Adapting driver behaviour for lower emissions

MODALES D5.4: Experimental Test Results and Initial Feedback on User Acceptance

WORK PACKAGE	WP5: Guidelines and tools for low emission training
TASK	T5.4: Testing and technical verification of tools
AUTHORS	Nikos Dimokas, Dimitri Margaritis, Katerina Touliou – CERTH *(Hellenic Institute of Transport)* Ramiro Camino, Sébastien Faye, Neamah Hasan Al-Naffakh, Joan Baixauli, Maël Cornil – LIST *(Luxembourg Institute of Science and Technology)*
DISSEMINATION LEVEL	Public (PU)
STATUS	Final, approved by the European Commission
DUE DATE	30/09/2022
DOCUMENT DATE	02/12/2022
VERSION NUMBER	1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 815189.

Quality Control

	Name	Organisation	Date
Editors	Nikos Dimokas, Dimitri Margaritis	CERTH	29/11/2022
Peer review 1:	Ted Zotos	IRU	30/11/2022
Peer review 2:	Andrew Winder	ERTICO	30/11/2022
Authorised by (Technical Coordinator):	Dimitri Margaritis	CERTH	01/12/2022
Authorised by (Quality Manager):	Jean-Charles Pandazis	ERTICO	01/12/2022
Submitted by (Project Coordinator):	Andrew Winder	ERTICO	02/12/2022

Revision and history chart

Version	Date	Main author	Summary of changes
0.1	10/01/2021	Nikos Dimokas, Dimitri Margaritis	First draft of the deliverable with an initial Table of Contents and plan for each section to be discussed with partners.
0.2-0.4	01/05/2021	Nikos Dimokas, Dimitri Margaritis	Specific work on testing and verification of dashboard web application. More precisely, the content about testing data storage capabilities, source code quality and verification, integration, system and performance testing has been updated.
0.5	22/11/2022	Dimitri Margaritis, Sébastien Faye	Update and completion of Section 3.
0.6	23/11/2022	Joan Baixauli, Mael Cornil	Update of Section 2 regarding the testing of the app.
0.7	27/11/2022	Dimitri Margaritis	Version ready for the peer review, taking as an input the latest inputs from partners.
0.8	30/11/2022	Dimitri Margaritis	Update after reviews from Ted Zotos (IRU) and Andrew Winder (ERTICO)
0.9	01/12/2022	Dimitri Margaritis	Pre-final version following the peer review.
1.0	02/12/2022	Andrew Winder	Final version submitted to EC.

Legal disclaimer

This document is issued within the framework of and for the purpose of the MODALES project. This project has received funding from the European Union's Horizon 2020 Framework Programme, through the Innovation and Networks Executive Agency (INEA) under the powers delegated by the European Commission and under Grant Agreement No. 815189. Opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission. Neither the European Commission nor the MODALES partners bear any responsibility for any use that may be made of the information contained herein.

Table of Contents

Index of Figures

Index of Tables

List of abbreviations and acronyms

Abbreviation	Meaning
API	Application Programming Interface
App	Application
DoA	Description of Action
EOBD	European On-Board Diagnostics
PHP	Personal Home Page
CSS	Cascading Style Sheets
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ms	millisecond
OBD	On-Board Diagnostics
OS	Operating System
RDBMS	Relational Database Management System
REST	Representational State Transfer
RESTful	An API that adheres to the REST architecture
SUS	System Usability Scale
SQL	Structured Query Language
UI	User Interface
UX	User Experience
VIN	Vehicle Identification Number
VM	Virtual Machine
WiFi	Wireless Fidelity
WP	Work Package

Executive Summary

This deliverable is part of the EU-funded project's MODALES Work Package 5 "Guidelines & tools for low emission training," one of the project's five technical work packages. Furthermore, this deliverable is the result of the work performed in Task 5.4. "Testing and technical verification of tools", which aims at reporting on the testing and verification result of all the software developed and integrated in WP 5 to make sure that it is in line with the specifications determined in Task 5.2. "Functional specifications of tools".

The present document includes the testing and technical verification of the mobile application for low-emission driving and the dashboard web application that are developed in Work Package 5. This mobile app is then tested with volunteer drivers in several locations across Europe (trial sites) as part of WP6. More precisely, this document describes the testing of the data storage and the performance evaluation of the procedures, queries and the overall functionality that is related to the data storage. Moreover, the source code quality and verification subsection analyses and reports the static code analysis, the maintenance of code quality, the bugs inside the source detected and the security vulnerabilities detected. Additionally, the unit testing is presented that is used to verify that the code performs well and behaves as intended. It also presents the integration testing as well as the system testing. In subsection 2.5 the performance testing is analysed.

Additionally, D5.4 presents the user acceptance procedure that has been adopted and its corresponding initial results. The SUS was used for evaluating the usability of the app by 41 participants from all trial sites. The responses on a scale of 1-5 converted to a score of 62 which is considered "good", taking into account the fact that the app is a research prototype and still under fine tuning until the end of the trail period in 2023. As well as the 10 questions included in the SUS scale, all trial-site participants have been asked to notify any technical issues they encountered while using the app. All issues are listed and solutions are given by the MODALES development team. This problem reporting process is perceived as an on-going one until the end of the data collection period which is taking place in different city-regions in Europe during 2022-23.

Later deliverables after the completion of the on-road trials will present further information on user feedback and acceptance (including of the advice given by the app as well as its functioning / ease of use), building on the initial results in this report.

1. Introduction

1.1. Project overview

The MODALES project works towards reducing air pollution from all types of on-road vehicles by encouraging adoption of low-emission driving behaviour and proper maintenance choice.

MODALES pursues a user-centric approach to addressing all the challenges which on the one hand enhance low-emission practices and on the other hand suppress high-emission behaviour by researching, developing and testing several innovative and complementary solutions in four key areas (driver, retrofits, EOBD and inspection) in order to reduce vehicle emissions from three main sources: powertrain, brakes and tyres.

The main activities of MODALES are:

- Measurement of real-world vehicle emissions and driving behaviour to produce accurate correlation between them using advanced mathematical and statistical techniques.
- Exploration of the most advanced technologies for retrofits designed to substantially reduce powertrain emissions from all types of vehicles and to validate their effectiveness under different real-world traffic and environment conditions, and by various drivers.
- Undertaking an in-depth analysis of OBDs, periodic inspection and legal issues on tampering in Europe to help regulatory authorities put in place effective anti-tampering legislation, and to help owners properly maintain their vehicles.
- Conducting one-year long low-emission user trials (with both driving and maintenance practices), supported by awareness campaigns, to enhance public engagement and help drivers better understand the impact of their driving and maintenance behaviours in all situations.

1.2. Scope

This deliverable is part of Work Package 5 (WP5) on **Guidelines and Tools for Low-Emission Training**, which is one of the five technical WPs of MODALES (the two "non-technical" WPs include WP1 on Project Management and WP7 on Awareness, Communication and Dissemination). The other four "technical" WPs that are directly connected with WP5 are the following:

- **WP2: Defining low-emission factors**, which explored driving behaviour variability using existing available data. This WP delivered a first approach on driving behaviour patterns and powertrain, brake and tyre emissions. It also addressed the state-of-the-art in retrofits, inspection and maintenance (I/M) and legal issues regarding tampering in various EU Member States. Link with WP5: results from this WP (reported in Deliverables D2.1 and D2.2) were used to develop the guidelines earlier in WP5 (D5.1) and therefore the mobile app and the trainings.
- **WP3: Impact of user behaviours**, which undertook a series of measurement campaigns to establish the interconnection between driving behaviour and powertrain exhaust emissions, as well as fine particulates from brakes and mass-loss from tyres. Measurement campaigns were also carried out to address the impact of poor maintenance and deliberate tampering of the emissions control system. Link with WP5: results from this WP (reported in Deliverable D3.2) were used to develop the guidelines and implement part of the profiling methods in WP5.
- **WP4: Effectiveness of inspections and depollution systems**, which used the findings of WPs 2 and 3 as a basis to investigate and propose solutions that will contribute to emission monitoring via the

EOBD protocol and systems that detect lack of maintenance and tampering. It also investigated the potential of enhancing existing retrofit systems. Link with WP5: results regarding OBD use (reported in Deliverable D4.1) were used in WP5to integrate all the necessary data into the mobile app.

- **WP6: User trials and evaluation**, developed an evaluation plan and to test and evaluate with real-world trials the functionality of the innovations developed in MODALES, their effects on driver acceptance and performance, and their potential wider impact (their predicted overall effects on vehicle emissions). These trials use volunteer drivers using the app in a naturalistic way for their normal driving purposes in different cities/regions of Europe (currently in the metropolitan areas of Helsinki, Leeds, Luxembourg, Barcelona, Bergamo, Thessaloniki and Istanbul). Link with WP5: the mobile app is used as the main tool in the on-road trials and analysis in WP6.

The figure below shows how these deliverables fit into the project and highlights related deliverables which will consider the content of this one.

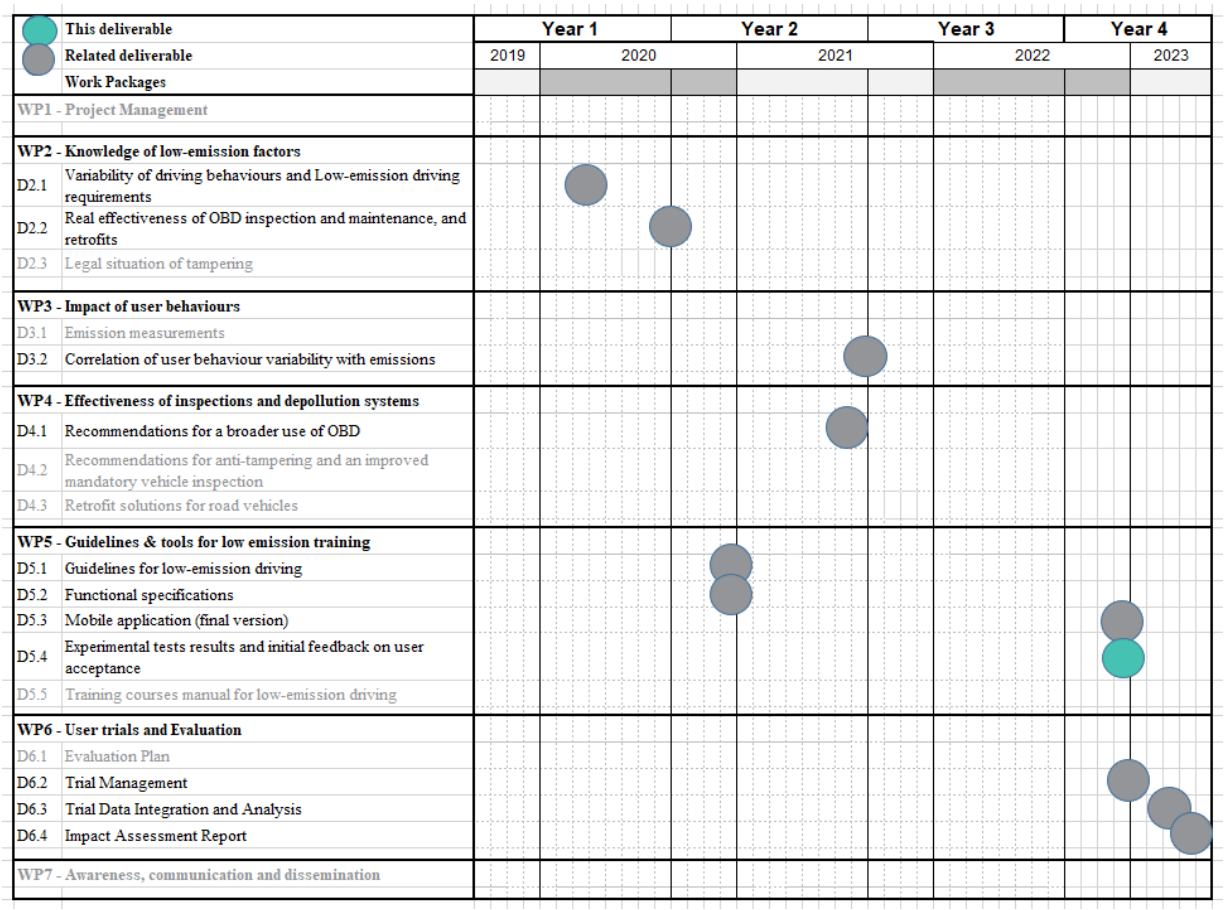

Figure 1: D5.4 in the context of related MODALES tasks and deliverables

WP5 includes five tasks: **Task 5.1. Guidelines for low-emission driving**, **Task 5.2. Functional specification of tools**, **Task 5.3. Low-emission driving assistance tools**, **Task 5.4. Testing and technical verification of tools** and **Task 5.5. Developing trainings for low-emission driving**.

1.2.1. Scope and intended audience of this deliverable

The current document is related to the **Task 5.4. Testing and technical verification of tools** and the main objective is to present the testing and verification result of all the software developed and integrated in WP 5. At the same time, it presents the user acceptance procedure and the results.

This document is organized as follows:

- It starts with an introduction, which recalls the project's objectives, states the deviations from the Declaration of Action, and delimits this deliverable's scope.
- The second section analyses and presents the testing and verification of the mobile application and dashboard web application.
- The third section presents the user acceptance method adopted.
- Finally, the fourth section presents the conclusions resulted from this activity.

The content of this deliverable is Public (PU).

1.3. Deviations from the Description of Action (DoA)

1.3.1. Alignment with the DoA and content deviations

This deliverable is aligned with the content of the DoA.

1.3.2. Time deviations

This deliverable was originally scheduled for Month 24 of the project but rescheduled to Month 37 (September 2022), and then submitted with a further two months delay for the following reasons:

- The delay in the final version of the app (see MODALES Deliverable D5.3), which consequently delayed its evaluation.
- The delay in the scoring system development and implementation.
- The delay in the app usage by the participants that affected the user evaluation phase.

2. Testing and technical verification of tools

The tools that are tested and verified are:

- The **mobile assistant** and its associated services,
- The **internal reporting platform**, which is a web application being used by the trial site leaders allowing to manage the data collected from the mobile application,
- The project's dashboard web application.

The section presents the testing of the data storage capabilities, the source code quality and verification, the unit testing that is used to verify that the code performs well and behaves as intended, the integration testing, the system testing and the performance testing.

2.1. Testing data storage capabilities

The data storage is threefold. Initially, the **mobile application** stores the information retrieved from the OBD and the mobile device's sensors internally, using the memory of the phone. At this stage, the user has a great level of flexibility to manage this data. The mobile application is then frequently sending the datasets to a server, located at MODALES partner LIST (situated in Luxembourg). When the data is retrieved by the server, several filtering and data augmentation processes are triggered. The figure below represents the database structure of this server, containing driving data.

