GitHub Username: isdsava

BrowserWowser

Point of Note!

Dear Reviewer… This is going to become a real world app. One issue with that is you need your mobile phone number to receive your user-key to access the data. This might be an issue if you are not in Australia or running an emulator. Please let me know if this problematic? This data is coming from a 3rd party API that is pretty protective of the data, so they are very strict with accessing.

Description

Fuel is expensive in Australia. We have long distances to cover. This app provides the user with access to exclusive data, only previously available to the petrol companies about current, up to the minute prices.

Intended User

Our users will be anyone who drives in Australia. Anyone who hates paying for fuel only to drive two minutes down the road and find it 5 cents a litre cheaper. Anyone who is travelling on the road, in a new town or just in the next suburb.

Features

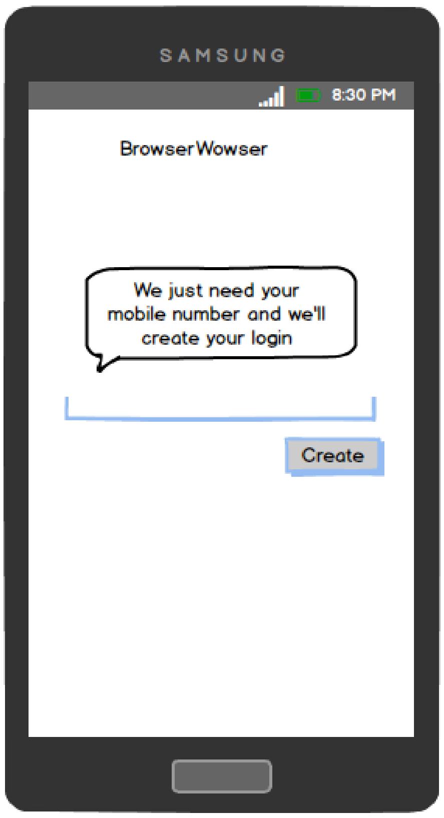
The app finds the nearest fuel stations for your current location and gives their current prices. You will be able to set the radius for the search to enable you to expand or contract your possible matches. It will save you fuel preference type and display the calculated longitude and latitude in its settings.

Future Features

App in future will interface into Android auto and possibly wearable to provide key information to user. For example the app on auto/wearable may let the user know that they have just entered an area that has the cheapest fuel in to current town/area/city. This can be switched off too.

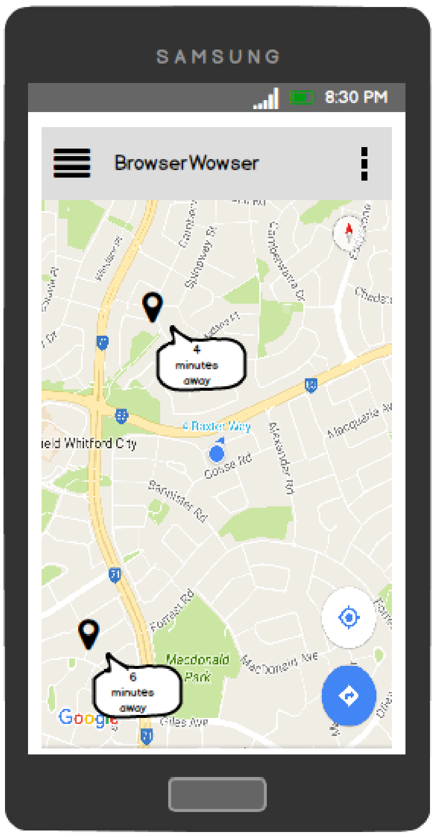
User Interface Mocks

Create Login screen- 1 time only



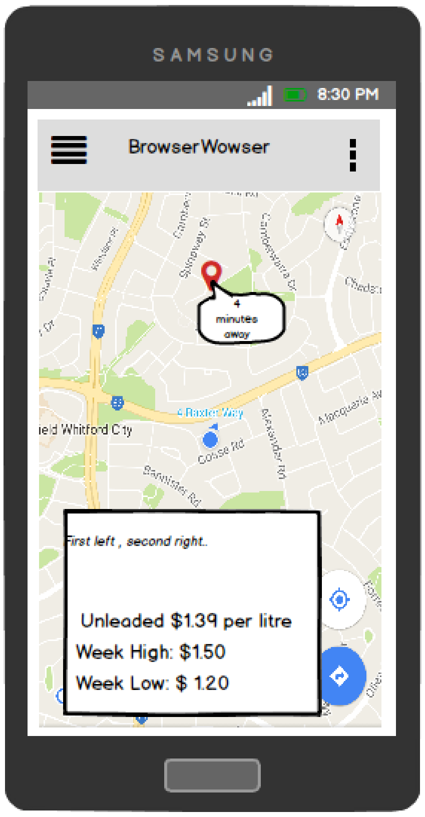
The external API has strict controls on who accesses the data, hence we need to use a mobile phone number to register for the user-key. Unfortunately there is no way around this, well maybe !

