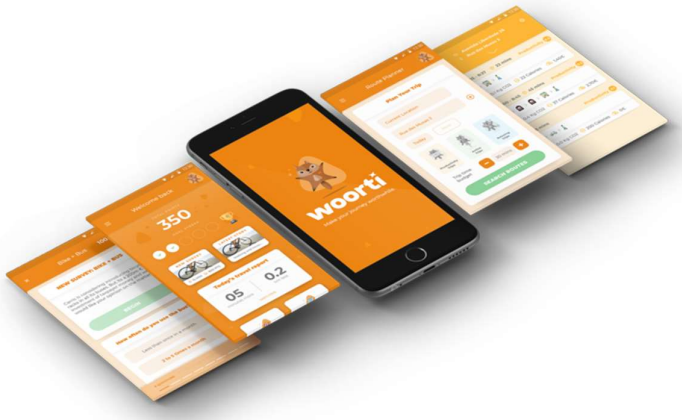
MoTiV Project
University Science Park
Univerzitná 8215/1, 010 26 Žilina, Slovakia

D3.4 MoTiV App



Project	MoTiV - Mobility and Time Value
Project no.	770145
Deliverable no.	D3.4
Type of deliverable	P (demonstrator) – accompanying reference document
Work package	WP3
Due date of delivery	30.09.2019
Actual date of delivery	30.09.2019
Author(s)	Luís Veiga (INESC ID), João Bernardino (TIS)
Responsible Partner	INESC ID
Document Version / Status	3.0 / Final
Date	30.09.2019
Reviewed by	Heikki Waris (CoRE), Viet Hang Nguyen (rRANK), Martin Hudák and Yannick Cornet (UNIZA)
Approved by	Project Board
Dissemination level	Public
Project website	www.motivproject.eu
Project Manager	Giuseppe Lugano | UNIZA | giuseppe.lugano@uniza.sk

Horizon 2020
European Union Funding for Research and Innovation
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 770145

Short report serving as a reference to the development work of the public demonstrator (MoTiV App).

D.3.4 is associated to the MoTiV Task 3.3 and Task 3.5, described below.

Description of Task 3.3 "App Implementation"
Task 3.3 App Implementation: In this task the app will be implemented, according to the specifications defined. Some budget will be kept in reserve to deal with possible implementation risks (Task 3.5). The development tasks will include further adaptation to accommodate communication and UX/UI improvements identified during the testing phase. INESC will also provide technical support during the MoTiV project lifetime (Task 3.5). In parallel to the MoTiV app implementation, specific functionalities of the MoTiV app will be implemented to support the case study of crowdsourced micro-tasks (including delivery of small goods, but not only).

Description of Task 3.5 "Testing, Adjusting and Support and Database Management"
Task 3.5 Testing, Adjusting and Support and Database Management: Three phases of testing will be carried out: - Internal testing by the developers - Reduced testing with users – led by the developers with a small pool of users - Wide testing with users – led locally by the MoTiV partners responsible for the collection of survey, with users in their sites A testing plan will be prepared and specific testing guidelines and user interface will be created. The results of the testing, together with the UX/UI inputs developed in Task 3.4, will be translated into consistent adjustments to the app. The task also includes the technical support to the tool, guaranteeing response to any technical issues or minor needs.

Document Revision History

Version	Date	Change description	Author
1.0	17/10/2018	Draft version	Luís Veiga (INESC ID)
2.0	29/03/2019	Updated draft	Luís Veiga (INESC ID)
2.1	08/04/2019	Reviewed draft	Heikki Waris (CoRE)
2.2	09/04/2019	Reviewed draft	Viet Hang Nguyen (rRANK)
2.3	12/04/2019	Reviewed draft	Martin Hudák and Yannick Cornet (UNIZA)
3.0	26/09/2019	Updated draft	Luís Veiga (INESC ID), João Bernardino (TIS)
4.0	30/09/2019	Quality check and final version	Giuseppe Lugano and Jana Velecká (UNIZA)

Disclaimer

Table of Contents

About MoTiV

The Horizon 2020 project MoTiV (Mobility and Time Value) addresses the emerging perspectives on changing Value of Travel Time (VTT). Accordingly, it explores the dynamics of individual preferences, behaviours and lifestyles that influence travel and mobility choices. In other words, what does value of travel time mean for the end users, in relation to their travel experience?

The MoTiV project addresses VTT from the perspective of a single individual with a unique combination of personality, preferences, needs and expectations, in contrast with the traditional viewpoint of the economic dimension (time and cost savings). Its approach aims at achieving a broader and more interdisciplinary conceptualisation and understanding of VTT emphasising its "behavioural" component.

The main goal of the MoTiV project is to contribute to advance research on VTT by introducing a conceptual framework for the estimation of VTT at an individual level based on the value proposition of mobility. The conceptual framework will be validated through data collection and evaluation in at least 10 EU countries. The mobility and behavioural dataset will be collected using a mobile application developed by the project consortium, which will combine and integrate in an innovative way features from a multi-modal "journey planner" and an "activity/mobility diary". With this mobile app, end-users will be able to more easily track, understand, and re-evaluate travel decisions to make the most of their free time in accordance with personal preferences, lifestyle, interests, and budget. The target is to engage in the data collection process a minimum of 5.000 participants actively using the MoTiV app and contributing with at least 70,000 trips to the MoTiV dataset. Besides validating the conceptual framework, the dataset will be made available to the scientific community as an Open Dataset to stimulate further research in this area.

The MoTiV project findings will produce scientific and policy outcomes, as well as potential business developments, including the development of new mobility services and the extension of existing applications, such as the ones offered by the business partners of the Consortium (i.e. *routeRANK journey planner*[1] and the *PiggyBaggy*[2] app for crowdsourced deliveries).

Partners

[1] https://www.routerank.com

[2] http://piggybaggy.com

List of Abbreviations and Acronyms

DCC	Data Collection Campaign
EC	European Commission
EU	European Union
GA	Grant Agreement
WP	Work Package

MoTiV Consortium Partners and Acronyms

Acronym	**Full name**
UNIZA	Žilinská univerzita v Žiline
CoRe	CoReorient Oy
ECF	European Cyclists' Federation ASBL
EUT	Fundació Eurecat
INESC ID	Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento em Lisboa
rRANK	routeRANK Ltd
TIS.pt	Consultores em Transportes Inovação e Sistemas S.A.

1. Executive Summary

The Woorti smartphone application (the Woorti app) was designed to collect mobility data from European travellers and their subjective assessments on use and valuation of travel time. Specifically, the collected data includes:

- *Traveller (user) profile:* socio-demographic details, transport preferences and attitudes towards travel time;
- *Trip data:* origin/destination, travel time, travel distance, modes of transport, travel purpose;
- *Traveller's contextual assessment of worthwhile travel:* traveller's trip rating in terms of worthwhileness, details on type of value (e.g. productivity, enjoyment, fitness) and activities carried out during the trip;
- *Contextual experience factors:* individual assessments on how features of the transport system (e.g. infrastructure, services, external conditions such as weather) influence mobility choices and perceived time value;
- *Mobility behaviour statistics:* aggregated information regarding personal travel behaviour (e.g. travel time, travel distance, overall trip worthwhileness, most common activities, calories burned, carbon footprint) and comparison of the statistics to users in the same community.

The design of the Woorti app took into account the requirement of user engagement, needed to recruit participants to the European-wide MoTiV data collection campaign (DCC). The main user value proposition chosen was to "*make travel time worthwhile*". This means that Woorti should enable the user to obtain information and reflect on her mobility choices with regards to the use and value of the time spent in his/her travels. Additionally, as a secondary value proposition, the user would be asked to support a research project with the aim of improving transport systems putting the traveller perspective and needs at the core of the innovation process.

The main features of the Woorti app were chosen to support the two interrelated objectives, to collect data and to engage users. This resulted in the following functionalities of the app:

- **My Trips:** detection, visualization, validation and reporting of trips.
- **Dashboard:** mobility behaviour aggregated statistics based on validated trips, to display at an individual level and by comparison with the community of Woorti app users.
- **Mobility Coach:** the coach presents stories to the user that incorporate interesting information and lessons regarding the use and worthwhileness of travel time.
- **Surveys:** possibility to collect additional survey data to complement the MoTiV dataset.
- **Targets & rewards:** information on targets and incentives associated to a specific local data collection campaign.

The biggest technological challenge, which is common to all apps collecting transport data, was the development of the mobile phone-based trip and mode detection engine. There are several specific challenges with this:

- The correct detection in space and time of trips, and its legs and transfers;
- The correct detection of transport modes;
- Performing the detection of trips and transport modes on the device, in near-real time, and without making use of any online or ulterior cloud backend support for processing.

Woorti strives to collect high-quality data, rather than just quantity. This principle is embedded in the interaction process with the user. Woorti does not automatically stores in its server all the detected (but not validated) trips. Instead, it collects only those trips that are reviewed and validated by the user. The review and validation require an extra effort to the user, so Woorti offers a set of incentives for users (e.g. personalised statistics in the app

dashboard, point system, possibility of connecting points to campaign rewards) to validate as many trips as possible. This approach to data collection, among others, is also privacy-enforcing as it is the user to choose which trips to validate and submit to the server.

The Woorti online back-office allows data collection campaign managers to manage their campaigns and accessing the collected data. The back-office allows setting up a data collection campaigns, monitoring user engagement, creating surveys, downloading data, and creating tailored target/reward schemes for users to share their data.

The Woorti app was launched on 1 May 2019 and it publicly available for the Android and iOS operating systems.

2. Overview

This document provides a summary on the scope, functionality and development of the architecture and implementation work of the MoTiV data collection application (i.e. the Woorti app), including the mobile applications, the web server application and the backend for data storage and processing.

The Woorti application was created specifically for the MoTiV project, with the purpose of collecting mobility data and user information and opinions on the value of travel time of the trips made by individuals.

This document includes the following main chapters:

- Application Scope and Design: The specific objectives, challenges, value proposition for users and main design choices and elements of the app are described;
- App Functionalities: A more detailed description of each functionality of the application;
- Software Architecture: Describing the main elements and approaches on the software development

Additionally, after concluding remarks, a more detailed description of some core software elements, such as the main data structures and algorithms developed, is provided in an Annex.

3. Application Scope and Design

3.1 Objectives

The Woorti mobile app has the primary objective of data collection. Specifically, it is an app that collects and processes different types of data on users' mobility behaviours, including their transport choices, and the context and information within which these choices are made, particularly:

- *Traveller (user) profile:* socio-demographic details, transport preferences and attitudes towards travel time;
- *Trip data:* origin/destination, travel time, travel distance, modes of transport, travel purpose;
- *Traveller's contextual assessment of worthwhile travel:* traveller's trip rating in terms of worthwhileness, details on type of value (e.g. productivity, enjoyment, fitness) and activities carried out during the trip;
- *Contextual experience factors:* individual assessments on how features of the transport system (e.g. infrastructure, services, external conditions such as weather) influence mobility choices and perceived time value;

- *Mobility behaviour statistics:* aggregated information regarding personal travel behaviour (e.g. travel time, travel distance, overall trip worthwhileness, most common activities, calories burned, carbon footprint) and comparison of the statistics to users in the same community.

3.2 Value proposition

The design of the MoTiV data collection app was conceived based on the assumption that besides serving the purpose of data collection, it had to be attractive to users and stakeholders. A study of users and stakeholders was undertaken with the purpose of understanding the best ways to obtain their engagement.

Following this study, an engagement strategy was created, which encompassed the guiding pillars of the app design. A user discovery exercise revealed various value proposition possibilities for users: Improving transport services; Activity and health information; Information on travel options availability; Information on wider costs and benefits of travel options; and Entertainment / gaming, rewards.

