GitHub Username: mdalai

TrackVan

Description

This app is designed to van owner for tracking its usage within different drivers. For example, in a high school, the principal can be the owner of van and use this app to track and manage van's usage. The van can be shared between football team, soccer team, basketball team and international center.

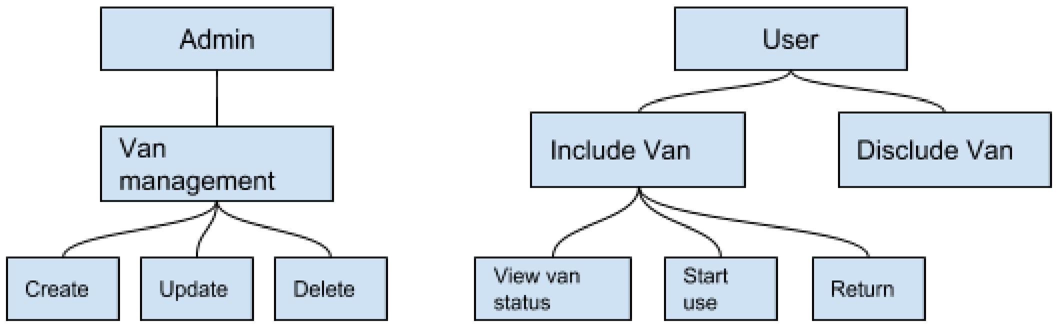
Intended User

Any organization or individual who share a van within group and wants to track its van usage.

Features

List the main features of your app. For example:

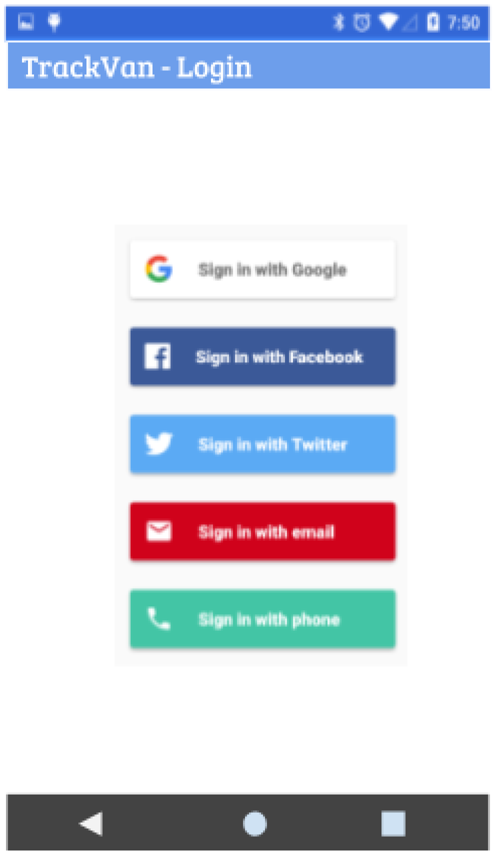
- User authentication - sign in, sign out.
- Van admin: create, update, delete van. Share van unique code to users by email or msg.
- User: include, disclude van. To use a van, has to include it first. User can do: view van status, start using van, return van.



User Interface Mocks

Screen set 1 - User authentication

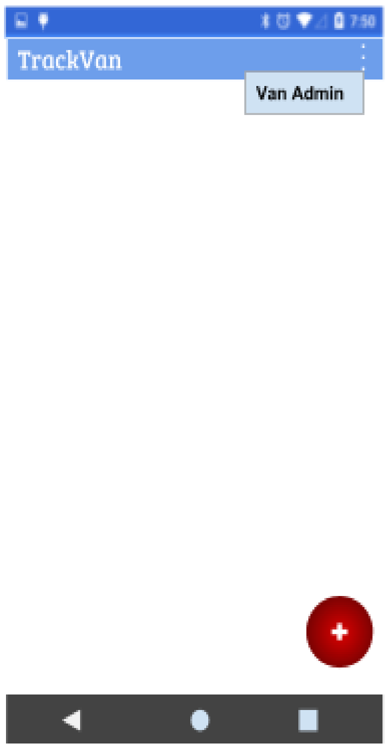
Login first, has following options to signup.

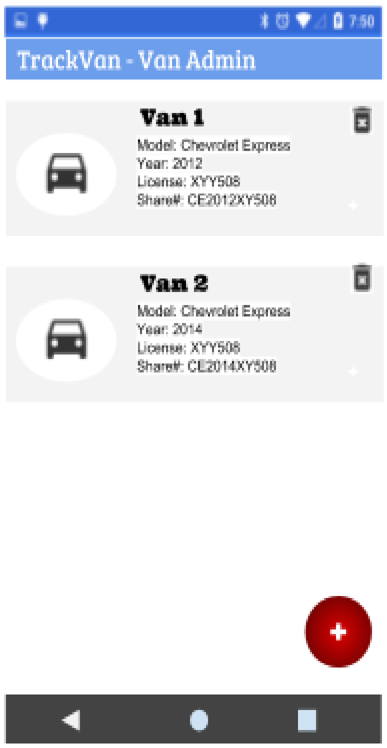


Screen set 2 - Van Admin

Go to Van Admin from Action bar.

