

Not approved yet



## Autonomous Vehicles to Evolve to a New Urban Experience

---

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769033

### DELIVERABLE

#### D4.9 Third-iteration Out-of-vehicle services

# Disclaimer

This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

## Document Information

Grant Agreement Number	769033
Full Title	Autonomous Vehicles to Evolve to a New Urban Experience
Acronym	AVENUE
Deliverable	D4.9 Third iteration Out-of-vehicle services
Due Date	31.08.2021
Work Package	WP4
Lead Partner	MobileThinking
Leading Authors	Kevin Salvi, Mattia Gustarini
Contributors	Hassan Helbawi
Dissemination Level	Public

## Document History

Version	Date	Author	Entity	Description of change
0.1	21.06.21	Kevin Salvi, Hassan Helbawi	MobileThinking	first version of D4.9 out-of-vehicle services - Structure creation
0.2	26.07.21	Mattia Gustarini	MobileThinking	Initial list of achievements
0.3	09.08.21	Kevin Salvi, Hassan Helbawi	MobileThinking	Completed the first draft of document for technical sections
0.4	13.08.21	Kevin Salvi	MobileThinking	Reviewed and commented version of Deliverable
1.0	11.10.21	Kevin Salvi, Hassan Helbawi, Mattia Gustarini	MobileThinking	First final draft

# Table of Contents

List of Figures	6
Executive Summary	9
1 Introduction	9
1.1 On-demand Mobility	10
1.2 Fully Automated Vehicles	10
1.2.1 Automated vehicle operation overview	11
1.2.2 Automated vehicle capabilities in AVENUE	12
1.3 Preamble	12
1.4 Structure of the deliverable	13
2 Software components	13
2.1 Application Backend	13
2.1.1 The Application backend objectives	14
2.1.2 The Application Backend architecture	15
2.1.2.1 Architecture overview	15
2.1.2.2 Core packages	17
2.1.3 Application Backend software updates	17
2.1.3.1 API V1	17
2.1.3.2 New API V2	17
List of all available vehicles	17
List of all available Stops	18
List of the lines	18
List of services	18
Get the service by Id	19
Get the stops of the service	19
2.1.3.3 Notifications features	19
2.1.3.4 Server Maintenance	19
2.2 Traveler APP	20
2.2.1 Main services	20
2.2.1 iOS and Android native	20
2.2.2 UI/UX	21
Ux / UI Design Improvements	21
New icons	21
Quote selection process	21
On-demand traveling	23
2.2.3 Accessibility	24
2.2.3.1 Analysis	24



2.2.3.2	Work done	24
2.2.3.3	Next Step	26
2.2.4	PTO versions	26
2.2.4.1	Two applications for each PTO	26
2.2.5	Translations	28
2.2.5.1	Work done	28
3	Out-of-vehicle services	29
3.1	Passenger presence	29
3.1.1	Initial vision	29
3.1.2	Work done	29
3.1.3	Next steps	30
3.2	On-Demand Real-time visualization	31
3.2.1	Initial vision	31
3.2.2	Work done	31
3.2.3	Next steps	34
3.3	On-demand stop	34
3.3.1	Initial vision	34
3.3.2	Work done	35
3.4	Automatic trip planning suggestion	35
3.4.1	Initial vision	35
3.4.2	Work done	35
3.4.3	Next steps	35
3.5	Trip planned via call centers	35
3.5.1	Initial vision	36
3.5.2	Work done	36
3.5.3	Next steps	36
3.6	Digital or human information points	36
3.6.1	Initial vision	36
3.6.2	Work done	37
3.6.3	Next steps	37
3.7	On-demand zone	37
3.7.1	Initial vision	37
3.7.2	Work done	37
3.7.3	Next steps	39
3.8	Single button vehicle calls and help request	39
3.8.1	Initial vision	39
3.8.2	Work done	39
3.8.3	Next steps	41

3.9	Online ticketing services / Convoy service	42
3.9.1	Initial vision	42
3.9.2	Work done	42
3.9.3	Next steps	42
3.10	On-demand booking	42
3.10.1	Initial vision	42
3.10.2	Work done	43
	Booking	46
	Journey	47
	Ride	47
3.10.3	Next steps	55
3.11	User feedback	56
3.11.1	Initial vision	56
3.11.2	Work done	56
3.11.3	Next steps	58
3.12	Follow my kid - Out of vehicle version	58
3.12.1	Initial vision	58
3.12.2	Work done	58
3.12.3	Next steps	61
3.13	Notifications	61
3.13.1	Initial vision	61
3.13.2	Work done	62
3.13.3	Next steps	65
3.14	POI	65
3.15	Initial vision	65
3.16	Work done	65
3.17	Next steps	66
4	Operational work	67
4.1	Maintenance	67
4.2	Bug Correction	67
4.3	On-site tests	67
5	Conclusion	67
6	Next steps	67

# List of Figures

Figure 1 ©2020 SAE International	11
Figure 2 The interaction of the AVENUE Application Backend with the world.	15
Figure 3 Example of a data request flow from an AVENUE application	15
Figure 4 The main internal components of the AVENUE Application Backen	16
Figure 5 Old map icons compared to new map icons	Error! Bookmark not defined.2
Figure 6 Trip recap page	Error! Bookmark not defined.3
Figure 7 Quote card update	Error! Bookmark not defined.4
Figure 8 Screenshots of the zooms depending on the booking status	25
Figure 9 Trip options with talk back	26
Figure 10 Basic style vs accessible style	26
Figure 11 Before after text resize	27
Figure 12 Traveler App' PTO versions	28
Figure 13 Passenger presence - Traveler App'	31
Figure 14 Passenger presence next step - AVENUE services software ecosystem	32
Figure 15 Real-time visualization - Traveler App Fixed Mode'	33
Figure 16 Real-time visualization -Traveler On Demand Mode	34
Figure 17 Real-time visualization -AVENUE services software ecosystem	34
Figure 18 Real-time visualization - AVENUE services software ecosystem	36
Figure 19 On-demand zone - Bestmile's dashboard	4Error! Bookmark not defined.
Figure 20 On-demand zone - Traveller App' Map	41
Figure 21 Single button vehicle calls and help request - Traveler App' button display4Error! Bookmark not defined.	
Figure 22 Single button vehicle calls and help request - Traveler App' settings screen	43
Figure 23 Single button vehicle calls and help request - AVENUE services software ecosystem	44
Figure 24 Fixed-line vs On-demand booking	45
Figure 25 - Choose a destination - On-demand booking	46
Figure 26 On-demand booking demonstration video	47
Figure 27 Screenshots of the map zooms depending on the booking status	53
Figure 28 Cancel a bookings	54
Figure 29 On demand booking statuses diagram	55
Figure 30 On-demand booking - Quote selection process new designs	56
Figure 31 On-demand booking - Quote selection process new designs	56
Figure 32 On-demand history list	57
Figure 33 On-demand history detail	58
Figure 34 Next steps - On-demand accessibility improvements	59
Figure 35 User feedbacks - Services idea votes results	60
Figure 36 User feedback idea planning	60
Figure 37 Follow my kid - share user's trip	62
Figure 38 Follow my kid - share trip button	63
Figure 39 Follow my kid - Following a trip	64
Figure 40 Notification - Asking background location permission	66
Figure 41 Notification - Asking notification permission for boooing status update	66

Figure 42 On-demand Notification of availability of the service	68
Figure 43 On-demand Notification of the traveling status	69
Figure 44 Point of Interest screen	70

## Acronyms

ADS	Automated Driving Systems	GNSS	Global Navigation Satellite System
AI	Artificial Intelligence	HARA	Hazard Analysis and Risk Assessment
AM	Automated Mobility	IPR	Intellectual Property Rights
API	Application Protocol Interface	IT	Information Technology
AV	Automated Vehicle	ITU	International Telecommunications Union
BM	Bestmile	LA	Leading Author
BMM	Business Modelling Manager	LIDAR	Light Detection And Ranging
CAV	Connected and Automated Vehicles	MEM	Monitoring and Evaluation Manager
CB	Consortium Body	MT	MobileThinking
CERN	European Organization for Nuclear Research	OCT	General Transport Directorate of the Canton of Geneva
D7.1	Deliverable 7.1	ODD	Operational Domain Design
DC	Demonstration Coordinator	OEDR	Object And Event Detection And Response
DI	The department of infrastructure (Swiss Canton of Geneva)	OFCOM	(Swiss) Federal Office of Communications
DMP	Data Management Plan	PC	Project Coordinator
DSES	Department of Security and Economy - Traffic Police (Swiss Canton of Geneva)	PEB	Project Executive Board
DTU test track	Technical University of Denmark test track	PGA	Project General Assembly
EAB	External Advisory Board	PRM	Persons with Reduced Mobility
EC	European Commission	PSA	Group PSA (PSA Peugeot Citroën)
ECSEL	Electronic Components and Systems for European Leadership	PTO	Public Transportation Operator
EM	Exploitation Manager	PTS	Public Transportation Services
EU	European Union	QRM	Quality and Risk Manager
EUCAD	European Conference on Connected and Automated Driving	QRMB	Quality and Risk Management Board
F2F	Face to face meeting	RN	Risk Number
FEDRO	(Swiss) Federal Roads Office	SA	Scientific Advisor
FOT	(Swiss) Federal Office of Transport	SAE Level	Society of Automotive Engineers Level (Vehicle Autonomy Level)
GDPR	General Data Protection Regulation	SAN	(Swiss) Cantonal Vehicle Service
GIMS	Geneva International Motor Show	SDK	Software Development Kit
		SLA	Sales Lentz Autocars
		SMB	Site Management Board

SoA	State of the Art
SOTIF	Safety Of The Intended Functionality
SWOT	Strengths, Weaknesses, Opportunities, and Threats.
T7.1	Task 7.1
TM	Technical Manager
TPG	Transport Publics Genevois Union Internationale des
UITP	Transports Publics (International Transport Union)
URI	Uniform Resource Identifier
V2I	Vehicle to Infrastructure communication
WP	Work Package
WPL	Work Package Leader



# Executive Summary

This is the third version of T4.3 Out-of-vehicle services deliverable which is due at month M44. Its main objective is to describe the new development done for all the services, more precisely the updates brought to the services since the delivery of the document D4.8 and the new services developed. As a reminder, these services are used by users mainly outside of the AVs (autonomous vehicles) although some will find their use inside an Av.

The list of services established with the whole consortium and quoted in document D4.8 has been enriched. The following out-of-vehicle services have been added, they are fully or partially developed:

- Visualization in real time of the path/position of shuttle
- On-demand stop
- On-demand booking
- On-Demand zone
- Follow my kid
- Notification

The purpose of this document is to highlight the state of the art of the existing services, to describe in detail the updates made and the new features.

## 1 Introduction

AVENUE aims to design and carry out full-scale demonstrations of urban transport automation by deploying, for the first time worldwide, fleets of Automated minibuses in low to medium demand areas of 4 European demonstrator cities (Geneva, Lyon, Copenhagen and Luxembourg) and 2 to 3 replicator cities. The AVENUE vision for future public transport in urban and suburban areas, is that Automated vehicles will ensure safe, rapid, economic, sustainable and personalised transport of passengers. AVENUE introduces disruptive public transportation paradigms on the basis of on-demand, door-to-door services, aiming to set up a new model of public transportation, by revisiting the offered public transportation services, and aiming to suppress pre scheduled fixed bus itineraries.

Vehicle services that substantially enhance the passenger experience as well as the overall quality and value of the service will be introduced, also targeting elderly people, people with disabilities and vulnerable users. Road behaviour, security of the Automated vehicles and passengers' safety are central points of the AVENUE project.

At the end of the AVENUE project four-year period the mission is to have demonstrated that Automated vehicles will become the future solution for public transport. The AVENUE project will demonstrate the economic, environmental and social potential of Automated vehicles for both companies and public commuters while assessing the vehicle road behaviour safety.

## 1.1 On-demand Mobility

Public transportation is a key element of a region's economic development and the quality of life of its citizens.

Governments around the world are defining strategies for the development of efficient public transport based on different criteria of importance to their regions, such as topography, citizens' needs, social and economic barriers, environmental concerns and historical development. However, new technologies, modes of transport and services are appearing, which seem very promising to the support of regional strategies for the development of public transport.

On-demand transport is a public transport service that only works when a reservation has been recorded and will be a relevant solution where the demand for transport is diffuse and regular transport is inefficient.

On-demand transport differs from other public transport services in that vehicles do not follow a fixed route and do not use a predefined timetable. Unlike taxis, on-demand public transport is usually also not individual. An operator or an automated system takes care of the booking, planning and organization.

It is recognized that the use and integration of on-demand Automated vehicles has the potential to significantly improve services and provide solutions to many of the problems encountered today in the development of sustainable and efficient public transport.

## 1.2 Fully Automated Vehicles

A self-driving car, referred to in the AVENUE project as a **Fully Automated Vehicle (AV)**, also referred as Autonomous Vehicle, is a vehicle that is capable of sensing its environment and moving safely with no human input.

The terms *automated vehicles* and *autonomous vehicles* are often used together. The Regulation 2019/2144 of the European Parliament and of the Council of 27 November 2019 on type-approval requirements for motor vehicles defines "automated vehicle" and "fully automated vehicle" based on their autonomous capacity:

- An "automated vehicle" means a motor vehicle designed and constructed to move autonomously for certain periods of time without continuous driver supervision but in respect of which driver intervention is still expected or required
- "fully automated vehicle" means a motor vehicle that has been designed and constructed to move autonomously without any driver supervision

In AVENUE we operate **Fully Automated minibuses for public transport**, (previously referred as Autonomous shuttles, or Autonomous buses), and we refer to them as simply *Automated minibuses* or *the AVENUE minibuses*.

In relation to the SAE levels, the AVENUE project will operate SAE Level 4 vehicles.



## SAE J3016™ LEVELS OF DRIVING AUTOMATION

	SAE LEVEL 0	SAE LEVEL 1	SAE LEVEL 2	SAE LEVEL 3	SAE LEVEL 4	SAE LEVEL 5
What does the human in the driver's seat have to do?	You <u>are</u> driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering  You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			You are <u>not</u> driving when these automated driving features are engaged – even if you are seated in "the driver's seat"  When the feature requests, you must drive  These automated driving features will not require you to take over driving		
What do these features do?	These are driver support features  These features are limited to providing warnings and momentary assistance  These features provide steering OR brake/acceleration support to the driver  These features provide steering AND brake/acceleration support to the driver			These are automated driving features  These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met  This feature can drive the vehicle under all conditions		
Example Features	<ul style="list-style-type: none"> <li>• automatic emergency braking</li> <li>• blind spot warning</li> <li>• lane departure warning</li> </ul>	<ul style="list-style-type: none"> <li>• lane centering OR</li> <li>• adaptive cruise control</li> </ul>	<ul style="list-style-type: none"> <li>• lane centering AND</li> <li>• adaptive cruise control at the same time</li> </ul>	<ul style="list-style-type: none"> <li>• traffic jam chauffeur</li> </ul>	<ul style="list-style-type: none"> <li>• local driverless taxi</li> <li>• pedals/steering wheel may or may not be installed</li> </ul>	<ul style="list-style-type: none"> <li>• same as level 4, but feature can drive everywhere in all conditions</li> </ul>

### 1.2.1 Automated vehicle operation overview

We distinguish in AVENUE two levels of control of the AV: micro-navigation and macro-navigation. Micro navigation is fully integrated in the vehicle and implements the road behaviour of the vehicle, while macro-navigation is controlled by the operator running the vehicle and defines the destination and path of the vehicle, as defined in the higher view of the overall fleet management.

For micro-navigation, Automated Vehicles combine a variety of sensors to perceive their surroundings, such as 3D video, LIDAR, sonar, GNSS, odometry and other types of sensors. Control software and systems, integrated in the vehicle, fusion and interpret the sensor information to identify the current position of the vehicle, detecting obstacles in the surrounding environment, and choosing the most appropriate reaction of the vehicle, ranging from stopping to bypassing the obstacle, reducing its speed, making a turn etc.

For the Macro-navigation, that is the destination to reach, the Automated Vehicle receives the information from either the in-vehicle operator (in the current configuration with a fixed path route), or from the remote control service via a dedicated 4/5G communication channel, for a fleet-managed operation. The fleet management system takes into account all available vehicles in the services area, the passenger request, the operator policies, the street conditions (closed streets) and send route and stop information to the vehicle (route to follow and destination to reach).



## 1.2.2 Automated vehicle capabilities in AVENUE

The Automated vehicles employed in AVENUE fully and automatically manage the above defined, micro-navigation and road behaviour, in an open street environment. The vehicles are Automatically capable to recognise obstacles (and identify some of them), identify moving and stationary objects, and Automatically decide to bypass them or wait behind them, based on the defined policies. For example with small changes in its route the AVENUE mini-bus is able to bypass a parked car, while it will slow down and follow behind a slowly moving car. The AVENUE mini-buses are able to handle different complex road situations, like entering and exiting round-about in the presence of other fast running cars, stop in zebra crossings, and communicate with infrastructure via V2I interfaces (ex. red light control).

The mini-buses used in the AVENUE project technically can achieve speeds of more than 60Km/h. However this speed cannot be used in the project demonstrators for several reasons, ranging from regulatory to safety. Under current regulations the maximum authorised speed is 25 or 30 Km/h (depending on the site). In the current demonstrators the speed does not exceed 23 Km/h, with an operational speed of 14 to 18 Km/h. Another, more important reason for limiting the vehicle speed is safety for passengers and pedestrians. Due to the fact that the current LIDAR has a range of 100m and the obstacle identification is done for objects no further than 40 meters, and considering that the vehicle must safely stop in case of an obstacle on the road (which will be “seen” at less than 40 meters distance) we cannot guarantee a safe braking if the speed is more than 25 Km/h. Note that technically the vehicle can make a harsh break and stop with 40 meters in high speeds (40 -50 Km/h) but then the break would be too harsh putting in risk the vehicle passengers. The project is working in finding an optimal point between passenger and pedestrian safety.

Due to legal requirements a **Safety Operator** must always be present in the vehicle, able to take control any moment. Additionally, at the control room, a **Supervisor** is present controlling the fleet operations. An **Intervention Team** is present in the deployment area ready to intervene in case of incident to any of the minibuses.

## 1.3 Preamble

**This section remains unchanged from the previous document as it describes the purpose of the project which is immutable.**

**The AVENUE project** is set up to offer on demand door-to-door solutions integrated within existing public transportation services, and evaluates the feasibility of operating automated shuttles with routes and schedules based on real-time passenger demand, instead of following fixed itineraries and predetermined timetables.

AVENUE’s objective is to showcase these customized transport solutions at demonstrator sites in Copenhagen, Geneva, Luxembourg and Lyon, and later duplicate them in several other European cities.

Work package **WP4** aims to design, develop, adapt and integrate services to support users of automated vehicles before the trip, during the trip, and at the end of the trip. The main objective of WP4 is to provide services in order to demonstrate that the user experience can be seamless and secure, and that people embrace this new technology. Hence, we have to include the following topics:

- Adapt and integrate existing transport services
- Develop automated vehicle specific services
- Provide services that foster the acceptance of driverless vehicles by both passengers and people interacting with the shuttles
- Introduce safety related services

The target of **task T4.3** is to develop, test and integrate innovative out-of-vehicles services, in collaboration with the four operators in Lyon, Luxembourg, Geneva and Copenhagen, respectively. The out-of-vehicle services should in combination with the in-vehicle services support a holistic service for travellers commuting with the automated vehicles.

Out-of-vehicle services are services developed to improve the user experience when travelling with automated vehicles. The services are user-centric and focus on supporting travellers with smart solutions both *before* and *after* they use the automated vehicle.

