HELASCOL Helping elders to live an active and socially connected life by involving them in the digital society	Deliverable reference: **D3.1**	Date: 19th November 2013
	Title: **System Architecture Specification**	
Project Title: Helping elders to live an active and socially connected life by involving them in the digital society	Responsible partner: IncreaseTime	
	Editors: Gil Gonçalves; Raquel Sousa	
Contract no. AAL_08-1-2011-00011 (Magyar)	Approved by:	
AAL AMBIENT ASSISTED LIVING ASSOCIATION AAL-2011-2	Classification: Confidential	

Abstract:

This report presents a description of the architecture of a web responsive application directed to the "assisted living" market. It describes the technologies and presents the modular diagram of the application and its components interaction.

Keywords:

This page was intentionally left blank

Project:	HELASCOL
Contract no.	AAL_08-1-2011-00011 (Magyar)
Start – End dates	
Deliverable	D3.1
Date	November 2013
Version	6.0

This page was intentionally left blank

Document History

Revision	Date	Author	Organization	Description
0.1	04.09.2013	Gil Gonçalves	iTime	Initial Structure and Content
0.2	10.09.2013	Raquel Sousa	iTime	Chapter 1 and 2
0.3	27.09.2013	Raquel Sousa	iTime	Chapter 3 and 4
0.4	18.03.2014	Raquel Sousa	iTime	Chapter 5 and 6
0.5	19.03.2014	Raquel Sousa	iTime	Chapter 2
0.6	03.04.2014	Ferenc Nemecsek	Kapsch	Chapter 4

This page was intentionally left blank

Table of Contents

Glossary

HTML5: HTML5 is a markup language used for structuring and presenting content for the World Wide Web and a core technology of the Internet.

CSS3: “CSS” is an acronym for Cascading Style Sheets, a web-based markup language used to describe the look and formatting of a website to the browser, most commonly used in HTML or XHTML web pages. “CSS3” simply refers to the latest incarnation of CSS, with additional capabilities far beyond the scope of the first two generations.

JSON: JSON is short for JavaScript Object Notation, and is a way to store information in an organized, easy-to-access manner. In a nutshell, it gives us a human-readable collection of data that we can access in a really logical manner.

JQuery: A lightweight cross-browser JavaScript library that emphasizes interaction between JavaScript and HTML.

ASP.NET: ASP.NET is a Web platform that provides services to build enterprise-class server-based Web applications. ASP.NET is built on the Microsoft's .NET Framework. Applications can be written in any language that is compatible with the common language runtime (CLR), including Visual Basic and C#.

Web Service: A Web service is a method of communications between two electronic devices over the World Wide Web. It is a software function provided at a network address over the web with the service always on as in the concept of utility computing.

MVC: In object-oriented programming development, model-view-controller (MVC) is the name of a methodology or design pattern for successfully and efficiently relating the user interface to underlying data models.

ASP.NET MVC : ASP.NET MVC is a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and that gives full control over markup for enjoyable, agile development. ASP.NET MVC includes many features that enable fast, TDD-friendly development for creating sophisticated applications that use the latest web standards.

1 Introduction

1.1 Project scope

The „Helping elders to live an active and socially connected life by involving them in the digital society" project addresses the objectives of the call by offering a 360 degree user involvement methodology to examine how a new approach towards digital technologies can be harnessed to support the involvement of elderly people in digital society. The proposal intends to synthesize the skills, experience and knowledge of the consortium members in developing a state-of-the-art platform and service package backed with feasible business models which supports the on time and on budget realization and market introduction of the call objectives.

The project focuses on providing an enriched communication experience, anywhere, anytime and to any device with accessible, intuitive, easy to use, multimodal User Interfaces. We believe that the right service and the right content are only accepted by the end users if it is delivered on the right device, one that they are used to. This can be the screen of the television, mobile phones, etc. Our goal is to enable elderly people, their family and social surrounding to share their everyday experiences anytime, anywhere and help them make use of existing and currently developed multimedia services to generate the sense of closeness and community belonging they are searching for. This enriched experience, which allows users to share their emotions and experiences in a vivid and interactive way, requires a new approach both in services and the technology that supports them.

1.2 Purpose of this document

The objectives to be achieved within WP3 consist on defining and specifying the overall system and subsystems architecture and the definition of the external interfaces of the various integrating modules. This comprises providing the architecture and specifications, including the definition and description of the modules and subsystems inside it.

