TECHNISCHE UNIVERSITÄT DRESDEN

Master Thesis

Generic Frontend for Exploring Sensor and Information Services

Author:
M.Sc. Uliana Andriieshyna
Matrikel-Nr: 3828303

Supervisors:
Dr.-Ing. Josef Spillner
Prof. Dr. rer. nat. habil.
Dr. h. c. Alexander Schill

April 10, 2014

Declaration of Authorship

I, Uliana Andriiehyna, declare that this thesis, titled "Generic Frontend for Exploring Sensor and Information Services", and the work presented in it have been done on my own without assistance. All information directly or indirectly taken from external sources is acknowledged and referenced in the bibliography.

Dresden, XX.XX.2013

Contents

Chapter 1

Introduction

The increasing numbers of sensor devices has increased the number of sensor-specific protocols, platforms and software. As a result various approaches have been proposed to interconnect, maintain and monitor various type of sensors[1, 2, 3]. That specificly focused on a platform development, protocol definition and software architecture for the concreate user-oriented requirements and for a narrowly focused areas of usage instead of defining a common system approach. Mainly in proposed approaches was discovered such type of questions as security and privacy, easy of development and monitoring, social dimensions [4], that completely don't consider requirements and criteria of an end-user. Therefore the area of research of this master thesis is dedicated to define generic frontend for exploring sensor and information services. The folowing sections ground the motivation for the chosen research field, define the central research questions and goals of this master's thesis, and describe the overall structure of the work.

1.1 Motivation

In the recent years with the technological progress in the computer science, information systems, and in particular sensor data systems, have become an essential part in daily life of the modern society. People have started to use them more often not only for manufactory, business, education but also for private reasons. Currently, most of the research is concerned with the protocol and middleware levels, whereas the potential of a generic interactive access to sensor and information services needs to be explored. This involves their selection, mashups, and usage within a client-controlled interface. In this master thesis, a first web-based prototype (portal) for such services is to be created. Users should be able to explore not just services, but also the information provided by them, and eventually be led to advanced usage patterns such as the development of third-party applications to access the information data and real-time streams.

1.2 Research Questions and Goals

Creating composite third-party services and applications from reusable components is an important technique in software engineering and data management. Although a large body of research and development covers integration at the data and application levels, weak work has been done to facilitate it at generic level. This master thesis discusses the existing user interface frameworks and component technologies used in presentation integration, illustrates their strengths and weaknesses, and presents some opportunities for future work.

As mentioned in the previous section, there are already exist many solutions for creating sensor-aware applications. But these platforms focuses on a single area of usage and they are not commonly suitable to support the dynamic and adaptable composition and usage of different type of sensors in one portal.

- Concept for a generic information and sensor service portal
- Development of the portal and associated dependency tools
- Demonstration using a convincing scenario

Therefore, this thesis is aimed at the development of a concept that provides users a possibility to personalize their current environment indepently from any type and kind of devices.

1.3 Structure

The thesis is structured in the following way:

Chapter 2 defines the background of the master's thesis describing the basic used terminology and the foundation platforms. A reference scenario and the requirements to a concept that has to be developed are also introduced in this chapter.

Chapter 3 is devoted to the state of the art analysis. The related research works in the areas of the sensor-driven platforms, the component based groupware systems, the browser based and non-browser based systems are investigated and evaluated against the defined requirements.

Chapter 4 focuses on the concept of the generic Frontend for exploring sensor and Information sevices, considering possible approaches, strategies, frameworks and necessary criteries, defined in *Chapter 3*.

Chapter 5 provides the implemented functionalities of the concept and describes evaluaion of results.

Chapter 6 concludes the master's thesis underlining the achieved goals and providing prospects for the future work.

Chapter 2

Foundations and Requirements Analysis

The fundamental terms used in this thesis are described below for better understanding of the presented research work.

2.1 Frontends Requirements

In computer science, the frontend is responsible for collecting input in various forms from the user and processing it to conform to a specification the backend can use. The frontend is an interface between the user and the backend[5] and the separation of software systems into front and back ends simplifies development and separates maintenance. Therefore need to be distinguished what are the main requirements to a generic frontend for exploring sensed data, i.e. :

- Loose coupling: each of systems components has, or makes use of, little or no knowledge of the definitions of other separate components. Where the main goal is to avoid dependencies between components and easy deployment of future enhancement.

- Fine-grained structure: split the system into a small parts, such that it can be distributed across internet, by avoiding single point of failure.

- Multy-user capability: defining to every user according to their type and rights visibility rules and concreate interface view targeting.

- Cross-platforming or multi-platform: a possibility of application to be run on any type of device(e.g., smartphone, notebook, tablet) without special preparation or changes.

- Adaptivity: an ability of a user-friendly interface to automatically adapt to any size of device screen, by provisioning high usability performance.

- User friendly or usability: web-based interface, that can be easily explored by user, without any knowledge about current system.

2.1.1 Loose Coupling

Together with a loose coupling comes into a picture fine-grained system structure. For both of them works the principle to distribute the systems by using small components.

2.1.2 Multy-User Capabilities

Before user loggin into a system, without any knowledge about system, should be clarified what shold be shown on a main screen and how to define usability steps and where.

2.1.3 Design Strategy

Because of the competing interests of cross-platform compatibility and advanced functionality, numerous alternative web application design strategies have emerged. Such strategies include:
Graceful degradation
Graceful degradation attempts to provide the same or similar functionality to all users and platforms, while diminishing that functionality to a ‘least common denominator’ for more limited client browsers. For example, a user attempting to use a limited-feature browser to access Gmail may notice that Gmail switches to "Basic Mode", with reduced functionality. Some view this strategy as a lesser form of cross-platform capability.
Separation of functionality
Separation of functionality attempts to simply omit those subsets of functionality that are not capable from within certain client browsers or operating systems, while still delivering a ‘complete’ application to the user. (See also: Separation of concerns).
Multiple codebase
Multiple codebase applications present different versions of an application depending on the specific client in use. This strategy is arguably the most complicated and expensive way to fulfill cross-platform capability, since even different versions of the same client browser (within the same operating system) can differ dramatically between each other. This is further complicated by the support for "plugins" which may or may not be present for any given installation of a particular browser version.
Third-party libraries
Third-party libraries attempt to simplify cross-platform capability by "hiding" the complexities of client differentiation behind a single, unified API.
Responsive Web design
Responsive web design (RWD) is a Web design approach aimed at crafting sites to provide an optimal viewing experience—easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices (from mobile phones to desktop computer monitors).

2.1.4 Usability

Intuitive interfaces The primary notion of usability is that an object designed with a generalized users' psychology and physiology in mind is, for example: More efficient to use—takes less time to accomplish a particular task Easier to learn—operation can be learned by observing the object More satisfying to use

ISO defines usability as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use." The word "usability" also refers to methods for improving ease-of-use during the design process. Usability consultant Jakob Nielsen and computer science professor Ben Shneiderman have written (separately) about a framework of system acceptability, where usability is a part of "usefulness" and is composed of:[4] Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design? Efficiency: Once users have learned the design, how quickly can they perform tasks? Memorability: When users return to the design after a period of not using it, how easily can they re establish proficiency? Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors? Satisfaction: How pleasant is it to use the design?

2.2 Data Sources

Main focuse -types of common data sources in Web
-interface for interconnection

2.2.1 Sensors

2.2.2 Generic Considerations

2.3 Concept Requirements

Like most modern applications, each of these is structured into three layers: presentation, application (also called the business-logic layer), and data.

1. the granularity of the functions that the component applications provide is generally well suited for high-level integration (for example, we can tell an application to begin monitoring machine xyz without considering how this activity will affect data in the integrated application's database)

2. it's more stable because the component application is aware of the integration (it exposes the API) and will attempt to stabilize the interface across versions

2.4 Summary

Chapter 3

State of the Art

The following chapter covers an overview and analysis of the existent solutions in the related research areas of web-based third-party applications, which were designed specially for retrieving differet type of sensed data to the user. At the beginning the prominent examples of dashboards platforms are studied and evaluated against the requirements described in the previous chapter with a purpose to clarify their individual capabilities.

