"This project has been funded under the third AAL call, AAL-2010-3. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein"

PROJECT Nº: AAL-2010-3- 014

D22-Validation Tests Report

Start Date of Project : 01/03/2011 **Duration :** 30 months

PROJECT FUNDED BY THE AAL JOINT PROGRAMME	
Due date of deliverable	M26
Actual submission date	30/04/2013
Organisation name of lead contractor for this deliverable	HI-IBERIA
Author(s)	HI-IBERIA, UNIGE, BZN
Participant(s)	UNIGE, BZN
Work package	WP5 – User Acceptance Tests and Validation
Classification	PU
Version	1.0
Total number of pages	38

DISCLAIMER

The work associated with this report has been carried out in accordance with the highest technical standards and WayFiS partners have endeavoured to achieve the degree of accuracy and reliability appropriate to the work in question. However since the partners have no control over the use to which the information contained within the report is to be put by any other party, any other such party shall be deemed to have satisfied itself as to the suitability and reliability of the information in relation to any particular use, purpose or application.

Under no circumstances will any of the partners, their servants, employees or agents accept any liability whatsoever arising out of any error or inaccuracy contained in this report (or any further consolidation, summary, publication or dissemination of the information contained within this report) and/or the connected work and disclaim all liability for any loss, damage, expenses, claims or infringement of third party rights.

List of Authors

Partner	Authors
HIB	Alberto del Moral, Anna Mereu, Inmaculada Luengo
UNIGE	Jérôme Marchanoff, Katarzyna Wac
BZN	Attila Török

Table of Contents

List of Figures

Glossary

Acronym	Meaning
GTFS	General Transit Feed Specification
OSM	Open Street Map
PHP	PHP Hypertext Pre-processor
HTTP	Hypertext Transfer Protocol.
PoI	Point of Interest
HTML	HyperText Markup Language
API	Application Programming Interface
UPM	User Profiling Module
KDM	Knowledge Discovery Module
RPM	Route Planning Module
DGM	Data Gathering Module

1. Introduction

1.1. Objective of the Task 5.2

The D22 – Validation Tests Report deliverable describes the results of the Task 5.2 titled "Validation Results and Testing". The objective of this task has been the technical validation of WayFiS services, including the design, planning and execution of tests. The tests will be focused on core modules functionalities (User Profiling Module, Knowledge Discovery Module, Route Planning Module, Data Gathering Module, Localization and Positioning Module) as well as final services functionalities validation (Web application, Mobile Service). The objective is therefore the testing of the service in order to verify whether the technical specifications have been satisfied and justify possible discrepancies or design changes with respect to the defined service specifications in the D9 Technical Specification.

First of all, the Test Design will be described in Section 2 where the test types and design will be described as well as the test templates will be presented: the list of performed test cases will be presented in Section 2.3. The test planning is presented in Section 3 and the results are presented in Section 4. The conclusions are derived in Section 5.

1.2. Testing Methodology

This deliverable concerns the results obtained after software validation and verification for the WayFiS components developed during the project.

The methodology used to conduct the tests on the WayFiS platform followed a progressive approach. This implies that firstly conducted tests were on the most basic functionalities and once these were completed, we moved on to more complex ones. In turn, the initial scope of the tests was only local, being the modules the first entities tested independently, to pass later on to the integration and communication testing, and eventually to the services testing.

The first tests were conducted on the core modules independently, ensuring they provided the expected functionalities and were working properly. Once the core modules were tested and working, then we moved on to the modules integration testing, which was crucial to the implementation of some of the most important features involving intercommunication between different modules. After confirming the modules were able to communicate properly, we proceeded to test the services together. These services, as defined in the technical specifications of D9, are derived from the user requirements, and therefore, they are expected to provide the most important WayFiS functionalities to the users. Then, being this the most visible layer to the users, it needed to be tested extensively.

Tests have been performed and controlled manually: the outcome of each test has been analyzed in detail in order to derive from it the possible errors (or suggestions for improvements) that may exist.

The test results are shown in a table following the pattern OK, OK *, NOK. OK is given to cases in which results were fully satisfactory, OK * represents a successful test case but which has certain minor errors, and finally NOK is for test cases that did not end with the appropriate result.

2. Test Design

2.1. Test Types and Scenarios

Different types of tests have been performed in order to test the different parts of the service. But first of all, to understand the purpose of the tests is necessary to know the structure of the system and the main modules in which it is divided. The following illustration depicts schematically WayFiS technical architecture:

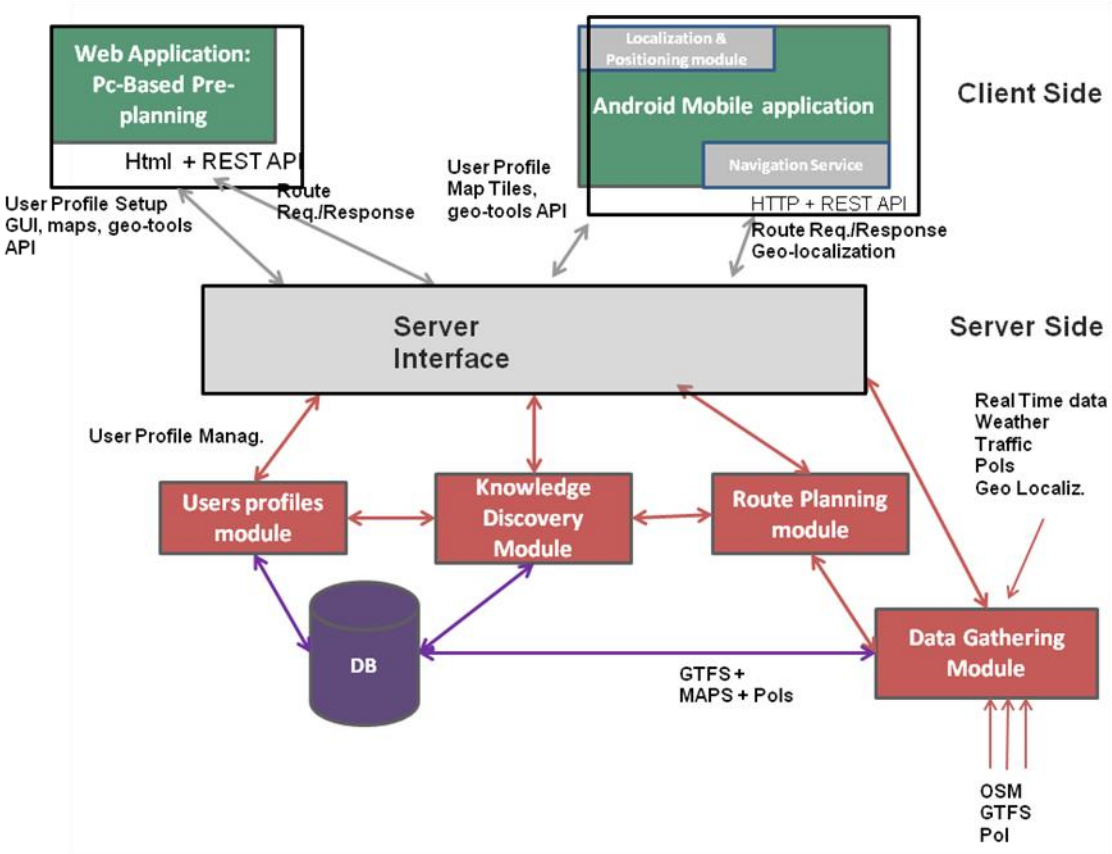

Figure 1 WayFiS Architecture

In this picture three main parts can be identified:

-The client side: composed of *Web Application: PC-Based Pre-planning* and *Android Mobile Application*

-The server side: composed of the *database* and the four main modules, *User Profile Module*, *Knowledge Discovery Module*, *Data Gathering Module* and *Route Planning Module*.

-The server interface: the layer that allows communication between the components of the client with the server side. As shown in the illustration, client components can only communicate with the server modules through the interface of the server, the server modules being the only with direct access to the database.

The tests were first conducted to check that all of these components work correctly in an independent manner, and then it was checked if they were able to function satisfactorily

together. For this we proceeded to verify the communication between the server-side modules, and the clients' interactions with the server and the database through the server interface.

Integration testing performed on the server-side modules is directly linked to the relationships between these, as can be seen in the figure. That is, the integrations to check would be UPM-KDM, KDM-RPM and RPM-DGM, and secondly UPM, KDM and DGM with the database.

With few exceptions, the requests to the database by the server modules are made through access server side scripts implemented in the UPM. The services and access points offered by these scripts are part of the server interface, and have a secure access layer implemented by including OAuth authentication protocol. This ensures that only the owner of the data can access them, keeping them well away from outside intervention. Queries by the modules to retrieve user data must be properly made on behalf of the user, signing petitions with the corresponding user tokens to ensure the authenticity of the request.

The components forming part of the client side are largely dependent on the services offered by the server and its interface, and do not provide great value or functionality themselves. Because of this, tests on clients in isolation are not very significant, because even for less important tasks need to communicate with the server, either to send or retrieve data. Therefore there is not a section on testing clients independently as such, however, they are already present as part of the services testing. That is because, as already mentioned above, the clients are dependent upon the services of the server to perform its tasks.

The criterion for the classification of test cases is based on the procedure followed during the testing process. That is, independent local testing of components, followed by integration testing and, eventually, service integration testing. The latter two in particular provide significant added value to the system considering that they allow the system components to cooperate and perform as one entity, increasing the overall complexity but, above all, the features and capabilities of the system.

The types of tests that have been carried out on the platform are:

Type of test	Description
Core Module test	Testing of the core modules functionalities
Modules Integration test	Testing the integration between modules
Service Integration modules	Testing that the services are well integrated within the infrastructure
Service functionalities Testing	Testing that the functionalities provided by the services adhere With the specifications

Table 1 Test Types

Tests have been carried out in specific settings and contexts, which endow each test with a particular significance.

The scenarios in which the tests are developed have been individually chosen to represent a particular situation that would provide a specific value to the test that would be conducted. In

certain cases, this means that the changes in the environment of the scenario have a major impact on the outcome, and therefore have a decisive role in the test as a whole.

The scenario in which a test was performed is specified in each test, and in general, each test was run in a single scenario. The scenarios used in these tests include:

Scenarios	Description
Scenario 1	User not registered (no account), no personal settings
Scenario 2	User registered. Personal settings set by default.
Scenario 3	OSM maps of Madrid, no GTFS data, with PoIs
Scenario 4	OSM maps of Budapest, with GTFS data, with PoIs
Scenario 5	OSM maps of Budapest, with routes recently created by user

Table 2 Test Scenarios

2.2. Test Template

The template used to document the test cases is as follows:

Code	
TestCase#01	
Purpose/Title:	Description of the test purpose
Type of test	Type of test with respect to Table 1 Test Types
Scenario of test	Scenario of test with respect to Table 2 Test Scenarios
Description	
This part describes the typical scenario through which the user could effectively use the service/modules capabilities	
Functionality to be validated	This part lists all the different service functionalities that will be tested.
Reference System Requirements (ref. D9)	
Functionalities with regards to the coding methodology used in D9.	
Notes	

Table 3 Test Template

2.3. Test Cases

This section describes the tests made on the WayFiS platform in order to check the system complies with the system functionalities specified in the D9. At this point, these functionalities are expected to be fully covered and operative, hence the final results need to be positive to consider this as a testing success. As it was explained at 2.1, these tests are broken down into categories, which in turn are divided into subcategories. Categories comprise the different types of functionality and integration tests, being the subcategories related to the modules or blocks that are specifically tested.