## 1.4 Structure of the deliverable

This deliverable is structured in three main sections:

1. An introduction
2. A description of the different software components and methods used in the project with an emphasis on accessibility
3. A extended description of the out-of-vehicle services implementation and status
4. A description of the operational work done since the last deliverable
5. A conclusion and wrap-up section together with a set of next steps concludes the deliverable.

# 2 Software components

In this document we only describe the Application Backend with an emphasis on the v2 of it as well as all the improvements and new features of the Traveler App. The driver App has been abandoned as it's functionalities are now replaced by automated and more advanced ones (Bestmile's Driver App for OnDemand).

We use the services provided by Bestmile documented at the following url <https://api.bestmile.com/documentation>

## 2.1 Application Backend

Between D4.8 and D4.9 the definitions of the Application Backend software component hasn't changed. For convenience, we repeat it's main definitions. The chapter 2.1.3 Application Backend Software update is entirely new in the Deliverable iteration.

### 2.1.1 The Application backend objectives

The main task of the Application Backend is to aggregate data from different sources and provide a standard API to redistribute the data to different clients. The different sources are providers such as Bestmile that can provide information on an on demand trip or Navya providing vehicle data directly. The system should allow to replace both sources Navya and Bestmile with other providers for the same data or even augment the capabilities with new sources of data. Everything must be transparent for the clients using the standard API that the Application Backend exposes to them.

Concretely, a client can be a mobile application that allows the users of a specific transport company to check the status of an on demand service and book on demand trips. The transport company owns automated vehicles of manufacture X and uses the external service provider Y to manage its fleet. In practice, service provider Y can schedule on demand trips on behalf of the transport company with vehicles provided by manufacturer X. The Application Backend allows the transport company to aggregate the data of companies X and Y to offer a unified experience to their users via a single mobile application. However, in a second moment the transport company decides to buy new vehicles from a new company Z and to manage this new fleet by themselves. The Application Backend supports the integration of the new data providers and acts as a bridge so that the client application can offer the new vehicles and the new trip management to the end users transparently. In principle, the mobile application developers would not need to modify the application to add the new configuration.

To summarize the Application Backend objectives are:

1. **To aggregate data and functionalities of different data and service providers.** For examples, as we stated above, aggregate data from vehicles of company X and service provider Y, but also and very importantly allow ourselves to create new services able to enrich the data to create innovative services;
2. **To expose via a standard API aggregate data and functionalities of point 1 to provide augmented services to end users of PTO (Public Transport Operator) or any actor in the mobility ecosystem.** For example, the developers of a public transport application using the API (Application Protocol Interface) should be able to develop that application as a: web desktop app, PWA, mobile hybrid app, mobile native app' - iOS and Android, and more;
3. **To be multi-tenant and manage different configurations of data and services providers depending on the needs of the PTO or other mobility actors using the standard API.** For example, operator A wants to use vehicles of company Y and manage its fleet with service provider Z. Operator B wants to do the same, but with vehicles of company J and service provider K. On the Application Backend there will be two configurations one for operator A and one for operator B both accessing the same standard API (the same endpoint) but receiving data and providing services via their own registered data and services providers;
4. **To transparently swap and/or integrate new data and service providers depending on the needs of the PTO or other mobility actors while in operation without the need of changing the standard API** (of course, we can version and upgrade the API if we integrate new data or new services from scratch). The example is similar to the above scenario, but the important point is to avoid having to republish or update the client applications when there are underlying changes to service and data providers for a given operator, so everything happens transparently for the end users and greatly facilitates the maintenance of the application itself.



5. **To store the data aggregations in a common place.** For example, to use this data to perform analysis to improve the performance of the system as a whole or offer to PTOs and service/data provides a centralised entity to perform statistical analysis for their own interest.

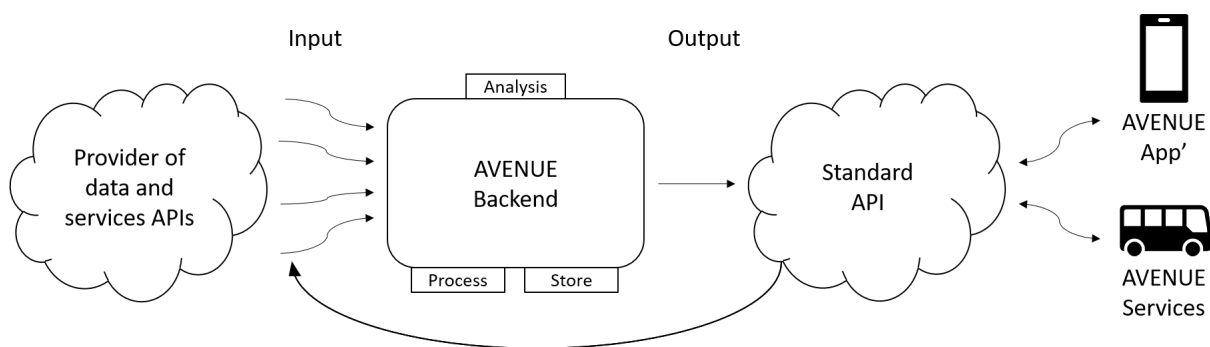
Next we will explain how we dealt with each of these objectives and what is currently in place to achieve the goal of providing innovative services to the end users of AVs.

## 2.1.2 The Application Backend architecture

To tackle the challenges of the Application Backend objectives we decided to use the Laravel framework<sup>1</sup>. It is based on php, fully open source, long term maintained, fully stable, has a big community behind, a lot of open source modules to fulfill many complexities, and is usable for enterprise applications. Laravel allowed us to create a modular Application Backend structure. Each module is a package that can be deployed in any Laravel instance making our work for AVENUE generic and usable in other unrelated projects having the same challenges (our plan is to publish the packages as open source projects to the Laravel community).

### 2.1.2.1 Architecture overview

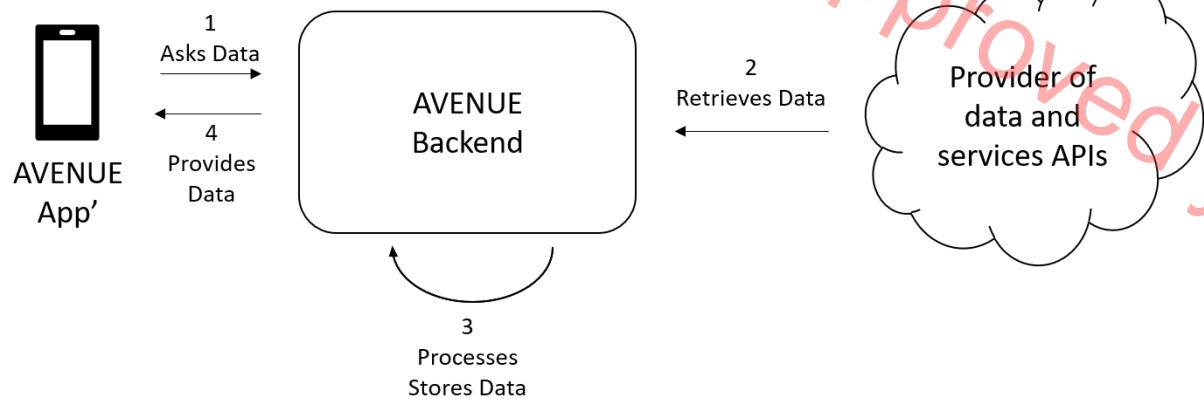
The Application Backend fundamentally connects the AVENUE applications and services to a heterogeneous set of data and service providers (for example Bestmile services, Navya vehicles or services, or even AVENUE apps and internal services). To achieve this it is able to retrieve data via APIs, process it internally, store it if needed, and offer it via a standard API to AVENUE applications and services. You can see this simple data flow in the schema in Figure 2. To provide a clearer picture of what is the role of the AVENUE Application Backend we provide a schema in Figure 3 that shows an example of an execution of a request for data from an AVENUE application to a specific data provider with the AVENUE Application Backend acting as a broker.



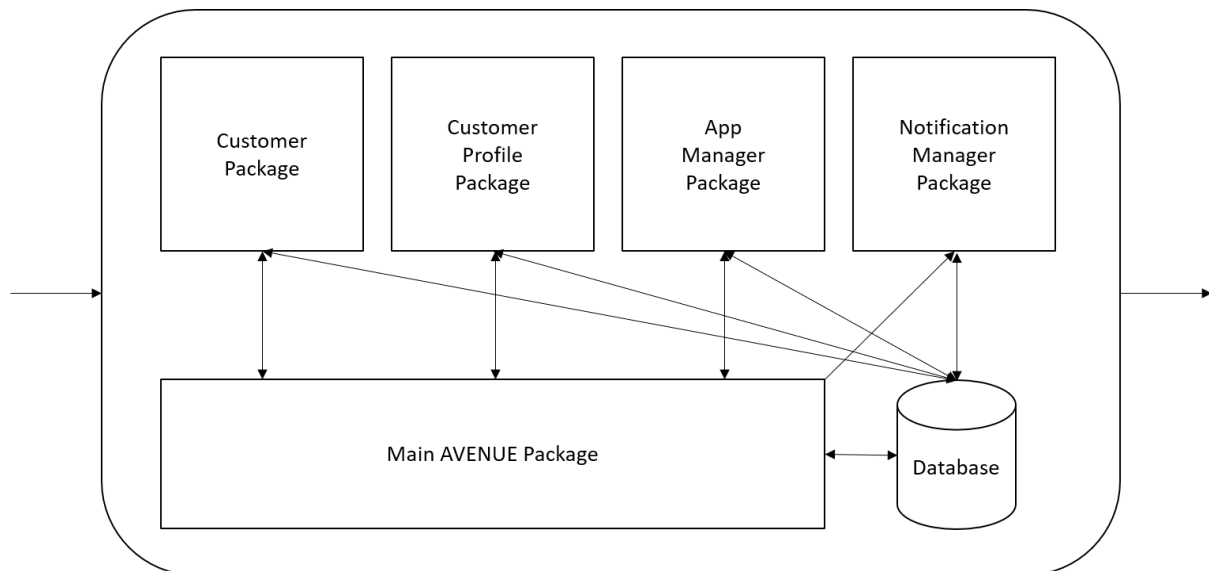
*INPUTS: are any data that the AVENUE Application Backend can retrieve via providers of data and services with an API available and AVENUE applications and services implementing the standard AVENUE API.*

*OUTPUTS: all the data offered from the AVENUE Application Backend to the AVENUE applications and services via the standard AVENUE API.*

<sup>1</sup> <https://laravel.com/>



The AVENUE Application Backend contains 4 generic packages enabling the majorities of its functionality, one main package and a database. This is a very high level schematization, in reality there are further complexities, but they are not relevant to understand the overall Application Backend architecture. In Figure 4 we provide a schema of the internals of the Application Backend and how they interact within each other.



In the next sections we detail each of the 4 generic packages and their role as well as the main AVENUE package.



### 2.1.2.2 Core packages

At the moment we have defined and developed 4 generic packages that used together with the main package specific to AVENUE (c.f. section Main package) allow to cover the 5 objectives we depicted above achieving a decoupled architecture that is flexible, extensible, and reusable. We developed all the packages in the context of AVENUE, but they are generic and usable elsewhere.

The Core packages definition will not be redescrbed in this version of the document as it is already done in D4.8 in the following sections:

- Customer package - Chap 2.1.2.2.1
- Customer Profile package - Chap 2.1.2.2.2
- API Manager package - Chap 2.1.2.2.3
- Notification Manager package - Chap 2.1.2.2.4
- Main package - Chap 2.1.2.2.5

## 2.1.3 Application Backend software updates

### 2.1.3.1 API V1

We've completed the implementation of the API V1 for the mobile applications for the on-demand service based on the API V1 of Bestmile for their booking system (testing on Bestmile's simulator).

### 2.1.3.2 New API V2

Following tests on their simulator and real life tests, we raised issues related to the use of the on demand service. In response, Bestmile has completely redesigned the API and provided a new version of its API. To get there, we were in interaction with Bestmile to define and redefine the api. This new version required almost a full rewrite of the API of the Application Backend.

In this section, we describe the features by describing the related services and entities that are being acquired from Bestmile.

Based on the new API developed by Bestmile for their booking system (testing on Bestmile simulator and at Belle Idee with TPG), we refactored the API V1 to V2 for the mobile applications for the on-demand service. This step required almost a full rewrite of the API V1 of the application backend. Further improvements were needed following tests on the real Belle idee site. The improvements were two folds technical (to introduce extra functionalities to the app, improve the performances, and manage changes on the booking system) and functional to adjust the application UI interface following users feedback.

The Application Backend has been refactored accordingly, thanks to bestmile documentation available at the time of writing at this URL: <https://api.bestmile.com/documentation>.

The new Api Calls for the fixed line service are described as the following:

#### List of all available vehicles

The Application Backend calls these endpoints from bestmile:

1. <https://api.bestmile.com/documentation#get-/transportation/v1/services>

The first call is required to get the services of the pto and query the vehicles of the desired service.

2. <https://api.bestmile.com/documentation#get-/transportation/v2/services/-ServiceID-/vehicles>

The entity returned is an array of vehicles described as the following *JSON* object:

- id: string
- name: string
- service\_id: string
- lat: number
- lon: number
- bearing: number
- occupancy: number
- capacity: number

### List of all available Stops

The Application Backend calls these endpoints from bestmile:

1. <https://api.bestmile.com/documentation#get-/transportation/v1/services>
2. <https://api.bestmile.com/documentation#get-/transportation/v2/services/-ServiceID-/stops>

The entity returned is an array of stops described as the following *JSON* object:

- id: string
- name: string
- service\_id: string
- lat: number
- lon: number

### List of the lines

The Application Backend calls these endpoints from bestmile:

1. <https://api.bestmile.com/documentation#get-/transportation/v1/services>
2. <https://api.bestmile.com/documentation#get-/transportation/v2/services/-ServiceID-/lines>

The entity returned is an array of lines described as the following *JSON* object:

- id: string
- service\_id: string
- color: string
- geometry: GeoJSONObject

### List of services

The Application Backend calls these endpoints from bestmile:

1. <https://api.bestmile.com/documentation#get-/transportation/v1/services>

The entity returned is an array of services described as the following *JSON* object:

- id: string
- name: string,
- timing: ServiceTiming,
- circular\_area: (can be null)
  - center: Array<number>,
  - radius: number

With ServiceTiming this an enum type of the following strings:

- OnDemand
- FrequencyBased
- TimetableBased

### Get the service by Id

The Application Backend calls these endpoints from bestmile:

1. <https://api.bestmile.com/documentation#get-/transportation/v1/services>
2. <https://api.bestmile.com/documentation#get-/transportation/v2/services/-ServiceID-/area>

Returns the desired service corresponding to the given id.

### Get the stops of the service

The Application Backend calls these endpoints from bestmile:

1. <https://api.bestmile.com/documentation#get-/transportation/v1/services>
2. <https://api.bestmile.com/documentation#get-/transportation/v2/services/-ServiceID-/stops>

Returns a list of the stops that are bound to the provided service.

Addition of all the previous APIs, we have also implemented the possibility to share and follow a trip in which a user is not the creator. We've adapted the call to get the booking in order to mention if it's a read only booking.

## 2.1.3.3 Notifications features

In order to inform travelers of their trip status in case they are not on the app, we implemented a push notification system. Provided that the user accepts the notifications, a device token, which identifies the smartphone, will be sent to the server in order to register it in the list of devices that accept to receive notifications. Our server takes care of the distribution of the push notifications through google's third party service called Firebase. This also required managing multi-brand applications to send notifications to the right application in case the user installed several Traveller Apps from different PTOs.

A new Api call has been added that allows the user to register to server notifications. The phone of the user sends his unique identifier (token) to the server which will register it. To notify the travelers about their trip status, we have integrated third party services Google, Firebase and Apple to the server. This also required the handling of multi branding applications to send to the correct application and user the notifications.

Other improvements were necessary following the test on the real site of Belle idee. The improvements were twofold:

1. technical (to introduce additional functionalities into the application, improve performance and manage changes to the booking system)
2. functional (to adjust the application's user interface following user feedback).

## 2.1.3.4 Server Maintenance

Since the deliverable 4.8, the laravel community has updated the technology of 2 versions. Thus, we maintained the server technology up to date.

## 2.2 Traveler APP

The traveler Application is core to the WP4 work. A lot of services that will be provided to the traveler are based on a mobile application. Of course, most of them should also be available for people without smartphones. An emphasis on accessibility has been made.

The traveler application is not a service per say, but represents the backbone of most services. A great effort has been made to implement a robust and easily improvable application.

The following chapters are describing the main work done regarding the application. Some of the content might be copied from D4.8 but only if we believe it serves to put some new work in context.

### 2.2.1 Main services

We mainly focused the development on the implementation of the on demand service and the follow my kid service. Moreover, after some alpha testing with the PTOs for the UX and Siemens for accessibility we've made some changes on the Traveller Applications.

#### 2.2.1 iOS and Android native

Nothing has changed regarding the main technologies of the Traveler Application since D4.8. They are still developed using Native Android and iOS languages. As a mobile development company, MobileThinking strongly recommended this choice for the following reasons.

##### Better performance

When we analyze the performance of hybrid applications compared to native applications, it is clear that native applications are faster. They are built with the framework that is native to the platform.

##### Third party dependencies

Using a hybrid approach would bind the entire AVENUE project to a third party company. Even if now they have proven stability, doing so is too risky for a project of the scale/lengths of the AVENUE project.

##### Data protection

It is much easier to secure a native application. In our opinion, it is such an important point that by itself is enough to take the decision of using native languages.

##### Functionalities

Using native languages improves the ability to connect the devices hardware features and databases. With hybrid languages, it would be possible but only by using external plugins/libraries.

##### User experience

Users will have a much better user experience in terms of performances and fluidity. In addition some information can be accessible offline.

##### Sustainability

The sustainability of the application is more easily guaranteed in the long term by using native languages. This is because any third-party services used in hybrid applications can stop overnight, making them obsolete and unusable in the future.

#### Remark

At M40 this choice is still the best one. Given the long duration of such a project, using third party hybrid development technologies would have forced us to update these technologies at least two times yet which would have resulted in a consequent waste of time and resources.

## 2.2.2 UI/UX

Back in the deliverable D4.8 chapter 3.2.2 UI/UX, we explained that we created a prototype of the Traveler application and developed its design which has been validated by the PTOs. We implemented the designs in the Traveler application, and reworked them after user testing.

In this section, we describe the UX/UI improvements we made. Some of them are based on user feedback, others to improve the accessibility of the application to people with disabilities.

### Ux / UI Design Improvements

#### New icons

These new icons replace the old ones:

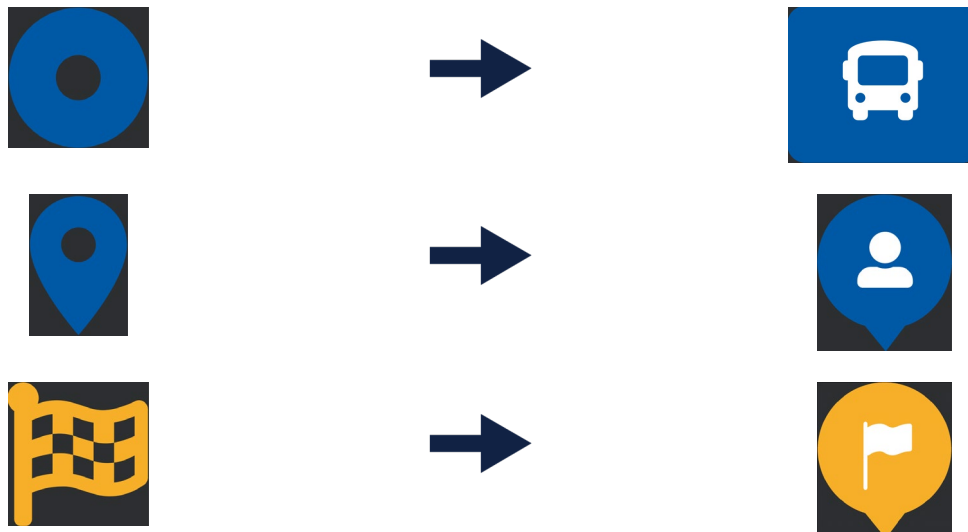


Figure 5 - Old map icons compared to new map icons

- the quote card

The trip recap page has evolved since the last iteration. We have reworked it to be more visual and reused the components set previously which are the map and the selected quote. We have also added a card at the bottom that recapitulates the user's extra options for their trip: the number of passengers and the icon of the options selected.

