Project ref no	AAL-2013-6-131
Project acronym	Elders-Up!
Project full title	Adaptive system for enabling the elderly collaborative knowledge transference to small companies
Dissemination level	Public
Date of delivery	7/7/2015
Deliverable name	DR2.7 First Functional Requirements and API Specification for Services
Type	Report (R)
Status	Final
WP contributing to the deliverable	WP 2
Main contributors	ISOIN
Other contributors	GEO, IDENER, TUC
Author(s)	José Antonio Carvajal Sosa, Manuel Cano
Keywords	Functional requirements
Abstract (for dissemination)	The purpose of this internal report is to specify the first functional requirements for Elders-Up! project. The system is comprised of various sub-systems to be implemented by different partners, which must be interconnected and work together. Requirements are specified both with regards to what functionality the system will provide to end users, and with regards to what interfaces and functionality the various sub-systems will

	provide to users and to the other sub-systems.

Elders-Up!: Adaptive system for enabling the elderly collaborative knowledge transference to small companies

AAL-2013-6-131

Deliverable

R.2.7 First Functional Requirements and API specification for Elders-Up! services

Public

VERSION HISTORY

Version	Edited by	Date	Description
1.0	J. Antonio Carvajal	25st March 2015	Initial Outline
1.3	Alex Garcia	26th May 2015	Review
1.4	Jose Antonio	26th May 2015	Corrections & adjustment
1.5	Tudor Ciora	3th May 2015	Added some modules & review
1.6	Maarten Van Zomeren	5th June 2015	Added some modules & review
1.7	Maarten Van Zomeren	18th June 2015	Added some modules
1.8	Mark Fergusson	18th June 2015	Added some modules
1.9	Carlos Leyva	25th June 2015	Added system integration
2.0	Juan Rodríguez, José Antonio Carvajal	2nd July 2015	Final review

Table of Contents

1 Context and Background

The Elders-Up! project follows a user-centric system design methodology, in which participatory design techniques are used throughout the project (Figure 1). Deliverable 2.1 describes the user research. The user requirements are based on the findings from the user studies. The user requirements are input for the concept and architecture design (D2.6).

Figure 1: The five stages towards functional requirements in the Elders-Up user-centric system design methodology.

The findings from the user research have been studied in detail, and user requirements have been identified. The user requirements have been clustered in five categories: general, first-time registration, matchmaking, collaboration and adaptation. Next, the user requirements have been prioritized. The resulting user requirements have been validated by the project consortium, resulting in D2.3.

1.1 Goals

The main idea behind Elders-Up! Project is to bring elderly's valuable experience to start-ups and small companies, addressing intergenerational knowledge transfer to use skills and competencies based on experience.

The aim of the project is to facilitate transfer of knowledge from older adults to companies. User research has been conducted in three countries: The Netherlands, United Kingdom and Cyprus.

1.2 Guide to this document

This document is a result of the technical analysis, specification and early system design work in the Elders-Up! project. The main goal is to specify requirements, both with regards to what functionality the system will provide to end users and with regards to what interfaces and

functionality the system will provide to users. As the Elders-Up! system will be comprised of several sub-systems, developed by different partners in different corners of Europe, defining the roles of the sub-systems and the interaction between them is essential for the success of the project.

In the following section a brief introduction of the system will be explained. In chapter 3 the functionality of each module of the Elders-Up! system will be described. Once the functionality of each module is described in chapter 4 the integration between the system modules will be shown. Chapter 5 shows the set of functionalities of the whole Elders-Up! system. In order to complete this section the uses cases will be explained in Chapter 6. Once the functionality of the whole system is explained in chapter 7 we will see a priority list for the different functionalities. Finally, in chapter 8 and chapter 9 the figure list and tables will be shown.

2 System overview

2.1 User Roles

The primary users of the system can be categorized in two groups:

1. **Experts (or Older adult User)**

They provide experience and knowledge to start-up companies. The main role of this target group is to provide the skills obtained during their working period and help the start-up and small companies to grow through the platform. Each user from this group has some level of expertise in some specific area.

2. **Companies (or Company User)**

Companies may benefit from the experience and knowledge of the older adults. If there is a job opportunity for the older adult, the company can make a new offer in the platform and find the suitable person to be the part of the company's team.

In the user research, an optional third group of users has been identified:

3. **Moderators**

The moderators facilitate the matchmaking process. They can for example support companies and experts in creating their profiles, in finding matches, and in starting a collaboration process.

The moderators can be employees of the company that provides the Elders-Up! service, they can be third-party professionals, or volunteers. The first integrated prototype will not include functionalities specifically for Moderators. Based on the findings from the first field study, it will be decided if functionalities need to be added for Moderators.

2.2 Overall Architecture

- The Elders-Up! system is composed of modules that can be seen in the following image. Some groups which can be seen in the Elders-Up!

Architecture: There are modules which provide interfaces user to different users for different actions (ISS, ICAW, ISEU).

- Knowledge Base. It is the knowledge core that the system acquires thanks to modules gathering data.
- There are modules that improve the system's adaptation to the senior expert. (SDC, SRC, DM, ADM)
- There are a group of blocks that represent the matching between the experts and companies (PR, SR, SMS).

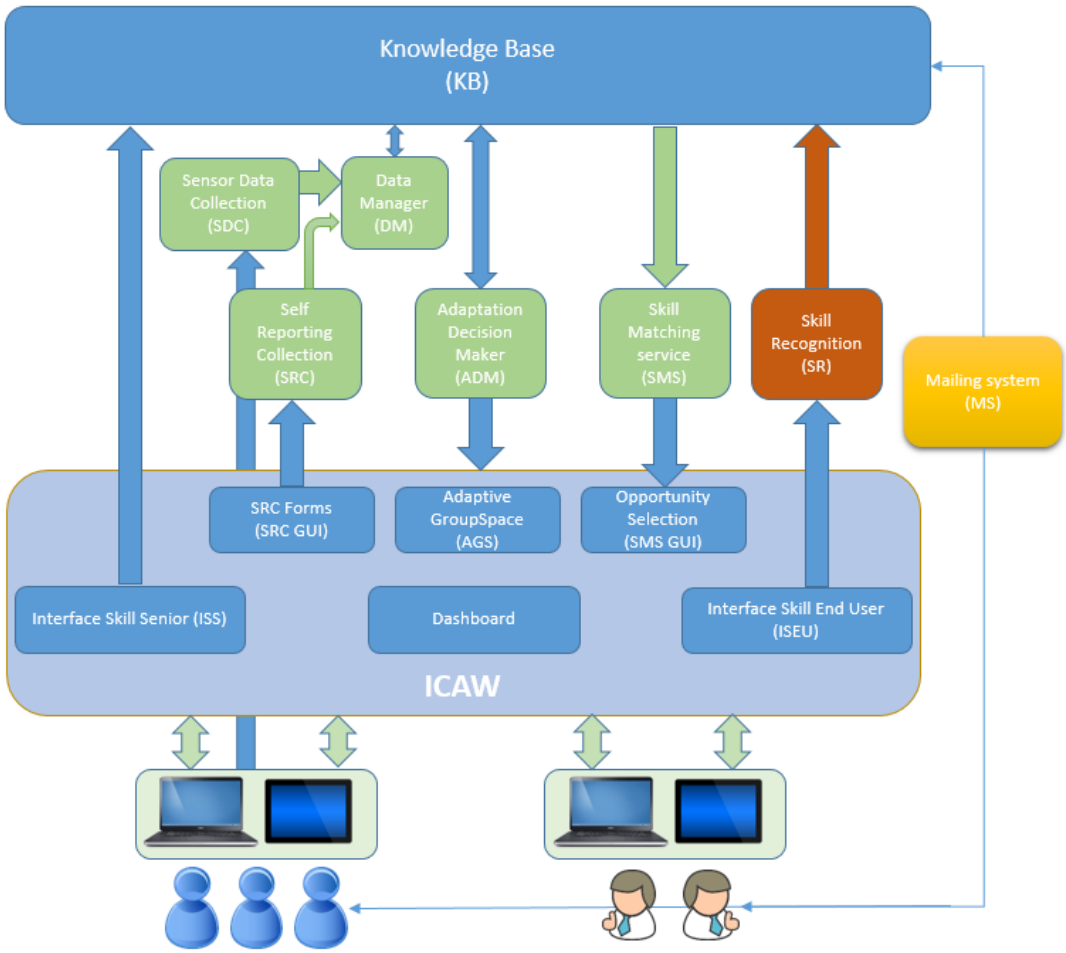


Figure 2: Elders-Up! application structure

3 ELDERS-UP! MODULES

In this chapter the functional requirements of the different modules will be described.

3.1 INTERFACE SKILL SENIOR (ISS)

The Interface Skill Senior (ISS) is the GUI that enables the senior to:

- Set up/amend profile
- Make manual adaptation amends (ie email frequency)
- View current opportunity matches
- Accept/reject opportunity matches

The ISS will initially allow the senior to input all of their profile information. It will store all of the relevant personal information, together with the users skills inputted either by the skills taxonomy or extracted from a free text box.

The user will then be able to amend profile settings, including the regularity, content and direction of flow of information from the system (ie emails updating them on current and potential opportunities and matches.)

The user can also view current matches and have the opportunity to accept reject any jobs they have been matched with.

3.2 DASHBOARD

The dashboard is the central menu page for the senior experts and the SME's. The dashboard provides access to the Adaptive Group Spaces, to the user profile, company profile, and to the search & match functionality.

More details and requirements can be found in D2.6 Concept and Architecture design.

3.3 Adaptive Group Space (AGS)

The workspace is the central location to support companies and their teams of experts in their day-to-day collaboration. Companies can ask for support, and both company members and experts are facilitated in communication, coordination and compensation.

The workspace consists of different elements. Requests can be used for tasks management. Within the collaborative agenda the team can manage their appointment. Contacting each other is made easy with the group messages. The messages are sent to the entire group. It is also possible to share documents.

The Adaptive Group Space can adapt itself to the cognitive conditions or physical limitations. The Elders-Up! system addresses these varying user capabilities by offering adaptation.