In order to illustrate these tests the template depicted above has been used.

2.3.1. Core Module Tests

2.3.1.1. *UPM*

Code	
TestCase#CMT01	
Purpose/Title:	Account management
Type of test	Core Module Test
Scenario of test	Scenario 2
Description	
The user logs into the system, enters and edits his account data, such as the account's password. Once saved, these changes are made permanent in the system. All these write and read operations are made by calling the write/read services offered by the UPM database scripts allocated in the server interface. The user logs out of the system and tries to log in again using the old password, but not getting access this time. Right after he tries with his new password and, as expected, he is granted access.	
Functionality to be validated	-Once logged in, the user is able to edit his account data and save it. -The profile data changes are effective right away. -After changing the account password, the previous one is no longer valid.
Reference System Requirements (ref. D9)	
UPM-01, UPM-02	
Notes	
The access to these resources is secured by OAuth in such a way a user can't access someone else's data but his, and vice versa.	

2.3.1.2. *KDM*

Code	
TestCase#CMT02	
Purpose/Title:	User data gathering for customized trip planning
Type of test	Core Module Test
Scenario of test	Scenario 3
Description	
The user goes to the trip planner application in the website. Then he sets the route start and destination points following any of the available procedures to do so, and finally plans the trip. Before the trip details are sent to the RPM, the KDM retrieves the user's data making a request in behalf of the user to the UPM's OAuth-secured script services and adds this data to the user's trip request. When the KDM receives a response from the RPM containing the outcome of the planning in a XML file, it analyzes the data and passes it on to the Client, who shows the information to the user in case it suits his expectations and limitations; otherwise it informs the user accordingly.	
Functionality to be validated	-The user is able to launch a route planning calculation. -The UPM is able to read from the user's data to complete the KDM's trip requests. -The KDM is able to communicate with UPM in order to complete the request data, so that the outcome of the RPM can be customized to the user needs. -The KDM is able to interpret the RPM's response.
Reference System Requirements (ref. D9)	
KDM-01, KDM-03, KDM-04, UPM-03	
Notes	

2.3.1.3. *RPM*

Code
TestCase#CMT03

Purpose/Title:	Route planning module tests
Type of test	Core Module Test
Scenario of test	Scenario 4
Description	
The Route Planning Module will have to gather the input from the Knowledge Discovery Module on route origin-destination, intermediate steps, required preferences, maps, POIs. The Route Planning Module will have to provide route planning function for public transportation with the inputs provided by the Knowledge Discovery module. The Route Planning Module will have to provide a set of optimal route to the Knowledge Discovery Module.	
Functionality to be validated	-The RPM is able to consider information coming from the KDM during the route planning procedures. These can include different parameters (origin-destination points, walk length), required types of PoI to be visited and user preferences. -The RPM offers multi-modal route planning and is able to provide input for KDM for further use. -The RPM is able to provide routes based on different criteria or preferences as requested by the KDM.
Reference System Requirements (ref. D9)	
RPM-01, RPM-02, RPM-03	
Notes	

2.3.1.4. *DGM*

Code	
TestCase#CMT04	
Purpose/Title:	Data Gathering Module tests
Type of test	Core Module Test
Scenario of test	Scenario 4

Description	
The Data Gathering Module will have to be connected to a map provider in order to allow having always up-to-date map files. The Data Gathering Module will have to be connected to official transportation agencies' transit database to collect public transportation schedules and associated geographic information. The Data Gathering Module will have to be enabled for synchronization of scheduling data with user-generated information provided with Open Street Map. The Data Gathering Module will have to be connected to source of information for POIs.	
Functionality to be validated	-The DGM is able to connect automatically to specified map providers from where it can download the recent map updates. -The DGM is able to connect automatically to specified transportation agency databases from where it can download the recent transportation schedules. -The DGM can synchronize the scheduling with the map information and generate the proper graph data for route planning purposes. -The DGM can connect to the PoI database and update it accordingly with the PoI information extracted from OSM data.
Reference System Requirements (ref. D9)	
DGM-01, DGM-02, DGM-03, DGM-04	
Notes	

2.3.1.5. *Localization and Positioning*

Code	
TestCase#CMT05	
Purpose/Title:	User current location
Type of test	Core Module Test
Scenario of test	Scenario 4
Description	
The user goes to the trip planner view in the mobile application. Then he click on the "my current location" button beside the departure field. The system displays his current	

location on the map with the departure marker. Given his current location, the user can generate a trip (from “here”) and starts the navigation. A marker with an arrow displays his location and direction. The user’s current location service includes intelligence upon how to get the best way his current location. Namely, the user enters a building, the GPS coverage is really low. The system is aware of the walking context and switches automatically to the Dead Reckoning provider. The user’s location is updated according to his steps. Additionally, when the user enter in the subway, there is no GPS coverage. The system is aware of the context and switches automatically to the Time Related provider. The system use the scheduling of the subway provided by the RPM to estimate the current location of the user.	
Functionality to be validated	-The system is able to get the location of the user when the user is outdoors using the iGPS module. -The system is able to get the location of the user when the user is indoors a building using the Dead Reckoning module. -The system is able to estimate the current location of the user when he/she takes public transportation based on the schedule. -The system is able to switch automatically between different providers when the user’s context changes.
Reference System Requirements (ref. D9)	
MOB-03, MOB-04, MOB-05	
Notes	