3.1 ?Web of Sensors?

Since the thesis is targeted at the creation of a generic user-friendly data stream interaction system and currently every project focused on a specific area of realization and concrete sensor types, such as urban environment[3]. The real-time environmental monitoring portal or geospatial infrastructure for effectively or efficiently collecting and serving vast field data over the web. Internet based urban environment observation system that can real-time monitor environmental changes of temperature, humidity, illumination or air components in urban area. It provides web-based platform, where end-user can simply monitor urban area in his/her city. In environmental monitoring, the field server for constructing outdoor sensor network is a Web-based field observation device, which detects field environmental parameters and publishes them on Internet in real-time.
Smart city[1]tool using information technology and communication (ICT) to help local government to monitoring what currently happened in the city. pplication for monitoring city in single dashboard to help summarize the current condition of city. The architecture system use network sensor consisting of sensor nodes that has function to capture city condition like temperature, air pollution, water pollution, traffic situation. Also we can add another information socio-economic situation like public health service, economic indicator, energy supplies, etc. We have successfully developed the prototype of the smart city dashboard the give more accurate information of Bandung City, one of big cities in Indonesia. This study has implemented prototype system of smart city dashboard at Bandung. It consists of network sensor, server, application and also communication protocol which is used for city monitoring. The summary information of city can be displayed in single view to help people watching, analyzing and action to what being happen at the real-time in the city.

*Microsoft SensorMap[6]*has proposed a system to monitor and present physical sensors in the real world. SensorMap allows owners of sensor networks to register their physical sensors and publish their data on SensorMap. They use GeoDB to store the sensor network information, DataHub to retrieve the new sensor data to enable real time services, and the aggregator to summarize sensor data in a specific area to clients.
LiveWeb Portal[7] presents the architecture, design, and application of a sensorweb service portal, where sensorweb is a global observation system for varied sensory phenomena from the physical world and the cyber world. This system has been used to represent and monitor real-time physical sensor data and cyber activities from ubiquitous sources. LiveWeb meets its goal of providing an efficient and robust sensor information oriented web service, enabled with real-time data representation, monitoringand notification.LiveWeb has the following properties: the system enables sensorweb service accessible from anywhere, makes sensor network data readable by anyone, sensor network sharing, the system make sensor data format transparent to data users, real-time data display suits sensor data properties, an offline alert system strengthens real-time features.
Internet of Things[2] where presented a service platform based on the Extensible Messaging and Presence Protocol (XMPP) for the development and provision of services for Internet of Things(IoT) mainly focusing on the integration of things based on service technologies, scenarios in domains like smart cities, automotive or crisis management require service platforms involving real world objects, backend-systems and mobile devices. And argued necessary usage of XMPP client as protocol for unified, real-time communication and introduce the major concepts of our platform. Based on two case studies we demonstrate real-time capabilities of XMPP for remote robot control and service development in the e-mobility domain.
Dynvoker Portal[8] a generic human-driven ad-hoc usage approach, by including rapid service testing and dynamic inclusion of services as plugins into applications. Dynvoker consists of a relatively small application core which can be run as a servlet, a web service or a command-line application. explore method-centric and resource-centric services alike, output forms in various formats or integrate GUI services to provide a richer user experience. The generic design of many parts of Dynvoker has yielded a lightweight architecture which is freely available to any interested person as an open source project.
Sensor Web Enablement project[9] is focused on developing standards to enable the discovery of sensors and corresponding observations, exchange, and processing of sensor observations, as well as the tasking of sensors and sensor systems. Open Geospatial Consortium, Inc. members specifies interoperability interfaces and metadata encodings that enable real time integration of heterogeneous sensor webs into the information infrastructure. Developers will use these specifications in creating applications, platforms, and products involving Web-connected devices such as flood gauges, air pollution monitors, stress gauges on bridges, mobile heart monitors, Webcams, and robots as well as space and airborne earth imaging devices. In this publication by OGC was defined such an important XML-based standatrds as: Sensor Model Language (SensorML), Sensor Observation Service (SOS), Web Notification Service (WNS) etc. As subproject calls SANY(Sensors Anywhere) focuses on interoperability of in-situ sensors and sensor networks. The goal for the SANY architecture is to provide a quick and cost-efficient way to reuse data and services from currently incompatible sensors

and data sources in future environmental risk management applications. By developing a standard open architecture and a set of basic services for all kinds of sensors, sensor networks, and other sensor-like services, the SANY IP supports and enhances both GMES (Global Monitoring for Environment and Security, a major European space initiative) and GEOSS (Global Earth Observation System of Systems) in the area of in-situ sensor integration. Though the SANY work enhances interoperability for monitoring sensor networks in general, the application focus is on air quality, bathing water quality, and urban tunnel excavation monitoring.
VICCI Project(Visual and Interactive Cyber-physical Systems Control and Integration)[10, 11]. The scope includes smart home environments and supporting people in the ambient assisted living, considers the software-technical side of so-called “Cyber-physical systems” (CPS). This term includes complex, embedded systems, which connect the virtual and the physical world with each other (IoT)in different application scenarios. The main uses of CPS are in logistics, traffic optimization, in the use of robots in the industrial and domestic sectors, in modern energy networks (Smart grid), in the building and factory automation (Smart factory), as well as in the field of intelligent office installations (Smart Office). The aim of project VICCI is the creation of software engineering principles that are necessary for the development of complex cyber-physical systems. Firstly, CPS should be made understandable and accessible by means of a comprehensive control centre. Secondly, platforms that enable the development and marketing of software for complex CPS through a pure control panel are to be developed. A domestic environment is considered a sample scenario in which a person with reduced mobility is supported by sensors, actuators and a service robot, which is currently seen as a complex cyber-physical system. No concreate frontend or any kind of user-friendly have been not yet developed.
A series of articles devoted to integrate sensed data into a Cloud. Special attention is given to privacy-relevant or otherwise sensitive information that stores in Cloud. SensorCloud[12], a cloud design for user-controlled storage and processing of sensor data proposed security architecture enforces end-to-end data access control by the data owner reaching from the sensor network to the Cloud storage and processing subsystems as well as strict isolation up to the service-level. In this paper authors implement transport security mechanisms for communication with the Cloud, applies object security mechanisms to outbound data items, and performs key management for authorized services. *CloudRemix[13]* a Personal and Federated Cloud Management Cockpit, an interactive cockpit to manage personal clouds and their federations. Is a new techniques for users to perform asset discovery, exchange and management in Cloud area. The CloudRemix prototype demonstrates its utility to manage personal clouds in both social and market-driven environments. The goal of CloudRemix is to be open, user-centric regarding the manageable assets, and flexible regarding their free or commercial exchange, with or without explicit contract negotiation. CloudRemix is an open-source web-based cockpit application with support for multiple users. Each user gets to see an aggregated list of both local and remote services of each of the asset types.

