GitHub Username: vrendina

Carculator - Car Lease and Loan Calculator

Description

You're shopping for a new car and you want to make sure you are getting the best deal. Whether you are in the market for a lease or you want to own your next car, Carculator provides accurate pricing calculations and allows you to quickly compare quotes from multiple dealerships.

Intended User

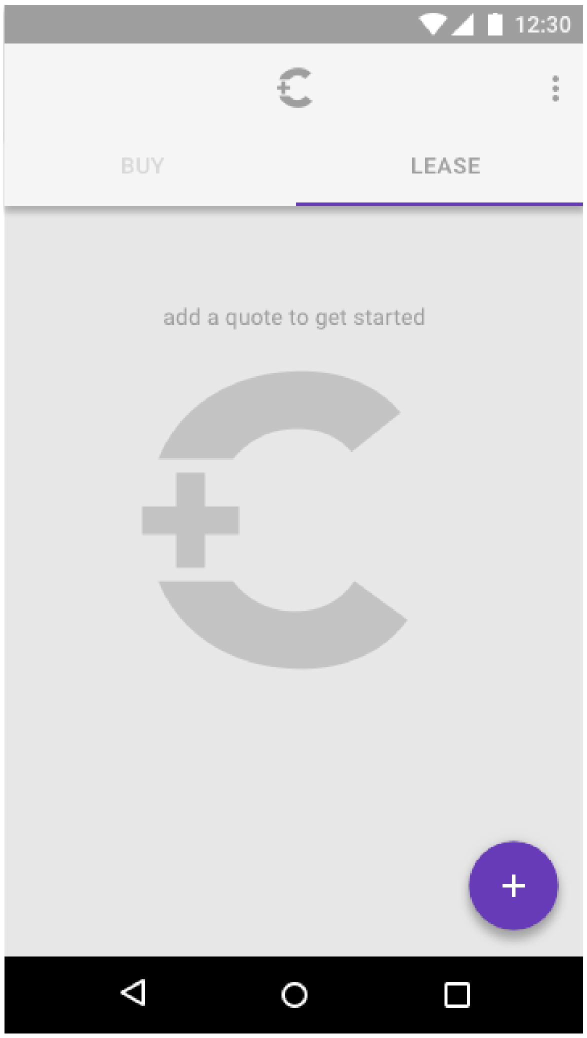
The intended user is anyone in the market for a new car who needs help determining if they are getting the best price.

Features

- Calculates payments for lease and loan quotes
- Displays costs in way that can be compared easily

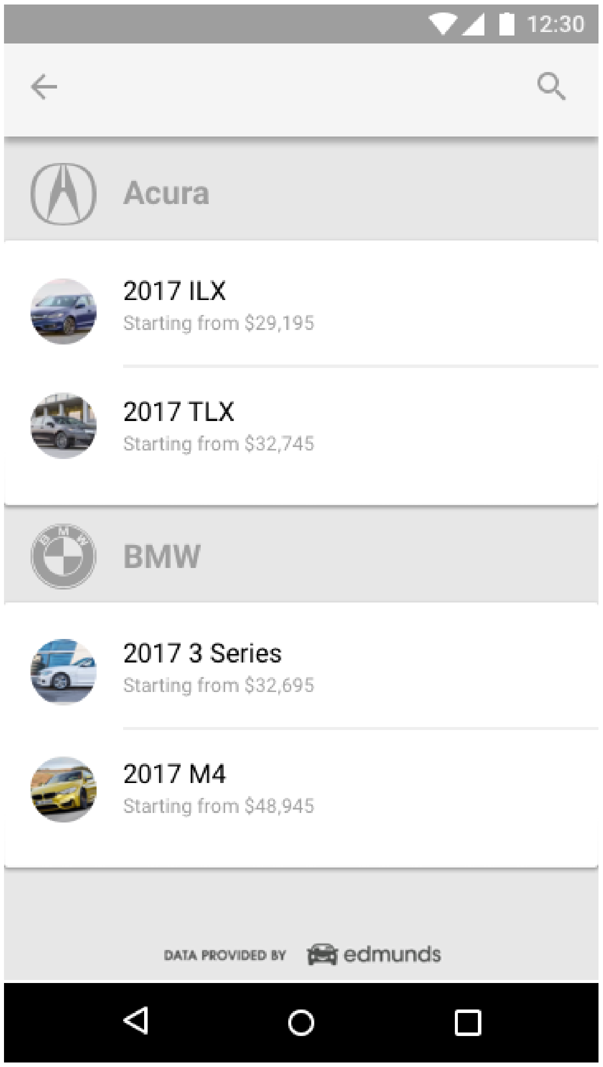
User Interface Mocks

User interface mockups created using Sketch.