2.3.2. Modules Integration Tests

Code	
TestCase#MIT01	
Purpose/Title:	KDM-UPM integration – User data retrieval
Type of test	Modules Integration Test
Scenario of test	Scenario 3
Description	

A user wants to make a new route, so he sets up all the points needed on the map and launches the planning. However, the trip plan request doesn't only need the route locations, but also the user's data to make possible the trip customization. Therefore, the KDM asks the UPM for the user's data. This request is made by the KDM in behalf of the user, who is the actual owner of the OAuth tokens needed to access this data. In order to be able to query the UPM database for data, the KDM calls the UPM script services. Then, the UPM provides the KDM with the information, so that the KDM can wrap up all the data and eventually send the request for the trip.	
Functionality to be validated	-KDM is able to communicate with UPM to retrieve user data. -UPM can provide KDM with user data. -KDM can process UPM-generated data and add it to a trip request.
Reference System Requirements (ref. D9)	
RPM-03, KDM-01, KDM-03	
Notes	

Code	
TestCase#MIT02	
Purpose/Title:	KDM-RPM integration – Route planning request
Type of test	Modules Integration Test
Scenario of test	Scenario 4
Description	
A user wants to plan a route, so he sets everything up on the map and launches the planning. Then the KDM gathers the info to complete the request and sends it to the RPM. This integration is performed by a REST API request to the route planner service. RPM receives a planning request, processes it and gathers context trip data (GTFS, PoIs, and so on) and plans the trip according to the user limitations and needs. RPM sends it back to the KDM, who processes it and passes it on to the Client to display it on the screen in case it fits what the user needed. This has been achieved thanks to the exchange of XML files including the calculated route response.	

Functionality to be validated	-KDM is able to communicate with RPM to make planning requests. -RPM is able to plan a trip meeting the specific needs of a user. -KDM is able to understand a trip plan and show it accordingly.
Reference System Requirements (ref. D9)	
KDM-03, KDM-04, RPM-01, RPM-02, RPM-03	
Notes	
Integration has been made possible thanks to the usage of defined REST APIs for querying the route planning modules GET request and by using defined XML scheme for route response.	

2.3.3. Service Functionalities Test

2.3.3.1. *Planning Route Service*

Code	
TestCase#SFT01	
Purpose/Title:	Route Planning Service
Type of test	Service Functionalities Testing
Scenario of test	Scenario 1 – Scenario 5
Description	
The Route Planning Service provides planning functionality in the context of the whole end-to-end WayFiS system, considering the requests and problems of the different Scenarios.	
Functionality to be validated	- The route planning service is able to accept requests coming from different sources (mobile or web browser), calculates the route plan based on request (parameters, preferences), composes the proper response (transport information, walking trail, transfers, etc.) and sends back this to the requester.

Reference System Requirements (ref. D9)
Service-03, Service-06, MOB-01, MOB-06, RPM-02, RPM-03
Notes

2.3.3.2. *Web App Service*

Code	
TestCase#SFT02	
Purpose/Title:	User registration and logging
Type of test	Service Functionalities Testing
Scenario of test	Scenario 1
Description	
A user, who doesn't have an account in the system, wants to create one and become a member. So he goes to the webpage and clicks the "Register" option. Then he fulfills the form with his username, email and password; in case the username is already taken he would have to choose a different one. In order to confirm that the email corresponds to the actual user, the system sends a confirmation email to the user's address, which needs to be validated in order to go on with the account creation. Once it has been validated clicking on the link provided in the email, the user account is created and the user can access the site with it. Now he goes back to the login screen and types in the username and password of his recently created account. If these credentials match his, he is granted access to the system and receives his personal OAuth tokens to access securely to his resources.	
Functionality to be validated	-The user is able to create a new account. -The user's email address validation is necessary for the account to be created. -After the account creation, the user's profile settings are set by default. -Once the user has successfully completed the account creation, he is able to log into the system.
Reference System Requirements (ref. D9)	

Service-01, UPM-01
Notes

Code	
TestCase#SFT3	
Purpose/Title:	Profile edition
Type of test	Service Functionalities Testing
Scenario of test	Scenario 2
Description	
The user logs into the system using his credentials and he is granted access into his account. As his account is brand new, the personal settings are set by default, waiting for the user to change them to the actual values that describe his health situation. Therefore, the user goes to the Profile Settings tab and edits his profile data. After modifying these values and saving the changes, the KDM gathers this information and calculates the health and mobility limitations out of these values. Then it commits the changes to the database through the service interface's scripts, so that they are available right away to be used as the user personal configuration in a trip plan. Now the user logs out, and logs back in. He goes to the profile settings page to check if the values now are the ones he set a while ago, and so they are.	
Functionality to be validated	-When a user gets access to the system, he can change his profile settings.
Reference System Requirements (ref. D9)	
Service-02, UPM-01, UPM-02	
Notes	