3.2 Frontend Development Approaches

In computer science, the frontend is responsible for collecting input from user and processing it to a backend system and another direction - collecting data from backend, namely sensor data steam, and processing it to the user-friendly interface. Therefore, on the one side, generic frontend has to satisfy architecture requirements from backend, such as: fine-grained distributred structure, cross-platforming, multy-user capabilities; and on the other side, define a dynamic user-friendly interface to a end-user. And to satisfy aforementioned requirements from backend server it is necessary to compare all available web-based applications.
To retrieve sensor data from different resources in one web-based interface existent next approaches :

- portal with portlets,
- mashup[1],
- HTML5 technology

Portal technology brings information together from diverse sources in a uniform way. Usually, each information source gets its dedicated area on the page for displaying information (a portlet); often, the user can configure which ones to display. The extent to which content is displayed in a 'uniform way' may depend on the intended user and the intended purpose, as well as the diversity of the content. Very often design emphasis is on a certain 'metaphor' for configuring and customizing the presentation of the content and the chosen implementation framework and/or code libraries[14, 15]. In portal technologies end-user can customize number of retrieved data sources, but for that he has to be aware what is it and how to integrate it in portal. User interface in portals have fixed layout, style and location on the web page. To make changes in it, end-user needs to have a deep knowlendge of the system architeture and of whole portal entirely.
Mashup is a web page, or web application, that uses content from more than one source to create a single new service displayed in a single graphical interface. The term implies easy, fast integration, frequently using open application programming interfaces (API) and data sources to produce enriched results that were not necessarily the original reason for producing the raw source data. The term mashup originally comes from pop music, where people seamlessly combine music from one song with the vocal track from another-thereby mashing them together to create something new. The main characteristics of a mashup are combination, visualization, and aggregation. It is important to make existing data more useful, for personal and professional use. To be able to permanently access the data of other services, mashups are generally client applications or hosted online. Both commercial products and research prototypes have a broad range of features that simplify a mashups design process, and provide mashups storage and publication. But to customize retrived resources end-user have no option, as use only predefined type and numbers of applications, that was created by application or platform developer. Also Mashup approach is strictly platform- and customer-oriented. It is simply provides stack of tools, by using which user

[1]http://www.programmableweb.com/applications

through an user-friendly interface The architecture of a mashup is divided into three layers:

- *Presentation / user interaction:* this is the user interface of mashups. The technologies used are HTML/XHTML, CSS, Javascript, Asynchronous Javascript and XML (Ajax).

- *Web Services:* the product's functionality can be accessed using API services. The technologies used are XMLHTTPRequest, XML-RPC, JSON-RPC, SOAP, REST.

- *Data:* handling the data like sending, storing and receiving. The technologies used are XML, JSON, KML.

Architecturally, there are two styles of mashups: Web-based and server-based. Whereas Web-based mashups typically use the user's Web browser to combine and reformat the data, server-based mashups analyze and reformat the data on a remote server and transmit the data to the user's browser in its final form[16].
Mashups and portals are both content aggregation technologies. Portals are an older technology designed as an extension to traditional dynamic Web applications, in which the process of converting data content into marked-up Web pages is split into two phases: generation of markup "fragments" and aggregation of the fragments into pages. Each markup fragment is generated by a "portlet", and the portal combines them into a single Web page. Portlets may be hosted locally on the portal server or remotely on a separate server.

3.3 HTML5 Technology

To satisfy one of the main requirement about dynamic user-friendly interface, adaptable to any kind of device, mashup architecture should be enhanced with a HTML5 Technology. Based on various design principles, that truly embody a new vision of possibility and practicality[17].

- Compatibility(inharit all previous techniques and standards)
- Utility
- Secure by Design(origin-based security model that is not only easy to use but is also used consistently by different APIs.)
- Separation of Presentation and Content(CSS3)
- Interoperability(Native browser ability instead of complex JavaScript code; a new, simplified DOCTYPE;simplified character set declaration; powerful yet simple HTML5 APIs)

- Universal Access(suport users with disabilities by using screen readers; media independence-HTML5 functionallity should work across all different devices and platforms; support for all world languages)

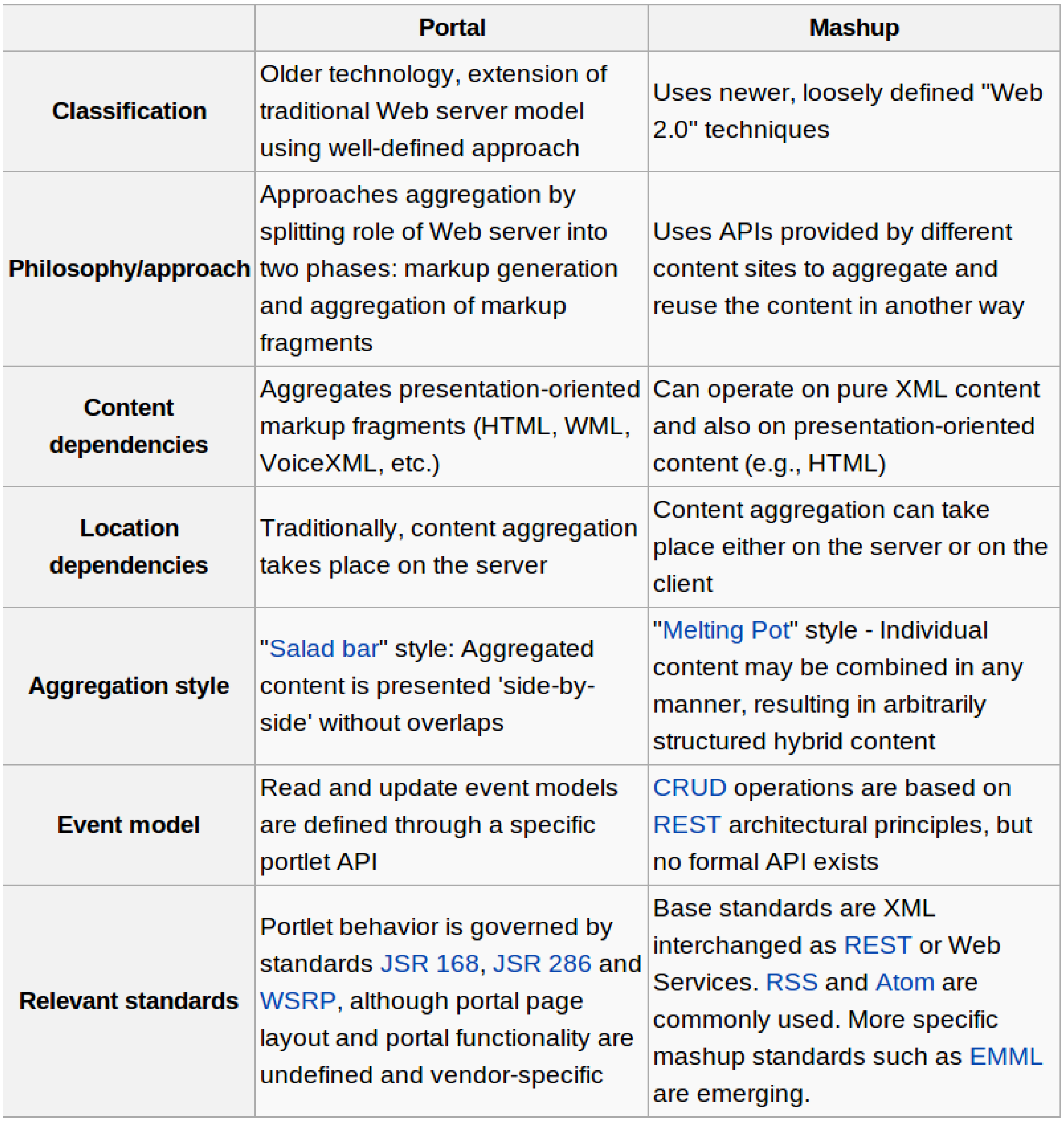
3.4 UI Usability

ISO defines usability as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use." Main goals which user-friendly interface have to satisfy from a user point of view are[18, 19, 20]:

- Clarity: a user have to easy understand what is the content, how to explore content that provided by application and which rights have a user
- Evidence: accordance of standardized icons, titles, layout (intuitive design)
- Satisfaction: How pleasant is it to use the design?
- Adaptivity: nicely feet to a different types of devices

3.5 Summary

This chapter briefly introduced main approaches for building web-based dashboards by retriving sensed data. Main focus was given to its multy-user usability, adaptive UI design, dynamic content composition. Where portal and mashup technology come into a picture.