The assessment of needs vs. satisfaction suggested that most of these possibilities had either difficulty competing with existing players that are already positioned in the same value space (Figure 1), or that the needs in question are not significant enough for most people.

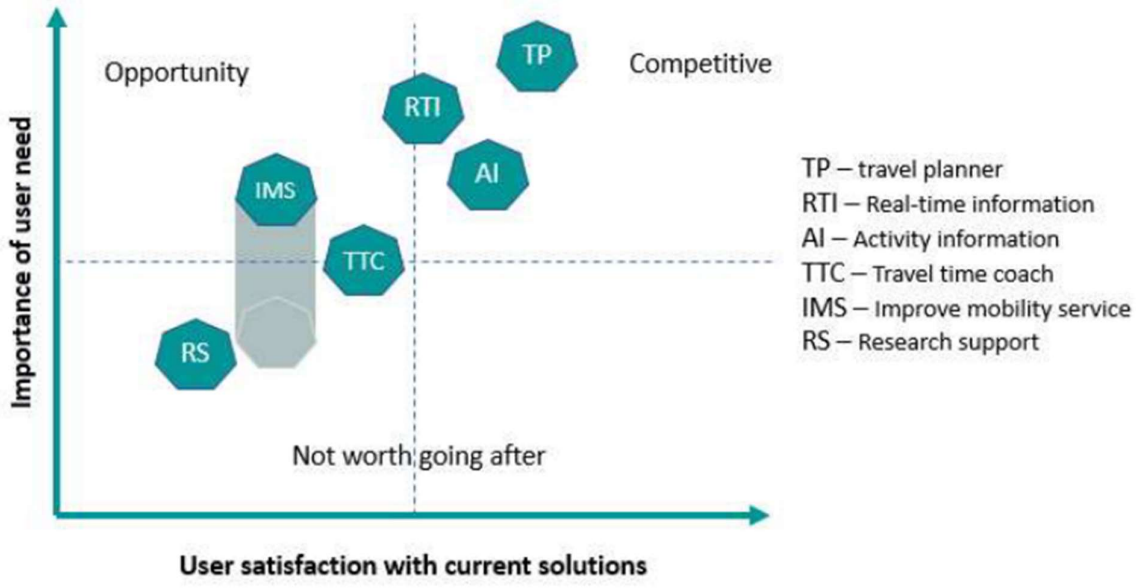


Figure 1: *Value proposition space*

Of the value spaces considered, the one that seemed to be more promising was the "travel time improver" or "coach", supported by motivating stories, mobility lessons and comparative statistics with the community. It focuses on an existing need (having a quality travel time) and is not addressed directly by existing applications. It also touches the core of the MoTiV project identity, making it easier to convey the message internally and externally.

In some contexts, other value propositions could be more promising. Since sample representativeness is as important as engagement volumes, it might be a necessary approach to use different user value propositions in instances of communication with different user segments. Therefore, the Woorti app branding and user experience

is compatible with different value proposition statements. The app addresses users with different expectations about the product. Gamifying the user experience and giving out material rewards were also deemed to give a strong push for engagement during the data collection period.

3.3 Branding

People engage with products or initiatives if they are appealing to them and if there is a trigger that drives them to it. This is true also for the Woorti app, which involves multiple types of challenges: among others, the very broad nature of the target groups to be reached and involved in the data collection campaigns, and the natural "competition" for user's attention by the Woorti app against all the other apps installed on the user's device.

The MoTiV Consortium decided to invest in a specific brand for the Woorti app with the objective of obtaining a higher attraction and engagement of users during the campaign. Additionally, this exercise would have also provided the foundations for a potential further exploitation of the app beyond the project life-time. The app branding is aligned with the value proposition and communication style defined in the design process of the application. The chosen tagline was "Make your journey worthwhile" (Figure 2).

Figure 2: *Woorti brand tagline*

The brand design also considered the desired positioning of the product perception in the dimensions of Human vs Machine and Quality vs Quantity, with the aim to differentiate the app in the Human/Quality quadrant (Figure 3).

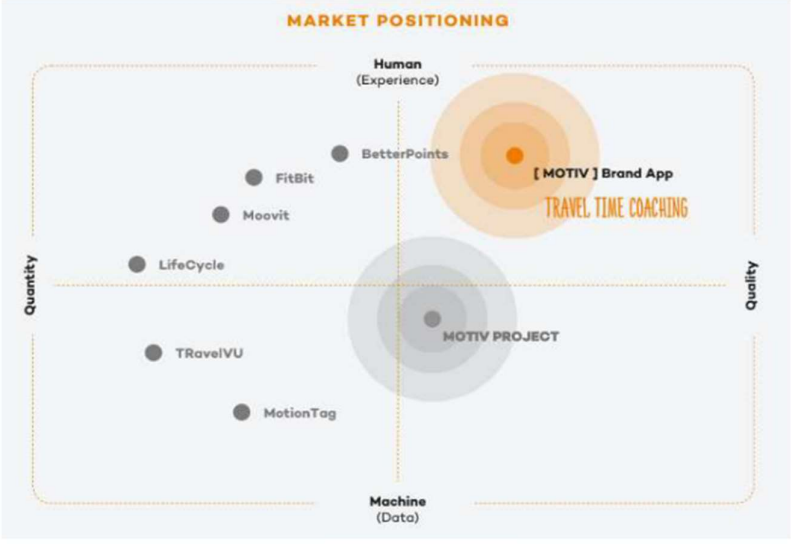


Figure 3: *Human/machine vs quality/quantity market positioning*

The chosen name of the brand was Woorti (Figure 4), a mix between the words worthwhile, time and travel. It was deemed to be short, universal, memorable and meaningful.

Figure 4: *Woorti app naming*

The visual identity (logo) of the brand has adopted a mascot. It was chosen in line with the brand DNA, according to its characteristics. The chosen mascot is a flying squirrel (Figure 5), an animal that expresses various values and characteristics aligned with the product: Energy; Play; Prudence; Balance; Socializing; Preparation; Resourcefulness; Discovery; Cool; Flexibility; Vision.

Figure 5*: Woorti visual identity*

3.4 Features for users

Following the analysis of the Woorti app value proposition, market positioning and branding identify, a set of relevant features to users was selected for implementation as application functionality. The Woorti app includes the following functionalities for the user: Onboarding, Home, Menu, My Trips, Profile & settings, Dashboard, Coach, Surveys and Targets & rewards (Figure 6).

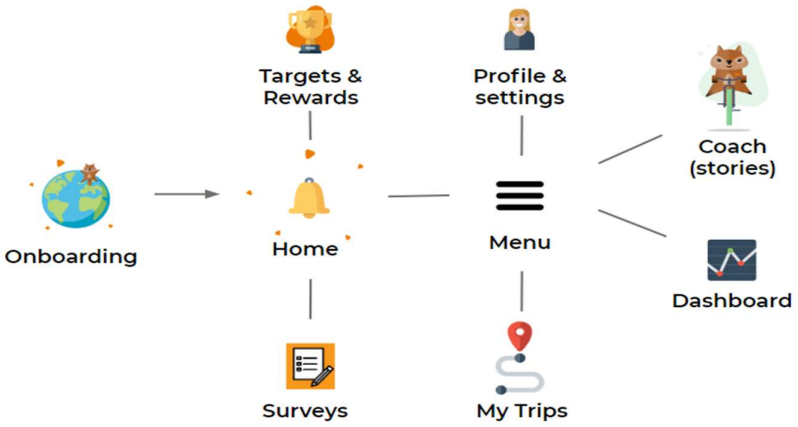


Figure 6*: Woorti app features for users*

- **Onboarding:** Through the onboarding process the user signs up, is introduced to the concept and objectives of the application and introduces basic profile information as well as travel time worthwhileness preferences, while onboarding the relevant local data collection campaign(s).
- **Home:** In the home screen the user gets the state of completion of her tasks (trip reporting, surveys, stories), a summary of trip data and the progress towards their targets and rewards.

- **Menu:** The menu tab allows the user to access any of the other app screens.
- **My Trips:** Visualization, validation and reporting of trips. This feature describes the data automatically collected by the app, the data validations requested to the user, and other data requested to the user about their trips (trip purposes, mood, assessment of worthwhileness, value of time according to the three components of worthwhileness, travel activities and experience factors).
- **Profile & settings:** In the profile and settings screen, the user can edit or introduce new individual data (including demographic data, home/work location and travel time worthwhileness settings), access tutorials, change app language, access rewards information, send feedback to the app developers and read the privacy policy.
- **Dashboard:** In the dashboard screen the user can access multiple statistics related to her validated trips, both at an individual level and by comparison with the community of Woorti app users.
- **Coach:** The coach screens presents stories to the user that incorporate interesting information and lessons regarding the use and worthwhileness of travel time.
- **Surveys:** Surveys originated from the research project are sent to users and presented at the home screen.
- **Targets & rewards:** According to each local data collection campaign, the user is presented with targets and possible rewards. This information is summarized in the Home screen and presented in detail in the Profile & settings screen.

4. App Functionalities

The Woorti app offers users a set of functionalities to provide information about their mobility, to track and validate trips, to answer surveys, to assess their usage and progress towards campaign rewards, and to consult mobility and sustainability related statistics, both of their own as well as of the communities they are enrolled in. This Section provides a non-exhaustive overview of the most relevant features provided by the Woorti app, as seen by the users (on mobile phones for regular users and on the web app for Campaign Managers).

Upon launching the application, the user is prompted to sign in (Figure 7). The user can log in to the app by either creating an account exclusive to the Woorti app, with the username and password previously defined, or by using an existing Google account:

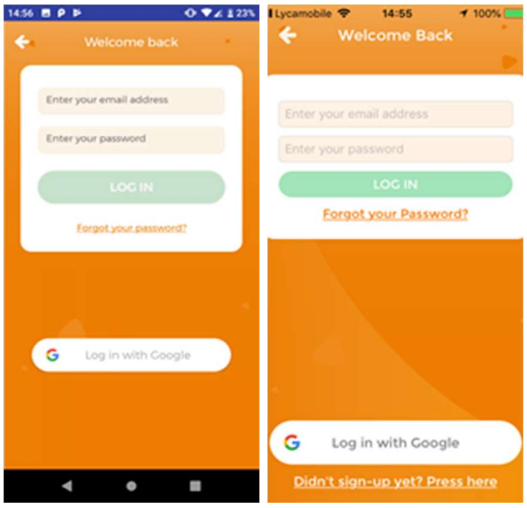


Figure 7: *Android (left) iOS (right) login interface*

4.1 On-boarding

The onboarding module is where the user makes his/her first relevant interaction with the app. In this interaction the user goes through a series of screens, presenting the context of the Woorti app (the MoTiV research project), followed by a tutorial on what is worthwhileness of a trip, referring to productivity, enjoyment and fitness (Figure 8).

