Final Report

Group 13 - Road Assistance System

Tobias Nersing: tobsson@gmail.com
Andrius Murnikovas: andrius.murnikovas@gmail.com
Christos Gialias: christosgialias@gmail.com
Per Skytt: perskytt@gmail.com
Magnus Selmemo: magnus.m.johansson@gmail.com
Suzanné Ali: suzan.alsamraai@gmail.com

Index

Introduction

The requirements for this project were simple. It had to be developed in Android Studio, it should be catered towards truck drivers and it should collect information from the Automotive Grade Android platform (AGA).
Based on these requirements we decided to create a navigation application. Our idea was to enable truck drivers, from any point in their trip, to search for nearby rest stops, including restaurants, gas stations and even parking lots.
That way they would simply need to press a button while driving instead of manually inputting and searching which decreases time spent focusing on the phone/tablet. To further emphasise how we wanted the driver to keep their focus away from the android device we decided to block non navigation features while the vehicle's in motion.
We still needed to implement data gathered from AGA into our app and we got the idea for a statistics page where the user could see information about how he fast he was driving, how distracted he was etc.
After several modifications and months of development, Road Assist was born.

Development Process

The methodology of our development for this project was scrum. While we did know what scrum was and how you worked in a scrum based environment, neither of us had any experience of working on it before.

The goals we set during our first sprint meeting was to set up our workspace and familiarise ourselves with Android studio and github. We also discussed what we wanted our application to be and took note of the more worthwhile ideas. During the meetings that followed we built upon our ideas and settled on making an application that would function as a navigation app and also help truck drivers find rest stops in close vicinity to their current location.

The next sprint meeting we focused on developing user stories and setting up our backlog as well as further solidifying our idea into a concrete and manageable application. So we decided that the application would be a navigation app, with which the user from the map screen could easily search for nearby rest stops, as well as view a statistics page where important data recorded during the trip would be saved, such as speed, distraction level etc.

For the navigation part of the app we choose to use Google Maps API since it's the most well known (which means lots of documentation is available) and since we figured that it also contains the most information when it comes to locations in comparison to other navigation APIs. This is crucial to the project since our app hinges on the ability to find a wide variety of possible locations of different categories.

At first we wanted the app to search for rest stops based on a schedule that the truck driver would input. The schedule would wait until the truck driver needed to take a break and then automatically search for somewhere to eat and rest. But we wanted to include more than just restaurants so we decided that the truck driver should have the choice between finding somewhere to eat, a gas station or a parking lot. So the schedule was removed in favor of a quick access menu where the truck driver could simply press what he was currently looking for and the app would find it.

For the third sprint meeting a simple alpha version had been created which contained a main screen and the ability to display its current location in google maps. The project didn't develop much during this sprint due to lots of work in the other courses and upcoming exams. The fourth sprint didn't see much development either due to holidays but we did look into alternatives to Google Maps Navigation, since there is no Google Maps Navigation API that would let us put the Google Maps Navigation inside our app, which severely hinders us from doing our own changes interacting with it.

For the beta presentation the application could search and display a navigation route that the user inputted. During the sprint meeting afterwards we designed the database to use for the statistics part as well as implementing it, first at an online host called freemysql but was later transferred to 000webhost due to the limitations in data storage that freemysql possessed.

Php files for the database connection were created and development for the quick access menu and the statistics page was underway. To get the data for the statistic page we used values from AGA and the information that we were interested in obtaining included, vehicle speed, fuel consumption, brake switch and distraction level.

The following sprint up until the final presentation a login page was created for the users of the application, the quick access menu was in its final stage and the statistics page displayed simplified statistics of the most recent trip as well as charts with more detailed information of all the previous ones. A limitation was implemented which would disable certain functionality of the application if it detected that the car was moving or if the driver was too distracted.

After the presentation the focus has been in bug fixing, implementing small features and reworking the graphics of the application to make it feel more connected with a shared colour and theme.

