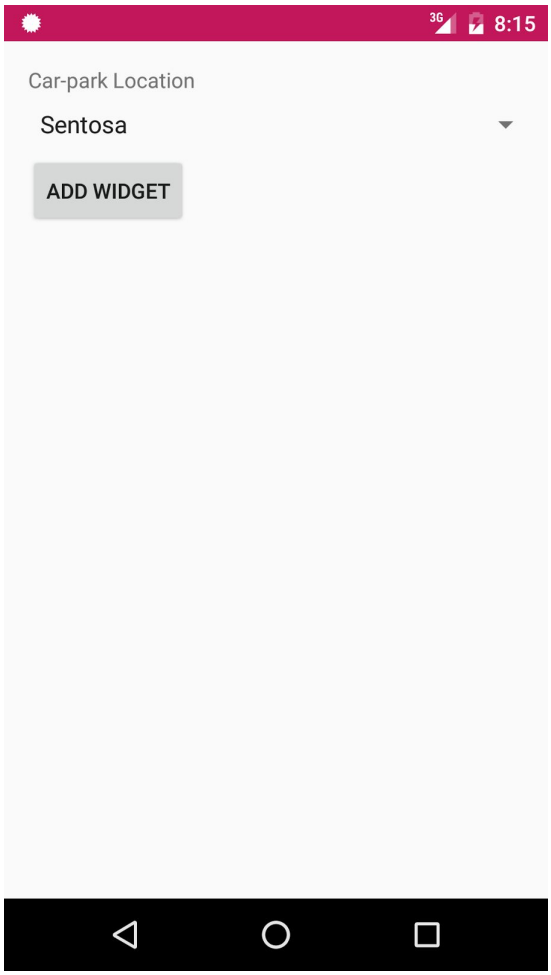

This is main screen of the app. The user is shown with a list of nearby car-parks. The numbers next to the car-parks' name indicate the numbers of vacant parking lots currently available in each of the car-parks. (The color of the numbers may be used to indicate the level of availability: e.g. green when > 50 lots are currently available, red when < 50 lots available).

## Screen 2



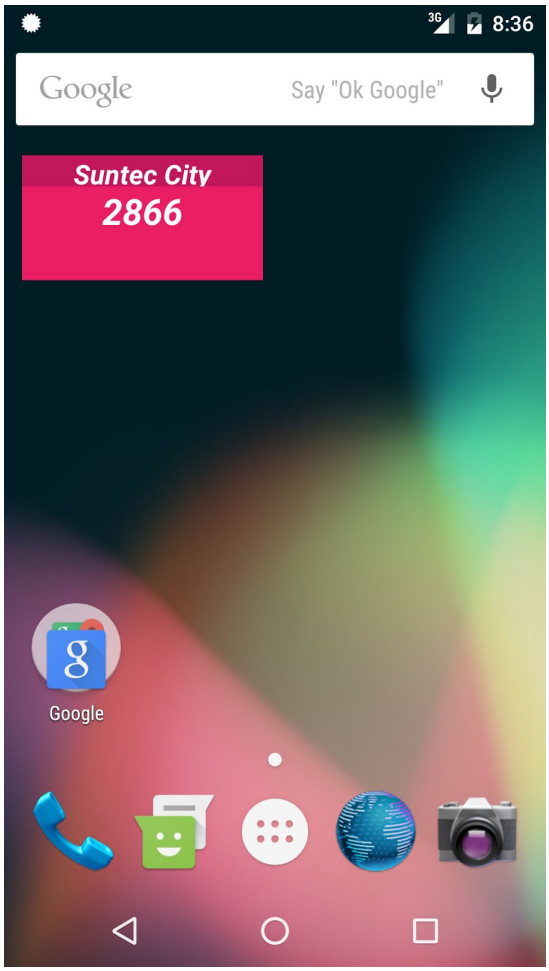

This screen is shown after the user click on an item on the list of the main screen. This screen shows the location of the chosen car-park, with a snippet displaying the number of vacant parking lots in the car park. It also shows the direction how to get to the car-park.

## Screen 3: App Widget Configuration




This screen is shown when users are about to create a new app widget. It allows the users to choose the carpark whose information they want to display on the new widget.

## Widget




This screen shows a widget that user can create to display the number of vacancies of a particular car park on the home screen.

# Key Considerations

### How will your app handle data persistence?

The app will build a new ContentProvider to store the data pulled from the LTA data service.