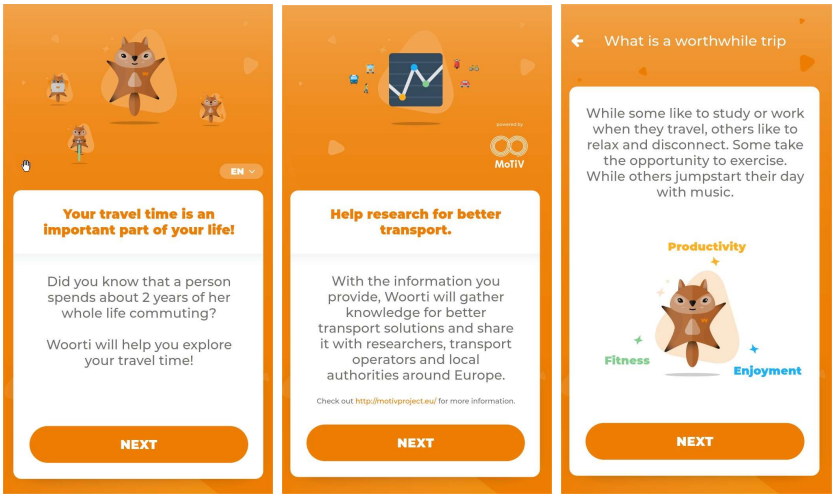


Figure 8: Initial onboarding sequence explaining the app context and worthwhileness proposition

As a next step, the user is asked to provide information regarding her specific trip worthwhileness parameters, and other preferences related to used modes of transport, and the relative importance of each worthwhileness factor for each transport mode (Figure 9). The on-boarding concludes with information on country and city of residence, information about compliance with GDPR policies, name, gender and age group, etc.

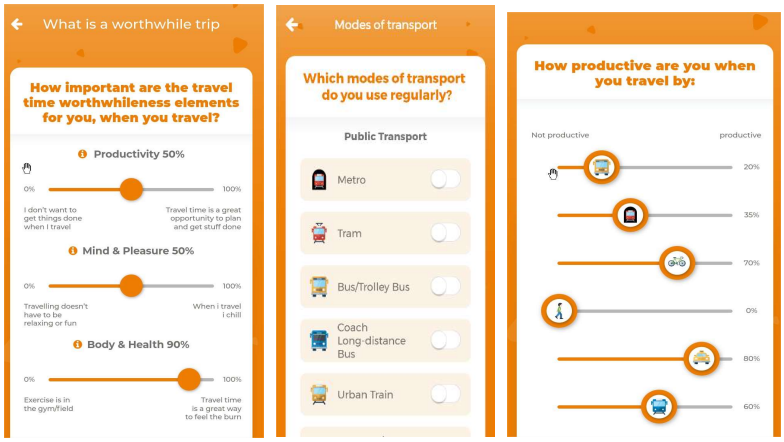


Figure 9: *Worthwhileness parameters and transport preferences in the user onboarding*

4.2 Main menu and modules

After logging in, the app takes the user to the main menu (Figure 10). The user can always go back to the Main menu by sliding it from the left-hand of the screen. Here users can access Woorti's main modules. The Home

module presents general information and reminds users of trips to be validated, surveys to answer, and stories to read. The My Trips module is the main module where users can check and validate their trips. The Dashboard presents more detailed information regarding the user's travelling over a recent period of days. The Mobility Coach module presents all the mobility related stories that are being unlocked one by one as the result of using the app. The Profile and Settings module allows a user to complete information about his/her profile and travelling preferences regarding modes of transport and activity goals.

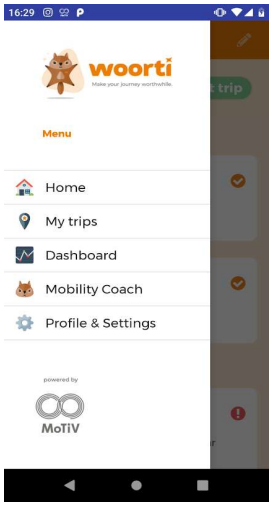


Figure 10: *Woorti main menu*

4.3 Home screen

The Home screen (**Figure 11**) is the landing page which appears when the user opens Woorti. The user can check at the top the total score from trips, total days with trips validated and the total number of trips validated. Then, for each of the rewards displayed (set up by Campaign Managers), users can check their progress (typically, relating to user's progress in trip validation for a given period or all time). At the bottom of the module, users are reminded trips pending validation, stories to read, surveys pending answering, and a short 7-day travel report.

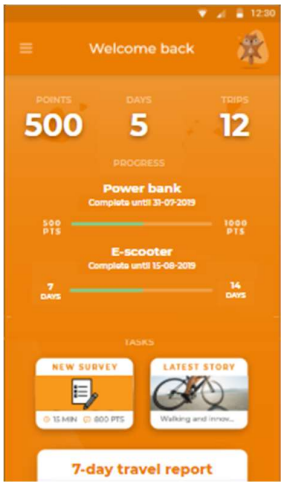


Figure 11: *Home screen*

4.4 My Trips

The My Trips module is the one with the key functionalities regarding detection and validation of users' trips.

Start/Stop detection

Woorti monitors the acceleration of the users' devices in order to detect when they started and finished a trip, so that it can avoid having the GPS always on with high-precision, in order to save battery. However, if the accelerations reported by the device are very subtle, a starting trip may still go undetected.

To assist Woorti in such cases, the user can directly inform Woorti that he/she is starting a trip. Similarly, at the end of a trip, the user can also directly inform Woorti that the trip is over to see and be able to validate it immediately. For this purposes, two buttons are available, "Start Trip" and "Stop Trip" (Figure 12).

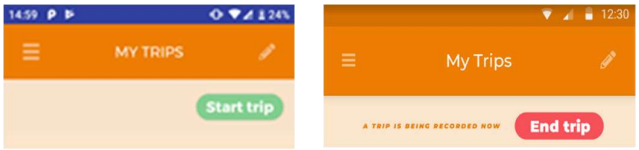


Figure 12: *Start/Stop trip button*

Viewing On-going Trips

When a trip is in progress, this is shown by a message left of the "Stop Trip" button. Pressing it will lead to a screen which shows the user with temporary information regarding the legs that were already identified and modes detected. This information is only final after trip completion.

Trip Validation

After trip completion, the user can select the most recent trip and is taken to a validation screen, in order to validate which transport mode was used in each of the trip's identified legs. These are shown in the app with specific icons for each mode of transport (Figure 13). In this activity, the user may also split one leg into two, merge a leg with the next one, or delete a leg, in an interactive fashion.

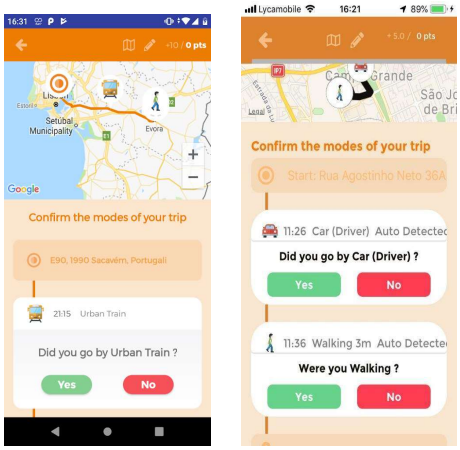


Figure 13: *Android (left) and iOS (right): Leg validation screens*

As users validate trips, they earn points (shown on top-right corner) and they are taken to a number of screens where they can provide more information (e.g., trip purpose, activities carried out during the trip, and relevant experience factors - Figure 14) about the trip as a whole, as well as for one leg that is selected as the most relevant.

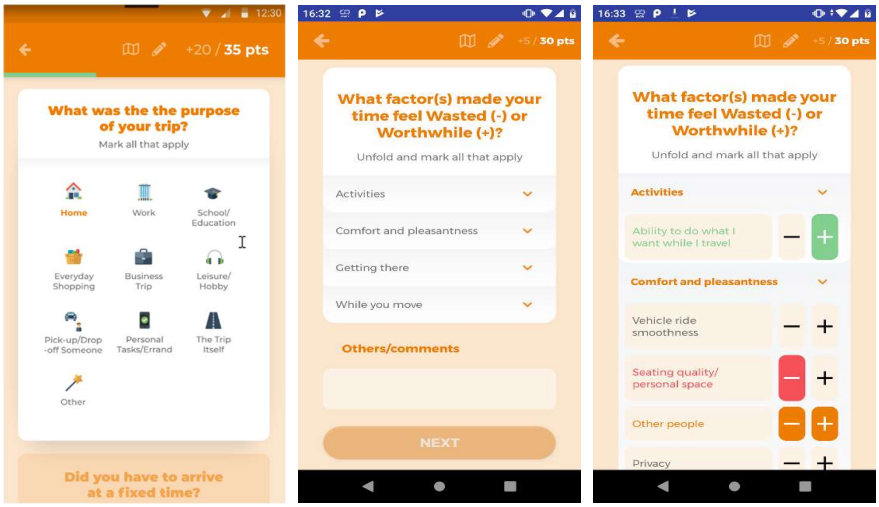


Figure 14: Information to describe trip purpose and experience factors

Trips timeline

Once there are completed trips, the user can also check them in the main view of the MyTrips in a timeline fashion (Figure 15) as list of trips already detected and their state (validated or not validated). The trips with the orange tick mark were already validated by the user and sent (or scheduled to be sent) to the server; the ones with red exclamation mark have not been validated by the user yet and need attention.

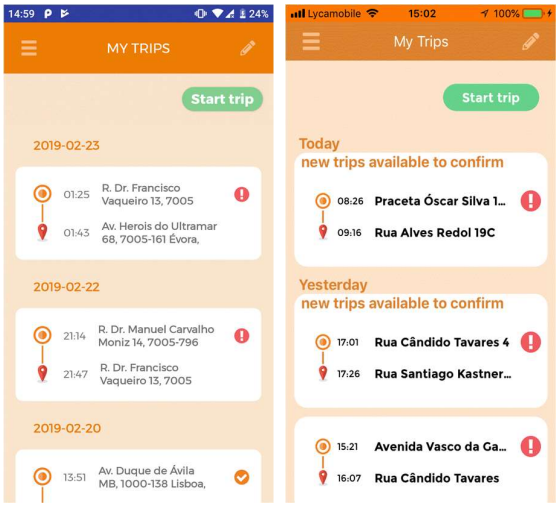


Figure 15: Trips timeline in Android (left) and iOS (right)

The user can also merge, delete and split trips in this module (pencil icon at the top-right corner). This may be needed in certain situations. Users can easily make these changes for trips that were incorrectly detected (e.g., due to noise/errors in readings from location sensors, despite filtering/cleaning). The user can also delete the trips

he/she wishes to remove from Woorti (before validation) and that therefore will not be included in the data analysis. Trip split and trip merge account for those situations where (e.g., due to absence of GPS coverage) one trip was incorrectly detected as two separate trips (can be merged), or when two or more individual trips (close in time) were incorrectly detected as legs of the same single trip (can be split).

Merge, Split and Delete Legs

Besides merging, splitting and deleting trips, the user can also merge, split and delete legs within each individual trip (Figure 16) to accurately reflect the actual trip that took place, in case it was not accurately detected. This may happen due to the different calibrations of sensors in different mobile devices, or due to unusual patterns not recognised by the trip detection algorithm. To perform this, the user must enter the trip validation screen for the already completed trip. The user can click on the pencil icon (highlighted) while viewing a trip to trigger the selection of the intended action (split, merge, delete) regarding the legs of the trip. This is also carried out in an interactive fashion, including selecting the point where a leg should be split in two (on the map view).