More details and requirements can be found in D2.6 Concept and Architecture design.

3.4 OPPORTUNITY SELECTION (SMS GUI)

The Opportunity Selection (SMS GUI) is the interface from where the users select matches that are provided by the skills matching service. It is separated into two parts:

1) The SME Matching selector
2) The Senior User Matching selector

The SME matching selector provides a list of senior users matched to the job opportunity input through the skill recognition (SR) module and is output via the SMS. The matches are weighted according to the algorithms within the SMS and ranked accordingly. The SME can then decide which of these matching profiles they wish to work with. The SME then contacts the user and invites them into the ICAW to collaborate.

Within the senior user matching selector, the senior is provided with 2 options:

- The chance to accept (or reject) a collaboration request from an SME (as detailed above). Should the user accept, then they are invited to collaborate in the ICAW along with the SME. Should they reject then the SME is notified and can then choose another potential match.
- The chance to browse current opportunities that the system calculates that the user has a high matching score with. The user can then contact the SME and request a match which, if accepted, they can both collaborate.

3.5 INTERFACE SKILL END USER (ISEU)

The interface Skill End User (ISEU) is the GUI that enables companies to:

- Set up/amend profile
- Add job opportunities
- View current matches
- Select users and invite them to collaborate.

The ISEU will allow the user to input a company profile which will be stored in the KB. It will also allow the user to input a job opportunity and include a list of required skills, which will then be matched by the SMS to give a list of matching users.

3.6 SELF-REPORTING COLLECTION (SRC)

The main goal of the Self-Reporting Collection (SRC) is to manage the different questions that will be asked to the End-user in order to analyze if it exists any difficulty to the Senior Expert User.

The general architecture designed for this module can be seen in the next figure. We proceed to the description.

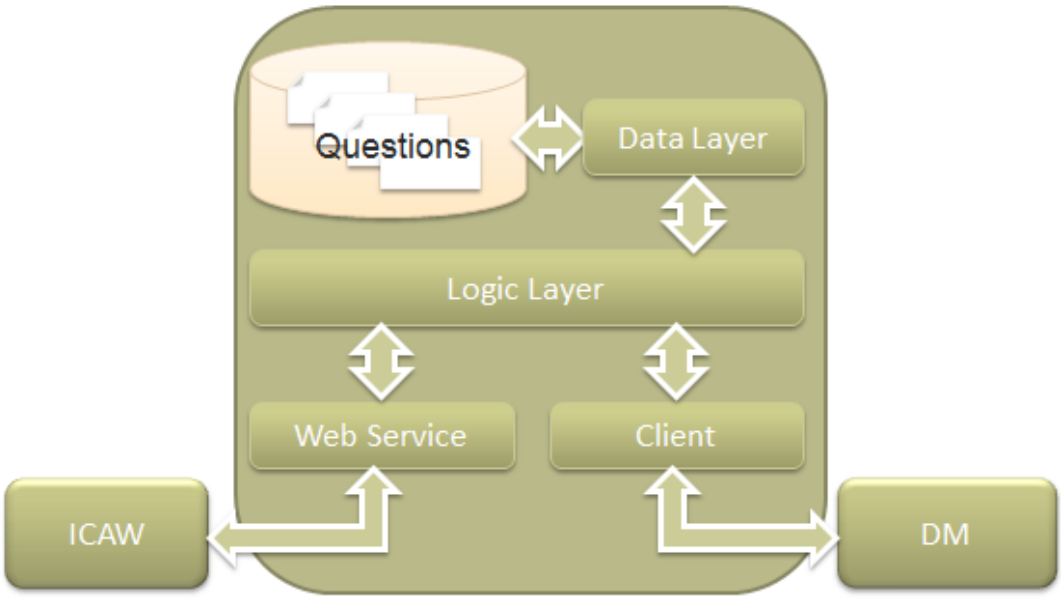


Figure 3: Self-Reporting Collection structure

As we can observe the SRC module communicates with the ICAW, which generates the necessaries interfaces for the SRC and with the Data Manager (DM) to output detected problems with the user.

Two modules are in charge of managing communications, WebServices and Client module.

- **Web Service**: This module provides ICAW with services relative to questions supply that will be asked to the user besides it will feature a service for gathering the responses from users.
- **Client**: This module is in charge of making requests to another Web service from the Data Manager to send parameters that are used to identify an impairment from senior users though the specific problem is not decided here.

The most important module within the SRC is the *Logic Layer* which has the application logic and it is the part of the system that decides which type of questions should show to the user.

Questions are stored in an intern data base and the module responsible for managing said data base is the *Data Layer*.

In relation to the questions, the SRC does not only throw questions to users, but depending on the specific users and their previous answers it focuses on providing some parameters that could give a hint about the user impairment. SRC do not detect or decide the problem but provide help for the ADM to decide.

Concretely the SRC will work in two stages:

- **Stage 1:** In this phase the SRC generates questions in such a way that the system will give values to some general parameters that will be used to grade different problems Elders-up! System may detect, such as visual or hearing impairments.
- **Stage 2:** In this second stage the SRC has stored parameters that indicate the user impairment e.g.: vision problem, and in the second stage will store more parameters or variables that could further help at detecting a more specific problem e.g.: Letter discrimination declines.

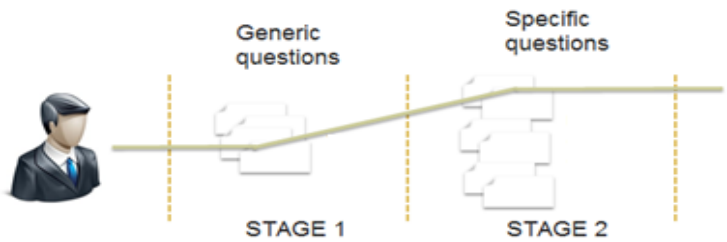


Figure 4: SRC Stages

One great contribution of the SRC is that thanks to the questions asked to the user, like for example the ones shown next; it is able to assess and determine a motivational level of the senior user.

E.g.:

- Rate app (1-5 stars)
- Does the app cause tiredness to you?

- How many hours can you keep using the app?

Once the System has graded the different parameters, SRC will use the client module to transmit in real time that a user has been found with a very likely specific issue based on the parameters.

3.7 SRC FORMS (SRC GUI)

SRC Forms is the section of the ICAW that handles the graphics part or user interfaces from the SRC. In these interfaces is visualised the different forms composed in the different SRC stages. These interfaces are created in a dynamic and customized way for each user. The main characteristics for the interfaces of the several forms are determined aiming at the user convenience, the main features are:

- Brief questions with no dense elaboration.
- Questions customized to specific user peculiarities.
- Forms might contain pictures or sounds.
- Forms contains two choices responses (Yes or No).
- Forms will not be long but sometimes to determine an impairment it might be required to answer two forms.
-

3.8 SENSOR DATA COLLECTION (SDC)

The main role of the module Sensor Data Collection is to collect the information coming from the different sensors and make the pre-analysis of the data obtained. This module is based on three main blocks.

Figure 5: Elders-Up! SDC System

These blocks are: Device, Server and Database. First block SDC Device is referred to the sensors inside of the user's device (tablet or computer), the second block is SDC

Server that communicates with those sensors and the third block is the internal database, where all the information coming from the sensors is saved.

There are two different versions for the SDC Device. One for Android and another one for Desktop PC in Java programming environment. The way sensors communicate is different depending on the device type. To associate data from sensors with the exact user, this component needs a login from the user.

When this component is installed on an Android operating system it can have the information from the following sensors:

- Light sensor
- Accelerometer
- Touch screen
- Microphone
- Proximity sensor
- Camera

On the other hand when the component is installed on Desktop PC's, these are the sensors that may be used:

- Keyboard
- Mouse
- Camera
- Microphone

In both cases the camera is the main sensor, but regarding the privacy of the user in none of the cases the information obtained from the sensors will be saved but only analysed like for example the luminosity of the picture.

Using the web service the information obtained from the devices is sent to the server and despite the different types of the data, the SDC Server will do a pre-analysis and will obtain the next entities:

Status:

- Pulse. Through the mouse movement (x and y position) it detects different levels of tremors.
- Sight. Through the camera an image is obtained and analysed searching for faces. The face is identified as the nearest obtained square covering the face. The closer the face, the bigger the square of the image is.
- **Key Strokes.** Through the keyboard is obtained a number of buttons pressed per minute from the user.
- **Mouse Clicks.** Through the mouse the number of clicks per minute is obtained**.**
- **Touches**. The data is obtained from the screen of the device on Android operating system and counts number of clicks per minute.
- **Faces.** Through the analysed image the total number of faces is processed as a way to know if the user is alone or not.

Environment:

Besides the status information from the sensors, there are two parameters that are referred to the user's environment:

- **Noise.** Microphone detects the noise in the user's environment and calculates and stores it in the internal database in dB (decibels).
- **Luminosity.** Luminosity can be detected with two different sensors. On Android devices with the existing luminosity sensor and on Desktop PCs with the analysis of pictures taken from the camera**.**

User:

The only thing that will be saved in the database regarding the user is the identifier (id_user). This value is the input for the Data Manager and it will be used to identify and know how to connect the specific user with the platform. In this way one historical overview of the users will be achieved.

3.9 DATA MANAGER (DM)

Data Manager is the module where data coming from SRC and SDC converge. Data Manager communicates via the KB with the ADM so this module can adapt the User Interface. Following we can see a figure showing the general architecture of the Data Manager.