Specific objectives within WP3:

- UI specifications
- Service delivery platform specification
- Integration/interoperability requirements and specifications
- Identify ambient intelligent approaches and to use/integrate smart home technologies

Furthermore we are also to research more deeply elderly people's habits for the use of robot control. In order to make smart home devices controllable we will base our approach on the use of a tablet device with specific applications for smart home environments, based on the underlying platform provide by the KeepCare© system from partner ITIME. This system

is highly extensible due to a plug-in framework based on standard web-service technology, making the integration of new functionalities in the system (e.g. control of smart devices in the home environment) very straight forward.

With the lead of ITIME, the other partners (KECELCOM, SUPSI, KAPSCH) define the overall system architecture having as reference the KeepCare© system architecture. In this task integration/interoperability requirements will also be defined.

A specification document that is to define the interdependency between the system and subsystems and serve as a basis for the other development tasks is published in M16

1.3 Structure of the document

This report presents on section 2 the technological overview where the technologies selected for developing the application are described. Sections 3 and 4 show the logical and physical architectures which corresponds, respectively, to the modular diagram of the application and to the description of the various components that comprise the system to be develop. In section 5, some solutions that have been exploited for the development of this application are described. Section 6 concludes the report.

2 Technology Overview

In this section we will describe the frameworks and technologies to be used on our application. As a web application one must expect the usual platforms such as HTML5, CSS3, JSON, JQuery, Bootstrap.

2.1 ASP.NET

The application will be constructed on the well know framework .NET. Being a web based app ASP.NET will be the base for the development. .NET is a software framework developed by Microsoft with a large reusable library of classes and a development environment that helps developers rapidly and graphically build applications.

2.2 Microsoft SQL Server 2012

For our database we picked microsoft's SQL Server 2012. It's interactions with .NET are well documented and its the go-to database when developing for this platform.

2.3 XSockets.NET

For the video module we intend on using XSockets.NET. XSockets.NET is a real time communication platform built on Microsoft.NET Technology that provides integration with WebRTC.

2.4 AngularJS

The client side of the application will be built on AngularJS. AngularJS is an open-source JavaScript framework, maintained by Google, that assists with running single-page applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.

2.5 ChatJS

We will be using ChatJS for our chat module. ChatJS is highly customizable and supports both the SignalR and the long-polling adapter which are essential to important features such as online and offline status.

2.6 D3.js

The visualisation of chart will be handled by D3.js. D3.js is a JavaScript library for manipulating data and allows us to create real-time graphics.

2.7 Google API's

We will use several API's supplied by google:

2.7.1 Google Calendar API

The Google Calendar API allows a program to perform many of the operations available via Google Calendar web interface. Using this API, it is possible to search for and view public calendar events. Authenticated sessions can access private calendars, as well as create, edit, and delete both events and the calendars that contain them.

2.7.2 Google Contacts API

The Google Contacts API allows client applications to view and update a user's contacts. Contacts are stored in the user's Google Account; most Google services have access to the contact list. The application use the Google Contacts API to create new contacts, edit or delete existing contacts, and query for contacts that match particular criteria.

2.7.3 Google Gmail API

The Google Gmail API allows client application to manage client’s gmail inbox on behalf of a user.

2.7.4 Google+ Sign-In

By adding Google+ Sign-In, you bring the power of Google to your site. When a user is signed in, you get an OAuth token for making API requests on their behalf, which you can use to better understand your user, connect them with their friends, and create a richer and more engaging experience.

The first time a user clicks on the sign-in button, they will see an authorization dialog. This dialog outlines how the application will use their data. The user then can consent to the authorization or cancel. After authorizing, a returning user will not be prompted again for authorization. A user always has the option to revoke access to an application at any time.

3 Logical Architecture

In the following diagram we give an overview of the several modules that comprise our application.

