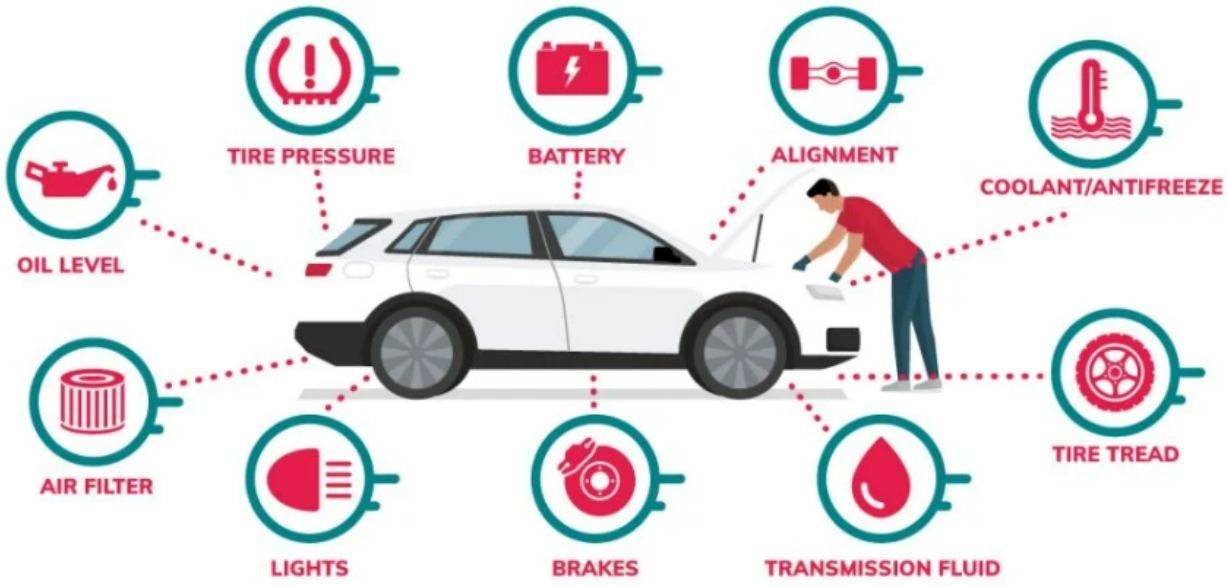
COURSE NAME : MOBILE APPLICATIONS (CS 443)

PROJECT TITLE:

CAR MAINTENANCE & MANAGEMENT APPLICATION



A Project Report by:

Heechul Choi

Karmesh S Chaudhari

Sukruth Kotturu

Project Statement

Car Management App

The “Car Management App” will help people manage and in a way keep track of their car maintenance history by assisting the owner of a respective car model by recording its history about maintenance, refueling, while considering the financial budget of the same. Additionally the user can also visualize the monthly fuel consumption of their car over an entire which is depicted by a bar chart.

Considering the current automobile market it has been observed that car owners apart from convenience and comfort also focus on the economical as well as environmental factors. Also, keeping a record of the information on a daily basis might be complicated and time consuming considering the user would be engaged with everyday tasks. Hence our team has come up with an idea for alleviating the problem while optimizing the costs. Since the users of this application could be car owners from the most common/basic model to high-end models all benefiting from a concept of recording maintenance history and keeping a track of their car-based expenses. Not only it would be beneficial to the users but also to the society where everyone would be preserving the environment by enhancing the efficiency of their motor vehicles.

Although there are several applications with the concept of recording maintenance history, very few can visualize the data into graphs and charts.

Also, just having a smart phone with an Android version 8.0 (minimum API 26) and an internet connection(WiFi or LTE) for updating the car database(maintenance history) would be more than sufficient to use the application.