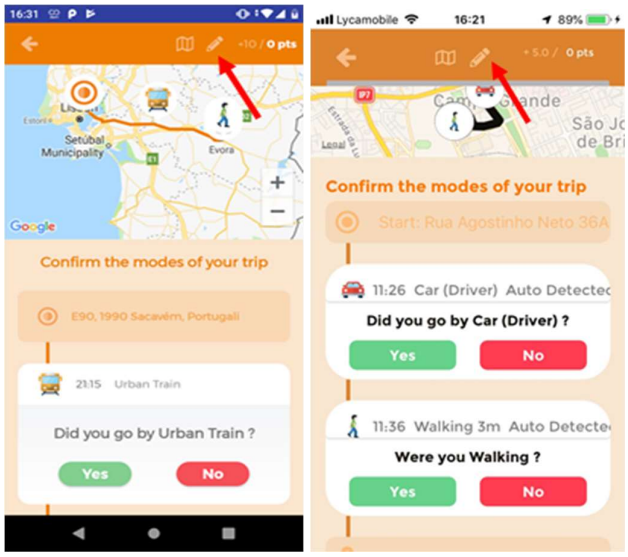


Figure 16: Trip view before starting leg merge, split or delete in Android (left) and iOS (right)

4.5 Dashboard

Dashboard enables users to consult statistical information about their mobility. The information includes distance walked, time spent traveling, worthwhileness indicators, both in general or specific for productivity, enjoyment and fitness.

Users can also check a ranking of their most common travel activities, and information regarding calories consumption and CO2 emissions. This information can be seen over different time spans: over the last day, 3 days, week, month, or year. Images below show the top and bottom view of the dashboard for the user's statistics *(You),* and for a community *(Lisbon)* that the user can select from those he/she is enrolled in (Figure 17 and Figure 18).

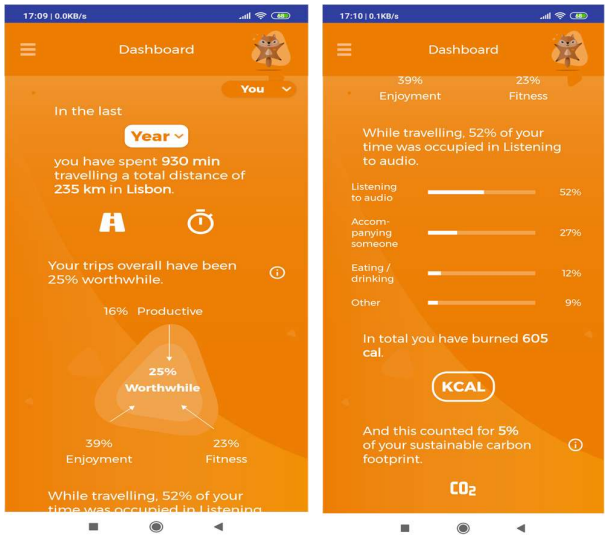


Figure 17: Dashboard top and bottom view (individual user)

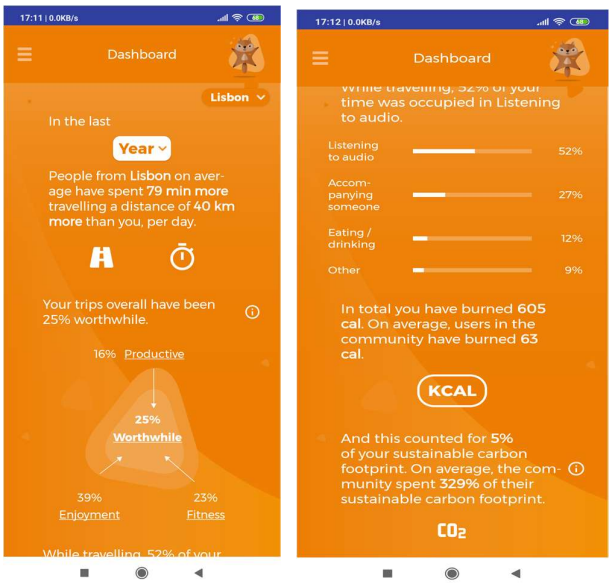


Figure 18: *Dashboard top and bottom view (community user)*

Furthermore, when clicking on specific parts of the Dashboard (distance travelled, time spent travelling, calories consumption and CO2 emissions), the user is brought to more detailed screens displaying, for each of those aspects, about total and percentage values per transport mode. These screens (Figure 19) also display comparisons with community-wide related information when the user has selected a community in the Dashboard main screen.

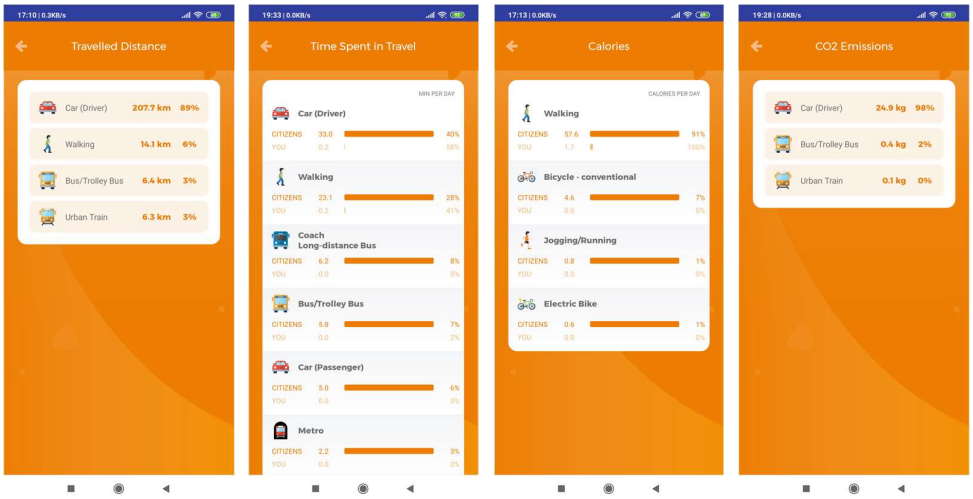


Figure 19: *Dashboard statistics – Travelled distance, Time spent, Calories, CO2 emissions*

Additionally, when pressing on the worthwhileness triangle diagram in the main Dashboard screen, the user is shown worthwhileness related statistics (Figure 20), that consider criteria related to productivity, enjoyment and fitness reported when trips are validated. The statistics can also be shown in comparison with community-wide values gathered over the period of time selected.

Figure 20: Dashboard statistics – Overall Worthwhileness statistics for community and user.

4.6 Mobility Coach

The Mobility Coach functionality allows the user to get and read new stories every day (Figure 21). In this screen, the user can see the list of available stories to watch; after reading one, a new story will appear on the next day.

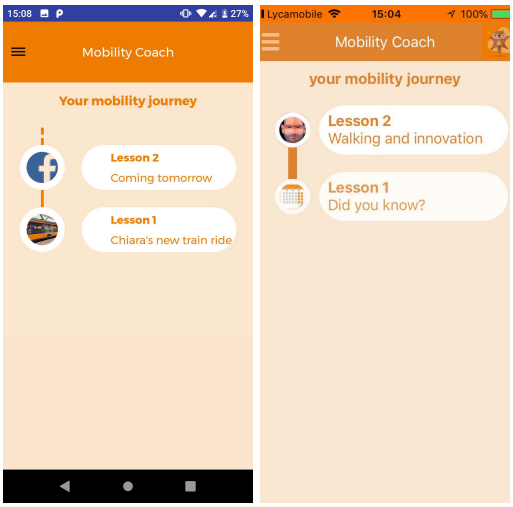


Figure 21: Mobility Coach stories in Android (left) and iOS (right)

4.7 Profile and Settings

The Profile and Settings functionality allows the user to complete and update their personal data (besides the data provided during Onboarding), such as name, home and workplace, age, gender, education, country, residence (Figure 22). The user may also change how important are productivity, enjoyment and fitness, what are the user's preferred modes of transport and how one can rate these with regard to productivity, mind and pleasure and body and health. The apps language[3] may also be changed and the user may log out of the application.

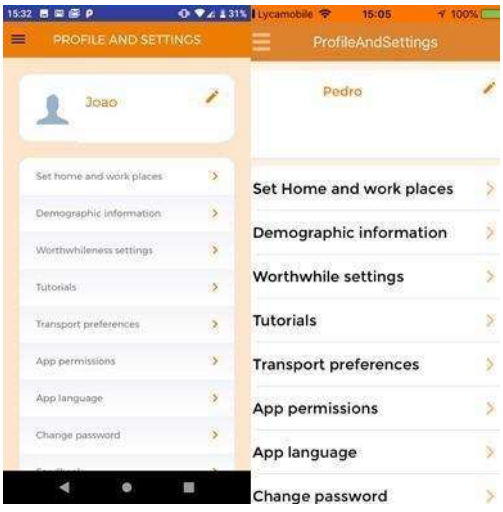


Figure 22: Profile and Settings in Android (left) and iOS (right)

In case the app has not worked as intended (e.g. a trip not detected when it should have been, a trip detected when no trip occurred, travel mode wrongly detected, user interface issues), the user can also send feedback reporting from the Profile and Settings module.

[3] The Woorti app is available in 12 languages: Catalan, Croatian, Dutch, English, Finnish, French, German, Italian, Norwegian, Portuguese, Spanish, and Slovak.

4.8 Admin Control Panel for Campaign Managers (WebApp)

Woorti includes an admin control panel for Campaign Managers provided as a web app (Figure 23). In the WebApp, users with access privileges (Administrators and Campaign Managers) can manage, configure and retrieve information regarding the main aspects related to data collection (e.g., campaigns, surveys, rewards). The interaction with the WebApp starts with the login screen to authenticate the user.

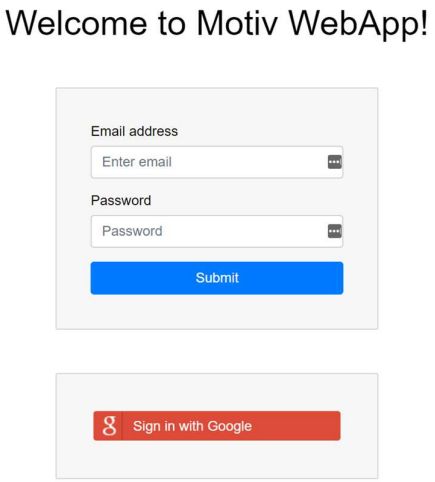


Figure 23: Woorti WebApp login screen

After successful authentication, the user is shown an initial screen (Figure 24) displaying global statistics about Woorti activity for different periods of time (current week, current month, and overall), with the number of active users as well as the number of trips validated by users and sent to the server.

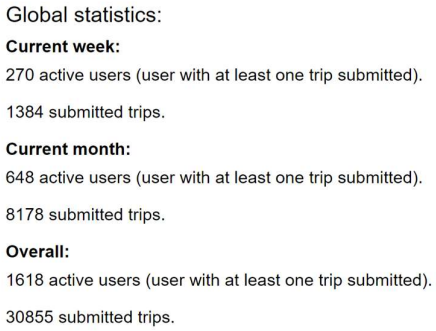


Figure 24: Woorti WebApp showing overall activity information

Campaigns

The main entity managed in the Woorti WebApp is a campaign. Campaigns are created and managed by Campaign Managers. As shown in Figure 25, a Campaign Manager can create a new campaign by specifying its name, whether it is public (specific targeted campaigns, for users to explicitly choose to enrol in, if interested) or private (for Woorti data collection in the context of a city or country, where users are automatically enrolled based on their profile information), and its geographical scope (country and specific city if desired). Irrespective of others created by

Campaign Managers, there is a predefined master campaign for each participating country in Woorti that allows to centralize data collection regarding that country.

Figure 25: Creating a new campaign in Woorti

Campaign Managers can further configure their campaigns by stating how users are awarded points by validating trips and providing additional information regarding worthwhileness, activities and experience factors (Figure 26). All elements can be awarded individual scores, as an incentive, depending how the Campaign Manager regards their relative importance for the data collected by the campaign.