	Portal	Mashup
Classification	Older technology, extension of traditional Web server model using well-defined approach	Uses newer, loosely defined "Web 2.0" techniques
Philosophy/approach	Approaches aggregation by splitting role of Web server into two phases: markup generation and aggregation of markup fragments	Uses APIs provided by different content sites to aggregate and reuse the content in another way
Content dependencies	Aggregates presentation-oriented markup fragments (HTML, WML, VoiceXML, etc.)	Can operate on pure XML content and also on presentation-oriented content (e.g., HTML)
Location dependencies	Traditionally, content aggregation takes place on the server	Content aggregation can take place either on the server or on the client
Aggregation style	"Salad bar" style: Aggregated content is presented 'side-by-side' without overlaps	"Melting Pot" style - Individual content may be combined in any manner, resulting in arbitrarily structured hybrid content
Event model	Read and update event models are defined through a specific portlet API	CRUD operations are based on REST architectural principles, but no formal API exists
Relevant standards	Portlet behavior is governed by standards JSR 168, JSR 286 and WSRP, although portal page layout and portal functionality are undefined and vendor-specific	Base standards are XML interchanged as REST or Web Services. RSS and Atom are commonly used. More specific mashup standards such as EMML are emerging.

Figure 3.1: Comparative Characteristic

Chapter 4

Concept

The chapter poses and describes a concept of a user-friendly generic frontend for exploring sensor data, through designing a software architecture and a mockup of a web-based user interface that in the same time controlled and provisioned by end users request. The concept is developed based on the analysis of the current state of the art, up-to-date technologies and usability characteristics. According to defined requirements in chapter 2 to the third-party services and applications and knowledges gained from the studied related works (chapter 3), was proposed concept based on 3-tier architecture and in details described in section 4.2 - 4.4.

4.1 3-tier Architecture

Since the concept of a generic frontend should be distributed as much as possible, it is necessary to determine software architecture to a 3-tier architecture, in which presentation, application processing, and data management functions are logically separated. Figure 4.1 shows this architecure:

- Data Tier: data from different types of sensors
- Application Tier: registry and proxy
- Client Tier: web-based UI

Client Tier hosts the presentation layer components. The main function of the interface is to translate tasks and results to graphical user interface that can be easily understandable and explorable from any kind of device. That satisfy requirements of the usability(section 3.4).
Application Tier includes business logic, logic tier and data access tier. It controls an application's functionality by performing detailed processing, transformation of one type data to another one, defines an interface of interconnection between client tier and data tier.
Data Tier consists source of data that have to be retrieved by application tier to a Client Tier, by request from a end user. This tier keeps data neutral and independent from application

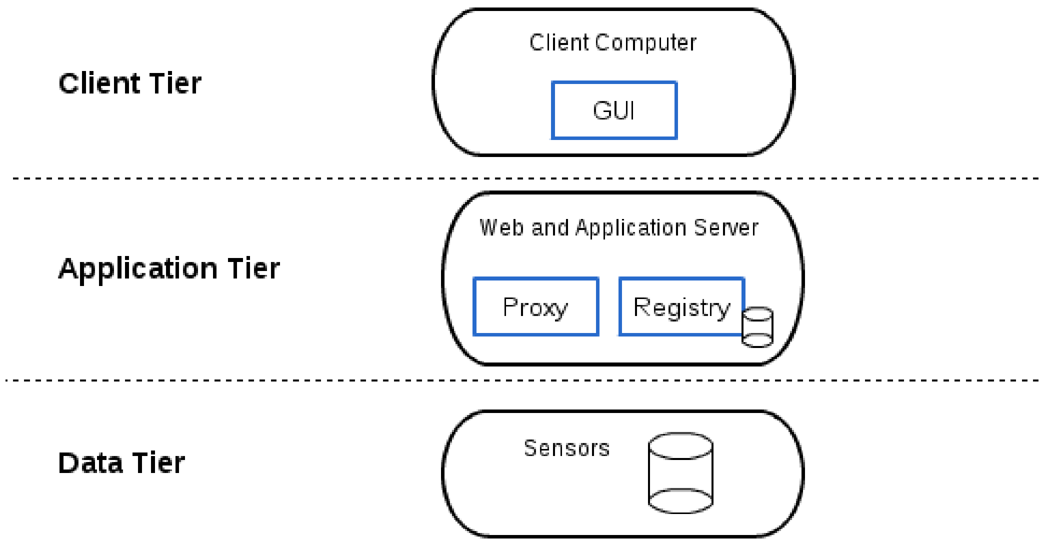


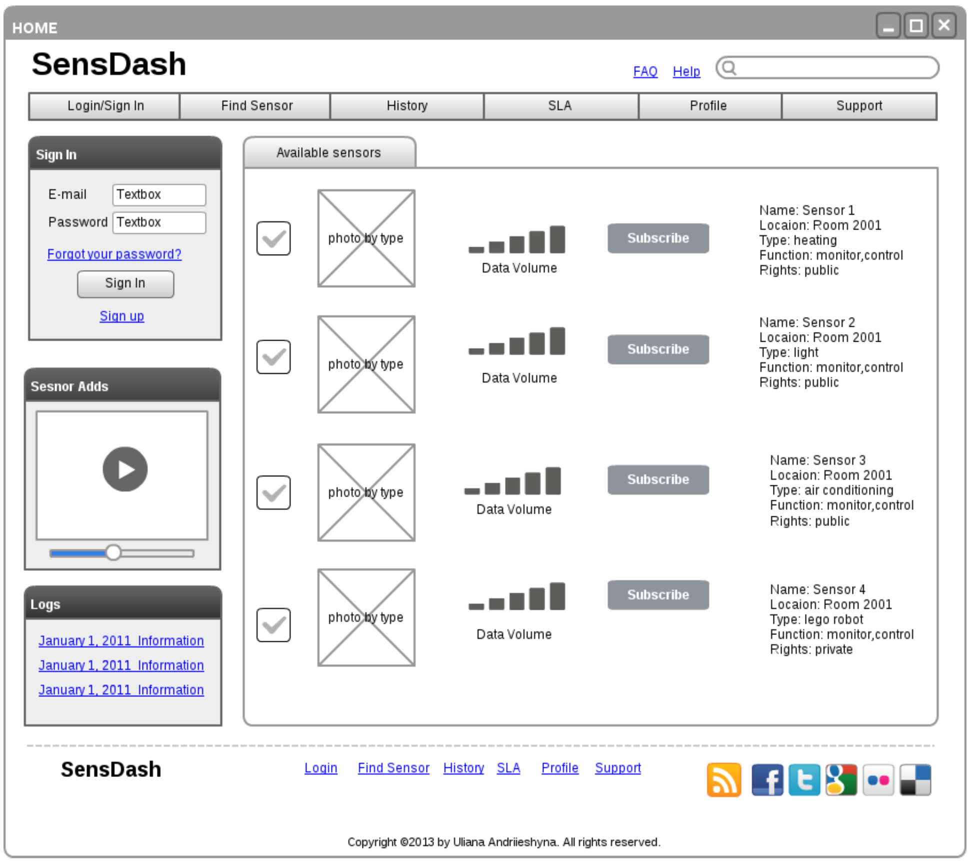
Figure 4.1: 3-tier Architecture

server or business logic. Giving data its own tier also improves scalability.
The three tiers architecture may seem similar to the model-view-controller (MVC) concept. However, topologically they are different. A fundamental rule in a three tier architecture is the client tier never communicates directly with the data tier; in a three-tier model all communication must pass through the middle tier. Conceptually the three-tier architecture is linear. However, the MVC architecture is triangular: the view sends updates to the controller, the controller updates the model, and the view gets updated directly from the model. From a historical perspective the three-tier architecture concept emerged in the 1990s from observations of distributed systems (e.g., web applications) where the client, middleware and data tiers ran on physically separate platforms. Today, MVC and similar model-view-presenter (MVP) are Separation of Concerns design patterns that apply exclusively to the presentation layer of a larger system. In simple scenarios MVC may represent the primary design of a system, reaching directly into the database; however, in most scenarios the Controller and Model in MVC have a loose dependency on either a Service or Data layer/tier. This is all about Client-Server architecture.
In the next section gives an detailed explaination about structure modules in 3-tier architecture.

