

Description

Intended User

Features

User Interface Mocks

Screen 1

Screen 2

Key Considerations

How will your app handle data persistence?

Describe any corner cases in the UX.

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

Next Steps: Required Tasks

Task 1: Project Setup

Task 2: Implement UI for Each Activity and Fragment

Task 3: Your Next Task

Task 4: Your Next Task

Task 5: Your Next Task

GitHub Username: raheel

Vehicle Maintenance Reminder

Description

This app shows the user the maintenance schedule for their vehicle and allows them to be set notification. It talks to the Edmunds Vehicle APIs to get the list of vehicle by year, make, and model, and allows the user to set reminders on each of the maintenance item.

Intended User

Car owners who like to stay organized and/or have a habit of forgetting about when to take their car for maintenance.

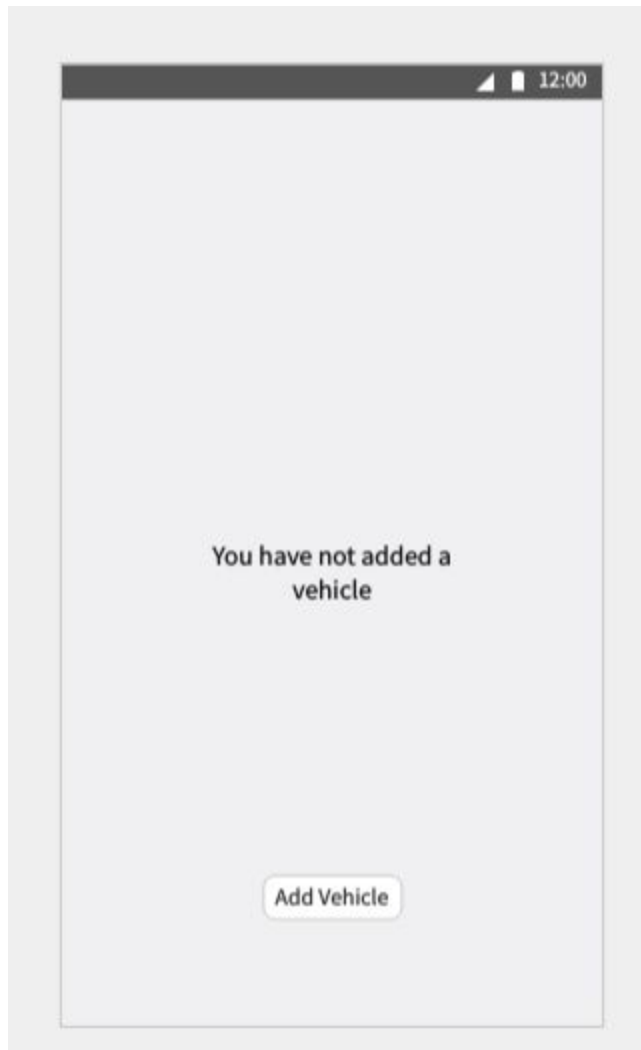
Features

- Saves Vehicle and Maintenance Schedule Information
- User is able to select which maintenance they want to set a scheduled notification for

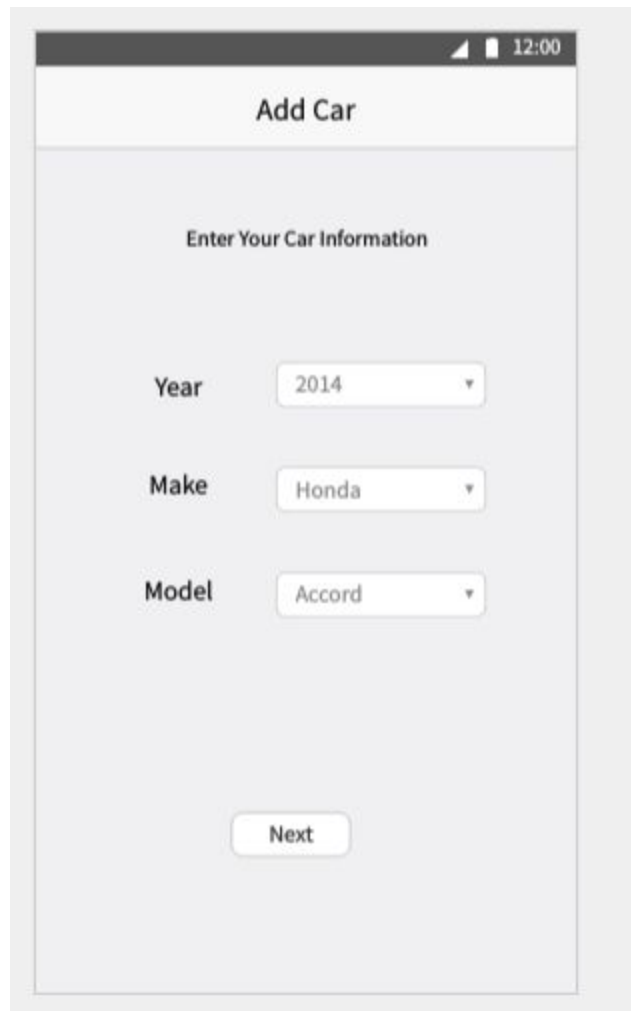
User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Home Screen (no vehicles)