Application Design

High-level design

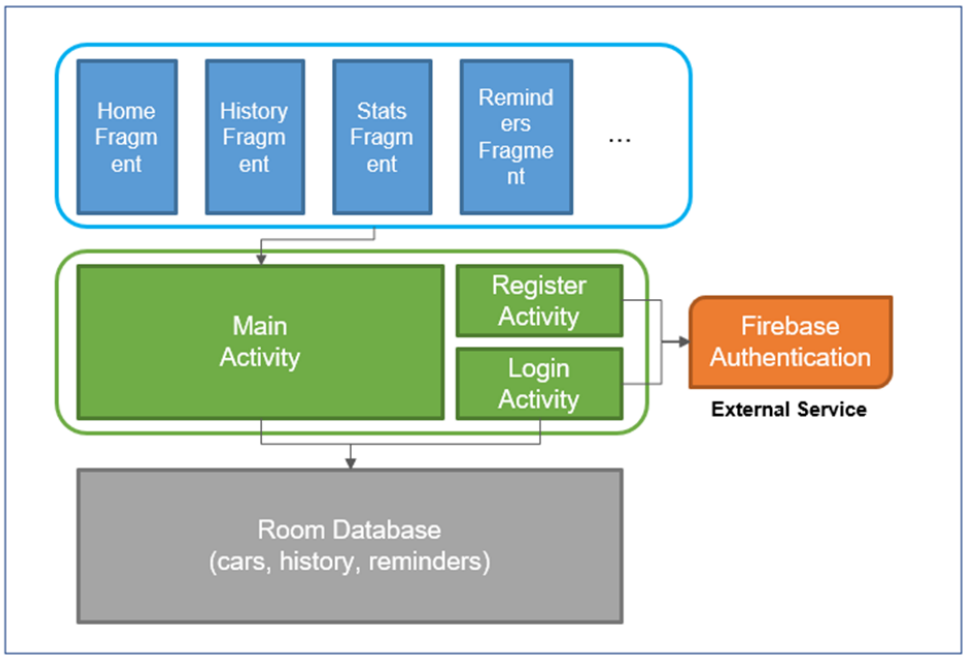


Figure 1:High Level Design Representation of the Car Management App

Fragment

A Fragment represents a reusable portion of your app's UI. A fragment defines and manages its own layout, has its own lifecycle, and can handle its own input events. Fragments cannot live on their own--they must be *hosted* by an activity or another fragment. The fragment's view hierarchy becomes part of, or *attaches to*, the host's view hierarchy.

Modularity

Fragments introduce modularity and reusability into your activity's UI by allowing you to divide the UI into discrete chunks. Activities are an ideal place to put global elements

around your app's user interface, such as a navigation drawer. Conversely, fragments are better suited to define and manage the UI of a single screen or portion of a screen.

Room Database

RoomDatabase provides direct access to the underlying database implementation.

Activities

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI.

Google Firebase

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in real time.

Considering Figure 1, when events occur, each fragment is replaced through Main Activity.

- When events occur, Home, History, Stats and Reminder fragments are replaced through Main Activity.
- Login Activity and Registration Activity were linked to Google Firebase.
- User data was stored in the app's Room database, and shared preference was used for simple preference.

Target Devices

Target devices are android devices with a minimum API 32 (Android version 12.0) with a minimum API 26 (Android version 8.0).

An emulator(Pixel 5.0) was used for the test purposes.

Application Implementation and Evaluation

Source Structure

The app was implemented by dividing the package on the basis of usage with respect to

→ **Adapter:**

An Adapter object acts as a bridge between an AdapterView and theunderlying data for that view. The Adapter provides access to the data items. The Adapter is also responsible for making a View for each item in the data set.

→ **Data:**

A persistable set of key/value pairs which are used as inputs and outputs for ListenableWorkers. Keys are Strings, and values can be Strings, primitive types, or their array variants.

→ **Fragments:**

A Fragment represents a reusable portion of your app's UI. A fragment defines and manages its own layout, has its own lifecycle, and can handle its own input events. Fragments cannot live on their own--they must be *hosted* by an activity or another fragment. The fragment's view hierarchy becomes part of, or *attaches to*, the host's view hierarchy.

→ **Notification:**

A class that represents how a persistent notification is to be presented to the user using the NotificationManager.

→ **Room:**

The Room persistence library provides an abstraction layer over SQLite to allow for more robust database access while harnessing the full power of SQLite.