4.2 Client Tier

The client tier or another name is presentation tier is a layer which users can access directly such as a web page by using browser. It is the first that user see. This tier consists logic and GUI by using which, user can communicate with sensor in a way that is presentd on Figure 4.2. Also Client Tier responsible to be adapted to any kind of mobile or desktop devices that user can use. Therefore as a cross-platforming approach was chosen web-based solution, where all communication flows through the browser.

Figure 4.2 presents simple content layout in which user can be interested in. It consists:



Name: Sensor 1
Locaion: Room 2001
Type: heating
Function: monitor,control
Rights: public

Name: Sensor 2
Locaion: Room 2001
Type: light
Function: monitor,control
Rights: public

Name: Sensor 3
Locaion: Room 2001
Type: air conditioning
Function: monitor,control
Rights: public

Name: Sensor 4
Locaion: Room 2001
Type: lego robot
Function: monitor,control
Rights: private

Copyright ©2013 by Uliana Andriieshyna. All rights reserved.

Figure 4.2: GUI Mockup

- Login form with user name and password fields. After user logged in, the system defines his/her rights and applies visibility rules according to credentials. User that have an admin rights receives an opportunity to control and manupulate sensors. Simple user will receive an opportunity to get a statistic and inforamtion from sensors.

- Sensor icon defines what is the current type of sensor, e.g. light, temperature, heating, robot lego, etc. In such a way user can easily, even in seconds, understand and catch what is the main fundtion of a sensor in the list.

- Availability or unavailability to see alive statitics. User can subscribe only to the available services. If some services become unavailable it will be automatically marked as inactive and after refreshing will be deleted from the list of available sensors and moved to a history tab.

- Data Volume icon shows what is the average data stream volume needed to retrieve sensor data(Kb/s). The dashboard should automatically adapt quality of streaming data to necessary and possible bandwidth of available connection(Wi-Fi, 3G, GPRS)

The common use case from a end-user point of view shown on Figure 4.3. User can use any type of mobile device and his favourite browser to receive informaion from the sensors by using web-site as a dashboard.

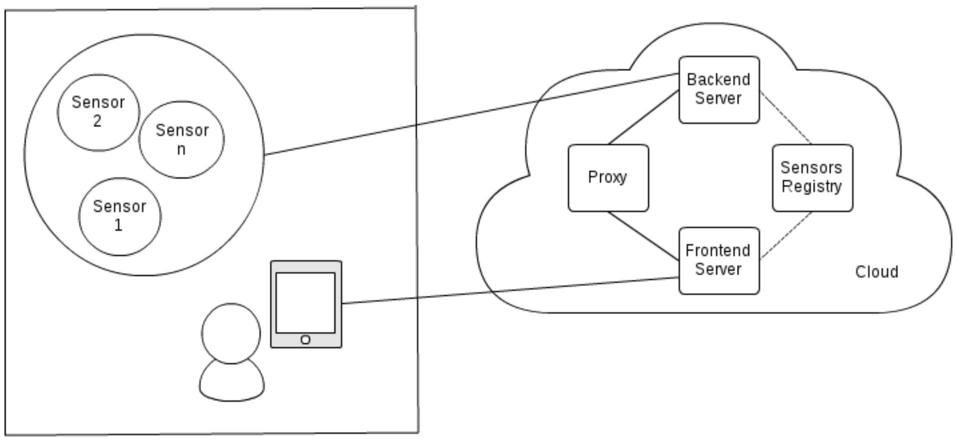


Figure 4.3: Use Case

4.3 Application Tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.
Application tier consists all logical modules as: Web server, Sensor registry, Proxy and Web-based Frontend. All these modules connects to each other as shpown on a Figure 4.4. Detailed information about functionality of every module described in sections above.

4.3.1 Web server

The primary function of a web server is to deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP). Pages delivered are most frequently HTML documents, which may include images, style sheets and scripts in addition to text content.
In proposed concetp Web server responsible for dynamic distribution of tasks between various modules of common system to serve efficiently static files. Such specific operation logic like authentication of user, registration of sensors and users are deligated to an external components such as Registry, Proxy and Frontend. these external components can be interchange without dependency to teh system itself.

4.3.2 Sensor Registry

Sensor Registry is a module resposible for keeping all required metadata of vailable sensors, for connecting arbitrary sensors. Frontend gets info from Registry about registered sensors and their availability. By using simple RESTful API, so that any Registry that implements

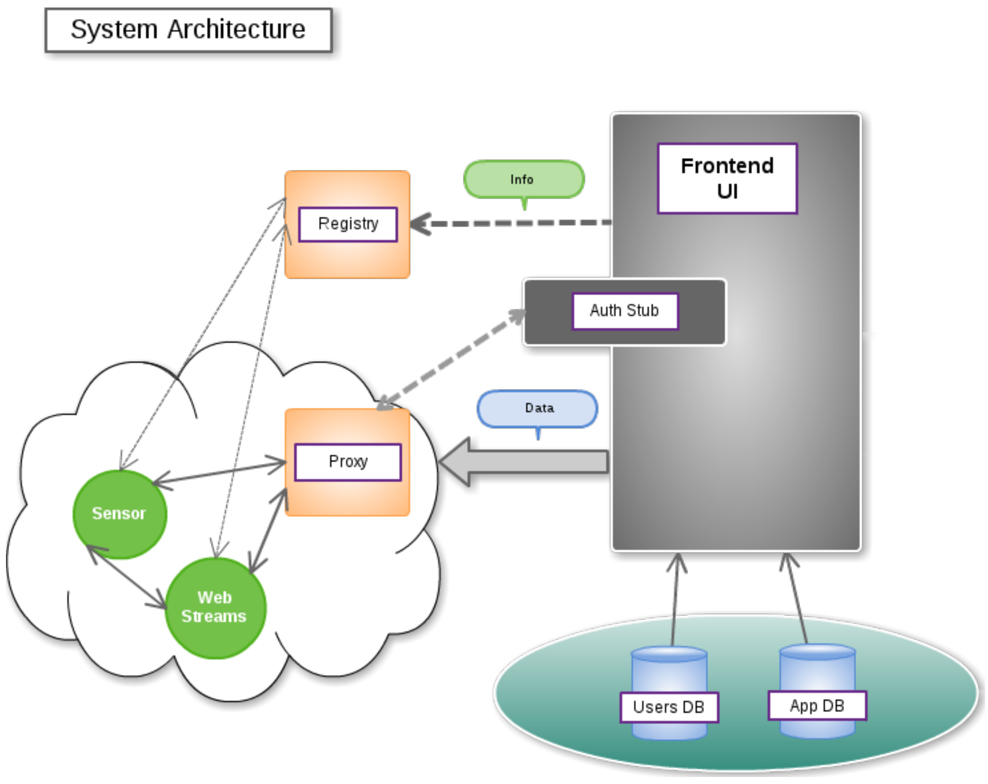


Figure 4.4: System Architecture

suggested API and returns valid JSON, which consits such info as: id of sensor, availability(true or false), SLA, necessary bandwidth for retrivng data, title and type. By using RESTful API, the concept provides a possibility to dynamically connect and disconnect different Registries, that are required in a different context of usage.