Add Vehicle Screen



A mobile application screen titled "Add Car". At the top, a status bar shows a signal icon, a battery icon, and the time "12:00". Below the status bar is a white header bar with the title "Add Car". The main content area has a light gray background and contains the text "Enter Your Car Information". Below this text are three rows of form fields, each with a label on the left and a dropdown menu on the right. The first row is labeled "Year" and has a dropdown menu showing "2014". The second row is labeled "Make" and has a dropdown menu showing "Honda". The third row is labeled "Model" and has a dropdown menu showing "Accord". At the bottom of the form is a white button with the text "Next".

Add Car

Enter Your Car Information

Year 2014 ▼

Make Honda ▼

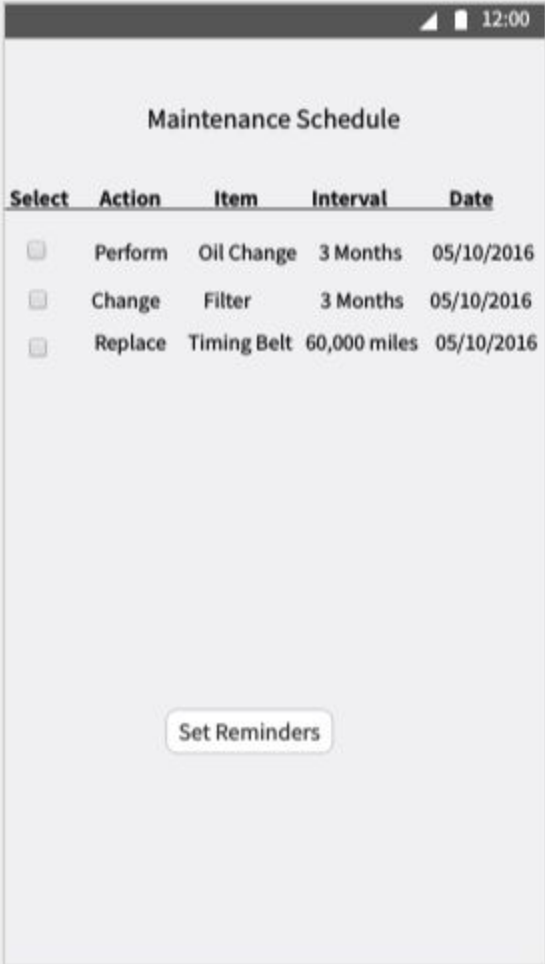
Model Accord ▼

Next

Add Additional Information Screen - This will be used to calculate on when the next maintenance is due.

The image shows a mobile application interface for car maintenance. At the top, a dark status bar displays the time 12:00. Below this, a white header bar contains the title "Car Maintenance". The main content area has a light gray background and is titled "Enter Additional Information". It contains two text input fields: the first is labeled "How many miles do you drive in a week?" and the second is labeled "Odometer Reading". Both fields have a light gray border and rounded corners. At the bottom of the screen, there is a white button with the text "Next".