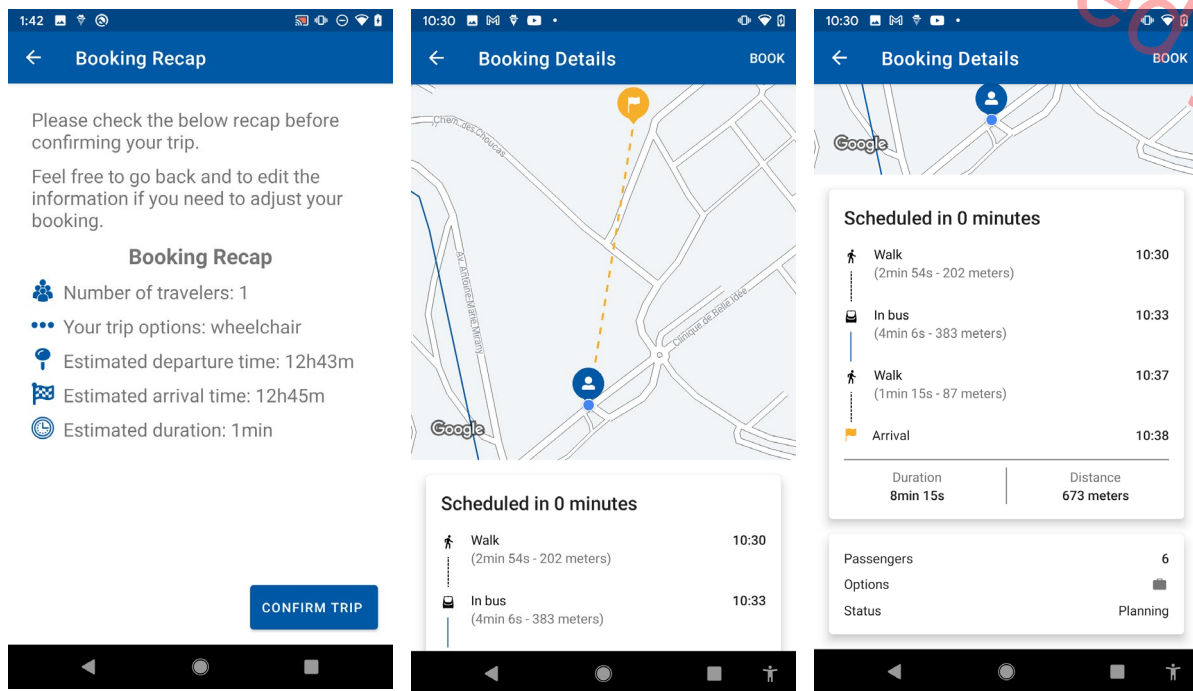


Figure 6 - Trip recap page

### Quote cards

We've changed the information displayed on the card and added more relevant ones.

For instance, the previous quote card was only showing the departure time and the arrival time without more explanation. "Is the departure time the time the bus leaves, or is the time to start walking to the bus stop?" were the users wondering.

To answer this issue, we decided to add explicit and self-explanatory steps with relevant icons:

- Walk - with the start time and an estimation of the distance and the duration
- In Bus - with the departure time and an estimation of the distance and the duration which indicates the estimated arrival at destination time

In the last iteration, selecting a quote card would immediately navigate to the next page. We've changed that, as now the user has to select a card and click on the next button if they want to go forward. Furthermore, if there is only one quote available, the system preselects it.

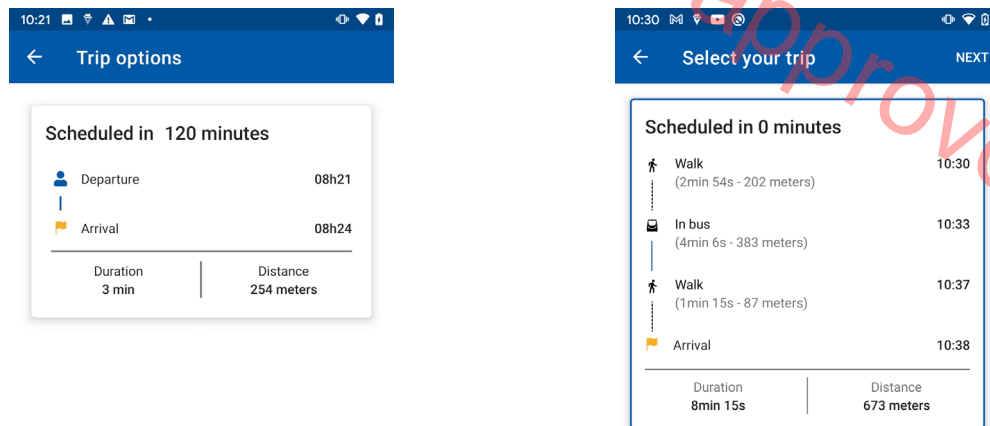
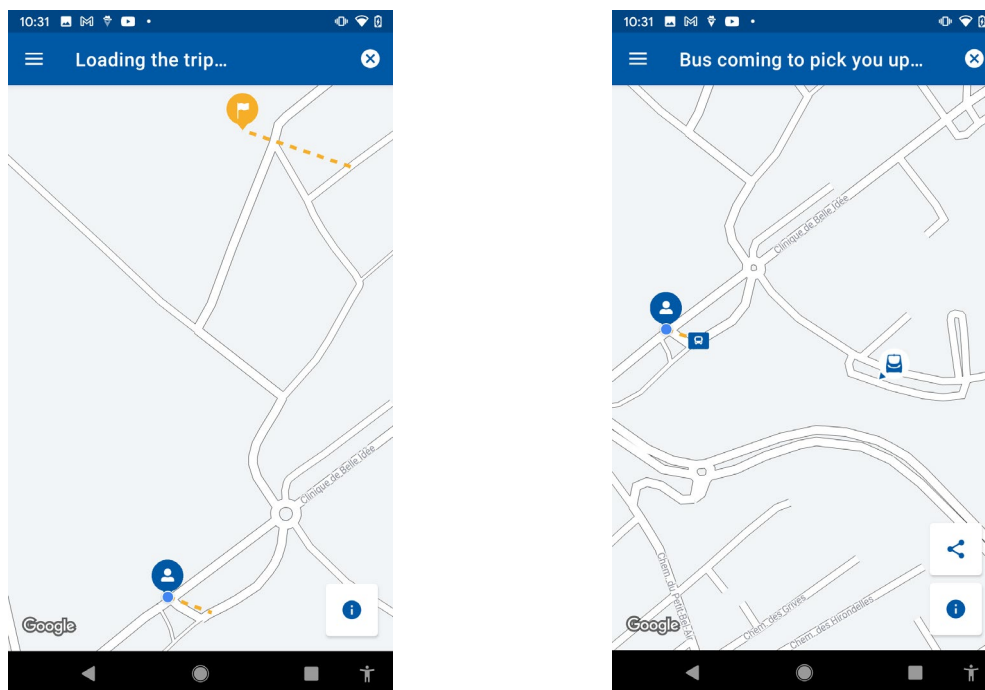


Figure 7 - Quote card update

### On-demand traveling

While on a 'On-demand' trip, to give a better visibility to the user, the map changes its zoom when the status of the travel changes. In addition, the title at the header of the page shows textual information about the booking status. We will explain this process further in the chapter 3.10 On-Demand of this document.



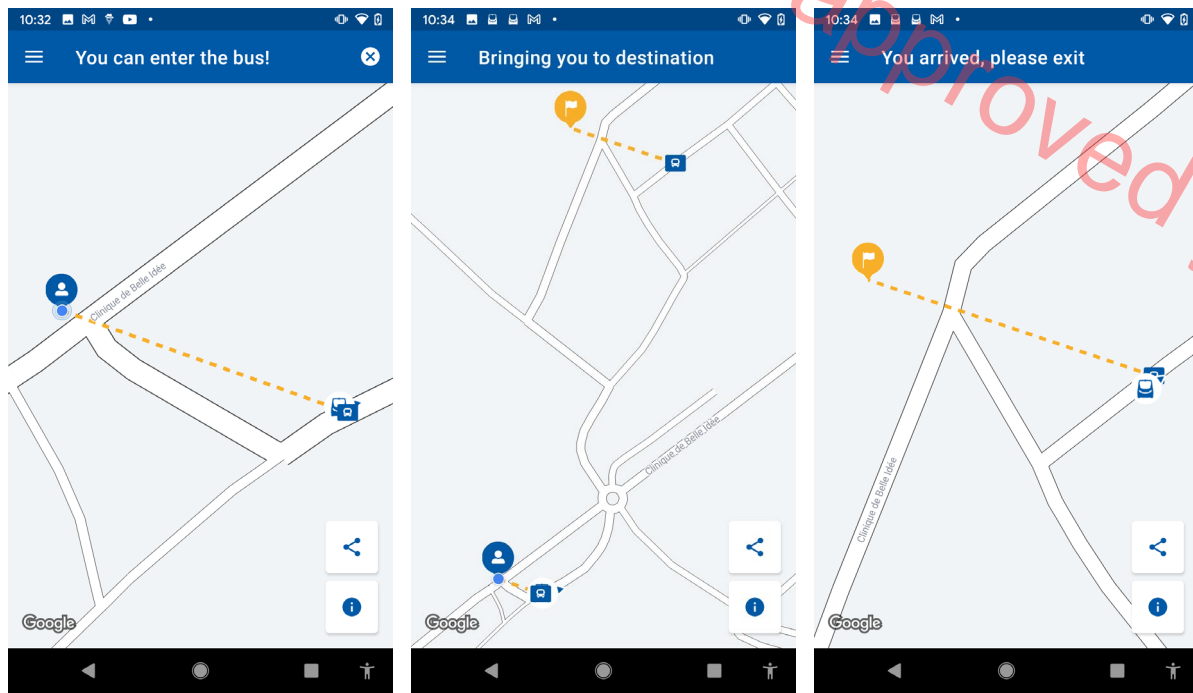


Figure 8 - Screenshots of the zooms depending on the booking status

## 2.2.3 Accessibility

### 2.2.3.1 Analysis

In the last deliverable, we reported on our audit of the app using Google's Accessibility Scanner. The report created by this application allowed us to make changes in the Traveler App' to make the experience as pleasant as possible for people with disabilities.

### 2.2.3.2 Work done

First of all, we have described all the visual contents of the application: logos, icons and other images are described with text so that people with visual disabilities can "listen" to the visuals thanks to the ScreenReader / VoiceOver technology on IOS and the ScreenReader Talkback technology on Android. When the user navigates in the application, he will be able to hear the description of the element he is focused on, and navigate from one element to another by swiping with their finger.

For example, for the option selection page in the quote creation process, the buttons that represent wheelchair, stroller, briefcase, and pet are respectively described as "I am in a wheelchair", "I have a stroller", "I have a luggage", and finally "I have a pet". Thus, when the user scrolls through the items using Text to Speech technology, the system will read the description of the button. The audio result of the following figure would be "I have a stroller selected" and "I have a luggage selected".



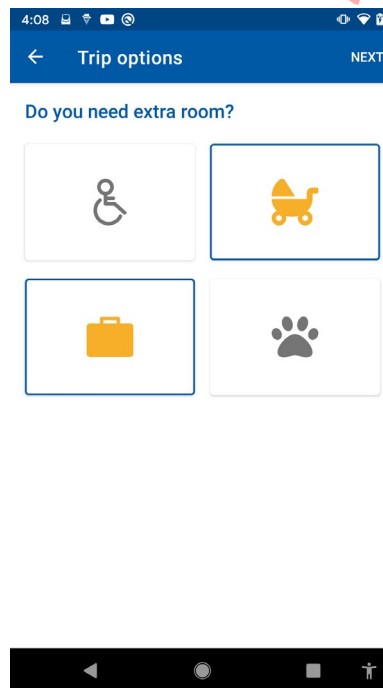


Figure 9 - Trip options with talk back

Secondly, the google report criticized the tone of the white text on its yellow background for not being readable enough. To rectify this, we have created a new style file that can replace the current one. We decided to let the user choose to change the style and switch to accessibility friendly mode from the settings.

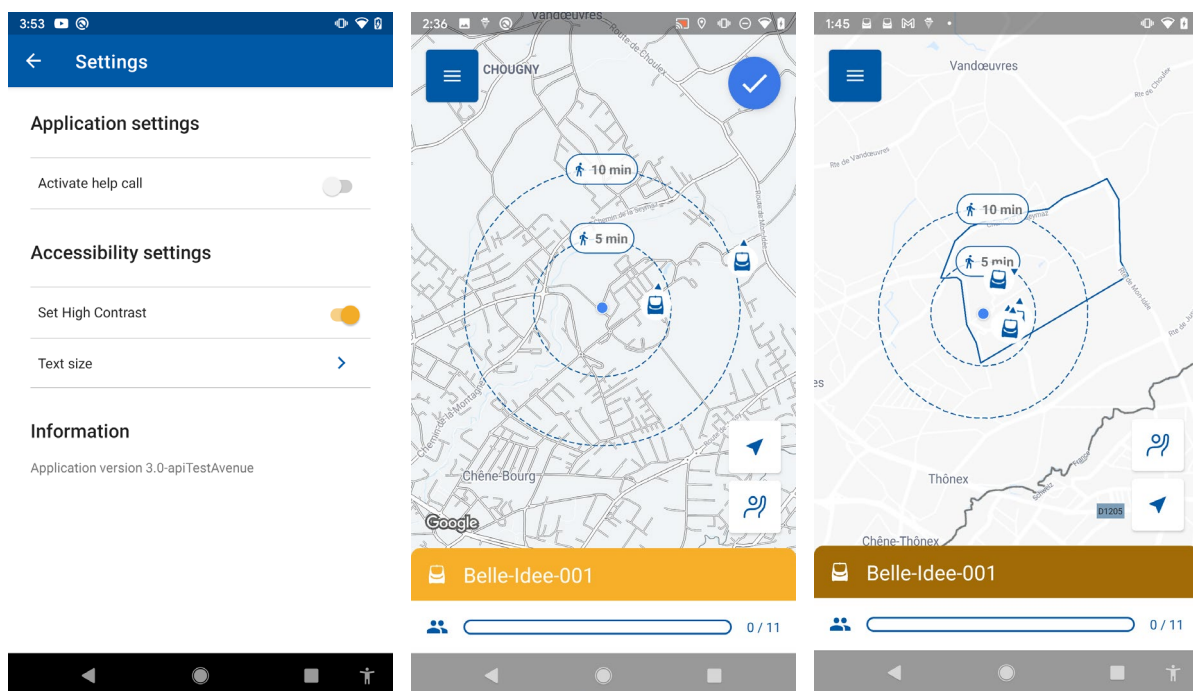


Figure 10 - Basic style vs accessible style

By doing this, we give the PTOs the ability to provide their own "basic" and "accessible" style if needed.

Third, again according to the report, the text size was apparently too small in some situations. Here again, we decided to give the user the choice to change the size. To do this, the user can go directly to the settings of his smartphone to change them. The application will adapt the size of the text according to the one chosen in the settings. We then adapted the size of the elements containing the texts as well as possible to avoid any strange overflow.

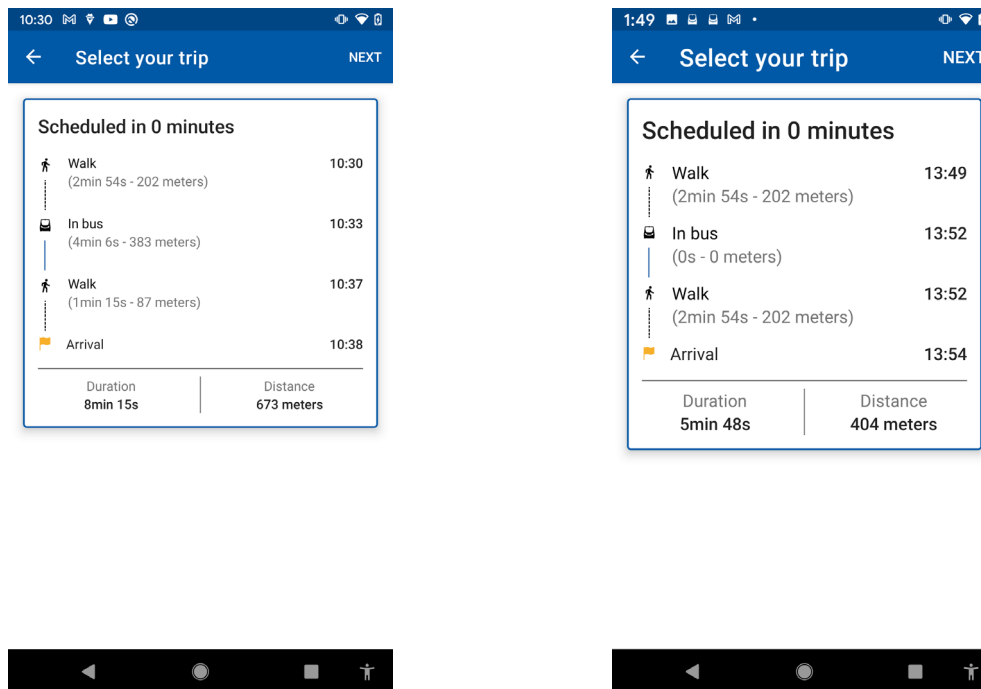


Figure 11 - Before after text resize

### 2.2.3.3 Next Step

The map is not a convenient component for people that have accessibility issues. Therefore, to make it more usable for them, we plan to develop a version of the app without any map. This particular component will be replaced by textual information which is much more accessible thanks to the ScreenReader Talkback technology on Android and ScreenReader / VoiceOver technology on iOS.

## 2.2.4 PTO versions

### 2.2.4.1 Two applications for each PTO

#### General approach

As stated in the Deliverable D4.8 chapter 3.2.2 PTO versions, one of the big specificities of the Traveler application is that it needs to be replicated for all the PTO's of the AVENUE project. On top of the current PTO's, we decided to create these versions in a way that makes it easy to integrate new PTO's if required (See the Technical approach section). To do so, we created a main AVENUE version that is used for tests and validation, and then derived from it all the other versions. The main information that can be changed for each PTO's are static pages, logo, colors and sites (sites being the area where autonomous vehicles are deployed for each PTO's).

Since the last iteration, we added LaPoste of Uvrier to the list of PTOs. Thus, we have deployed two new android and ios applications, branded to their identity. Bringing the total number of Traveler applications deployed for PTOs to 10.