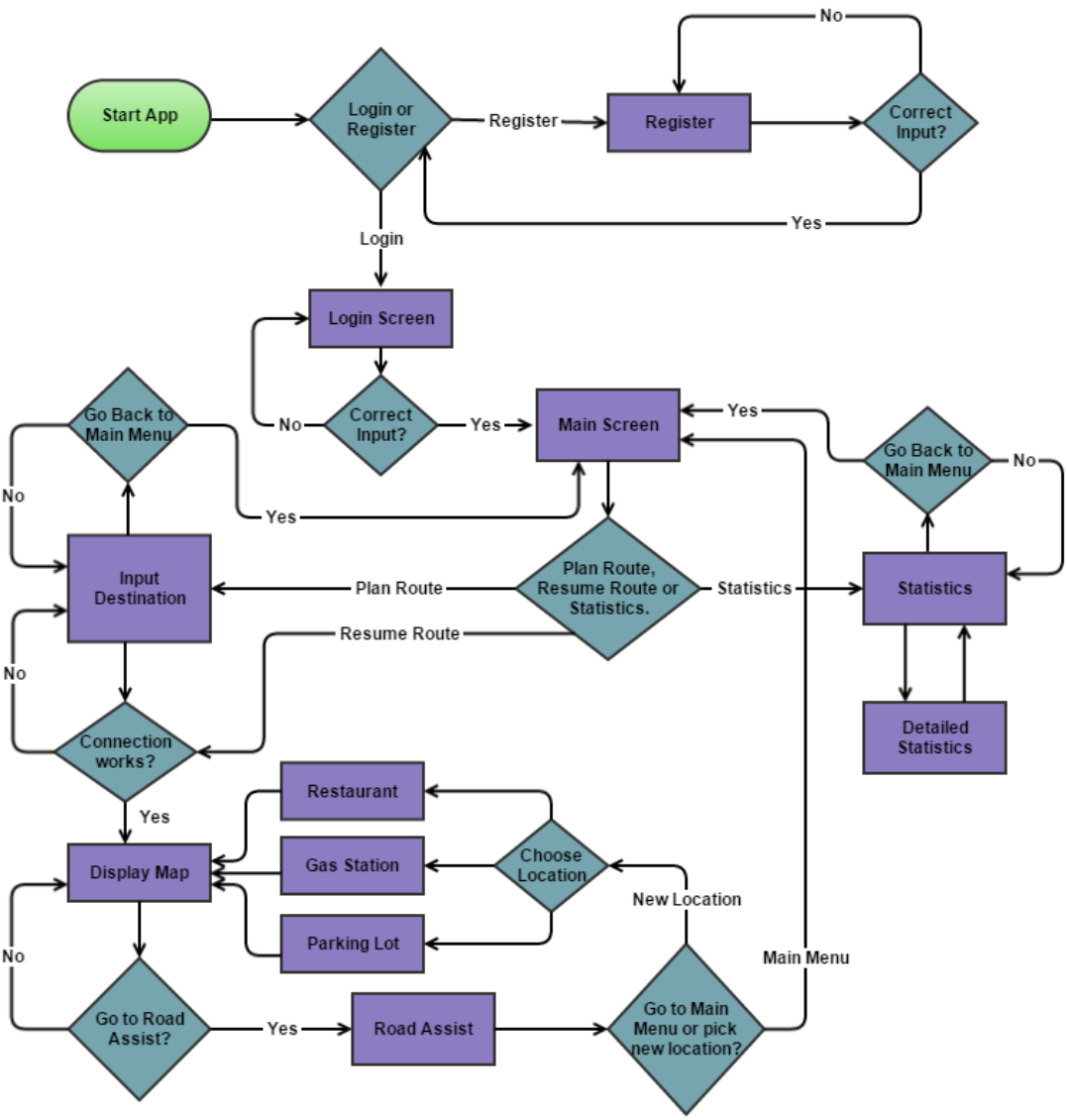
User stories

Here a a few of the user stories we created early in the project.

User Story	Acceptance Criteria
As a driver *I want to* see my current location on the map *So I can* [edit]	a) Current location is viewable in the app.
As a driver *I want to* be able to input a destination so that I can get a planned route. *So I can* [edit]	[edit]
As a driver *I want to* see the planned route. *So I can* [edit]	a) Ability to see the planned route in the app.
As a driver *I want to* loose the ability to interact with my phone when I am too distracted. *So I can* [edit]	a) Interface locks when the distraction level gets to high.
As a driver *I want to* login to get only my own stats shown. *So I can* [edit]	

Flow Chart

A flow chart of how the activities of the application are connected.



External Dependencies

Apart from the libraries provided by the professors (the Automotive API, the Automotive Services, the SDP, and the VIL) we used a few more external libraries.

