
By: Rohit Manikuttan
Student ID: 2386886

MOBILE AND UBIQUITOUS COMPUTING

TABLE OF CONTENTS

- 01.** Abstract
- 02.** Application domain and Problem
- 03.** Context Widget
- 04.** Smart phone app
- 05.** w3 Thing Description
- 06.** Learning and Intelligence
- 07.** Managing Human-Computer Interaction
- 08.** Managing Security and Privacy
- 09.** Approaches to Evaluation

In our world of nowadays, users have faced problems of time wasting and stress of locating their parked cars in crowded or unknown areas, which is troubling and distressing and there has also been worry over the possibility of vehicle theft or misuse that ultimately leaves anxiety and security challenges for users. To alleviate this predicament, I have tried integrating the Internet of Things (IoT) solution of Location Beacons for Vehicle Tracking so that users can be able to get their car's location remotely, get immediate alerts, and optimize fleet management which enhances safety and efficiency.

At the core of this solution is the powerful Location Beacon, which seamlessly works with the NavAuto app to provide simple and secure automotive location tracking. By understanding the needs and priorities of stakeholders including drivers, fleet managers and law enforcement officials this solution focuses on user design and functionality. The inclusion of an LED indicator and audible alerts, in the beacon ensures that users are well informed about status conditions like vehicle lock/unlock or alarm triggers. Moreover the GPS and accelerometer sensors in the beacon give users vehicle positioning and dynamic motion detection capabilities enabling features such as collision detection and towing alerts. The compatibility with OBD II devices not only improves functions but also provides valuable insights, into vehicle performance and operations. In scenarios where GPS coverage may be limited, seamless integration with Bluetooth trackers ensures proximity-based locating within designated ranges, enhancing overall reliability and accessibility. Robust encryption mechanisms shield the transmission of data from and to the application, ensuring the privacy and safety of the end users. Additionally, in case of an emergency, the NavAuto app has an SOS button that links to the relevant authorities or support contact. Such signals show the extent to which the solution design aims at ensuring the users' well-being. Also, the solution is voice controlled, allowing for eyes-free use. By collecting data from the GPS, the accelerometers and the OBD-II device, the end user solution enables predictive analysis. As a result, a dashboard can track the car's performance and safety by predicting factors such as repair demand and driving performance.

Due to the fact that the NavAuto platform is suitable for a variety of cars and contains individually configured vehicle profiles, it provides the customer with an opportunity to get personalized functionality and the most rational recommendations. Overall, this game-changing approach provides customers with unprecedented convenience and confidence in multiple transportation options; furthermore, by not only resolving the existing issues but also establishing a new standard for the most comprehensive and user-generated vehicle localization technology.



APPLICATION DOMAIN & PROBLEM

Application Domain: The application field is all about following cars and ensuring their safety in towns where people find it difficult to locate their vehicles in crowded places, worry about losing them or using them without permission and require effective management of the group of vehicles.

Problems being Addressed:

1. **Difficulty in Vehicle Location:** Car owners occasionally can't find the spaces in which they have parked their vehicles, particularly in crowded or strange areas, that is, the expenses to be intentionally wasted and their stress levels are growing.
2. **Security Concerns:** Issues of unauthorised use or car robbery worsen the problem that lead to anxiety, psychological effects and ensure the safety of the driver or car is compromised.
3. **Inefficient Fleet Management:** Fleet managers find fleet operations optimization to be a real challenge, as well as ensuring the security of vehicles. These inefficiencies bring about operational inefficiencies and a matter of increased risks.

Why it requires solving: Overcoming these challenges will be essential if we aim to improve the overall experience when the vehicle is being owned, to promote the safety and security of vehicle owners and fleet managers then, and finally to facilitate fleet management operations. Not doing away with warehouse management issues poses a risk of theft, loss of valuable time and inefficient running of operations.

Who Benefits from This Solution:

1. **Vehicle Owners:** People who own vehicles have enhanced convenience, increased security, and a sense of general well-being of knowing that they can trace down their vehicles through the app and will receive notifications in case of unauthorized access or theft.
2. **Fleet Managers:** Managers operating over fleets of vehicles enjoy reduced operational costs with streamlined fleet management procedures, tightened security mechanisms and advanced tracking up to real time.
3. **Law Enforcement Officials:** On the other hand, authorities responsible for solving vehicle theft cases are provided with better location identification irrespectively of the time of day, which lead to faster response time and better law enforcement practices.

Current Approaches to This Problem:

“OnStar” was Developed by General Motors (GM) in the United States, OnStar is a subscription-based service that provides vehicle tracking, navigation, roadside assistance, and remote diagnostics which was launched in 1996.

“Tracker Network” is a vehicle tracking and recovery service based in the United Kingdom which was established in the 1990s. It uses GPS and VHF technology to track stolen vehicles and assist law enforcement in recovering them.



Why My Approach Could be Superior:

My approach, leveraging Location Beacons for Vehicle Tracking and integrating IoT technology with advanced sensors and encryption protocols, offers several advantages over traditional approaches:

1. **Real-Time Monitoring:** My solution is a real-time system for vehicles with location tracking and monitoring features, which can send alarms immediately in the event of theft or other security breaches.
2. **Enhanced Accuracy:** I accomplished this task by enabling integration of GPS and accelerometer sensors. Now my device can provide accurate vehicle positioning and motion detection which ensures dynamic tracking even in urban areas which deal with the problem of limited GPS signal coverage.
3. **User-Centric Design:** My idea is focused on addressing what end users want and require, including creating an option of user profiles and supporting voice commands which is considered a great improvement in usability, therefore boosting user experience.
4. **Comprehensive Security Measures:** The secure data transmission through the strong process of encryption protocols and the tamper-resistant elements are provided, which also secure the privacy and security of user data in IoT based applications and help in the growing security concern in IoT.
5. **Streamlined Fleet Management:** Fleets manager who is responsible for the reforming, monitoring, and ensuring the safety of fleet processes will typically witness all this processes get improved to one that is more efficient and has fewer risks from the monitoring capabilities of the system.

CONTEXT WIDGETS

01. Vehicle Location Widget:

The Vehicle Location widget shows live tracing of the vehicle's estimate location. Indicates the vehicle location on a map screen, letting users like to see where the vehicle is in real time. This also enables the mapping base layer to have an interactive feature such as zoom in/out and that the map is panning for improved navigation. The widget would use GPS technology to check the location and condition, and automatically updating real-time as the vehicle moves, allowing users to have a flow of live updates on the vehicle's location.

Widget Class		Vehicle Location Widget
Attributes		
Vehicle ID		Unique identifier for the vehicle
Latitude		Current geographic latitude
Longitude		Current geographic longitude
Last Updated		Timestamp of the most recent location update.
Vehicle Status		Indicates current state (parked, moving)
Battery Level		Battery level of the location beacon
Map View		Type of map displayed
Zoom Level		Current zoom level of the map
Callbacks		
- LocationUpdate	Receives updated location data (latitude, longitude, etc.)	Triggers UI update during any movement.
- BatteryLevelLow		Triggered when the battery level of the location beacon falls below a certain threshold.
-VehicleStatusChange		Triggered when the vehicle status changes (e.g., from parked to moving).

02. Emergency Assistance Widget:

A built-in Emergency Assistance Widget offers users an instant way to instantaneously connect to emergency services or roadside assistance. Users can easily call for help or inform about an emergency status by the one-touch shortcut activation and it helps to cut response time in a crisis. Along with sending distress signals it also provides an option of sharing your current GPS position with emergency services so that the official can receive help in the fastest possible time. The widget nicely fits with the application interface inside the mobile application; therefore, all the main dashboard ramps will be short cut.