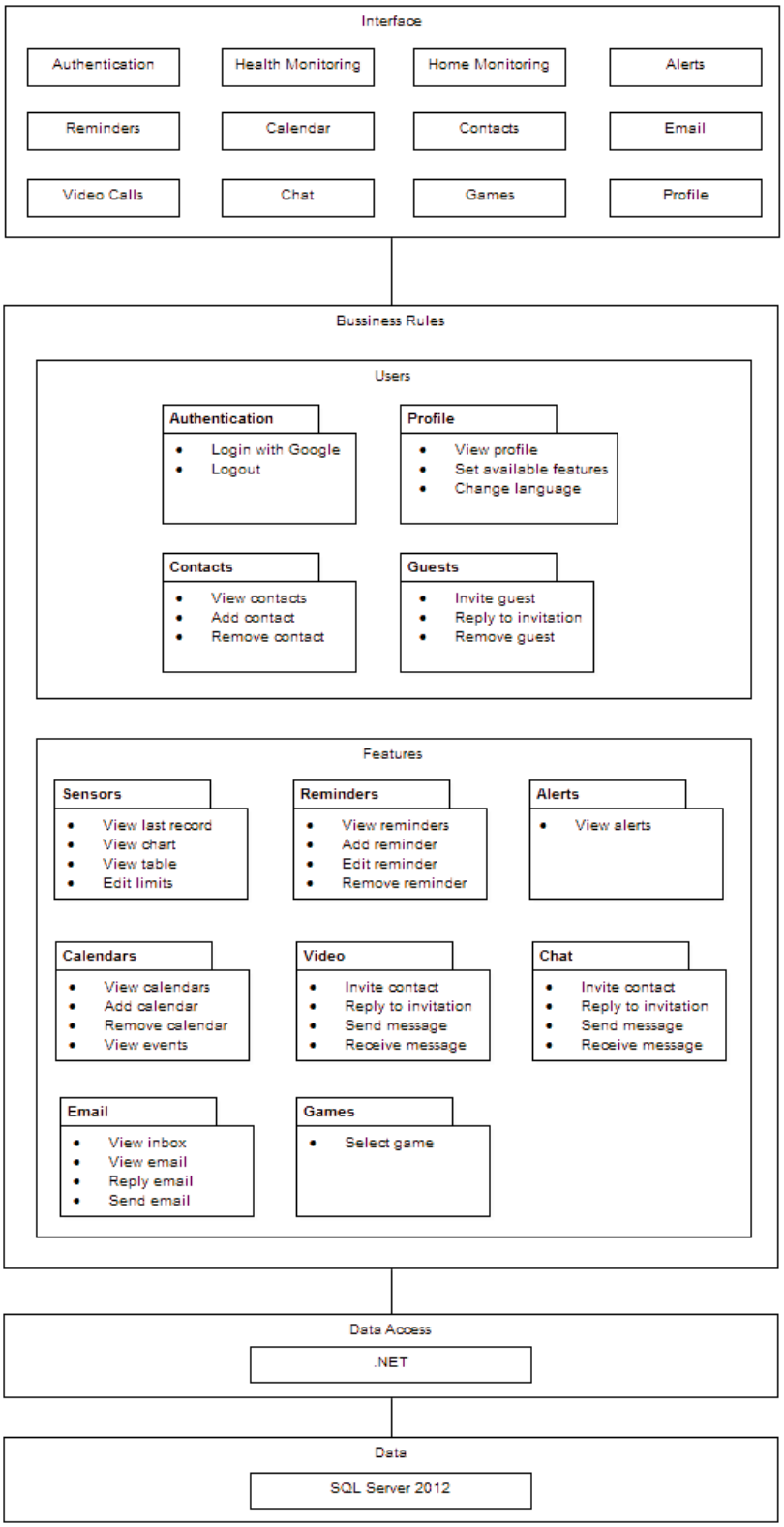

Figure 1. Logical architecture diagram

4 Physical Architecture

In this section we describe the different components involved and how they are represented in a physical environment.