Code

TestCase#SFT04	
Purpose/Title:	Trip planning and personal sites creation
Type of test	Service Functionalities Testing
Scenario of test	Scenario 4
Description	
A user goes to the trip planning tool and decides to create a new route. So first of all he needs to set the source and destination points that will define the trip. For the start point he chooses to type in the address. As he types, the system displays a list of possible matches for the address the user is looking for, so that he can just select one of the suggestions to auto-complete it. These suggestions are loaded in real time, as the user types, from a geo-coding service, so the quality of the results is directly dependant of the extension and detail of the list of addresses of this service (if the service doesn't consider the user's address matches any known spot, there would be no results to show as suggestions). After having typed in some letters he finds the address among the suggestions, so he clicks on it and the point is automatically set on the map as the trip's departure point. Now he needs to set the destination point, so he decides he will do it by right-clicking on the exact point of the map the place is at. He zooms in to the street where his friend Steve lives, clicks the right button on the spot and sets the destination point. He decides it would be nice to stop by a shop, so he types in the shop's address in the text box for intermediate points, auto-completes the address and sets it up. Now that everything has been set up, the user proceeds to plan the trip. Before anything is sent to the RPM, the KDM gathers the user's personal settings from the UPM (public transportation, maximum walking distance, and so on) wraps it all up and sends it to the RPM. The planning request to the RPM is made through the RPM's REST API, a service that allows the clients (the KDM in this case) to specify the parameters of the route in a GET call as part of the URL. Then, the response the KDM receives from the RPM is a XML file containing the trip itineraries (in case there is more than one suitable path), the steps to follow and all the transit and PoI details. The KDM process the response and checks whether it fits the user needs, and if so, it passes it on to the Client to display the itineraries on the map. Each itinerary is labeled accordingly with its most relevant features (times, duration, transfers and so on).	
Functionality to be validated	-Set a point by typing in the address. -Set a point by clicking on a place on the map. -Set the source point of a route. -Set the destination point of a route.

	-Set an intermediate point to a route. -Plan a trip. -Choose among the possible itineraries for a trip. -KDM's retrieval of the user's personal data from UPM to complete a trip plan request.
Reference System Requirements (ref. D9)	
Service-03, Service-04, Service-07, KDM-03, KDM-04	
Notes	
Every point set on the map is stored, transparently to the user, in the UPM database as a user's recent point in the corresponding category (source, destination, intermediate). Every request and update made by the KDM to the UPM uses the OAuth authentication protocol to ensure the content stored is only accessed by its rightful owner.	

Code	
TestCase#SFT05	
Purpose/Title:	Re-selection of a recent route
Type of test	Service Functionalities Testing
Scenario of test	Scenario 5
Description	
The user created two routes recently, and now he creates a new one. He wants to check the first of the three, so he goes to the tab panel where all of them are and clicks on the corresponding tab, getting the old route displayed. He thinks he will not ever need to check that route again, so he closes it. But after a while he realizes he needs to check something again, so now he has two options: either go back to the trip planning tab and click on the start and end text fields to display a list with the most recent points used in each category or go to the recent routes menu and select the route there. He chooses the first option, so he goes to the departure point text field, clicks on it and selects the address he wants from the drop-down list containing his five most recent departure points. But right after he realizes that it will be faster if he sets up everything at once, so he clicks on the recent routes button and selects the route among the latest five (getting both the departure and destination points set up). Either way, the user launches the route planning and the system proceeds as usual.	

Functionality to be validated	-Have a look at a previously planned route -Reuse a recently used address -Reload the addresses of a recently calculated route
Reference System Requirements (ref. D9)	
Service-05	
Notes	
The recent routes list and the recent points showed as suggestions are both retrieved from the UPM by the KDM, through the UPM's request scripts service. These requests need the user's OAuth authentication as usual.	

Code	
TestCase#SFT06	
Purpose/Title:	Saving a route
Type of test	Service Functionalities Test
Scenario of test	Scenario 4
Description	
The user wants to save a newly planned route in the application, and gives the order to save the route. The system prompts him to enter an identifying name for the route, and then proceeds to store it. The route is expressed in an XML document, which must be attached to the rest of the route data that make up the record. The XML file will contain only the itinerary highlighted in the moment it was saved (in case there were many, only that one is saved). This communication is done via a POST call to a script in the server interface, which receives as parameters the data of the route (start point, end point, etc.) and the route as XML. All this is signed by the OAuth protocol with the user's key. Upon receiving this request, the UPM divides the process into two. First it transforms the received XML into a file and stores it in the appropriate directory, and then, extracts the remaining parameters and performs an insert into the database by entering a new entry in the corresponding table including the link to the XML file. After that, the system sends a confirmation message to the user to let him know that the process has completed successfully.	

Functionality to be validated	-Web Client is able to communicate with UPM through the server interface to save information -Web Client is able to save routes -UPM is able to store routes
Reference System Requirements (ref. D9)	
-	
Notes	
This functionality was added after the D9 was released.	

Code	
TestCase#SFT07	
Purpose/Title:	Saving a personal site
Type of test	Service Functionalities Test
Scenario of test	Scenario 4
Description	
The user wants to save a point of interest in the application, and gives the order to save the point. The system prompts him to enter an identifying name for the point, and then proceeds to store it. This communication is done via a POST call to a script in the server interface, which receives as parameters the data of the point (latitude, longitude, name, address, etc.). All this is signed by the OAuth protocol with the user's key. Upon receiving this request, the UPM extracts the parameters and performs an insert into the database by entering a new entry in the corresponding table. After that, the system sends a confirmation message to the user to let him know that the process has completed successfully.	
Functionality to be validated	-Web Client is able to communicate with UPM through the server interface to save information -Web Client is able to save points -UPM is able to store points of interest

Reference System Requirements (ref. D9)
Service-07
Notes

Code	
TestCase#SFT08	
Purpose/Title:	Loading a saved route
Type of test	Service Functionalities Testing
Scenario of test	Scenario 5
Description	
The user wants to load a route he saved some days ago, so he goes to the “Favorite Routes” section in the website and selects one of his saved routes. To make easier recognizing the route, the system shows the name of the route (given by the user) and the start and end point addresses. Then he clicks “View” and the route is retrieved and shown in the planning application. The routes are stored in the UPM as XML files, so every time the KDM wants to show a personal route it has to ask to the corresponding UPM’s script service for the route’s data, to which the UPM responds with the route information and the XML file containing the route itself in a format understandable by the trip planning application.	
Functionality to be validated	-Load a route
Reference System Requirements (ref. D9)	
-	
Notes	
The saved route request adheres to the usual OAuth authentication requirements. This functionality was added after the D9 was released.	

