CSCI 4100U Project Proposal

Vehicle Management App:

- Add or remove vehicles
- Input vehicle year make, model and vin number
- Ability to track your distances and estimate your gas mileage for trips
- App will suggest next oil change based off the odometer readings
- App will store and suggest service based on service and usage history
- Update vehicle service history and kilometers and store previous history
- Additional suggestion related to Make Model, and Year, utilizing AI and LLM

Project Overview:

Our Vehicle Management app will allow users to track and manage their vehicles. The features will contain an estimated gas mileage for trips taken in their vehicle. The trip computer calculates the current mileage, and stores the vehicle's previous mileage history. Additionally the product will store and update service history regarding common service items such as brakes, oil changes, and tire related maintenance. Finally, predict maintenance requirements based on Vehicle Make Model and Year. It will also estimate the next service requirements. The design will be simplistic and easy to use, allowing users to quickly add the relevant information required to calculate their estimated mileage for their trip and their next oil change.

The app will consist of four main screens, the home screen, the vehicle screen, the vehicle data screen, and the trip/service screen. The home screen will be a hub for the app where users can choose whether they want to view the vehicle page (which houses all of the users' saved vehicles), the vehicle data page (housing all data related to a saved vehicle), or the trip screen (where users can plan a trip and view previous trips, and the service screen, where service history can be viewed and or upcoming service suggestions, for both 1 vehicle, and multiple vehicles.

The vehicle screen will remain blank initially, and allow users to either:

- Select a currently saved vehicle to track.
- Edit the information of a currently saved vehicle.
- Create a new vehicle to track.
- Delete a currently saved vehicle from the list.

Confirmations for user actions on this screen will be displayed using a snack bar to help users confirm their decision.

The vehicle data screen will be where the user enters required data on their vehicle to calculate the gas mileage for a trip. Oil change intervals shall be added to let you know if you will need an oil change. This page will be a form field or similar, and will ask the user to enter data one at a time such as vehicle name, year, make, amount of gas, and odometer reading, etc.) This data will then be saved to a single vehicle object which will appear in the vehicle screen.

The trip screen will allow users to plan a trip using one of their given saved vehicles. Once their origin and destination addresses are entered, they will get a GPS route to their desired destination, as well as a notification of their estimated gas mileage for the trip (calculated using the vehicle data they have entered). This will utilize the user device's location settings and geolocation to calculate their route and mileage.

Summary:

Our Vehicle Management Application named Vroom, aims to be a quick, convenient, point of control for individuals struggling to keep track of their vehicular maintenance. Additionally it will store much needed data related to their trips, milage, vehicle history, and service history

Group Members and Responsibilities:

Almas Alam - Vehicle Service and Vehicle odometer service, front end
Taha Rana - Back-end Services, User Authentication, Backend Classes, API's
Ahad Abdul -, front end components, UI design, styling
Justin Li - Vehicle/Vehicle edit pages (implement local storage for vehicles and pages for adding/editing data), front end UI

Features and Functional Requirements:

Functional requirements

<u>Dialogs and pickers</u>

When adding, editing, or deleting vehicles from the list, there will be several dialog boxes and pickers that will acknowledge the actions you have done (to ensure that the user knows exactly what they have done in the app).

<u>Autofilling Vehicle information</u>

There should be an existing list of vehicles that should autofill as user types to make adding vehicles convenient and consistent.

Multiple screens and navigation

There will be several buttons on the homepage that will lead to different parts of the app (vehicles, edit vehicles, trip/service screen). Each of these pages will be a separate screen from each other, and navigation will be made simple with the use of previous and next buttons when filling out forms, as well as X and checkmark buttons for confirming changes such as trips or oil change intervals.

Snack bars

Snack bars will appear when the user makes edits to their vehicle list, as well as when they select a certain vehicle for editing (see figure 2 below).

Notifications

Notifications will pop up when the user needs to get an oil change, tire rotation, gas filling, brake change. Additionally, navigation notifications will let you know when the optimal time to leave should be and how long it will take to get there.

Local storage (SQLite)

All vehicles and relevant data to these vehicles will be stored on the local device storage, ensuring that the app user will always have access to the vehicle data without any potential data loss, however vehicle data will be backed up to the cloud.

Cloud storage (Firestore, MongoDB, or other)

We will use a database such as mongodb for the app. This will allow us to keep track of vehicle data as well as any user profile information that is on the app. Local gas prices may also be stored on the cloud for users to know how high/low the current price for fuel is and how that will affect their overall mileage.