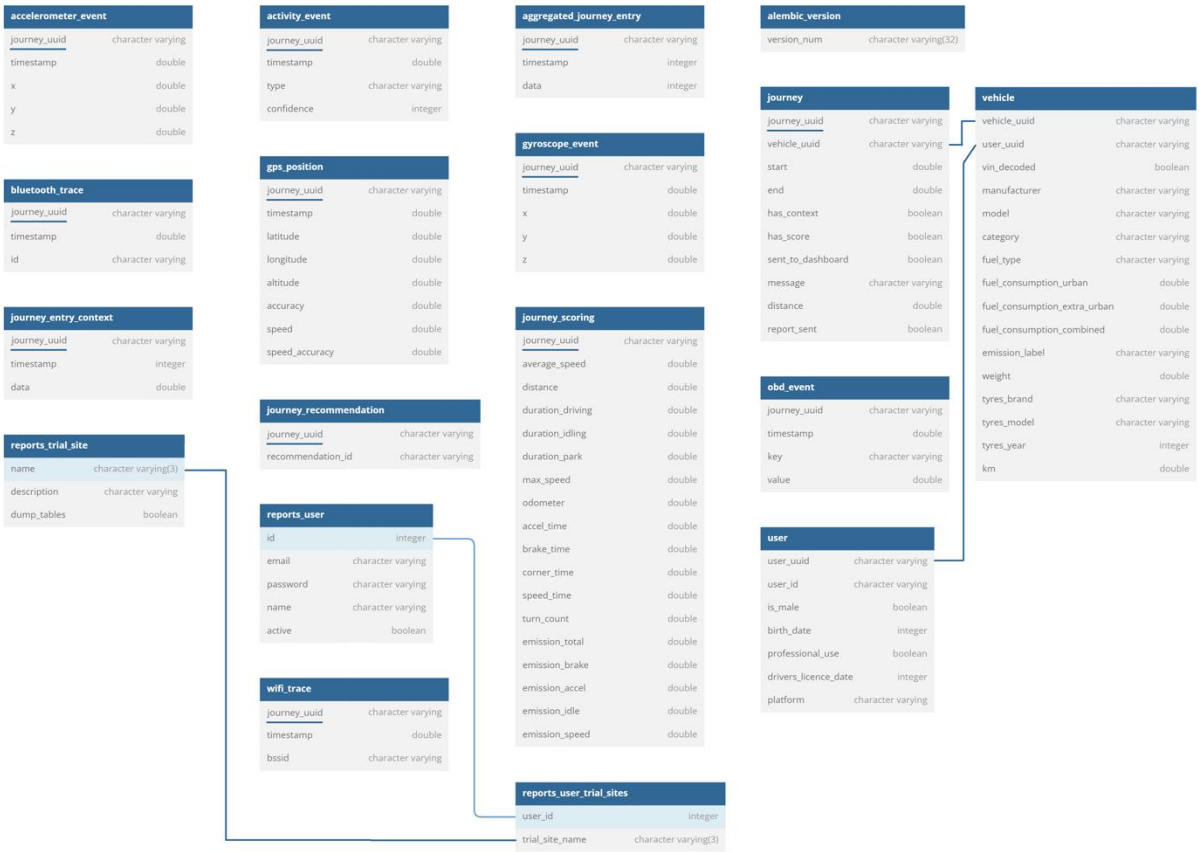

Figure 2: Database structure of the MODALES server

The datasets mentioned above are stored on a PostgreSQL database and are made accessible to the relevant partners using an **internal reporting platform**, where the data is filtered for each of the existing trial sites. A set of APIs and web services are available, so that the consortium can fully exploit the data.

Capacity tests were carried out on the databases, both local (phone memory) and stored on the LIST server. Some conclusions and statistics are presented in the sections below. In all cases, the impact on the end user is extremely low, since:

- The data stored by the mobile application is only text data, regularly sent to the project server and removed from the phone. The synchronisation mechanisms have been tested from different angles.
- The data stored on the server is not used by the drivers. Access is restricted to the project partners. The impact is therefore very low. This is also why in this section we choose to focus on the web dashboard.

The selected data required by the **dashboard web application** are sent to dashboard and stored in a Relational Database. The dashboard application requires a data storage infrastructure in order to generate the aggregated information. The data storage has been implemented with MySQL Relational Database Management System. The data model developed like a relational schema consists of various tables and the corresponding relations among them. The entity relational model below presents the tables and the corresponding attributes (columns).

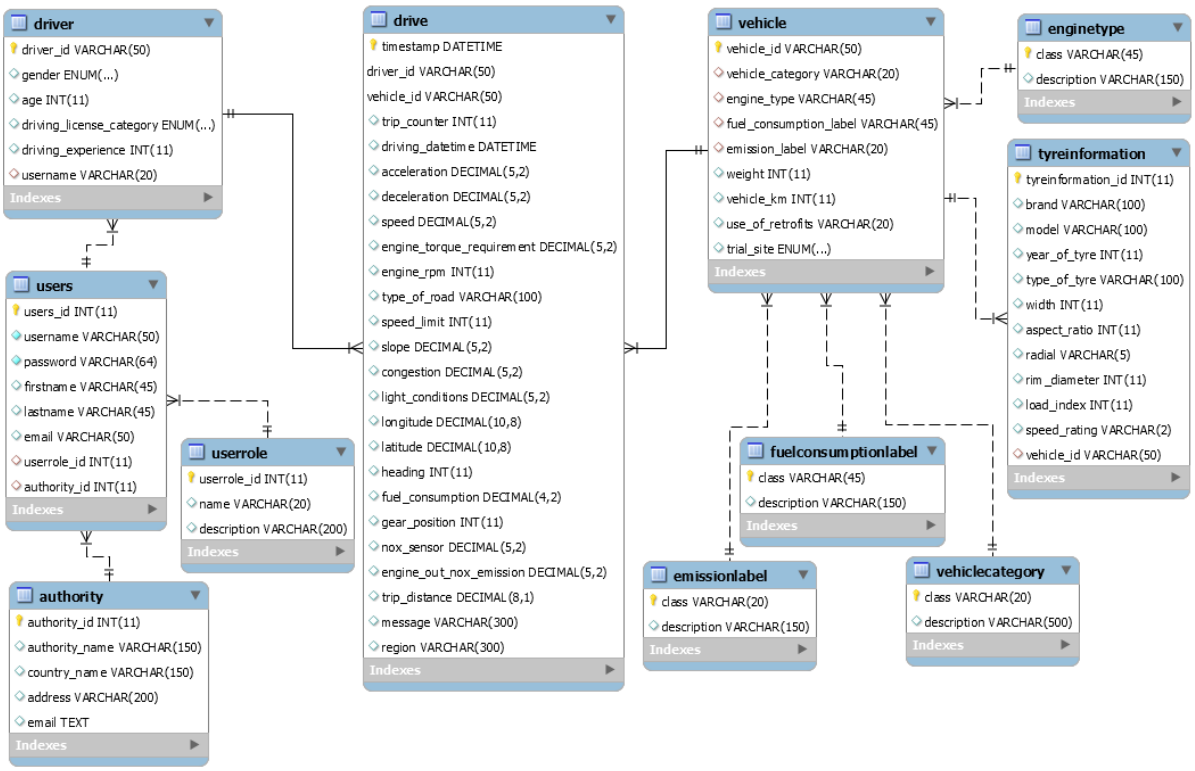

Figure 3: Entity Relational Model of dashboard's data storage

The driver table contains information about the drivers, while the users, "*userrole*" and authority tables stores data about the users. The user information is crucial for the dashboard web application to implement the authentication process. The information about each vehicle is stored in tables

vehicle and "*tyreinformation*", while the tables "*enginetype*", "*fuelconsumptionlabel*", "*emissionlabel*" and "*vehiclecategory*" keeps auxiliary data like the different emission labels or the vehicles' categories etc. The drive table keeps the data from each trip. The "*drive*" table contains a huge number of records since it stores all the interesting data while the driver is on his/her way. The frequency of data is based on amount of data the mobile application receives from the OBD.

Initially, the implementation of the dashboard web application was realized with the use of SQL Views. A number of SQL Views has been implemented inside the relational schema to facilitate the information retrieval. The SQL Views are listed below:

Table 1: SQL Views

View	Description
avg_fuel_consumption_per_roadtype_trialsite	Computes the average fuel consumption grouped by trial sites and road type.
avg_fuel_consumption_per_trial_site	Computes the average fuel consumption grouped by trial sites.
avg_fuel_consumption_per_trip	Computes the average fuel consumption grouped by trip.
avg_nox_per_age_group	Computes the average NOx emissions grouped by different age groups.
avg_nox_per_age_group_trialsite	Computes the average NOx emissions grouped by trial sites different age groups.
avg_nox_per_driving_experience	Computes the average NOx emissions grouped by driving experience. The levels of driving experience are 0-5 years, 6-10 years, 11-20 years and above 20 years
avg_nox_per_driving_experience_trialsite	Computes the average NOx emissions grouped by trial sites and driving experience. The levels of driving experience are 0-5 years, 6-10 years, 11-20 years and above 20 years.
avg_nox_per_emissionlabel	Computes the average NOx emissions grouped by trial sites and emission label.
avg_nox_per_gender	Computes the average NOx emissions grouped by driver's gender.
avg_nox_per_gender_trialsite	Computes the average NOx emissions grouped by trial sites and driver's gender.
avg_nox_per_licensecategory	Computes the average NOx emissions grouped by driving license category. The different categories are "professional" and "non-professional".
avg_nox_per_licensecategory_trialsite	Computes the average NOx emissions grouped by trial sites and driving license category. The different categories are "professional" and "non-professional".
avg_nox_per_roadtype	Computes the average NOx emissions grouped by road types. The road types are "national highway", "rural", "state highway" and "urban".
avg_nox_per_roadtype_trialsite	Computes the average NOx emissions grouped by trial sites and road types. The road types are "national highway", "rural", "state highway" and "urban".
avg_nox_per_trial_site	Computes the average NOx emissions grouped by trial sites.
avg_nox_per_trip	Computes the average NOx emissions grouped by trial sites and trip.
avg_nox_per_vehiclecategory_trialsite	Computes the average NOx emissions grouped by trial sites and vehicle category.
total_km_per_roadtype	Computes the overall distance travelled by vehicles grouped by trial sites and road type.

View	Description
user_avg_fuel_consumption_per_roadtype	Computes the average fuel consumption per user and road type.
user_avg_fuel_consumption_per_trip	Computes the average fuel consumption per user and trip.
user_avg_nox_per_roadtype	Computes the average NOx emissions per user and road type.
user_avg_nox_per_trip	Computes the average NOx emissions per user and trip.
user_total_km_per_roadtype	Computes the overall distance travelled by user grouped by road type.

During the testing of data storage capabilities, the execution time of each SQL View has been computed for different volumes of data. The table below presents the average execution time (in seconds) of each View about:

- 36K records.
  (Considering that the frequency of driving data retrieval can be 1Hz then 36K records correspond to ten hours driving.)
- 100K records
- 200K records
- 500K records
- 1M records

Table 2: Average execution time in seconds of SQL Views

View	36K	100K	200K	500K	1M
avg_fuel_consumption_per_roadtype_trialsite	0.06014400	0.69305250	1.32218450	3.02919900	9.40966850
avg_fuel_consumption_per_trial_site	0.03736475	0.42406375	1.19995175	1.74182075	8.17666700
avg_fuel_consumption_per_trip	0.03969625	0.45748225	0.87028950	1.92020800	8.28103675
avg_nox_per_age_group	0.04736325	0.50139700	0.82263875	1.98471300	15.7398107
avg_nox_per_age_group_trialsite	0.05047925	0.58499375	1.13037525	2.54868675	15.6385770
avg_nox_per_driving_experience	0.03814575	0.45642650	0.85871550	1.95850525	14.4275760
avg_nox_per_driving_experience_trialsite	0.05610475	0.62342525	1.12520500	2.43128525	15.0422937
avg_nox_per_emissionlabel	0.04626150	0.51664700	0.97504925	3.01439475	4.74518050
avg_nox_per_gender	0.04115350	0.47443575	0.72990900	1.61256000	11.5589660
avg_nox_per_gender_trialsite	0.04571375	0.51481875	1.50086275	3.18460875	14.7163302
avg_nox_per_licensecategory	0.03757725	0.44982400	1.17835275	1.76697650	13.8468465
avg_nox_per_licensecategory_trialsite	0.05227750	0.57766100	0.99371225	2.20526575	14.1251385
avg_nox_per_roadtype	0.05180200	0.56925375	1.04522650	2.59427200	5.97550475
avg_nox_per_roadtype_trialsite	0.06276750	0.46110425	0.94303975	1.94168325	4.64214475
avg_nox_per_trial_site	0.04972275	0.42348225	0.77512950	1.51034000	4.09793425
avg_nox_per_trip	0.05259825	0.45671175	0.84535550	1.79198400	4.39342000
avg_nox_per_vehiclecategory_trialsite	0.05601300	0.60734375	0.95798750	1.96841350	4.55164150
total_km_per_roadtype	0.05114475	0.58872375	0.90060200	2.10959575	7.04268150
user_avg_fuel_consumption_per_roadtype	0.05349000	0.50951200	1.03632300	2.37791125	17.6023595

View	36K	100K	200K	500K	1M
user_avg_fuel_consumption_per_trip	0.06420225	0.49981900	0.96350975	2.14789150	19.7397810
user_avg_nox_per_roadtype	0.06116425	0.50521250	1.07331200	2.14089100	17.0318570
user_avg_nox_per_trip	0.05416475	0.51585625	0.94082700	2.08060075	18.5009415
user_total_km_per_roadtype	0.06185125	0.50626200	1.03776700	2.58314175	20.0146670

The execution time grows as the number of records increases. In case of a huge volume (1 million records for example) of data the execution time is above 10 seconds in some cases. So, the user experience will be poor due to slow response time by the system. Since the dashboard web application provides aggregated data periodically and is not an application providing real-time data, we decided to generate Summary Tables in place of SQL Views. An event has been created that generates the Summary Tables periodically (every day). Thus, the response time of the whole system has been improved significantly since the execution time is short in all cases as depicted in the following table. This is because the summary tables contain only aggregated data (a few records per table).

Table 3: Average execution time in seconds for retrieving data from summary tables

Summary Table	36K	100K	200K	500K	1M
tbl_avg_fuel_consumption_per_roadtype_trialsite	0.0002770	0.0002921	0.0003211	0.0003483	0.0003947
tbl_avg_fuel_consumption_per_trial_site	0.0002055	0.0002203	0.0002387	0.0002732	0.0002867
tbl_avg_fuel_consumption_per_trip	0.0041647	0.0047500	0.0051992	0.0053657	0.0058210
tbl_avg_nox_per_age_group	0.0035280	0.0038927	0.0040017	0.0044421	0.0046229
tbl_avg_nox_per_age_group_trialsite	0.0029212	0.0029895	0.0030485	0.0031537	0.0033776
tbl_avg_nox_per_driving_experience	0.0031840	0.0034063	0.0036642	0.0038827	0.0036733
tbl_avg_nox_per_driving_experience_trialsite	0.0037685	0.0038569	0.0040039	0.0043580	0.0045432
tbl_avg_nox_per_emissionlabel	0.0029327	0.0034715	0.0037762	0.0038821	0.0039939
tbl_avg_nox_per_gender	0.0033252	0.0034222	0.0036898	0.0037664	0.0038772
tbl_avg_nox_per_gender_trialsite	0.0032200	0.0033311	0.0035799	0.0037039	0.0040202
tbl_avg_nox_per_licensecategory	0.0032035	0.0032562	0.0033354	0.0034422	0.0037001
tbl_avg_nox_per_licensecategory_trialsite	0.0028822	0.0029007	0.0029984	0.0034173	0.0036940
tbl_avg_nox_per_roadtype	0.0031520	0.0031903	0.0032110	0.0034065	0.0035370
tbl_avg_nox_per_roadtype_trialsite	0.0036772	0.0037842	0.0037995	0.0038412	0.0039748
tbl_avg_nox_per_trial_site	0.0039240	0.0040076	0.0041054	0.0044451	0.0044890
tbl_avg_nox_per_trip	0.0030455	0.0031460	0.0031773	0.0033354	0.0033986
tbl_avg_nox_per_vehiclecategory_trialsite	0.0029585	0.0029789	0.0030442	0.0031109	0.0031295
tbl_total_km_per_roadtype	0.0029947	0.0030175	0.0030847	0.0030992	0.0033221
tbl_user_avg_fuel_consumption_per_roadtype	0.0031097	0.0031901	0.0033551	0.0034341	0.0035870
tbl_user_avg_fuel_consumption_per_trip	0.0044860	0.0046025	0.0046110	0.0047723	0.0048504
tbl_user_avg_nox_per_roadtype	0.0022635	0.0024563	0.0024908	0.0028232	0.0030002
tbl_user_avg_nox_per_trip	0.0043390	0.0046909	0.0048910	0.0054769	0.0057055
tbl_user_total_km_per_roadtype	0.0030525	0.0033881	0.0033919	0.0038564	0.0039919

The database includes also stored procedures that are called by the RESTful web services to retrieve data. The first procedure is called "*getDriver*" and returns driver's related data, while the second procedure is called "*getVehicle*" and return vehicle's related data. The third procedure is called "*isUserRegistered*" and returns whether the user is registered or not. The following table presents the execution time of each stored procedure for different volume of data stored in the database.