Widget Class		Emergency Assistance Widget
Attributes		
Emergency Contact List		Pre-configured phone number for emergencies.
Geolocation Accuracy		Desired accuracy level for location sharing
Callbacks		
- onSOSButtonPressed		Triggered when SOS button is pressed

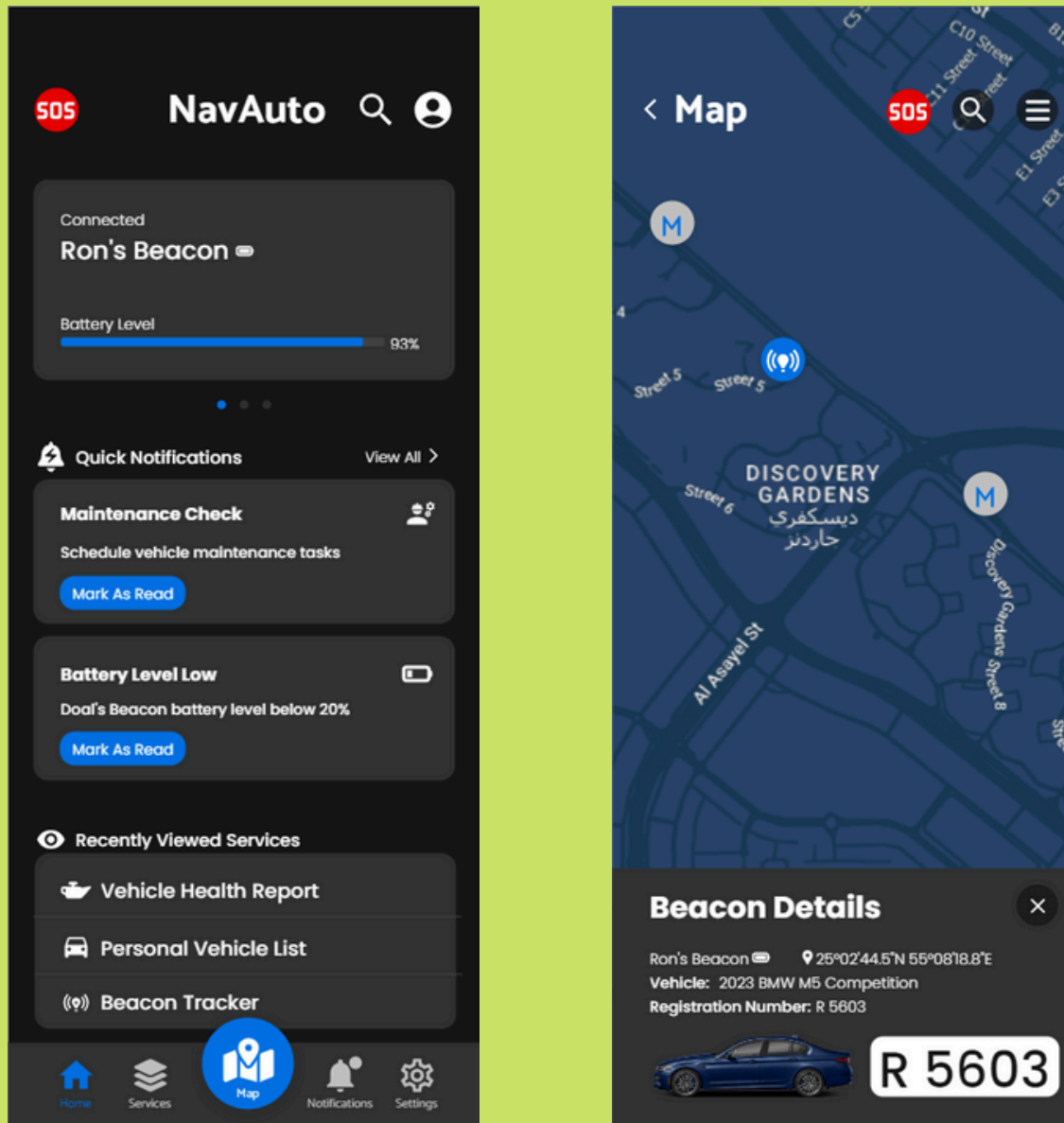
03. Vehicle Profiles Widget:

The "Vehicle Profile Widget" empowers users to manage vehicles profiles within the application so they can carry out swiftness to change to a different vehicle of their own. Users are able to create and personalize a vehicle profile for each vehicle they own or frequently use. These profiles are very helpful when it comes to storing the data from different types of vehicles such as cars, motorcycles, trucks, and scooters. Every vehicle can have their profiled vehicle include the vehicle brand, model, year, license plate number and the profiled vehicle identifiers. The action of the widget is to facilitate interaction by providing an intuitive interface for users to switch to vehicle profiles flawlessly, thus ensuring that the system keeps the settings as per the user's current vehicle as well as customized preferences.

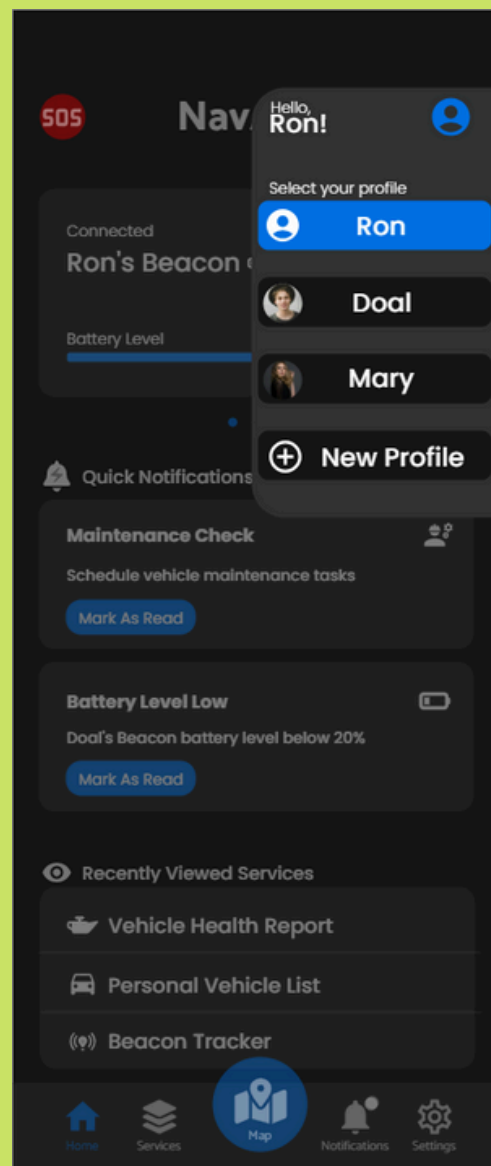
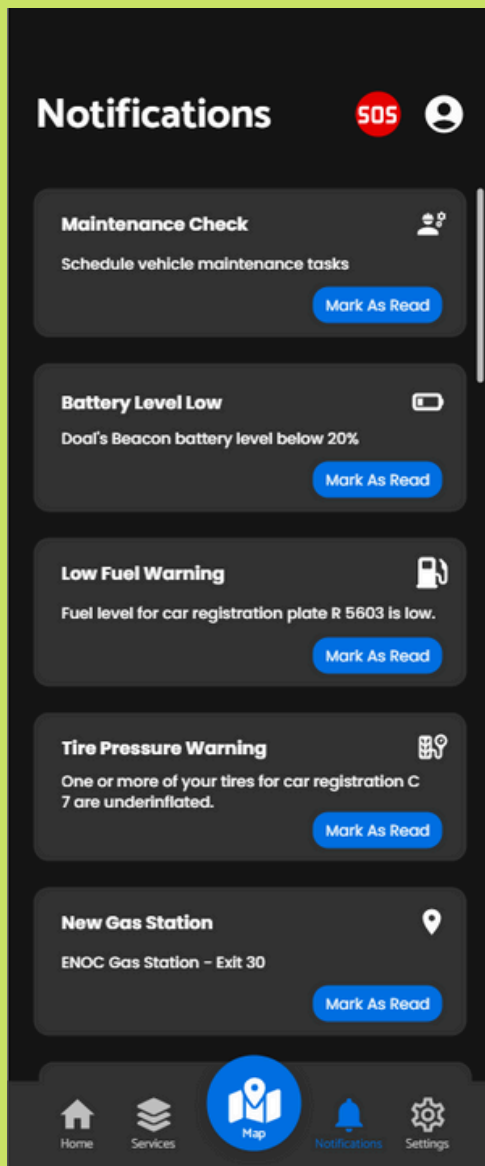
Widget Class	Vehicle Profiles Widget
Attributes	
Current Profile	Selected Vehicle profile
Vehicle List	List of all vehicle profiles
Vehicle Name	User-configured name for the vehicle
Last Updated	Timestamp of the latest profile data update
Callbacks	
- onVehicleListUpdate	Triggers UI update to display available profiles
- Profile Selection	Triggers UI changes and updates chosen profile.