HTTP requests

Requests will be authorization requests for users that are signed up to be able to receive their user data from the database, among other actions that an app user would need authentication to perform.
Additional request functionality should be added to find the cost of gas at the current time using either public API or data

Data Tables

Data tables will be used heavily for handling and manipulating data in the database. Data such as vehicles and user information will be stored in data tables for use in the online database.

Optional functional requirements

Maps

Maps will be a key component in the trip/mileage calculations. The user will be able to utilize the map to plot a travel route from their current location to their desired destination. We will also show travel data with the map such as ETAs, along with the estimated mileage for the trip.

Geolocation

We plan to implement geolocation by utilizing a public API to calculate the price of gas near your current location. This can also be utilized to find any nearby gas stations, as well as compare prices between each station and filter out any redundant stations near you.

Artificial Intelligence and Smart Suggestions

Vehicles have tons of specializations when it comes to maintenance depending on their make model and year, optionally this could be incorporated by periodically plugging in vehicle information, and service history into LLMs to find common problems and service requirements for that year, and notifying the user that this might be an issue.

Non functional requirements

Color Scheme

The color scheme of the app will be simplistic with colors directly relating to the pages that they will lead to. Each section will have its own theme and template which will help be a visual aid for users when navigating through the app (see mockup for examples).

Buttons/Navigation

The amount of buttons will be minimal in our app design. With only a few screens to swap from, the buttons can be enlarged to take up more screen space and make it easier and more deliberate for the user to press. Navigation buttons will always be present as well to ensure that users can go back and forth between pages should they desire to.

Minimal content per page

Content will be spread out over multiple pages to keep the screen less cluttered for the user. For example, form fields that need to be filled out by the user will only have

at most two fields per page. Directional buttons will be implemented to go to the next page for any additional information that needs to be entered.

Help tab

A help tab will be implemented for users who are confused on how to navigate through the app. Detailed instructions will be provided to ensure that the user is able to have a stress free experience using the app.

Social space

A space for the user to interact with other users may be implemented. This will give a chance for all users to share, discuss, and chat with each other, giving tips and praise for each other and their vehicles. The social space could be expanded to include a market place, this allows for a uniform view of vehicle history allowing users to view a listings service history, as well additionally collision history.