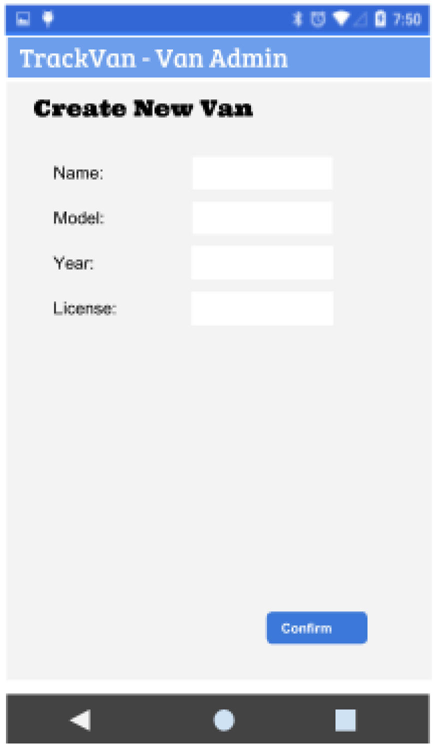
Van Admin: view, create, delete van.





Van 1
Model: Chevrolet Express
Year: 2012
License: XYY508
Share#: CE2012XY508

Van 2
Model: Chevrolet Express
Year: 2014
License: XYY508
Share#: CE2014XY508

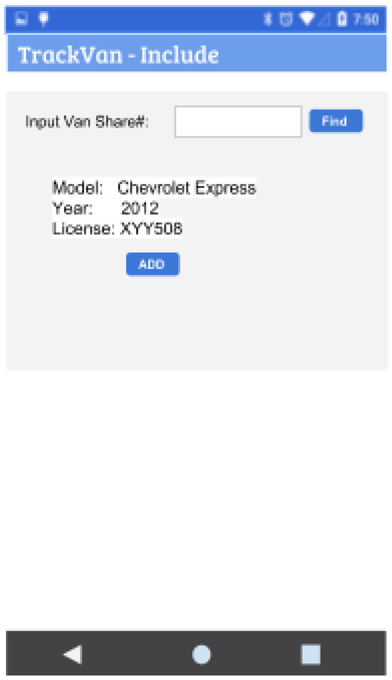


Create New Van

Name:	
Model:	
Year:	
License:	

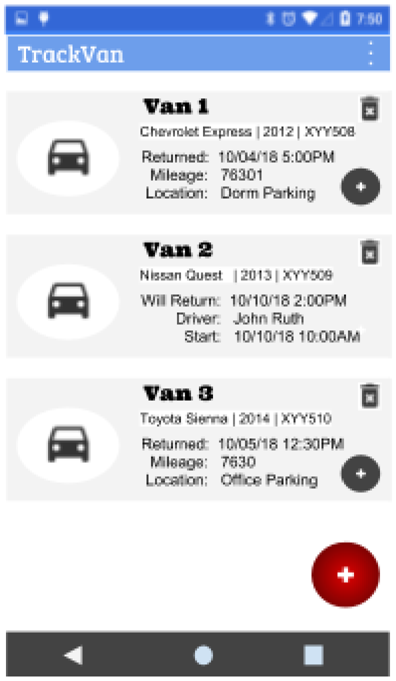
Screen set 3 - user

To use a van, has to include it first.
After login in, all vans included by users displayed. Each van should show the status of available or not available.
User can start using van if it is available.
User return van after finish using it.



Input Van Share#:

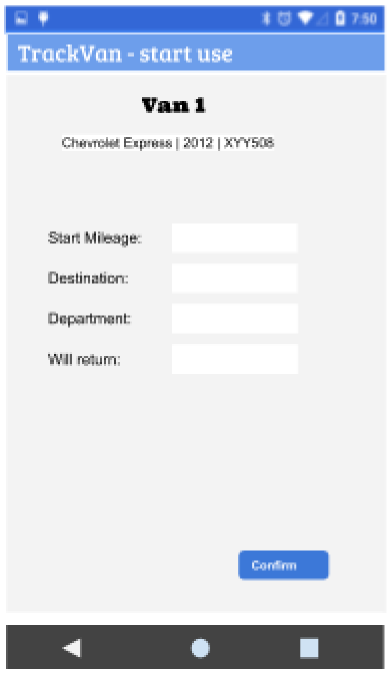
Model: Chevrolet Express
Year: 2012
License: XYY508