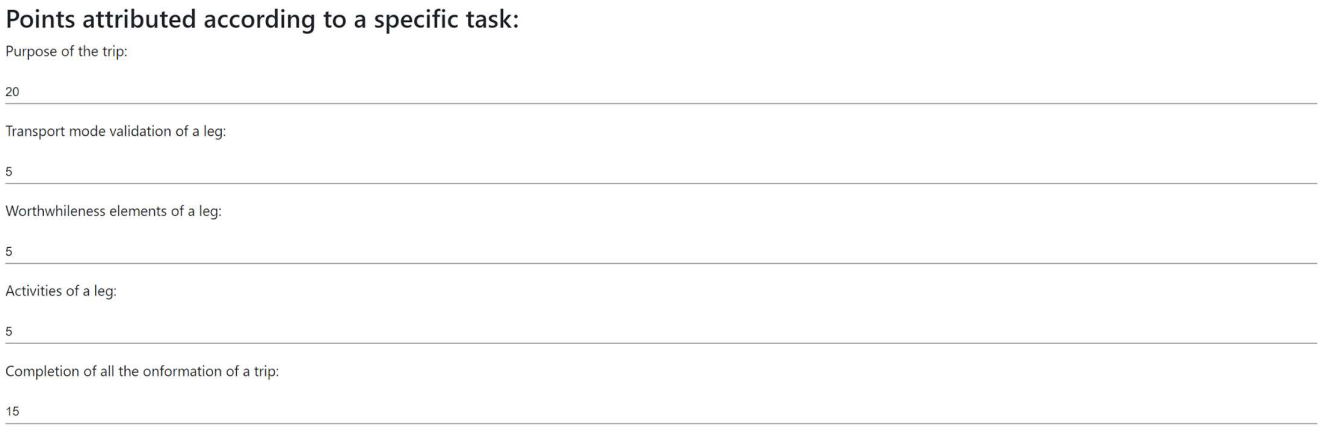


Figure 26: Configuring point system for user participation in a campaign

4.9 Surveys

Besides validating trips and providing trip info to Woorti, users can also contribute to data collection campaigns by answering the surveys that are presented in the mobile app. In the WebApp, Campaign Managers can create surveys targeting the users enrolled in a given campaign or set of campaigns. Answering the survey will award a specified score to the user (Figure 27).

Campaign Managers can define the period during which the survey is valid (afterwards it is no longer presented to users). Although the survey functionality allows several survey triggers, in MoTiV campaigns surveys are triggered once, based on a specific selected period during which the survey is open and available to users.

Surveys are created with a default language selected, but they are presented to users in their own language if the questions have been translated (addressed next).

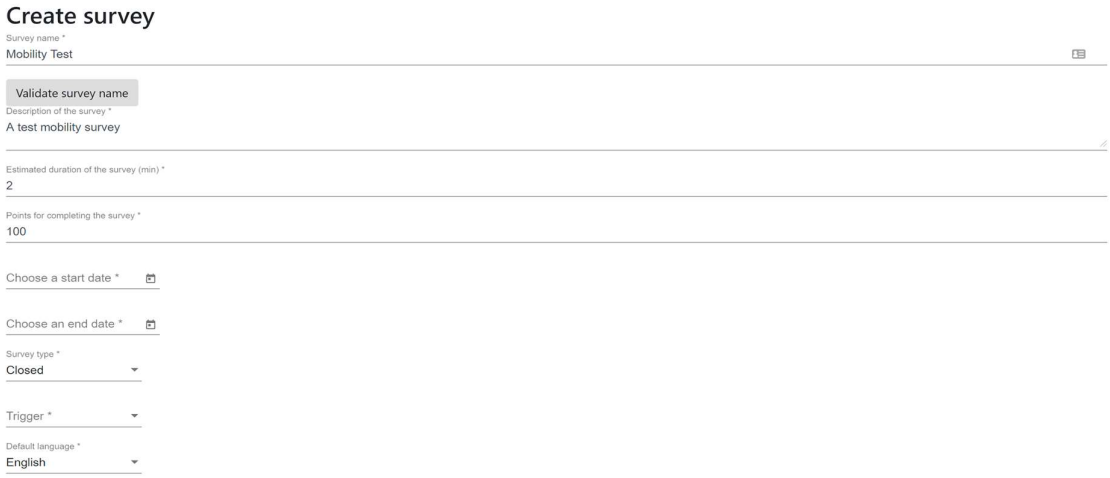


Figure 27: Creating a survey in the Woorti WebApp

Surveys in Woorti are comprised of sets of questions (Figure 28). Each question can have a simple text answer (short, long), or be answered with single or multiple predefined options (checkboxes, radio buttons, dropdown). Questions can even be reused across surveys, a very useful feature for Campaign Managers, provided that they are not modified (otherwise, surveys can be duplicated and further edited by adding or removing questions).

Despite Woorti being used in several countries, data collection in several campaigns may share specific goals and types of information to be gathered from users. For this, Campaign Managers can reuse questions and translate them (question and answer options text) for each specific language. Therefore, whenever a question is available in the user's language, it will be presented as such. Otherwise, the default language is used. This way, similar information can be requested to users across countries, presented in their language, but later processable in a combined way.

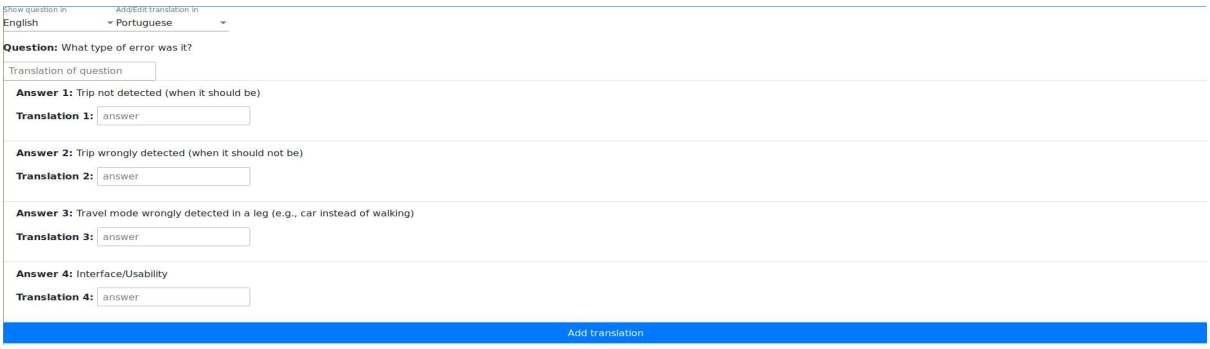


Figure 28: Creating and translating questions for surveys

Rewards

An additional mechanism developed to support user engagement in Woorti are the rewards. Campaign Managers can define rewards associated with a specific campaign (Figure 29). Rewards are created with name, description (that can be translated to other languages) and a link to contact or dedicated web page.

Rewards are defined with a target goal (either the score in the campaign, number of trips validated, or the number of days with validated trips) and a time validity (either a specific period of time with a start and end date when the target can be achieved, or over all time of Woorti usage).

Campaign Managers can track the current situation of all users enrolled in campaign towards achieving the target goals (and users can track their individual progress in the mobile app).

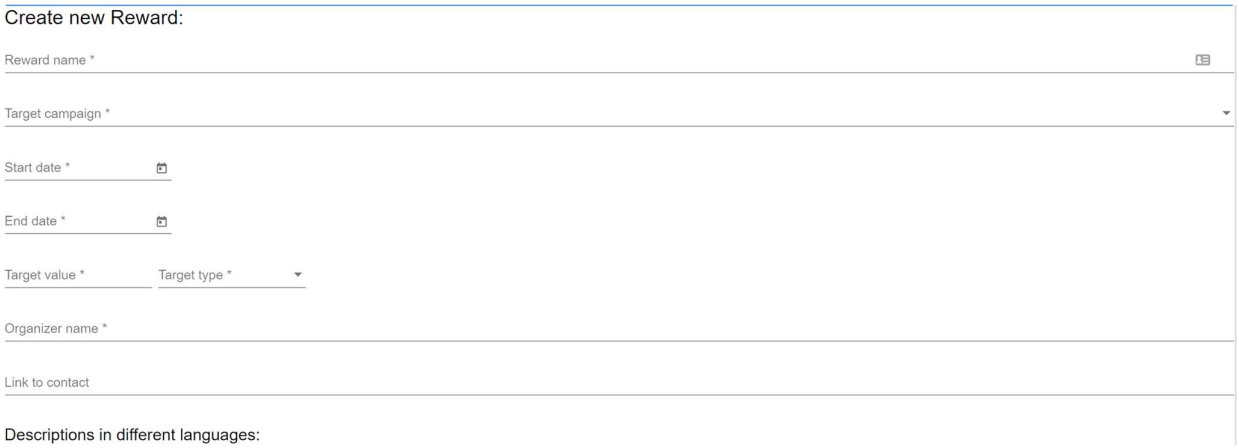


Figure 29: Creating a reward (e.g., gift, prize, experience) for further user engagement

5. Software Architecture

Following the application requirements (specified in MoTiV deliverable 2.3), the Woorti app was implemented with the architecture described in Figure 30. The modules follow a layered approach: i) user interface for user interaction, ii) engine for processing, iii) data storage for persistent storage of data. This is mirrored in the mobile app as well as in the backend (that includes the Campaign Managers back-office).

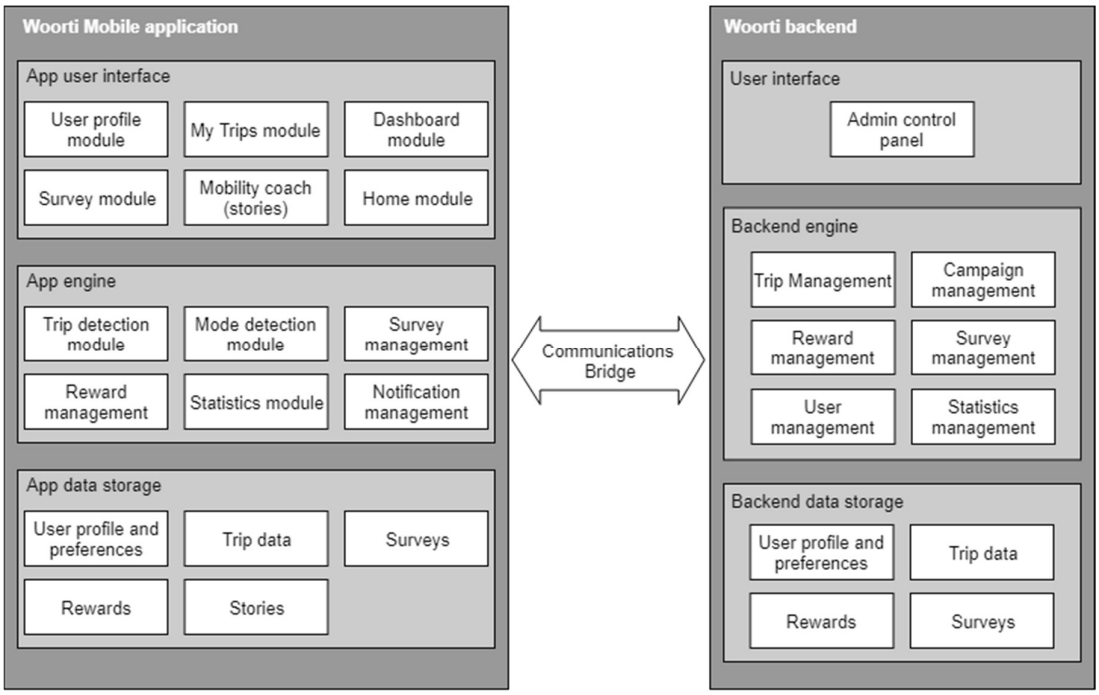


Figure 30: Module structure for the Woorti app.

