[Description](#_Toc522297803)

[Intended User](#_Toc522297804)

[Features](#_Toc522297805)

[User Interface Mocks](#_Toc522297806)

[Key Considerations](#_Toc522297807)

[How will your app handle data persistence?](#_Toc522297808)

[Describe any edge or corner cases in the UX.](#_Toc522297809)

[Describe any libraries you’ll be using and share your reasoning for including them.](#_Toc522297810)

[Describe how you will implement Google Play Services or other external services.](#_Toc522297811)

[Next Steps: Required Tasks](#_Toc522297812)

[Task 1: Project Setup](#_Toc522297813)

[Task 2: Database setup](#_Toc522297814)

[Task 3: Data Models](#_Toc522297815)

[Task 4: Initialize Menus, Arrays, Colors, Strings & other resources](#_Toc522297816)

[Task 5: Vehicles List](#_Toc522297817)

[Task 6: Vehicles Entity](#_Toc522297818)

[Task 7: Expenses List](#_Toc522297819)

[Task 4: Your Next Task](#_Toc522297820)

[Task 5: Your Next Task](#_Toc522297821)

**GitHub Username**: kalxasath

**GitHub Profile:** <https://github.com/kalxasath>

Capstone – Vehicle expenses tracking

# Description

Simple and effective application to track vehicles consumption and expenses. The application supports more than one vehicle and for each vehicle the application will show the total running costs.

For each vehicle the users can choose a representative photo from gallery, a name, the make, model, plate no, initial mileage, the distance unit (Kilometers, Miles, Hours), tank volume, the volume unit (Liters, Gallons), notes.

Each consumption or expense refers to a particular vehicle, and the data are type, subtype, date, odometer, fuel quantity (if type is refuel), amount, notes.  
The types and subtypes are: (Refuel)[Full], (Bill)[Parking, Toll, Insurance], (Service)[Basic service, Damage]

For each vehicle in the dashboard screen the user can see the vehicle’s total running costs divided in the 3 type categories (Refuel, Expenses, and Service)

Also there is a Youtube search option where the users without leaving the app can direct search related videos according his search criteria.

In the phones home screen the user can add a widget for a particular vehicle to have a look of the vehicle’s total running costs without to enter in the app. The user can add as many widget he likes, each widget displays the data from the selected vehicle on widget creation.

# Intended User

The app is intended for vehicle owners, giving him the ability to track the vehicle’s consumption and expenses, and provide him information about vehicle’s total running costs.

# Features

* Saves information about users vehicles
* Display a list of all of his vehicles
* Track consumption and expenses
* Display a history list of the consumptions and expenses
* Display for each vehicle analytical total running costs
* In app search YouTube for related videos according his search criteria
* Home Screen Widget
* At the end of the dashboard total running cost information there will be an AdMob horizontal banner.

# User Interface Mocks

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Application Menu Drawer | Dashboard | Vehicle List |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Vehicle Entity | Expenses List | Expense Entity |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Date picker | Search YouTube | Search YouTube Results |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Home Screen Widget configuration screen | Home Screen Widget |  |

# Key Considerations

### How will your app handle data persistence?

1. Application will store & retrieve all data entities via a Content Provider in SQLite
2. Application will use complex SQL queries using the “CapstoneDBHelper” extened from “SQLiteOpenHelper”
3. Application will use Shared Preferences to store specific information for each home screen widget, the vehicle ID

### Describe any edge or corner cases in the UX.

* Back button on the title bar of the entities screens will navigate back to the parent screen i.e. Vehicles List or Expenses List
* Moving back from the four main activities i.e Dashboard, Vehicles List, Expenses List or YouTube search will return back to the home screen.
* If the user taps on an item in the Vehicles List or Expenses List then the corresponding entity screen will open to edit or delete the entity.
* If the user taps on an item in the YouTube search results then the app will try to play the selected video via the installed YouTube app, if such an app is not installed in the phone then the app will try to play the video via the web. If the app fails to start the YouTube app or the web then it will show a toast that informs the user with the message ‘An associated YouTube player couldn't be found!’
* If the user taps on the home screen widget or the dashboard list, nothing will happen.
* If the user selects the capstone widget to put it in the home screen then a configuration widget screen with the list of all the vehicles will be displayed. If the user taps on a vehicle the widget is displayed on the home screen displaying the vehicle’s total running costs, otherwise the screen will close and no widget will displayed on the home screen.