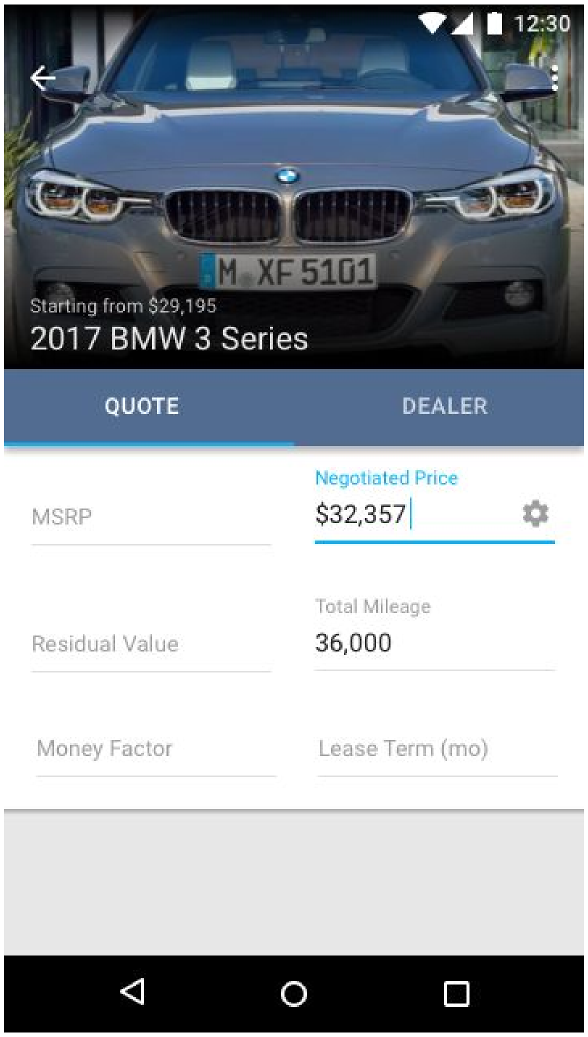
Home Screen, No Quotes Entered Yet (QuoteListActivity)

Entry screen for application. TabLayout will be used to differentiate between vehicle quotes for buying or leasing. The floating action button will be used to add quotes.



Vehicle Selection (VehicleListActivity)

List of vehicles will be displayed in a RecyclerView with a header row for the vehicle make. A SearchView will need to be implemented to filter the RecyclerView so a vehicle can be quickly selected. This data will be retrieved from a server and then cached locally in a ContentProvider backed by a SQLite database. The Firebase JobDispatcher will be used to schedule an update every 2 weeks.