Van 1
Chevrolet Express | 2012 | XYY508
Returned: 10/04/18 5:00PM
Mileage: 76301
Location: Dorm Parking

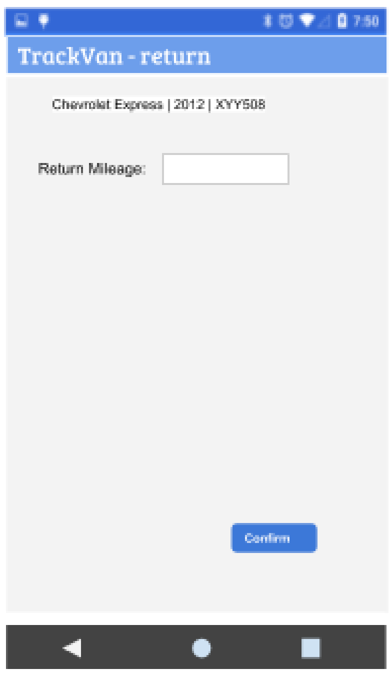
Van 2
Nissan Quest | 2013 | XYY509
Will Return: 10/10/18 2:00PM
Driver: John Ruth
Start: 10/10/18 10:00AM

Van 3
Toyota Sienna | 2014 | XYY510
Returned: 10/05/18 12:30PM
Mileage: 7630
Location: Office Parking



Van 1
Chevrolet Express | 2012 | XYY508

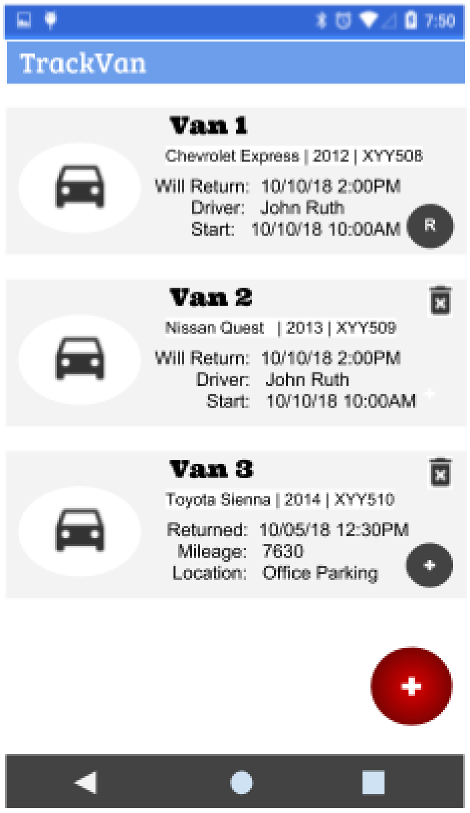
Start Mileage:
Destination:
Department:
Will return:



Chevrolet Express | 2012 | XYY508

Return Mileage:

Only user that is using the car can see the return button. See the difference between below 2 screen.

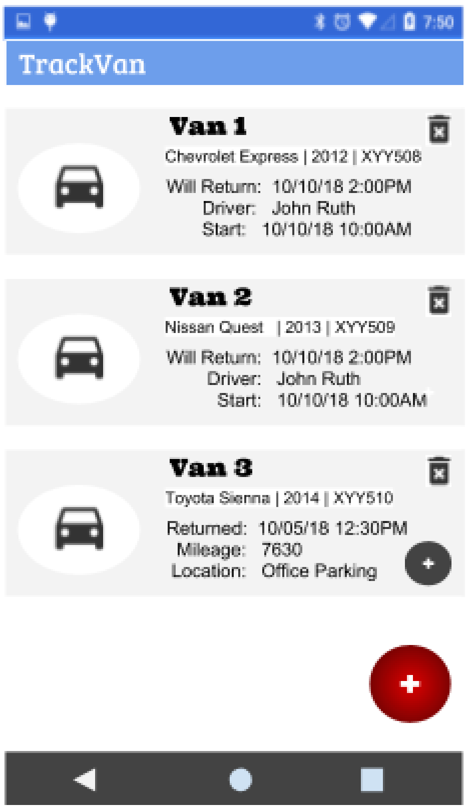


TrackVan

Van 1
Chevrolet Express | 2012 | XYY508
Will Return: 10/10/18 2:00PM
Driver: John Ruth
Start: 10/10/18 10:00AM

Van 2
Nissan Quest | 2013 | XYY509
Will Return: 10/10/18 2:00PM
Driver: John Ruth
Start: 10/10/18 10:00AM