Code	
TestCase#SFT9	
Purpose/Title:	Loading a personal site
Type of test	Service Functionalities Testing
Scenario of test	Scenario 5
Description	
A user wants to create a new route, and he intends to use as start point one of his personal sites, already saved in a previous route. Then, the user opens the personal sites menu and selects the personal site he wants to use and sets it as start point. Immediately after, the point is loaded from the UPM and set on the map.	
Functionality to be validated	-Load a previously saved point.
Reference System Requirements (ref. D9)	
-	
Notes	
Every point set on the map is stored, transparently to the user, in the UPM database as a user's recent point in the corresponding category (source, destination, intermediate). Every request and update made by the KDM to the UPM uses the OAuth authentication protocol to ensure the content stored is only accessed by its rightful owner.	

2.3.3.3. *Mobile Service Module*

Code	
TestCase#SFT10	
Purpose/Title:	User registration and logging (mobile service)
Type of test	Service Functionalities Testing

Scenario of test	Scenario 1
Description	
A user, who doesn't have an account in the system, launches the mobile application for the first time. He want to create one, so he clicks the "Register" button. Then he fulfills the form with his username, email and password; in case the username is already taken he would have to choose a different one. In order to confirm that the email corresponds to the actual user, the system sends a confirmation email to the user's address, which needs to be validated in order to go on with the account creation. Once it has been validated clicking on the link provided, the user account is created and the user can access the mobile app with it. Now he goes back to the login screen and types in the username and password of his recently created account. If these credentials match his, the account has been well created and he is granted access to the system. Then he closes the app. Some time later he launches again the application. The application remembers his credentials and logs him in automatically.	
Functionality to be validated	-The user is able to create a new account. - The user's email address validation is necessary for the account to be created. -Once the user has successfully completed the account creation (and validation over email), he is able to log into the system. -The application remembers the user's credentials and logs him in automatically.
Reference System Requirements (ref. D9)	
Service-01, UPM-01	
Notes	

Code	
TestCase#SFT11	
Purpose/Title:	Profile creation / edition (mobile service)
Type of test	Service Functionalities Testing
Scenario of test	Scenario 2

Description
The user goes to the “Profile Settings” page for the first time. All of them are set by default to a default value. He updates his settings. When modifying these values, the KDM gathers this information and calculates the health and mobility limitations out of these values. Then it commits the changes to the database through the service interface’s scripts so that they are available right away to be used as the user personal configuration in a trip plan. Now the user closes the application, and launches it again. He goes to the profile settings page to check if the values now are the ones he set a while ago, and so they are.

Functionality to be validated	-After the account creation, the user’s profile settings are set by default. -The user can change his profile settings.
Reference System Requirements (ref. D9)	
Service-02, UPM-02	
Notes	

Code	
TestCase#SFT12	
Purpose/Title:	Trip planning (mobile service)
Type of test	Service Functionalities Testing
Scenario of test	Scenario 4
Description	
A user goes to the trip-planning tool and decides to create a new route. So first of all he needs to set the source and destination points that will define the trip. In order to do so, he can choose among three different procedures to set a point on the map. Now he needs to set the departure point, so he decides to just click on the “current position” button beside the departure field. The approximated address of his position is set in the field and a point indicates his position on the map with a departure marker. For the destination point he chooses to type in the address. As he types, the system displays a list of possible matches for the address the user is looking for, so that he can just select one of the suggestions to auto-complete it. These suggestions are loaded in	

real time, as the user types, from a geo-coding service, so the quality of the results is directly dependent of the extension and detail of the list of addresses of this service (if the service doesn't consider the user's address matches any known spot, there would be no results to show as suggestions). After having typed in some letters he finds the address among the suggestions, so he clicks on it and the point is automatically set on the map with a destination marker.

During his trip, he would like to pass by his friend's house. So he zooms on the street where his friend lives and long-press on his place. A selector is displayed and he selects the "Add Via-Point" option. A new field is created, filled with this address and a via-point marker is added to the map at this place.

Now that everything has been set up, the user proceeds to plan the trip.

Before anything is sent to the RPM, the mobile app gathers the user's personal settings from the User Profile (public transportation, maximum walking distance, and so on) wraps it all up and sends it to the RPM. The planning request to the RPM is made through the RPM's REST API, a service that allows the clients (the mobile app in this case) to specify the parameters of the route in a GET call as part of the URL.

Then, the response the mobile app receives from the RPM is a XML file containing the trip itineraries (in case there is more than one suitable path), the steps to follow and all the transit and PoI details. The mobile app processes the response and displays a choice windows were the user could choose his preferred itinerary according to the itinerary features (times, duration, transfers and so on).

The user selects an itinerary and the mobile displays it on the map.