Below, you can find, in that order, a screenshot of the TPG, Keolis, Sales-lentz and the new LaPoste Uvrier (the colors are from Uvrier, the simulator is from Luxembourg) variants:

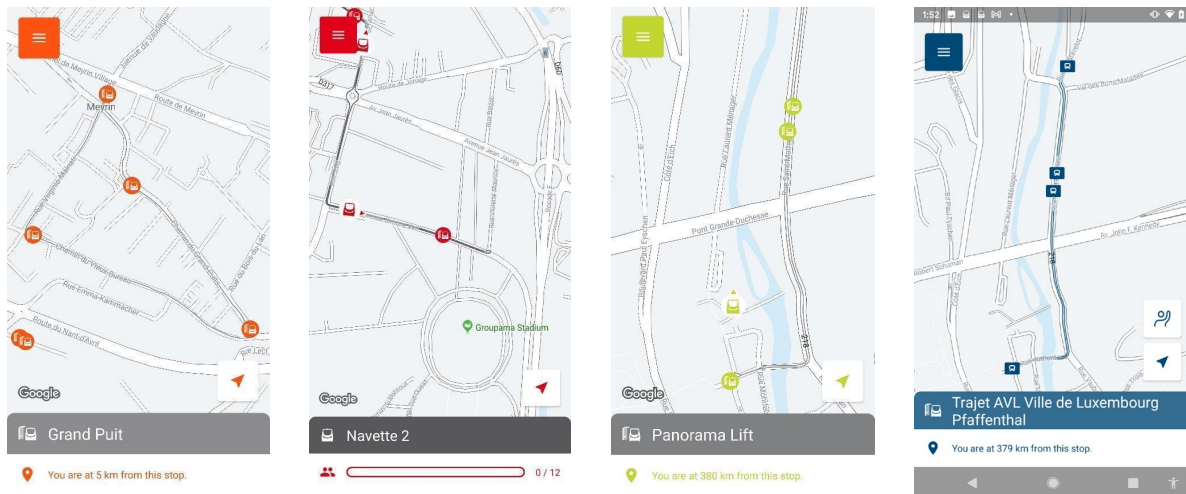


Figure 12 - Traveler App' PTO versions

This has been done for iOS and Android.

### Technical approach

#### **Android**

In Android, it is possible to build several variants of the same app' thanks to the configuration of the gradle.build file. Thus, developers are able to work and maintain one application environment. The gradle.build file allows configuring the build types of the app, the flavours and the combination variants.

The Traveller App' has 3 build types:

- Debug: for development purpose
- Staging: for test and pre-release
- Release: final product that is delivered and deployed

Flavours allow developers to use specific resources like branding resources and API resources depending on the options specified while building the APK.

Branding will include elements such as different logos, primary colours and other individual brand elements, while keeping the layout and user interface constant across all variants.

As we mentioned in the document D4.8, the flavours approach allows the developer, with minimal configuration, to easily add a new branded application to the build. In this case, to add the Uvrier brand we added to the gradle file:

```
brandingLaposte {
    dimension "branding"
    applicationIdSuffix ".laposte"
    versionNameSuffix "-laposte"
    buildConfigField("String", "DEEP_LINK_IOS_BUNDLE", "\"eu.avenue.mobilethinking.traveller.laposte\"")
}
}
```

And the following API configuration:

```
apiLaposte {
    dimension "api"
    versionNameSuffix "-apiLaPoste"
    buildConfigField("String", "DEEP_LINK_URL", "\"https://avenue.mobilethinking.ch\"") // host
    should correspond to deeplink_host manifest placeholder
    buildConfigField("String", "SHORT_LINK_URL", "\"https://avenue2020.page.link\"")
    buildConfigField("String", "API_BASE_URL", "\"https://avenue.mobilethinking.ch\"")
    buildConfigField("String", "API_CLIENT_ID", "\"9\"")
    buildConfigField("String", "API_CLIENT_SECRET", "\"yDAZgSq5RHmc5Jgpq1jacobYo7X7sDdptCpAqUGTJ\"")
    // The map key is owned by developer@mobilethinking.ch in the project Avenue-Traveller
    manifestPlaceholders = [google_maps_key: "AlzaSyDjf2FeX4HBwgmXemMofPCQvqISnUz_Nzw",
        awareness_key: "AlzaSyBsQ-AUVLgjkx_cuwmcwW0emJTUOfPKJ8",
        deeplink_host: "avenue.mobilethinking.ch"]
}
}
```

Then, we added the new logo, colors and texts that are linked to this branding. This added 4 new builds variants to the project:

- brandingLaPosteApiLaPosteDebug
- brandingLaPosteApiLaPosteRelease
- brandingLaPosteApiLocalAvenueDebug
- brandingLaPosteApiTestAvenueDebug

## IOS

In iOS, to achieve the same capabilities as the above Android branding and API configurations, we use targets and schemas as described in D4.8 - Chapter 2.2.4. The creation of the Postauto version has been as fluid as for Android.

## 2.2.5 Translations

### 2.2.5.1 Work done

We developed the application so that it detects the language of the user's smartphone, and loads the corresponding language file. Thus, all the texts of the application have been translated into English and French and can easily be translated into all other languages thanks to json files. We have created one file per language that will be loaded by the system according to the language of the user's smartphone.

Each file is composed of a dictionary with a key and a value. The key is used in the code while the value is loaded dynamically when the application starts.

## 3 Out-of-vehicle services

In this section, we present the out-of-vehicle services developed so far as well as the services selected for phase 3 (M26-M40) in detail.

### 3.1 Passenger presence

No major changes have been made for this service. Hereafter is a general description of the service that is partially taken from D4.8 chapter 3.1 and enhanced with new developments.

#### 3.1.1 Initial vision

Userstory: As a user I want to know the amount of people in the shuttle that I will take.

Use case: From an **end-user perspective**, knowing the remaining available space inside the vehicle is a key piece of data to decide whether to wait for the next vehicle, book another vehicle or whether there is enough space for them and their party/personal belongings.

Comment: In order to allow for fully on-demand autonomous operations, the means for knowing at any moment how many people are inside the shuttle and, more importantly, the remaining capacity inside the vehicle, needed to be put in place.

Service: in-and-out-of-vehicle service

#### 3.1.2 Work done

Feature description:

The Driver App previously developed has not been in use for the second part of the project. As the operators already had to use an Operator App provided by bestmile, it was not possible to ask each operator to use two separate applications. On top of that, CERTH and NAVYA are working on a camera based passenger counting service.

This passenger counting service described in D4.6 will be used in the future to provide data to the Traveler App instead of the previously used Driver App (described in D4.8).

No changes have been made regarding the Passenger Presence service since D4.8.

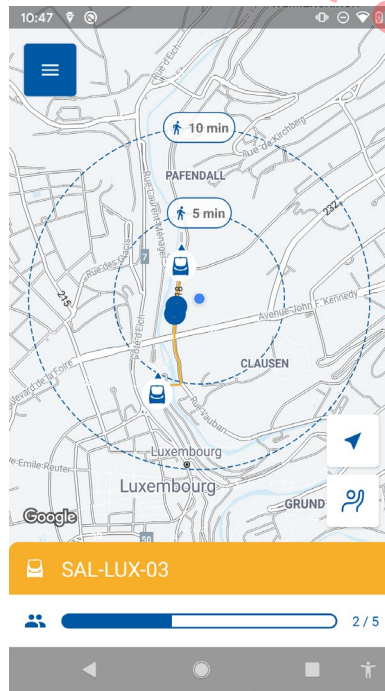


Figure 13 - Passenger presence - Traveler App'

#### Technical description:

No updates have been made regarding that service. For detailed information about it, please refer to D4.8 chapter 3.1.

### 3.1.3 Next steps

The next step for this service is still the automation of the passenger counting. This is currently studied jointly between MobileThinking, CERTH and Navya. The in-vehicle service has been developed by CERTH and is currently tested by Holo.

Once this service is approved, we will very easily be able to plug it to our infrastructure as follows.

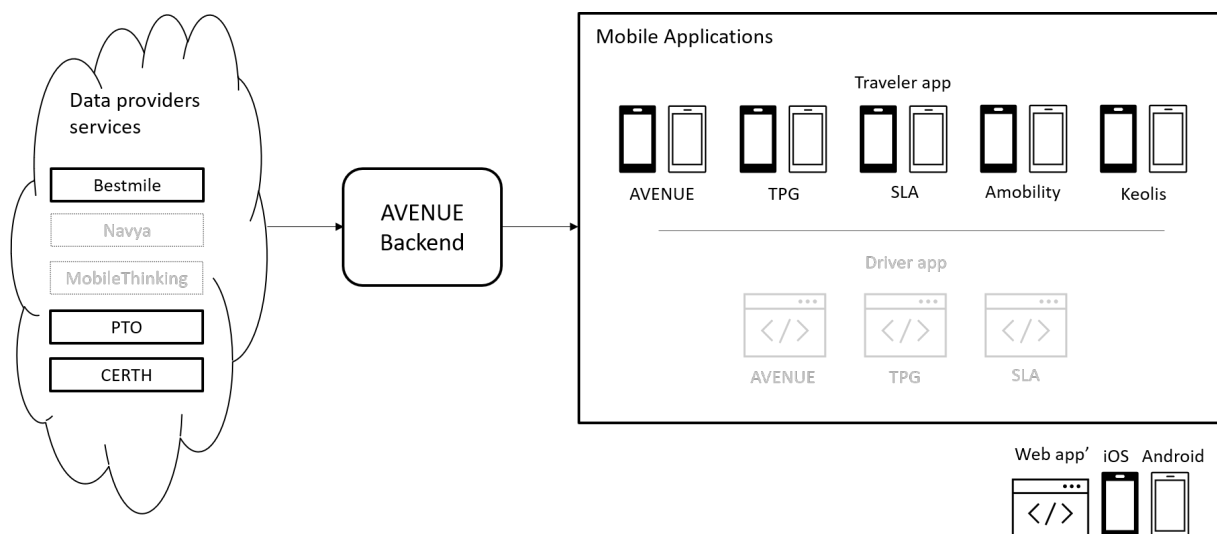


Figure 14 - Passenger presence next step - AVENUE services software ecosystem

## 3.2 On-Demand Real-time visualization

Hereafter is a general description of the service that is partially taken from D4.8 chapter 3.2 and enhanced with new developments: the real-time visualization has been extended to the on-demand service.

### 3.2.1 Initial vision

Userstory: As a user, I want to know my real-time itinerary and where my On-demand bus is located.

Use case: The main goal of this service is to display the real-time position of an autonomous shuttle at all times on a map. Knowing both before and during the ride the real-time position of the shuttle increases the usefulness of the system as a whole, and as a consequence, user adoption. Feeling in control is one of the key psychological factors to increase trust in autonomous public transport. In particular, we have to explore how to convey trust to users since the main objective is that at some point the vehicle will not follow a predefined path anymore, but it is re-routed depending on the destinations of the passengers. Passengers may feel lost if they can figure out why the vehicle is not performing the usual path. Furthermore, in order to not disturb the user, once they book a trip, the on-demand map shows only the bus that is related to the trip.

Service: in-and-out-of-vehicle service

### 3.2.2 Work done

Analysis:

Having implemented in the Traveler App the functionality related to the On-Demand service, it was necessary to add the real time visualization of the available buses or of the user's trip.

The mechanics are similar to what has already been done for the real time visualization of the fixed routes but we have added some specificities for what is related to the 'On-Demand' service.

Feature description:

This service is implemented in the Traveler apps, which are native Android and iOS applications. It also requires the Application Backend we developed.

There are two different setups for this service. The Site can either be set to Fixed-line or On-demand. In both settings, the screens are mostly composed of a Map allowing the user to visualize the available shuttles with very few actions possible.



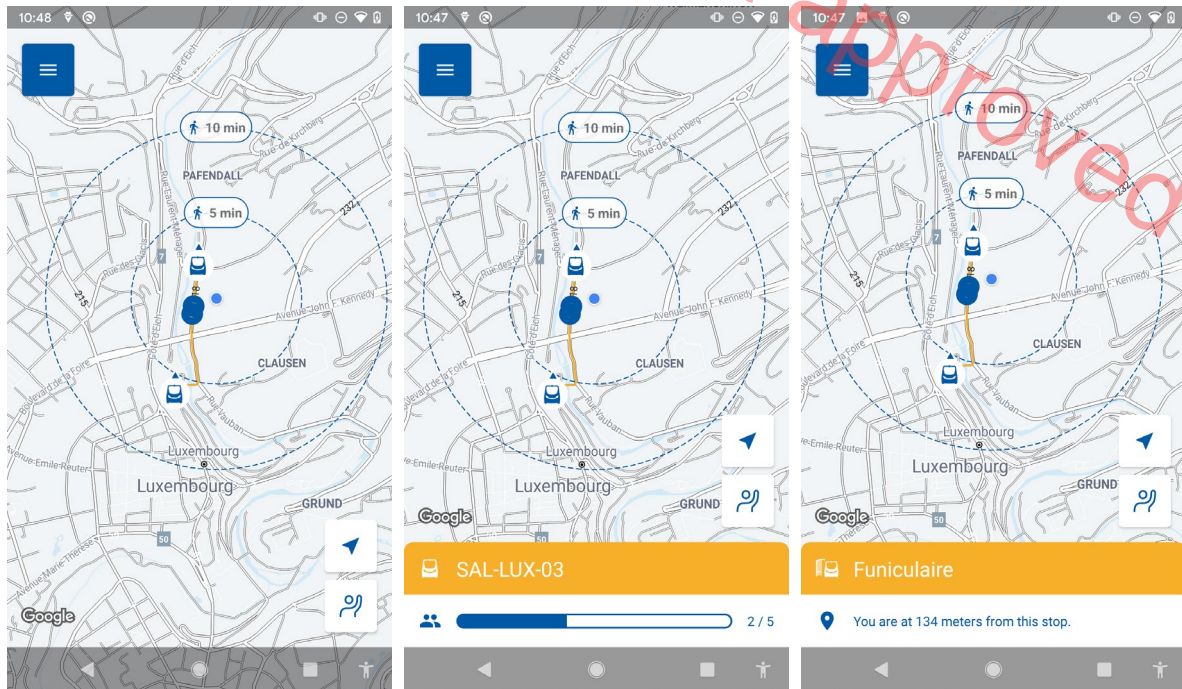


Figure 15 - Real-time visualization - Traveler App Fixed Mode'

Since the last deliverable we updated the real time visualization of the On-Demand. First, in each On-demand zone, the app displays on the map the available buses, their real time position and their occupancy.

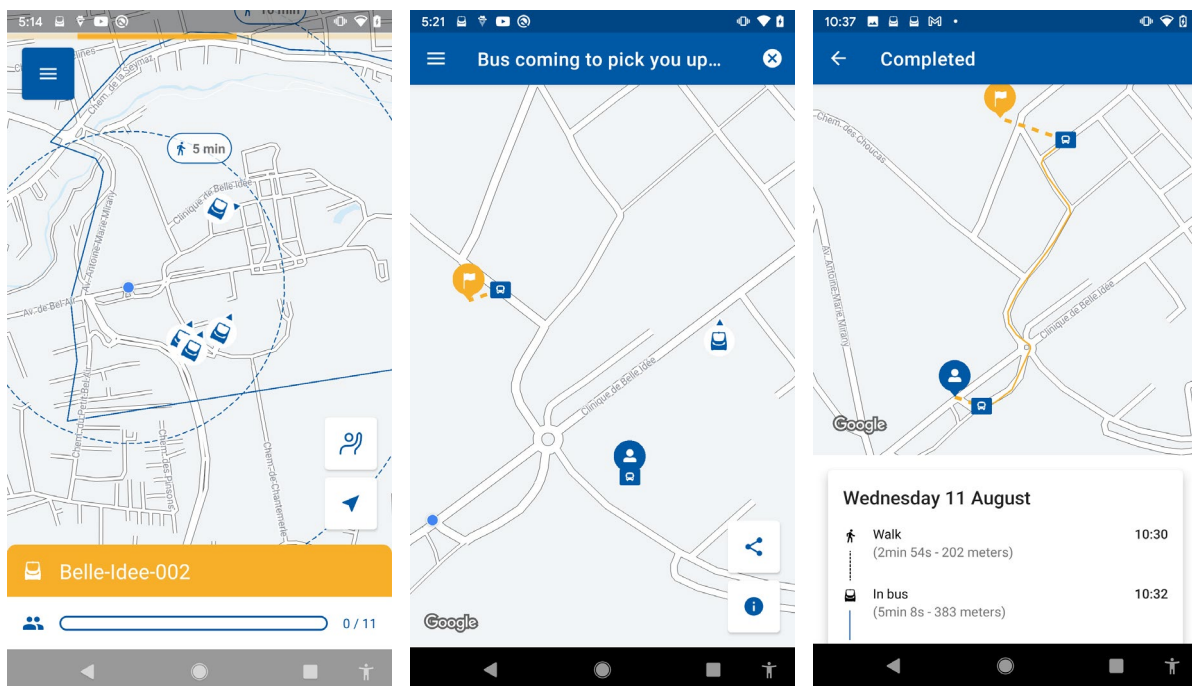


Figure 16 - Real-time visualization -Traveler On Demand Mode

As a reminder, in the On-demand mode, there is no fixed line and no stop. But we choose to show to the user the bus and the stops once they are traveling. They can only see their bus on the map with its real time position and trip status. At the end of their trip, the user also see the path of the shuttle as well as the departure and arrival positions (cf. section On-demand booking service).

#### Technical description:



## Application Backend

In the same way as we explained in deliverable D4.8, chapter 4.2 Real-Time visualization, we retrieve the data from a data provider (such as Bestmile) and after transformation and storage we return the 3 entities, stop, lines and vehicles to the Traveler application.

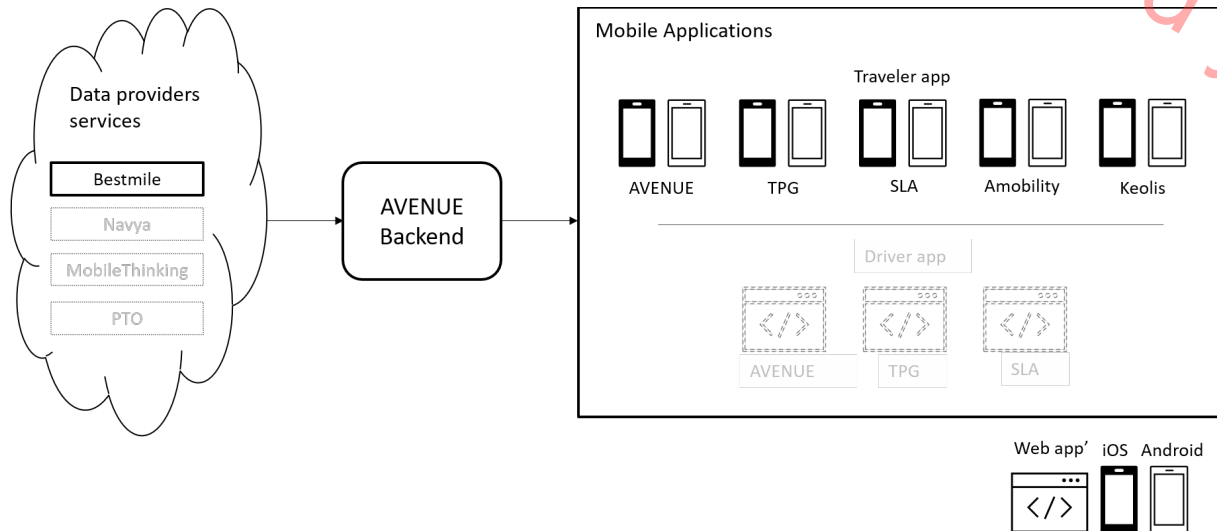


Figure 17 - Real-time visualization -AVENUE services software ecosystem

Since in an On-demand area there are no stops nor lines, the data provider only returns the vehicles present in the area when the user is in the main map of the application.