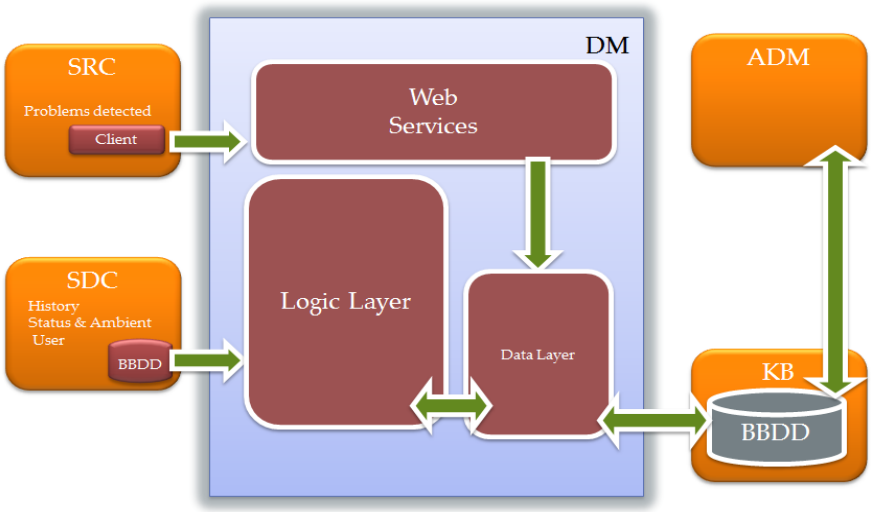


Figure 6: Elders-Up! DM System

The Data Manager joins data and normalize all parameters entering the SRC and SDC.

With help from the data gathered by the SDC, it can be obtained a historical of habits and situations from the user, like for example if he usually is accompanied or always suffers from high level of ambient noise. Moreover, data from the SDC complement perfectly with data recollected directly through the SRC, all this data together will be analysed and treated in the logic layer of the DM to keep coherence of all the data from the user state and environment.

Through the Data Layer, the system manages the communication with the KB, overseeing information transference between the different modules of the Elders-up¡ architecture.

3.10 ADAPTATION DECISION MAKER (ADM)

The objective of this module is to provide the underlining decision making functionality for adapting the senior workspace to his/her profile (see Figure 3).

Functional requirements to be implemented into the first prototype which should be available in M14:

- Decision making algorithm design and implementation;
- Proof of concept of adaptation features for a test case GUI.

Functional requirements to be implemented into the second prototype which should be available in M27:

- Algorithm improvement and adaptation features prioritization;
- Integration with Interface Collaborative Adaptive Workspace (ICAW) templates.

The inputs of this module are:

- The request for workspace adaptation for a specific senior;
- ICAW Templates;
- Senior ICAW interaction data.

The outputs of this module are:

- Adapted ICAW templates to be loaded.

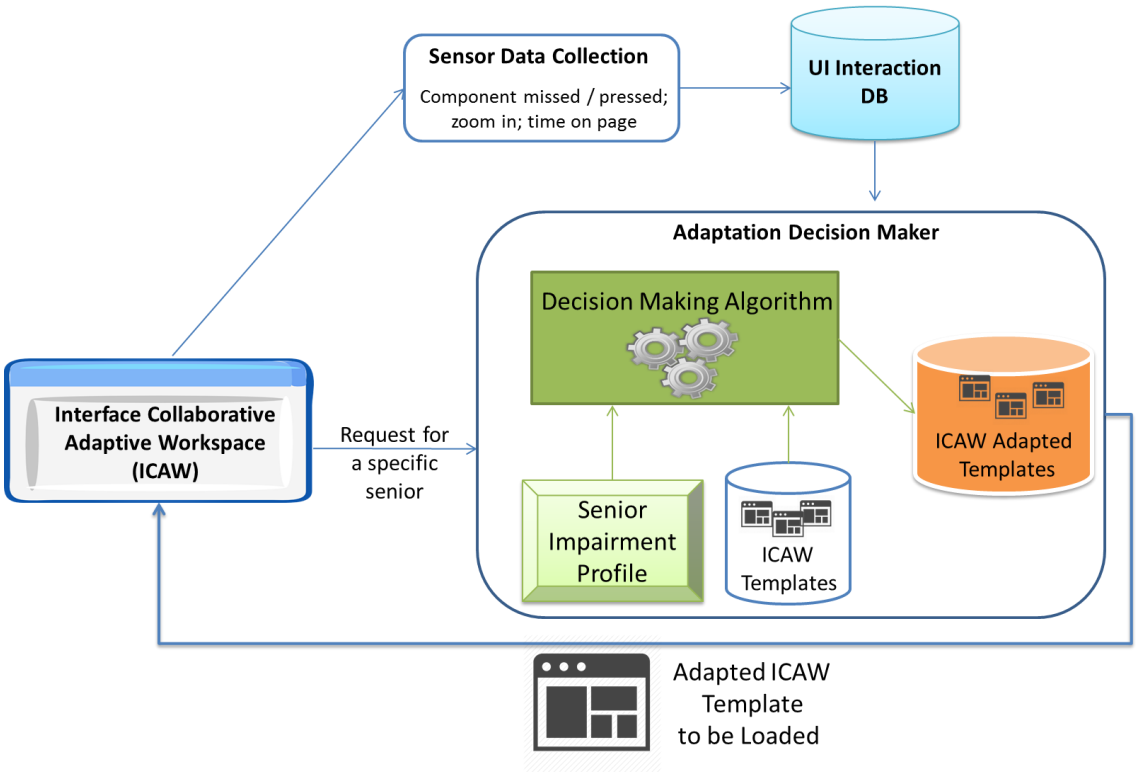


Figure 3: The Adaptation Decision Maker

3.11 SKILL MACHING SERVICE (SMS)

The objective of this module is to provide the underlining functionality for matching the senior skills (out of their job requests) with the companies job offers (see Figure 4).

Functional requirements to be implemented into the first prototype which should be available in M14:

- Match the skills of seniors with those required by the companies (1 to 1 Skill Matching).
- Implement a metric for calculating the matching degree.

Functional requirements to be implemented into the second prototype which should be available in M27:

- Discover skills out of a text description provided by seniors.

- Construct multidisciplinary workforce of seniors based on skills to perform a job for companies.

The inputs of this module are:

- The Job Request which is acquired through the Interface Skill Senior and stored in the Skills Knowledge Base;
- The Skills Taxonomy which is stored in the Skills Knowledge Base;
- The Job Offer which is acquired through the Interface Skill End User (Company) and stored in the Skills Knowledge Base.

The output of this module is the Skill Matching Results which will be stored in the Skill Knowledge Base to be displayed by the appropriated GUI modules.

q

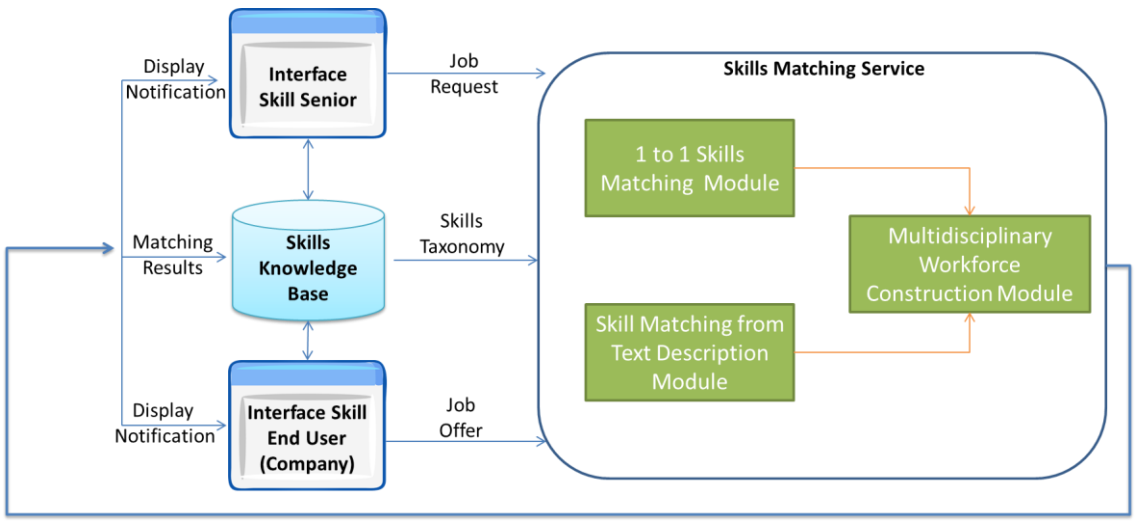


Figure 4: The Skill Matching Service architecture, inputs and outputs

3.12 SKILL RECOGNITION (SR)

The skill recognition module (SR) is the interface which allows the SME to input the opportunities into the KB along with the skills required.

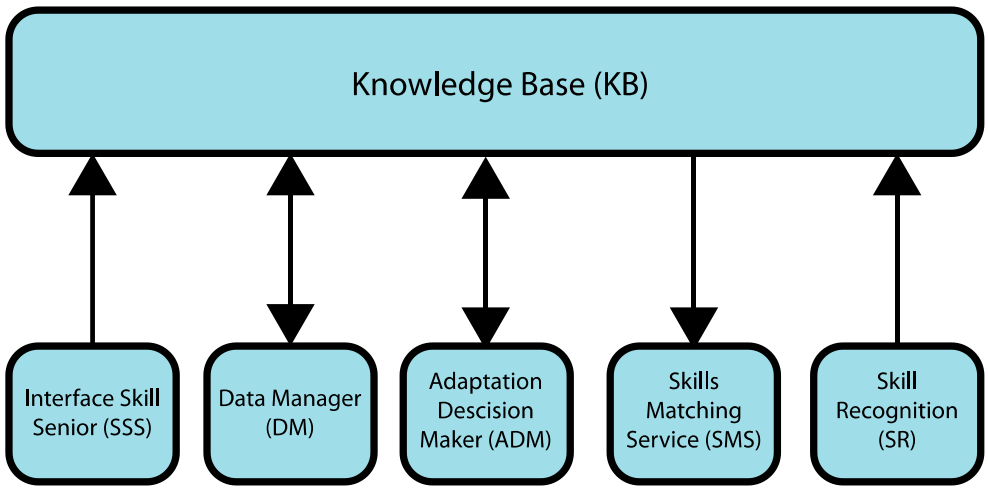
The SME enters the required skills via the taxonomy system similar to that which has been developed for elderly users to enter their skills. The tree system means that

branches as well as leaf nodes can be selected. This means that the SME can be exact in their skill requirements or more general. The branch would then include a subset of skills which are then ranked and matched within the SMS accordingly. (ie exact skill matches are ranked higher than subset skill matches).