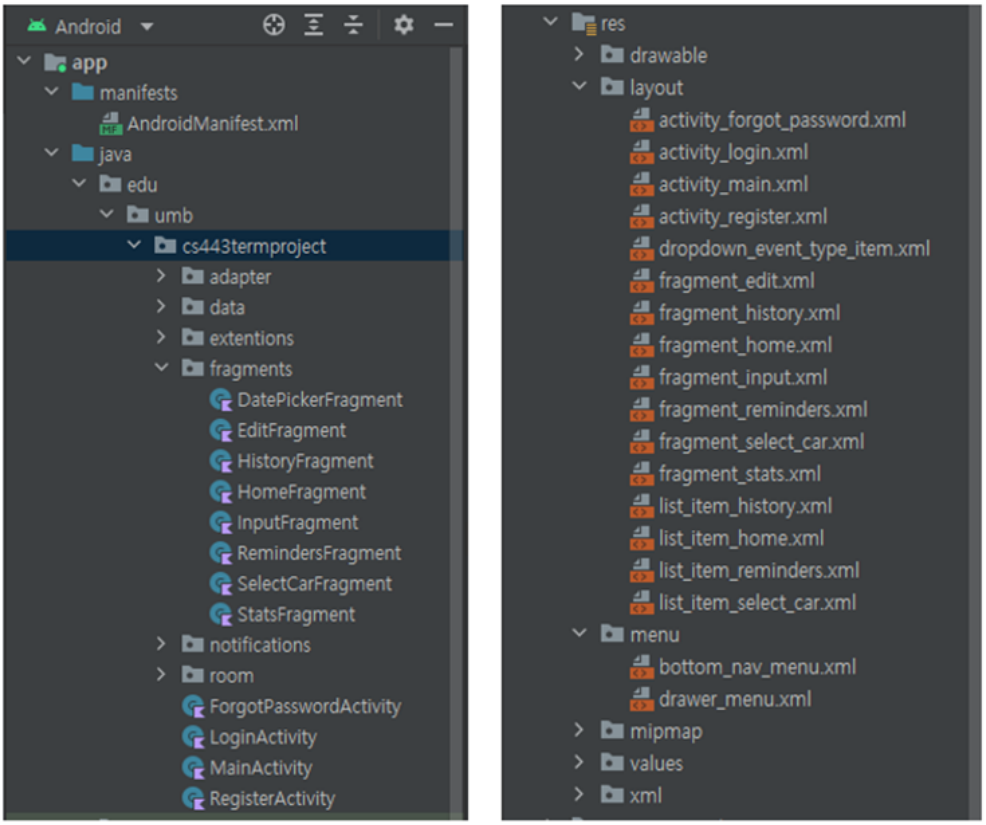


Figure 2: Source Structure

Features

1. Login / Register / Logout

You may register or log in with your email and password. (eg. test@gmail.com / test123). The login/Register process relates to Firebase Authentication.