## List of all available vehicles

The Application Backend gathers first from bestmile all the available services, and then loops over them to fetch the available vehicles.

1. <https://api.bestmile.com/documentation#get-/transportation/v1/services>

The first call is required to get the services of the pto and query the vehicles of the desired service.

2. <https://api.bestmile.com/documentation#get-/transportation/v2/services/-ServiceID-/vehicles>

The entity returned is an array of vehicles described as the following JSON object:

- id: string
- name: string
- service\_id: string
- lat: number
- lon: number
- bearing: number
- occupancy: number
- capacity: number

When a user starts an 'On-Demand' session, the backend requests the vehicle, the stops and the path related to the journey through the following api call: <https://api.bestmile.com/documentation#post/booking/v5/booking> which returns the booking object and its related information (cf. chapter 3.10 On-demand of this document).

## Frontend

As for the fixed line real time visualization, while on a 'On-Demand' trip mode the Traveler App makes the API call to fetch the booking every 3 seconds. The real time position and bearing of the AV are updated and with an animation method that, thanks to vectorial calculus, makes the buses "slide" from point A to B. The status of the travel is also updated and displayed to the user in the header of the app.

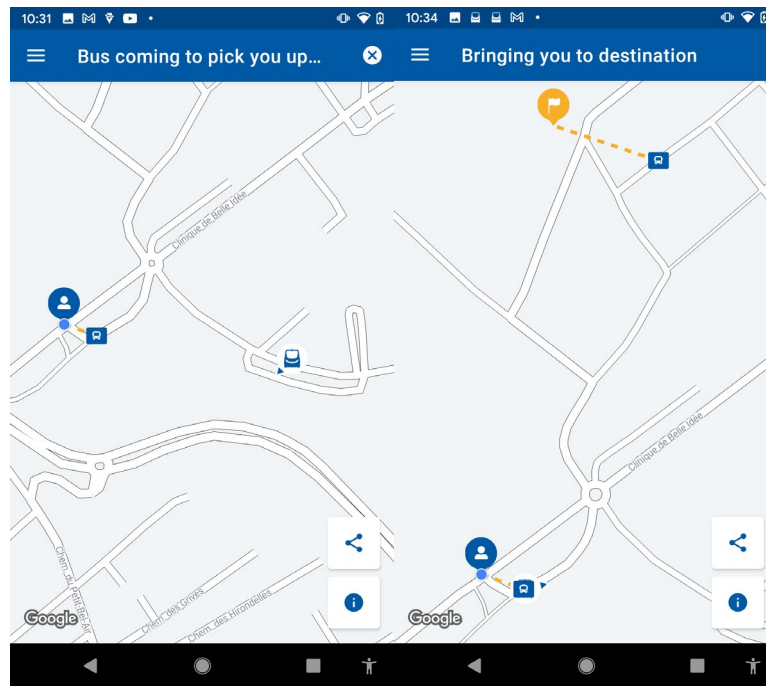


Figure 18 - Real-time visualization - AVENUE services software ecosystem

### 3.2.3 Next steps

This service has been fully implemented and extended to meet the needs of the users in the 'On-Demand' process. We can consider it finished. However, in the future, we plan to add and display all the information that will be available through Bestmile (or other) API, such as Battery level, Status of the Doors, and every information that might help the user feel at ease.

## 3.3 On-demand stop

No major changes have been made for this service. Hereafter is a general description of the service that is partially taken from D4.8 chapter 3.3 and enhanced with new developments.

### 3.3.1 Initial vision

Userstory: As a user I want to request the AV to stop so I can jump in or get out.

Use case: From an **end-user point of view**, which is what concerns the in- and out-of-vehicle services, this service enables them to easily interact with the autonomous vehicle from the outside (*but also from the inside*) and to instruct it to stop when the situation requires it (i.e. when waiting at a stop that is deserved by the AV only on-demand)

Comment: Before AVENUE is ready for fully on-demand operations without the notion of fixed routes and stops, autonomous shuttles in each of the partner sites follow predefined routes with predefined stops.

Service: in-and-out-of-vehicle service

### 3.3.2 Work done

On the application side, everything is ready and could be used should a PTO want to use this service. However, as explained in D4.8 chapter 3.3 this service was mostly a first step towards On-Demand.

## 3.4 Automatic trip planning suggestion

No major changes have been made for the automatic planning suggestion service. Hereafter is a general description of the service and the work previously done that is taken from D4.8 chapter 3.4.

### 3.4.1 Initial vision

Userstory: As a user, I want to be able to receive automatic trip planning suggestions

Use case: The goal here is to be able to use the PTO's AV as any other PTO means of transportation. This will be possible when the Autonomous Vehicle will be fully integrated in the service offer.

Service: out-of-vehicle service

### 3.4.2 Work done

Analysis: The complexity of such a task is that it must connect to third party API's which are not always open. This service will be developed only if AVENUE PTO's are willing to integrate it in their offer. Their position is vital for this service as we will need to access very sensitive data (scheduling, live). Therefore we can only develop it with the full collaboration of PTO's. At the moment, none of the PTO's have requested such integration as we haven't tested the on-demand in real situations yet.

However, the analysis done so far will be very useful to establish recommendations. For example, in order to have a fully automatic trip planning it is necessary to have an open and secured access to the Information System of the PTO that wishes to provide such service. This can seem obvious but the current state of PTO's Information System is, for most of them, not ready.

Development: None

### 3.4.3 Next steps

Possible implementation in Year 4 if at all.

## 3.5 Trip planned via call centers

No major changes have been made for this service. Hereafter is a general description of the service that is taken from D4.8 chapter 3.5 the work previously done.

### 3.5.1 Initial vision

Userstory: As a user, I want to be able to plan a trip even without using IT devices

Use case: One limitation of modern technology is accessibility for people with limited access to technology. To face this problem PTOs could have a call-centre dedicated to help non-autonomous and not tech-savvy people plan/book their trip. This call-centre could be human at first and automated when ready.

Service: out-of-vehicle service

### 3.5.2 Work done

Analysis: Even if this service can seem fairly easy to implement, it is very important for PTO's to provide it to their customers. Depending on the kind of service they plan to offer, it is almost mandatory. For example, with a fully On-demand booking site, people will need to use the mobile application to book a ride. However, not everybody is able, or want, to use the mobile application. For such people, a simple phone call will allow them to use the service.

Development: Thanks to Bestmile's Booking Management, it is already possible for the operator to make bookings directly from the Bestmile Dashboard when they receive a booking request via a call. Bestmile is currently working on improving all aspects of Booking Management for bookings from the Dashboard: creation, real-time tracking, cancelation, rematch, history reporting, etc.

### 3.5.3 Next steps

The implementation of the service depends on the will of PTO's. It is ready and can be implemented at any given time. PTOs simply need to setup a dedicated line, hire operators, and advertise the phone number for people to call.

## 3.6 Digital or human information points

No major changes have been made for this service. Hereafter is a general description of the service that is taken from D4.8 chapter 3.6 the work already done.

### 3.6.1 Initial vision

Userstory: As a user, I want to be able to get information about the Autonomous Vehicle service.

Use case: The AVENUE project and more specifically PTOs should provide information points to help users with any administrative tasks (booking, creating a membership, planning a trip etc.). It could be done via telephone, via the operator's control center or via digital means (website and app' tutorials).

Service: out-of-vehicle service

### 3.6.2 Work done

Analysis: This service requires infrastructure installed (for the digital part). We can develop different things like: a QR code at bus stops that sends to a web page will inform us about the bus stop etc, an installation of information borne (actually upgrade the existing ones installed by the operators). For human interaction, this is already there by the PTO's (information centers). For AV services the intervention team can also act as an information point (not fixed). They can be reached directly within the app, or by phone call.

Development: In terms of technological development, an entire tutorial section will be created on the avenue website. This section should describe how to use the system as well as the mobile application. This tutorial will be accessible via a shortcut to the website in the mobile application.

### 3.6.3 Next steps

This tutorial section should be developed in year 4. The objective is to have it ready whenever the On-demand booking service is available to the public.

## 3.7 On-demand zone

### 3.7.1 Initial vision

Userstory: As a user, I want to know when I enter an on-demand zone, where I can order autonomous vehicles.

Use case: In a futuristic vision, there will no longer be any fixed bus stops. Therefore it is mandatory to signal users "zones" where public transport is available.

Service: out-of-vehicle service

### 3.7.2 Work done

Analysis: When an autonomous vehicle is driving On-demand models, it is difficult to detect where the buses are actually offered. The idea is therefore to build a service that makes the user aware that he/she is now in an area where the On-demand shuttles can be ordered. This has been done via the Traveler app'. Whenever a user enters an AV zone, they can order a bus to travel in that zone. These zones are indicated on the map of the app.

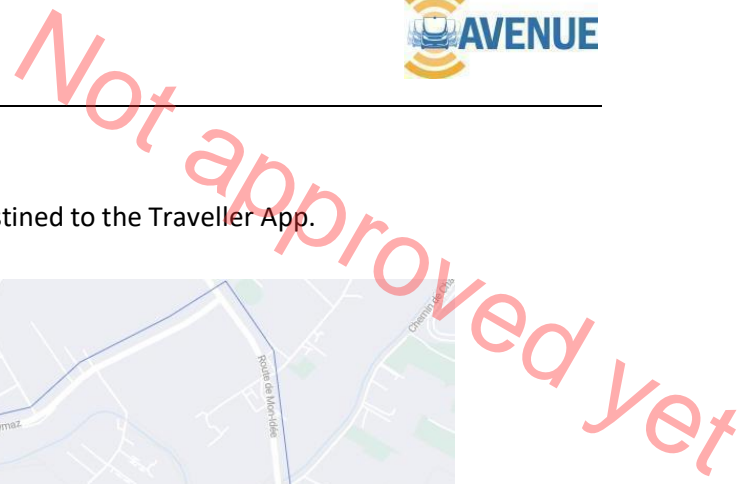
Development: The geofencing feature was already developed in the previous iterations. It has been adapted to alert the user once they are in an AV zone.

#### Backend

The API to get the zone's polygon was already developed in the backend server. This call to Bestmile's API allows it to get the On-demand zone as seen in the following image of Bestmile's dashboard. The Application Backend calls the following endpoint from bestmile to get all the services.

1. <https://api.bestmile.com/documentation#get-/transportation/v1/services>

Continued to the Traveller App.



Continued to the Traveller App.

Continued to the Traveller App.

Continued to the Traveller App.

Continued to the Traveller App.

Continued to the Traveller App.

Continued to the Traveller App.

- Continued to the Traveller App.

Continued to the Traveller App.

Continued to the Traveller App.

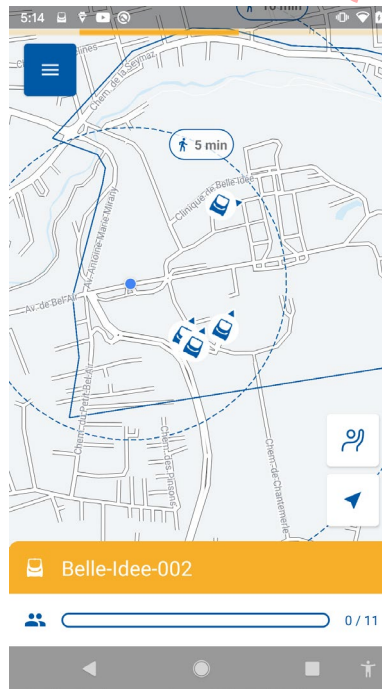


Figure 20 - On-demand zone - Traveller App' Map

### 3.7.3 Next steps

Thanks to Bestmile's API, we were able to fully implement this out-of-vehicle service by giving the user the ability to know when they enter an 'On-demand' zone to be able to book a trip. And to see on the map of the app to the zones available.

## 3.8 Single button vehicle calls and help request

No major changes have been made for this service. Hereafter is a general description of the service that is taken from D4.8 chapter 3.8. The only modification that occurred since the last deliverable was the design of the setting page.

### 3.8.1 Initial vision

Userstory: As a user, I want to be able to call for help in an extremely easy and fast manner.

Use case: The goal here is to provide simple ways to people to call for help. Either through an application or via a dedicated physical button in the AV.

Service: out-of-vehicle service

### 3.8.2 Work done

Analysis: Providing a phone number might not be enough for everybody. Specially for non tech-savvy people it might not be a reflex to go dig in the application settings. We decided to add in the "accessibility" settings a way to display permanently on top of the map a shortcut to contact a PTO



responsible person as a first stop. From there we will be able to build on top of this service depending on the need of travelers as well as wishes from PTO's.

**Feature description:** In the first version of the mobile application the user can activate the "Help call" button. Doing so will display a red call button on the top right corner of the application. Clicking on it will call the closest PTO operator. In a later stage and if needed by the PTOs, we will additionally send a message to the Bestmile operator dashboard with all the contextual information we can gather (position, shuttle, etc.).

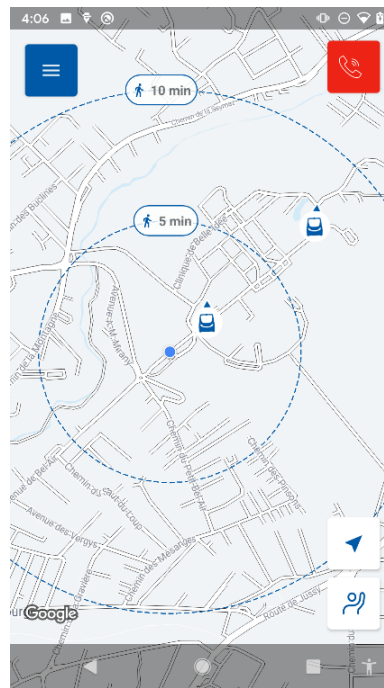


Figure 21 - Single button vehicle calls and help request - Traveler App' button display

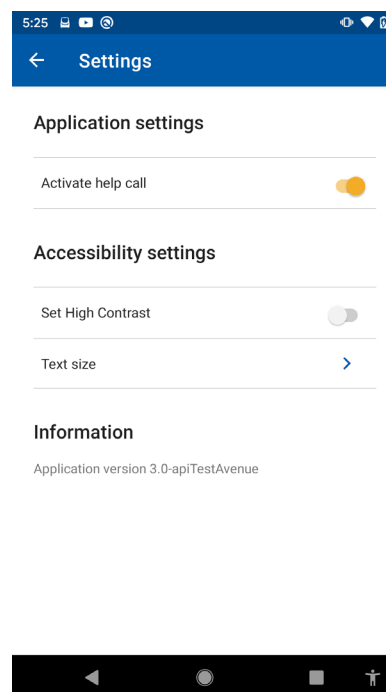


Figure 22 - Single button vehicle calls and help request - Traveler App' settings screen



### Technical description:

#### Application Backen

The role of the AVENUE Application Backend in this service is to provide the PTO phone number so that the Traveler application can instruct the operating system (OS - Android or iOS) to initiate the phone call. As we explained in section Core packages each PTO is a customer and each of them has a customer profile. Each PTO can register an application and therefore when the application asks the PTO customer profile the Application Backend answers with the correct data. For example, if the Android application of TPG requests the TPG profile the Application Backend knows that the owner of that application is TPG and therefore returns the TPG profile data. The profile data of each PTO contains the contact information of the PTO, some addresses, some customer care numbers, and an emergency contact. The specific PTO application uses the latter to initiate the emergency call via the OS.

In the future, the Application Backend via the standard AVENUE API will also expose one or multiple endpoints to automatically send to the PTO operator extra data when the user initiates the emergency call. For example, the id of the vehicle in which the user in need of help is currently traveling.

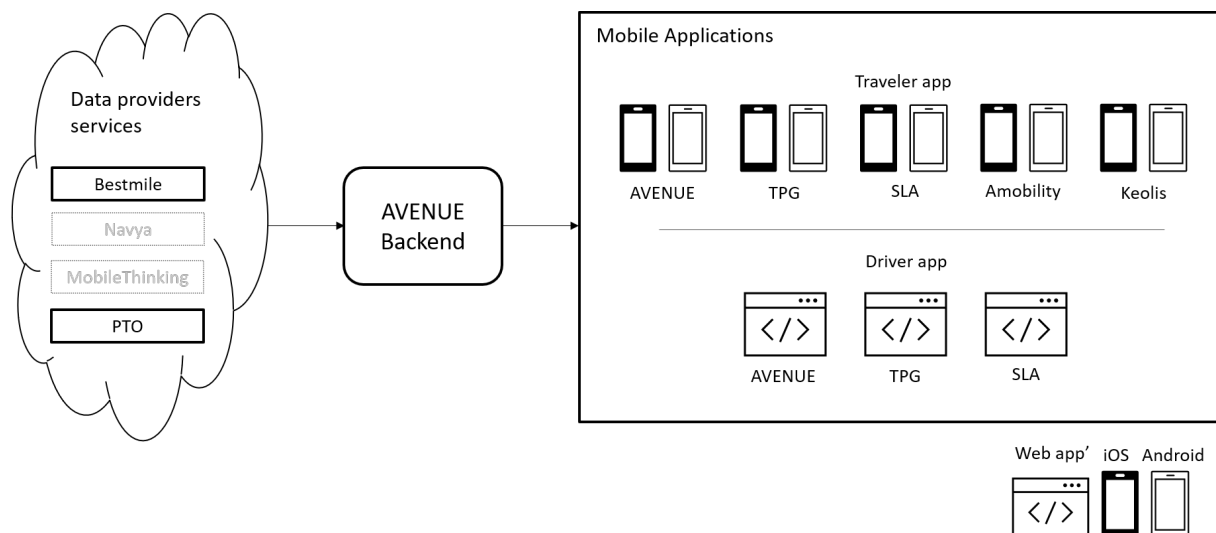


Figure 23 - Single button vehicle calls and help request - AVENUE services software ecosystem

#### Frontend

When the user activates in the setting the “help call” toggle it set the value of a general boolean to true. The visibility of the call button is linked to that boolean: true it appears, false it disappears.

As written in the feature description, clicking on that button allows the user to contact the PTO. The Traveler app’ opens the native phone call application of the user’s phone via Android and iOS internal intents. The intent sent to the phone application contains the number of the PTO that the Traveler application retrieved from the AVENUE server at start up time. The user is automatically ready for the call. They will only have to press their “call” button to contact the service.

### 3.8.3 Next steps

A first version is already usable in the Traveler App'. This service, if required by the PTO, could be improved in year 4 by adding lat/long position through text messages that would be sent to the operator. It would be possible to add a video conference system in the Traveler app' if needed. For now, this is not a priority for the PTOs.

## 3.9 Online ticketing services / Convoy service

No major changes have been made for this service. Hereafter is a general description of the service that is taken from D4.8 chapter 3.9.

### 3.9.1 Initial vision

Userstory: As a user, I would like to book trip for me / As a user, I would like to book a group trip involving several autonomous vehicles

Use case: The possibility to book online (e.g., mobile application) or through a call centre or at the public transport kiosks single person trip or several autonomous vehicles, so that a group of people can move in the city as a convoy of smaller vehicles.

Service: out-of-vehicle service

### 3.9.2 Work done

Analysis: The online ticketing should and will be implemented directly by the PTOs in their version of the mobile application. For example, at TPG, they are using SMS based ticketing. The Convoy service is already possible through Bestmile orchestration. It will not guarantee that the shuttles are following each other, but will ensure that enough AVs will come to pick up and drop off the requested number of people.

Development: None planned

### 3.9.3 Next steps

None planned

## 3.10 On-demand booking

Major changes have been made for the On-demand booking service. In order to remind the context of this service, we have repeated some parts of what was written in chapter 3.10 On-demand booking of document D4.8.