12:30

Starting from $29,195

2017 BMW 3 Series

QUOTE DEALER

MSRP

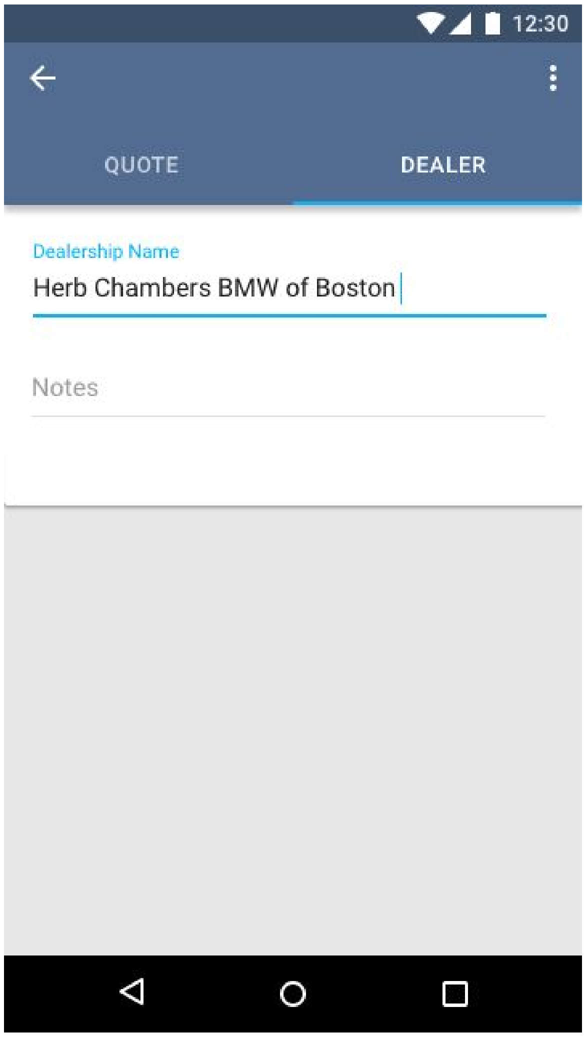
Negotiated Price: $32,357

Residual Value

Total Mileage: 36,000

Money Factor

Lease Term (mo)


12:30

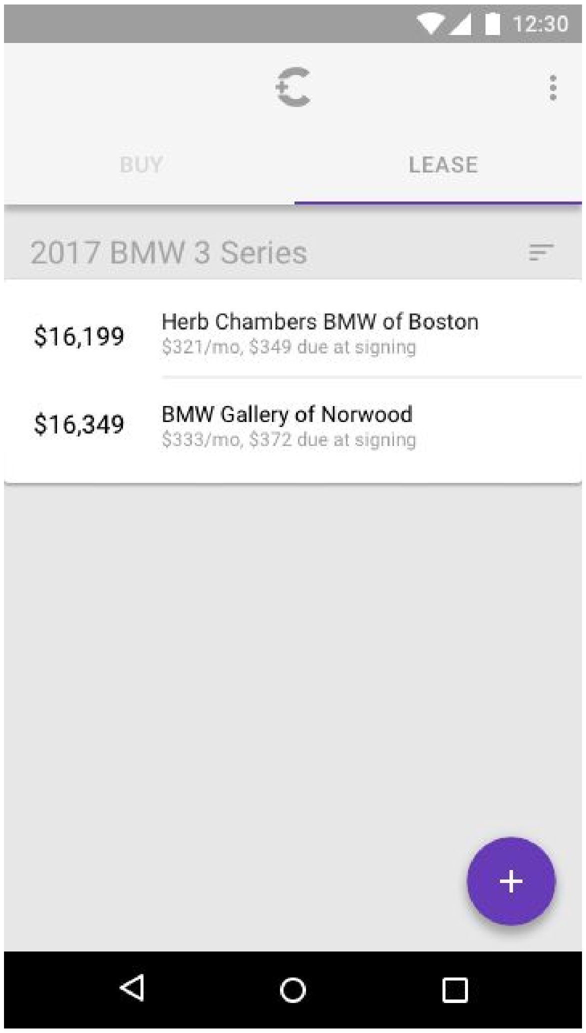
QUOTE DEALER

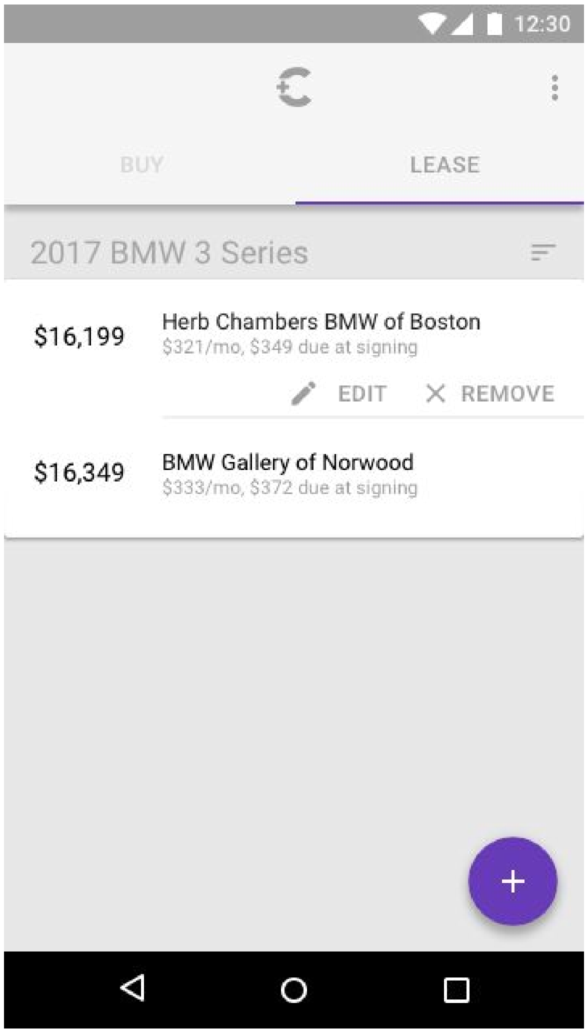
Dealership Name: Herb Chambers BMW of Boston

Notes

Quote Entry (QuoteEntryActivity)

Quote entry will consist of a RecyclerView will a GridLayout. Once the quote has been entered pressing the back button will automatically save the quote or save the progress and return to the home screen (not the vehicle selection Activity).