Locations Overview



Based upon you long/lat the app will give you a google maps overview of the fuel stations and the estimated drive time to reach.

View Single Station



First left , second right..

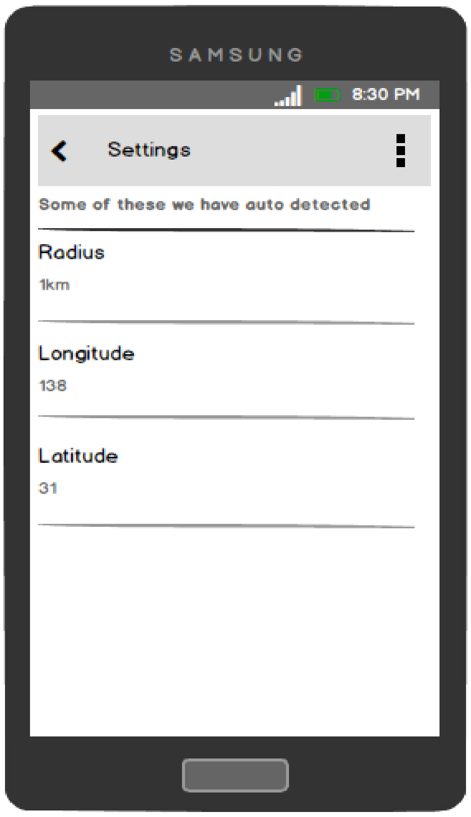
Unleaded $1.39 per litre

Week High: $1.50

Week Low: $ 1.20

By selecting the station the app report the price for you selected fuel type via a card. The card will display address and directions.