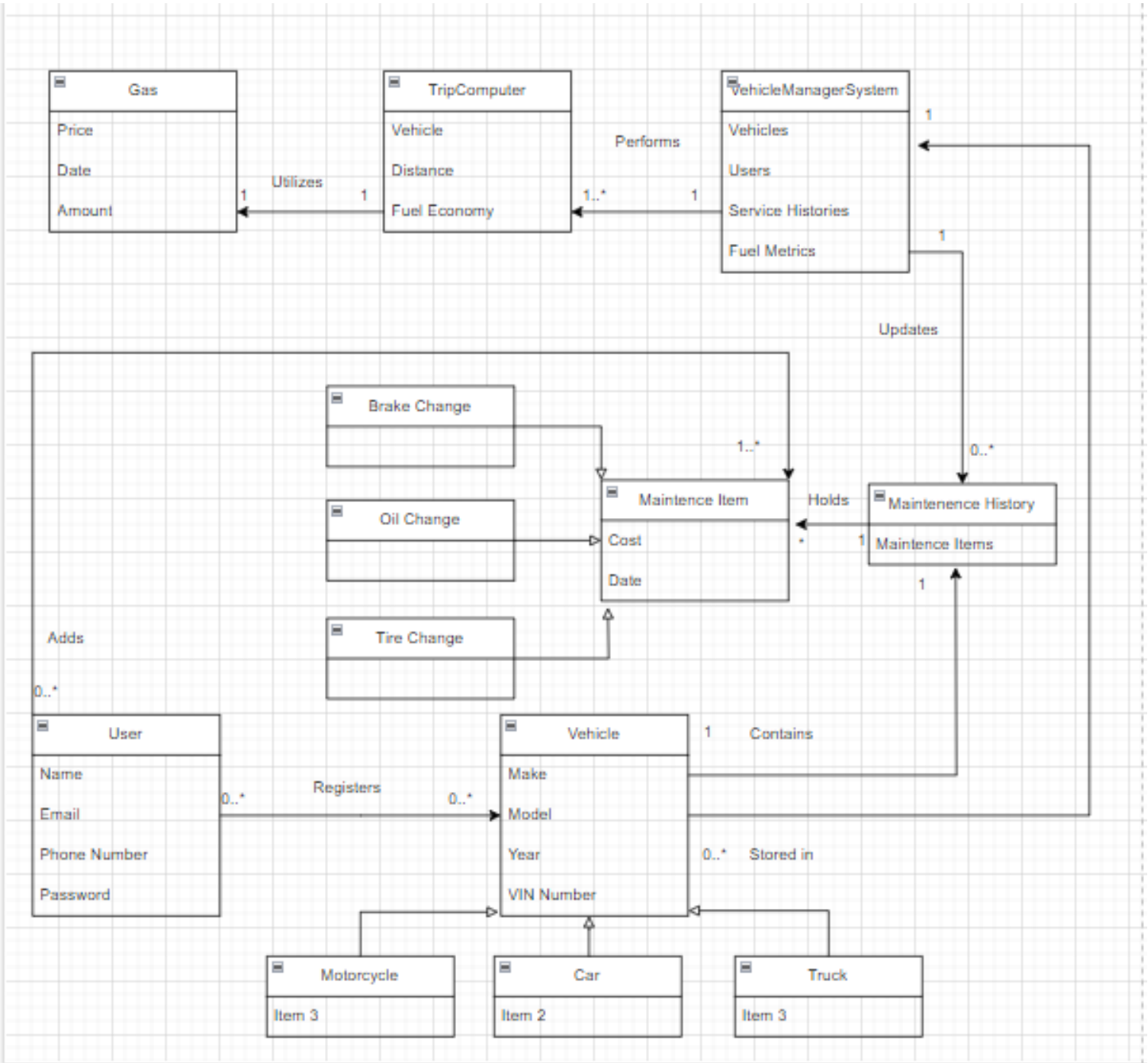
Trends graph

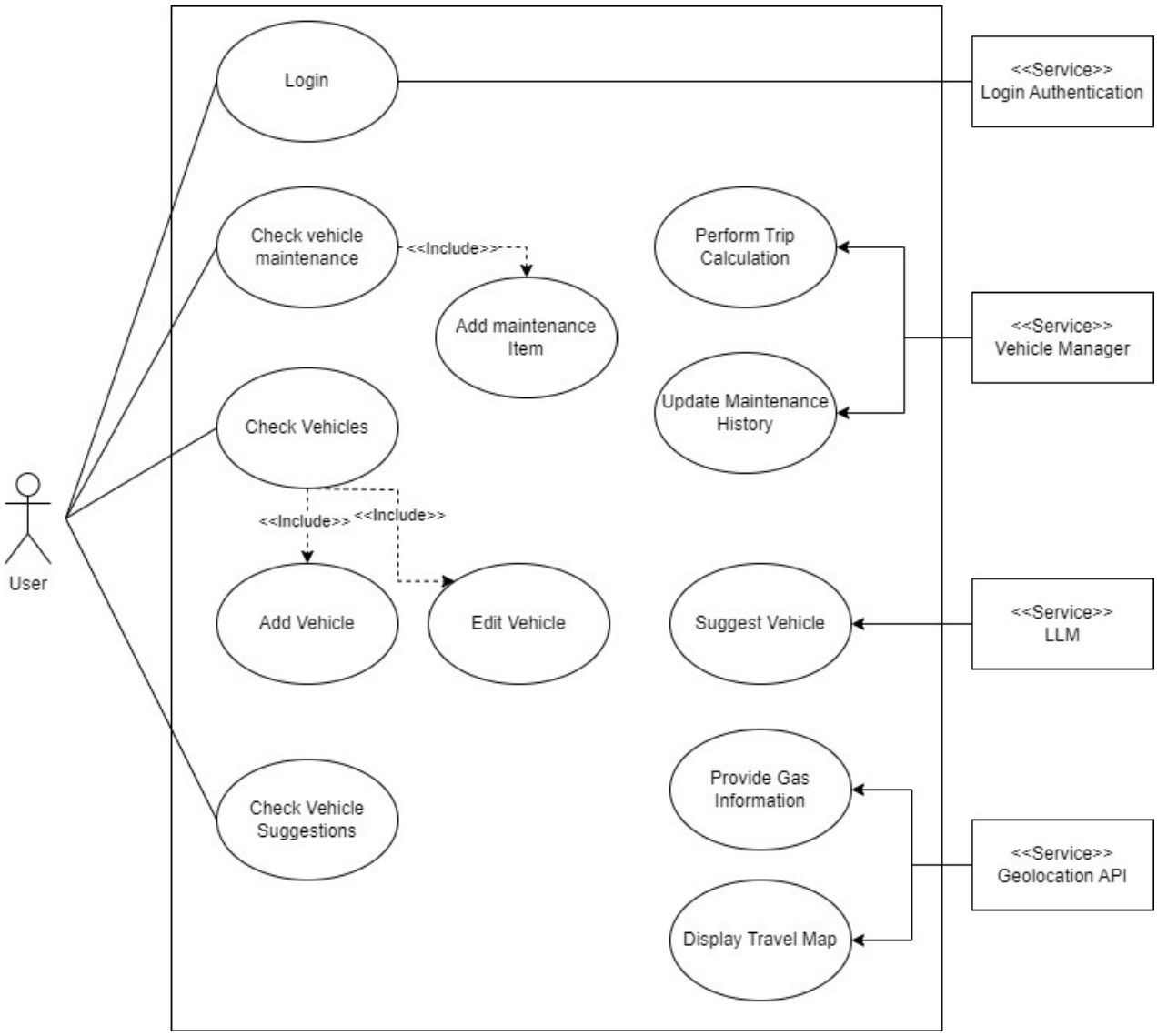
Track your fuel economy, mileage, and history over time. Showcase trends to users pertaining to increases and decreases in driving metrics.