### 3.10.1 Initial vision

Userstory: As a user I want to be able to order an AV to come and pick me off at my doorstep and drop me off where I desire.

Use case: Specially in the "lastmile" type of services, there is a strong desire to cover a larger zone in order to help every citizen use the public transport. The objective here is to be able to cover areas that

are not densely populated. And fully On-demand booking service would solve this issue as the AV would cover an entire area instead of a fixed line.

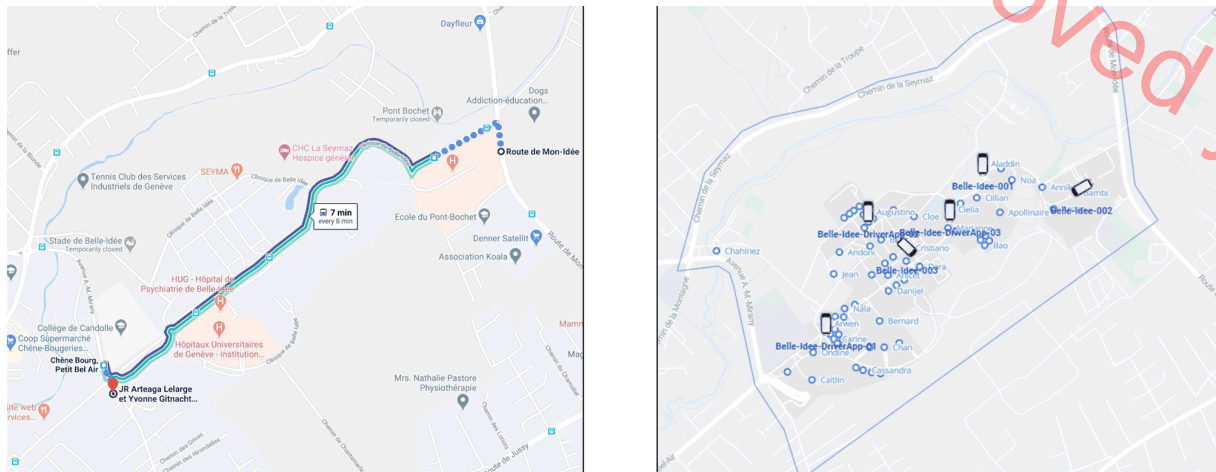


Figure 24 - Fixed-line vs On-demand booking

Service: out-of-vehicle service

### 3.10.2 Work done

Analysis: This service is probably the most important service of the WP4. The entire success of the project rests on the fact that the AV can function On-demand. When we started implementing this feature, Active mission was not accessible. This means there was no way to give orders to the AV remotely in order to pick up passengers. As opening an Active mission outside of Navya's infrastructure is an extremely sensitive task, we decided to anticipate it's availability.

To do so, we worked very closely with Bestmile and their simulator. This work allowed us to fully build the service on the mobile application side as well as improving the simulator itself.

Feature description: When the user clicks on the "on-demand" button, the Traveler app starts the quote selection process. This process is composed of a succession of pages where the user has to provide their journey's information.

The first step is to provide the destination and the departure position: the app provides two ways to accomplish this:

1. via a draggable pin icon on a map
2. by selecting a Point Of Interest. (cf. section 3.14 Point of interest of this document).

For the map with a draggable pin icon, when the user drags the icon to the desired location, its longitude and latitude are saved. On the other hand, if a Point of Interest is selected, the pin on the map goes to the selected location. If the user selects a POI then changes the pin location on the map, the POI selection is cleared.

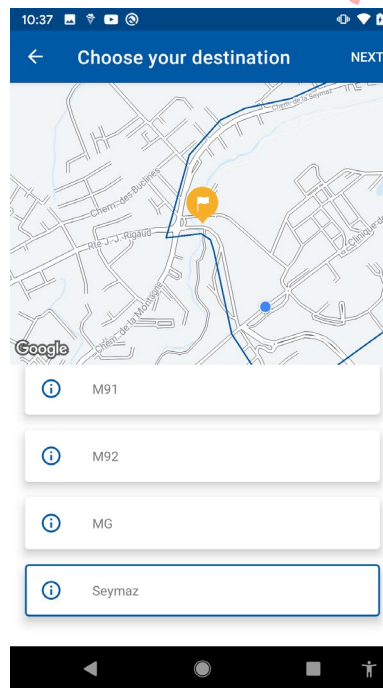


Figure 25 - Choose a destination - On-demand booking

Once the user confirms the desired destination and departure, the Traveler app' asks them about their trip options. First, the user selects the number of travelers by clicking on the corresponding button. The app' shows a list of buttons from 1 to 11 (11 being for now the maximum decided capacity). When clicked, the button color changes to let the user know about their choice.

After that, the app' asks the user to select their special requirements, if any, by clicking on the corresponding button. Each requirement is represented by a specific icon: a dog, if travelling with an animal, a stroller, a suitcase and a wheelchair. These elements are an initial list that can be improved through time. If the user has none, they can continue with the process by simply clicking the next button.

From there, the user receives "quotes". They are displayed to the user with information about departure time and estimated duration. The user chooses the quote that suits them the best.

They are then redirected to a map visualization page where the bus coming to pick them is displayed, along with the destination and departure stops and the positions filled in previously.

At the very end of their trip, the user can see a summary of it's trip and can rate it using a simple Smiley method (cf. section User feedback service).

A video demonstrating the On-demand booking service using Bestmiles' API is still available following this link: <https://youtu.be/X7XO27kr1KM>.

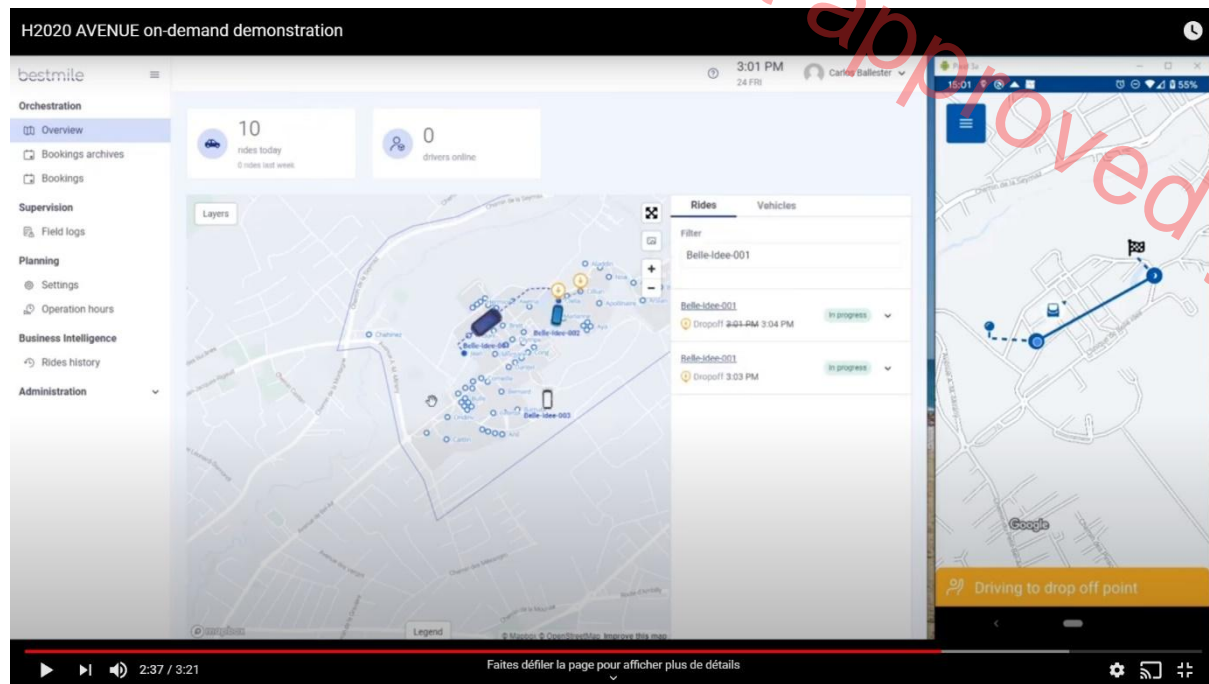


Figure 26 On-demand booking demonstration video

#### Technical description:

##### Application Backend

The AVENUE Application Backend does the heavy lifting for this service. It exposes three endpoints via the standard AVENUE API that the client applications can exploit to perform a full on-demand session.

The first endpoint allows the client to get the quotes, so that the user can select the trips it prefers. For this endpoint the Application Backend expects to know the departure and destination point of the user, how many people are traveling and the accessibility options (e.g., wheelchair and stroller). The Application Backend communicates all this data to the service provider of the on-demand service for the given PTO, such as Bestmile, and returns a list of quotes to the Traveler application.

With the new Api, the endpoint used to construct and get the quotes is the following:

1. <https://api.bestmile.com/documentation#get-/booking/v5/quotes>

The backends receives the following body object from the app:

- originLat: a number that corresponds to the latitude of the origin position
- originLon: a number that corresponds to the longitude of the origin position
- destinationLat: a number that corresponds to the latitude of the destination position
- destinationLon: a number that corresponds to the longitude of the destination position
- numberOfTravelers: the number of travelers

To note that the accessibility options are not linked yet and not taken into account by the data provider.

The backend sends a list of quotes to the app with the following properties:

- id: which is the unique identifier of the quote
- distance: the distance of the journey
- duration: the duration of the journey
- start\_time: the departure time of the journey
- finish\_time: the end time of the journey
- service\_id: the identifier of the service the quote is attached to
- journeys: Which is an array of the steps of the journey described as follow:
  - type: JourneyType: *Transfer* or *Ride*, typically “In bus” or “walking”
  - start\_time: the start time of the step
  - finish\_time: the end time of the step
  - distance: the distance of the ride / walk
  - duration: the duration of the ride / walk

Once the user chooses the quote that suits them the best and sends their choice to the server, the data provider should be able to send back a booking object.

The second endpoint is about creating the booking. Once the user selects the quote they prefer, the AVENUE Application Backend uses the quote to create a booking. When the booking exists the on-demand session starts. The procedure looks simple, but the AVENUE Application Backend has to perform several API calls to the on-demand service provider to negotiate and put together the final booking data that the Application Backend transmits to the client application. The booking contains its status, the origin and destination locations, the journey steps, and the current ride information if the user is in the vehicle while performing their journey (e.g., the position of the vehicle). We carefully designed the structure of the booking data to very easily display everything we want to the users via the client application.

The following request to bestmile’s Api with its payload makes the data providers create the booking:

<https://api.bestmile.com/documentation#post/booking/v5/bookings>

with the following data payload

- quoteID: string
- travelerID: string

In addition to its general data, a booking is composed of an object ride and a list of journeys. A journey can be of type ride or transfer.

A transfer is when the user travels on foot, while the ride is when the user is inside AV.

The following data represent the booking and its subsequent entities that are returned by the server:

#### Booking

- id
- status: BookingStatus (see Booking status)
- origin\_lat
- origin\_lon
- destination\_lat
- destination\_lon
- distance

- duration
- journeys: Array<JourneyJson> (see Journey)
- ride: RideJson (see entity Ride)
- hash: correspond to the journey

#### Journey

- booking\_id
- step\_number
- origin\_lat
- origin\_lon
- destination\_lat
- destination\_lon
- type
- start\_time
- finish\_time
- distance
- duration
- status: JourneyStatus (see Journey status)
- route: JSONObject

#### Ride

- booking\_id
- step\_number
- load
- vehicle\_id
- lat
- lon
- bearing
- status: JourneyStatus (can be null)
- desired\_pickup\_time
- origin\_lat
- origin\_lon
- destination\_lat
- destination\_lon
- start\_time
- finish\_time

#### Booking status

- Created
- Accepted
- Canceled
- Rejected
- Completed
- Error



### Journey Status

- Matching
- Rejected
- Scheduled
- ApproachingPickup
- WaitingForBoarding
- Boarding
- WaitingForDeparture
- Traveling
- WaitingForAlighting
- Alighting
- PassengersDroppedOff
- Completed
- Canceled
- NoShow
- Interrupted
- Closed
- Failed
- Rescheduling
- ReschedulingFailed

Once the booking is scheduled, a user can share his journey thanks to the hash generated booking hash.

The third endpoint allows the client to retrieve the booking updates so that it can show visual updates while the user is traveling. This endpoint works very similarly to the previous one. Each time the client requests a booking update, the Application Backend puts together the above information. As before, to get all the information it has to perform several API calls to the on-demand service provider. The Application Backend calls this endpoint from bestmile:

1. <https://api.bestmile.com/documentation#get-/booking/v5/bookings/-BookingID>

A booking can also be canceled with the provided booking id. This can only be done before the user gets on the bus. In the Application Backend server we call this endpoints from bestmile:

1. <https://api.bestmile.com/documentation#get-/booking/v5/bookings/-BookingID/-cancel>

The overall architecture has not changed since the deliverable D4.8. Please refer to chapter 6.10 On-Demand booking for more information.

### Frontend

From the last deliverable, the main changes have been done in the frontend by integrating the new api, enhancing the user experience with new designs, notifications to notify the user about the status of their travel, and a booking history page.

### Integration of the new API

We have updated the calls to communicate with the new api to use the new on demand service of Bestmile. Once a quote is created, these are the services that are available from the app related to the booking:

- create a booking which takes the id of the quote and the id of the traveler as parameters and returns the booking object described by the following attributes:
  - *id* → id of the booking
  - *status* → information on the status of the trip and its progress
  - *origin\_lat* → latitude of the starting point
  - *origin\_lon* → longitude of the starting point
  - *destination\_lat* → latitude of the destination point
  - *destination\_lon* → longitude of the destination point
  - *distance* → the distance travelled to make the journey
  - *duration* → the travel time
  - *journeys* → a table of objects journey that represents the stages of the journey
  - *ride* → which represents the bus trip made for this booking
  - *hash* → a unique and secure series of characters to identify a booking mainly to be able to share it with others
- get a booking details, using its id, this call retrieves all the information about the booking as described in the create a booking service. When the user is currently in a booking session, the endpoints to get the bookings are called every 3 seconds to simulate a real time visualization.
- cancel a booking, using its id, the user can cancel a reservation. However, this cannot be done at any time. Indeed, to be able to cancel a reservation, the traveler must not be in the AV which starts the trip. It is no longer possible to stop the AV at this moment.
- get a booking session from a hash, using a hash which is a unique identifier of a booking, a user can retrieve the details of a booking even if he is not the originator. This service is used for example for the Follow my kid.
- register for booking notifications, it is possible for a user to accept to receive notifications, the application sends to the server a token, which allows to identify the user's phone, as well as the id of the booking in question. This feature will be described in more detail in the Notifications section.

### User Experience and User Interface Updates

One of the main user improvements for 'On-Demand' service was the map zoom depending on the status of the booking and the notification received while traveling. The goal is to display a coherent view related to the current step to the travel or inform them about it if the app is not in foreground.

#### Matching and Scheduled

When the data provider is loading the trip and assigning it to a vehicle, the status sent by the data provider is 'Matching', then 'Scheduled' once the AV is assigned. There, the map zooms to show the departure pin and the destination pin on the map. The header text changes to "Loading the trip..." and the app is waiting to receive the AV information.

#### Approaching pickup

When the AV is assigned and moves to the departure stop, the Traveler App receives the status 'ApproachingPickup'. The map zooms to get the departure point marker, the departure stop marker and the bus icon. The header text changes to "Bus coming to pick you up..".

#### Waiting for boarding, boarding and waiting for departure

Once the AV arrives at the departure location, it waits for the traveler to board. The status of the booking changes to . The map zooms to get the departure point marker, the departure stop marker and the bus icon. The header text changes to "Bus coming to pick you up..".

#### Traveling

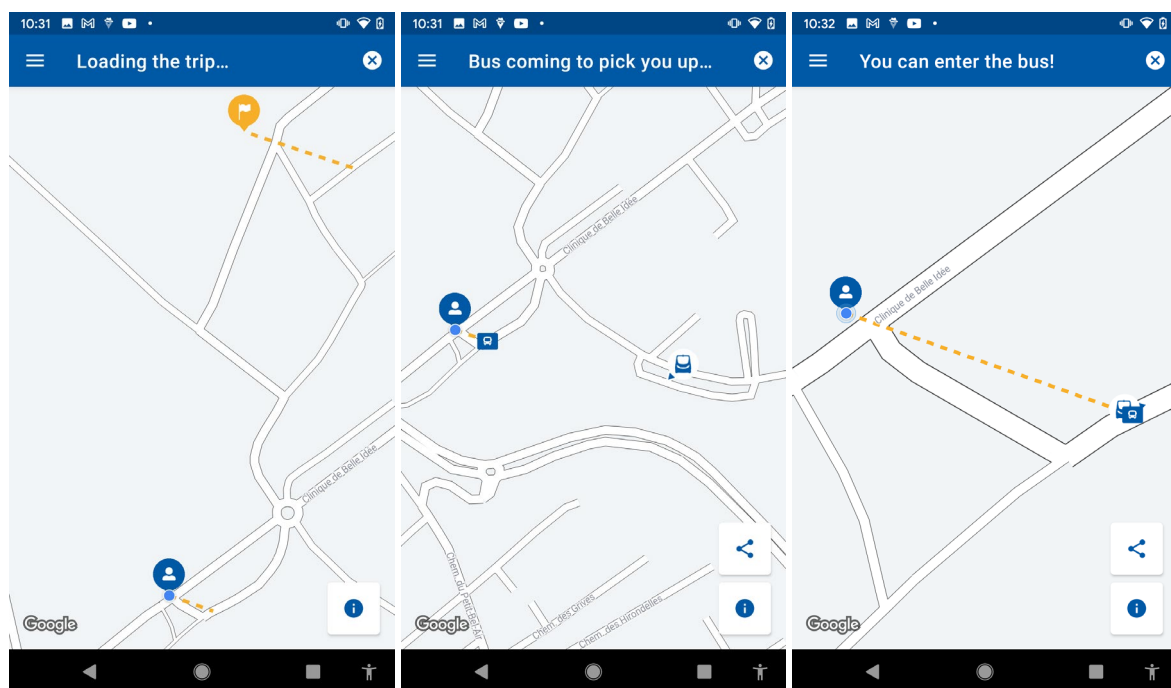
When the AV begins its trip to its destination, the Traveler App receives the status 'Traveling', until it arrives at the destination stop and the header changes to The header text changes to "Bringing you to destination". The map zooms to get the user departure marker, the departure stop marker, the user destination marker, the destination stop marker and the bus icon.

#### Waiting for alighting, Alighting

When the AV arrives at the destination stop, the Traveler App receives the status 'WaitingForAlighting'. Once the bus is ready to let the traveler to alight, the status changes to 'Alighting'. The header text changes to "You arrived, please exit". The map zooms to get the user destination marker and the destination stop marker.

#### Passengers dropped off, Completed

When the user alights from the bus, the status of the booking changes to 'PassengerDroppedOff' then 'Completed'. Here the map zooms to get all the markers of the trip including the path taken by the bus to arrive at destination.