Functionality to be validated	-Set a point by typing in the address. -Set a point using the "current location button". -Set a point by long pressing on a place on the map. -Set the source point of a route. -Set the destination point of a route. -Set an intermediate point to a route. -Plan a trip. -Choose among the possible itineraries for a trip. -Mobile app's retrieval of the user's personal data from UPM to complete a trip plan request.

Reference System Requirements (ref. D9)

Service-03, Service-04

Notes

Every point set on the map is stored, transparently to the user, in the UPM database as a user's recent point in the corresponding category (source, destination, intermediate).

Every request and update made by the mobile app to the UPM uses the OAuth

authentication protocol to ensure the content stored is only accessed by its rightful owner.

Code	
TestCase#SFT13	
Purpose/Title:	Saving a route (mobile service)
Type of test	Service Functionalities Test
Scenario of test	Scenario 4
Description	
The user wants to save a newly planned route in the application, and gives the order to save the route. The system prompts him to enter an identifying name for the route, and then proceeds to store it. The route is expressed in an XML document, which must be attached to the rest of the route data that make up the record. The XML file will contain only the itinerary highlighted in the moment it was saved (in case there were many, only that one is saved). This communication is done via a POST call to a script in the server interface, which receives as parameters the data of the route (start point, end point, via-points etc.) and the route as XML. All this is signed by the OAuth protocol with the user's key. Upon receiving this request, the UPM divides the process into two. First it transforms the received XML into a file and stores it in the appropriate directory, and then, extracts the remaining parameters and performs an insert into the database by entering a new entry in the corresponding table including the link to the XML file. After that, the system sends a confirmation message to the user to let him know that the process has completed successfully.	
Functionality to be validated	-The mobile app is able to communicate with UPM through the server interface to save information -The mobile app is able to save routes -UPM is able to store routes
Reference System Requirements (ref. D9)	
-	
Notes	
This functionality was added after the D9 was released.	

Code	
TestCase#SFT14	
Purpose/Title:	Saving a point (mobile service)
Type of test	Service Functionalities Test
Scenario of test	Scenario 4
Description	
After having created a route, the user wants to save one of his point as a point of interest in the mobile app. He clicks on the action menu and gives the order to save the point. The system prompts him to choose between the departure or the destination point and to enter an identifying name for the point, and then proceeds to store it. This communication is done via a POST call to a script in the server interface, which receives as parameters the data of the point (latitude, longitude, name, address). All this is signed by the OAuth protocol with the user's key. Upon receiving this request, the UPM extracts the parameters and performs an insert into the database by entering a new entry in the corresponding table. After that, the system sends a confirmation message to the user to let him know that the process has completed successfully.	
Functionality to be validated	-The mobile app is able to communicate with UPM through the server interface to save information -The mobile app is able to save points -UPM is able to store points of interest
Reference System Requirements (ref. D9)	
Service-07	
Notes	

Code

TestCase#SFT15	
Purpose/Title:	Loading a saved route (mobile service)
Type of test	Service Functionalities Testing
Scenario of test	Scenario 5
Description	
The user wants to load a route he saved some days ago, so he goes to the “Favorite Routes” section in the mobile app and selects one of his saved routes. To make easier recognizing the route, the system shows the name of the route (given by the user) and the start and end point addresses. Then he clicks on the row he wants and the route is retrieved and shown in the planning view. The routes are stored in the UPM as XML files, so every time the mobile app wants to show a personal route it has to ask to the corresponding UPM’s script service for the route’s data, to which the UPM responds with the route information and the XML file containing the route itself in a format understandable by the trip planning application.	
Functionality to be validated	-Load a route
Reference System Requirements (ref. D9)	
-	
Notes	
The saved route request adheres to the usual OAuth authentication requirements. This functionality was added after the D9 was released.	

Code	
TestCase#SFT16	
Purpose/Title:	Loading a saved point PoI (mobile service)
Type of test	Service Functionalities Testing
Scenario of test	Scenario 5
Description	

The user wants to create a new route using on of the points (PoI) he has saved before. He goes to the “Personal Sites” view and selects the place he wants. These places are identified by a name (set by the user) and the corresponding address. Then he clicks on the row he wants, then chooses if he wants to set this point as departure or destination and finally the planning view is displayed with this point set according to his choice. The point is stored as a personal point in the UPM database so to display the list of personal sites, the mobile app ask to the corresponding UPM’s script to which the UPM responds with the list of personal sites for the user.

Functionality to be validated	-Load a personal place
Reference System Requirements (ref. D9)	
-	
Notes	
The saved points request adheres to the usual OAuth authentication requirements.	

Code	
TestCase#SFT17	
Purpose/Title:	Navigation (mobile service)
Type of test	Service Functionalities Testing
Scenario of test	Scenario 4
Description	
The user has created an itinerary with public transportation. He launches the navigation by clicking on the “play” button. The system shows him his position on the map and the path to follow. On the top of the screen, a panel displays the next move and the distance. He walks to the corner of the street and turns where the app shows him. The panel is updated with the new next move information. After few minutes, he arrives at the end of the walking path and has to take the subway. The system shows him a pop-up window, with information about the line, direction, and scheduling of the next subway train. The user enters in the subway station and goes to the good platform. Then, the train arrives and the user presses the “I’m in” button. The app zooms out and displays the full trip of the subway. The top panel displays the distance and name of the exit station. During the full trip, the estimated location of the user is displayed on the map. 1 minute before arriving at the exit station, the app displays	

another pop-up window saying that the user should exit the train soon with once again the name of the exit station. The user exits the station and continues to follow the path displayed on the map. He turns few times according to the indication provided by the app and reaches his destination. The app displays a green check image (indicating that the trip has been done successfully).	
Functionality to be validated	- Real time navigation - Switching mode between Location providers - Positioning module
Reference System Requirements (ref. D9)	
- MOB-01, MOB-03, MOB-05	
Notes	

Code	
TestCase#SFT18	
Purpose/Title:	Voice guiding (mobile service)
Type of test	Service Functionalities Testing
Scenario of test	Scenario 3
Description	
The user has created a walking itinerary from his location. He launches the navigation and a voice (in preferred language) indicates to him the current move and the distance. He walks and each time around 10 meters before the next move, a voice tell him that he has to turn.	
Functionality to be validated	-Voice guiding (different languages)
Reference System Requirements (ref. D9)	

- MOB-02
Notes
The voice is only available in English, French and Spanish (Hungarian not supported yet).