### Describe any corner cases in the UX.

- If the data requested has not been received: Display on the main screen that the data is being pulled. Also, display a laoding icon.
- If the received data is empty (potentially there is no carpark with available lots, or some error on the LTA server), display a message on the main screen explain the situation to the user
- If current location cannot be read (location service is turned off or user does not grant location permission to the app): sort the car-park list with respect to a “default” location (aka the centre of Singapore), an on the Map activity, show only the location of but not the direction to the car-park.
- If no route (between the current location and the car park) found, display a short Toast explain the situation to the user.

### Describe any libraries you’ll be using and share your reasoning for including them.

- Provigen (https://github.com/TimotheeJeannin/ProviGen) to quickly generate a ContentProvider for handling data persistence
- Retrofit (http://square.github.io/retrofit/) to handle http data request from the LTA server and transforming the resulted data request (in json format) to a data model
- PermissionDispatcher (https://github.com/hotchemi/PermissionsDispatcher) to manage run-time permision (location and map permission) in Andorid 24
- Google-Directions-Android (https://github.com/jd-alexander/Google-Directions-Android) to make Google Direction request and generate routes between two locations

### Describe how you will implement Google Play Services.

- Location service: The main activity will maintain a continuously updated location request to Google Api client so that it can continuously update the list with nearby car-parks to the distance of the car parks to the user location

- Map service: The main activity will pass the car-park information the the Map Activity, which will make request to the Google Map service to display the map portion around the chosen car-park location, and a marker indicating the chosen car-park
- Direction service: this is handled by using a third-party library (Google-Directions-Android). The map service receives the information about the current location and the location of the car-park from the Main Activity, It will then make request to the Direction service via Google-Directions-Android and display the best routes accordingly

# Next Steps: Required Tasks

## Task 1: Project Setup

- Create a new empty project
- Set targetSdkVerion 24 (newest version) and minSdkVersion  to 21 (covering 70% as of Aug 2016)
- Configure libraries: use latest android support library (24.2.0), add gradle dependencies for ProviGen, retrofit, Google play services, android support design, Google_Directions-Android, permissionsdispatcher
- Configure permission in the manifest: ACCESS_FINE_LOCATION, INTERNET, READ_SYNC_SETTINGS, WRITE_SYNC_SETTINGS, AUTHENTICATE_ACCOUNTS

## Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity using AppBarLayout with NestedScrollView on a RecyclerView
- Build UI for Item Fragment on the RecyclerView (displaying information on each individual  car-park)
- Build UI for Map Activity (using SupportMapFragment)
- Test the Main UI with a random list
- Tst the Map UI with a random location

## Task 3: Implement ContentProvider

- Implement CarParkContract (using ProviGen) to describe an entry on the CarPark sqlite table
- Implement CarParkContractProvide with the following query:
    - Query information on a particular car-park using car park ID
    - Query a list of car-parks sorted by their distance to a location (latitude, longitude)

### Task 4: Implement Data request to LTA CarParkAvailability data service

- Obtain the API key to the data service from LTA datamall.transport.sg
- Build a data model base on the json structure returned from LTA
- Use retrofit, implement a helper function to pull data from LTA data service and convert it the app's data model

### Task 5: Implement SyncAdapter and SyncService to automatically pull the data in real-time

- Create the corresponding Authenticator and AuthenticatorService to work with the SyncService
- Implement SyncAdapter:
    - Configure periodic Sync: because the LTA data service is updated every 5 minutes, we will also set the period sync interval to 5-10 minutes
    - Implement  onPerformSync: call the above implemented data-pull request to pull data from the LTA data service, convert it to ContentValues vector and insert it to the local database using ContentProvider

### Task 6: Implement Cursor RecyclerView Adapter

- Implement an Adapter for the Recyclerview that binds views to a cursor

### Task 7: Implement location request for Car Park List Fragment

- Obtain a Google API key
- Build, connect and disconnect when appropriately a Google API client
- When the Google API client is connected, build a location request to continuously obtain the user location
- When the user location change, update the cursor loader accordingly

### Task 8: Implement communication between the MainActivity and the MapActivity

- Add a Listener to listen to click event on each item in the carpark list
- When user click, the Listen obtain information about the carpark and the user location, pass it to a new Intent that start a Map Activity
- The Map Activity read all the information from the Extras bundle in the Intent

## Task 8: Implement displaying location of and direction to the car-park

- Map Activity read the location information passed via Intent, and use Google Map service and Google-Direction-Android to display the location and direction on the Map

## Task 9: Implement app widgets

- Implement Widget Configuration Activity to allow users to choose which car park to be displayed on the widget
- Implement the App widget that shows the number of vacant slots in the corresponding car park

## Task 9: Handle corner cases

## Task 10: Test, sign and build the release version

---

**Submission Instructions**

1. After you've completed all the sections, download this document as a PDF [ File → Download as PDF ]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"