In the course of the definition of the app specifications, some modules were grouped (typically, related modules even if residing in different layers), so that as much as possible they could be defined and developed independently. The most relevant modules in the Woorti mobile application include the following:

- **Trip module:** This module contains the data collection and user interaction elements related to the trips recorded by the app. Essentially, it describes the data automatically collected by the app, the data validations requested to the user, and other data introduced by the user. *[MyTrips, Trip detection, Mode detection, Trip Data]*
- **Survey module:** A survey is a question or set of questions or information, defined by Campaign Managers, and shared with the user that is being targeted and requested to answer it. The survey module handles surveys targeted to the user, triggers them at the specified time, and collects users' answers. *[Survey module, Survey Management, Surveys]*
- **User activity information (quantified self):** This module defines the data that is collected and processed to show users information about their activity (distance, time, transport modes, calories, CO2), as well as access to global statistics (calculated by the backend) regarding the activity of users in the same communities (e.g., city, country). *[Dashboard, Statistics, Notification]*
- **User onboarding:** The process of onboarding users in the app, when they first download it. It allows users to define and update the information about their demographics, transport preferences, worthwhileness criteria and initially enroll in campaigns. *[User profile, User profile and preferences]*
- **Mobility coach:** The travel time coach is a form of introducing the user to a sequence of various types of contents related to her travel time experience, with the main objective of creating engagement with the app through a valuable feature. *[Mobility Coach, Stories]*

- **Rewards:** The rewards are the means, designed during application development, to further incentivize user participation in Woorti (by validating trips and answering surveys) in order to be eligible for awards, gifts, experiences as defined by the Campaign Managers. *[Reward management, Rewards]*
- **Home module:** Finally, the Home module is the main provider of high-level information to users summarizing their situation regarding trips completed, days with trips, pending surveys to answer, current progress towards rewards eligibility and a short 7-day travel report.

The most relevant modules in the Woorti backend, that provides services to the mobile application and the web app for the administration back-office are the following:

- **Admin control panel:** The control panel is the part of the Woorti backend that provides a user interface to Woorti administrators and Campaign Managers (the WebApp), where they can perform all the management tasks of data collection campaigns (e.g. monitoring progress of the campaign) and invokes the appropriate lower-level modules in charge of each specific task.
- **Trip Management:** This module is in charge of receiving trip information from the mobile application once users have validated a trip and there is connectivity to send it to the backend. Trip information includes date and time, the comprising legs and their locations, their modes of transport (both detected and corrected), as well as all the additional information users provide about the trip experience (e.g., worthwhileness and experience factors). *[Trip data]*
- **Campaign Management:** This module handles the management of campaigns in Woorti. Campaigns allow the Campaign Managers to define sets of users based on geographic criteria (e.g., a country, a city, any location in a country within a radius from a given location), a period, and also by enrolling individual users specifically. Such sets of users can become the target of surveys and also eligible for rewards. This allows managing users' scores (based on trips validated and answers submitted) and providing Campaign Managers with aggregate information regarding each user community (e.g., user profiles, number of trips).
- **Survey Management:** The survey module handles the definition of surveys (sets of questions), their propagation to target users (typically one or more target campaigns), and the storage of survey answers received. Surveys can be created to be triggered at, or after, specific dates and times, or on recurring events, such as when detecting a starting trip. *[Surveys]*
- **Reward Management:** This module defines and handles rewards, that are created in the context of campaigns. They state goals for the enrolled users to achieve over a specific period of time or globally, namely a given score, number of validated trips, days with validated trips. *[Rewards]*
- **User Management:** This module manages the information received from the mobile application when users define or update their profile information and transport preferences. *[User profile and preferences]*
- **Statistics Management:** This module calculates periodically the global statistics regarding the behaviour of each relevant user community (e.g. city, country, campaign) and provides them to the mobile apps of users in each community, when they connect to the backend. To achieve this, it iterates over Trip data received and generates aggregate results for each community and specific period of time (1-day, 3-days, week, month, year).

The interaction between the Woorti mobile app and the backend is carried out through a **Communications Bridge** that mediates the invocation of services and service responses and notifications. Through this bridge, the Woorti application communicates with:

- Backend server to send/retrieve application-related information, namely about: Trips, Surveys, Users, Profiles, Preferences, Rewards, Statistics.
- Firebase Authentication to authenticate users and later communicate with the backend server.

- Firebase Crashlytics to send crash reports.
- Firebase Cloud Messaging for Push notifications sent to mobile phones informing a new survey is available to download.

The most relevant modules of the Communications Bridge include:

- **Server communication**: This serves as an interface to handle all network communication of the mobile app with the Woorti backend in an asynchronous way. This is important in order to prevent the mobile application user interface from blocking, and to enable delayed communication with the backend when there is no connectivity (in particular, to send the info of validated trips). It translates the information sent by the server to information needed by the app and vice-versa. It also notifies the app when specific requests are finished.
- **Authentication**: It handles apps sign in and out functionalities, maintaining a reference to the currently logged in user and therefore enabling communication with the backend server. Only logged in users can send trips to the server. This module handles logins via requests to a support service (Firebase) and then manages the user's session with Woorti.
- **Push notifications**: It handles all push notification events sent from server. These are sent to notify the app that it has to download more surveys from the server. After receiving the notifications, it requests the Server Communication module to get the missing surveys from the server. To tolerate loss of push notifications, the app also refreshes this information periodically when it reconnects to the server.

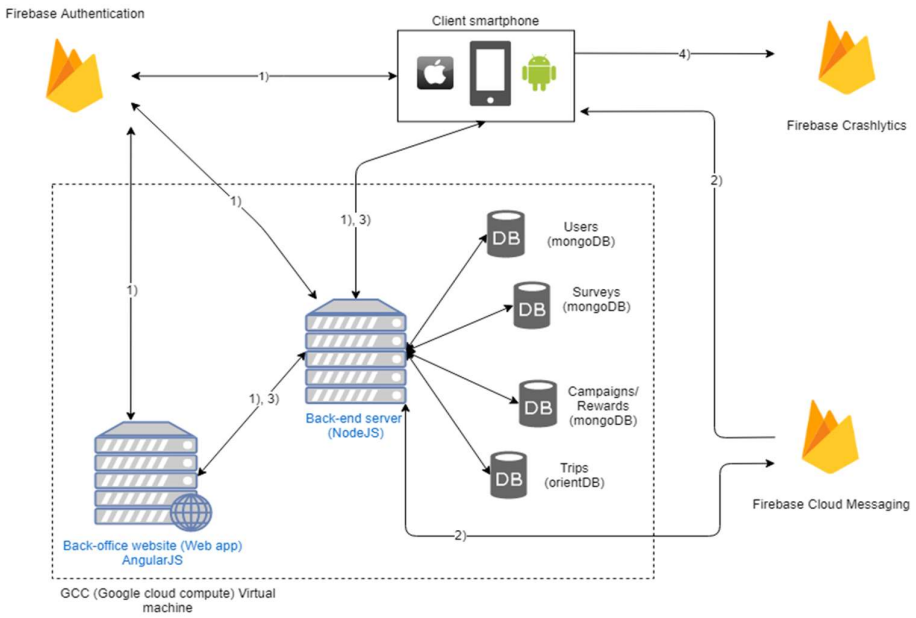


Figure 31: Distributed architecture of Woorti

Figure 31 above further details the distributed architecture and the most relevant interactions within Woorti. It displays the following types of connections:

1. Firebase authentication.
2. Push notifications.

3. Internal communication between the mobile app and the backend server.
4. Firebase Crashlytics.

In the back-office we have:

- 4 Databases (3 in mongoDB, 1 in orientDB).
- 1 Backend Server developed in node.Js.
- 1 WebApp developed in Angular 6.

The main interactions depicted in the distributed architecture are the following:

- Information about Trips is sent from the mobile phone to the Server (**3**) when a user confirms a Trip with the Woorti app.
- Surveys are sent from the server (**3**) to the mobile phone when it asks to receive new surveys (either periodically or acting upon a push notification).
- User information is sent to server (**3**) either when performing login, sign in or updating user preferences.
- Campaign information is requested by the mobile phone (**3**) when the user is asked to choose campaigns during the onboarding. This information will be sent to the server along with the information about user preferences.
- Firebase Authentication (**1**) service is used to authenticate all users. This makes the process of managing users more agile, allowing a lot of sign in options (eg: email/password, google) and guaranteeing a secure authentication process for the user.
- Firebase Authentication (**1**) service is also invoked for the management of nameless tokens that provide for user anonymity and confirm that a user is legitimate. Such tokens are managed by Firebase.
- Firebase Crashlytics (**4**) is used to receive crash reports from every user, remotely and sent when user has connectivity.
- Firebase Cloud Messaging is used to trigger Push notifications (**2**), that are used in Woorti to send information from the server to the mobile phone with notifications. This mechanism is used to instruct the mobile phone to download new surveys. When a survey is launched from the back-office, a push notification is sent to all the mobile phones of the users that were targeted by the launched survey.

All of the components mentioned earlier are deployed in a single virtual machine but can function in separate ones. Each of the four databases has information about one specific topic of the application. This enables the possibility of having each database in a different machine, if it is later required for increased performance and/or to reduce the risk of user information exposure if one of the servers is attacked.

The databases containing user, campaign and survey information were developed in MongoDB (NoSQL) instead of the standard SQL. The difference between both is that NoSQL is a lot faster for large volumes of unstructured or semi-structured data. Since Woorti has to process lots of unstructured and semi-structured data, noSQL is the best option for scalability.

The other database used was OrientDb. This was used to store trip information due to its ability to store and query large amounts of data using multiple data models. Since It is also a graph database it allows to represent trajectory data in a more efficient way.

The backend server's purpose is to be an interface between the human interfaces (Woorti mobile app and the Web portal for the back-office, the WebApp) and the four databases. Node.Js was chosen due to being easy to use, easy

to learn and easy to develop while maintaining all the features required for a backend server. This also has a lot of community support, with an active and vibrant community. To follow the same paradigm as in node.js we chose to develop the WebApp in Angular 6. The languages are very similar, easy to use and this web application framework is also widespread.

6. Development life-cycle

The development of the Woorti app was carried out following an agile approach, in parallel, and receiving input from: MoTiV project Task 3.2 (app design), Task 3.4 (UI/UX and branding), and input and results from simultaneous work from other MoTiV project work packages (e.g., WP2 for application requirements, WP4 to support Campaign Managers, and for WP5 for data export for analysis).

Over the duration of the application development, in successive (typically weekly) iterations, the implementation of the user interface and functionality of the Woorti mobile app was carried out following the updated set of specifications from UX/UI and feature design (also resulting from work, requirements and requests originating in other WPs). This work was partially reiterated as new versions of the UX/UI designs became ready, and also resulting from work in Task 3.2 with input from discussions from the previously mentioned WPs.

An issue reporting scheme was employed in two separate instances (but following a similar structure, where issues and requests were reported, indicating affected module, optional screenshot, description of the issue/request and its perceived importance):

- **Alpha:** for core testers that receive earlier updates with new functionality after preliminary testing (internal to some elements of WP3 partners); and
- **Beta:** extended to additional testers and Campaign Managers, that received updates later with the functionality after it has been provided to alpha testers with the issues found and addressed.

According to the relative priority of the issues and requests put forward, as input from the Campaign Managers and WP3 partners, assessments were then made by partner TIS (in the role of *product owner*), the addressing of the situation by INESC-ID, and final settlement/decision/resolution.

Iteratively, in parallel and based on results from trip data, work also refocused on addressing trip detection errors and the demanding accuracy/energy-efficiency tradeoffs, regarding battery consumption and strategies to further minimize drain/waste, functional bugs, application blocking, and continuous adjustment to newly found app requirements resulting from preliminary usage and testing.