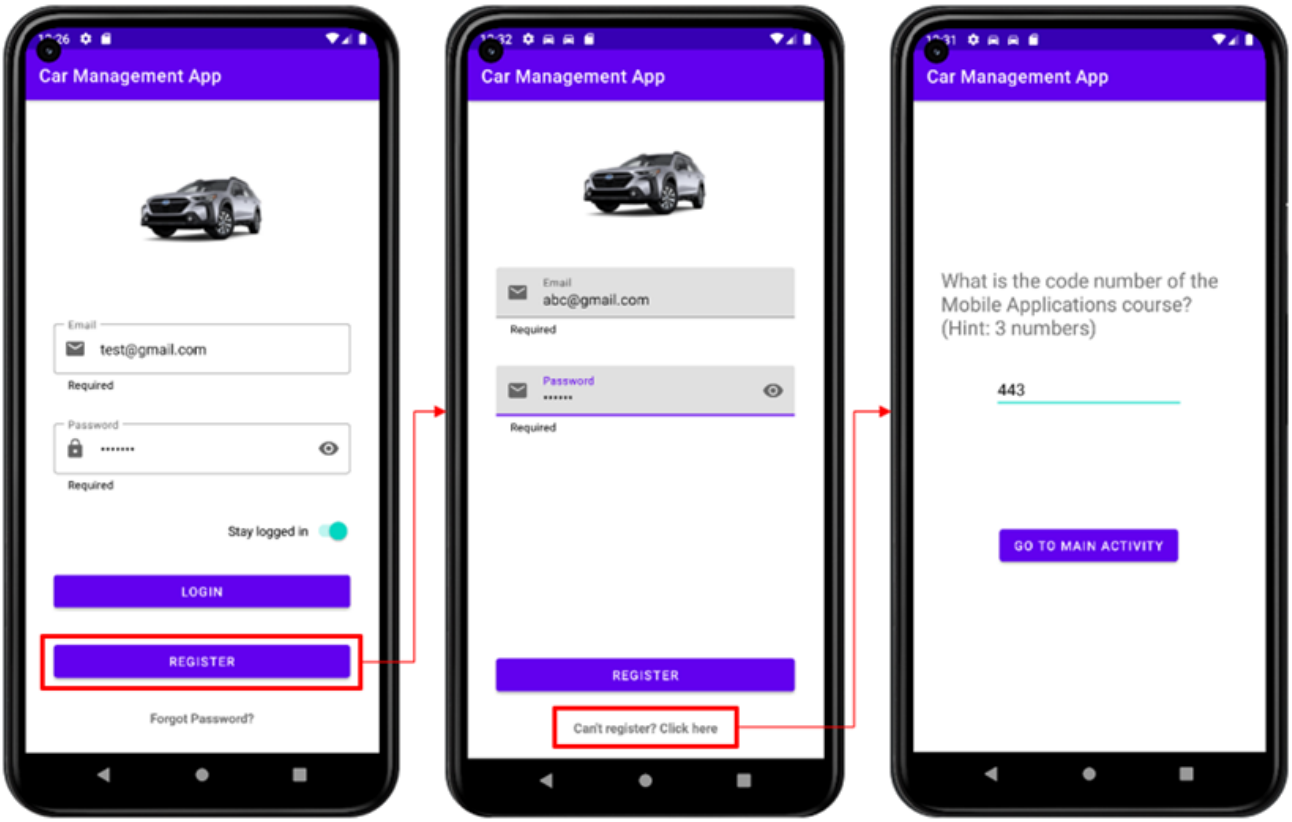


Figure 3: Login page UI

If Firebase Authentication failed, you can click the "forgot password" link on the Login page or the "Can't register" link on the Register page to see the Main page.