Code Design:

Generalized Domain Model Diagram:

This generalized domain model diagram contains our classes for our app. There is a user class that registers vehicles into a VehicleManagerSystem Class, that acts as the generalized system for our application. This same vehicle management system class is responsible for the TripComputer class which accesses Current Gas Price to perform its calculation. The Vehicle Manager System also updates Maintenance History which contains Maintenance items with child classes for 3 maintenance items we intend to implement, Brakes, Oil, and Tires. Maintenance history However is stored within the Vehicle class. Vehicle has 3 additional child classes for 3 valid types we plan to implement, Truck, Car, and Motorcycle.


Gas
Price
Date
Amount
Utilizes
1
1
TripComputer
Vehicle
Distance
Fuel Economy
Performs
1..*
1
VehicleManagerSystem
Vehicles
Users
Service Histories
Fuel Metrics
1
1
Updates
0..*
Brake Change
Oil Change
Tire Change
1..*
Maintence Item
Cost
Date
Holds
*
1
Maintenence History
Maintence Items
1
Adds
0..*
User
Name
Email
Phone Number
Password
0..*
Registers
0..*
Vehicle
Make
Model
Year
VIN Number
1
Contains
0..*
Stored in
Motorcycle
Item 3
Car
Item 2
Truck
Item 3



The above Use Case diagram depicts the main actions a user might make while using
the application. Before being able to use the application, the user will have to input their login information which will then be authenticated by a third party service. Once logged into the application, the user will be able to view their registered vehicles, vehicle maintenance information, and vehicle suggestions. The user will also be able to add/edit registered vehicles and add/edit maintenance information. On the back-end the application will have a vehicle manager that will update the maintenance history as needed as well as calculated fuel mileage for trips. The application will also utilize a geolocation API to render a viewable map for the user as well as provide information pertaining to nearby gas stations and their current specific prices. Finally, the application will use an LLM and AI to notify the user about any issues or suggestions about their vehicles.

UI Mockup:

The following are prototype UI designs that outline the functionality of our application. The various figures show a visual representation of what the app will look like on the device, as well as any features that the user may access at a given page. Both functional and non-functional requirements are also shown in the following mockups, outlining how and when they are used.