Table 4: Average execution time in seconds of stored procedures

Stored Procedure	36K	100K	200K	500K	1M
getDriver	0.00011550	0.00020217	0.00026703	0.00354521	0.00482950
getVehicle	0.00010725	0.00015221	0.00025776	0.00459910	0.00582064
isUserRegistered	0.00011775	0.00014409	0.00034925	0.00663388	0.00700081

Finally, the testing of data storage capabilities also includes the performance evaluation of the data insertion operation. We tested the delay time for sending the data from the back-end (web services) software to MySQL database and storing them to the tables. The performance evaluation implemented by sending random driving data that correspond to 1-hour driving (corresponds to 3600 records in "*drive*" table). The testing procedure executed one hundred times and after that we computed the average execution time. It is important to mention that storing driving data means storing data to "*drive*", "*driver*" (data for each driver is stored only once, the first time the system recognizes that it has no information for that driver), "*vehicle*" (data for each vehicle is stored only once, the first time the system recognizes that it has no information for that vehicle) and "*tyreinformation*" tables. The average execution time was 447 milliseconds. Moreover, we also tested the insertion of huge volume of data by sending 100K, 200K, 500K and 1M records and we simple noticed a linear increase of execution time.

2.2. Source code quality and verification

In this section, we relied on SonarQube [1], a software tool for automatic source code review to:

- Perform static code analysis
- Maintain code quality
- Detect bugs inside the source code
- Detect security vulnerabilities

The tool provides reports that include complexity, comments and duplicated code. Moreover, it operates according to a set of rules that incorporates coding standards, coding cases that downgrades the computing performance, security vulnerabilities, etc. SonarQube has been used to verify the quality of the web application. More precisely, the source code has been tested thoroughly with the tool and the report generated by SonarQube is presented below.

The results attested that the three tools of the project have no bugs, vulnerabilities, security hotspots and code smells, as shown below. The source codes received an "A" grade for:

- Reliability
- Security
- Security Review (expected for the mobile app, due to the permissions needed by the service)
- Maintainability

2.2.1. Mobile assistant

SonarQube tool has provided the following outcomes for the case of the mobile application, developed in Flutter/Dart:

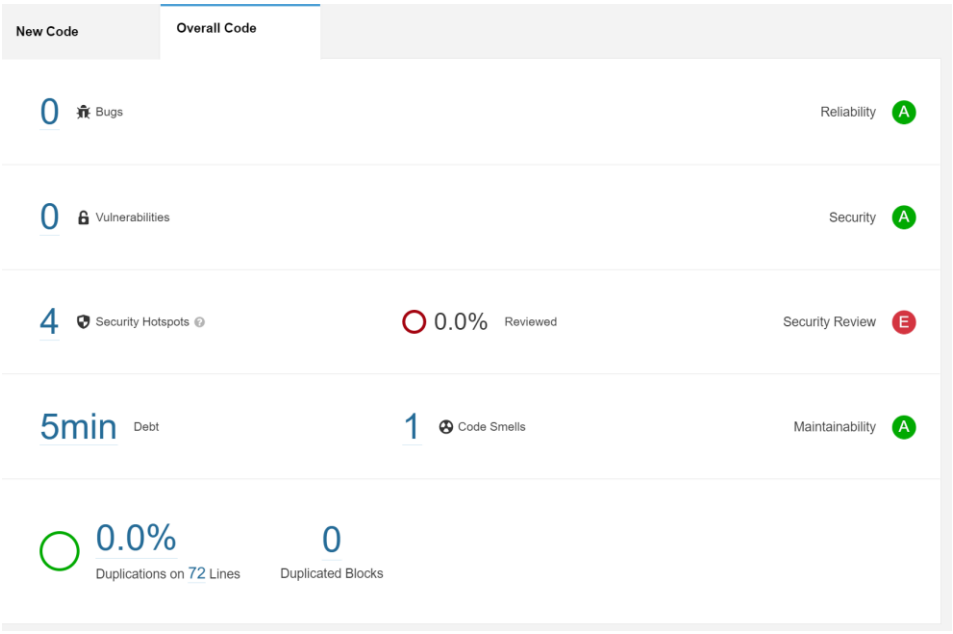

Figure 4: SonarQube results of the mobile application

There are 4 security hotspots in the analysis, however they are needed in the application, as the permissions are needed for the application in order to collect the data.

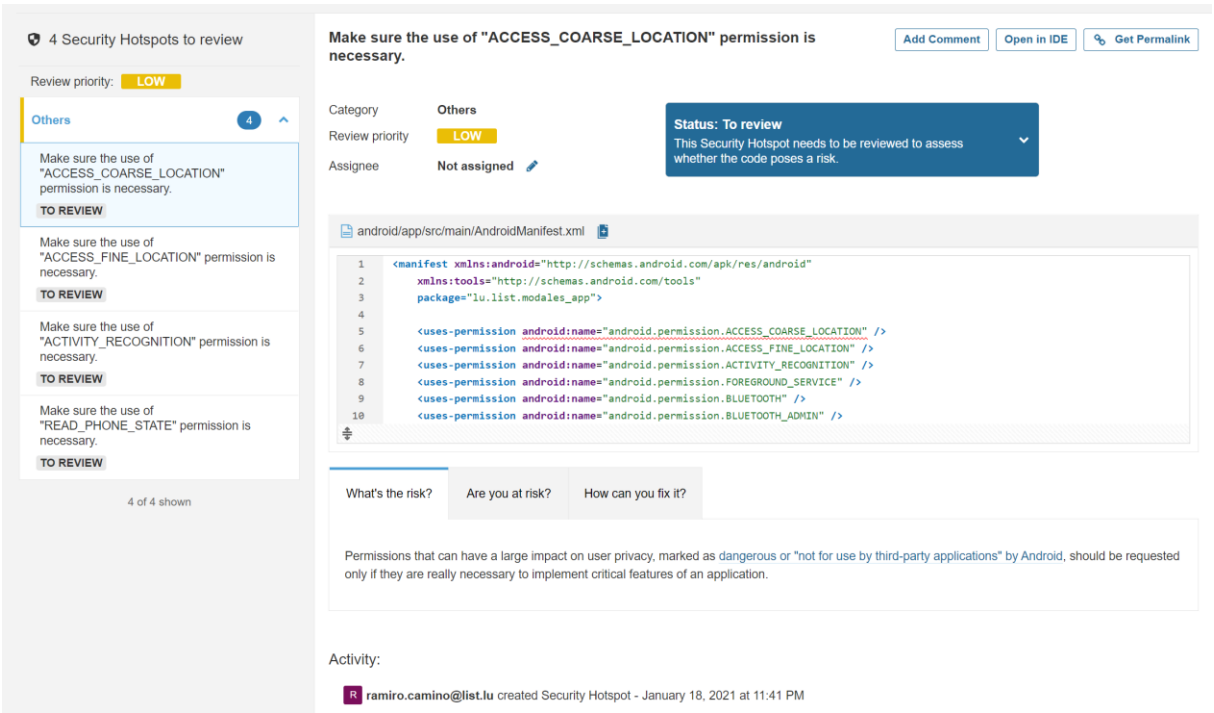

Figure 5: Security hotspots of the mobile application

The results are therefore fully in line with the initial expectations.

2.2.2. Internal reporting platform for trial site and data management

The internal reporting platform met all conditions with SonarQube, as reported below.

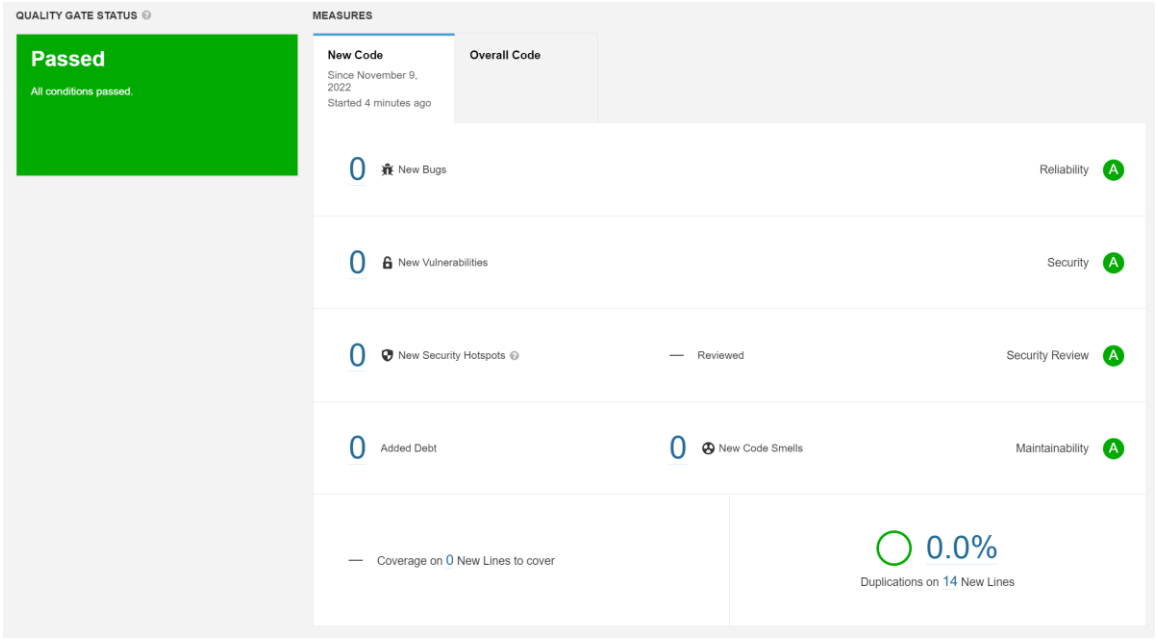

Figure 6: SonarQube results on the internal reporting platform

2.2.3. Dashboard web application

As for the previous applications, the web application passed all conditions with SonarQube, as shown in the figure below.

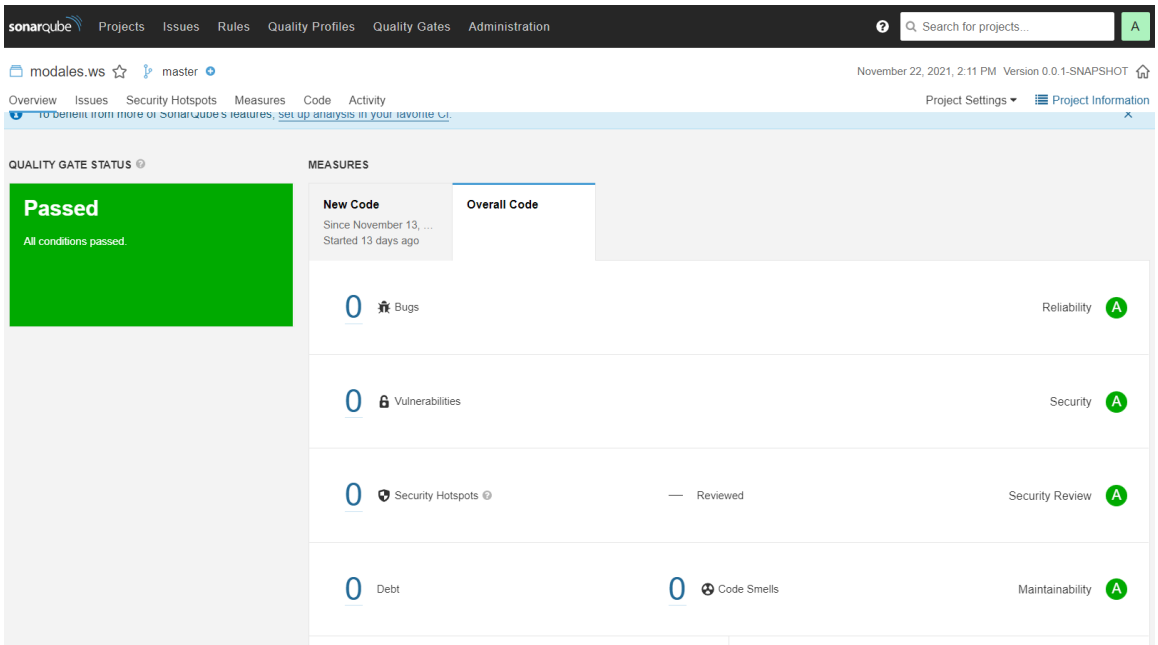

Figure 7: SonarQube results on the dashboard web application

Unit testing is a software testing method that is used to verify that the code performs well and behaves as intended. The back-end software of the dashboard web application has been built with JAVA programming language. The back-end has been implemented as RESTful web services that serves the User Interface through remote procedure calls. We used JUnit [2], which is an open-source unit testing framework for JAVA, in order to check the functionality of the web services and ensure that the execution corresponds to the design and the requirements. The table below presents the input received for each function, the output and the response status.

Table 5: Detailed results of each unit test of dashboard API

Unit Test		Description Dashboard Web application			
Date	**Function**	**Definition**	**Input**	**Output**	**Response. status**

25/11/2021	getUsers	Returns a json string array containing an array with users' data	No Input	[
				{
				"user_id": 1,
				"firstname": "Nikos",
				"lastname": "Dimokas",
				"email":
				"dimokas@certh.gr",
				"authority_id": 1,
				"authorityname":
				"CERTH",
				"userrole_id": 1,
				"rolename": "admin"
				}
				]
					HTTP 200

25/11/2021	getUserValues	Returns a json string containing the values about different user roles and different authorities' names	No Input	{
				"userrole": [
				{ "userrole_id": 1,
				"name": "admin" },
				{ "userrole_id": 2,
				"name": "registrar" },
				{ "userrole_id": 3,
				"name": "user" },
				{ "userrole_id": 4,
				"name":
				"registereduser" }],
				"authority": [
				{ "authority_id": 1,
				
				"authority_name":"CERTH"
				},
				{ "authority_id": 2,
				"authority_name":
				"Test" },
				{ "authority_id": 999,
				"authority_name":
				"Unknown" }]
				}
					HTTP 200

25/11/2021	addUser	Add user data	{ "firstname": "Nikos", "lastname": "Peios", "email": "peios@gmail.com",	No output data	HTTP 200

Unit Test		Description Dashboard Web application			
			"username": "peios", "password": "cGVpb3M=", "authority_id": 1, "userrole_id": 3 }		

25/11/2021	addUser	Add user data	{
			"firstname": "Nikos",
			"lastname": "Peios",
			"email":
			"peios@gmail.com",
			"username": "peios",
			"password":
			"cGVpb3M=",
			"authority_id": 1,
			"userrole_id": 3
			}
				No output data	HTTP 400 (Since the user already exist)

25/11/2021	addUser	Add user data	{
			"lastname": "Papas",
			"email":
			"papas@gmail.com",
			"username": "papas",
			"password":
			"cGVpb3M=",
			"authority_id": 1,
			"userrole_id": 3
			}
				No output data	HTTP 200

25/11/2021	addUser	Add user data. The information about username is mandatory. An HTTP 400 response will be returned.	{
			"firstname": "foo",
			"lastname": "foo",
			"email":
			"foo@email.com",
			"password":
			"cGVpb3M=",
			"authority_id": 1,
			"userrole_id": 3
			}
				No output data	HTTP 400 (Since the username field does not exist)

25/11/2021	addUser	Add user data. The information about password is mandatory. An HTTP 400 response will be returned.	{
			"firstname": "Nikos",
			"lastname": "Peios",
			"email":
			"peios@gmail.com",
			"username": "peios",
			"authority_id": 1,
			"userrole_id": 3
			}
				No output data	HTTP 400 (Since the password field does not exist)

25/11/2021	addUser	Add user data. The information about authority_id is mandatory. An HTTP 400 response will be returned.	{
			"firstname": "Nikos",
			"lastname": "Peios",
			"email":
			"peios@gmail.com",
			"username": "peios",
			"password":
			"cGVpb3M=",
			"userrole_id": 3
			}
				No output data	HTTP 400 (Since the authority_id field does not exist)