4.3.3 Proxy

Since concept of generic frontend is meant to support any type of sensor, regardless of its interface, Proxy is responsible for mapping interface of particular sensors data stream into JSOn message, delivered through XMPP protocol. Requirements to Proxy are:

- convert sensor data into particular JSON supporting scheme(specification*)
- implement XMPP protocol to provide exchange message with XMPP server
- get and parse sensor data

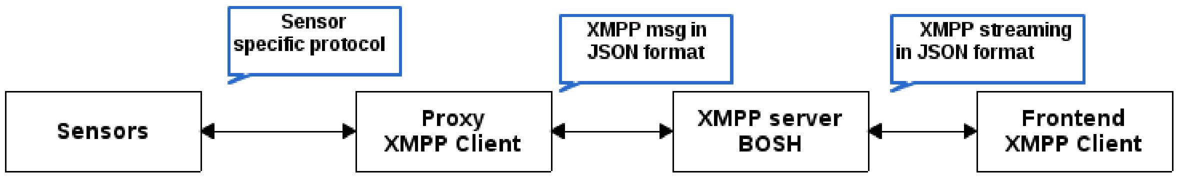


Figure 4.5: Protocol flow

4.3.4 Web-based Frontend

Interfaces

According to RESTful API and simple structure of registry data, communication between Frontend and Registry will done by using HTTP get request and JSON format.
According to requirements to retrieve streaming data from the sensors, it is necessary to use XMPP protocol[21]. XMPP powers a wide range of applications including instant messaging, multi-user chat, voice and video conferencing, collaborative spaces, real-time gaming, data synchronization, and even search. Although XMPP started its life as an open, standardized alternative to proprietary instant messaging systems like ICQ and AOL Instant Messenger, it has matured into an extremely robust protocol for all kinds of exciting creations.
The core of XMPP is the exchange of small, structured chunks of information. Like HTTP, XMPP is a client-server protocol, but it differs from HTTP by allowing either side to send data to the other asynchronously. XMPP connections are long lived, and data is pushed instead of pulled. Because of XMPP's differences, it provides an excellent companion protocol to HTTP. XMPP-powered web applications are to AJAX what AJAX was to the static web site; they are the next level of interactivity and dynamism. Where JavaScript and dynamic HTML have brought desktop application features to the web browser, XMPP brings new communications possibilities to the Web. XMPP has many common social web features built in, due to its instant messaging heritage. Contact lists and subscriptions create social graphs, presence updates help users keep track of who is doing what, and private messaging makes communication among users trivial. XMPP also has nearly 300 extensions, providing a broad and useful range of tools on which to build sophisticated applications.
XMPP, like all protocols, defines a format for moving data between two or more communicating entities. In XMPP's case, the entities are normally a client and a server, although it also allows for peer-to-peer communication between two servers or two clients. Many XMPP servers exist on the Internet, accessible to all, and form a federated network of interconnected systems. Data exchanged over XMPP is in XML, giving the communication a rich, extensible structure. Many modern protocols forgo the bandwidth savings of a binary encoding for the more practical feature of being human readable and therefore easily debugged. XMPP's choice to piggyback on XML means that it can take advantage of the large amount of knowledge and supporting software for dealing with XML. One major feature XMPP gets by using XML is XML's extensibility. It is extremely easy to add new features to the protocol that are both backward and forward compatible. This extensibility is put to great use in the more than 200 protocol extensions registered with the XMPP Standards Foundation and has provided developers with a rich and practically unlimited set of tools. XML is known primarily as a document format, but in XMPP, XML data is organized as a pair of streams, one stream for each direction of communication. Each XML stream consists of an opening element, followed by XMPP stanzas and other top-level elements, and then a closing element. Each XMPP stanza is a first-level child element of the stream with all its descendent elements and attributes. At the end of an XMPP connection, the two streams form a pair of valid XML documents. The Extensible Messaging and Presence Protocol (XMPP) is the IETF's formalization of the base XML streaming protocols for instant messaging and

presence developed within the Jabber community starting in 1999. This page provides a brief chronology of Jabber/XMPP technologies from the perspective of standardization[22].

- *Decentralization*
  The architecture of the XMPP network is similar to email; anyone can run their own XMPP server and there is no central master server.

- *Open standards*
  The Internet Engineering Task Force has formalized XMPP as an approved instant messaging and presence technology under the name of XMPP (the latest specifications are RFC 6120 and RFC 6121). No royalties are required to implement support of these specifications and their development is not tied to a single vendor.

- *History*
  XMPP technologies have been in use since 1999. Multiple implementations of the XMPP standards exist for clients, servers, components, and code libraries.

- *Security*
  XMPP servers can be isolated from the public XMPP network (e.g., on a company intranet), and strong security (via SASL and TLS) has been built into the core XMPP specifications.

- *Flexibility*
  Custom functionality can be built on top of XMPP; to maintain interoperability, common extensions are managed by the XMPP Standards Foundation. XMPP applications beyond IM include groupchat, network management, content syndication, collaboration tools, file sharing, gaming, remote systems control and monitoring, geolocation, middleware and cloud computing, VoIP and Identity services.

The XMPP network uses a client–server architecture (clients do not talk directly to one another). However, it is decentralized—by design, there is no central authoritative server, as there is with services such as AOL Instant Messenger or Windows Live Messenger. Some confusion often arises on this point as there is a public XMPP server being run at jabber.org, to which a large number of users subscribe. However, anyone may run their own XMPP server on their own domain. Every user on the network has a unique Jabber ID (usually abbreviated as JID). To avoid requiring a central server to maintain a list of IDs, the JID is structured like an email address with a username and a domain name (or IP address[16]) for the server where that user resides, separated by an at sign (@), such as username@example.com.

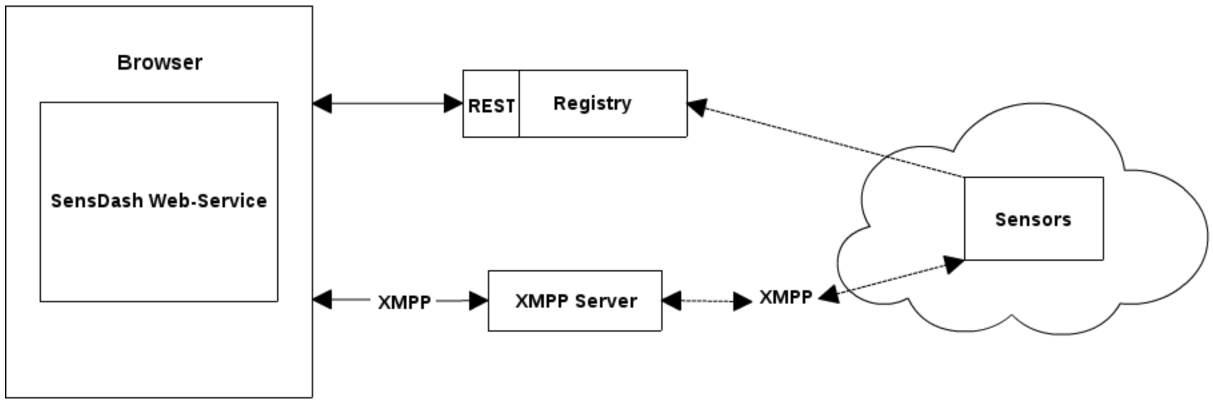


Figure 4.6: Interface

Authentication Stub

4.4 Data Tier

As was mentioned in section 4.1 Data Tier consists source of data that have to be retrieved by application tier to a client tier. Where is source of data is a data provided by sensors, in a specific type by using specific protocol, that will be nadled by Application Tier. The main focuse of this section determine possible characteristics of streamed data that can be retrieved by Client Tier in a lightweight scenario for mobile devices.
Streaming media is multimedia that is constantly received by and presented to an end-user while being delivered by a provider. Its verb form, "to stream", refers to the process of delivering media in this manner; the term refers to the delivery method of the medium rather than the medium itself. In general, media files can be delivered in one of three ways, via streaming, progressive download, or adaptive bitrate streaming. Each has its purpose.
Streaming involves delivering the media to the client via a server process using specific streaming protocols (such as XMPP). Video playing begins almost immediately, especially if the video file was encoded at a data rate similar to the effective bandwidth of the target viewer. Streaming video is also often not cached by the client so a local copy of the video is not held in its entirety on the client machine. While it is not impossible for an enterprising person to capture and hold a copy of the stream, it takes more effort than the casual viewer may be willing to take on. To adapt for the slowest common denominator in regard to end-user bandwidth, streaming videos are often encoded at lower quality and data rates.
Progressive download simply delivers a media file via traditional webserver technologies. The file begins playing on the client as soon as enough data has been buffered to provide a smooth uninterrupted viewing experience. Progressive downloaded files are easier to capture since an entire copy of the file is downloaded to the local device. Also, the quality of the file can be higher simply because a user on a slower connection will just have to wait longer for the viewing to begin.
Adaptive bitrate streaming is a kind of best of both worlds. As the name implies, adaptive bitrate is a streaming technology and generally requires a dedicated streaming server. In this case, media files are transcoded into multiple bitrates with the appropriate streaming