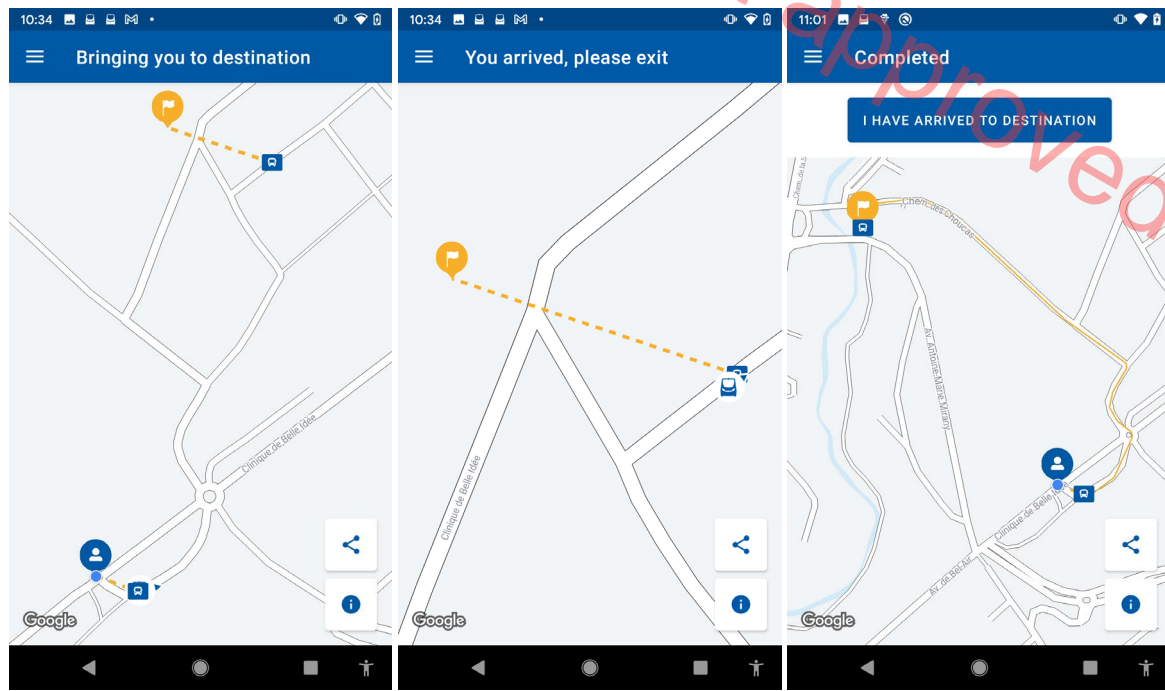


Figure 27 - Screenshots of the map zooms depending on the booking status

#### Cancelled, Interrupted, Closed, Failed

These Booking statuses bring the user back to the homepage with a toast message informing the user that his trip has been cancelled, Interrupted, Closed or has failed.

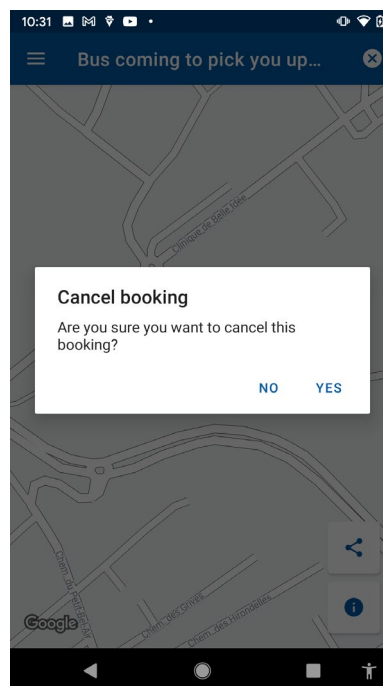


Figure 28 - Cancel a bookings

To sum up, the following diagram shows the process of the statuses of the booking and the statuses of the journey.

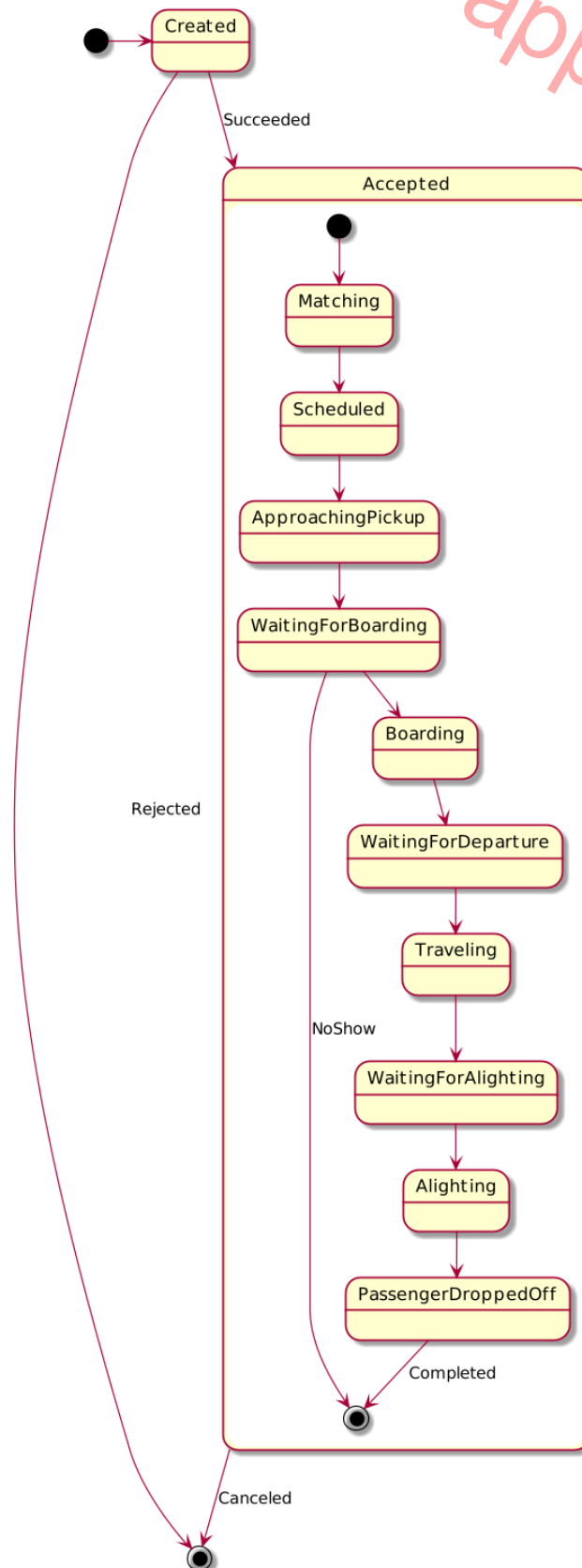


Figure 29 - On demand booking statuses diagram

Note: If the user accepts the notification, each of these statuses will be received by their phone via a web push notification. (See Chapter Notification).

Regarding other UI upgrades, from the last deliverable, we have updated the interface of the quote selection according to users feedback. We mainly worked on the size of the items and standardized the sizes of the texts and the margins.

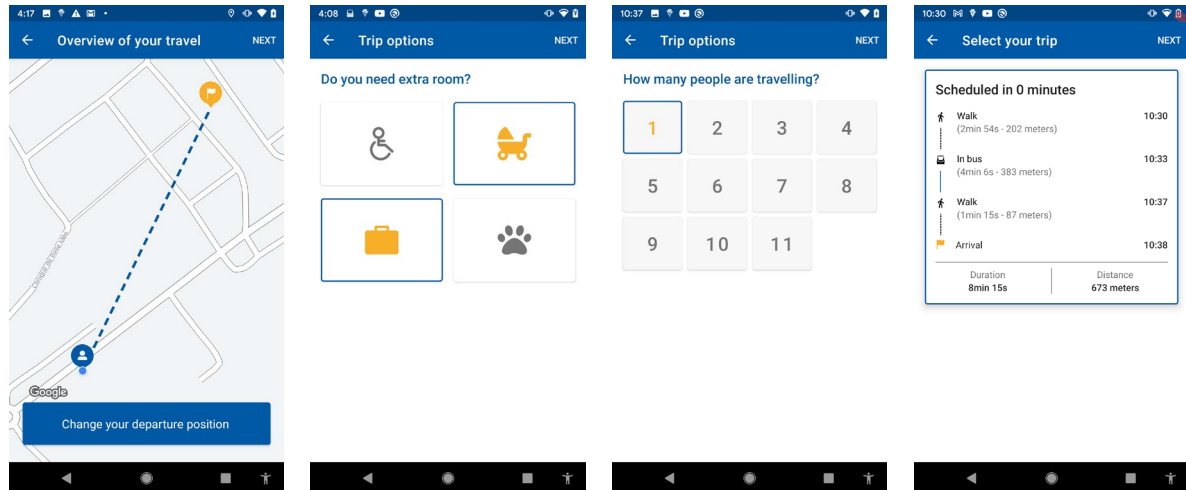


Figure 30 - On-demand booking - Quote selection process new designs

We also reworked the final recap page at the end of a trip, when the user clicks the button “I have arrived”.

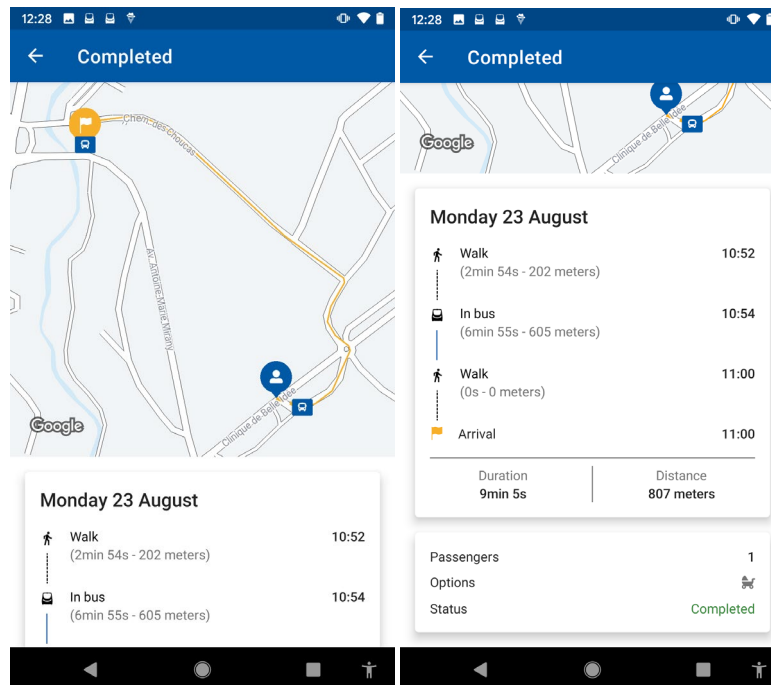


Figure 31 On-demand booking - Quote selection process new designs

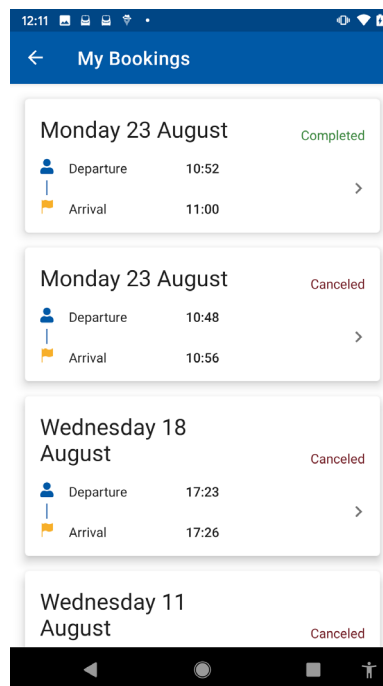
This view is divided into the 3 reusable components we described earlier:

- a static map that displays the information about the booking
- the information about the trip (date, steps, time, distance and duration)

- the information about the user's options (number of passengers, extra space) and the status of the booking.

### **Booking history**

The booking history page lists all the past bookings made by the user. They are stored in the internal memory of the phone. The first page is composed of cards that summarize the journey with the date, the status of the booking, and the time of departure and arrival.



*Figure 32 - On-demand history list*

The card is clickable and redirects the user to the detailed page of the booking. Here, the information is obtained from the server thanks to the id of the booking.

The page is similar to the last trip recap once the user finish their journey thanks to the following reusable components:

- the static map that displays the information about the booking
- the information about the trip (date, steps, time, distance and duration)
- the information about the user's options (number of passengers, extra space) and the status of the booking.



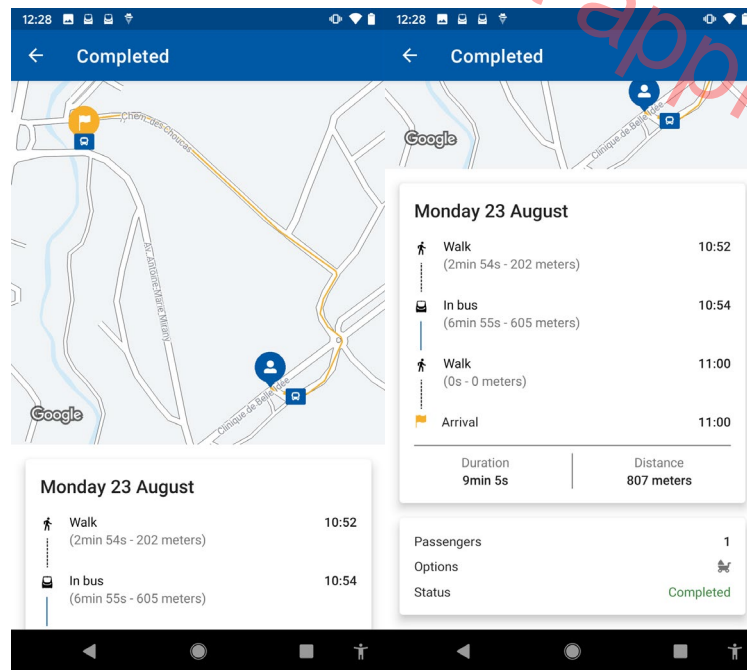


Figure 33 - On-demand history detail

### 3.10.3 Next steps

For Link the extra options with the space available for instance, someone with a stroller would need the equivalent of 2 seats in the AV.

As stated in the D4.8, chapter 3.10 “On-demand booking”, one of the main improvements this service needs is the improvement of its accessibility. It is mandatory to implement a second set of screens (a parallel workflow) that would allow a person with impaired vision to navigate the Traveler app’ and book an AV.

In short, the screens should focus on textual descriptions and not contain any Maps. The screens that were previously shared of what the On-demand booking for visually impaired people will look like are the following:

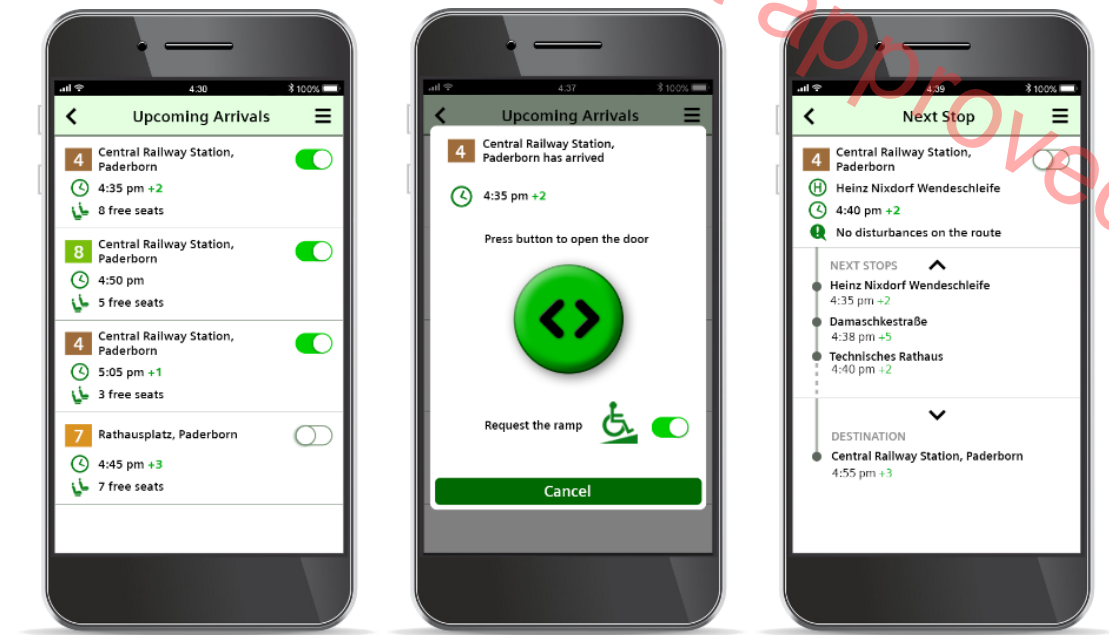


Figure 34 - Next steps - On-demand accessibility improvements

## 3.11 User feedback

Simple user feedback has been implemented but deactivated as it is not considered useful for the moment as no large scale testing has been done yet. We focused the development on the other services mentioned in this document. Hereafter is a description of the service User feedback that is taken from D4.8 chapter 3.11.

### 3.11.1 Initial vision

Userstory: As a user I want to be able to give my opinion on the service

Use case: User satisfaction is a huge concern to every Public Transport Operator. It is extremely important in usual environments but even more in a new, driverless environment. This service will be developed as a two way communication. Some feedback will be pushed by the PTOs, after a ride for example. Some will be pushed by the travelers to signal a broken seat, or any other problem.

Service: in-and-out-of-vehicle service

### 3.11.2 Work done

Analysis: The analysis of this service has been done jointly with the entire consortium through a WP2 workshop. All the results of these workshops are available in “D.14 Second Definition of AVENUE services”. Even if the User feedback service is not present in the original AVENUE proposal, it was the most voted service by the PTO’s.

User story: Service idea	Partners votes
Provide feedback: Via the mobile application	7
Be entertained: Via Wi-Fi	3
Be entertained: Via news and updates relevant to the area	3
Get assistance / aid: Via a direct call button to police, ambulance, assistance, etc.	4
Manage my trip: Via the mobile application	5
Know if there is room: Via the mobile application	6
Be informed about status: Via the mobile application	2
Feel safe: Via a call button inside the shuttle and at bus stop	7
Feel safe: Via sound / audio system that announces that the shuttle is approaching/leaving	3

Figure 35 - User feedbacks - Services idea votes results

From there, we defined several ways to “Provide feedback” as shown in the following table (extract from D2.14, chapter “2.1.1.As a user, I want to be able to provide feedback about the service”.

Idea	Feasibility	Partners involved & comments
Via a questionnaire provided inside the shuttle	Phase 2	MT Questionnaire could be integrated in the application
Via operator website formula	Phase 2	Operators
Via smiley system (buttons) inside the shuttle	Phase 2	Operators
Via photo upload to operator	Phase 2	MT / Operators Integration of a photo uploading function in a feedback form of the application
Via QR code in- and outside shuttle	Phase 2	MT / Operators A QR code placed in the shuttles could trigger a questionnaire in a feedback section in the mobile application
Via the mobile application	Phase 2	MT
Emotion-detection built-in camera inside the shuttle	Phase 3 / Not possible	Navya / CERTH Ongoing scientific research, see chapter 3.
Via thumbs up / thumbs down to camera inside the shuttle	Not in scope	
Via avatar on screen inside shuttle	Not possible	
Picture feedback (augmented reality -> via bus)	Not possible	

Table 1: Outcome feasibility study - user story 1: provide feedback

Figure 36 - User feedback idea planning

### 3.11.3 Next steps

In the coming month, Q4 2021/Q1 2022, we will implement more ways for the user to provide feedback. For example, we will provide a menu entry where the traveler will be able to submit a feedback with a Title, Content and upload/take a picture.

Once this mechanism is set, we plan to question the PTO's on their needs for feedback collection. We envision ways for PTO's to push bigger questionnaires with specific triggers. For example we could imagine that a PTO wants to push a satisfaction questionnaire to all the travelers that used the On-demand service within the last 30 days.

## 3.12 Follow my kid - Out of vehicle version

Major updates have been implemented for the service follow my kid - out of vehicle version. As a reminder, we've taken from the deliverable D4.8 the initial vision of this service and some of the feature description.