25/11/2021	addUser	Add user data. The information about	{ "firstname": "Nikos",	No output data	HTTP 400 (Since the

Unit Test		Description Dashboard Web application			
		userrole_id is mandatory. An HTTP 400 response will be returned.	"lastname": "Peios", "email": "peios@gmail.com", "username": "peios", "password": "cGVpb3M=", "authority_id": 1 }		userrole_id field does not exist)
25/11/2021	addUser	Add user data	{ "firstname": "Nikos", "lastname": "Peios", "email": "peios@gmail.com", "username": "peios", "password": "cGVpb3M=", "authority_id": 15, "userrole_id": 3 }	No output data	HTTP 400 (Since the authority_id field has not a valid value)
25/11/2021	addUser	Add user data	{ "firstname": "Nikos", "lastname": "Peios", "email": "peios@gmail.com", "username": "peios", "password": "cGVpb3M=", "authority_id": 1, "userrole_id": 30 }	No output data	HTTP 400 (Since the userrole_id field has not a valid value)
25/11/2021	updateUser	Update user data	{ "users_id": 826, "firstname": "F", "lastname": "L", "email": "l@gmail.com", "username": "l", "password": "cGVpb3M=", "authority_id": 1, "userrole_id": 3 }	No output data	HTTP 200
25/11/2021	updateUser	Update user data	{ "firstname": "F", "lastname": "L", "email": "l@gmail.com", "username": "l", "password": "cGVpb3M=", "authority_id": 1, "userrole_id": 3 }	No output data	HTTP 400 (Since the users_id field does not exist)
25/11/2021	updateUser	Update user data	{ "users_id": 826, "lastname": "L", "email": "l@gmail.com",	No output data	HTTP 200

Unit Test		Description Dashboard Web application			
			"username": "l", "password": "cGVpb3M=", "authority_id": 1, "userrole_id": 3 }		

25/11/2021	updateUser	Update user data	{
			"users_id": 826,
			"firstname": "F",
			"email":
			"l@gmail.com",
			"username": "l",
			"password":
			"cGVpb3M=",
			"authority_id": 1,
			"userrole_id": 3
			}
				No output data	HTTP 200

25/11/2021	updateUser	Update user data	{
			"users_id": 826,
			"firstname": "F",
			"lastname": "L",
			"username": "l",
			"password":
			"cGVpb3M=",
			"authority_id": 1,
			"userrole_id": 3
			}
				No output data	HTTP 200

25/11/2021	updateUser	Update user data	{
			"users_id": 826,
			"firstname": "F",
			"lastname": "L",
			"email":
			"l@gmail.com",
			"password":
			"cGVpb3M=",
			"authority_id": 1,
			"userrole_id": 3
			}
				No output data	HTTP 400
					(Since the username does not exist)

25/11/2021	updateUser	Update user data	{
			"users_id": 826,
			"firstname": "F",
			"lastname": "L",
			"email":
			"l@gmail.com",
			"username": "l",
			"authority_id": 1,
			"userrole_id": 3
			}
				No output data	HTTP 400
					(Since the password does not exist)

25/11/2021	updateUser	Update user data	{ "users_id": 826, "firstname": "F", "lastname": "L", "email": "l@gmail.com", "username": "l", "password": "cGVpb3M=", "userrole_id": 3	No output data	HTTP 400 (Since the authority_id field does not exist)

Unit Test		Description Dashboard Web application			
			}		

25/11/2021	updateUser	Update user data	{
			"users_id": 826,
			"firstname": "F",
			"lastname": "L",
			"email":
			"l@gmail.com",
			"username": "l",
			"password":
			"cGVpb3M=",
			"authority_id": 1
			}
				No output data	HTTP 400 (Since the userrole_id field does not exist)

25/11/2021	updateUser	Update user data	{
			"users_id": 826,
			"firstname": "F",
			"lastname": "L",
			"email":
			"l@gmail.com",
			"username": "l",
			"password":
			"cGVpb3M=",
			"authority_id": 15,
			"userrole_id": 3
			}
				No output data	HTTP 400 (Since the authority_id field has not a valid value)

25/11/2021	updateUser	Update user data	{
			"users_id": 826,
			"firstname": "F",
			"lastname": "L",
			"email":
			"l@gmail.com",
			"username": "l",
			"password":
			"cGVpb3M=",
			"authority_id": 1,
			"userrole_id": 6
			}
				No output data	HTTP 400 (Since the userrole_id field has not a valid value)

25/11/2021	updateUser	Update user data	{
			"users_id": 879,
			"firstname": "F",
			"lastname": "L",
			"email":
			"l@gmail.com",
			"username": "l",
			"password":
			"cGVpb3M=",
			"authority_id": 1,
			"userrole_id": 6
			}
				No output data	HTTP 400 (Since the users_id field has not a valid value)

25/11/2021	removeUser	Remove user	{
			"users_id": 823
			}
					HTTP 200

25/11/2021	removeUser	Remove user	{
			"users_id": 879
			}
					HTTP 400 (Since the users_id field has not a valid value)

2.3. Integration Testing

2.3.1. Mobile application

The figure below shows the different elements that come with the mobile application. These components are detailed in Deliverables D5.2 and D5.3 of the project.

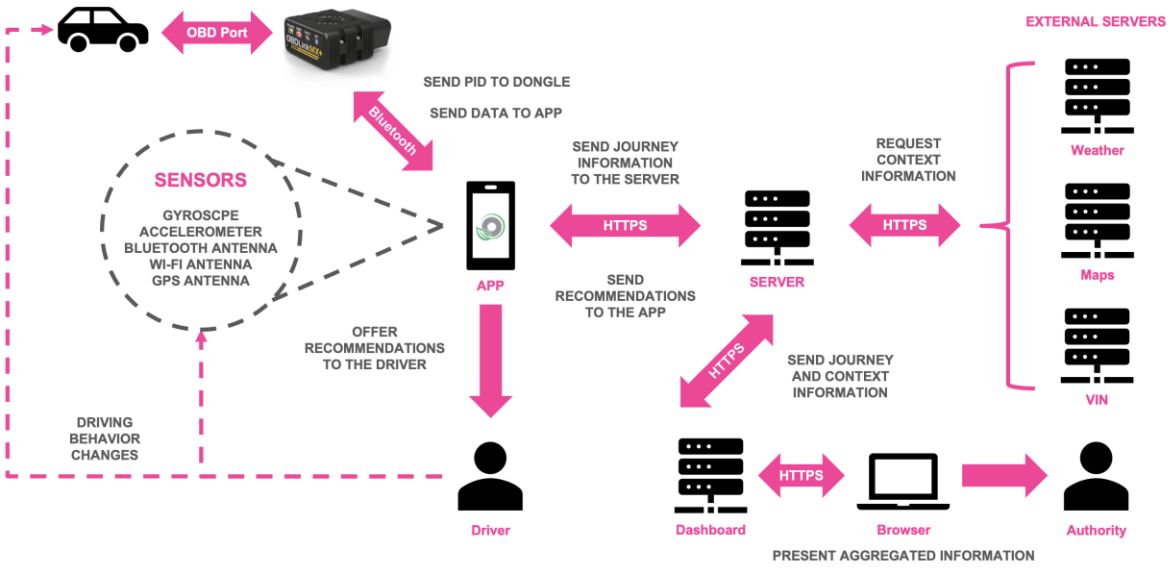

Figure 8: Overall architecture of the mobile app and its components

On the one hand, the mobile application collects data from two main sources:

- The phone's sensors, allowing for accelerometer, position, activity data, etc.
- A Bluetooth-connected OBD dongle, allowing access to real-time vehicle information. The developments made for the OBD part were tested with an OBD emulator, provided by Freematics, and allowing to reproduce a complete system in the laboratory, so to test and read OBD PIDs.

On the other hand, the application regularly sends this information to a server located at LIST, which performs filtering and processing operations before storing the data. Two external services are used:

- A data augmentation service provided by *Motion-S*, which allows to replace GPS data with useful contextual information (full list in D5.3), in order to remove the GPS positions afterwards.
- A service to convert the Vehicle Identification Number (VIN) into useful information (vehicle, make, year, etc.), provided by *Vincario*.

2.3.2. Synchronisation between mobile app and Server

Several internal tests have been performed to check the connectivity between the mobile app and the server to send journeys. The app can send data to the server either manually or automatically when there is Wi-Fi or mobile data connection available in a given interval of time. For our test, we have chosen to send data manually since it is the most imminent choice for testing.

From the application, we shall make some test runs, to collect the data in an internal file, and then we can synchronise the data to the server.

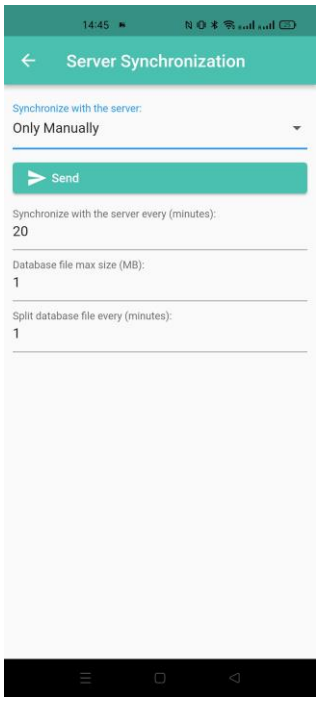

Figure 9: Server synchronisation options on the mobile app

As for the server side, we get the following message when everything goes as expected:

```
INFO:werkzeug:192.168.178.69 - - [18/Nov/2022 15:17:46] "POST /api/system_logs HTTP/1.1" 200 -
INFO:werkzeug:192.168.178.69 - - [18/Nov/2022 15:18:15] "POST /api/sensor_logs HTTP/1.1" 200 -
INFO:werkzeug:192.168.178.69 - - [18/Nov/2022 15:18:15] "POST /api/system_logs HTTP/1.1" 200 -
```

Figure 10: Server synchronisation confirmation on the server

Afterwards the data is saved in the server in a PostgresSQL database, as detailed above.

2.3.3. Dashboard web application

The integration testing of the dashboard web application involves turning on the servers and performing a successful login. This way, we can ensure that the application's deployment performs well and the integration of the website with the web services and the database is realised.

The web services have been implemented with RESTful web services using the Jersey REST implementation for the JAVA programming language. The deployment of the web services requires an HTTP web server that can execute JAVA code. We used the Apache Tomcat web server [3] which is one of the most popular JAVA-based web servers. Thus, the web services have been deployed on Tomcat web server as depicted in the figure below and the server started running.

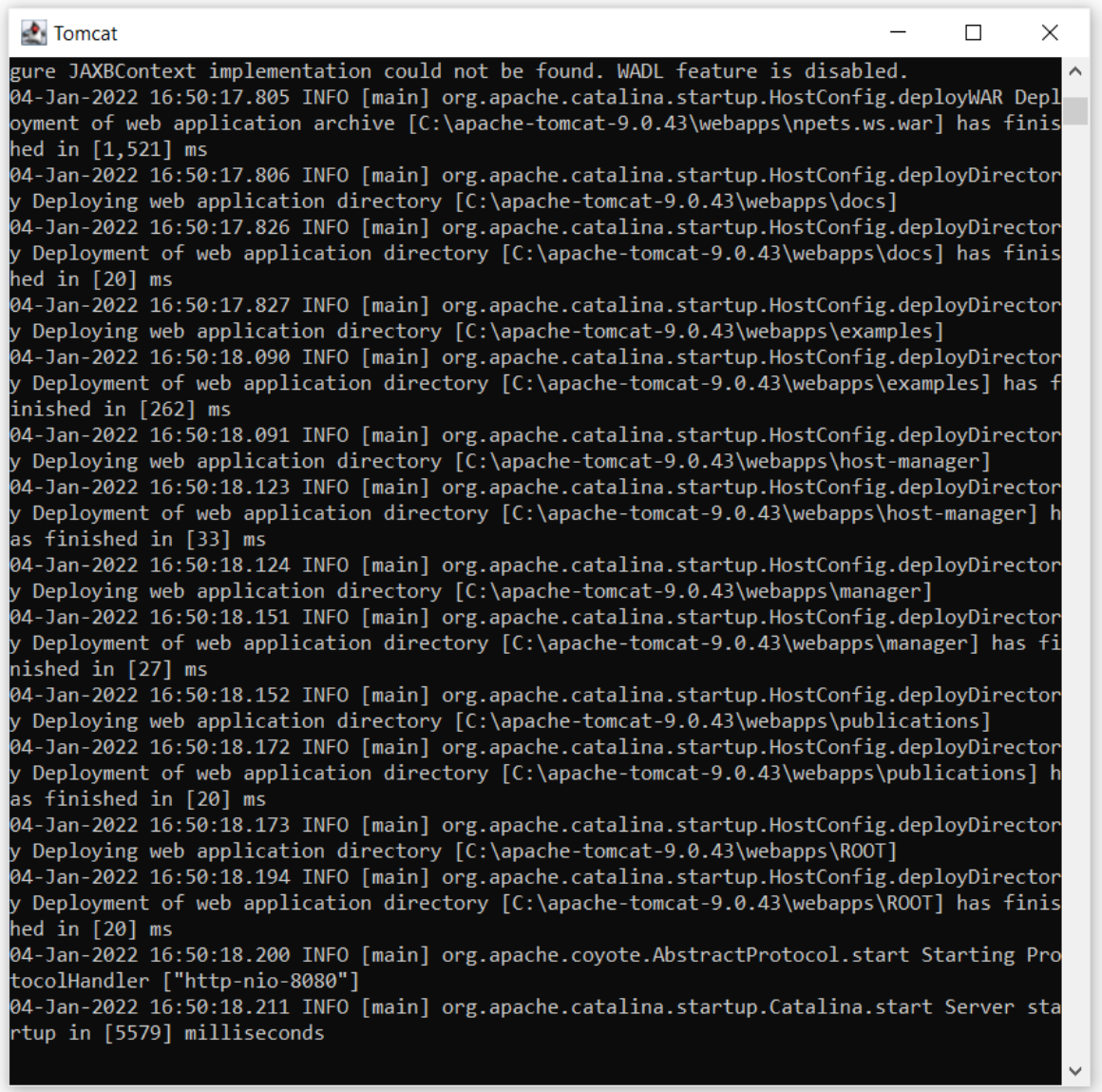

Figure 11: Tomcat web server running

The User Interface implemented with HTML, PHP, Javascript and CSS have been deployed in Apache HTTP web server [4]. After that, the server started running as depicted in the figure below.