This job opportunity, once entered is then stored in the KB and is also processed in the SMS to feedback matches to the SME.

3.13 KNOWLEDGE BASE (KB)

The knowledge base will be the database that stores all of the user profile information, the taxonomy of skills, job opportunities and sensor data collected by the SDC. It will consist of four primary data tables, storing data specific to users, along with several ancillary tables storing data used by the application itself (e.g. the skills taxonomy and list of languages) and some pivot tables that enable the use of many-to-many relationships.



This diagram shows the principal modules the interact with the KB. Namely

- Interface Skill Senior (ISS)
- Data Manager (DM)

- Adaptation Decision Maker (ADM)
- Skills Matching Service (SMS)
- Skills Recognition (SR)

Data flows from other modules into these and then in and out of the KB.

The ICAW is stored in a separate database on a separate server. Whenever a user's profile is updated, the ISS makes a request to an API endpoint on the ICAW server, and the ICAW will sync the database (by directly connecting and grabbing whatever it wants). Similarly, when a user is first created in the ICAW, it makes a request to an API endpoint on the ISS server, which will create an analogous user and return the user's id for the ICAW to make future requests.

3.14 MAILING SYSTEM (MS)

The mailing system takes care of the communication of the system to the users, experts and company users. It is meant to pull the user back into the Elders-UP! system. Sometimes to do things that are necessary for the system: e.g. you are invited for the following group space or appointments. The other one is to notify what is happening in the group space: e.g. these requests have been performed and these messages have been send.

Technically the mailing system works in two ways the invitations and or notification are done upon user input. So the mailing system exposes a few functions that can be triggered by other modules. Informing about the status of the system is triggered from within the Mailing System. It sends these information when something important has happened in the last period or weekly.

4 SYSTEM INTEGRATION

In this section, a detailed view of how the integration of the whole platform is being carried out is provided. To do so, the following sub-sections will delve into the selected system architecture, the designed application structure, the interconnection between the different modules and the roadmap to be prepared for the last months of the first prototype development. The first prototype development will be focused in the functionality of the modules and in the basic integration of its components.

4.1 System Architecture

The EldersUp! Software modules will be deployed on a single server (from now on, the EldersUp! Server, EUPS), where all the information will be stored, treated and served. The EUPS will be thus the central point for deploying the platform services.

From the user perspective, there will be two main ways to connect to the EldersUp! Platform: by using the HTML5 web application present in the EUPS or by using a dedicated Android application that is currently being developed.

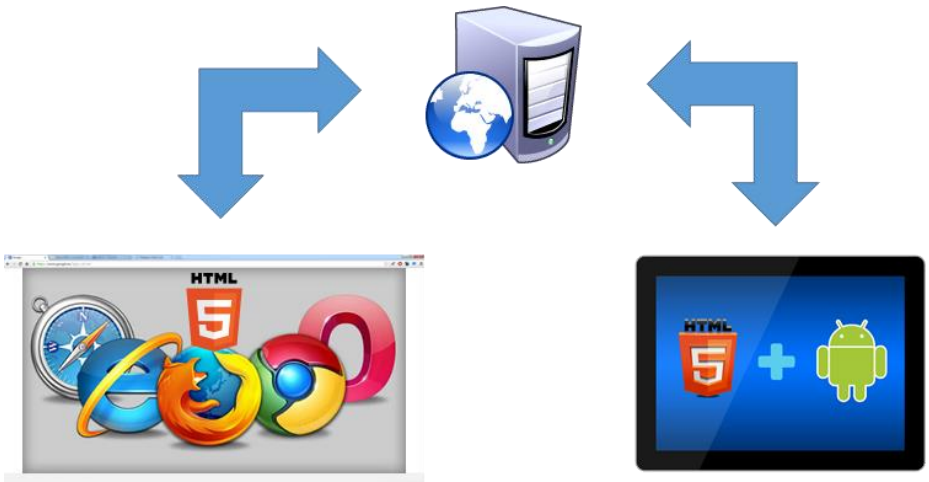


Figure 7: User possibilities to connect to the EldersUp! platform

The application is being developed as an HTML5 dynamic web page. This helps guaranteeing usability and compatibility with a wide range of devices (computers, mobile phones/tablets, etc.). In addition, a hybrid application is being developed for the Android platform, mixing native code with the HTML5 application, in order to fulfill a proper sensor integration with tablet/mobile devices. During the development, special

care is being taken when coding in order to allow further integrations with other mobile platforms (i.e. Apple iOS) in a quick way.

To host such services, the following detailed technical infrastructure is being used:

Figure 8: EldersUp! Server

It includes an Apache Server with some modules required by the different services being implemented, a MySQL Database Server and some extra tools for handling Secure Access (SSL certs) and application server (i.e Tomcat for Java based server application). It also has PHP 5.5.9+ and Composer (PHP dependency manager) installed in the server as well as several JAVA libraries required by some of the implemented services. In addition, a GIT repository is deployed to allow users to work with the development server.

In order to avoid problems during the development, two sets of databases and HTML folders have been deployed so one can be used for the development test platform and the other one as the production one.

4.2 Application Structure

In the following figure (already presented in section 2 but included again here for sake of the clarity) a scheme of the Elders-Up! Application architecture is presented.

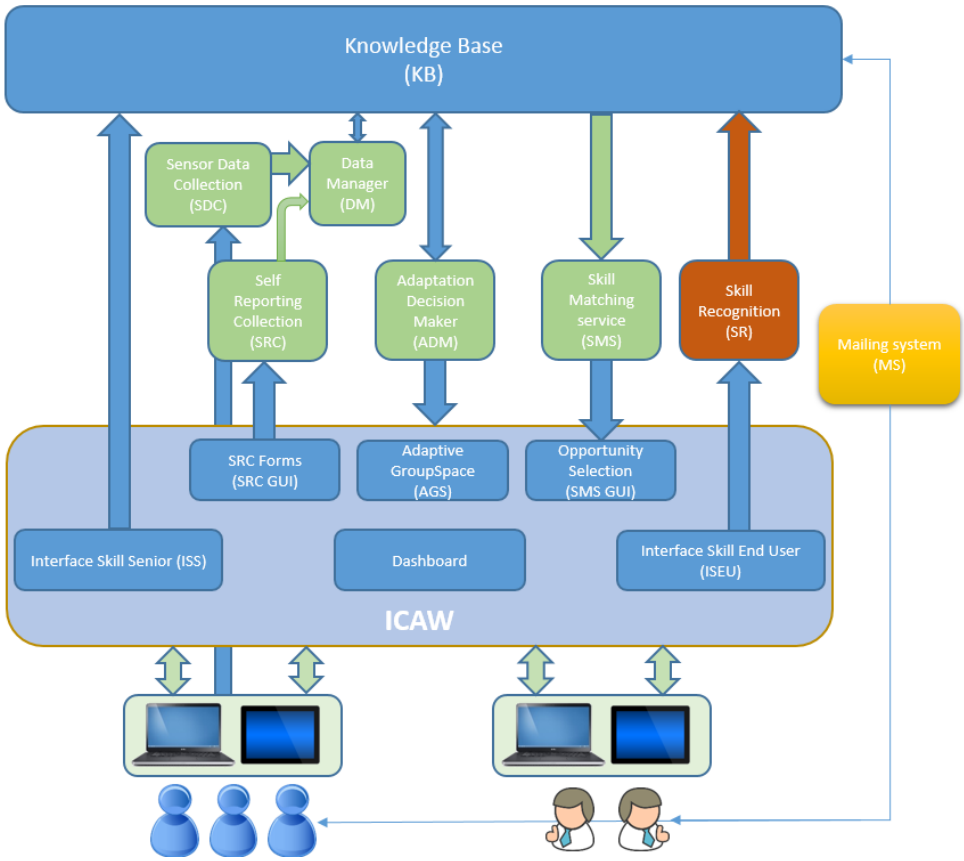


Figure 9: E_UP Aplication structure

The list of the different modules is presented in the previous section (**Error! eference source not found.**) as well as the different inputs and outputs. In the next paragraphs some details regarding how these modules are being integrated is provided.

4.3 Interconnection between modules

The main integration point for the information provided by the different modules will be the Knowledge base, which is implemented as a relational database in MySQL. Each one of the different service developers is preparing the data model for the information they need to store in the database. These data models will be implemented in the database in the next few weeks.

The modules showed in green in the above figure represent the services that will be running into the application server. Their connection to the Knowledge base (and therefore to the main database) will be carried out by using a direct connection with the APIs available in the languages each service is using to implement it. An exception to this approach are the SDC and the SRC modules. These two modules will use the DM as a gateway to store the information and will supply such information using the web services provided by the DM.

Before start sending information to the server, the SDC needs to check the user through a log in system. For the first prototype, the way to perform such check will be different in the general platform (PC and Android devices) than when using the hybrid Android application. The Android app (or the Java application) will ask the user through a dialog, its user and password and will store it encrypted in the device. When the application is loaded, the Android application will send an html request to the HTML5 application including in a POST the user and password of the user. The HTML5 application will authenticate it and will provide in return an URL that the Android app should load in its WebView. In the case of the JAVA application of the SDC, an API will be created to enable the remote login of the user, this API will receive the username and password and will return a JSON object indicating the fail or success of the call and user information like the user id. For the second prototype this authentication will be modified to a more secure one. A token based authentication system for the additional modules (SRC, Android application, etc.) will be developed.

In addition to these services, the SR (shown in red in the above figure), the ISS and the ISEU will also use the direct connection system to store the information into the database.

The SRC and SMS both have dedicated GUI's (SRC GUI and SMS GUI). For both GUI modules the interchange of information with the database will be thus different than in the previous cases. The information recovered by the SRC GUI will be supplied to the SRC using web services and later stored in the database through the DM. In the case of the SMS GUI, the information will flow from the SMS to it. This will be accomplished by

developing web services in the SMS that will be used by the SMS GUI to recover the information that has to be presented.