Figure 1: Main Menu

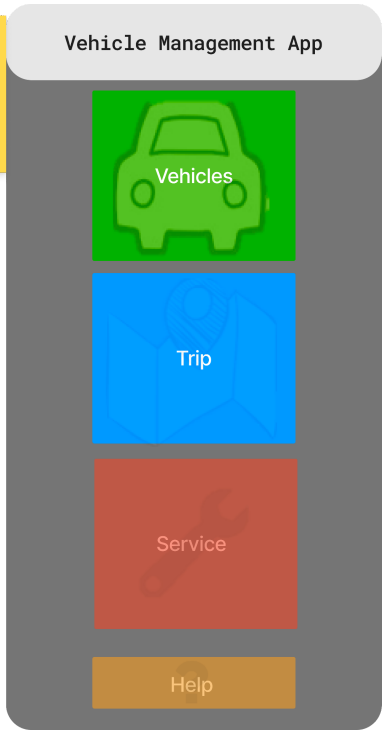
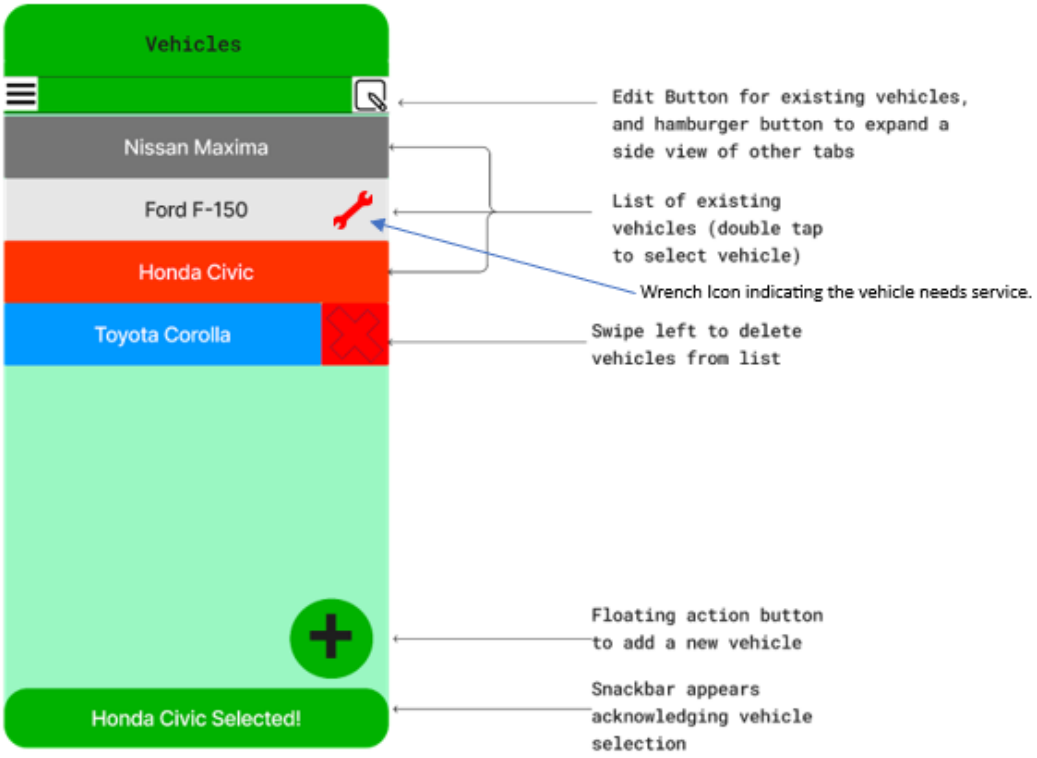


Figure 2: Vehicle page



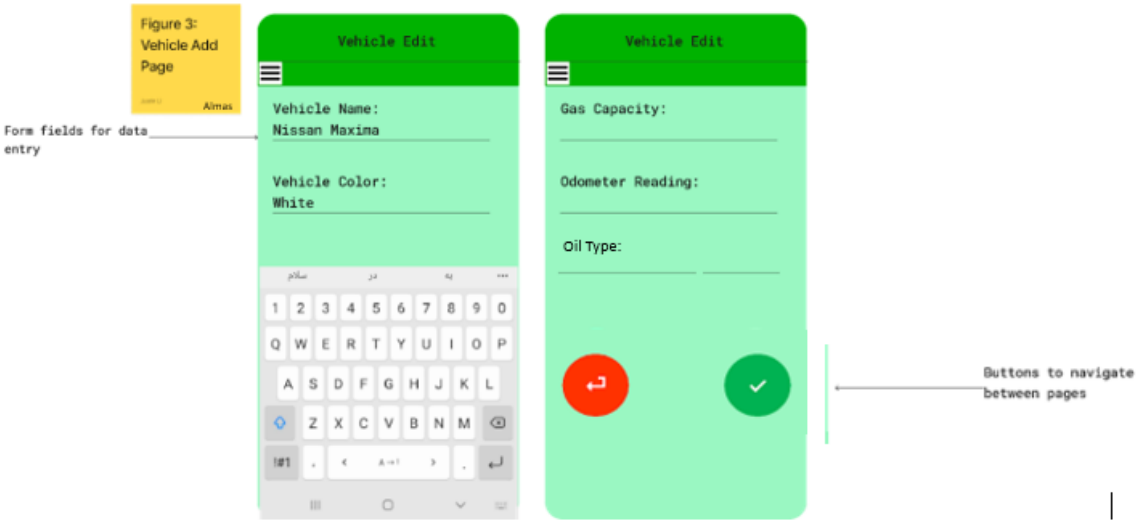


Figure 3:
Vehicle Add
Page
Almas
Form fields for data entry
Vehicle Edit
Vehicle Name:
Nissan Maxima
Vehicle Color:
White
Vehicle Edit
Gas Capacity:
Odometer Reading:
Oil Type:
Buttons to navigate between pages