4.1 Component diagram

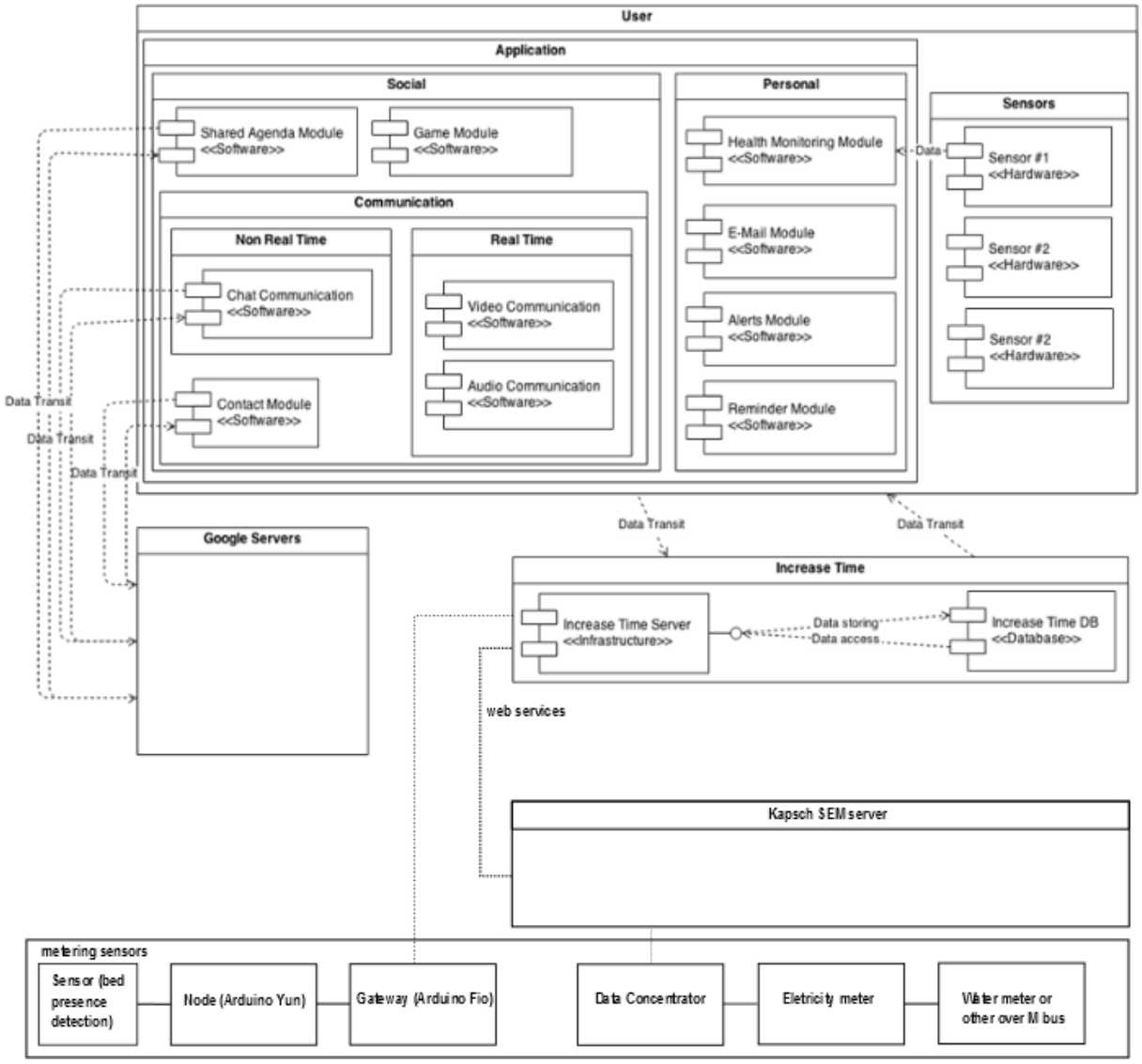

Figure 2. Component diagram

4.2 Deployment diagram

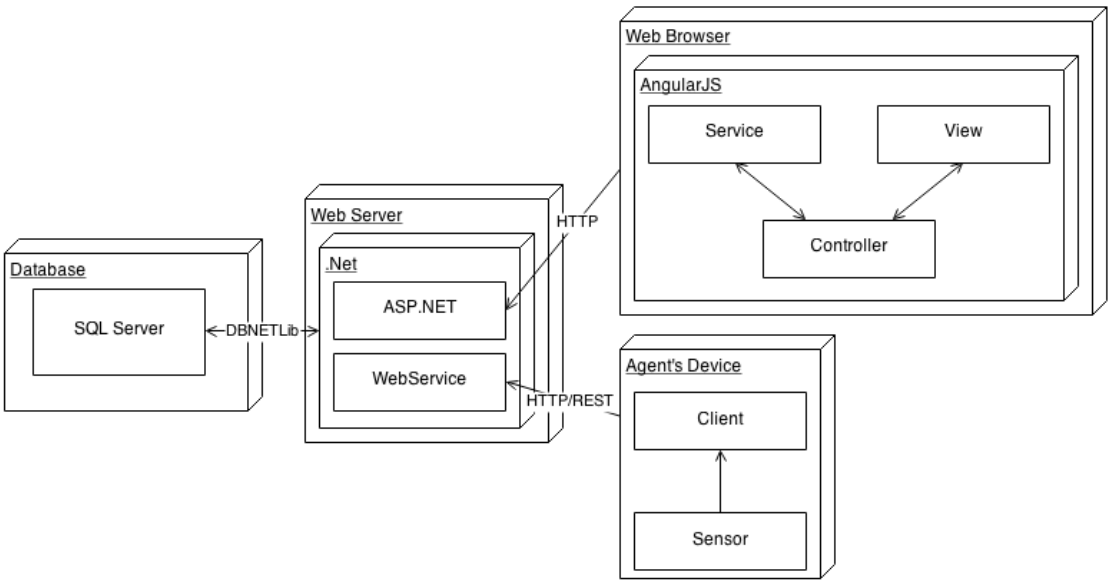

Figure 3. Deployment diagram

4.3 Physical architecture description

4.3.1 WebServer

The web server will be running on the .NET platform and providing web support via the ASP.NET framework. It will also provide separate web server that implements the REST protocol for interactions with potential devices such as health and home sensors. This web server will be hosted on the IncreaseTime facilities.

4.3.2 Database

The database will be created on SQLServer 2012 and it too wil be hosted on cloud.

4.3.3 WebBrowser

The client side will be defined on the user's browser and will be supported by AngularJS to create a seamless interaction with the WebServer.

4.3.4 Agent's Device

The agent's device will most likely be an android phone/tablet that connects to a sensor. This device should be able to communicate with our WebServer via our WebService in order to insert health and environmental data into our system.

4.3.5 Kapsch SEM server

Kapsch SEM server's task is to collect data received from the meters transferred over the data concentrator. Data concentrator is communicating over the in-house installed router and

Internet modem. All the information received is stored on the MS SQL data base on the SEM server. The received data is generally either consumption data or alarm information. Increase Time server receives the consumption data from SEM server over the web services. Alarms that are coming from the meters and data concentrator are coming to SEM server as the event occurs and SEM server sends this alarm information immediately to Increase server over another web service defined on the Increase server side. Consumption and alarms are giving additional information on the in-house behavior of the elder people.

4.3.6 Metering sensors

Metering sensors in Helascol project comprise of the meters and bed presence sensor. The bed presence senor is detecting whether the person is in the bed and over his node and gateway the data is sent to Increase Time server. Meters are described in previous heading.

5 Key Design Decisions