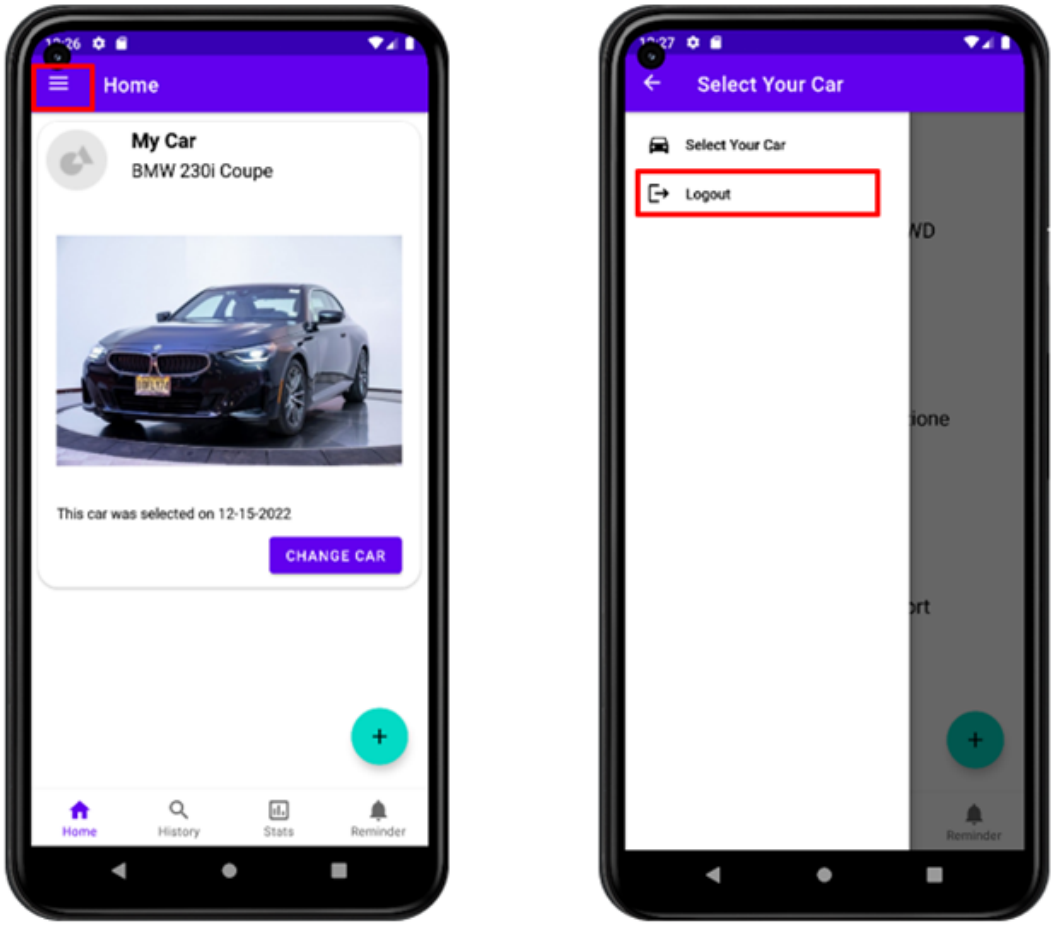


Figure 4: Source Structure

You can find the logout button by pressing the button on the upper left of the app.

2. Select Your Car (Home Menu)

There are 50 cars you can choose from. You may change your car by clicking the "Change Car" button or "Select Your Car" in the drawer menu.

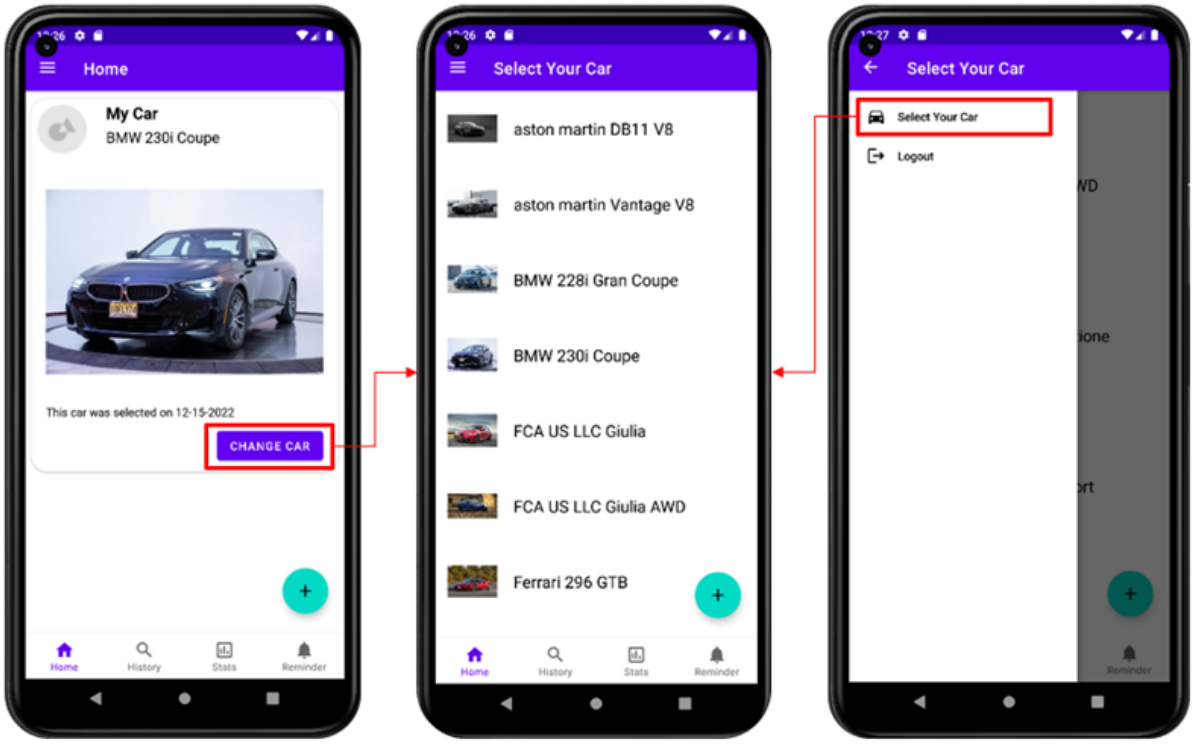


Figure 5: Car Selection Menu

3. History Menu

If there is no history data, you can see "ADD DUMMY DATA" button. You may add dummy data for test by clicking the "ADD DUMMY DATA" button. You can edit or delete the car maintenance history by clicking the item.