SMART PHONE APP

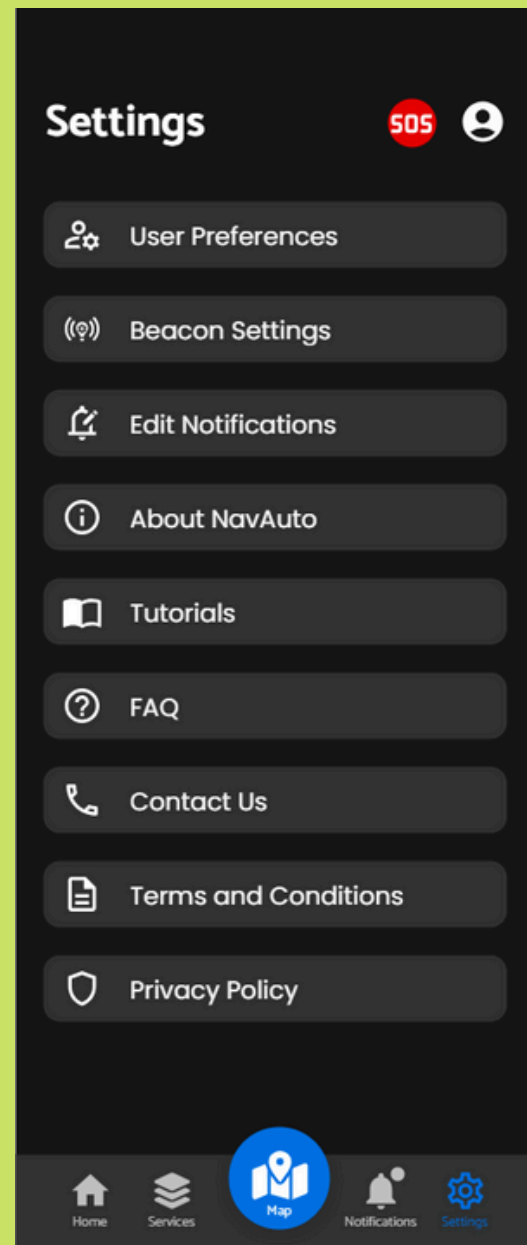
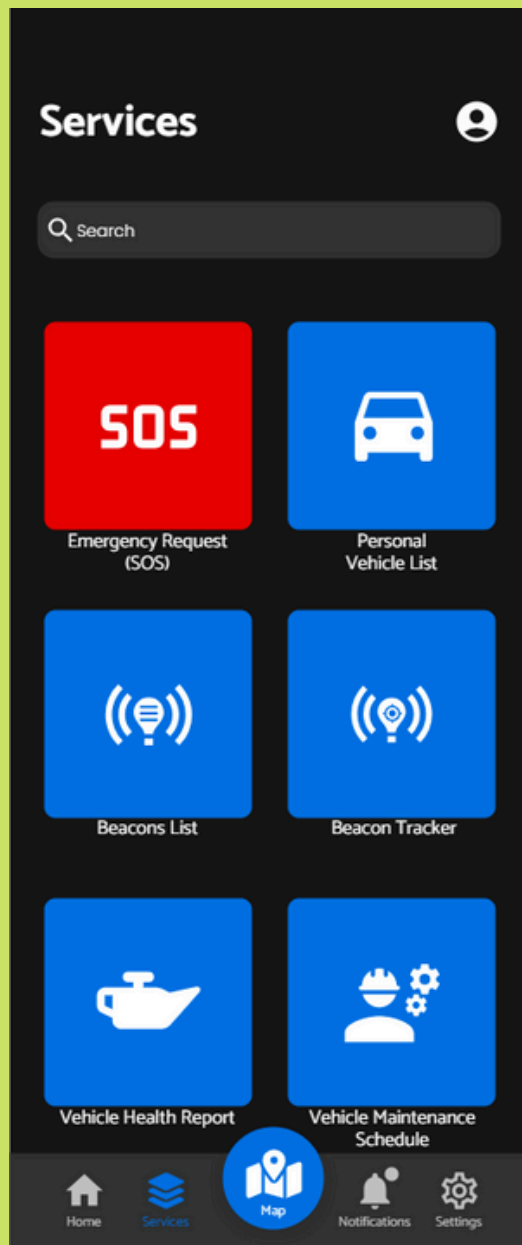
Using Uizard I've built the following interface as my Hi-Fi prototype. Going forward in the future, the application will have support for both light and dark mode as well as any extra additional feature's which would seem fit.



Dashboard page is a home base inside the NavAuto app, which provides an user with a full set of information regarding their trips in their cars. It offers on demand feedback about actual car's location, status, and alerts to the users. This keeps them updated and with feeling in the dark. With its simple and customizable layout, the dashboard will provide users, a rich and different experience as they will be able to customize everything according to their taste and preferences. Beyond just displaying the data, the dashboard becomes an active tool that provides useful information such as fuel consumption and maintenance needs, and many other aspects based on user inputs. Additionally to that, the dashboard comes with the multi-staff features such as notification and Socouro, to improve the vehicle management system and overall performance. In sum, the dashboard page is like a significant piece of the puzzle for the users to Hassle-freely oversee, take the reins, and finetune their vehicles' operations.



The NavAuto application provides a notifications page where users will get real time alerts and regular updates on then status of their automobiles as a centralized hub. It keeps users updated with notifications about car alarms, scheduled maintenance, safety announcements and other immediate issues- all to ensure that the user is informed and reactive. With the ability to choose notifications settings the users can customize the settings to the fit such that they enhance their overall driving experience. However, the car profiles page is where the users of the app can easily organize, edit, and personalize their vehicle profiles. With users provided with the opportunity to create and customize their profile for different vehicles including their particular make, model and preferences. With that, these pages place users in the position of having every tool at their disposal to stay connected, informed and a part of controlling their auto management experiences.



The users experience has been streamlined by NavAuto through the integration of various services like maintenance scheduling, SOS and vehicle diagnostics streamlining their vehicle maintenance needs. Additionally, the settings page serves as the focal point for the app, where users can tailor their app's preferences, set their privacy options, and choose the notification preferences for a personalized interface.