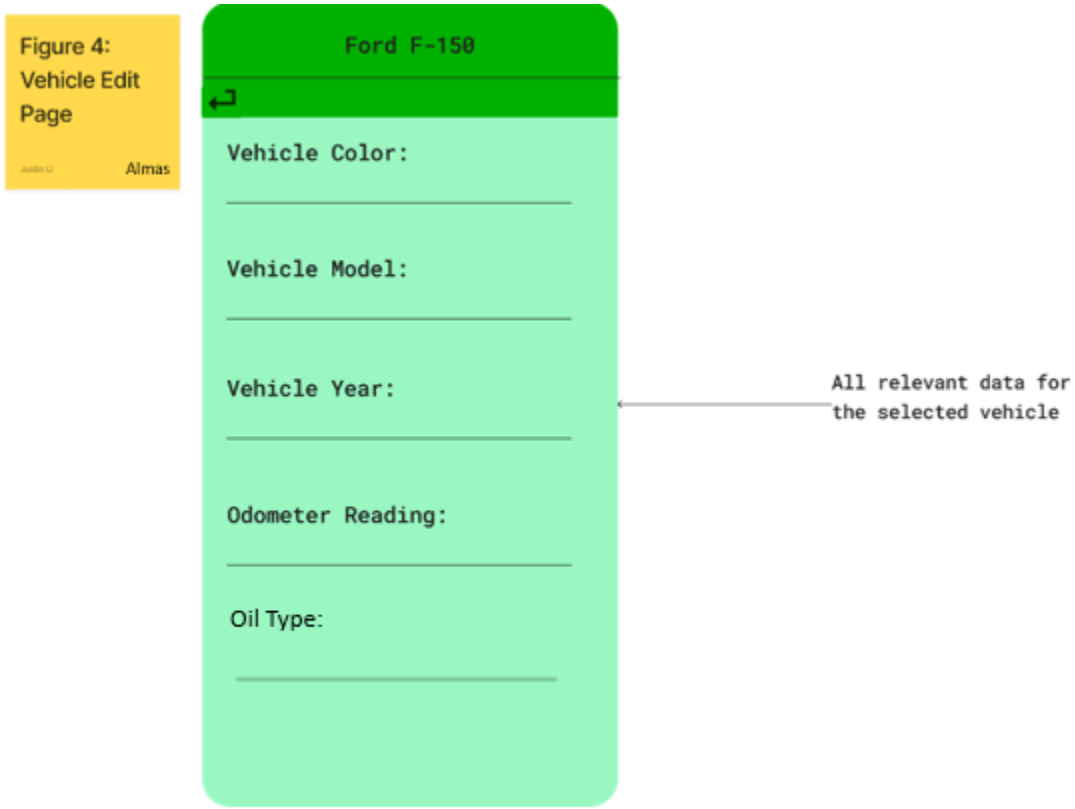
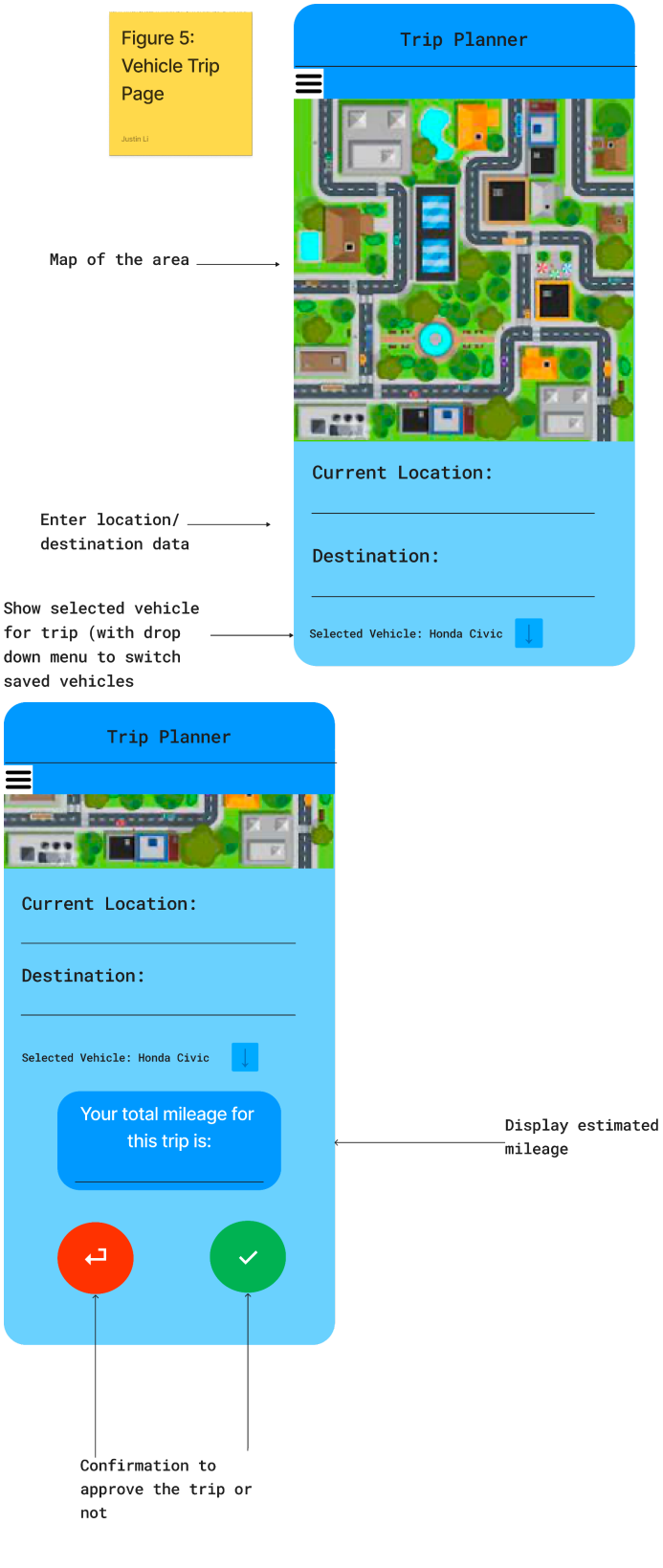