In order to achieve a more robust application it's important to recognize that nowadays there are good solutions for many of problems that may exist. This section describes some of that solutions that will bring more confidence to the final product and will allow a lower development time.

5.1 MVC pattern

The Model-View-Controller (MVC) pattern will be used in the server for separating the representation of the database tables from the database operations and from the server responses (either HTML for the web application or JSON for the agents API or the web application AJAX requests). Considering only the client side AngularJS also uses MVC empowering the relation between HTML and JavaScript.

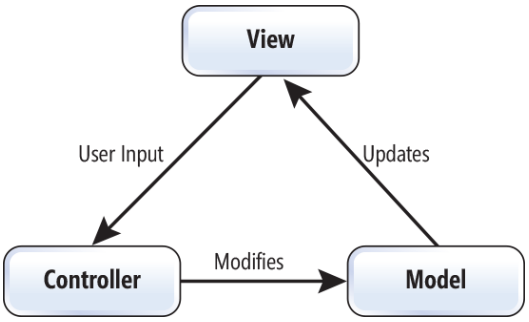

Figure 4. MVC pattern diagram

5.2 MVVM pattern

Considering the all system it's used the Model-View-ViewModel (MVVM) pattern so that the data being presented to the user is synchronized with the database. In this case as the model is far (server-side) from the view (client-side) the MVC pattern is not applicable and the updates to the model have to be made by the View Model that is in the client-side.

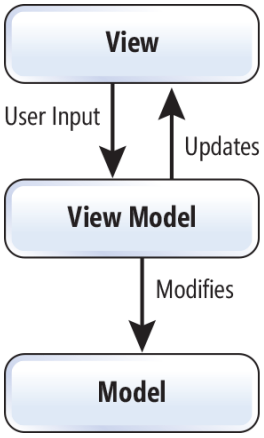

Figure 5. MVMM pattern diagram

5.3 CRUD

The database CRUD operations will be done using the Entity Framework that allows developers to work on a higher abstraction level than SQL resulting in a less code application with high robustness.

5.4 REST

Despite of not being a standard REST is known as a good practice to implement web services because it ignores implementation details and enables users to better focus on their purpose.

6 Conclusion

In this preliminary report of architecture we presented a brief description of the technologies that will be used in the development of HELASCoL platform, as well as the diagram of the modules and the application components. We also presented technical options that were considered interesting for the development of the application and their main advantage and disadvantages.

This report should be considered as a preliminary report as the development of the application itself has not started. In the development stage of the application a more complete report of the architecture will be developed.