Widget TREE

```

1 class NavAutoHomePage extends StatefulWidget {
2   @override
3   _NavAutoHomePageState createState() => _NavAutoHomePageState();
4 }
5
6 class _NavAutoHomePageState extends State<NavAutoHomePage> {
7   @override
8   Widget build(BuildContext context) {
9     return Scaffold(
10       appBar: AppBar(
11         title: Text('NavAuto'),
12       ),
13       body: Padding(
14         padding: const EdgeInsets.all(16.0),
15         child: Column(
16           children: [
17             Row(
18               children: [
19                 Container(
20                   color: Colors.grey[200],
21                   padding: EdgeInsets.all(16.0),
22                   child: Text('Connected'),
23                 ),
24                 Spacer(),
25                 Container(
26                   color: Colors.grey[200],
27                   padding: EdgeInsets.all(16.0),
28                   child: Text('My Beacon'),
29                 ),
30               ],
31             ),
32             Row(
33               children: [

```

```

class NavAutoHomePage extends StatefulWidget {
    class _NavAutoHomePageState extends State<NavAutoHomePage> {
        Widget build(BuildContext context) {
            return Scaffold(
                body: Padding(
                    child: Column(
                        Expanded(
                            child: Container(
                                width: double.Infinity,
                                color: Colors.grey[200],
                                padding: EdgeInsets.all(16.0),
                                child: Column(
                                    children: [
                                        Row(
                                            children: [
                                                Text('Battery Level'),
                                                Spacer(),
                                                Text('93%'),
                                            ],
                                        ),
                                        Row(
                                            children: [
                                                Text('Quick Notifications'),
                                                TextButton(
                                                    onPressed: () {
                                                        // Handle "View All >" button press
                                                    },
                                                    child: Text(
                                                        'View All >',
                                                        style: TextStyle(color: Colors.blue),
                                                    ),
                                                ),
                                            ],
                                        ),
                                    ],
                                ),
                            ),
                        ),
                    ),
                ),
            );
        }
    }
}

```



```
Untitled-1.ipynb • class NavAutoHomePage extends StatefulWidget •
6 class _NavAutoHomePageState extends State<NavAutoHomePage> {
8   Widget build(BuildContext context) {
9     return Scaffold(
13       body: Padding(
15         child: Column(
146           children: [
147             Expanded(
148               child: Container(
149                 alignment: Alignment.center,
150                 padding: EdgeInsets.all(16.0),
151                 child: Text(
152                   'My Beacon Tracker',
153                   style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.bold),
154                 ),
155               ),
156             ),
157           ],
158         ),
159         Row(
160           children: [
161             Expanded(
162               child: Container(
163                 padding: EdgeInsets.all(8.0),
164                 child: Row(
165                   mainAxisAlignment: MainAxisAlignment.spaceEvenly,
166                   children: [
167                     Icon(Icons.home),
168                     Text('Services'),
169                   ],
170                 ),
171             ),
172           ],
173         ),
174       ),
175     );
176   }
177 }
```

```
Untitled-1.ipynb • class NavAutoHomePage extends StatefulWidget •
6 class _NavAutoHomePageState extends State<NavAutoHomePage> {
8   Widget build(BuildContext context) {
9     return Scaffold(
13       body: Padding(
15         child: Column(
163           padding: EdgeInsets.all(8.0),
164           child: Row(
165             mainAxisAlignment: MainAxisAlignment.spaceEvenly,
166             children: [
167               Icon(Icons.home),
168               Text('Services'),
169             ],
170           ),
171         ),
172       ),
173     );
174   }
175 }
```

It is shown here the widget tree code expressing the structure of the new homepage interface for our flutter application. It begins with an AppBar having the title "NavAuto." It has multiple rows with each row comprising of containers displaying various sections – battery status, notifications, maintenance checks and so on. Within these units, the text widgets show the corresponding information, and the button offers for interaction elements like marked tasks as read. Two at the bottom are icons displaying navigation options for about talking to the support and settings. Altogether, the widget tree builds up the homepage in a straight-forward way, which results in the users finding it easy to navigate and also to interact with the NavAuto application.

Flutter Code : Dashboard Page

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: CarDashboard(),
    );
  }
}

class CarDashboard extends StatelessWidget {
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('NavAuto'),
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            CarInfoSection(),
            QuickNotificationsSection(),
            BeaconTrackerSection(),
            ServicesSection(),
            SettingsSection(),
          ],
        ),
      );
    }
  }

class CarInfoSection extends StatelessWidget {
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(16.0),
      child: Row(
        children: [
          Text('Battery Level:'),
          Spacer(),
          Text('93%'),
        ],
      ),
    );
  }
}

class QuickNotificationsSection extends StatelessWidget {
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(16.0),
      child: Row(
        children: [
          Icon(Icons.notifications),
          SizedBox(width: 8.0),
          Text('Maintenance Check'),
        ],
      ),
    );
  }
}

class BeaconTrackerSection extends StatelessWidget {
  Widget build(BuildContext context) {
    return Container(
      height: 200.0,
      color: Colors.grey[200],
      child: Center(child: Text('Map Placeholder')),
    );
  }
}

class ServicesSection extends StatelessWidget {
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(16.0),
      child: Row(
        children: [
          Icon(Icons.build),
          SizedBox(width: 8.0),
          Text('Schedule Service'),
        ],
      ),
    );
  }
}

class SettingsSection extends StatelessWidget {
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(16.0),
      child: Row(
        children: [
          Icon(Icons.settings),
          SizedBox(width: 8.0),
          Text('App Settings'),
        ],
      ),
    );
  }
}
```

W3 THING DESCRIPTION

01

Location Beacon

```
{
  "name": "Location Beacon",
  "type": "thing",
  "description": "A device attached to a vehicle that transmits its location data",
  "properties": {
    "batteryLevel": {
      "type": "number",
      "description": "Remaining battery power of the beacon (percentage)"
    },
    "status": {
      "type": "string",
      "description": "Current status of the beacon (e.g., active, inactive)"
    }
  },
  "actions": {
    "sendData": {
      "description": "Sends location data to the NavAuto app",
      "type": "void",
      "parameters": {
        "latitude": {
          "type": "number",
          "description": "Geographic latitude of the vehicle"
        },
        "longitude": {
          "type": "number",
          "description": "Geographic longitude of the vehicle"
        }
      }
    }
  },
  "events": {
    "lowBattery": {
      "description": "Battery level falls below a predefined threshold",
      "data": {
        "batteryLevel": {
          "type": "number",
          "description": "Current battery level of the beacon"
        }
      }
    }
  }
}
```

This gadget sends the vehicle's position and monitors its own status. It monitors the battery level (%) and current state (active/inactive). It delivers location information (latitude and longitude) to the NavAuto application. When power decreases below a certain level, it causes a "lowBattery" event to occur.

Affordances:

- Transmission: Because of NavAuto app location can transmit latitude and longitude data by location beacon.
- Status indication: The beacon also states the status of the beacon on property (active/inactive).
- Alert generation: When the battery refuses to only visit this threshold, e.g., when the battery level falls below a certain value, the beacon will notify the notification system.

```
{
  "name": "SOS Button",
  "type": "thing",
  "description": "A button on the NavAuto app for emergency situations",
  "properties": {
    "enabled": {
      "type": "boolean",
      "description": "Indicates if the SOS button is enabled"
    }
  },
  "actions": {
    "triggerSOS": {
      "description": "Initiates an emergency SOS call",
      "type": "void"
    }
  },
  "events": {
    "SOS triggered": {
      "description": "SOS button is pressed",
      "data": {
        "location": {
          "type": "object",
          "description": "User's current location data"
        }
      }
    }
  }
}
```

This button in the NavAuto app initiates emergency calls and shares location. It can be enabled or disabled by the user. It triggers an emergency SOS call upon pressing. Pressing the button fires an "SOS triggered" event with user location data.

Affordances:

- Emergency call initiation: The SOS button can trigger an emergency SOS call to predefined contacts or emergency services.
- Location sharing: During an SOS event, the button can share the user's current location data with emergency personnel.
- Activation control: Users can enable or disable the SOS button based on their needs.