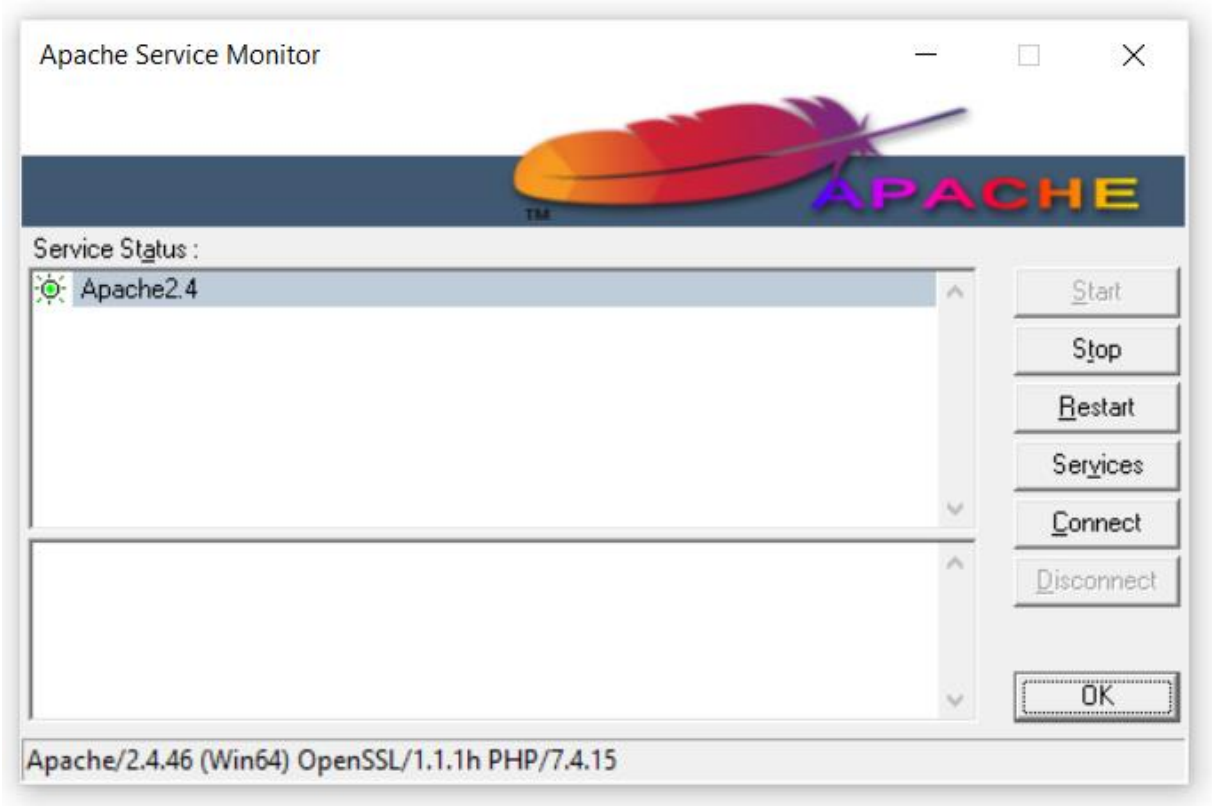

Figure 12: Apache web server running

After the two web servers started running, the website is accessible through a web browser. The user fills in the username and password and clicks on "Login" button as depicted below. Then, the dashboard web page is loaded (Figure 14).

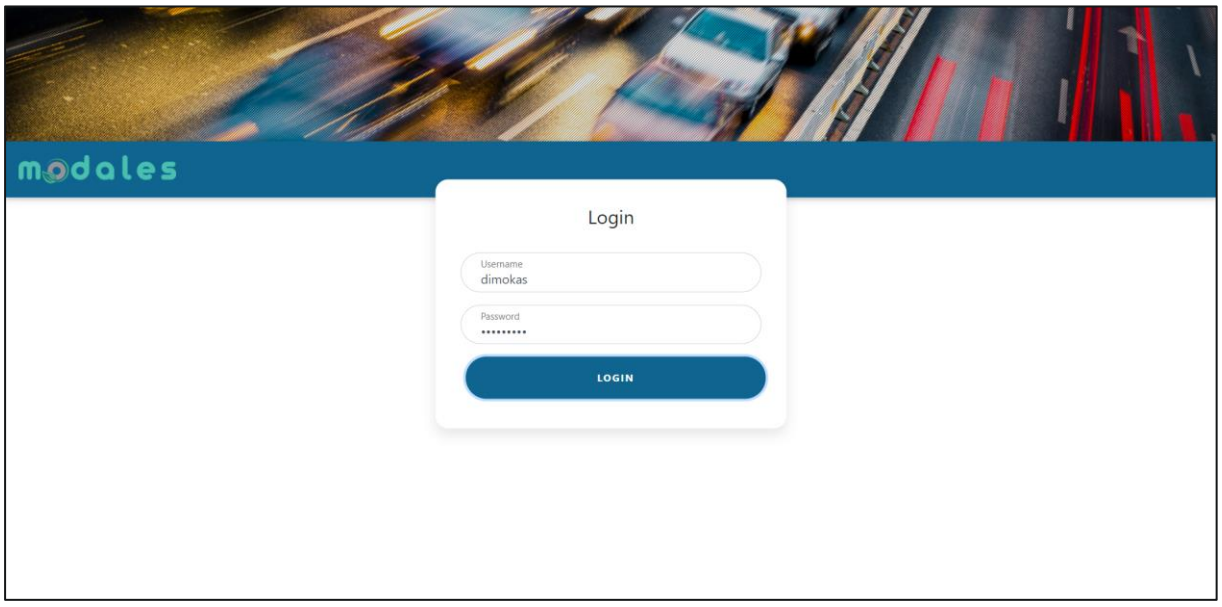

Figure 13: Login page for internal reporting platform

Figure 14: Internal reporting platform Dashboard web page

The Dashboard of the Tomcat web server also presents the successful login of the user. It is important to notice here, that the login operation (authentication) includes a query to the database in order to verify whether the user is registered or not. Thus, the login operation confirms the successful system integration.

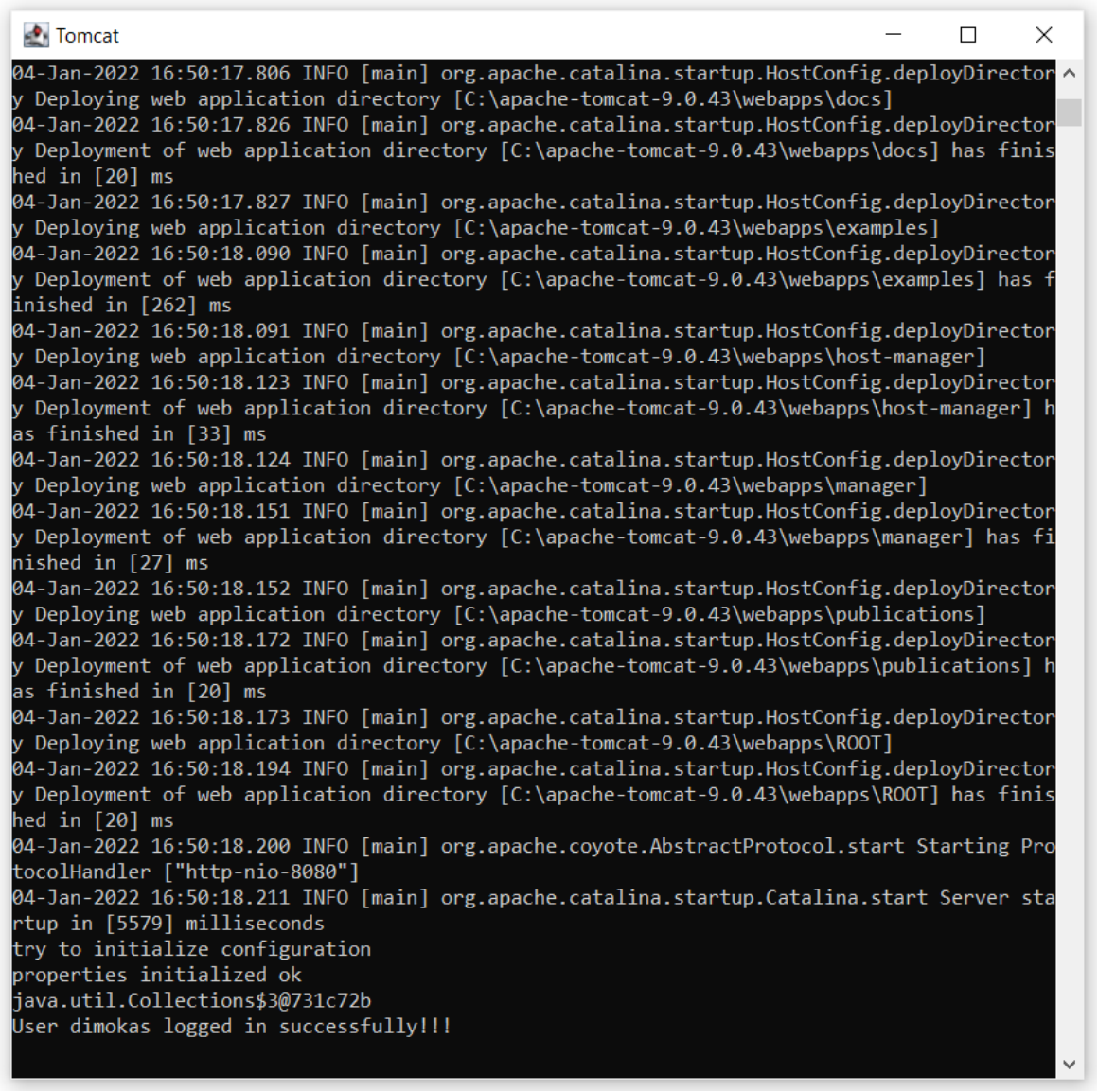

Figure 15: Successful login operation

2.4. System Testing

The system testing of the three tools includes:

- Testing all links and buttons and the related actions (for example submitting a form should also change the corresponding data table).
- Testing different user inputs.

We do so for each of the tools below.

2.4.1. Mobile application

All the links have been tested, be it by internal users (consortium) or external test users all along the development of the app. Below is a summary of the main views offered by the app.

The first menu to appear in the application is the main menu to start a journey while accessing the OBD dongle communication and the selected vehicle.

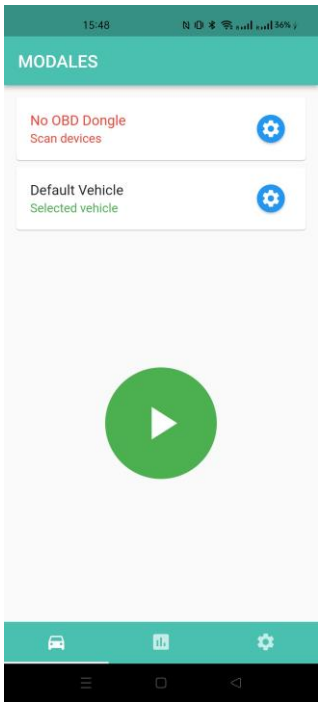

Figure 16: Homepage of the MODALES app

When we start a journey, the next screen will show up with the active driving recommendations along the journey.

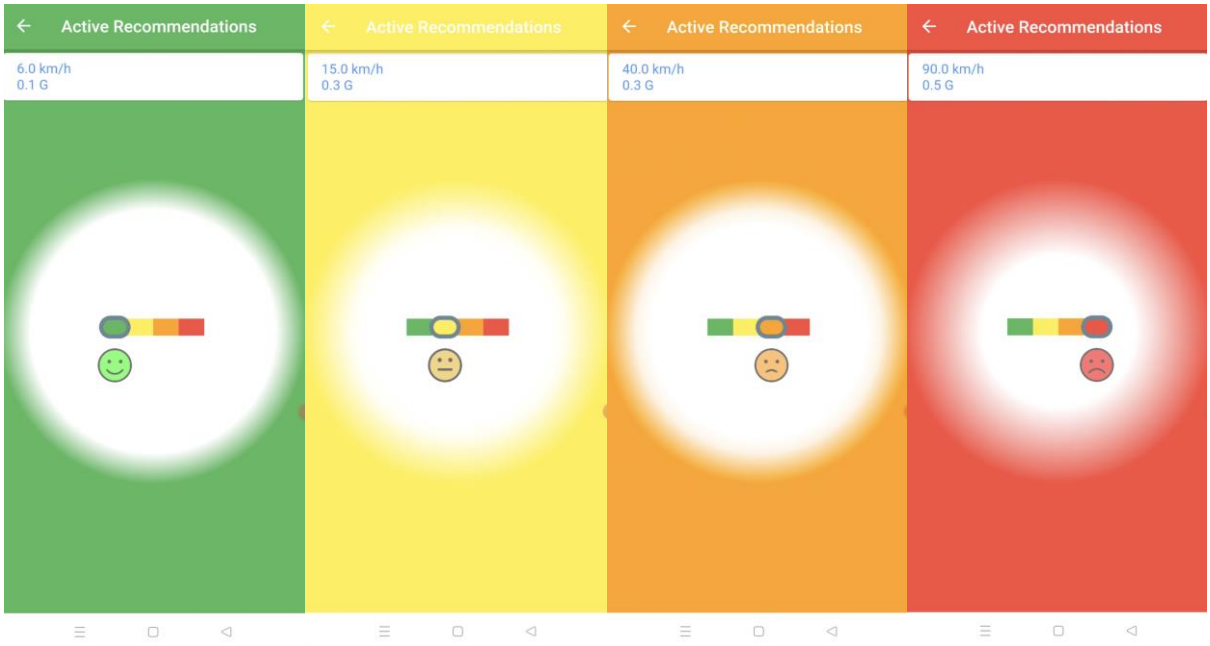

Figure 17: Active recommendation system in the MODALES app

The middle bar button opens the menu with the list of passive driving recommendations.

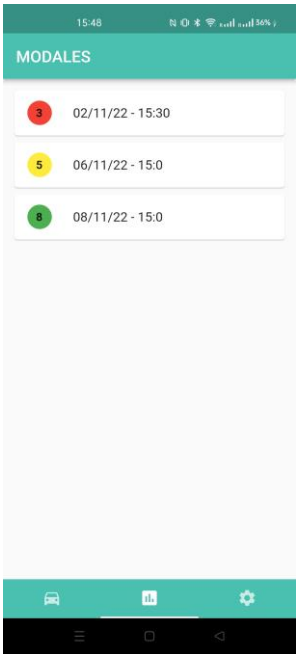

Figure 18: Overview of journeys in the MODALES app

After selecting one recommendation in the list, the information about the particular journey will be highlighted.

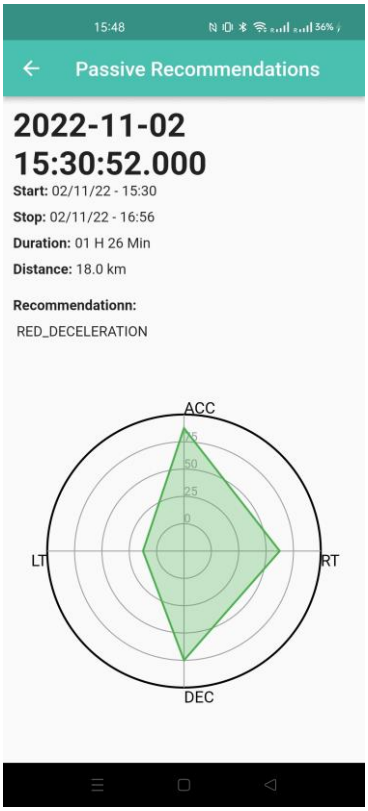

Figure 19: Overview of passive recommendations in the MODALES app

The right bar button shows the settings of the application:

- User profile information
- OBD dongle connection
- Garage with all the vehicles registered and the selected one
- Data collection
- Server synchronisation
- Clear stored data

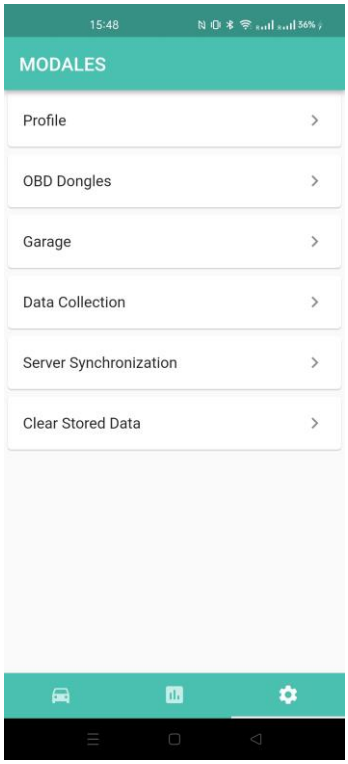

Figure 20: Parameters of the MODALES app

2.4.2. Internal reporting system

The internal reporting system was not planned at the beginning of the project but was created with the intention to facilitate the management of trial sites and the sharing of data. Like the app, all the links have been tested successfully. Below is a summary of the different pages that compose the tool.

The first screen that the user can see is for logging in. Only administrators of the platform can create an account. Each trial site user has also created an account.