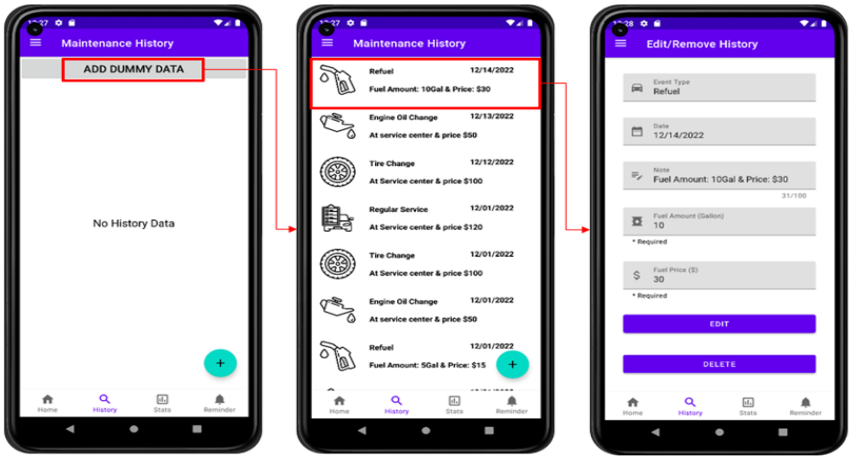


Figure 6: Maintenance History UI

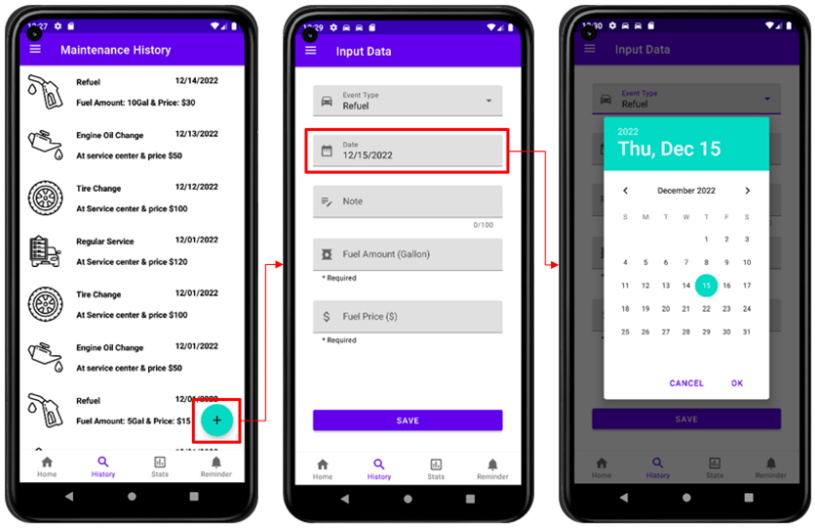


Figure 7: Adding data to maintenance History

You can enter your new car maintenance history (ex., refuel, change engine oil, change a tire, get regular service) by clicking the floating action button (+) on the lower right.

4. Statistics Menu

You can visualize the current year's fuel statistics as a sum of each month's fuel usage consumption amount.

