Developer documentation

Group 6:The Minions

EcoDriver

1.VISION

The app is for drivers of trucks, buses and cars who want to drive in a safe and eco-friendly way. Eco Driver, an Android app that's safe to use while driving will coach the user towards safer, cheaper and greener driving patterns. Unlike our competitors our coaching will be with our users for all trips that they choose to take, which will allow the user to save money by reducing fuel consumption.

2.User stories

Examples of user stories:

As a developer I want to figure out to what data to use for the application.
As a driver I want to analyze my trips so that I can drive more eco friendly
As a driver I want to get tips on how to lower fuel consumption before driving.
As a developer I want to have a running GUI prototype so that I can interact with App.
As a driver I want to get continuous feedback while driving so I can improve my driving.
As a driver I want to get tips on how to improve my driving in order to drive more eco friendly.

Examples of developer Tasks:

-Track the RPM signals .
-Have warning when RPM is too high.
-Set up a database for information from GPS.
-Retrieve all the information for the database.
-Check through AGA signal and pick out the one that we want to use.

3.General system description

The system is built on Android (minSdkVersion 19, targetSdkVersion 21). The app has two major parts Track Driving and Analyze Driving.

3.1.Back-end

Our system back-end consists of a database, which was implemented by the Android built-in SQLite.

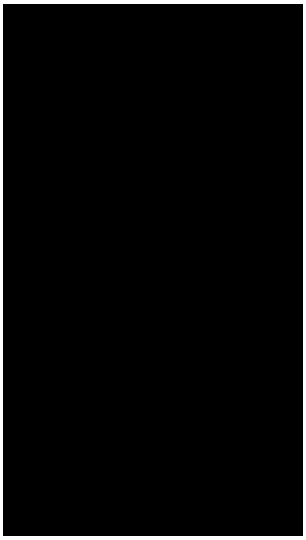
3.2.Front-end

The front-end of our system is made with a combination of Java and XML. The XML is used for designing the GUI, and Java is used for the main functionality of our app, the logic part.

4.Major parts/component

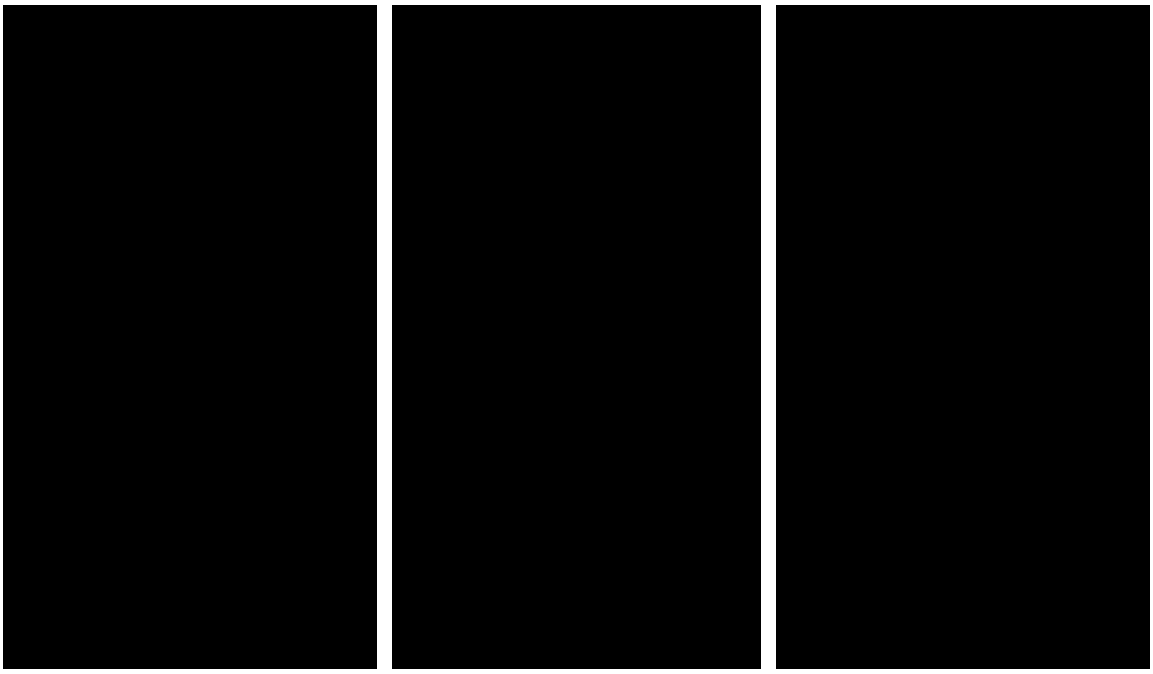
4.1.Track Driving

Track Driving is the real time tracking part of Eco Driver. It utilizes the AGA signals in order to warn the driver when he is not driving eco friendly. This will reinforce good driving behavior while being safe. The sounds when the driver breaks the limits set by the app, such as driving at a too high RPM or accelerating or decelerating to quick, are unobtrusive and allows the driver to keep his eyes on the road.