This resulted in several (over 20) prototype versions of the Woorti mobile app for Android and iOS, as well as updates to backend functionality, server side data storage and analytics.

7. Concluding Remarks

The Woorti app is publicly available and ready to fulfil the objectives of the MoTiV project as well as future mobility data collection initiatives.

Technologically, it tackled several challenges related to the mobile phone-based transport detection engine, particularly the detection of trips in space and time, without resorting to any online or ulterior cloud backend

support for processing, being mostly self-reliant in the mobile phones for the purpose of trip detection and travel mode detection.

A particular innovation of the Woorti towards the quality of the collected data, is the process of user interaction and rewarding that, on the one hand, only considers user "validated" trip data for research uses, and on the other hand, creates a set of incentives for users to validate as many trips as possible, both by means of material incentives and access to information (the user is only able to access the interesting dashboard information after validating the existing trips).

Furthermore, the app design had the focus of enabling the engagement of users, creating a value proposition and features that have the potential to raise the interest and increase the engagement of the application users. Therefore, the cost-effectiveness between user recruitment and user participation is expected to be higher than in data collection campaigns that rely only on the data collection instrument but not on additional features to attract and give something in return to users.

The level of quality and reliability of the app has reached an acceptable level for most phone devices and operating system versions and is fully usable for data collection campaigns.

8. Annex - Data Structures and Algorithms

In this annex we provide more information regarding the main data structures and algorithms developed for Woorti.

8.1 Main Data Structures

Data structures in both the iOS and Android applications have been designed and implemented to facilitate the development and reduce the number of issues that may surface due to dealing with both platforms and exchanging information with the backend. Although internally they may have differences, when these structures are transmitted to and from the server they look identical. To achieve that, since the beginning, classes were implemented so that they could be sent in JSON format to the server and dealt with equally independently of the client (iOS, Android and even the backend).

In terms of local persistence, in the iOS version, all the data is stored using the Core Data framework whilst the Android version stores the data using multiple persistence storage mechanisms: SQLite, Shared Preferences and through files written on the disk.

Trip Data

A trip (class named FullTrip) is mainly composed of a list of trip parts (class FullTripPart). These trip parts include: the list of locations of the segment; information such as the initial and final locations and timestamps, activities, relaxing and productivity ratings assigned by the user upon validating the trip.

In addition to these, each specific type of trip part has additional fields:

- **Legs (Class Trip):** Mode of transport detected, corrected mode of transport, and other statistics regarding trip time duration, speed and acceleration.
- **Waiting Event:** Average location.

In the Android version, each trip is converted to JSON and written as a file in the file system.

Upon receiving the trips' data, the server takes it and stores it in a graph database, which is more suited and provides better performance when querying large amounts of trajectory data.

Survey data

Surveys are created and launched in the back office, being downloaded by the mobile applications. Data kept regarding surveys includes:

- **Questions** (of multiple types): Yes/No answer, Checkboxes, Radio buttons, Short text, Dropdown, etc.
- **Triggers:** The type of situation where the survey is presented to the user. Triggers may be timed triggers or recurring triggers.
- **Launch:** This describes when the survey is to be launched, i.e. made available to be presented to the user.

There is also a class associated with the one just described. This class is SurveyStateful and, in addition to the Survey itself, it contains the information relative to the user answering the survey: ID of the user, answer list, and the

timestamp of the answer. For each question type there is a corresponding answer type. In the Android version, this information is stored in the SQLite local database.

User data

Data about the user is stored in the app for ready access when without connectivity, besides being kept in the backend. UserSettings is composed of the user information provided upon filling the onboarding screens. Woorti also keeps information for the app to track whether the settings, when updated, have already been sent to the server.

In the Android version we use the Shared Preferences native interface which allows to easily and efficiently store simple key/value based data.

Campaign data

Woorti stores information about the ongoing campaigns. This information consists of geographical information, name, campaign ID, and the points to be awarded to the users of the campaign, based on the information they provide with validated trips. In the Android version, campaign data is stored in the SQLite local database, associated with Surveys and Rewards.

Trip state machine and machine learning mechanism data

The trip state machine/transport mode detection mechanisms are driven by a set of data structures present in the app. These comprise several classes, the most relevant being:

- **Trip State machine:** The trip state machine is the class that receives location and acceleration data and decides whether a trip has started, if a leg has ended, if a waiting event has started, or if a trip has ended. It then passes these locations and accelerations to the machine learning mechanism which will infer which were the most probable transport modes used throughout the legs.

  This class is also responsible for temporarily saving the trips' state data persistently in order not to lose all its data when there are any failures (e.g., the Operating System killing the app and bringing it back a few minutes later). The Android version stores this information in the SQLite database.

- **Package dataML:** Intermediary data structures necessary for the machine learning mechanism to infer modes of transport.

8.2 Algorithms

The Woorti mobile application incorporates two main algorithms developed inside two modules. One of them is included in Trip Detection Module (TDM), and is responsible for identifying the trip start, leg start, leg end, and trip end events. The other is implemented in the Transport Mode Detection Module (TMDM) and is responsible for the identification of the transport modes used during the trip.

Trip Detection Module

The overall algorithm running in the Trip Detection Module operates as a state-machine enforcing the following transition rules:

- While **Stopped**:
  - Turns on the GPS if the mobile phone detects that the user is moving.
  - If the mobile phone detects that the user has traveled a distance of more than **100 meters** since the last Trip, start detecting a **Leg**.
  - If the GPS is turned on, it will turn off if the trip does not start in due time (5 minutes) and remain in **Stopped**.
- While on **Leg:**
  - If the user does not travel a distance of more than **100 meters** in **5 minutes**, preserve information recorded during the Leg and proceed to detect a **Transfer**.
- While on **Transfer**:
  - If the user travels a distance of more than **100 meters** from the first location where the **Transfer** was initially detected, proceed to start detecting a new **Leg**.
  - If the user remains inside the same **100 meters,** from the first location where the **Transfer** was initially detected, for **25 minutes**, we proceed to **Stopped** and finish this trip.

These states and transitions between them are illustrated in the state machine diagram of Figure 32:

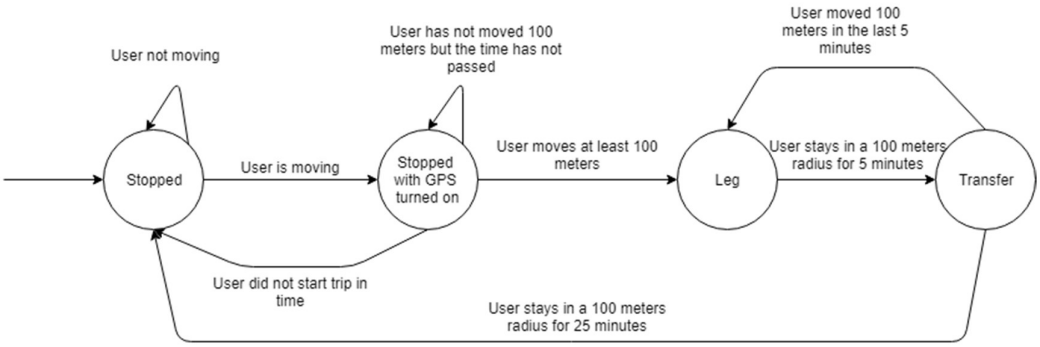


Figure 32: Trip state machine states and transitions.

Besides the overall approach described earlier, there are two detailed issues that stem from the execution in the mobile phones, which are prone to variations in resource capabilities, sensor quality (errors) and operational conditions: (i) relevant motion detection, and (ii) reliable distance measurement.

Relevant Motion Detection Heuristics

In perfect, error-free conditions, the overall algorithm in the TDM is sufficient. In real scenarios, where there are errors in sensor readings, and at times, absence of location or acceleration information, additional heuristics are required in the implementation, as addressed next.

Considering relevant motion detection, in Android and iOS the detection that a user is moving and the time it takes to turn off the GPS while on the stopped state are different. This is due to a difference in the way Android and iOS deliver Location, Acceleration and Motion information.

iOS: In iOS, to detect a user is moving when the GPS is off, we use the standard Activity detection from iOS. This activity detection notifies the app when the user has started an activity involving motion (i.e., walking). Additionally, the Activity detection notifies the app when there is no motion-related activity, but some residual acceleration was detected (e.g., stationary), and in a loosely periodic fashion (every 4 or 5 seconds). We have found these events to be related with motion with very soft accelerations but that may accumulate over time across a significant distance (typically when traveling by train where for large distances acceleration may always be very soft). Therefore, we piggyback on these events and periodically (roughly about 3 minutes) check whether the distance covered in the last 5 minutes is enough to proceed to start detecting a Leg (regardless acceleration readings).

Android: In Android, we use raw accelerometer data to check if there is a high chance that the user is in movement, by computing the average acceleration for each ten second period (checking whether it is over 2.5 m/s2, similar to walking). Additionally, to account for situations where long periods of time, with very soft acceleration, may accumulate across large distances (typically, train), we try to detect a starting trip periodically every 3 minutes (by checking whether the distance covered is over the 100 m threshold).

When it is determined there is no relevant movement, In iOS the time until we turn off the GPS is 1 minute. This is done to prevent a high use of battery when not in travel, suitable for iOS due to the GPS having a relatively higher accuracy rate. In Android, where accuracy can be significantly lower at the beginning in several models, the time to turn off the GPS is 5 minutes to reduce the probability of a false negative happening (missing a starting trip).

Relevant Distance Measurement

As mentioned earlier, inaccurate location readings from sensors (mainly when the user is stationary and inside buildings) happen frequently (particularly in Android where GPS is fused with readings from Wi-Fi access points signal strength). Acceleration readings are also subject to errors but they are more reliable.

Therefore, the thresholds to start detecting a leg may be met, in error, due to one or more spurious readings from the location sensors providing erroneous data (wrong locations and, surprisingly, wrong error bounds on the accuracy of the locations provided). This would incur in wrongly detecting the start of a leg or trip. This is challenging as trip detection is being carried out in real-time.

To address this, in order to provide the TDM algorithm with reliable distance measurement (to determine the start of a trip or leg correctly, in real-time, in spite of errors in sensor readings), we employ the following buffering and filtering scheme.

First, locations with error bound considered unacceptably high (e.g., over 200m, much higher than the threshold to determine trip start) are discarded. When locations do fall within error bounds, we still do not immediately consider them as good to calculate distances, as we deem them as *suspects,* holding them in a buffer, thus slightly delaying the decision to start a trip or leg to near real-time.

These buffered suspects are maintained until they are trusted enough to trigger the appropriate state transition. Several approaches for filtering are employed (bear in mind we must be fast and non-resource intensive as this must be run in the mobile phone), namely: when suspect locations are found to be scattered around a large area, alternating between closer and farther away locations, regarding the current user's location, they are discarded.

When such scattering is limited, the suspect locations follow a consistent near-increasing distance from the current user's location. As the number of suspects not filtered out becomes large enough to be trustworthy, e.g., 5 (as we cannot delay trip detection indefinitely), and the distance covered exceeds the threshold, a trip start (or a new leg) is determined and the locations are included in the current leg.

When, instead, it is determined that the suspects do not constitute a consistent initial path of a trip, the current state is maintained. After the time threshold for trip detection is expired, they are naturally dropped.

Transport Mode Detection Module

In Woorti, we employ machine learning algorithms to perform the identification of the transport mode. The identification of the transport mode consists in selecting a label from a set of known labels, which means it is a classification problem.