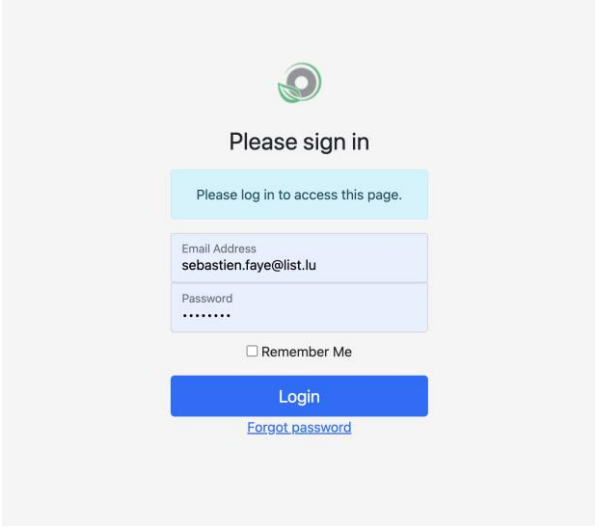

Figure 21: Log-in screen of the internal reporting platform for MODALES administrators

After log-in, the user has access to various functionalities as listed below.

Firstly, the user can get access to database dumps, belonging to his/her affected trial sites (all datasets for administrator users).

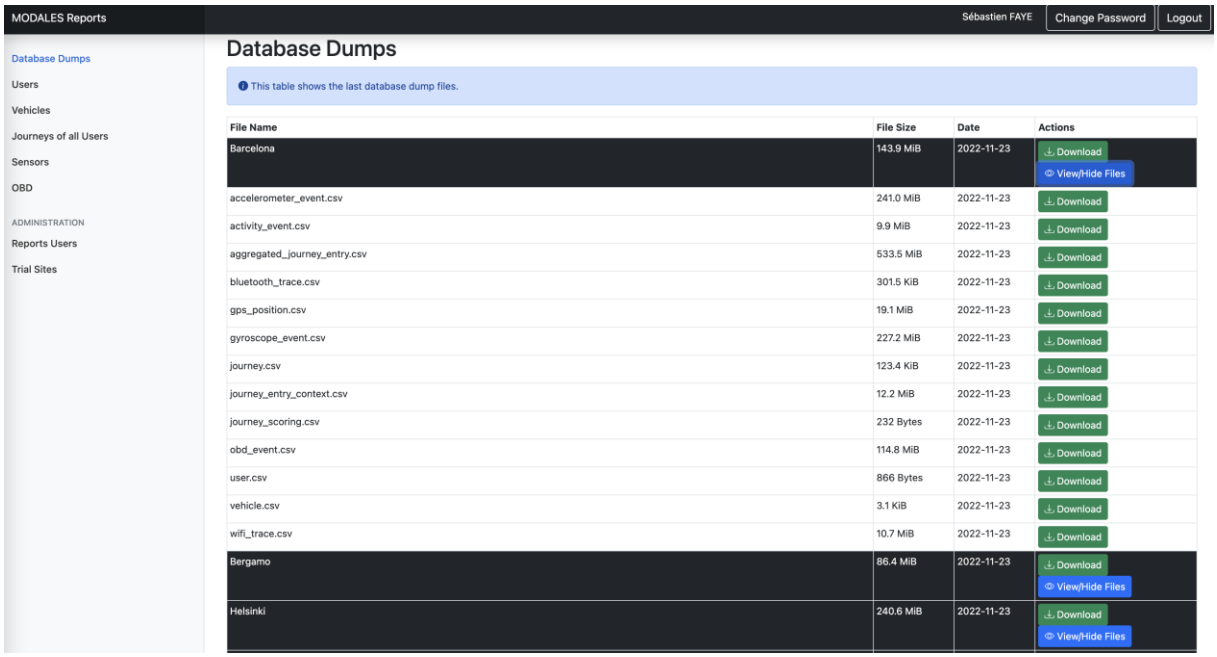

Figure 22: Database backup screen

Secondly, the user has access to statistical data, to follow and analyse the participation of users (drivers) on his/her trial sites. This includes information about: the users, the vehicles, the journeys of all users, the sensor data collected and the OBD data collected.

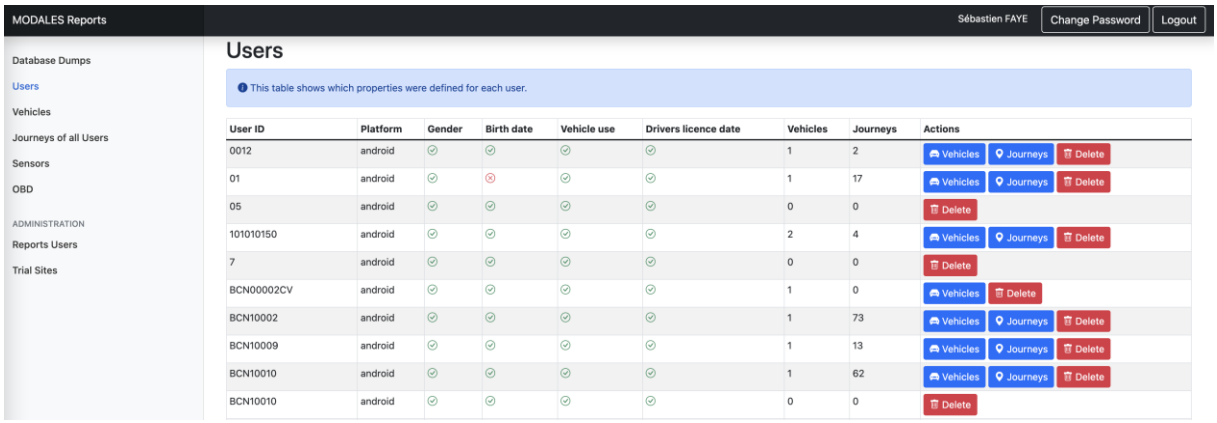

Figure 23: Overview of users collecting data in the platform

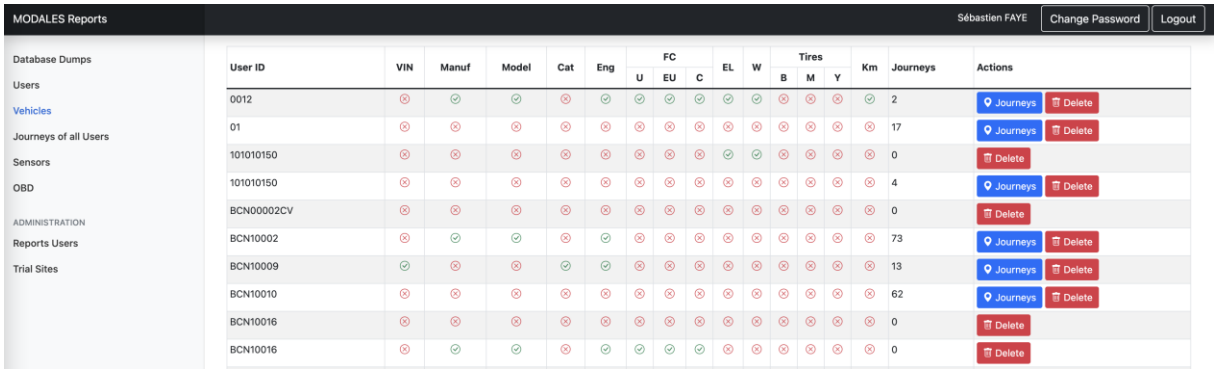

Figure 24: Overview of vehicles

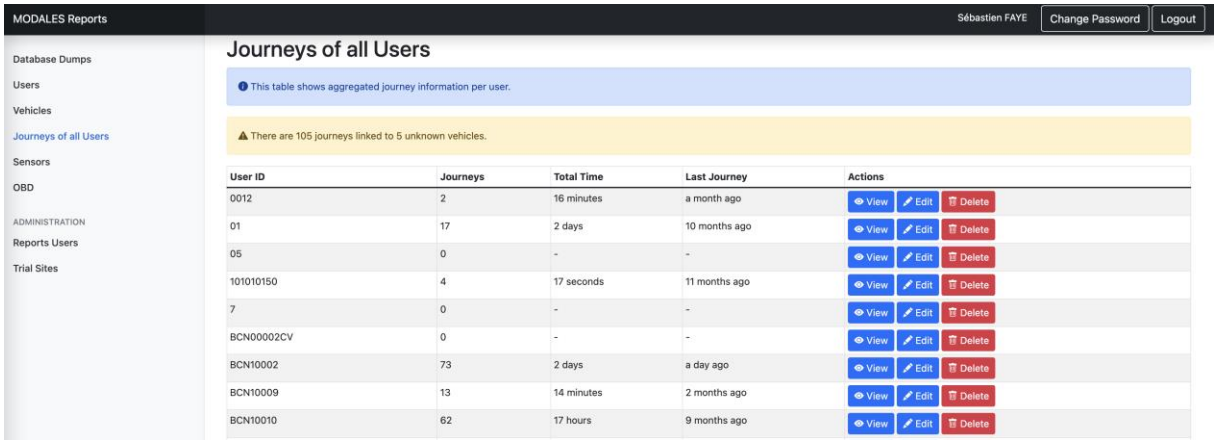

Figure 25: Overview of all users' journeys

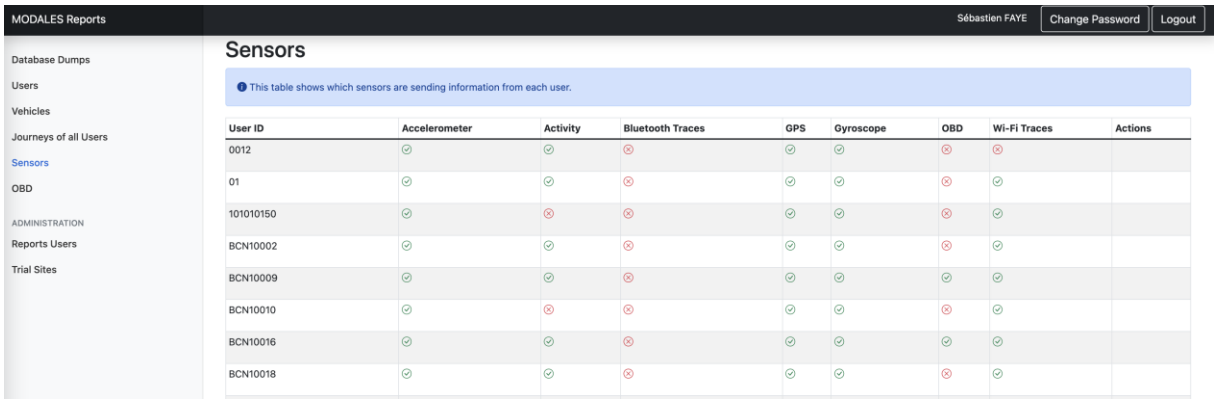

Figure 26: Overview of sensor data collected

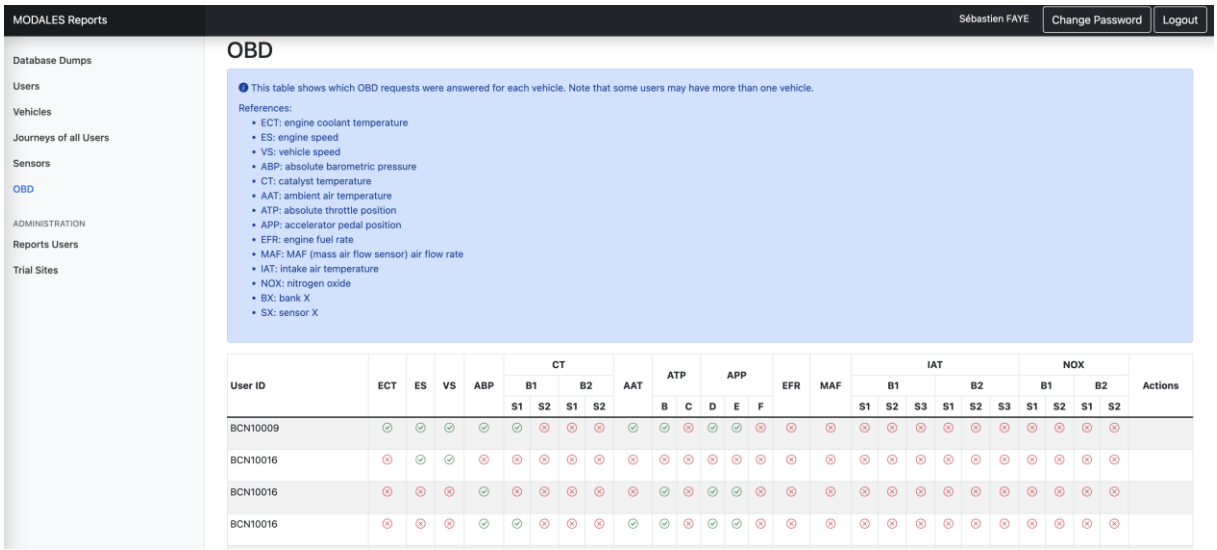

Figure 27: Overview of OBD data

The administrators have the ability to manage the users that get access to the platform, including their affected sites, and are also able to define new trial sites on the fly, based on a code (defined in the data management plan of the project).

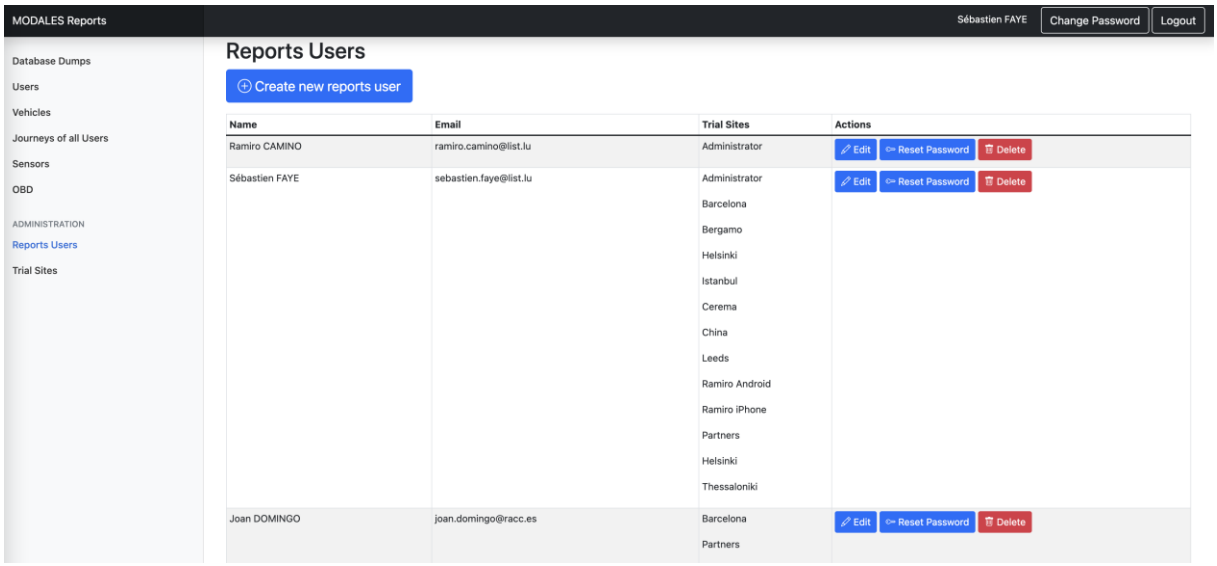

Figure 28: Administrator panel: manage user access to the reporting platform

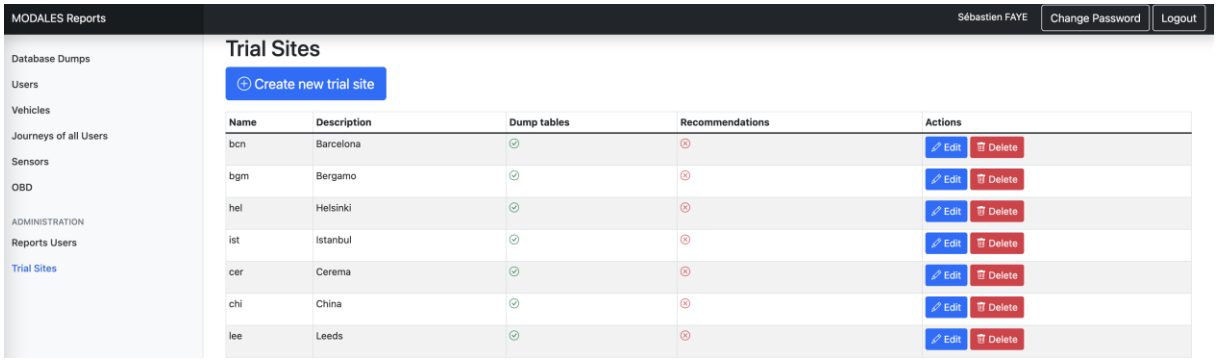

Figure 29: Administrator panel: manage the trial sites of the reporting platform

2.4.3. Dashboard web application

After the successful login (already presented in Figure 13 and Figure 14), the user can click on “Region” button, the “Users” button or the “Logout” button. We click on “Region” button and the region web page is successfully loaded (Figure 30). The same happens when we click on the “Users” button (Figure 31).