4.2.Analyze Driving

Analyze Driving is where the driver can analyze the performance of past drives. Each drive consists of information related to the drive such as when it began, how long and how far the drive was. There is also a map where the drive is plotted.



4. Design decision

For designing the user interface for our application, the first thing that comes to our mind was to focus on user interaction and experience, while being efficient, simple, clear, and straightforward. We made it so simple yet straightforward since our target users are drivers and they should not be distracted by the app when they are using it while driving a vehicle.

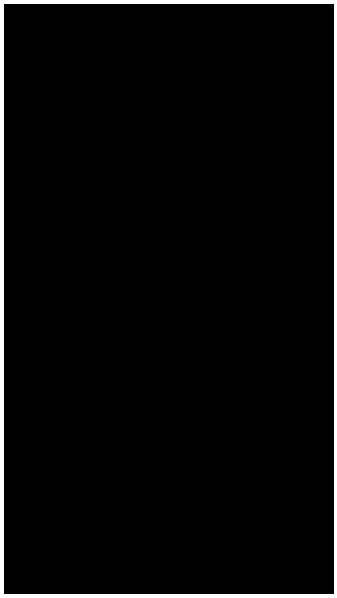
We chose to go with a green and black color scheme since it has a high contrast which allows for good readability while conveying the feeling of being environmentally friend. Seeing as the eye is well adapted to the soft green color that we have used will not make it demanding on the driver's eyes. As opposed to if we used a high contrast color such as yellow which is to high contrast which makes it difficult to read on a black background.

For the buttons we chose to go with big actionable buttons and also trying to play on the associations that a user will be aware of such as a green button is start and a red button is stop. For the size of the buttons we tried to make them so that they could easily be pressed by a thumb. For our use case this is especially important since the app is used while driving.

The main screen has two big buttons, Track driving and Analyze driving giving the user the option to choose between real time tracking and post driving analysis. Visibility of

application status is one of the most vital elements of a good user interface. To achieve this we designed progress bars with gradient colors to show progress whenever speed, rpm, and fuel are changing during driving in Track driving part.

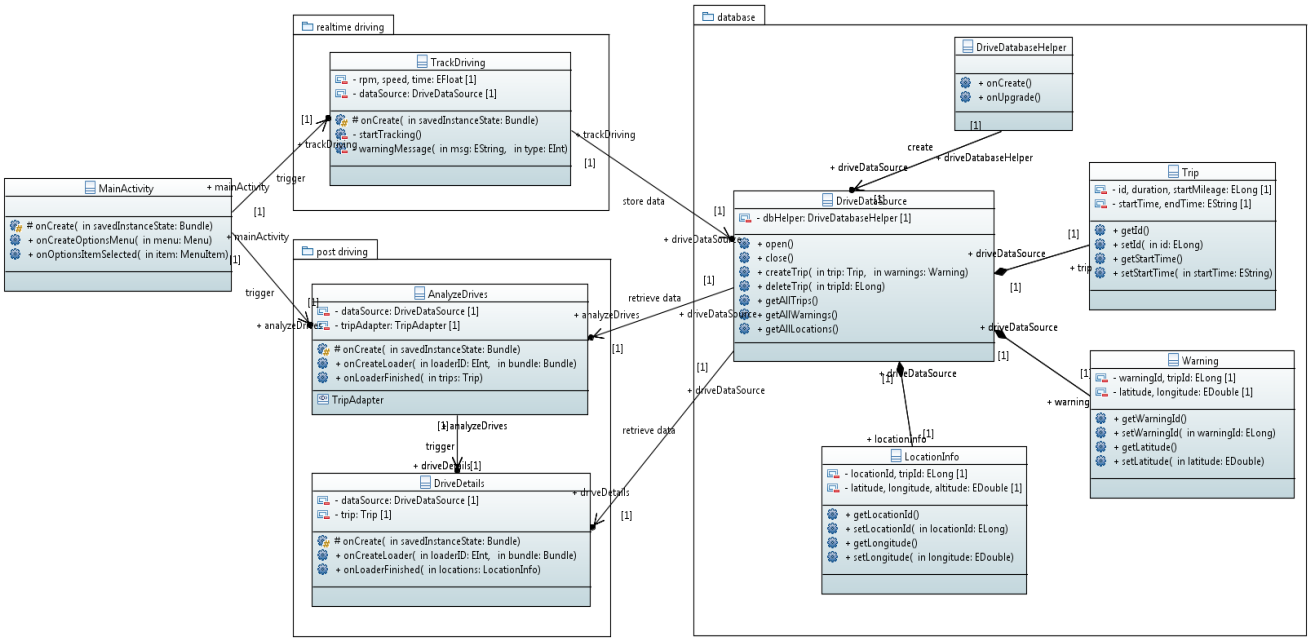
For the purpose of the app we could have gone without any progress bars on the Track Driving part of the app since the information that we display is already visible on the dashboard of the car. However our tests showed that it was confusing since the user might think that the app is stuck and is not tracking the drive. Here is the main screen of the application:



5.UML

5.1.Class diagram

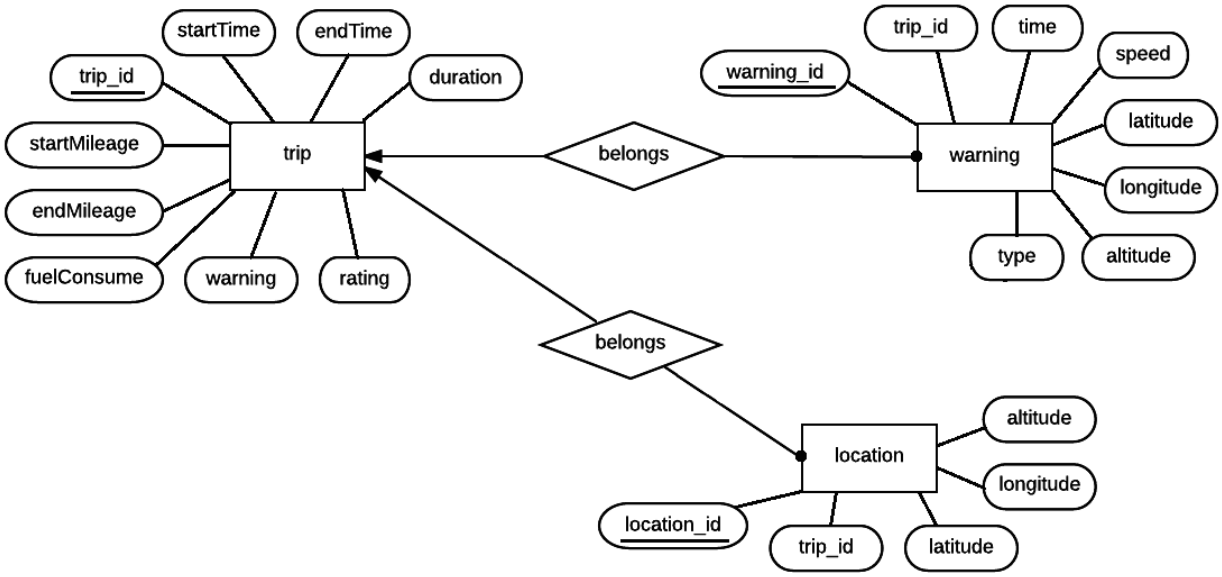
The class diagram of our application consists of two main parts. One part is the database, which is including five classes, and the other part is GUI with AGA functionality.



5.1.2.ER diagram for the database

There are three tables in our database that are Trip table, Warning table and Location table. In our database Trip table and Warning table are related one to many, it is the same for Trip table and Location table with one to many relation. The trip_id is the primary key in the Trip table, which gives every trip a unique ID. This ID is used to specifically identify each trip in the database.

There are eight attributes in the Waring table with warning_id as primary key and one foreign key of trip_id that is referred from the Trip table. It is almost the same for Location table, consisting of five attributes with location_id as primary key and trip_id as foreign key.



6.External dependencies

We have used Google Maps API for Android for our map part which requires Google Play Services (7.3.0).
AGA related dependencies:
Automotive-API-1.1.jar
lib.jar
SDP-1.1.jar
VIL-1.1.jar
and slf4j-android:1.7.10