The construction of the machine learning classifier can be divided in three main steps: (i) collection of the raw data from mobile phone sensors, (ii) processing of the collected data, (iii) train the classifier using processed data. These steps are described in more details in the following subsections.

In order to be possible to identify the mode of transport in near real time, the classifier must run locally on the mobile phones (both Android and iOS operating systems).

Collection of the raw data

To collect the raw data for training, we resorted to a controlled environment, running a system composed of two main components, a central server and a prototype of the Woorti mobile phone application (in Android and iOS), for the sole purpose of collecting raw data regarding location and acceleration during controlled trips, still without transport mode detection.

The application allowed the user to start and end raw data (location and acceleration data) collection explicitly through interface buttons. During the trip, the raw data from GPS and accelerometer sensors was collected and saved in the internal memory of the mobile phone. The accelerometer data was collected once per second and the GPS data was collected every 10 seconds. These frequencies were selected in order to minimize the battery consumption. At the end of a trip, the collected data was validated by the user, and sent to the central server if correct.

In this controlled environment, data about 537 trips was collected by 15 users. The total duration of collected data is approximately 265 hours. On average, each trip has 2.1 transport mode used. **Figure 33** shows the total duration (in hours) of the raw data collected for each mode of transport.

Walking	Bicycle	Car	Bus	Train
62.07h	15.53h	69.68h	65.82h	49.92h

Figure 33: *Total duration of the raw data for each mode of transport collected*

To obtain the useful input parameters (features) for the machine learning algorithm it was necessary to process the raw data collected. The raw data procession was performed in 3 steps: (i) manual verification and filtering of the reported data, (ii) the division of the filtered data into segments of the same size, (iii) feature extraction from each of the segment.

The use of noisy data during the training phase can reduce the accuracy of the classifier. This step was performed in order to filter the data that is noisy or contains errors. For example, some of the trips were found to have been reported with wrong transport mode and these trips were thus removed from the dataset.

8.3 Data segmentation

The task of identifying the mode of transport is complicated by the fact that people can use more than one transport mode in each trip. This means that, first of all, it is necessary to find out the number of modes of transport used during a trip.

To identify when the transport mode changes, the raw data being processed must be split into segments. Otherwise, if considered as just one segment (the full raw data collected), trips without significant stops in the order of minutes (waiting events) would be classified wrongly, as comprising a single leg with just one transport mode used (typically the one used the longest, regardless of the succession set of transport modes the user may have taken). Conversely, using too short segments will incur overhead, and may exacerbate the effects of errors in the raw data.

The raw data was split into segments of fixed duration and classifier was applied on each of these segments. In this way it is possible to identify when a sequence of segments with the same transport mode is followed by another sequence with a different transport mode, which allows to identify when a change of the transport mode occurs.

To better identify when the transport mode changes, the overlap between segments has also been applied, as a sliding window. This means that a part of the raw data of one segment is also used as a part of the next segment. Figure 34 represents the possible sequence of raw data defined by accelerations (**a**) and locations (**l**) and the division of this sequence in 4 segments.

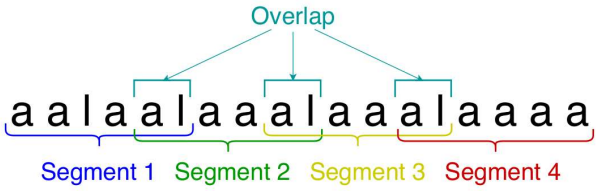


Figure 34: Example sequence of raw data divided in 4 segments

Segment data division is performed by calculating the time between two sensor readings. This calculation is possible because each raw data contains information about the timestamp when the data was obtained and the raw data was collected in chronological order. When the time between two timestamps is equal to the size of the segment, it means that all the data collected between these readings belongs to the same segment.

Experimentally, we obtained the best results using the segment size of 90 seconds and the overlap size of 45 seconds.

Feature extraction

From the raw data of each segment was calculated a set of features that describes the corresponding segment. The resulting features was used to train the machine learning algorithm.

In order to determine the mode of transport independently of the mobile phone orientation, for each accelerometer reading was calculated the correspondent magnitude (M) using the formula $M = \sqrt{x^2 + y^2 + z^2}$, where *x*, *y* and *z* represents the acceleration force along the corresponding axis.

One limitation of the GPS sensor is the loss of signal in certain locations such as tunnels, underground spaces or indoors. Without locations it becomes impossible to calculate the features related with speed and therefore there is a loss of useful information that helps to identify the mode of transport.

In total, 23 features were extracted from the raw data of each segment. The extracted features are:

- avgAccel, maxAccel, minAccel, stdDevAccel - Average, maximum, minimum and standard deviation of magnitude of acceleration.
- avgFilteredAccel - Average of magnitude of acceleration after removing the values below acceleration filter.
- accelsBelowFilter - Percentage of accelerations with values below the acceleration filter.
- accelBetw-03-06 - Percentage of accelerations with values between 0.3 and 0.6 m/s2.
- accelBetw-06-1 - Percentage of accelerations with values between 0.6 and 1.0 m/s2.
- accelBetw-1-3 - Percentage of accelerations with values between 1 and 3 m/s2.
- accelBetw-3-6 - Percentage of accelerations with values between 3 and 6 m/s2.
- accelAbove-6 - Percentage of accelerations with values above 6 m/s2.
- avgSpeed, maxSpeed, minSpeed, stdDevSpeed - Average, maximum, minimum and standard deviation of speeds.
- avgAcc, maxAcc, minAcc, stdDevAcc - Average, maximum, minimum and standard deviation of precision associated with location coordinates.
- gpsTimeMean - Mean time between 2 consecutive locations received. Calculated in seconds.
- distance - Distance traveled during the segment. Calculated in meters.
- OS - Operating system of the device where the raw data was collected (Android or iOS).
- estimatedSpeed - represents if the speed-related features were calculated using the location of the segment or were estimated using the locations of the neighbouring segments.

The intervals defined for accelerations represented by features accelsBelowFilter, accelBetw-03-06, accelBetw-06-1 were defined after analysing the acceleration data related to several modes of transport. It was observed that modes of transport such as car, bus and train tend to have low accelerations (rarely exceeded 1 m/s2. Therefore, in order to facilitate the distinction between these modes, values between 0 and 1 m/s2 were divided into 3 categories of different intervals and for each segment, the percentage of the values in each interval was calculated.

In addition, the value chosen for the accelsBelowFilter feature was 0.3 m/s2, because in the analysed trips this value represents a border below which it is possible to consider that the user is staying still and not performing any movement.

It was also observed that in general the accelerations associated with bicycle are usually below 6 m/s2, while the accelerations associated with walking may exceed this value. For this reason, it was considered that 6 m/s2 is a useful value to define a boundary between intervals and the accelAbove-6 feature was defined. Finally, it was also decided to split the interval between 1 m/s2 and 6 m/s2 in two to obtain a better separation of accelerations in this interval. As a result, features accelBetw-1-3 and accelBetw-3-6 were defined.

Training phase

The classifier was trained from the features calculated for each segment in the data processing phase.

During the training phase it was necessary to test several alternatives of classifier configurations (algorithm, features, etc.) in order to find the configuration that allows to achieve the best results.

For each feature, it was evaluated the corresponding importance on the correct transport mode identification. The importance was calculated using the RFE algorithm (Recursive Feature Elimination algorithm) and the features that worsened the results of the classifier were ignored.

The classifier was trained in Python, using the machine learning algorithms defined in scikit-learn library scikit-learn. In order to be used on Android and iOS, the resulting classifier was exported using two different formats, one for each operating system.

To be used on Android, the classifier was exported to the PMML format using the JPMML library. The JPMML library allows to export the model to PMML format and to execute this model in Java programming language that is used to create application in Android.

To be used on iOS, the classifier was exported to the Core ML model format. The Core ML framework allows to export and use the machine learning models on iOS devices.

With JPMML and Core ML it was possible to create the machine learning model in scikit-learn and then use that model on the Android and iOS operating systems.

Transport mode identification

The classifier obtained during the training phase allows to identify the transport modes in which there is movement (walking, car, train, etc.). However, people frequently stop for short periods of time during the trip (e.g., stopping at traffic lights, stopping to buy a newspaper, etc.). If the person is stopped for a period of time, the corresponding segments of raw data will be classified accordingly to the 5 modes of transport known by the classifier (walking, bicycle, car, bus and train). Since the classifier does not know the still mode, these segments will be mistakenly identified with another most probable, yet wrong, mode of transport. And the more segments are poorly identified, the more difficult it is to identify correctly the modes of transport used in a trip. Therefore, to decrease the amount of noise, besides employing the segmentation with overlapping windows, it is also necessary to identify the still mode.

The still mode was identified resorting to additional rules by inspecting the data. This decision was made for two main reasons. First, these small stops are usually not reported by the users. For example, from the point of view of the user, being stopped at a traffic light makes part of riding a car, so it is expected the user to report only the car mode of transport.

Second, the still mode can be easily confused by the classifier with other modes of transport. The train, subway or even the car has relatively low accelerations as well as the still mode and when the location information does not exist or is less accurate, the classifier may fail more easily to identify the correct mode of transport.

Additionally, it is relatively easy to set the values of the still mode boundaries, because standing means that speeds and accelerations are low.

The still mode was identified by checking the values of 2 features, namely the avgSpeed and accelsBelowFilter. Then the value of avgSpeed is less than or equal to 2.5 km/h and the value of accelsBelowFilter is greater than or equal

to 0.8, the transport mode of the segment is considered to be still. If the transport mode is not still, the segment is evaluated by the classifier accordingly to the 5 modes of transport known by the classifier.

In this way, the TMDM allows to identify 6 modes of transport (walking, bicycle, car, bus, train and still).

Finally, when the still mode is detected covering a significant distance enough to be reported as a leg, this means that there was unequivocal movement, but the classifier was unable to detect the transport mode used. From the raw data collected, we have found empirically that this happens more often when the train is being used. Therefore, another rule is applied, and the mode is converted to train in such legs.

8.4 Final remarks

Globally, the proposed objective of performing local (on the mobile phone) and near-real time detection of trips and travel mode, without resorting to any online or ulterior cloud backend support for processing, is very challenging, regarding error correction, and resource intensive in mobile phones with more limited capabilities.

Conversely, it has two significant advantages. First, it relies solely on the execution at the mobile phone, being prompt and resilient, even when there is no connectivity, and even if the backend would be inoperative for a period of time. Second, it allows to distribute most of the workload at the edges of the network, leveraging the relevant resources already available in many phones, while reducing the load imposed on the backend, and avoiding the need for a more expensive and complex up-start computational infrastructure to deploy it.

Mobile phone operating systems have very stringent constraints (namely iOS that shuts down applications using too much CPU over often short periods of time) and often high variability in behaviour and reliability across models. In particular, in Android, localization sensors often provide wrong readings and with wrong error estimations (bounds) that make filtering more difficult, also varying across devices.

Additionally, such errors in location information deceive the perceived speeds (that are a key input to the classification). When combined with very small or missing acceleration readings, this contributes to wrong results when classifying transport modes. Due to near real-time and battery usage requirements, the filtering approach in the app is able to detect outliers but with moderate efficacy, negotiating a difficult trade-off between correction (no false positives, i.e. no fake trips) and completeness (no false negatives, i.e. no trips undetected).

Finally, regarding mode detection, this is often further hampered by travelling with the mobile phone in bags, softening acceleration and hindering localization information, handling phones and changing relative device orientation/attitude during trips (e.g., walking and talking), and the frequent absence of localization information coverage on tunnels, namely the subway. This poses additional challenges when aiming at detecting trips and modes of transport in real-time without resorting to the cloud and further processing based on street and route maps.