Our application includes a detailed statistics page with two charts for the speed of the vehicle and the distraction level of the driver during the trip. For creating these charts and displaying the data we used the **MPAndroidChart Library** which provides a great variety of chart types and plenty of options for customizing their appearance.

Furthermore, we used the Google Play Services API, Google Places API and Google Locations API to be able to get the location of the user and to give functionality to some other small features of our app, such as the autocomplete in the location search bar and predictions for the desired destination.

Finally, we used the v7 appcompat library in order to be able to support some new UI features, such as the action bar and the Facebook Rebound library to give a cool spring effect to the buttons in the Navigation Menu (Restaurants, Parking, Gas Stations and Exit)

What major parts/components are there in the application?

Unfortunately, there are no separate packages in our application, which was caused by lack of time in the last days of the development, but three pretty distinct parts can be found in our application which have very limited direct interaction with each other.

The first one is the Log In and Registration activities which communicate with an online mysql database. They check the credentials of the current user and if they are right, the user is allowed access to the application's main features. The registration page/activity allows the registration of a new user/driver and saves his credentials in the online database for future reference.

The second part of our app which is also the main feature consists of plenty of activities. After the user has successfully logged in he is directed to the main screen where he can use the "Plan Route" function of our application. The user inputs a destination

(autocomplete and suggestions function is available) and then he is redirected to the Google navigation application which starts a turn-by-turn navigation to the requested destination. The “RoadAssist-menu”-button also appears on the screen during navigation, which when clicked brings opens a menu with buttons for different kinds of stops available in our app (eg. restaurants, gas stations and parkings) and the “Exit button” which returns the user to the homescreen. One of the most important activities of this component is the timer which collects the values from the AGA simulator every few seconds and then sends them and saves them in our MySQL server with the help of our ApiConnector class which is responsible for the handling the connection and all the requests to our server.

The third part of RAS and the secondary feature is the two statistics pages for the previous trips. The simplified statistics page which shows some info about the last trip is reached by pressing the “STATISTICS” button in the main screen. The user can view the date and time the trip started and finished, the average speed, the average distraction level during the trip, how many times he exceeded the vehicle’s speed limit (set at 90km/h). A list with the IDs of the previous trips can be found in this page as well and by clicking on any of them the user can view a page with detailed statistics where two different charts can be found: A speed/time line chart, where it is also possible to view how many times the speed limit was exceeded and a distraction level/time bar chart (time intervals for both charts are set at 10 seconds). All the statistics values are taken from our online MySQL database and used in the graphs with the help of our ApiConnector class..

Design decisions

We decided to use SDK with minimal API Level 17. The reason for this is that most devices today are using API with level higher than 17. Using API lower than that could cause problems or unwanted behavior in modern devices. Also the decision for this was affected by the devices we own and on which we ran the application for the presentation. Another reason was that this APIs supports all features of coding needed during the application’s development such as Org.JSON with all JSONArray and JSONObject classes to get data and store data for statistics. Android.Location which contains the framework API classes that define Android location-based and related services was also useful during development of the navigation feature.

During our meetings we agreed to design the application with a GUI as minimal and simple to use as possible with easy to read text and big buttons, keeping in mind safe

driving and minimizing the need and somewhat restricting the ability for interaction with the device when the user is driving.

We decided to use the Google Maps App for the turn-by-turn navigation while adding our own menubutton as an overlay to access the roadassist-services while driving, with only one click. We researched several different SDK's and API for making our own turn-by-turn navigation directly implemented in our app, but came to the conclusion that Google Maps app was the best option for our project in this regard.

Protocol

Our application produces a rather big amount of data during the trip which we needed to store somewhere. For storing all the data produced by the application we used a free online MySQl server service (http://www.000webhost.com/). The decision to choose an online MySQL database instead of a local Sqlite one, was affected by the fact that we originally wanted to have a "Fleet Manager" page in our app, which made an online database necessary. Unfortunately this feature wasn't implemented because of the lack of time and the rather insufficient management in our group. Editing the database was done using phpMyAdmin. Communication between the database and the application was done using php files stored in the same server, since direct connection between java and mysql databases is not yet possible. All the results were returned using jsonArrays and jsonObjects which were later processed in the app in order to be properly used by the various functions.