The AGS will need to access specific data generated by the ADM. To do that, ADM developers will share with the AGS developers the Data Model used to store such information into the DB. The AGS will then access the DB using its own connector to retrieve such data directly from the database.

All the GUIs being developed will be design using templates provided by the ICAW developers as to maintain a smooth transition between the different modules assuring a satisfactory user experience. These templates will include a dynamic CSS file that will be used to maintain the same style when drawing basic HTML elements.

ISS and ISEU GUI's will interact directly to the DB, storing and retrieving their related information from the DB by directly accessing it using the corresponding APIs. The SR module (shown in red in the figure) is not a service running in the background (hence the different colouring) but specific functions used in by the ISEU to translate the skills required by a given company for a job to the ontology used to store the skills in the database.

The MS will get from the DB the information generated by the Motivational Model (included in the ADM) regarding which information each user should receive at what time. To do that, the ADM developers will share the Data Model used to store such information in the DB with the MS developers. The MS will then connect directly to the DB through the proper connector to retrieve this information, prepare the corresponding emails and then send them to the users.

4.4 Roadmap for the development of the first prototype

In order to develop the first prototype within the project expected timetable the following work plan has been sketched as for the final months of its development:

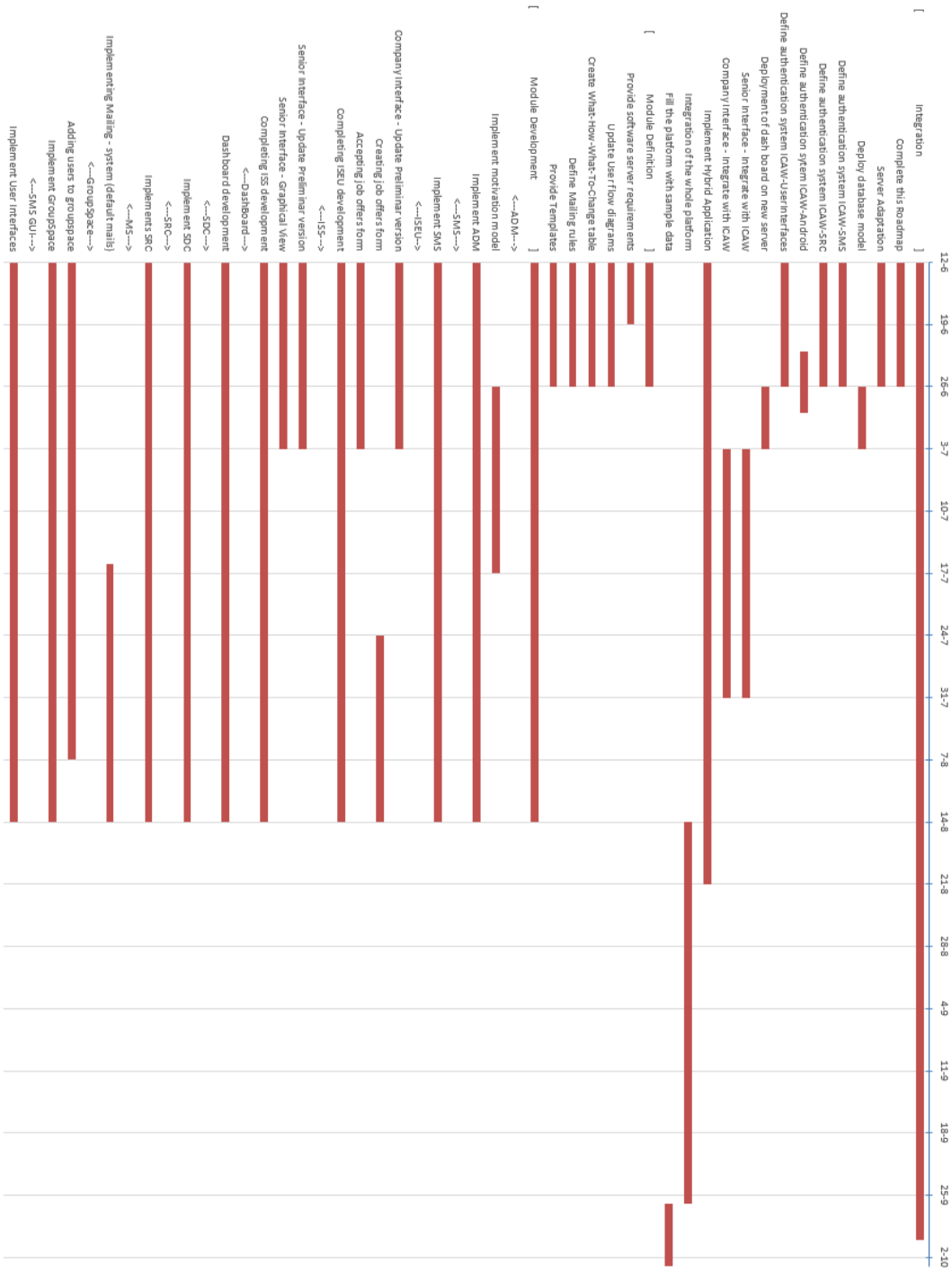


Figure 10: Roadmap for the first prototype

As seen in the above figure, 16th of August is set as the date where all the partners should deliver the first production version of its modules, so it can be integrated in the next weeks. The development of such modules will continue but the improvements carried out will not be included in the 1st prototype unless the integration is finished earlier.

5 Functional View

This chapter lists system functionalities, and provides priorities for the implementation. The functionalities are achieved from the end user perspective.

5.1 Function Specification

The functionality is grouped in main categories, designated by letters, with a numbered list of functions in each category, so that we can refer to functions in this form: A.1, D.4, etc.

A. Manage user account

Primary user needs to have an account in the system. This means that previously the profile needs to be created.

A.1. User Registration

An account is created through the Elders-Up! system with an email address and password for authentication. All end users can enter in their contact details:

A.1.1 First Step: Add personal information.

Users fill in the personal information with the simple form. This is the first step of the registration. Besides the first and last name, user needs to enter the address, e-mail, and contact phone.

A.1.2 Second Step: Add Employment and Skills.

The second step allows the older adults to add their employments and skills using a structure of the simple selection (tree based) with all the possibilities which allow the selection in less than three clicks to make it easier for older adult user.

A.1.3 Third Step: Tell us about yourself.

The third step allows the user to put the other information that cannot be selected in the previous step with the tree containing standard skills. This free text field will be

processed from the user input so the important information for the future matching can be recognized.

A.2. Update your profile

End users can modify any field of the user profile. User can update for example the address or add a new skill and save it again.

A.3. Sign In

The first action to enter in the platform is to login, so the users can use the system.

A.4. Sign out

If the user wants to leave the Elders-Up! system he/she can sign out. When the user goes out of the platform the previous temporary private information of the user is deleted, so he/she needs to sign in if wants to use the platform again.

B. Primary Actions

Primary actions are those which are not related to any specific job, and can be accessed from the main menu.

After the login the first window for the user is a Dashboard where end users (older adults and companies) can choose between the different primary actions in the platform.

B.1 Show Job opportunities

Users can see the list of suggested job opportunities based on a personalized match performed by the system in which they can collaborate. Each of the opportunities has the following characteristics:

- Percentage of the match between the older adult skills and job opportunity
- Show company's information
- Skills required for the job

- The responsible of the opportunity and his/her profile
- Link for chatting with the responsible of job opportunity
- Link to accept the opportunity
- Description and another features of the job opportunity

B.2 Create a workspace (A new opportunity of job)

Companies can add a new workspace with needed skills to collaborate for the help in their company, generating the job opportunity and making the workspace for this collaboration.

B.2.1 View job opportunities

Companies can see the different job opportunities generated and edit or delete the existing ones.

B.2.2 Edit job opportunities

Companies looking at job opportunities can press the edit button and a form appears where company can change the details and save it again.

B.2.3 Delete job opportunities

Companies can delete the job opportunity, removing the job opportunity from the system will update the current list removing the opportunity deleted so it is not shown any more to the older adult user.***B.3. Enter to workspace***

In the main menu the user has the link with the different job opportunities in which he is involved. In each of the links user can access to the different environment with the job opportunities.

If the user is an older adult, he will have an access to the list of applied jobs, in the case of a company the user will be able to see the users applied for the job opportunity.

B.4 Search for companies or experts

The users can use the search engine to look for companies or experts (older adult). The search option is available on main page of the Elders-Up! system. If the user is older adult it will look for companies, on the other side, for companies it finds an expert.

B.5. Use Coaching module

The coaching module is located in the top of the dashboard. This module shows relevant next actions for the older adult. For example, if the user profile has not been completed yet, the coaching module will suggest the older adult to complete his profile.

B.6. Use Request module

The older adult experts can receive collaboration requests from companies requiring their skills. This module is in charge of the communication between the end users (older adults and companies). This module is consists of:

B.6.1 Add requests

Create new request and fill in the next values:

- Deadline of the request
- The type of the request (a question, an office task or coaching)
- Assign the request to a team member

B.6.2 Respond to a request

User can respond to the requests.

B.6.3 Close a request

When the request is responded and the matching is done it can be closed.

B.7. Success case module

In this module are shown the successful matches. All the users can enter and see these matches.

C. Workspaces

Workspace is the environment inside of the platform where the users are interacting between themselves regarding the specific job opportunity.

Inside of each workspace the user can see the different objects and attributes:

- Members group
- See the To-do-Tasks and see status of task process
- Responsible person from the company and the company characteristics
- Documentation repository
- Working agreements within the workspace

C.1 Manage workspaces

A company administrator must be able to manage the settings for a workspace, e.g. manage who has access to the workspace, and what information is available to them.

C.2. Send messages

Send pictures and text with other members of the GroupSpace in the platform.

C.3. Send email

The platform can show a link to the external e-mail client with the recipient to all members of the GroupSpace. This way users within the GroupSpace may email the other members in an easy manner.