Code	
TestCase#SFT19	
Purpose/Title:	Manual re-routing – “New trip from here” (mobile service)
Type of test	Service Functionalities Testing
Scenario of test	Scenario 5
Description	
The user is following the instructions provided by WayFiS. On the road, he sees a beautiful park and decides to do a walk inside. When he leaves the park by the other side, he is far away of the predefined path, so he clicks on the action menu and select the "New trip from here" option. The app requests the OTP server for a new trip from his current location to the previously selected destination and displays the response on the map.	
Functionality to be validated	- Triggering of path re-routing based upon the explicit user request
Reference System Requirements (ref. D9)	
- MOB-06	
Notes	

Code	
TestCase#SFT20	
Purpose/Title:	Manual re-routing – “Go back” (mobile service)

Type of test	Service Functionalities Testing
Scenario of test	Scenario 5
Description	
The user has made a trip from home to somewhere in the city using the app. He reached his destination and has made what he had to do. Now, he wants to go back home. As he has not closed the app, he clicks on the action menu and selects the "Go back" option. The mobile app requests the OTP server with his current location as departure and the previous departure point as destination. The OTP server returns the xml file containing the possible itineraries and the app displays the itinerary selector.	
Functionality to be validated	- Triggering of path re-routing based on the explicit user request
Reference System Requirements (ref. D9)	
- MOB-06	
Notes	

Code	
TestCase#SFT21	
Purpose/Title:	Automatic re-routing / off road detection (mobile service)
Type of test	Service Functionalities Testing
Scenario of test	Scenario 5
Description	
The user is following the instructions provided by the app to reach his destination. Lost in thought, he forgot to turn and continued for a while on the wrong way. The app detects that the user is no longer on the good way. The system displays a message, the phone tilts and vibrates but the user doesn't hear it. When he realizes that he forgot to follow the instructions, he looks at his phone and sees the pop-up messages offering him to get him back on the good way by creating a new itinerary from his current location or to let him going back by himself on the path	

displayed on the map. The user choses to accept the re-routing and clicks on "accept". The app requests the OTP server for a new trip with the current position of the user as departure point and the same destination as previously. The app displays the response using the itinerary selector.	
Functionality to be validated	- Automatic triggering of path rerouting from mobile real time info. - Route update Accept/Discard
Reference System Requirements (ref. D9)	
- MOB-07, MOB-08	
Notes	

3. Tests Timeline

The validation and testing phase took place from month M24 to M27, which is along the period between the beginning of March and the end of May 2013. The following table summarizes the timeline of the tests.

WayFiS Validation Tests - TIMETABLE	March 2013				April 2013				may-13			
	1st week	2nd week	3rd week	4th week	1st week	2nd week	3rd week	4th week	1st week	2nd week	3rd week	4th week
Core Modules Tests												
#CMT01												
#CMT02												
#CMT03												
#CMT04												
#CMT05												
Modules Integration Tests												
#MIT01												
#MIT02												
Service Functionalities Tests												
#SFT01												
#SFT02												
#SFT03												
#SFT04												
#SFT05												
#SFT06												
#SFT07												
#SFT08												
#SFT09												
#SFT10												
#SFT11												
#SFT12												
#SFT13												
#SFT14												
#SFT15												
#SFT16												
#SFT17												
#SFT18												
#SFT19												
#SFT20												
#SFT21												

Figure 2 Validation tests Timetable

4. Tests Results

Code	Result and Remarks
#CMT01	OK
#CMT02	OK
#CMT03	OK
#CMT04	OK
#CMT05	OK
#MIT01	OK
#MIT02	OK
#SFT01	OK
#SFT02	OK
#SFT03	OK
#SFT04	OK
#SFT05	OK
#SFT06	OK
#SFT07	OK
#SFT08	OK
#SFT09	OK
#SFT10	OK
#SFT11	OK
#SFT12	OK
#SFT13	OK
#SFT14	OK
#SFT15	OK
#SFT16	OK
#SFT17	OK
#SFT18	OK
#SFT19	OK

#SFT20	OK
#SFT21	OK

5. Conclusions

This documents reports the activities performed along task 5.2 “Validation results and testing”. In particular the testing of the different core modules has been reported with details on the specific testing scenarios and steps to be validated.

In particular a test template has been produced at the beginning of the task detailing all the information that needs to be specified and the functionalities/steps of execution that were needed to be verified in order to consider the test as successfully completed.

Scenarios of tests were also specified identified, different service implementation setups in terms of prototypes and maps details, detailing the available data for the considered tests. Moreover, each test made reference to a specific set of functionalities as defined in the D9 Technical Specification document.

The timeline of the tests has been also presented and all were conducted following the planned scheduling, only minor delays were encountered that did not compromise the planned flow of testing steps. Final results of validation tests have been also reported.

The successful execution of the validation tests has allowed meeting the planned deadlines for the availability of the prototype to be used along the end user test trials.