being delivered to the user based on their available bandwidth. Adaptive streaming servers can also dynamically change the bitrate as network conditions dictate[23].
This explanation shows that adaptive bitrate streaming is the most valuable and suitable for concept of a generic frontend. But it is necessary to go deeply in details to define limits and understanding of "good quality", "bad quality", "excellent quality". All three delivery methods are forms of Adaptive Bit Rate Streaming. This delivery method will have a massive impact on every aspect of Internet video delivery because it allows the stream to actually adapt the video experience to the quality of the network and the device's CPU.
In other words, the video stream can increase or decrease the bit rate and resolution of the video (its quality) in real time so that it's always streaming the best possible quality the available network connection can support. The better the network connection, the better the video image quality. The fact that the stream handles all of this complexity means the mobile video viewer doesn't have to do anything; everything is left to the stream and the player.
So how does this all work? To prep your video content for HLS, you start off with a high quality version of your video and encode multiple copies of it using MPEG-4 H.264. These copies are at various bit rates and resolutions ranging from lower quality renditions appropriate for slower 3G connections, up to extremely high quality renditions suitable for fast devices on fast networks. The renditions are then wrapped into MPEG-2 Transport Streams and chopped up into 10 second segments or chunks. It's these segments that are eventually streamed to an HTML5 Video Player on a mobile device, browser or set-top box, and because the player receives the video in 10 second chunks and can detect the quality of the network connection, it can switch to a higher or lower quality video segment every ten seconds if bandwidth conditions change.
Mobile platform such as iOS/Mobile Encoding supports at least two video types: 3GP + MPEG-4 for less sophisticated devices, and H.264 + MP4 for smartphones. One output video can cover all of smartphone users – iPhone/iPad/iPod, Android, and (for the most part) Blackberry too. Toss in PSP, PS3, and Xbox 360 for good measure. Mobile devices well using a handful of standard encoding profiles. Start with the Universal Smartphone Profile for wide compatibility; add in an Advanced Smartphone Profile version for the more advanced devices; and round out mobile list with a legacy profile for widest compatibility – either our Legacy Smartphone Profile (below), or even a 3GP video for even wider compatibility. The following defaults are the starting point for these profiles. Default these settings by default, but you can replicate them easily enough in whatever encoding tool you're using. Defaults: Video: H.264, Level 3.0, Baseline profile Audio: AAC, 1-2 channels

4.5 Summary

In this chapter, a first web-based concept for sensor streaming services is to be created. Along with it, a light-weight scenario service registry will be needed. Users should be able to explore not just services, but also the information provided by them, and eventually be led to advanced usage patterns such as the development of third-party applications to access the information data and real-time streams. When dealing with streamin data, delivering the best possible video experience is the ultimate goal for every web team. Proxy responsibiliies:

- 1
- 2
- 3

Registry responsibiliies:

- 1
- 2
- 3

Web-server responsibiliies:

- 1
- 2
- 3

Frontend responsibiliies:

- 1
- 2
- 3

Chapter 5

Implementation and Evaluation

The chapter contains practical part of the work, describing implementation of suggested prototype in the section 2.3. The prototype implements the major aspects proposed in the concept (chapter 4). The implementation consists the major aspects proposed in the concept, according to 3-tier architecture. Namely next components:

- **Client Tier** presents adaptive to different screens GUI, dynamically changed content and multy-user access
- **Application Tier** consists Apache Web-server, XMPP server and garantee appropriate interface of collaboration between iers via JSON format and defined structure
- **Data Tier** consists cookies for authorization

To make evaluations real, system will use data from the VICCI(Visual and Interactive Cyber-physical Systems Control and Integration) project at the Faculty of Computer Science of the Dresden University of Technology. The scope includes smart home environments and supporting people in the ambient assisted living. Also it was connected by using Data Hub as a Proxy, that is the part of Master Thesis of Luiz Alberto Borges, "Data Hub for Adaptive Data Services".

5.1 Develpment Environment

As a programming languages for implementation of this work jQuery[1] and HTML5 together with CSS3 are chosen. jQuery is a fast, small, well documented, easy and widely used and feature-rich JavaScript library. In addition it has such an important properties as: chaining, easy-to-use AJAX, event handlers, CSS selectors, pluins. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. It enables the project code to be portable over different platforms and provides opportunity for robust and effective development. The choice is dictated mostly by two aspects. On the one hand, the

[1]jQuery programming language, `http://jquery.com/`

system that is being developed is distributed by it's nature. On the other hand, jQuery has low entry barrier, and the code written in this language is extremely readable, laconic and understandable. These facts make further support of written code much easier for other developers that have an experience with any other JavaScript library.

Target	jQuery	Dojo	Prototype	YUI	ExtJS
License	MIT	BSD & AFL	MIT	BSD	GPL and Commercial
Size	32 KiB	41 kB	46–278 kB	31 kB	84–502 kB
Source language	JavaScript	JavaScript + HTML	JavaScript	Javascript + HTML + CSS	JavaScript
Grid	yes	yes	yes	-	yes
DOM wrapped	yes	yes	yes	no	yes
Other data retrieval	XML, HTML	XML, HTML, CSV, ATOM	-	yes	XML
DOM wrapped	yes	yes	yes	no	yes
Server push data retrieval	yes	yes	-	via Plugin	yes
GUI page layout	with Plugin	yes	yes	-	yes
Touch events	with Plugin	yes	yes	-	yes

Table 5.1: Comparison of JavaScript frameworks

Also the most important part is a version of browser support. jQuery[2], Dojo[3], Prototype[4] , YUI[5], ExtJS[6]

Target	jQuery	Dojo	Prototype	YUI	ExtJS
Chrome	1+	3	1+	-	10+
Opera	9+	10.50+	9.25+	10.0+	11+
Safari	3+	4	2.0.4+	4.0	4+
Mozilla Firefox	2+	3+	1.5+	3+	3.6+
Internet Explorer	6+	6+	6+	6+	6+

Table 5.2: Browser Support

[2]jQuery browser support, http://jquery.com/browser-support/
[3]Dojo browser support,http://livedocs.dojotoolkit.org/releasenotes/1.4
[4]Prototype browser support, http://prototypejs.org/doc/latest/Prototype/Browser/index.html
[5]YUI browser support, http://yuilibrary.com/yui/environments/
[6]ExtJS browser support, http://www.sencha.com/products/extjs/

5.2 Web-based Framework Analysis

- **Bootstrap**
  Bootstrap is the most popular and widely used framework, nowadays. It's a beautiful, intuitive and powerful web design kit for creating cross browser, consistent and good looking interfaces. It offers many of the popular UI components with a plain-yet-elegant style, a grid system and JavaScript plugins for common scenarios.

  It consists of four main parts: Scaffolding – global styles, responsive 12-column grids and layouts. Bear in mind that Bootstrap doesn't include responsive features by default. If design needs to be responsive this functionality have to be done manually. Base CSS – this includes fundamental HTML elements like tables, forms, buttons, and images, styled and enhanced with extensible classes. Components – collection of reusable components like dropdowns, button groups, navigation controls (tabs, pills, lists, breadcrumbs, pagination), thumbnails, progress bars, media objects, and more. JavaScript – jQuery plugins which bring the above components to life, plus transitions, modals, tool tips, popovers, scrollspy (for automatically updating nav targets based on scroll position), carousel, typeahead (a fast and fully-featured autocomplete library), affix navigation, and more.