Settings



Some of these we have auto detected

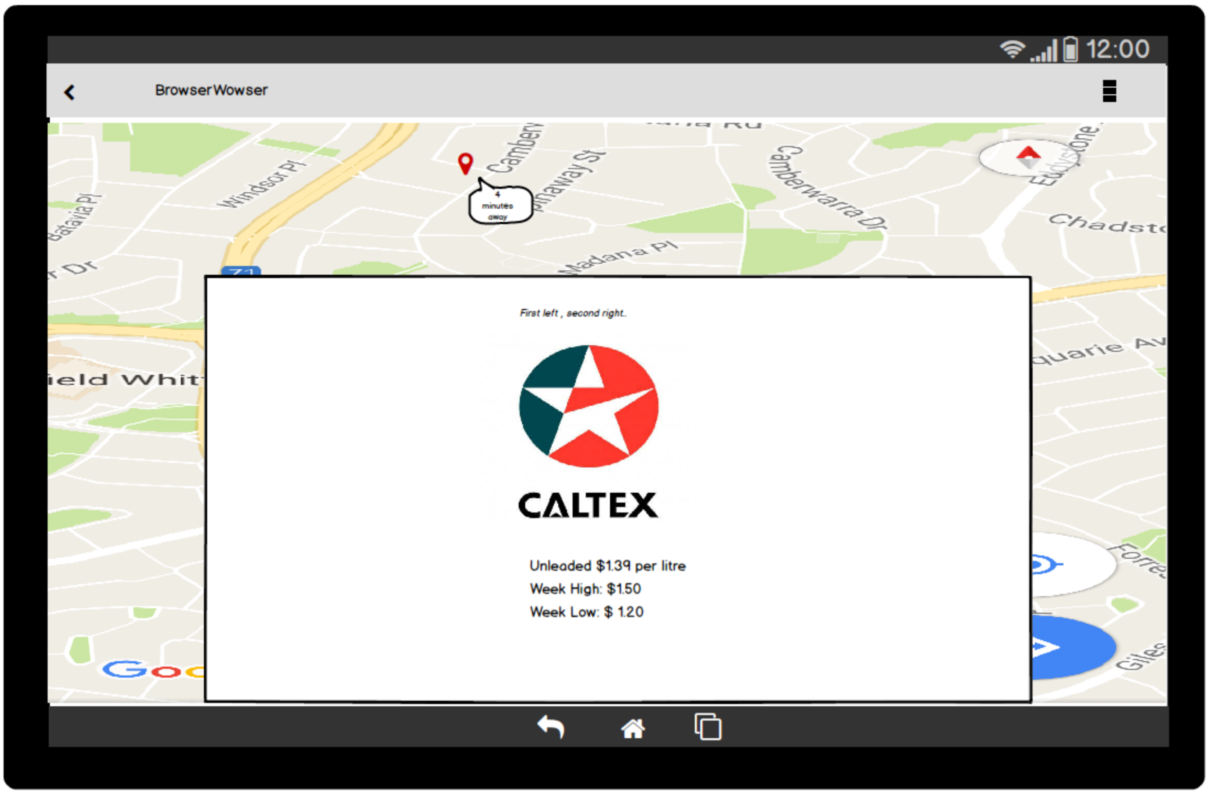
Radius
1km

Longitude
138

Latitude
31

Tell the app how far afield you want it to look for fuel, perhaps if you travelling that day you might expand the radius. This will also need to store your preferred fuel type.

Tablet View



Unleaded $1.39 per litre
Week High: $1.50
Week Low: $ 1.20

Tablet view with card details and directions

Key Considerations

How will your app handle data persistence?

- The app will implement both a preferences settings and Content provider backed by an SQLite database. The preferences will hold basic user locational, preferential information, fuel type and API user keys. The Content provider will persist pricing data until it requires refreshing.

Describe any corner cases in the UX.

- Using the Content provider/SQLite set up will enable the user to have some responsiveness from the app even it internets are lost.
- Default long/lat will ensure if for whatever reason unable to connect to fusionlocation that some data is displayed.

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

- We'll also be including Retrofit for handling Json parsing and Picasso for any images.

Describe how you will implement Google Play Services.

- We'll be including both google play services location and google play services maps. Location to utilise the fusedlocation api and maps for the obvious functionality they will bring to the app. We'll also be using admob to provide add revenue for the app.

Task 1: Project Setup

- Create an android studio project ensuring gradle has included the required google play libraries and also the retrofit and Picasso libraries too.
- Review and adjust the manifest as needed.

Task 2: Create Google Cloud end point- App Service

- The app service needs to take the mobile phone supplied in task 3 and add app token and subscriber tokens to request to the registration token. This will be a java module that will handle incoming requests and store them to the SQLite database. The service will need to take a Uri/Url and extract and act on the data. It will need to schedule its own updates to location data and store them in SQLite database.

App service will make the calls to following API's, process and store

- GetBrands
- GetCities
- GetFuelTypes
- GetFullSiteDetails

Task 3: Create Google Cloud end point- SQLite database

- This will store core login details for users. It will also store data that isn't frequently refreshed, eg store locations and other details. Not static data but data that is refreshed only every 24hours and is initiated by the app service not the users.

Task 4: Implement Login/Authorisation activity

- Login Creation: Have a stripped back, ultra-simple UI for the initial start of the app to capture the mobile phone number. We need to create a very simple way to grab the resulting text message for the API provider for the user key. Perhaps the undocumented "content-sms". Otherwise user needs to access and then submit the auth-key before accessing the API.

- This needs to be relayed through an app service. Upon success this will see a registration token sent the end user app which then needs to be supplied to the data provider API directly- not via the app service. Upon success of this the user will have User token to be permanently stored on the user device.

Task 5: Main Activity and maps fragment

- Create the default UI for the app. This is the screen the user will see on all subsequent starts of the app. It needs to access fusedlocation api to the users current location and then apply these to googlemaps and the API provider for access the fuel data. It will also serve ads via admob.

- The app will submit a RetreiveCheapestSitePrices(RCSP) request. This will need to be made from the context of the users current location and compare to the locations supplied. The App service will need a method to work out the closest sites to the app user from data supplied by the app user. This will need to be returned to the app to help formulate the request to the RCSP.

Task 6: Detail Fragment

- Create the UI for the displaying the selected fuel station. Needs to provide directions and price. Will provide a card to display this.

- This will need to take the tapped station and provide basic directions using google maps. It will also need to query the API for in-depth price data.

- May consider and interstacial ad here too, prior to detail launch.

Task 7: Create preferences UI

- Create the UI for setting preferences. Needs to handle storing the detect lat/long and also the users fuel type and radius.

Task 8: Create the SQL database contract and content provider

- Extend the SQLiteHelper
- Extend the Content Provider
- Create the Contract

Task 9: Create the Sync Adapter and associated services

- Extend the Sync Adapter
- Create the Adapter

Task 10: Create the Widget

- Extend the AppWidget Provider
- Extend the IntentService
- Create the preview image