```

{
  "name": "Vehicle Dashboard",
  "type": "thing",
  "description": "A user interface component displaying vehicle information",
  "properties": {
    "selectedVehicle": {
      "type": "string",
      "description": "ID of the currently selected vehicle profile"
    },
    "batteryLevel": {
      "type": "number",
      "description": "Battery level of the selected vehicle"
    },
    "odometer": {
      "type": "number",
      "description": "Total distance traveled by the selected vehicle"
    },
    "fuelLevel": {
      "type": "number",
      "description": "Current fuel level of the selected vehicle (optional)"
    },
    "status": {
      "type": "string",
      "description": "Current status of the selected vehicle (e.g., parked, running)"
    }
  },
  "actions": {
    "refreshData": {
      "description": "Retrieves and updates the displayed vehicle data",
      "type": "void"
    },
    "selectVehicle": {
      "description": "Selects a different vehicle profile",
      "type": "void",
      "parameters": {
        "vehicleId": {
          "type": "string",
          "description": "ID of the vehicle profile to select"
        }
      }
    }
  },
  "events": {
    "data

```

User selected vehicle details are shown by this interface widget. It not only displays battery level, odometer reading, and current status (parked/running) for the preferred vehicle. Scrolling through the menu options, the user can refresh the displayed data and choose different vehicle attributes. It might be all about updating the interface based on new records or choosing a profile.

Affordances:

- Visualization: Displays various vehicle parameters like battery level, odometer reading, fuel level, and current status.
- Selection: Allows users to choose a different vehicle profile to view its specific information.
- Data Update: Enables users to refresh the dashboard with the latest vehicle data through the "refreshData" action.
- Information Access: Provides a centralized location to access real-time and historical vehicle information.

THING FUNCTIONALITY

Thing 1: Vehicle Location Tracker

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

const int analogOutPin = 5; // Analog output pin for controlling LED brightness
int sensorValue = 0;        // Value read from the analog input pin
int outputValue = 0;        // Value output to the PWM (analog out)

TinyGPSPlus gps;            // Initialize TinyGPS++ library for GPS functionality
SoftwareSerial gpsSerial(2, 3); // RX, TX pins for GPS module

void setup() {
  Serial.begin(9600);        // Initialize serial communications at 9600 bps
  gpsSerial.begin(9600);     // Initialize GPS serial communications
  pinMode(analogOutPin, OUTPUT); // Set analog output pin for LED
}

void loop() {
  // Increment the analog input value
  sensorValue = (sensorValue + 10) % 1024;

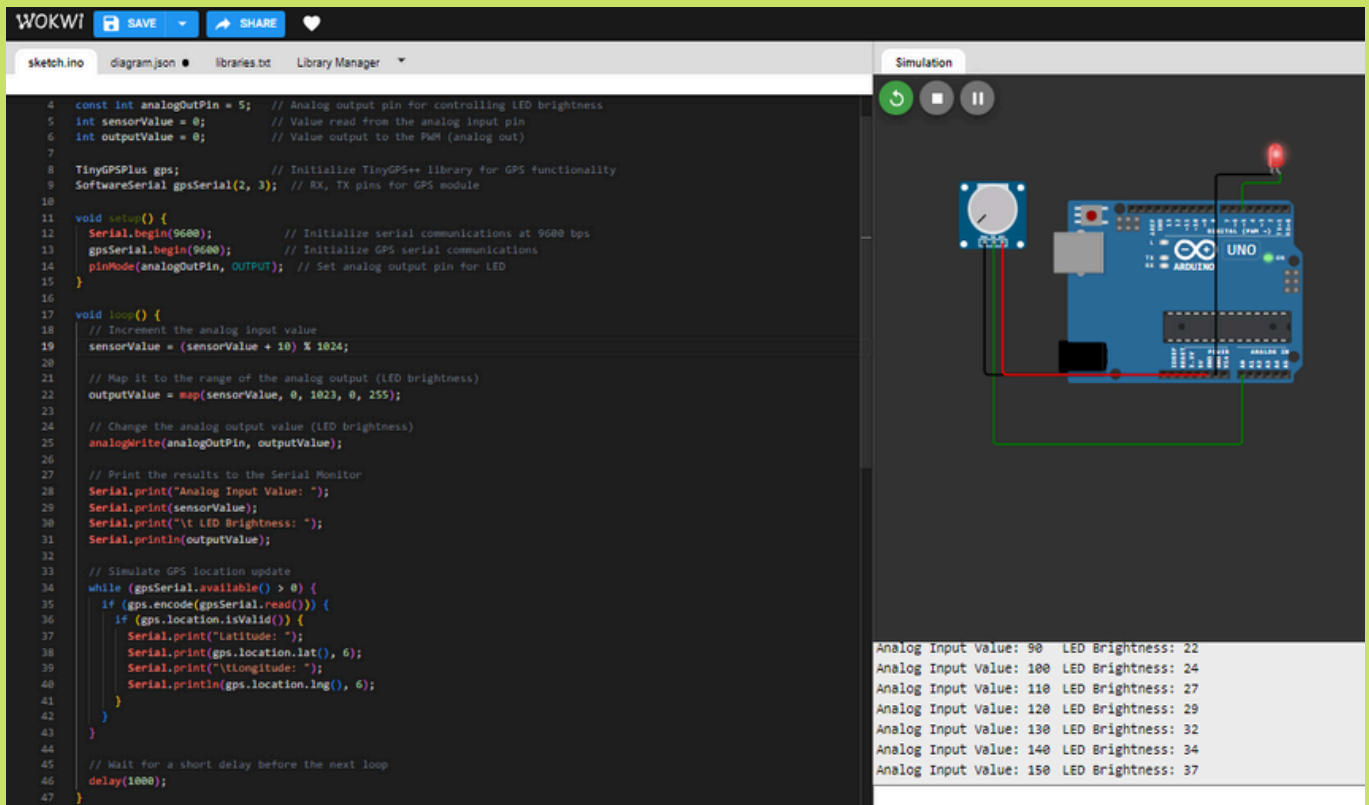
  // Map it to the range of the analog output (LED brightness)
  outputValue = map(sensorValue, 0, 1023, 0, 255);

  // Change the analog output value (LED brightness)
  analogWrite(analogOutPin, outputValue);

  // Print the results to the Serial Monitor
  Serial.print("Analog Input Value: ");
  Serial.print(sensorValue);
  Serial.print("\tLED Brightness: ");
  Serial.println(outputValue);

  // Simulate GPS location update
  while (gpsSerial.available() > 0) {
    if (gps.encode(gpsSerial.read())) {
      if (gps.location.isValid()) {
        Serial.print("Latitude: ");
        Serial.print(gps.location.lat(), 6);
        Serial.print("\tLongitude: ");
        Serial.println(gps.location.lng(), 6);
      }
    }
  }

  delay(1000);
}
```



Thing 2: Emergency Assistance

```

const int emergencyButtonPin = 2; // Digital pin for the emergency assistance button
bool emergencyActive = false; // Flag to track emergency status

```

```

void setup() {
  Serial.begin(9600); // Initialize serial communications
  pinMode(emergencyButtonPin, INPUT_PULLUP); // Setting emergency button pin as input with internal pull-up resistor
}