### 3.12.1 Initial vision

User story: As a user, I would like to see that my relative/friend is safe / As a user I would like to feel safe when travelling with the Autonomous Vehicles.

Use case: This service is designed to increase autonomy of non-fully autonomous people (Kids, Grandma, disabled people etc.). It would allow carers or family members to be sure that their beloved family members are safe while moving around the city using public transports. On the other hand, it would increase confidence to the non-fully autonomous people to use public transports knowing that their family can "be with them". The service and scenario proposes a full-fledged solution that allows designated "guardians" to follow the journeys of more vulnerable people, since the guardians can check the trip via a dashboard or mobile app, receive notifications via mobile app, add people to their "guarded" list, and share trips/position and E.T.A. with others.

### 3.12.2 Work done

Analysis: The initial version of this service is to allow a trusted person to "follow" another *followed* person on GPS based technology. The goal is to display on the *caregiver's* application the position of the *followed* person. On top of the GPS position, the *follower* can also see the booking status, just as they were the one using the AV service.

#### Feature description

To implement the Follow my Kid service in the Traveller App, we added two new features to the application: "share my trip" and "read only mode".

For a trusted person to "follow" another *followed user*, like a mom and her child, we have implemented in the Traveller App, the ability to share the user's booking. For instance, once a kid starts their travel in an AV, the app shows an icon to share it thanks to a link.

From there, the mother that receives the link clicks on it. The app opens the booking page, which is the same page for a normal booking, but in “read only” mode.

Until the trip of the kid is over the mother can see

- the kid’s position on a map as well as all the different booking status
- when the Kid is pickup by the AV
- the mother can get a notification when kid exit the AV or arrives to their destination
- From there, the mother can not follow her kid anymore.

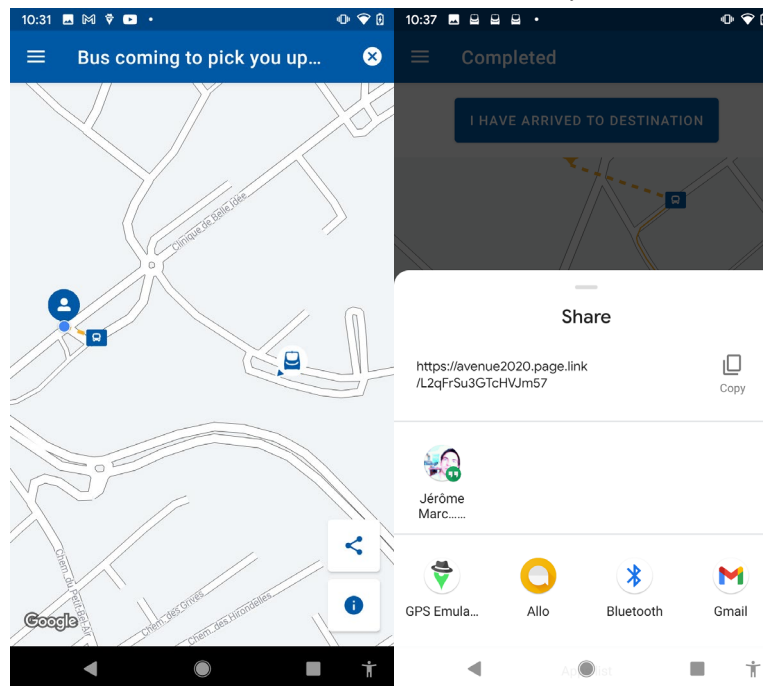


Figure 37 - Follow my kid - share user’s trip

Furthermore, as stated in the D4.8, “Follow my kid - out of vehicle version”, the more *innovative version* of this service (which is an In-Vehicle service and is described in D4.5, “2.6 Service: Follow my kid/grandparents”) is actually an addition to the initial version. Using face recognition and camera techniques, we plan to detect the face of the *followed* person and to provide either the videofeed to the *follower* while blurring the other people in the shuttle, or simply a notification that the camera based system also validates the presence of the *followed* person.

### Technical Description

#### Backend

This service required the integration of the new APIs to implement the possibility to follow a trip and to share a trip. Once the backend receives the booking from the data provider, it creates a hash that identifies it before adding it to the booking object that is sent to the Traveler App. The hash is a unique and secure series of characters that will be used by the server to retrieve a booking when one is shared. A new API request has been created for the Traveler App to get a booking from a hash.

Furthermore, this service required the integration of the new APIs to subscribe to notifications updates for a specific booking.

The server exposed those two api calls to the app to achieve this:

1 - `api/v2/booking/hash/{booking_hash}`, a get function that takes a hash and returns the corresponding booking

2- `api/v2/booking/notifications/register`, a post function that takes a booking notification request with the booking id, the token of the phone and a read only Boolean, as a body and gives the ability for the app to register to the corresponding booking notifications.

## Frontend

Implementing this service in the frontend, required the development of mobile deeplinks for the sharing feature and the development of the reception of the URL to redirect to the specific UI. A mobile deeplink is a URL that when clicked, opens directly the desired page of the application with the required resources. The application should be installed in the smartphone of the user, otherwise, they will be redirected to the Avenue H2020 website.

Once the “followed” person has started its on-demand journey, they can share by clicking on the new “share” button that we’ve added and that appears at the top of the information button in the bottom right corner of the page.

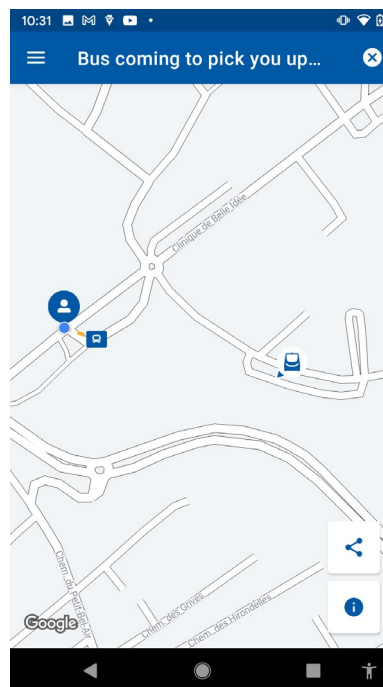


Figure 38 - Follow my kid - share trip button

To do so we’ve implemented the ability for the app to open in the specific page once it receives such a link using an external google service called Firebase. The deeplink is being read by Firebase and redirects the user to the app, at the condition that they are using a smartphone and that they have installed the app.

We’ve created one base deeplink “<https://avenue2020.page.link>” to which the application adds the hash of the booking. Then the process is the following:

1. Firebase is configured to open the app
2. At the starts of the application the URL is parsed to extract the hash, and it calls the server to get the booking from it
3. The app receives the booking and redirects the user to the booking page in a “read-only” mode.



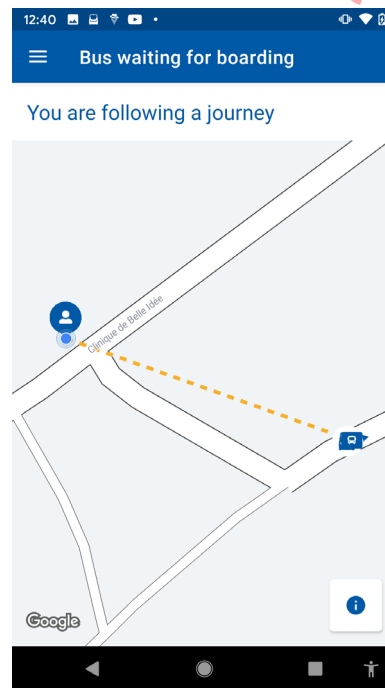


Figure 39 - Follow my kid - Following a trip

The read only mode shows the on-demand page with a text that indicates that the users is following a journey, without the ability to cancel a journey nor to share it. Furthermore, the header texts are adapted. For instance: “Bus coming to pick you up” would be “Bus coming to departure point”.

Finally, we’ve added the notification feature so that the following person can be notified even if the app is not active. The notification logic required integration for both platforms (Android and iOS) of the logic to manage notifications (See chapter 3.13 Notifications).

### 3.12.3 Next steps

Out-of-vehicle version of the Follow-my-kid service has been fully implemented. The next step would be to test and integrate the in-vehicle version of the feature. See D4.6.

## 3.13 Notifications

### 3.13.1 Initial vision

User Story: As a user, I would like to be notified when I am near an on-demand zone. I also want to know the current status of my trip even though the Traveller app is not in the foreground of my phone.

Use Case: This service is designed to give the user information without the Traveller Application being opened. Thus, thanks to the notification system we implemented two new types of service to the user:

1. If the user is inside an On-Demand area, they will be notified that they can order an AV from their position and travel inside the On-Demand zone.



2. If the user is traveling in an AV in On-Demand mode, they receive notifications about the progress of their trip. This way, they are aware of the steps of their trip without having to keep the application open.

### 3.13.2 Work done

**Analysis:** This service is an extension of the On-Demand service and the On-Demand zone. This feature allows the user to free his attention from the application .

#### Feature description

Thanks to the geofencing feature, the application can detect if the user is inside an On-Demand zone (cf. chapter 3.7 On-Demand zone). To extend this service, we added a notification feature that is triggered to the user's phone once they are inside the zone. In this case, the native system of the user's phone takes care of firing the notification. To be able to receive such a notification the user must complete their first On-Demand journey. At the end of the trip in the trip recap page, an alert is prompted to the user explaining the geofence notification feature. By clicking on the 'Set permission' button, they are redirected to their system settings where they can activate the background permission.

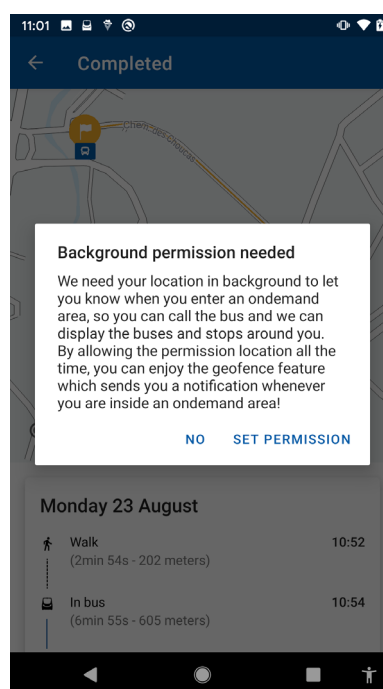


Figure 40 - Notification - Asking background location permission

To inform travelers of their On-Demand trip status in case they are not on the app, we implemented a push notification system. Provided that the user accepts the notifications, a device token, which identifies the smartphone, will be sent to the server to register it in the list of devices that accept to receive notifications. In this case, our server takes care of the distribution of the push notifications through Google's third-party service Firebase. This also required managing multi-brand applications to send notifications to the right application in case the user installed several Traveller Apps from different PTOs.

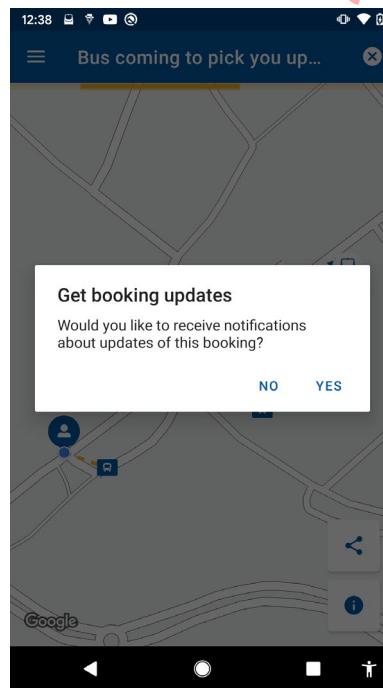


Figure 41 - Notification - Asking notification permission for booking status update

## Technical Description

### Backend

For the notifications about the status update of an On-demand trip, the application backend server needs to register to a webhook, by calling the *create webhook subscription* as described in the besmitle api documentation: <https://api.bestmile.com/documentation#get-/webhook/v1/webhooks/>

This request requires the following body:

#### BookingNotificationRequestJson

- **booking\_id**: representation the identifier of the booking related to the status notification
- **token**: string identifier of the user's phone
- **read\_only**: boolean to know if the request is made by a follower or not

The request creates a webhook subscription. Each time the ride status or the booking status changes, it will trigger a push notification, represented by the following 'BookingNotification' object and sent to the Traveler App, until the journey is completed.

#### BookingNotification

The booking notification is represented by the following data structure:

- **booking\_id**: representation the identifier of the booking related to the status notification,
- **booking\_status**: See Booking status in the chapter 'On-Demand'
- **read\_only**: boolean to know if the request is made by a follower or not
- **ride\_status**: see Journey status in the chapter 'On-demand'
- **type**: BookingNotificationType (enum with for now only one property: "booking\_update") and will be extended to other types in the future.

## Frontend

As explained, we extended two services to add the notification features.

The geofencing notifications for the On-Demand zone are exclusively managed in the Traveler App. The backend is not called here.

if, as explained in the feature description, the user enters the zone, the Traveler App creates a Notification and displays it to the user. Clicking on the notification opens the Traveler app at the homepage.

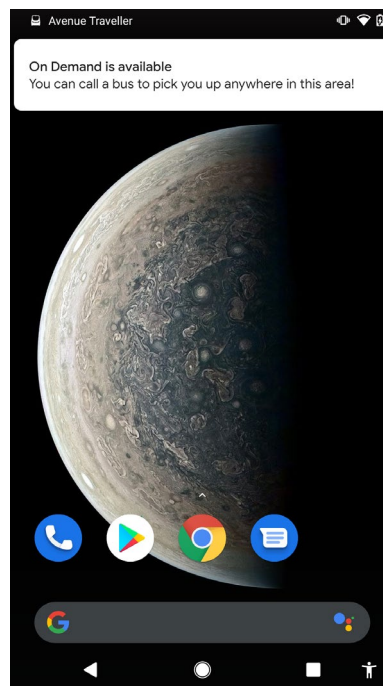


Figure 42 - On-demand Notification of availability of the service

The second notification service is about the booking status update. To do so, we integrated the firebase notification system in the Android and the IOS applications. The workflow for a user to receive the notifications is as follow:

- At the first launch of the application, the Traveler App generates a unique token to identify the phone and is sent to the server.
- When a user starts an On-Demand journey, and when the booking status is accepted, the application asks the user if they want to get updates about this booking.
- If the users clicks on yes, the application sends the following data to the post request "api/v2/booking/notifications/register":
  - booking\_id: the id of the booking the user wants to register to.
  - token: the unique identifier of the phone
  - read\_only: false if the notification is in read only mode

Note: In the Follow my kid service, the follower can also register to get notified by the updates of the booking of the followed person, the difference is in the texts of the notifications.

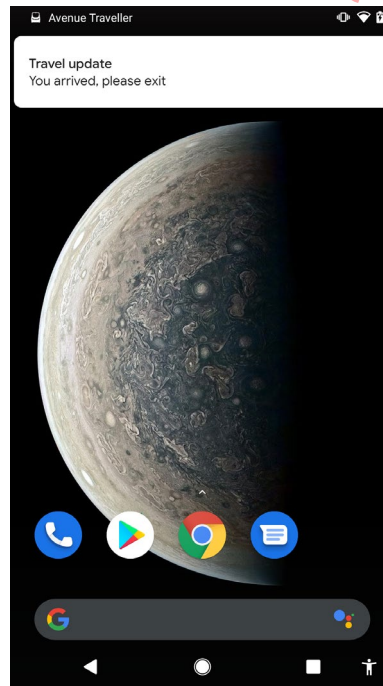


Figure 43 - On-demand Notification of the traveling status

### 3.13.3 Next steps

The notification feature can extend several out-of-vehicle services. For the moment the features described in this part were the unique ones decided by the consortium, but as the more the application will be used the more we will find new applications of it.

## 3.14 POI

### 3.15 Initial vision

User story: As a User, I would like to see what are the points of interests that I can travel to in the Area and be able to select them in the app.

Use case: This service came from feedback from Belle-Idée testers. The user of the service can discover the main points of interest in the “on Demand” area they are in. This allows the PTO to choose the points they wish to show the user. It also becomes easier for a user to select directly the point they want to go to and the one they want to travel from. Furthermore, it improves the accessibility of destination/departure selection as the map was not suitable for users with disabilities.

### 3.16 Work done

Feature description: From the quote selection process, at the destination selection page and the departure selection page, the list of the points of interests appears below the map. By selecting a point, the pin on the map updates its location. The main purpose of this service is to provide another easier and quicker way of selecting a destination point to the user.

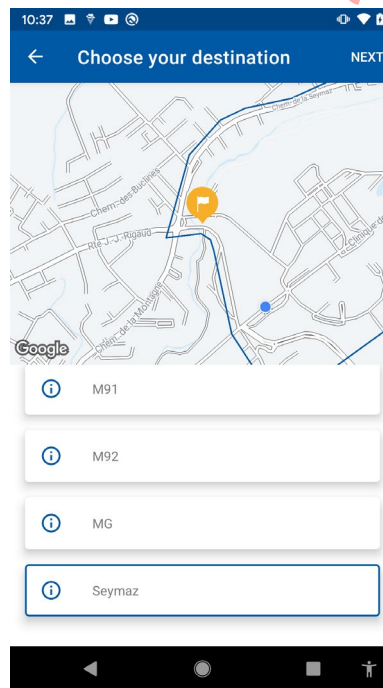


Figure 44 - Point of Interest screen

**Technical description:** The points of interest are configured in Bestmile's server and provided through an API call. The call gets the stops related to a given service.

### Backend

We have Implemented the API call that allows the application to get the list of all the stops provided by a given Service. The backend requests the list of the stops that are related to a particular service. The backend exposes the following API call to the frontend:

`api/v2/services/{service_id}/stops`

### Frontend

We've implemented the request that calls the server to get the stops related to a given service. The list of the stops is then processed in the application: we only display the name of the stop to the user and use its latitude and longitude to update the pin on the map once an item is selected. Furthermore, the list that displays the name of the point of interest is scrollable and fully accessible to users with disabilities.

The frontend gets the stops related to the service thanks to the following request:

→ `@GET("api/v2/services/{service_id}/stops")`

This call requires the id of the service from which the user makes the request.

## 3.17 Next steps

The feature has been implemented, tested and validated by the Belle-Idée users.

## 4 Operational work

### 4.1 Maintenance

We have maintained the Applications up and running (during more than a year of development at least 2 versions of both OS (Android and iOS) have changed requiring the retro support of the initial supported version plus the integration of the new version to guarantee full support on both platforms).

We also maintained and upgraded the version of several libraries / framework used on the Application server.

### 4.2 Bug Correction

Bug corrections and several UI improvements have been made throughout the entire project. The improvements had to be done on both Android and IOS platforms. That way the two applications would be at the same level.

### 4.3 On-site tests

We have been going to Belle-idée sites multiple times to test the application. This allowed us to test the On-demand process on-site and to identify areas of improvements.

## 5 Conclusion

In his deliverable we presented the services we developed in the third phase of T4.3 as well as the status of the deliverables originally described in the AVENUE proposal. We detailed all the work done: improved the architecture of the AVENUE ecosystem including the Application Backend and the Traveler App. Developed new features and reach a point where we could Book an automated shuttle using the Traveler Application without the intervention of an operator from A to Z.

Next, we present a list of next steps that we tackle in the following months and that we will integrate in the next deliverable.

## 6 Next steps

- Improve the overall accessibility of the Traveler Application
- Allow any source to plug-in to the AVENUE Application Backend
- Connect with the in-vehicle version of the Follow-My-Kid service
- Improve Android and iOS Traveler application based on User-feedback
- Create an operator Dashboard to monitor video based security service developed by CERTH (See D4.6).