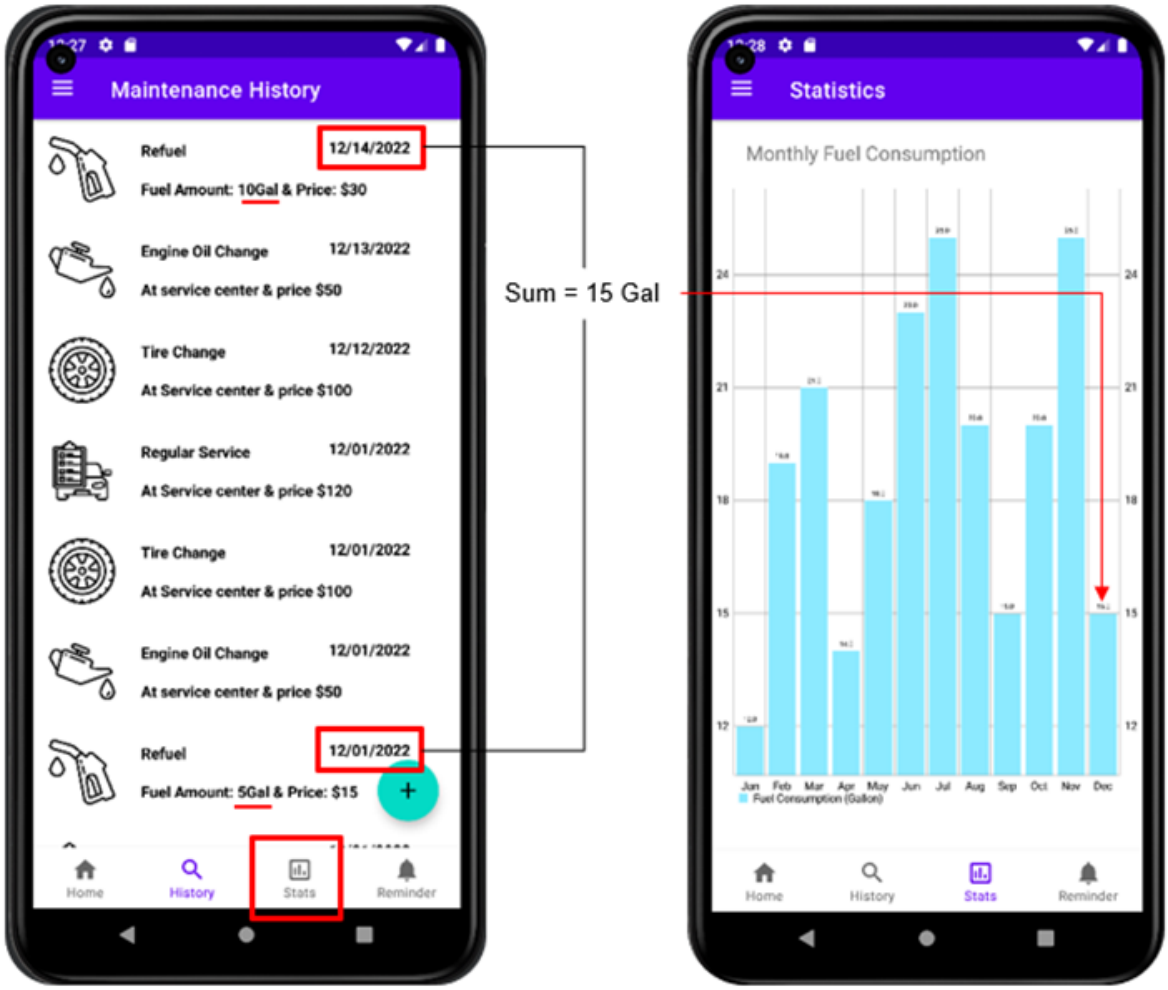


Figure 7: Visualizing Data using Statistics Menu

5. Reminder Menu

The app calculates the recommended next date for the car maintenance action (ex., refuel, Change Engine Oil, Change Tire, do regular service) and sets the reminders for the target date and time. You may turn on/off the reminders and alter the all reminder's alarm time.