```

```

void loop() {
  // Read the state of the emergency button
  bool buttonState = digitalRead(emergencyButtonPin);

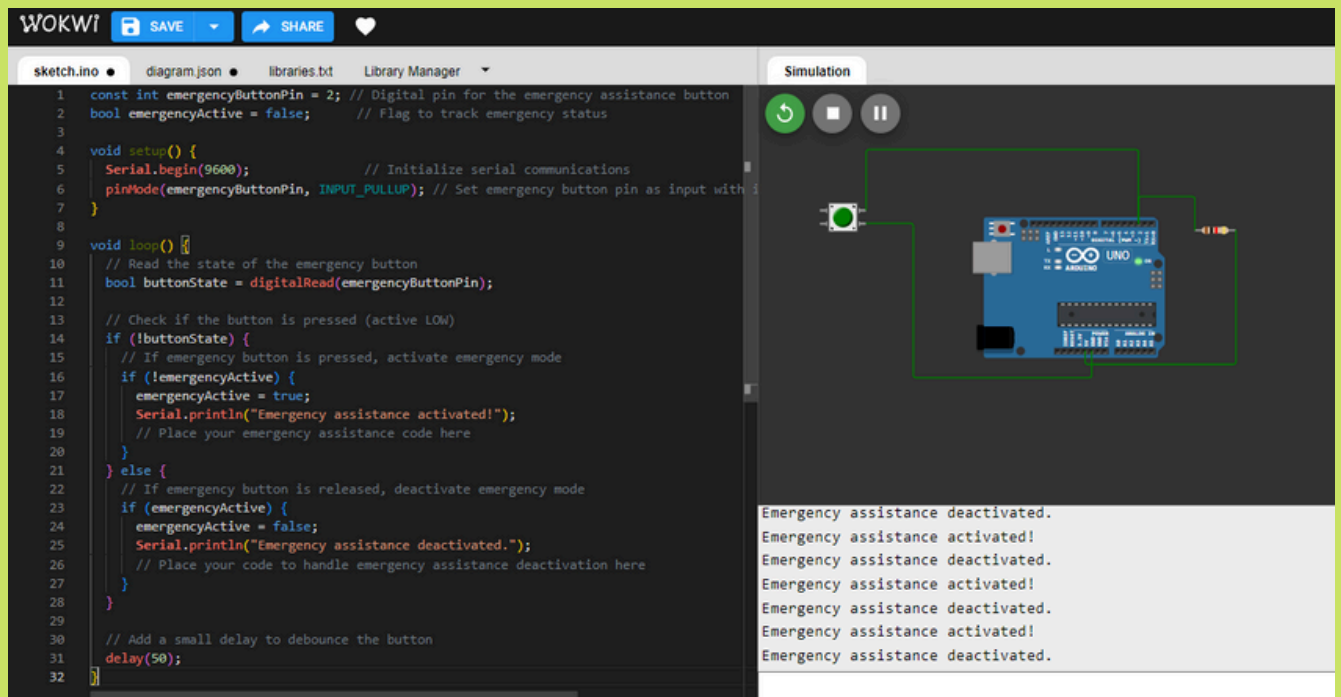
  // Check if the button is pressed (active LOW)
  if (!buttonState) {
    // If emergency button is pressed, activate emergency mode
    if (!emergencyActive) {
      emergencyActive = true;
      Serial.println("Emergency assistance activated!");
    }
  } else {
    // If emergency button is released, deactivate emergency mode
    if (emergencyActive) {
      emergencyActive = false;
      Serial.println("Emergency assistance deactivated.");
    }
  }
}

```

```

// Added a small delay to debounce the button
delay(50);
}

```



Thing 3: Vehicle Profiles

```

struct VehicleProfile {
    String type;
    String make;
    String model;
    // Add more attributes as needed
};

```

```

// Creating an array to store multiple vehicle profiles
VehicleProfile profiles[3];

```

```

void setup() {
    Serial.begin(9600);

```

```

    // Initialize vehicle profiles
    profiles[0] = {"car", "Toyota", "Corolla"};
    profiles[1] = {"motorcycle", "Honda", "CBR600RR"};
    profiles[2] = {"truck", "Ford", "F-150"};
}

```

```

void loop() {
    // Print each vehicle profile
    for (int i = 0; i < 3; i++) { // Assuming 3 profiles
        printProfile(profiles[i]);
        delay(1000); // Delay for readability (adjust as needed)
    }
}

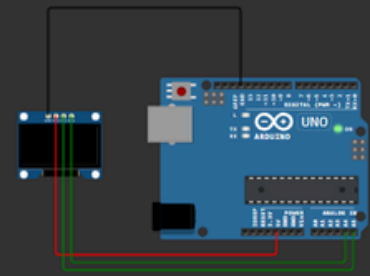
```

```

// Function to print a single vehicle profile
void printProfile(VehicleProfile profile) {
    Serial.print("Vehicle Type: ");
    Serial.println(profile.type);
    Serial.print("Make: ");
    Serial.println(profile.make);
    Serial.print("Model: ");
    Serial.println(profile.model);
}

```

```
2 struct VehicleProfile {
3   String type;
4   String make;
5   String model;
6   // Add more attributes as needed
7 };
8
9 // Create an array to store multiple vehicle profiles
10 VehicleProfile profiles[3];
11
12 void setup() {
13   Serial.begin(9600);
14
15   // Initialize vehicle profiles
16   profiles[0] = {"car", "Toyota", "Corolla"};
17   profiles[1] = {"motorcycle", "Honda", "CBR600RR"};
18   profiles[2] = {"truck", "Ford", "F-150"};
19 }
20
21 void loop() {
22   // Print each vehicle profile
23   for (int i = 0; i < 3; i++) { // Assuming 3 profiles
24     printProfile(profiles[i]);
25     delay(1000); // Delay for readability (adjust as needed)
26   }
27 }
28
29 // Function to print a single vehicle profile
30 void printProfile(VehicleProfile profile) {
31   Serial.print("Vehicle Type: ");
32   Serial.println(profile.type);
33   Serial.print("Make: ");
34   Serial.println(profile.make);
35   Serial.print("Model: ");
36   Serial.println(profile.model);
37 }
```



Vehicle Type: car
Make: Toyota
Model: Corolla
Vehicle Type: motorcycle
Make: Honda
Model: CBR600RR
Vehicle Type: truck

LEARNING AND INTELLIGENCE

The NavAuto application needs to be taught with assistance of a combination of supervised learning and data that we can obtain from several different data sources. Here's an outline of how the system could learn:

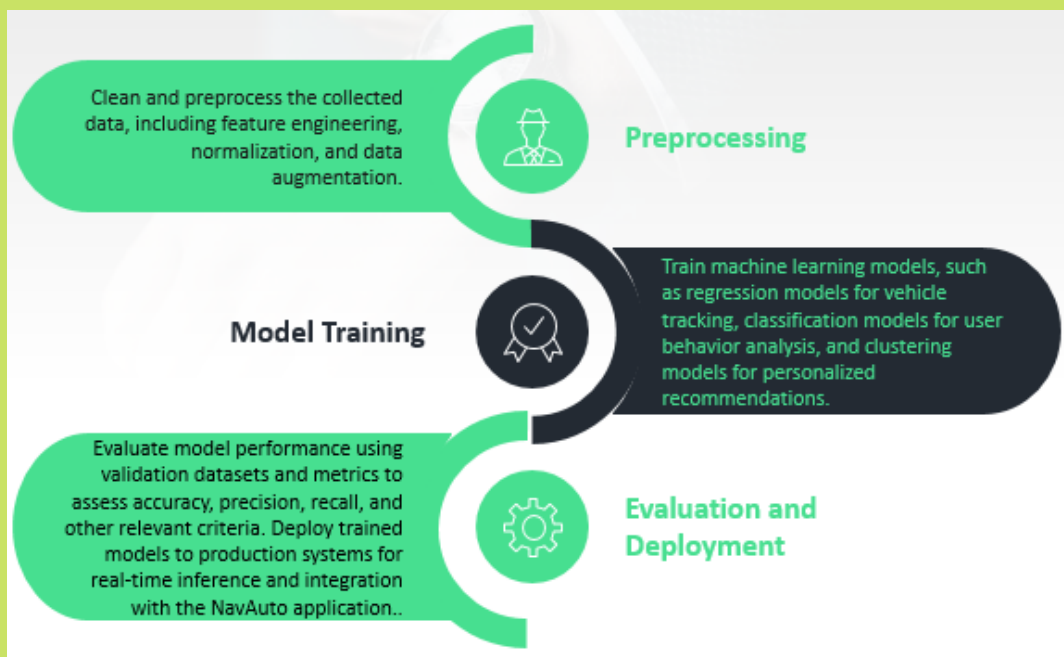
In my opinion I would first collect data by the using the following methods:-