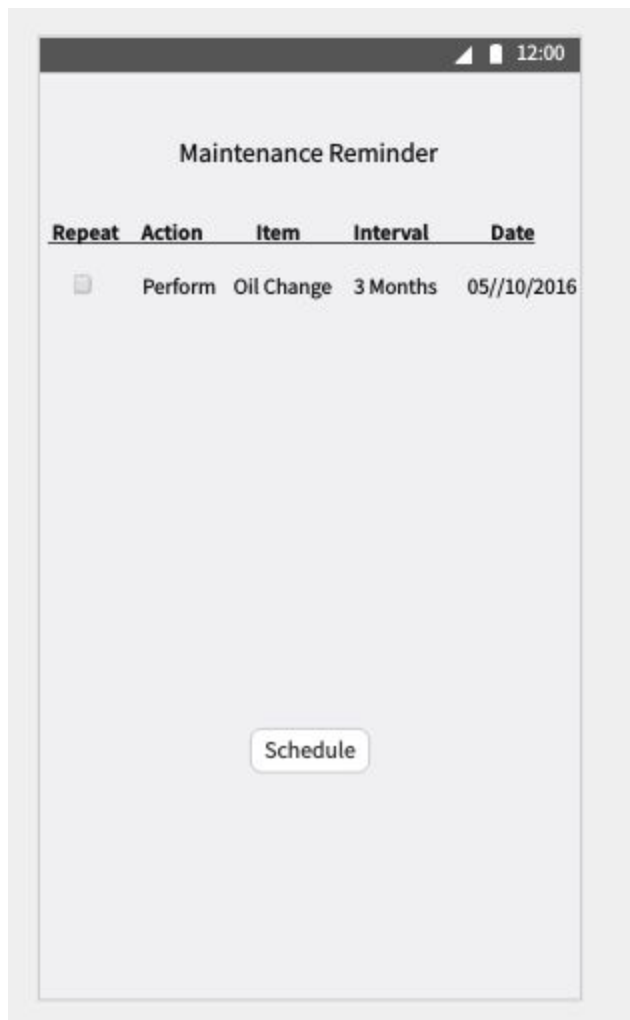
Display All Maintenance Screen



Select	Action	Item	Interval	Date
<input type="checkbox"/>	Perform	Oil Change	3 Months	05/10/2016
<input type="checkbox"/>	Change	Filter	3 Months	05/10/2016
<input type="checkbox"/>	Replace	Timing Belt	60,000 miles	05/10/2016

Set Reminders

Setting Maintenance Reminders - also allows user to select if they would like to repeat the reminder



The screenshot shows a mobile application interface titled "Maintenance Reminder". At the top, there is a status bar with a signal icon, a battery icon, and the time "12:00". Below the title, there is a table with the following columns: "Repeat", "Action", "Item", "Interval", and "Date". The table contains one row of data: a checkbox icon in the "Repeat" column, "Perform" in the "Action" column, "Oil Change" in the "Item" column, "3 Months" in the "Interval" column, and "05//10/2016" in the "Date" column. At the bottom of the screen, there is a button labeled "Schedule".

Repeat	Action	Item	Interval	Date
<input type="checkbox"/>	Perform	Oil Change	3 Months	05//10/2016

Schedule

Home Page (with vehicle selected)



Key Considerations

How will your app handle data persistence?

Data will be stored in an SQLite database

Describe any corner cases in the UX.

When a user first uses the app, they will be shown a message saying that they will need to save a vehicle. There will be a button that goes to the enter your vehicle screen, once they sets his

vehicle, they will be shown their vehicle's recommended maintenance schedule, and they will be given the choice to set reminders.

If a vehicle has been set and the user starts the app, they will be shown the "My Scheduled Reminder" page.

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

For example, Picasso or Glide to handle the loading and caching of images.

- Picasso - Loading/caching images
- Retrofit - API calls
- RxJava - working with Observables
- Dagger - Dependency Injection
- Buterknife - View injection
- Android Job Library - Job scheduling

Next Steps: Required Tasks

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

Task 1: Project Setup

Write out the steps you will take to setup and/or configure this project. See previous implementation guides for an example.

You may want to list the subtasks. For example:

- Configure libraries
- Something else

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.

Configure libraries

- Setup all the required libraries in the app's build.gradle
- Create Dagger components, modules, and integrate it with the Application
- Create API Service for making calls to Edmund's Vehicle Maintenance API using Retrofit
- Setup project hierarchy

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity. This will contain fragments for
 - Add Vehicle and mileage
 - List Recommended Vehicle Maintenance
 - Setup schedule page
 - Confirmation Screen
 - Delete Vehicle
- Build SettingsActivity

Task 3: Setup SQLite Database

- Create ContentProvider
- Setup tables
- Create models for each table

Task 4: Setup Job Scheduler Framework

- Add ability to add/delete jobs

Task 5: Create Layout

- Create layout and focus on Material Design concepts such as CoordinatorLayout, Transitions, AppBarLayout etc.

Task 6: Testing

- Create unit tests
- Manual testing