C.4. Start voice communication

All the users can start the voice conversation with other members of the GroupSpace.

C.5. Start video Communication

All users can start the video conference with members of the GroupSpace. This function is not fully decided and it won't be available on the first prototype.

C.6. Make and track appointments

All users can create and follow up created appointments. Actions available within this module are:

C.6.1. Create appointments (one-time and recurring events)

To create an appointment the user needs to fill in the next fields:

- Set date and time for the appointment
- Select a team member
- Join an appointment

C.6.2 Accept invitations

All older adult experts can accept the invitations sent by companies.

C.7. Show a calendar common in the workspace

All appointments appear in the calendar that is accessible for all the users that are members of the specific group (with the same job opportunity) of the workspace.

C.8 Use file sharing module

All users can share files inside of this module available for each GroupSpace. Allowed operations are:

C.8.1 Add or remove Files

Users can add new or remove existing files in the shared repository.

C.8.2. add or remove folders

Users can add or remove existing folders in the shared repository.

C.8.3. Edit shared files

Users can edit existing files on the platform repository.

C.9. Invite new members (only for companies)

Users that represent the company can invite new members to the GroupSpace. Invited users are older adults who can collaborate with a certain company.

D. Others Actions

With the term other actions we assume those which are not mentioned above, like for example functionalities executed automatically by the system on the background context.

D.1. Update notifications through e-mail

Users are informed through e-mail about the progress of the job opportunity where he/she collaborates. This service will keep the users up to date.

D.2. Adapt the user interface manually

The user can change his/her interface appearance at any time. Those changes can be the font, change the way the workspace looks (change resolution for example) etc.

D.3. Adapt user interface automatically

Through the data obtained from the user and the environment which involves the user different problems can be detected (problems to see the screen, tremor etc.). In this case the graphic interface adjusts so the problems of the users can be mitigated.

D.4. Skills matching automatically

Elders-Up! system matches automatically the needed skills from the job opportunity published in the platform with the ones that the older adult has in his/her profile. Job offers and skills provided by the user are saved in the Skills Knowledge base (SKB) from where Skills matching Service (SMS) module uses this information to find the most accurate match.

6 Use Cases

Based on the functionalities specified in the previous chapters, the most important use cases have been identified for the older adults and company users.

The following use cases are described using a common table schema. The main section is the Main Flow, where the use case is broken down into an ordered list of steps.

Some use cases are restricted to certain users. The following table describes each use case and its possible actors.

	Use Case	Type of user
1.	**Create and configure an account**	All users
2.	**Sign In**	All users
3.	**Sign Out**	All users
4.	**Update your profile**	All users
5.	**Accept the job opportunity**	Older adult user
6.	**Enter in a workspace**	All users
7.	**Search**	Older adult user
8.	**Use Coaching**	Older adult user
9.	**Add requests**	Company user
10.	**Response to a request**	Older adult user
11.	**Close a request**	Company user
12.	**See success cases**	Company user
13.	**Create Job opportunities**	Company user
14.	**Send messages or images**	All users
15.	**Send emails**	All users

16. Start Voice or Video communication	All users
17. Make appointments & track appointments	All users
18. Accept/Reject invitations	All users
19. See the shared calendar	All users
20. Add or remove files	All users
21. Add or remove folders	All users
22. Edit shared files	All users
23. Invite new members	Company users
24. Adapt the user interface manually	All users
25. Adapt the user interface automatically	Older adult users (indirectly)

Table 1. Use cases and types of actors

6.1 **Use Case:** Create and configure an account

Use Case Number	1
Use Case Name	**Create and configure account**
Actors	End Users and Elders-Up! system.
Summary	Covers all the steps of creating and configuring an Elders-Up! account through the Elders-Up! app.
Trigger / intent	User enters the Elders-Up! system front-end.
Pre-conditions	• The primary user is not yet registered in the system, but wishes to be a user of the Elder-Up! system. • The primary user has an email address not registered in the system.
Flow of events: (Main Flow)	1. **The users enters a username and password.** 2. Elders-Up! system checks if there is no user id that uses the same username (or email) 3. Enter *personal information.* 4. Enter *Employment and Skills*. 5. Enter *Tell us about yourself*.

	6. Elders-Up! system adds this new user to the Elders-Up! database. 7. The Elders-Up! system shows to the user that user's account has been created successfully.
Alternative flows	1. User is requested to choose another password and name. 2. Password and name don't match the correct format 3. User leaves the Elders-Up! system.
Exceptional flows	Operation fails: Account creation fails with error message.
Displayed information	Form to enter username, password, personal data, skill, CV file.
Post-conditions	The primary user has a configured account in the system, and may start using Elders-Up! collaboration platform.
Relation to other use cases	None of the rest of use cases can be performed unless this has been done successfully.

Table 2. Use Case of "Create and configuration account"

6.2 **Use Case:** Sign In

Use Case Number	2
Use Case Name	**Sign In**
Actors	End Users and Elders-Up! system.
Summary	A registered user wants to use Elders-Up! system and the first step is the sign in.
Trigger / intent	When the user completes the sign in form and presses *Sign In* button.
Pre-conditions	• User must be registered. • User shouldn't be signed in.
Flow of events: (Main Flow)	1. User completes the Sign In form 2. Presses the *Sing In* button 3. The Elders-Up! system checks introduced user and password. 4. If successful, the user can see the main menu of Elders-Up!
Alternative flows	1. User completes the login form 2. Pushes the Login Button 3. The Elders-Up! system checks the user and password of the user. 4. If not successful, the user can see over the login screen a notification with "The credentials are not valid"
Exceptional flows	
Displayed information	The user can watch main screen.

Post-conditions	The user can use the Elders-Up! system.
Relation to other use cases	This use case cannot be performed unless the use case "**Create and configuration account**" has been done successfully.

Table 3. Use Case of "User Login"

6.3 **Use Case:** Sign out

Use Case Number	3
Use Case Name	**Sign out**
Actors	End Users and Elders-Up! system.
Summary	A logged user wants to logout Elders-Up! system.
Trigger / intent	The user clicks or taps the *Sign Out* button
Pre-conditions	• The user must be logged
Flow of events: (Main Flow)	1. The user clicks or taps the *Sign out* button 2. Elders-Up! system removes the user session 3. The Elders-Up! system opens the Sign in screen and the user is signed out.
Alternative flows	
Exceptional flows	
Displayed information	User can see the Sign in screen
Post-conditions	User has to sign in if he wants to use the Elders-Up! system.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 4. Use Case of "Sign out"

6.4 **Use Case:** Update your profile

Use Case Number	4
Use Case Name	**Update your profile**
Actors	End Users and Elders-Up! system.
Summary	In his profile, the user can change information about him/her.
Trigger / intent	User activates *Profile* button.
Pre-conditions	• The user must be signed in.
Flow of events: (Main Flow)	1. The user clicks or taps *Profile* button. 2. A screen with his personal information is shown 3. The user adds, removes or changes outdated fields.

	4. Clicks save button. 5. His profile is updated.
Alternative flows	1. The user clicks or taps the *Profile* button. 2. A screen with his personal information is shown 3. The user adds, removes or changes outdated fields. 4. Clicks cancel button or closes the window. 5. His profile is not updated.
Exceptional flows	
Displayed information	User can see his profile information.
Post-conditions	His updated profile can be seen by other users.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 5. Use Case of "Update your profile"

6.5 **Use Case:** Accept the job opportunity

Use Case Number	5
Use Case Name	**Accept the job opportunity**
Actors	End Users and Elders-Up! system.
Summary	User older adult accepts a new job opportunity in a Current Opportunity Screen.
Trigger / intent	User older adult pushes *Confirm interest* button.
Pre-conditions	• The user must be signed in.
Flow of events: (Main Flow)	1. User older adult clicks or taps *View opportunity* from the Current Opportunities List. 2. User older adult reads current opportunity information 3. If the user agrees, he clicks *Confirm interest*.
Alternative flows	1. User older adult clicks or taps view opportunity from the Current Opportunities List. 2. He reads current opportunity information 3. User older adult doesn't confirm interest in the job.
Exceptional flows	
Displayed information	A description of the current opportunity is shown for the user older adult, the company that offers it and information about the job.
Post-conditions	User older adult can see this new working agreement in his workspace list, and he/she is able to enter it.

Relation to other use cases	This case cannot be performed unless the user older adult is signed in. A company user has executed the use case *Create a new job opportunity* before.

Table 6. Use Case of "Accept the job opportunity"

6.6 **Use Case:** Enter in a workspace

Use Case Number	6
Use Case Name	**Enter in a workspace**
Actors	End Users, Elders-Up! system.
Summary	Signed in user enters to a workspace.
Trigger / intent	User is in the Dashboard. Starts this use case clicking link with other links to "your workspace"
Pre-conditions	• User must be signed in. • User is in the Elders-Up! *dashboard*
Flow of events: (Main Flow)	1. Users sees the groupspaces the user is participating in. 2. User selects clicks or tabs on a groupspace. 3. The system shows the groupspace.
Alternative flows	
Exceptional flows	1. Users sees the groupspaces the user is participating in. 2. The users does not click on a groupspace. 3. The elders-up! System does nothing
Displayed information	Shows a workspace with its visible content.
Post-conditions	User is in the selected workspace.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 7. Use Case of "Enter in a workspace"

6.7 **Use Case:** Search

Use Case Number	7
Use Case Name	**Search**
Actors	End Users and Elders-Up! system
Summary	An older adult wants to find a company that needs help or a company user wants to find an older adult to collaborate.