Figure 30: Internal reporting platform: Region web page

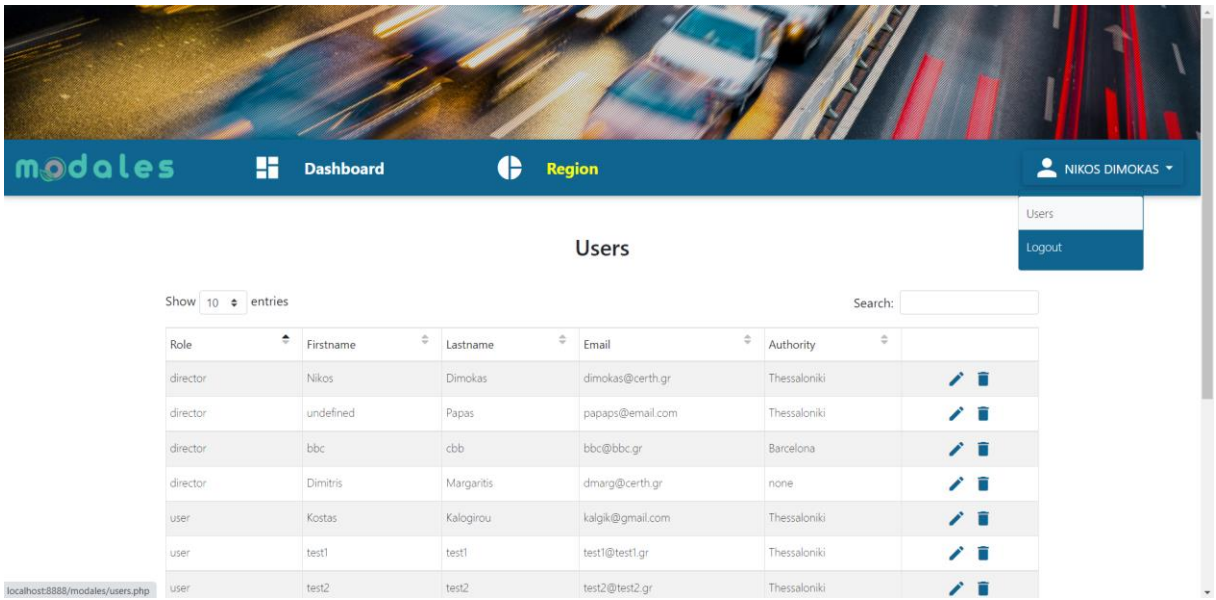

Figure 31: Internal reporting platform: Users web page

The “Users” web page includes a button for creating a new user, an image button for updating user’s data and an image button for deleting a user. We click on “New User” button and the web form is successfully loaded. The web form for updating the data is loaded after pressing on “pencil” button as presented in Figure 32, while the user’s data are successfully removed when we press on the “bin” button.

The web form for inserting a new user or updating an existing one is the same. The form includes two buttons. The first button is the “Register” button and the second one is the “Cancel” button. Both buttons have been tested and worked properly. For example, we tried to modify the name of the first user from to Nikos to Nikolaos. After pressing on “Register” button the name of the user

successfully changed (Figure 33). During system testing, we tested different values (alphanumeric, numeric and text) for all fields of the form. The tests proved successful in all cases.

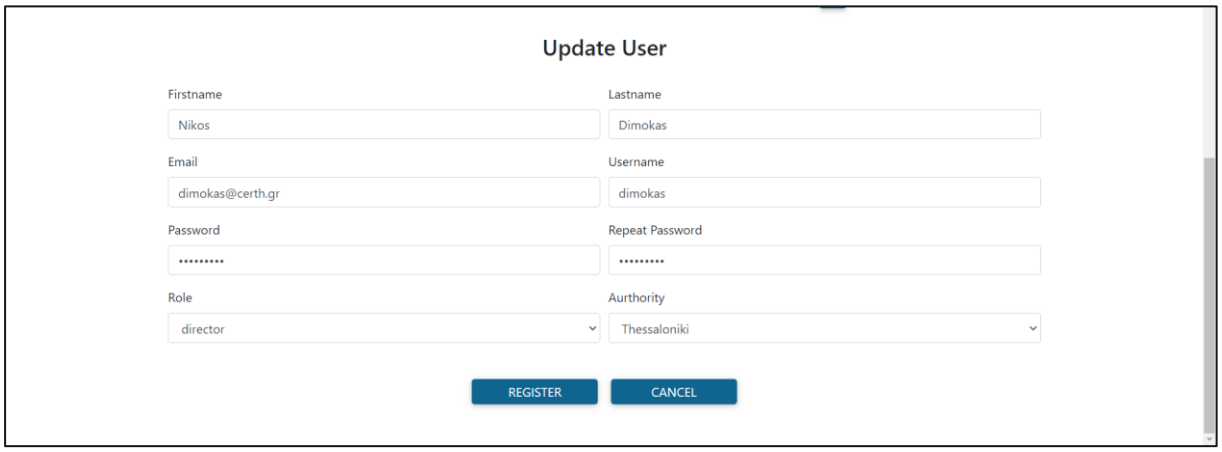

Figure 32: Updating user information

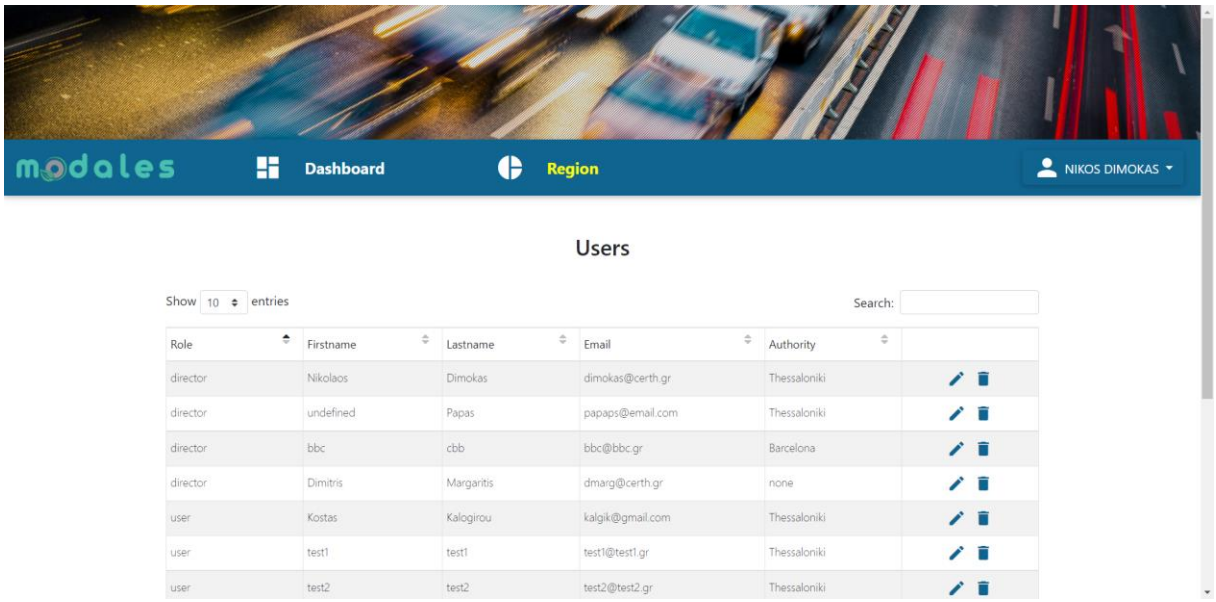

Figure 33: User's data updated successfully

Finally, an authentication error (Figure 34) is presented when the username and/or the password are not valid.

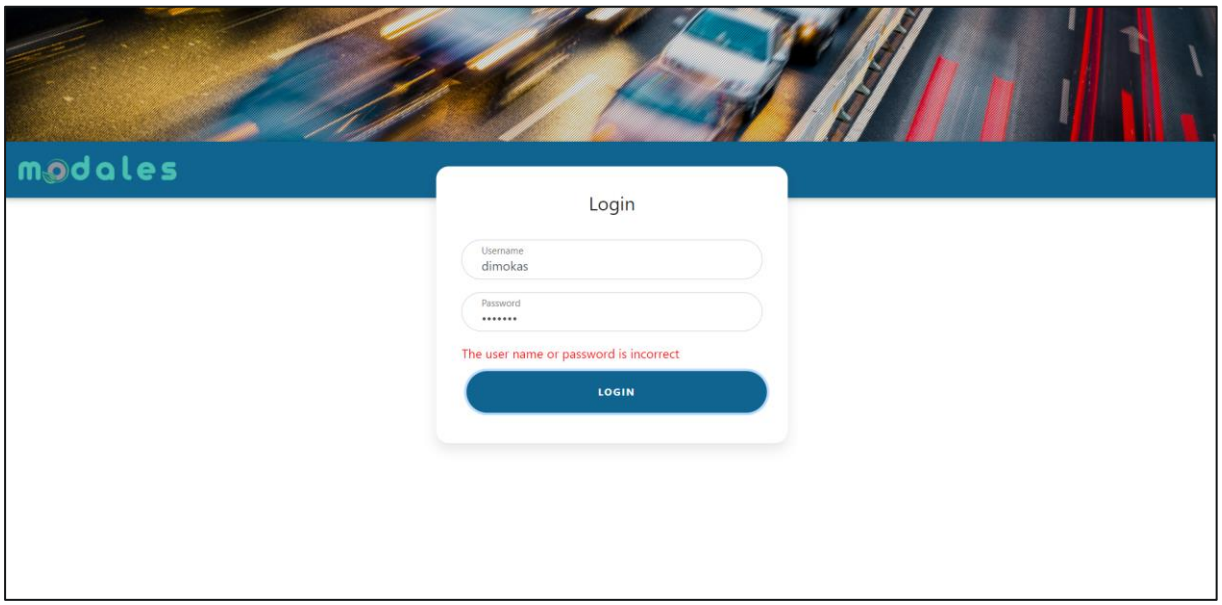

Figure 34: Authentication error

2.5. Performance Testing

2.5.1. Mobile application and server

One of the main performance issues related to the mobile application is the storage of data on the project server. Indeed, collecting data for several months, on up to 300 users, and at very low sampling frequencies (e.g., 100 ms for the accelerometer and gyroscope values) is potentially critical for the server hosting. The data presented here will be detailed and extended in a complementary technical analysis carried out and reported in deliverable D6.3.

This study was done on 9th November 2022. The figures below showcase the statistics and number of journeys, vehicles and driving hours stored on the project server at that time – accumulated all along 2022 following tests and early data collection campaigns.

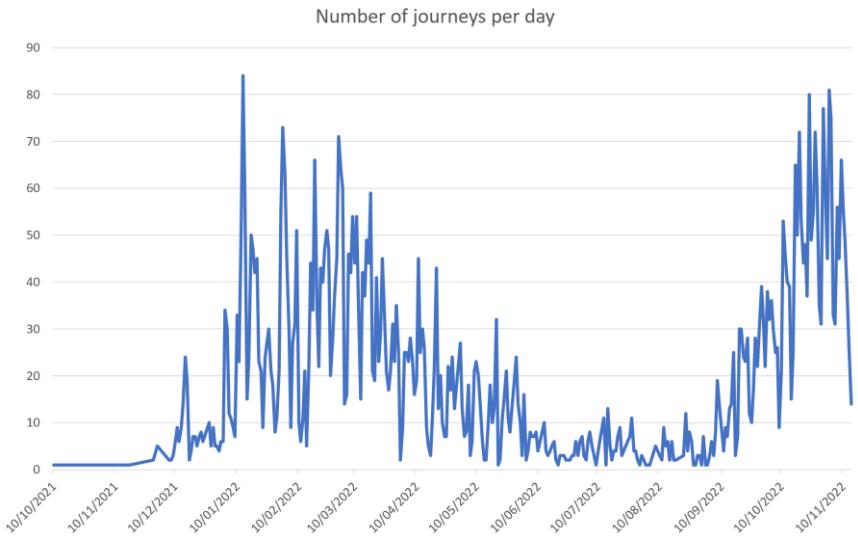

Figure 35: Number of journeys per day, 10/2021 – 11/2022

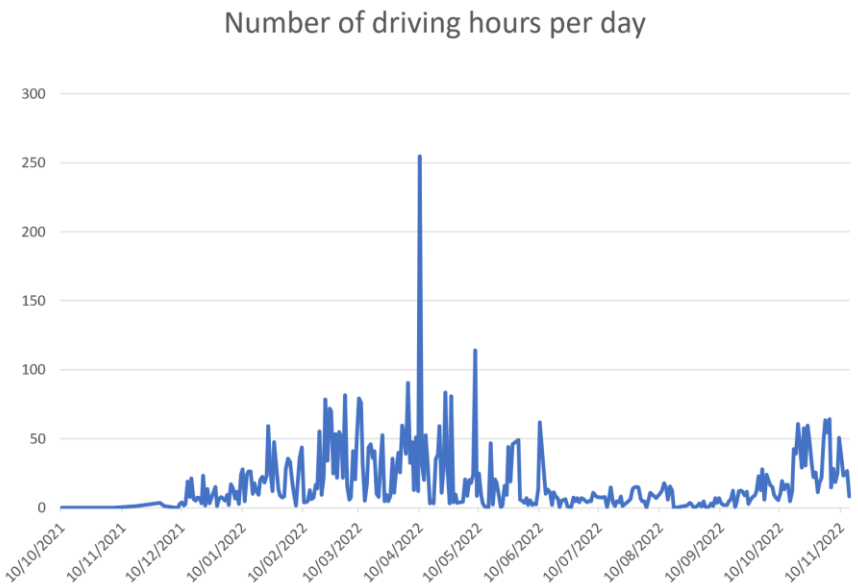

Figure 36: Number of driving hours per day, 10/2021 – 11/2022

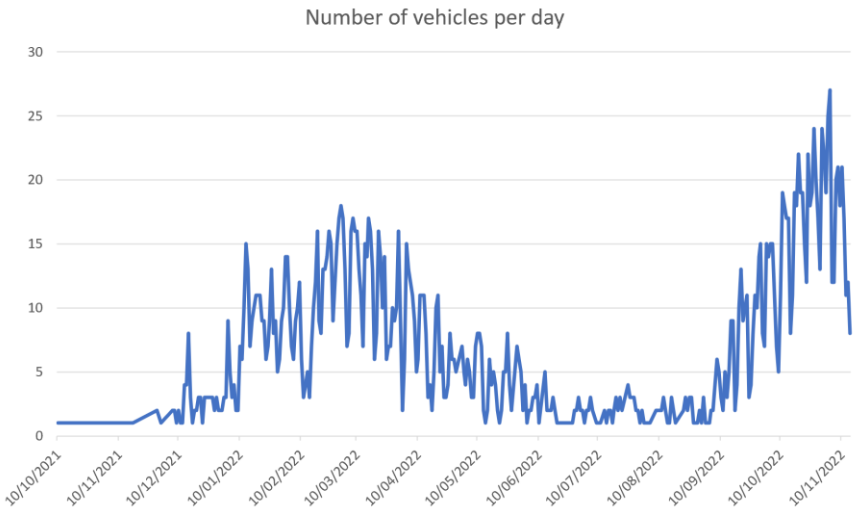

Figure 37: Number of vehicles per day, 10/2021 – 11/2022

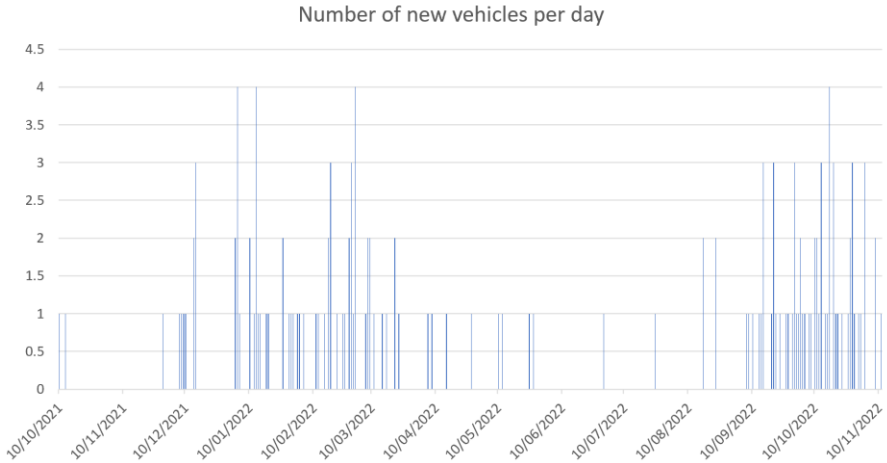

Figure 38: Number of new vehicles per day, 10/2021 – 11/2022

As of 9 November 2022, the data storage represented a total of 86 Gigabytes (GB) of data, including the OS itself. Between 18/10/2022 and 09/11/2022, there was an increase of 29 GB of data storage in the VM hosting the server. This increase includes data in the database itself as well as backups of data and logs. The figure below represents the usage accumulated during this period:

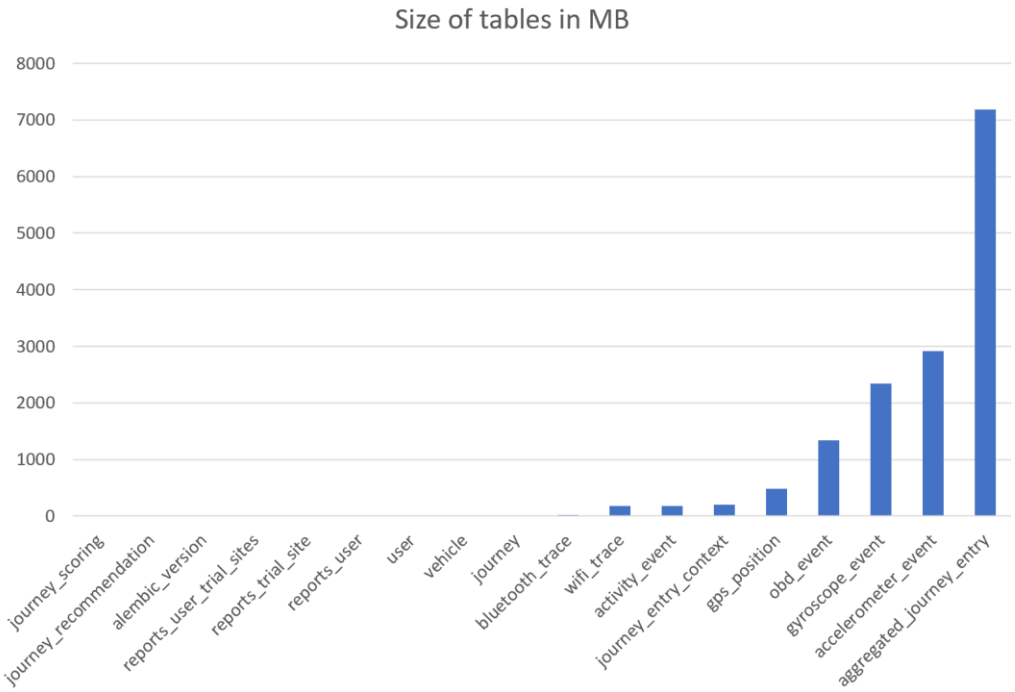

Table	Size in MB
journey_scoring	0.02
journey_recommendation	0.02
alembic_version	0.02
reports_user_trial_sites	0.02
reports_trial_site	0.03
reports_user	0.05
user	0.06
vehicle	0.11
journey	1.46
bluetooth_trace	22.91
wifi_trace	177.01
activity_event	177.19
journey_entry_context	202.15
gps_position	476.54
obd_event	1343.45
gyroscope_event	2345.98
accelerometer_event	2913.25
aggregated_journey_entry	7184.01

Figure 39: Amount of data accumulated between 18/10/2022 and 09/11/2022

2.5.2. Dashboard web application

The performance testing of the dashboard web application includes the computation of the response time of all operations per single request. More precisely, we tested thoroughly the operations by performing many requests (100) per operation and computing the average execution/response time.

The application has eight major operations. The login operation is related to the user's authentication, while the logout operation is executed when the user leaves the application. The

dashboard operation is executed when the dashboard web page is loaded. The dashboard web page includes data that are stored in the database and the creation of visual graphs. Similar to the previous one is the region operation that is executed when the region web page is loaded. Finally, the get users, get user's values, load user data and register user operations are related to the users' section.

The table below presents the operations and the average execution (response) time of each operation.

Table 6: Dashboard web application: Performance testing

Operation	Time (ms)
Login	31
Dashboard	1151
Region	515
Get Users	56
Get User's Values	40
Load User Data in Form	26
Register (Add or Edit) User	37
Logout	23

3. User acceptance procedure and results

3.1. Introduction

This Section is intended to fulfil one of the testing steps described in the DoA and more specifically the "*Usability (User Interface) Testing: assessment for its easy operation, content navigation, etc*".

Usability is the ability for users to complete their tasks within a website or an app. As a task-based assessment, improving upon usability generally relates to simplifying and streamlining the app, while making it more obvious to the user which interactions they need to complete.

The three fundamentals to be considered in such a process are:

- Usability: Making an app easy to use
- UI: Making an app attractive and effective according to users' preferences
- UX: Making users feel positive about an app

3.2. Method

3.2.1. Tools: System Usability Scale (SUS)

The System Usability Scale (SUS) was released by John Brooke in 1986 [5]. It was originally created as a scale for administering after usability tests on systems such as VT100 Terminal ("Green-Screen") applications. SUS is technology independent and has since been tested on hardware, consumer software, websites and smartphone applications. It has become an industry standard with references in over 1300 scientific publications.

SUS is a 10-item questionnaire with 5-response options. The questions are:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

The 5-response option is shown below.

Strongly Disagree 1	2	3	4	Strongly Agree 5
○	○	○	○	○

Figure 40: SUS (System Usability Scale) response format

The participant's scores for each question are converted to a new number, added together and then multiplied by 2.5 to convert the original scores of 0-40 to 0-100. Though the scores are 0-100, these

are not percentages and should be considered only in terms of their percentile ranking. Based on research of previous SUS scores, a SUS score above a 68 would be considered above average.

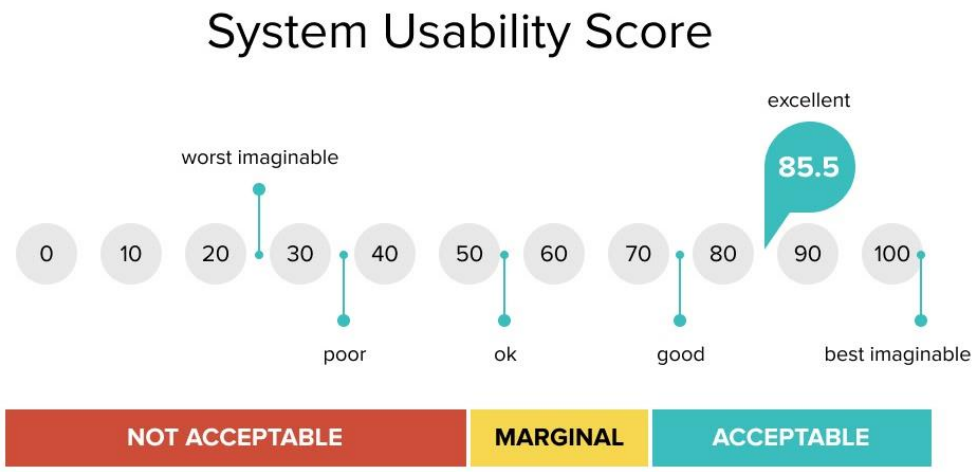

Figure 41: Usability score interpretation [6]

3.2.2. Data collection

An .xlsx file with the 10 questions along with instructions about the SUS was distributed to the trial site leaders. Each site leader interviewed 5 or more participants, in order to have a total sample of sufficient size. The final sample counts 41 participants from all trial sites.

Concerning the participant notification of troubles that have experienced during the app usage, this was (and still is, at the time of writing) an ongoing process. The trial site leader keeps a log file with such issues and informs the app development team (LIST) every week, if any problem occurs and spotted by the local participants.

3.2.3. Data analysis

The SUS responses are converted to a final score as described below:

- The users have ranked each of the 10 above mentioned questions from 1 to 5, based on their level of agreement.
- For each of the odd numbered questions, 1 has been subtracted from the score.
- For each of the even numbered questions, their value has been subtracted from 5.
- These new values, which have been found, should add up to the total score.
- The total score is multiplied by 2.5.

3.3. Results

3.3.1. Acceptance

The final score, derived from the feedback of the 41 participants, is 62, somewhat below the threshold of 68. According to Figure 41, the app is considered from “ok – good”. The result is quite acceptable taking into account that this is not a commercial app version but rather a functional prototype to be used in research activities. Some of the potential reasons for the score are:

- The mobile app that is evaluated by users is a pure data collection service, with a stop and play button, and parameters that can be adjusted according to the specifications made during the project. Independently on how professional the development of such app is, it is known from experience that this kind of service is usually negatively perceived by users.
- The first, test versions, of the app have biased the opinion of the users. The improvements are quite important in the last version of September 2022.
- The entire data collection process has to be considered for the score interpretation. For instance, the «*I needed to learn a lot of things before I could get going with this system*» question response is badly impacted by all the documentation that the end-users need to read and sign before using the app (this is the users' feedback) which are not associated with the usability of the app itself.
- The current app is a prototype, created in the framework of a RIA project. The app has been developed by professional developers and engineers within the scope of MODALES, not being a product off-the-self right after the end of the project.

3.3.2. User feedback

Except for the 10 questions included in the SUS scale, all trial-site participants have been asked to notify any technical issues they encountered during the period they have been asked to use the app. Some of the troubles are listed in the table below. It is expected that during the full trial deployment of the app, additional technical issues might occur and require a solution. Therefore, this table will be updated in deliverable D6.3 when using the full version of the mobile application from as many as possible volunteers.

Table 7: User feedback on app issues and technical solutions provided by the development team

No	Trouble	Potential Source	Therapy action
1.	Some users experienced a problem with data transfer, when using the following option: "*Only with WiFi*".	For the sake of flexibility and so that users do not have to use their data plan systematically, by default the transfer is done via Wi-Fi: as soon as the phone is connected to a WiFi network, an internal timer periodically checks (by default every 20 minutes) if there is data accumulated in the device that needs to be sent to our servers. The problem is that the timer is too long for some people, who close the application before being connected to a network, or who are never connected.	Encourage users to use "Wi-Fi and data" by default, reduce the timer or for users who wish to keep control, send data manually. All these options have been implemented in the application to give the end user more flexibility.
2.	Some users do not think to start or stop data collection.	Collecting data over a period longer than several days can be complicated for some users, especially when the application used does not offer a specific feature (which was the case for the first baseline version).	The start and stop of the data collection have been automated by detecting when the user is in the vehicle or not.

No	Trouble	Potential Source	Therapy action
3.	Some users reported problems with the graphical interface, such as colours or actions that were not appropriate on some phones.	The project's mobile application has been developed for recent phones running on iOS or Android. Some older versions may not be compatible with some of the graphical features used by the development framework.	Adaptations have been made where necessary, and the requirements for using the mobile application have been clearly explained. Android version should be 11 (API level 30) or greater. iOS version should be 9 or greater.
4.	Some users with specific phones had an error message: "Something went wrong, try again".	Potentially same as above. It is also possible that in the vast majority of cases, users have not been registered by the trial site leaders as being authorised to download the application.	Potentially same as above. Also check user access to the application, which is managed by the trial site leaders. To better control the users using the application, it has indeed been deployed on official beta-testing/private platforms: Testflight for iOS, requiring a special link; and a private version on the Android Play Store, where users have to belong to a Google Group.
5.	Some users have experienced problems with the iOS version specifically.	Apple's OS is less permissive than Android's, and special considerations must be taken into account, particularly with regard to data collection.	Permissions and licenses were set up specifically for the iOS version and the deployment was done on Testflight, Apple's official beta testing platform.
6.	Some users tried to change the 20 minutes synchronization to 1 minute so that data can be sent out sooner, but this wasn't working.	The default 20-minute interval has been implemented for testing purposes.	The functionality was subsequently restored.

4. Conclusions

The dashboard web application and the mobile application testing and verification are analysed and presented in this deliverable. The deliverable explains the procedure to test the data storage and evaluate the effectiveness of the processes, queries, and all other aspects of the data storage capability. Additionally, the deliverable presents the source code quality and verification.

More precisely, it analyses and reports the static code analysis, code quality maintenance, bugs detected in the source code, and security vulnerabilities detected. The deliverable also includes unit testing, which is used to confirm that the code operates properly and behaves as expected. System testing and integration testing are also covered in chapter 2. Regarding the testing and verification of the applications developed, the deliverable finally presents the performance testing. The whole procedure of testing and verification of the applications attests the efficiency of the applications (both the front-end and the back-end part), the computing performance, the source code quality and the absence of any security vulnerabilities.

Moreover, the deliverable presents the user acceptance method adopted. This covers the initial acceptance only, as up to the point of this report, the drivers were using the app in data collection mode only (without feedback or low-emission driving recommendations). For this task, the SUS was used for evaluating the usability of the app by 41 participants from all trial sites. The responses from a scale of 1-5 converted to a score of 62 which is considered “good”, taking into account the fact that the app is a research prototype and still under fine tuning until the end of the trail period in 2023. Later deliverables after the completion of the on-road trials (specifically D6.3: Trial Data Integration and Analysis, and D6.4: Impact Assessment Report, both due in 2023) will present further information on user feedback and acceptance, including of the advice given by the app as well as its functioning / ease of use.

5. References

1. Sonarqube.org. 2021. Code Quality and Code Security | SonarQube. [online] Available at: https://www.sonarqube.org/ [Accessed 15 December 2021].
2. Junit.org. 2021. JUnit 5. [online] Available at: https://junit.org/junit5/ [Accessed 15 December 2021].
3. Apache Tomcat. 2021. [online] Available at: https://tomcat.apache.org/ [Accessed 15 December 2021].
4. The Apache HTTP Server Project. [online] Available at: https://httpd.apache.org/ [Accessed 15 December 2021].
5. Brooke, J. (1996). "SUS: a "quick and dirty" usability scale". In P. W. Jordan; B. Thomas; B. A. Weerdmeester; A. L. McClelland (eds.). Usability Evaluation in Industry. London: Taylor and Francis.
6. https://xd.adobe.com/ideas/process/user-testing/sus-system-usability-scale-ux/ [Accessed November 2022]
7. MODALESD2.1: Variability of driving behaviours and Low-emission driving requirements. Report, March, 2020.
8. MODALESD2.2: Real effectiveness of OBD inspection and maintenance, and retrofits. Report, August, 2020.
9. MODALESD3.2: Correlation of user behaviour variability with emissions. Report, August, 2021.
10. MODALES D4.1: Recommendations for a broader use of On-Board Diagnostics (OBD). July, 2021.
11. MODALES D5.2: Functional specifications. Report, December, 2020.
12. MODALES D5.3: Mobile application. Report, December, 2022.

For more information:

MODALES Project Coordinator:

ERTICO – ITS Europe

Avenue Louise 326

1050 Brussels, Belgium

info@modales-project.eu
www.modales-project.eu

Adapting driver behaviour
for lower emissions

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 815189.