Figure 4:
Vehicle Edit
Page
Almas
Ford F-150
Vehicle Color:
Vehicle Model:
Vehicle Year:
Odometer Reading:
Oil Type:
All relevant data for the selected vehicle

Figure 5: Vehicle Trip Page

Justin Li



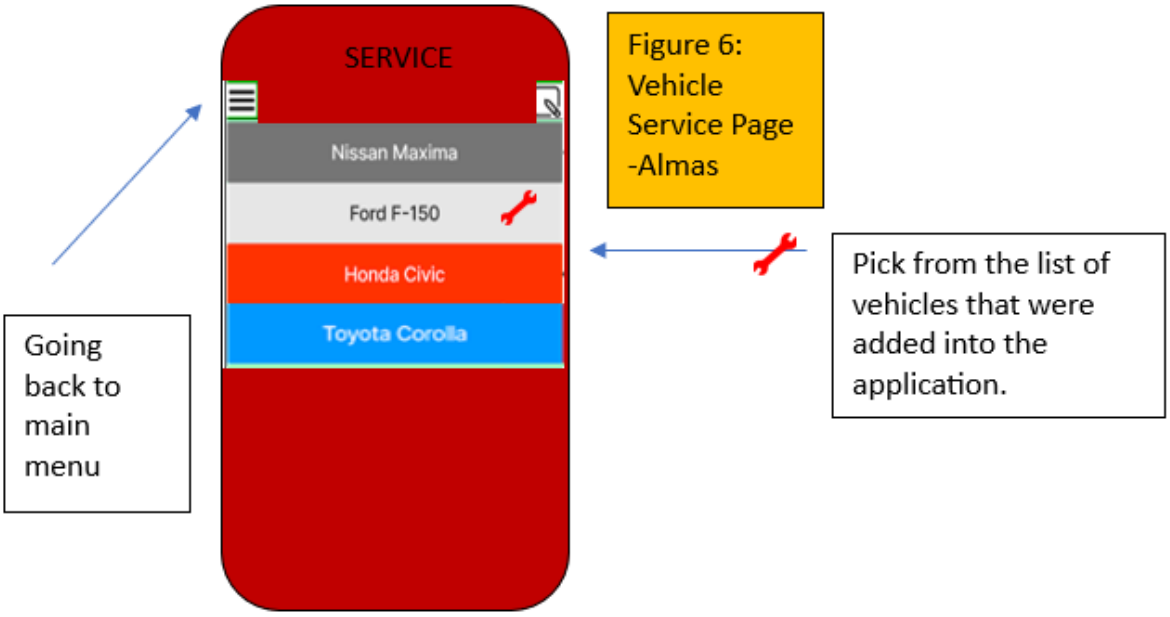


Figure 6: Vehicle Service Page -Almas

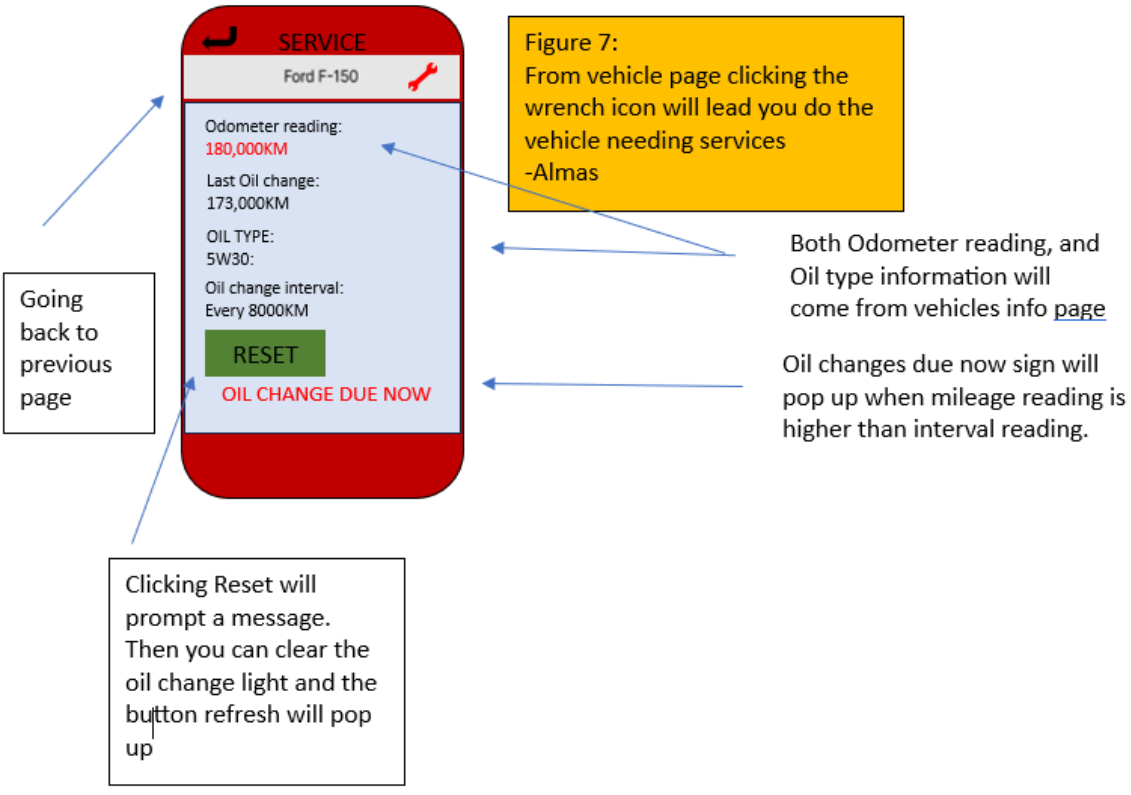


Figure 7:
From vehicle page clicking the wrench icon will lead you do the vehicle needing services