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

1. Google’s Compatibility Library will be used to ensure compatibility with older android versions
2. Picasso will be used for caching and displaying the application images
3. Butterknife will be used to eliminate any findViewById calls in adapters and activities
4. Google’s play-services-ads will be used to display the AdMob ads

### Describe how you will implement Google Play Services or other external services.

1. AdMob banners will appear at the end of the dashboard screen
2. YouTube search service will implemented via the ‘https://www.googleapis.com/youtube/v3’ api calls, reviewer will need his own API key to test the functionality.
3. YouTube videos playback will go through the installed YouTube app or the web service ‘https://www.youtube.com/watch’

Google services were choose from the ‘https://en.wikipedia.org/wiki/Category:Google\_services’

# Next Steps: Required Tasks

## Task 1: Project Setup

1. Create a clean project via the android studio
2. Add required google libraries for backwards compatibility, design, recycleview, cardview, constraint layout
3. Add required google libraries for the AdMob ads
4. Add required libraries for Picasso and butterknife
5. For the four main activities i.e. Dashboard, Vehicles List, Expenses List or YouTube search create a layout container that supports the ‘DrawerLayout’ in this layout include all the necessary components for the property use of the drawer, toolbar, floating action button, navigation view.  
   The layout container should be a ‘CoordinatorLayout’ component, name it ‘layout\_container’. Later each activity will inflate in this container its own views.
6. For the two entity activities i.e. Vehicle, Expenses create a layout container that supports the ‘CoordinatorLayout’ in this layout include all the necessary components for the property use of the AppBarLayout, CollapsingToolbarLayout with ImageView, Toolbar.  
   The layout container should be a ‘NestedScrollView’ component, name it ‘layout\_container’. Later each activity will inflate in this container its own views.
7. Helper functions for Network, Preferences, YouTube, Date etc will reside in the ‘package’ ‘utilities’

## Task 2: Database setup

1. SQLite will be used to store and retrieve the applications data
2. The required classes will reside in the ‘package’ ‘data’
3. For the two entities Vehicle and Expenses create the required Contract classes
4. To access and initialize the database create a class extended from ‘SQLiteOpenHelper’
5. To handle the database’s CRUD operations create a class extended from ‘ContentProvider’
6. For other database methods or functions create a class named DBQueries. For simple queries use the ‘ContentProvider’ created in step 5. For more complex queries use the ‘SQLiteOpenHelper’ created in step 4.

## Task 3: Data Models

1. In the ‘package’ ‘model’ create required classes with the required fields, getters, setters for the Expense, Vehicle, VehiclesTotalRunningCosts, VideosListItem, YoutubeSearchInfo

## Task 4: Initialize Menus, Arrays, Colors, Strings & other resources

1. The application will use the drawer menu for the four main activities. Create a xml menu file, name it ‘drawer\_main\_menu’ then add the descriptions for the four main activities i.e. Dashboard, Vehicles List, Expenses List or YouTube search
2. In the ‘package’ ‘navigation’ create a class name it ‘DrawerMenu’ inside create a method name it ‘navigate’. The method ‘navigate’ is responsible to handle the drawers selections and to start the associated activity.
3. For the two entities the toolbar options menu will be used. Create a xml menu file, name it ‘menu\_entity’ then add:
   1. a menu item with title ‘Done’ and icon ‘ic\_done\_white’
   2. a menu item with title ‘Delete’ and icon ‘ic\_delete\_forever\_black\_24dp’, this option will never be showing as icon in the title bar and in the options menu only if the entity is in edit mode.
4. For the YouTube search create a xml menu file, name it ‘menu\_search’ then add:
   1. a menu item with title ‘Search’ and icon ‘ic\_menu\_search’
5. In the ‘res.values’ folder create files for the colors, strings, styles, attrs, arrays, dimens.
6. In the arrays.xml create application standard data like:
   1. Arrays for the distance units one with full description and one with short description, the supported units are: Kilometers, Miles, Hours
   2. Arrays for the volume units one with full description and one with short description, the supported units are: Liters, Gallons
   3. Create an array for the expense type with the values: Refuel, Bill, Service
   4. Create an array for the subtype of expense Refuel with the value: Full
   5. Create an array for the subtype of expense Bill with the value: Parking, Toll, Insurance
   6. Create an array for the subtype of expense Service with the value: Basic Service, Damage

## Task 5: Vehicles List

This activity will display a list for all vehicles stored in the database entity ‘Vehicle’. This activity will use the component ‘recyclerview’ to handle and display the list, for the list population the ‘LoaderManager’ with ‘AsyncTaskLoader’ will be used to query the database and fill an array list of type ‘Vehicle’. The ‘recyclerview’ will display the data from this list via an associated adapter.