Van 3
Toyota Sienna | 2014 | XYY510
Returned: 10/05/18 12:30PM
Mileage: 7630
Location: Office Parking



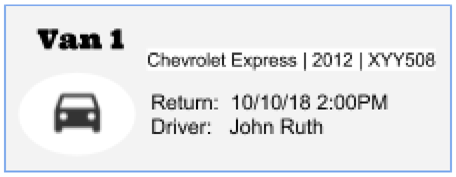
TrackVan

Van 1
Chevrolet Express | 2012 | XYY508
Will Return: 10/10/18 2:00PM
Driver: John Ruth
Start: 10/10/18 10:00AM

Van 2
Nissan Quest | 2013 | XYY509
Will Return: 10/10/18 2:00PM
Driver: John Ruth
Start: 10/10/18 10:00AM

Van 3
Toyota Sienna | 2014 | XYY510
Returned: 10/05/18 12:30PM
Mileage: 7630
Location: Office Parking

Screen set 4 - widget



Van 1
Chevrolet Express | 2012 | XYY508
Return: 10/10/18 2:00PM
Driver: John Ruth

Key Considerations

App development language

This app is going to be written solely in the Java programming language.

Stable release versions of all libraries, Gradle, and Android Studio

Item	Version
Android Studio	3.0.1
Gradle version	4.6
Android Plugin version	3.2.0-rc02

firebase-auth	15.0.0
firebase-database	15.0.0
play-services-ads	15.0.0
firebase-analytics	15.0.0
com.jakewharton:butterknife	9.0.0-rc1
com.jakewharton:butterknife-compiler	9.0.0-rc1
com.squareup.picasso:picasso	2.5.2

How will your app handle data persistence?

- Firebase user authentication
- Firebase realtime database

Describe any edge or corner cases in the UX.

For example, how does the user return to a Now Playing screen in a media player if they hit the back button?
No edge cases I can think of.

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

- Picasso for displaying icon image.
- ButterKnife for field and method binding for views.
- FirebaseUI library and Authentication for user sign-in and sign-out.
- Firebase Realtime Database for data persistence.
- Firebase AdMob for ad.
- Firebase Analytics for tracking user flows, to understand how users interacting with app.

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

- Firebase Authentication: adapt FirebaseUI that provides drop-in UI flows for use.
- AdMob: add AdView placeholder and Google handles the ad delivery for you.
- Analytics: add firebase-analytics dependency, and initialize it. You will be able to see the report in Firebase console.

Fetch image with AsyncTask

The van image is loaded from a web service using AsyncTask.

Other important considerations

- The App refer all the hardcoded strings from the strings.xml file.

- The app enables RTL layout switching to support accessibility on RTL supported languages.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

Create new project in Android Studio:
- Configure App Theme.
- Configure internet use permissions.
- Configure libraries such as ButterKnife, Picasso.

Create Firebase console project:
- Register and sign in Firebase. Go to the Firebase console.
- Add project and name it “TrackVan”.
- Connect the Android app.
- Add google-services.json file to your app
- Add google-services plugin to your app.
- Sync
- Check out this Firebase Android Colab for reference.

Setup firebase in your project:
- Firebase Assistant in the Android Studio is the simplest way to connect your app to Firebase. It deals with with all necessary gradle dependencies.

Task 2: Implement UI for Each Activity and Fragment

UI for User model
- Build UI for MainActivity. This shows list of van user included.
- Build UI for IncludeVanActivity.
- Build UI for StartUseVanActivity.
- Build UI for ReturnVanActivity.

UI for Van Admin model
- Build UI for VanAdminActivity. This shows list of van user created.
- Build UI for AddVanActivity.

Task 3: Create Authentication

Authentication

- Setup Rules: the Rules handles realtime database access control.
- Configure Auth APIs
- Add Auth dependency in Gradle.
- Modify MainActivity.java to send the user sign-in screen.
- Implement the sign-in screen. SignInActivity.java
- Check this colab for detail.

Task 4: Create Database

Database

- Add firebase-database dependency in app/build.gradle.
- Initialize database and add a listener to handle changes made to the data.
- Update RecyclerView adapter for showing data from database.
- Add database instance.
- Use push() to store data in database.
- Check this codelab for detail.

Task 5: Add AdMob

Ads

- Add AdMob dependency.
- Add ads namespace in the layout: `xmlns:ads="http://schemas.android.com/apk/res-auto"`
- Add AdView to main layout: `com.google.android.gms.ads.AdView`
- Add AdView variable, request Ad, handle lifecycle events.
- Check this codelab for detail.

Task 6: Add Firebase Analytics

Analytics

- Add Analytics dependency.
- Initialize FirebaseAnalytics in MainActivity.
- Check this codelab for detail.