-Almas

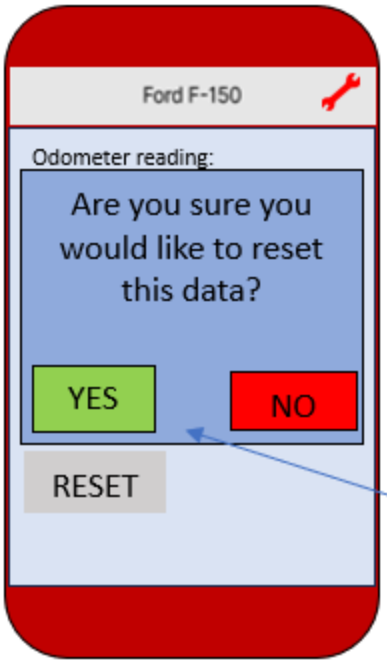


Figure 8:
The reset light asks a prompt so that it can reset and arrange the proper data.
-Almas

Clicking yes, will allow the page to refresh

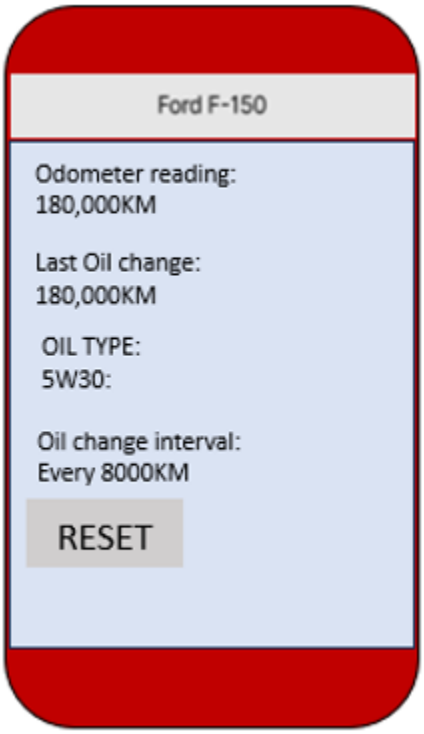


Figure 9:
The reset light stays greyed out until the next oil change interval in 8000km. The data has been refreshed and the service light is now removed
-Almas