In the right bottom corner a floating action button will be displayed with an icon of ‘ic\_add\_white\_24dp’, if the user taps on this fab the ‘VehicleEntityActivity’ is called and an empty vehicle entity is displayed.

If the user taps on the list item of a vehicle, then the ‘VehicleEntityActivity’ is called and screen is populated with vehicles data.

The name of the activity is ‘VehicleListActivity’, in the OnCreate method the content view is set to ‘app\_drawer’ then the ‘layout\_container’ is inflated with activities layout that is stored in the file ‘activity\_vehicles\_list’.

The associated ‘recyclerview’ adapter has the name ‘VehiclesListAdapter’ and is responsible to populate the ‘recyclerview’ with the data. For the Items it will use the layout file ‘item\_list\_vehicle’.

Each vehicle item is based on the component ‘CardView’ which holds an ‘ImageView’ that fills the whole ‘CardView’ with a ‘TextView’ for the vehicle’s name. The ‘TextView’ is above the ‘ImageView’ and in the bottom of the ‘CardView’.

## Task 6: Vehicles Entity

This activity will display the Vehicle’s Entity fields for new entry, editing or deleting. The user can select an image from gallery for the vehicle by tapping the fab with the gallery image.

In the toolbar options menu the action ‘done’ is always visible as icon. The option ‘delete’ is available if the entity is in edit mode.

Actions by the user:

* Taps the ‘done’ menu option, The record will validated for mistakes, when mistakes are found then the user will be informed to correct those mistakes, when validation is passing then the data will be stored in the database via the ‘ContentProvider’ by using the entity contract ‘VehiclesContract’.  
  If the data are for new record then a new record is inserted into the database, when the data are for an old record then the data are replacing the old data in the database.  
  The activity will finish and the screen returns to the vehicle list activity.
* Taps the options menu and the taps on the ‘delete’ option, Via an popup or dialog the user is asked to confirm the deletion, if the user answer with ‘yes’ then the record will be deleted plus all the assosiacted data from expenses for this vehicle if exists.  
  If the deletion occurs then the activity will finish and the screen return to the vehicle list activity, otherwise the activity will stay in the screen.
* Taps the back button on the toolbar, the activity will finish without saving the data and the screen returns to the vehicle list activity.

The name of the activity is ‘VehicleEntityActivity’, in the OnCreate method the content view is set to ‘app\_coordinator\_container’ then the ‘layout\_container’ is inflated with activities layout that is stored in the file ‘activity\_vehicles\_entity’.

## Task 7: Expenses List

This activity will display a list for all vehicles stored in the database entity ‘Vehicle’. This activity will use the component ‘recyclerview’ to handle and display the list, for the list population the ‘LoaderManager’ with ‘AsyncTaskLoader’ will be used to query the database and fill an array list of type ‘Vehicle’. The ‘recyclerview’ will display the data from this list via an associated adapter.

In the right bottom corner a floating action button will be displayed with an icon of ‘ic\_add\_white\_24dp’, if the user taps on this fab the ‘VehicleEntityActivity’ is called and an empty vehicle entity is displayed.

If the user taps on the list item of a vehicle, then the ‘VehicleEntityActivity’ is called and screen is populated with vehicles data.

The name of the activity is ‘VehicleListActivity’, in the OnCreate method the content view is set to ‘app\_drawer’ then the ‘layout\_container’ is inflated with activities layout that is stored in the file ‘activity\_vehicles\_list’.

The associated ‘recyclerview’ adapter has the name ‘VehiclesListAdapter’ and is responsible to populate the ‘recyclerview’ with the data. For the Items it will use the layout file ‘item\_list\_vehicle’.

Each vehicle item is based on the component ‘CardView’ which holds an ‘ImageView’ that fills the whole ‘CardView’ with a ‘TextView’ for the vehicle’s name. The ‘TextView’ is above the ‘ImageView’ and in the bottom of the ‘CardView’.

The

Describe the next task. For example, “Implement Google Play Services,” or “Handle Error Cases,” or “Create Build Variant.”

Describe the next task. List the subtasks. For example:

* Create layout
* Something else

## Task 4: Your Next Task

Describe the next task. List the subtasks. For example:

* Create layout
* Something else

## Task 5: Your Next Task

Describe the next task. List the subtasks. For example:

* Create layout
* Something else

Add as many tasks as you need to complete your app.

**Submission Instructions**

* After you’ve completed all the sections, download this document as a PDF [ File → Download as PDF ]
  + Make sure the PDF is named “**Capstone\_Stage1.pdf**”
* Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

* Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
* Add this document to your repo. Make sure it’s named “**Capstone\_Stage1.pdf**”