12:30
BUY
LEASE
2017 BMW 3 Series
$16,199
Herb Chambers BMW of Boston
$321/mo, $349 due at signing
$16,349
BMW Gallery of Norwood
$333/mo, $372 due at signing


12:30
BUY
LEASE
2017 BMW 3 Series
$16,199
Herb Chambers BMW of Boston
$321/mo, $349 due at signing
EDIT
REMOVE
$16,349
BMW Gallery of Norwood
$333/mo, $372 due at signing

Home Screen, Populated (QuoteListActivity)

The home screen will display quotes in a RecyclerView and the data will be stored in a Firebase Real Time Database. Since the RecyclerView is inside a TabLayout, using swipe actions would not be appropriate. Pressing on the quote in the list will expand a menu (right screenshot) that allows the user to edit their quote.

The sort icon in the top right of the vehicle section will allow the user to sort by total cost or monthly payments (in ascending order only). Pressing this icon should open a context menu to select the sort method.

Key Considerations

How will your app handle data persistence?

- A ContentProvider will be used to store vehicle make and image url location data and the data will be synchronized from a backend server using the Firebase JobDispatcher.
- Any user generated quote data will be stored in the Firebase Real Time Database.

Describe any corner cases in the UX.

I don't know if this is a "corner case" but hitting the back button from the QuoteEntryActivity will return directly to the QuoteListActivity and save any changes in progress.

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

Timber -- To simplify/centralize logging. Will also integrate a tree with Firebase Crash Reporting to provide additional information in the event of an application error or crash.
Glide -- For loading and caching vehicle image data
Butterknife -- To simplify getting references to views
Stetho -- For monitoring database contents during development
Firebase JobDispatcher -- To support job scheduling on pre API 21

Describe how you will implement Google Play Services.

Firebase Analytics -- Used for tracking user flow throughout the app and events will be triggered when the user adds a quote or deletes a quote.
Firebase Crash Reporting -- Keep track of application crashes in production
Firebase Real Time Database -- Used for storing user quotes
Firebase Authentication -- Only anonymous authentication will be used so that quotes can be stored for specific users in the Firebase database. This will allow a simpler transition to an actual authentication method in the future so users could synchronize their data across devices.

Next Steps: Required Tasks

Task 1: Project Setup

- Import required libraries
- Configure logging with Timber and activate Stetho which will aid in development

Task 2: Calculation Library

A Java library will be written that calculates lease/loan payments. This library will be reusable and imported into the application as part of the Gradle build process.

- Write lease/loan calculation library
- Write jUnit tests to verify calculations

Task 3: Testing Data

- Generate sample vehicle model data and store on remote server for synchronization
- Download images that can be used with tools:src for building the interface
- Add sample quote data to the Firebase Real Time Database

Task 4: Vehicle List

- Build UI for VehicleListActivity
- Implement service for downloading vehicle data in the background
- Write filtering code for the SearchView
- Implement scheduled job to update data every 2 weeks
- Write error handling for situations where data can not be loaded and no data is available (don't tell the user if the sync every 2 weeks fails, we will still have data and can silently retry)

Task 5: Quote Form

- Build UI for QuoteFormActivity and fragments for tabs
- Create a library to generate the appropriate form fields for leasing or buying
- Validate all user inputs to ensure they make sense (i.e. residual value must be < negotiated price)

Task 6: Quote List

- Build UI for QuoteListActivity and fragments for tabs
- Connect to Firebase Real Time Database
- Implement sorting functionality

Task 7: Widget

- Build a widget to display the user's quote data on the home screen.

Task 8: Verify Accessibility and RTL support

- Make sure all strings are placed in the strings.xml file
- Verify appropriate content descriptions are provided for all content and that the app can be navigated with a D-pad

Task 9: Release

- Generate signing configuration and include it in the git repository