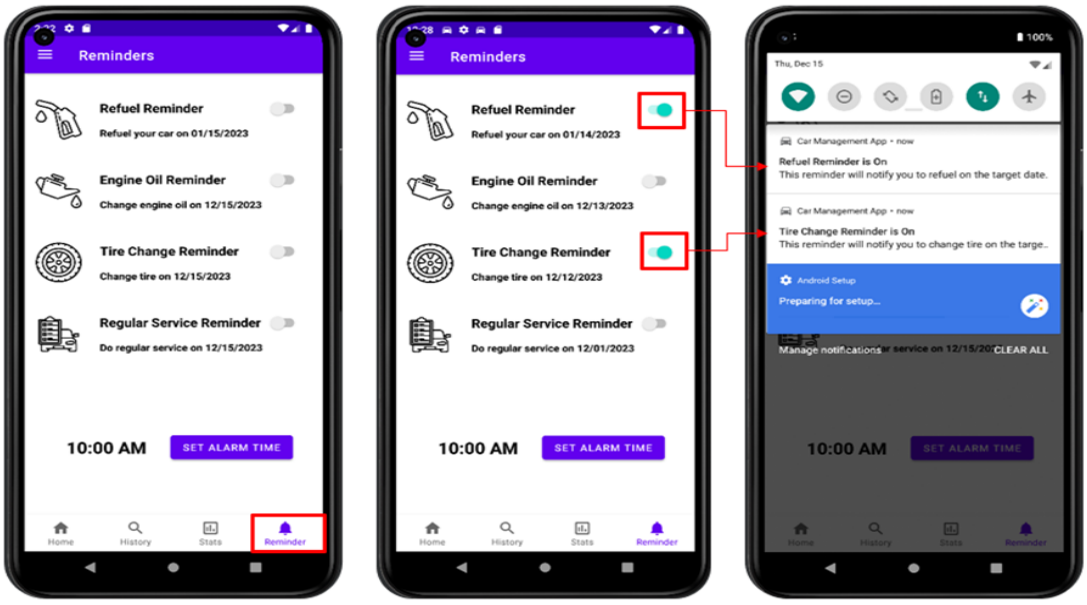


Figure 8: Reminder Menu

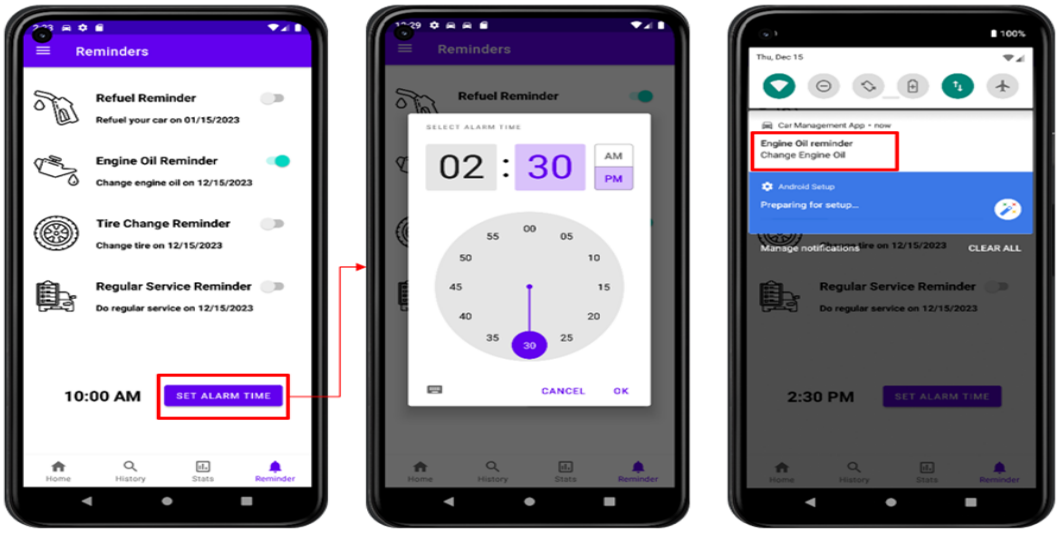


Figure 9: Setting a custom Alarm in Reminder Menu

Issues and Resolution

Issue_1: Time consumption was more since the data was filled in manually. Also the android emulator worked slower on some instances.

Possible resolution: Using JUnit for test automation would have saved some time.

Issue_2: There were bugs found during the testing phase due to the variation and differences in Android versions.

Possible resolution: Creating test cases for respective android versions and not just a target device.

References

→ Fragment

→ AdapterView

→ View

→ ListenableWorker

→ NotificationManager

→**Firebase Authentication**

https://firebase.google.com/docs/auth

https://firebase.google.com/docs/auth/android/email-link-auth

→**Bar Chart (MPAndroid Chart v3.1.0)**

https://github.com/PhilJay/MPAndroidChart

Experiences and Thoughts

Improvements

The database consisted of information for only 50 cars. The actual app will have to deal with most commercial cars. It should also be easy for users to search for models of their own cars.

In the statistics section, it would be more useful to show graphical comparison of more parameters in the vehicle management history.(Eg. Efficiency v/s time period)

Difficulties in developing

First, it was difficult to collect data related to cars that we wanted.

As features were implemented, the necessary parts continued to be not found.

From a user's point of view, we wanted to provide more useful information such as fuel usage efficiency and car parts efficiency of the car, but it was difficult to implement due to time constraints.

In addition, to provide more appropriate services, it is necessary to build a back-end server that can manage and update car information from time to time.

Overall, working on the project was a great experience and multiple concepts were learned and discussed during the implementation of this project.

Also, we would like to thank our instructor for the course Mobile Applications(CS 443), Professor Bo Sheng, for providing us with this wonderful opportunity to learn and work together.