- **Public Dataset:** Leverage public data including vehicle tracking / GPS data, vehicle diagnostics, and user behavior for evaluation. These datasets serve as extremely useful resources for building machine learning models, which need labeled datasets.
- **User Interaction:** Obtain data by monitoring the user interactions with the NavAuto app which include the in-app user preferences, vehicles utilization patterns, and system responses to the user queries.
- **Real-Time Data:** A collection of real-time data streams including GPS sensors, vehicle onboard diagnostics (OBD-II) and other IoT devices. New information is added to the system continuously.

The two challenges that I foresee for training are:

- **Data Quality:** Online engagement to capture and synthesize the digital conversations from various sources, quality and consistency of collected data can be particularly difficult. Such process will include the likes of noise, outliers, and missing values.
- **Labeling and Annotation:** Labeling of training data that have intermixing with machine learning, that will include: for example, vehicles location or user preferences, are often laborious to develop and may involve manual annotating or expert knowledge.

My training and inference pipeline would look like this:



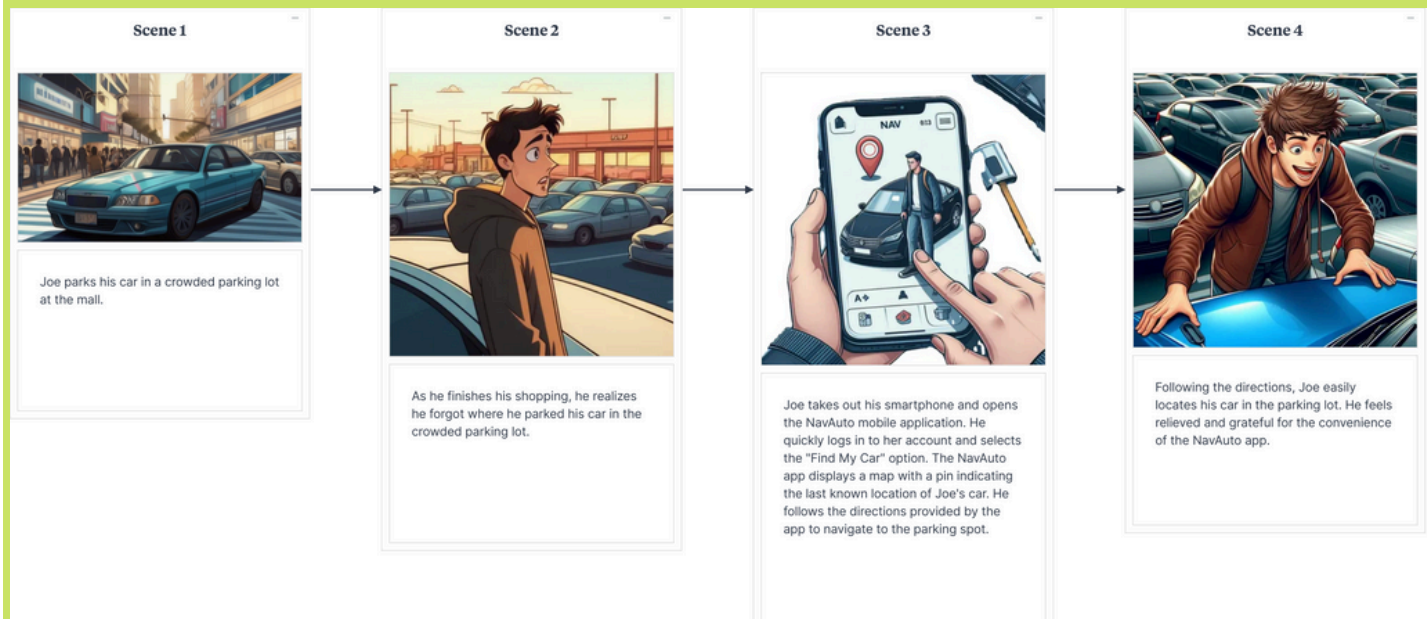
The four metrics which I would choose to measure the performance of the system are:

- **Accuracy:** Accuracy indicates how well the system predicts, i.e. its ability to predict correctly. In the perspective of NavAuto the accuracy signifies the system's capacity to properly track and locate vehicles so the input data supplied is accurate. It's a measure, where number of correct answered questions divided by number of all answers.
- **User Satisfaction:** User safety depicts a level of happiness of users towards correctness of features, recommendation, and overall convenience of our application. It is the one that can be evaluated on the basis of user feedback through surveys, forms, and app ratings. A high satisfactory feedback from these end-users symbolises the fact that the application fulfills users' needs and requirements optimally.
- **Latency:** Delay here means the time between an action or a request from the user and the system which gives the necessary output after this period. NavAuto contains latency that is represented in form of the speed with respect to the time for tasks like vehicle tracking update, alerts, and interactions between systems. Lower response time corresponds to swifter and more immediate interaction with the user which is aimed at improving user experience.
- **Robustness:** Robustness focuses on the ability of the system to tolerate noisy environments, to cope with variability of signals, and to perform well in difficult operational settings. In NavAuto, like any other embedded system, robustness refers to application reliability for different scenarios, some natural ones, e.g. poor GPS signal, severe weather, or network connectivity problems. A good system can maintain productive discourse and accuracy even in the height of challenge or hardship.