Trigger / intent	User is in the Dashboard. Presses *Search* button to find an older adult or a company.
Pre-conditions	• User must be signed in. • User is in the Elders-Up! *dashboard*
Flow of events: (Main Flow)	1. User presses *Search* button from the Dashboard. 2. The system opens a window containing a text field to write his search. 3. User writes in the text field a valid older adult or Company name. 4. The system shows a list with results. 5. User selects a person or company to have the possibility of interacting with him.
Alternative flows	1. User presses *Search* button from the Dashboard. 2. The system opens a window containing a text field to write his search. 3. User writes in the text field a valid older adult user or company user name. 4. The system can't find a result. 5. User returns to *Main menu*.
Exceptional flows	1. User presses *Search* button from the Dashboard. 2. The system opens a window containing a text field to write his search. 3. User leaves search and returns to menu.
Displayed information	A list of users and companies.
Post-conditions	Older adult has found a company or a company has found an older adult.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 8. Use Case of "Search"

6.8 **Use Case:** Use Coaching

Use Case Number	8
Use Case Name	**Use Coaching**
Actors	End Users, Elders-Up! system.
Summary	A carrousel, located in the top of the dashboard, shows relevant next actions for the older adult.
Trigger / intent	When the user begins use with the coaching module

Pre-conditions	• User must be signed in. • User is in the coaching module. • User is an older adult.
Flow of events: (Main Flow)	1. User starts Elders-Up! system. 2. Coaching module is always active in the top of the dashboard. It shows relevant actions for the older adult and actions that should be completed. 3. User interacts with coaching module completing some actions.
Alternative flows	1. User starts Elders-Up! system. 2. Coaching module is always active in the top of the dashboard. It shows relevant actions for the older adult and actions that should be completed. 3. User doesn't interact with coaching module completing some actions.
Exceptional flows	1. User starts Elders-Up! system. 2. Coaching module is always active in the top of the dashboard. It doesn't show relevant actions for the older adult.
Displayed information	Helping information and suggestions for the user. This is useful for a better user experience.
Post-conditions	If the user clicks on *Use Coaching* options, new interfaces can be opened: *Update your profile*, *search* some company user and other interfaces.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 9. Use Case of "Use Coaching"

6.9 **Use Case:** Add requests

Use Case Number	9
Use Case Name	**Add request**
Actors	End Users, Elders-Up! system.
Summary	In this use case new requests can be sent to the older adult by a company.
Trigger / intent	Company user pushes *add requests* button in the requests interface.
Pre-conditions	• Sender and receiver of the request must be registered • Both users are in the same groupspace.
Flow of events: (Main Flow)	1. Company user pushes *add a request* in the requests interface. 2. Elders-Up! system opens an interface to create a new request

	3. Company completes request information. 4. Company pushes *send request* button. 5. Elders-Up! system sends a new request to a user older adult.
Alternative flows	1. User pushes *add a request* in the requests interface. 2. Elders-Up! system opens an interface to create a new request. 3. User pushes *cancel* button.
Exceptional flows	
Displayed information	An interface showing information for a new request: -Deadline of request -Type of request -Description -Team member
Post-conditions	A user older adult receives the request. The user can see the request in the request module.
Relation to other use cases	This case cannot be performed unless the user is signed in. User must have accepted a job opportunity.

Table 10. Use Case of "Add requests"

6.10 **Use Case:** Response to a request

Use Case Number	10
Use Case Name	**Response to a request**
Actors	End Users, Elders-Up! system.
Summary	User older adult receives a request sent by another user and accepts or refuses it.
Trigger / intent	User older adult pushes a request in the *Requests* menu and selects *Read request*.
Pre-conditions	• Both users are registered. • A user company has sent a request. • Both users are in the same groupspace
Flow of events: (Main Flow)	1. Expert receives an email with a request. 2. The expert clicks on the request and a web browser opens with the specific request in the right groupspace. 3. The expert accepts the request. 4. The request is shown to be confirmed by the expert..

	5. A confirmation message is sent to the user who created the request.
Alternative flows	1. Expert receives an email with a request. 2. The expert clicks on the request and a web browser opens with the specific request in the right groups space. 3. Older adult clicks deny request. 4. A refusal message is sent to the user who created the request.
Exceptional flows	
Displayed information	An email describing the request to the user. **A pop-up message** describing the request and providing the option to accept, or reject the request among some other option.
Post-conditions	Both users have the request in the accepted requests menu.
Relation to other use cases	A company has sent the request in *Add request* use case.

Table 11. Use Case of "Response to a requests"

6.11 **Use Case:** Close a request

Use Case Number	11
Use Case Name	**Close a request**
Actors	End Users, Elders-Up! system.
Summary	A request can be closed by the user company who created it. This user closes the request from the requests menu.
Trigger / intent	The user who created the request clicks on it.
Pre-conditions	• Company is signed in. • Request has been created. • Request is open.
Flow of events: (Main Flow)	1. Company selects a request in the *request* interface. 2. Clicks close. 3. The system opens a confirmation message. 4. Company confirms he wants to close the request. 5. Elders-Up! system closes the request.
Alternative flows	1. Company selects a request in the *request* interface. 2. Clicks close. 3. The system opens a confirmation message.

	4. Company doesn't confirm to close the request.
Exceptional flows	
Displayed information	A confirmation message asking if the user wants to close the request.
Post-conditions	The request is removed from the requests menu.
Relation to other use cases	Company has created the request in *Add a request* and an older adult has accepted it in *Response to request* use case.

Table 12. Use Case of "Close a request"

6.12 **Use Case:** See success cases

Use Case Number	12
Use Case Name	**See success cases**
Actors	End Users, Elders-Up! system.
Summary	Some companies success cases of Elders-Up! system are described in this use case. Information about users or companies is not shown in this use case.
Trigger / intent	Company pushes the button *Success cases* in the dashboard.
Pre-conditions	• User is signed in.
Flow of events: (Main Flow)	1. Company clicks success cases button 2. Elders-Up! system opens a new screen showing the anonymous success cases.
Alternative flows	
Exceptional flows	
Displayed information	A description of the most important success cases.
Post-conditions	User company has seen Elders-Up! system success cases.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 13. Use Case of "See success cases"

6.13 **Use Case:** Create Job opportunities

Use Case Number	13
Use Case Name	**Create Job opportunities**
Actors	End Users, Elders-Up! system.

Summary	A user company wants to create a new job opportunity as his company needs an expert in a particular skill.
Trigger / intent	Starts when the Company presses *Create a workspace* in the dashboard.
Pre-conditions	• User must be signed in. • The Company requires particular skills.
Flow of events: (Main Flow)	1. Company pushes *Create workspace* in the Main Menu. 2. The system opens a window with an application form to fill it with relevant data. 3. Company fills the form. 4. Company presses accept. 5. The system registers the job opportunity and generates a link in the company's workspace.
Alternative flows	1. Company pushes *Create workspace* in the Main Menu. 2. The system opens a window with an application form to fill it with relevant data. 3. Company fills the form. 4. Company presses cancel. 5. The system notifies the user that the operation was cancelled and the user returns to Main menu.
Exceptional flows	
Displayed information	The system reports the workspace creation to the user.
Post-conditions	Workspace and job opportunity are generated.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 14. Use Case of "Manage workspaces"

6.14 **Use Case:** Send messages or images

Use Case Number	14
Use Case Name	**Send messages or images**
Actors	End Users, Elders-Up! system.
Summary	All users can send messages or images to all other members of the workspace.
Trigger / intent	User clicks *Send* button in *Messages* interface in the groupspace.
Pre-conditions	• User has to be signed in. • Sender and receiver are in the same groupspace.

	• Sender is logged in.
Flow of events: (Main Flow)	1. User opens Messages screen. 2. Writes the message in *type message*. User can also add an image. 3. Clicks send. 4. Elders-Up! system adds the message to the messages in the groupspace.
Alternative flows	1. User opens Messages screen. 2. Writes the message in *type message*. User can also add an image 3. The user doesn't click *Send* 4. The system doesn't send the message
Exceptional flows	
Displayed information	
Post-conditions	The message is added to the message list of the group space.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 15. Use Case of "Send messages or images"

6.15 **Use Case:** Send emails

Use Case Number	15
Use Case Name	**Send emails**
Actors	End Users, Elders-Up! system.
Summary	All users can send emails to other members of the groupspace.
Trigger / intent	User clicks opens the profile of a person in the groupspace.
Pre-conditions	• User has to be signed in. • Sender and receiver are in the same workspace.
Flow of events: (Main Flow)	1. Selects a user to send him the mail. 2. Pushes the send email button. 3. An external mail client opens with a predefined recipient. This email client handles the rest of the interaction with the user.
Alternative flows	1. Selects a user to send him the mail. 2. Pushes the send email button. 3. An external mail client opens with a predefined recipient. 4. The user closes the email client. No message has been send.

Exceptional flows	
Displayed information	
Post-conditions	Receiver has the new message in his email inbox.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 16. Use Case of "Send mails"

6.16 **Use Case:** Start Voice or Video communication

Use Case Number	16
Use Case Name	**Start voice or video communication**
Actors	End Users, Elders-Up! system.
Summary	User starts a new voice or video communication. Every user can start a communication with a member of a shared workspace.
Trigger / intent	Selects a user in the video call interface.
Pre-conditions	• User has to be signed in. • Both users have to be in the same collaboration group. • Contact must be added.
Flow of events: (Main Flow)	1. User opens video call interface. 2. Clicks a user to start a conversation. 3. User selects voice or video communication. 4. Elders-Up! system sends a confirmation message to the other contact. 5. The other contact accepts the connection. 6. The call starts.
Alternative flows	1. User opens video call interface. 2. Clicks a user to start a conversation. 3. User selects voice or video communication. 4. Elders-Up! system sends a confirmation message to the other contact. 5. The other contact refuses the connection.
Exceptional flows	
Displayed information	Video call: each user can watch video images of the other contact. Voice call: each user can watch the profile's image of the other contact.
Post-conditions	All users starting a voice or video communication have the opportunity to finish the communication.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 17. Use Case of "Start Voice or Video communication"

6.17 **Use Case:** Make appointments & track appointments

Use Case Number	17
Use Case Name	**Make & track appointments**
Actors	End Users, Elders-Up! system.
Summary	All users can make or follow appointments.
Trigger / intent	User pushes Appointments button in the workspace.
Pre-conditions	• User has to be signed in. • Both users have to be in the same workspace.
Flow of events: (Main Flow)	1. User clicks *Appointments* button. 2. User opens a form to generate an appointment. 3. User fills the form. 4. User selects other users that can track the appointment. 5. Selects a date. 6. Saves the appointment. 7. The system registers the appointment and shows it in the Elders-Up! system calendar. 8. Other users can accept or refuse an appointment.
Alternative flows	
Exceptional flows	
Displayed information	Elders-Up! system shows in the calendar the registered appointment.
Post-conditions	The appointment was registered.
Relation to other use cases	This case cannot be performed unless the user is signed in.

Table 18. Use Case of "Make appointments & track appointments"

6.18 **Use Case:** Accept/Reject invitations

Use Case Number	18
Use Case Name	**Accept/Reject invitations**
Actors	End Users, Elders-Up! system.
Summary	User accepts or rejects appointment invitations
Trigger / intent	User can select a new appointment in *Agenda.* New appointments also can be shown in *Use coaching.*
Pre-conditions	• User is signed in • Another user has created an appointment inviting him/her.
Flow of events: (Main Flow)	1. User selects an invitation. 2. A confirmation message is shown. 3. User accepts the appointment. 4. The updated appointment is stored in agenda.
Alternative flows	1. User selects an invitation. 2. A confirmation message is shown. 3. User rejects the appointment.

Exceptional flows	
Displayed information	Information about the invitation and two buttons; OK and cancel.
Post-conditions	The system registers the accepted or refused invitation.
Relation to other use cases	An appointment must have been created in the "Make & track appointments" use case.

Table 19. Use Case of "Accept/Reject invitations"

6.19 **Use Case:** See the shared calendar

Use Case Number	19
Use Case Name	**See the shared calendar**
Actors	End Users, Elders-Up! system.
Summary	User sees his appointments in the calendar.
Trigger / intent	User pushes Appointments button.
Pre-conditions	• User has to be logged in.
Flow of events: (Main Flow)	1. User pushes *Appointments.* 2. Elders-Up! system opens Agenda. 3. The system shows the calendar shared in the workspace.
Alternative flows	
Exceptional flows	
Displayed information	A calendar with accepted and invitations of new appointments is displayed.
Post-conditions	User is in a window that allows him to see the shared calendar.
Relation to other use cases	

Table 20. Use Case of "See the common calendar"

6.20 **Use Case:** Add or remove files

Use Case Number	20
Use Case Name	**Add or remove files**
Actors	End Users, Elders-Up! system.
Summary	User adds or removes files in *File Sharing* screen. In the file sharing system a folder for each project has been created. Every type of user can add or remove a file.
Trigger / intent	User pushes add file button to create a new file in a shared project. User selects a file and pushes *remove* button to delete a file from a shared project.
Pre-conditions	• User has to be signed in • To add or open files of a project he has to be part of it.
Flow of events: (Main Flow)	1. User pushes *file sharing.* 2. Elders-Up! system opens *File Sharing* interface. 3. User pushes *add new file*. 4. Elders-Up! system opens a window where the user can search and select the file he wants to upload. 5. User selects the file and clicks upload. 6. Elders-Up! system creates the file in the folder.
Alternative flows	1. User selects a file. 2. Pushes delete button.

	3. Elders-Up! system opens a confirmation message to delete the file. 4. User accepts to delete the file. 5. Elders-Up! system removes the file from the folder.
Exceptional flows	
Displayed information	A folders and files structure.
Post-conditions	Files were added/deleted from the shared folder.
Relation to other use cases	

Table 21. Use Case of "Add or remove files"

6.21 **Use Case:** Add or removes folders

Use Case Number	21
Use Case Name	**Add or remove folders**
Actors	End Users, Elders-Up! system.
Summary	User adds or removes folders in *File Sharing* screen. In the file sharing system a folder for each project has been created. All users can add and remove folders.
Trigger / intent	User pushes add folder button to create a new folder in a shared project. User selects a folder and pushes *remove* button to delete a folder from a shared project.
Pre-conditions	• User has to be signed in • To add or open files of a project he has to be part of it.
Flow of events: (Main Flow)	1. User pushes *file sharing.* 2. Elders-Up! system opens *File Sharing* interface. 3. User pushes *add new folder*. 4. Elders-Up! system opens a window where the user can write a name for the folder. 5. User writes the name and clicks upload. 1. Elders-Up! system creates the new folder.
Alternative flows	1. User selects a folder. 2. Pushes *delete* button. 3. Elders-Up! system opens a confirmation message to delete the folder. 4. User accepts to delete the folder. 1. Elders-Up! system removes the folder.
Exceptional flows	
Displayed information	A folders and files structure.
Post-conditions	Folders were added/deleted from the shared folder.
Relation to other use cases	

Table 22. Use Case of "Add or removes folders"

6.22 **Use Case:** Edit shared files

Use Case Number	22
Use Case Name	**Edit shared files.**
Actors	End Users, Elders-Up! system.

Summary	User edits a file from the *file sharing* interface. Every user can edit a file of a shared workspace.
Trigger / intent	User double-clicks a file (or press the button "Open file" when the file is selected)
Pre-conditions	• User has to be signed in • To edit a file of a project he has to be part of it.
Flow of events: (Main Flow)	1. User double-clicks a file. 2. Elders-Up! system opens the file. 3. User edits the file. 4. User saves the file. 5. Elders-Up! system opens a confirmation message to save the new file. 6. User accepts to save the changes. 7. Elders-Up! system saves the changes in the file.
Alternative flows	1. User double-clicks a file. 2. Elders-Up! system opens the file. 3. User edits the file. 4. User saves the file. 5. Elders-Up! system opens a confirmation message to save the new file. 6. User refuses to save the changes.
Exceptional flows	-
Displayed information	"Last modified by:" and "Modified on:" are updated with new values.
Post-conditions	The file is updated in the system.
Relation to other use cases	

Table 23. Use Case of "Edit shared files"

6.23 **Use Case:** Invite new members

Use Case Number	23
Use Case Name	**Invite new members**
Actors	Company, Elders-Up! system.
Summary	Company can invite new members to his group. Those invited users are older adults who can collaborate with the company.
Trigger / intent	User pushes invite new members' button.
Pre-conditions	• User has to be signed in as a company.
Flow of events: (Main Flow)	1. User pushes *invite new members*. 2. A list with users is shown. 3. Selects a user. 4. Pushes *add new member*. 5. Elders-Up! system sends a confirmation message to the user.
Alternative flows	
Exceptional flows	
Displayed information	Elders-Up! system shows the message: "A new request was sent".
Post-conditions	A new member's request is sent to the selected member.
Relation to other use cases	

Table 24. Use Case of "Invite new members"

6.24 **Use Case:** Adapt the user interface manually

Use Case Number	24
Use Case Name	**Adapt the user interface manually**
Actors	End Users, Elders-Up! system.
Summary	User opens settings and changes the graphical user interface options.
Trigger / intent	User clicks on *Settings*.
Pre-conditions	• User has signed in. • User must be an older adult user.
Flow of events: (Main Flow)	1. User clicks *Settings*. 2. User modifies view settings 3. Elders-Up! system shows the graphical user interface with the new settings. 4. User confirms changes.
Alternative flows	
Exceptional flows	
Displayed information	A Settings menu interface is displayed.
Post-conditions	The system changes the appearance of the graphical user interfaces.
Relation to other use case	

Table 25. Use Case of "Adapt the user interface manually"

6.25 **Use Case:** Adapt the user interface automatically

Use Case Number	25
Use Case Name	**Adapt the user interface automatically**
Actors	Elders-Up! system, older adult
Summary	In this use case, Elders-Up! system collects data from the user's ambient and situation adapting user's UI to the detected conditions.
Trigger / intent	
Pre-conditions	• User has installed Elders-Up! SDC. in his device. • SDC has registered user's valid ID. • User is logged in Elders-Up! application.
Flow of events: (Main Flow)	1. Elders-Up! system collects user data 2. Elders-Up! system checks collected data to detect potential problems 3. Elders-Up! system Detects a problem and propose a change of the graphical user interface 4. The older adult accepts the change 5. Interfaces are automatically updated by the system.
Alternative flows	1. Elders-Up! system collects user data 2. User interface is not updated.
Exceptional flows	
Displayed information	
Post-conditions	User interface has been automatically adapted for the user.
Relation to other use case	

Table 26. Use Case of "Adapt the user interface automatically"

7 Function Priority

The following table lists all the functions specified, along with priorities for the implementation of these functions in the Elders-Up! system. Priority values: System Components

1: Priority for the first prototype.

2: Priority for the final prototype.

3: Low priority –by the end of the project, but not essential.

Code	Function	Priority	Comments
A.1	User registration	1	
A.1.1	Add personal information	1	
A.1.2	Add Employment and Skills	1	
A.1.3	Tell us about yourself	1	
A.2	Update your profile	1	
A.3	Sign In	1	
A.4	Sign Out	1	
B.1	Show the job opportunities	1	
B.2	Create a workspace	1	
B.2.1	View Job opportunities	1	
B.2.2	Edit Job opportunities	2	
B.2.3	Delete Job opportunities	2	
B.3	Enter to workspace	1	
B.4	Search for user companies or older adult	1	
B.5	Use coaching	2	
B.6	Use Request module	1	
B.6.1	Add request	1	

B.6.2	Respond to a request	1
B.6.3	Close a request	1
B.7	Use success case module	3
C.1	Manage workspace	1
C.2	Send messages	2
C.3	Send email within the application	2
C.4	Start voice communication	2
C.5	Start video communication	3
C.6	Make appointments & track appointments	1
C.6.1	Add appointments	1
C.6.2	Accept invitations	1
C.7	Show a calendar common in the workspace	1
C.8	Use file sharing module	2
C.8.1	Add and remove files	3
C.8.2	Add and remove folders	3
C.8.3	Edit Share files	2
C.9	Invite new members	1
D.1	Update notifications through email	1
D.2	Adapt the user interface manually	1
D.3	Adapt the user interface automatically	2
D.4	Skills matching automatically	1

8 List of Figures

9 List of tables