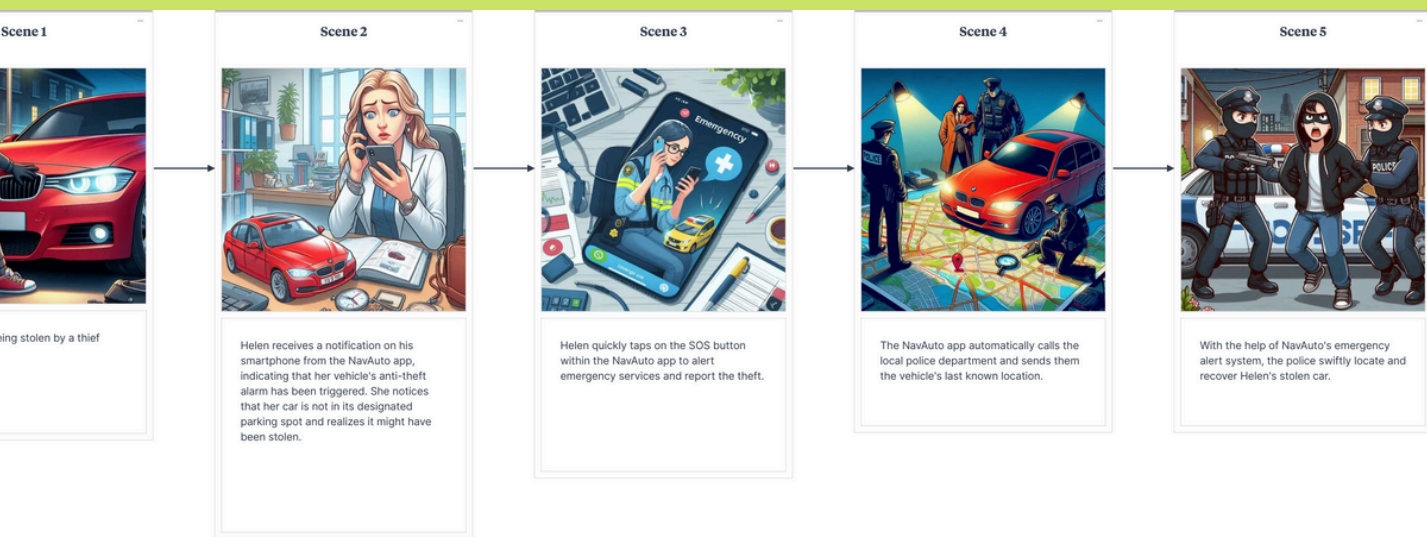
MANAGING HUMAN-COMPUTER INTERACTION

Provided here are three storyboards displaying the practical use-case of the proposed solution.

Storyboard 1: Finding your parked car in an crowded environment




Storyboard 2: Responding to Car Theft




Storyboard 3: Fleet Management

Scene 1




Alex is a fleet manager.

Scene 2




He logs in to his NavAuto account and navigates to the vehicle profiles section where various vehicles are found with its own details.

Scene 3



Using the NavAuto dashboard, Alex can monitor the real-time location and status of each vehicle in the fleet. He receives alerts for events such as vehicle leaving designated areas or low battery levels.

Scene 4



NavAuto helps Alex ensure compliance with regulations and safety standards by tracking vehicle maintenance and driver behavior.

MANAGING SECURITY AND PRIVACY

The context of Location Beacons for Vehicle Tracking in the NavAuto application implies that this process raises a number of questions regarding data protection and safety. Sufficient measures to save the data privacy and security should be provided. Some of these challenges include:

Data Privacy: Collecting and saving the data about users' vehicles' location as well as their vehicle identity details concerns the data privacy. These individuals may be concerned about disclosing their specific information, particularly regarding its storing and access by third parties.

Data Security: Transporting location data from beacon to mobile app and cloud servers must be done with strong encryption to ensure that no unauthorized party will have the ability to see or get the information.

Unauthorized Access: Besides, malicious observers can try to bypass the restrictions by unauthorized data usage of the NavAuto application or attack the firmware of the beacon to change/tamper location data, spy on humans, or disrupt the system.

Physical Security: Beacons positioning technologies fitted into vehicles might be susceptible to physical tampering or theft after which the system in place may fail, the integrity of the tracking system compromised.

User Authentication: Permitting only licensed users for the NavAuto app and enable them to get to know vehicle location data is essential to stay away from unauthorized following or the system's abuse.

Addressing these security and privacy challenges requires a multi-layered approach:

1. **Data Encryption:** The use of end-to-end encryption of data transfer between the beacon, mobile application and cloud servers determines the security of the users location data and saves it from unauthorized access or interception.
2. **Access Control:** Besides effective multi-factor authentication processes, robust user authentication mechanisms can serve to ward off unauthorized access to the NavAuto platform and users data.
3. **Firmware Security:** Continuously monitoring and updating firmware of location beacon, checking for potential security breaches resulting from this exploitation by malicious actors.
4. **Privacy by Design:** By providing NavAuto application with privacy preserving components, which could be anonymous location data processing or allowing users to choose data collection, will be elevating user trust and conformation with privacy laws.
5. **Physical Security Measures:** The application of physical controls, including the use of tamper-evident enclosures, or theft prevention measures, is a great deterrent to people with unauthorized access and tampering of beacons location.

In addition to security and privacy concerns, other forms of risk that need to be addressed in deployment include:

1. **System Reliability:** Making sure that the validity and fail-free performance of NavAuto app and location beacons to provide user with service continuity and the no downtime the user may experience.
2. **Regulatory Compliance:** Implementing meaningful policies in line with the territories of GDPR or CCPA that tackle the lawful and ethical treatment of user data and which does not come up with legal issues.
3. **User Education:** Usage of clear information to riders about the reasons, purposes, and protection of their data during its collection and processing helps NavAuto lessen privacy concerns and raise the level of trust in its system.

By proactively addressing these security, privacy, and deployment risks, the NavAuto application can provide users with a secure and reliable vehicle tracking solution while maintaining their privacy and data protection rights.

Approaches to Evaluation

To evaluate the design concept of the NavAuto application, the following criteria will be considered along with corresponding tests and acceptance criteria:

1. Functionality:

- a. Test: Make sure that users are able to think from a distance about their vehicle location, receive alerts, and manipulate fleet effortlessly.
- b. Acceptance Criteria: The software can resolve fleet management tasks with ease, it sends immediate alerts on the occurrence of ward-off events or towing and enables users to track vehicle location without faults.

2. Usability:

- a. Test: Do user testing to find out how simple or complex users find it to navigate the site, if the interface is clear or not and if features are performant or not.
- b. Acceptance Criteria: Users can perform common tasks (e.g., locating vehicle, setting alerts) without confusion, and feedback from user trials indicates a positive user experience.

3. Reliability:

- a. Test: Carry out a system-level test which will let us imitate different situation (for example, a bad GPS signal and network breaks) and observe the system's reaction.
- b. Acceptance Criteria: The system has a high operation efficiency, capable of running under different environmental elements and operations with a very low error rate.

4. Security:

- a. Test: Perform security assessments to detect vulnerabilities (e.g., encryption,)-access control, and penetration testing to simulate attempted attacks.
- b. Acceptance Criteria: It establishes robust encryption for data swapping, strong user authentication, and is actively resistant to intrusion without violating user data privacy or system integrity.

5. Scalability:

- a. Test: Carry out the performance testing scenario, based on an increasing number of loads (amount of data and users), so that scalability is ensured.
- b. Acceptance Criteria: Whether it is a large number of users or vast data usage, the system will perform smoothly without any depreciation of its performance or functionality.

6. Energy Efficiency:

- a. Test: Assess the energy consumption of the NavAuto application and the location beacons during the average usage orders.
- b. Acceptance Criteria: The beacons and apps additionally work on the principle of energy production, which gives better results and as purposed there is no impact on the car's performance.

7. Communication Bandwidth Requirement:

- a. Test: Measure the bandwidth usage of the NavAuto application during data transmission.
- b. Acceptance Criteria: The app does not demand much bandwidth on the wireless communication and this means that data usage and network congestion are lower compared to other options.

8. Data Privacy Compliance:

- a. Test: Assess compliance with relevant data protection regulations (e.g., GDPR, CCPA) and user privacy preferences.
- b. Acceptance Criteria: The privacy is given high priority. Data is kept secure, private information is inaccessible to unauthorized parties and people are able to change their privacy settings.

Two risks when deploying the system in the wild include:

1. **Data Breaches:** The risk of unauthorized access to user data due to security vulnerabilities or data breaches.
2. **User Adoption:** The risk of low user adoption or dissatisfaction with the application due to usability issues or lack of perceived value.

In the spirit of resolving those risks embarking on continuous monitoring with updates, user education and the privacy settings as well as asking the users for their feedback for improvement of the iterations. The evaluation will comprises thorough testing which includes component testing, system testing, user testing, and compliance assessment to confirm that NavAuto has implemented appropriate criteria and that human error is mitigated as much as possible.