- **Foundation**
  Foundation is a powerful, feature-rich, responsive front-end framework. With Foundation user can quickly prototype and build websites or apps that work on any kind of device, with tons of included layout constructs, elements and best practices. It's built with mobile first in mind, utilitizes semantic features, and uses Zepto instead of jQuery in order to brings better user experience and faster performance.
  Foundation has a 12-column flexible, nestable grid powerful enough to create rapidly multi-device layouts. In terms of features it provides many. There are styles for typography, buttons, forms, and various navigation controls. Many useful CSS components are provided like panels, pricing tables, progress bars, tables, thumbnails, and flex video that can scale properly your video on any device. And, of course, JavaScript plugins including dropdowns, joyride (a simple and easy website tour), magellan (a sticky navigation that indicates where is the user on the page), orbit (a responsive image slider with touch support), reveal (for creating modal dialogs or pop-up windows), sections (a powerful replacement for traditional accordions and tabs), and tooltips.

- **GroundworkCSS**
  GroundworkCSS is a new, fresh addition to the front-end frameworks family. It's a fully responsive HTML5, CSS and JavaScript toolkit built with the power of Sass and Compass which gives the ability to rapidly prototype and build websites and apps that work on virtually any device.

  It offers an extremely flexible, nestable, fraction-based, fluid grid system that makes creating any layout possible. GroundworkCSS has some really expressive features like tablets and mobile grids which maintain the grid column structure instead of collapsing the grid columns into individual rows when the viewport is below 768 or 480 pixels wide. Another cool feature is a jQuery ResponsiveText plugin which allows to have dynamically sized text that adapts to the width of the viewport: extremely useful for scalable headlines and building responsive tables. The framework includes a rich set of UI components like tabs, responsive data tables, buttons, forms, responsive navigation controls, tiles (a beautiful alternative to radio buttons and other boring standard form elements), tooltips, modals, Cycle2(a powerful, responsive content slider), and many more useful elements and helpers. It also offers a nice set of vector social icons and a full suite of pictographic icons included in FontAwesome. To see the framework in action user can use the resizer at the top center of the browser window. This way user can test the responsiveness of the components against different sizes and viewports while exploring the framework's features. GroundworkCSS is very well documented with many examples, and to get user started quickly the framework also provides several responsive templates. The only thing as a weakness is the missing of a way to customize download.

- **Gumby**
  Gumby is simple, flexible, and robust front-end framework built with Sass and Compass.

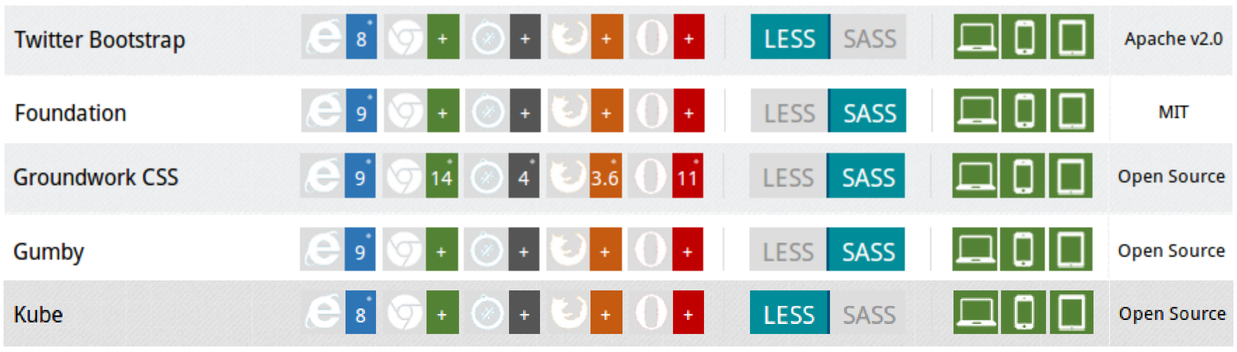
  Its fluid-fixed layout self-optimizes the content for desktop and mobile resolutions. It support multiple types of grids, including nested ones, with different column variations. Gumby has two PSD templates that get user started designing on 12 and 16 column grid systems. The framework offers feature-rich UI Kit which includes buttons, forms, mobile navigation, tabs, skip links, toggles and switches, drawers, responsive images, retina images, and more. Following the latest design trends the UI elements have Metro style flat design but can use Pretty style with gradient design too, or to mix up both styles. An awesome set of responsive, resolution independent Entypo icons, is completely integrated into the Gumby Framework. Gumby has also a very good customizer with color pickers which helps to build your custom download with ease.

- **Kube**
  Lastly, if user need a solid, yet simple base without needless complexity and extras, for your new project, Kube can be the right choice. Kube is a minimal, responsive and adaptive framework with no imposed styling which gives to user the freedom to create. It offers basic styles for grids, forms, typography, tables, buttons, navigation, and other stuff like links or images.

  The framework contains one compact CSS file for building responsive layouts with ease and two JS files for implementing tabs and buttons in your designs. If user is looking for maximum flexibility and customization, user can download developer version which includes LESS files, with variables, mixins and modules.

5.3 Data Flow Model

Data Flow model describes the



Twitter Bootstrap	8	+	+	+	+	**LESS** SASS	Apache v2.0
Foundation	9	+	+	+	+	LESS **SASS**	MIT
Groundwork CSS	9	14	4	3.6	11	LESS **SASS**	Open Source
Gumby	9	+	+	+	+	LESS **SASS**	Open Source
Kube	8	+	+	+	+	**LESS** SASS	Open Source

Figure 5.1: Framework Comparison[7]

5.3.1 XMPP implementation

XMPP Stanzas

Work is accomplished in XMPP by the sending and receiving of XMPP stanzas over an XMPP stream. Three basic stanzas make up the core XMPP toolset. These stanzas are <presence> , <message> , and <iq> . Each type of stanza has its place and purpose, and by composing the right kinds of quantities of these stanzas, sophisticated behaviors can be achieved. Remember that an XMPP stream is a set of two XML documents, one for each direction of communication. These documents have a root <stream:stream> element. The children of this <stream:stream> element consist of routable stanzas and stream related top-level children. Each stanza is an XML element, including its children. The end points of XMPP communication process input and generate output on a stanza-by-stanza basis.The following example shows a simplified and short XMPP session:: In this example, Elizabeth

```
<stream:stream>
  <iq type='get'>
    <query xmlns='jabber:iq:roster'/>
  </iq>

  <presence/>

  <message to='darcy@pemberley.lit'
           from='elizabaeth@longbourn.lit/ballroom'
           type='chat'>
    <body>I cannot talk of books in a ball-room; my head is always full of
      something else.</body>
  </message>

  <presence type='unavailable'/>
</stream:stream>
```

Figure 5.2: Stanzas example

created an XMPP stream by sending the opening <stream:stream> tag. With the stream open, she sent her first stanza, an <iq> element. This <iq> element requested Elizabeth's roster, the list of all her stored contacts. Next, she notified the server that she was online and available with a <presence> stanza. After noticing that Mr. Darcy was online, she sent him a short <message> stanza, thwarting his attempt at small talk. Finally, Elizabeth sent another <presence> stanza to inform the server she was unavailable and closed the <stream:stream> element, ending the session.

Server

The set of XMPP servers that can mutually communicate forms an XMPP network. The set of public XMPP servers forms the global, federated XMPP network. If a server does not speak the server-to-server protocol, it becomes an island, unable to communicate with external servers. An XMPP server will usually allow users to connect to it. It is, however, also possible to write applications or services that speak the server-to-server protocol directly in order to improve efficiency by eliminating routing overhead. Anyone can run an XMPP server, and full-featured servers are available for nearly every platform. Ejabberd, Openfire, and Tigase are three popular open source choices that will work on Windows, Mac OS X, or Linux systems. Several commercial XMPP servers are available as well, including M-Link and Jabber XCP.

Connection

Before any stanzas are sent, an XMPP stream is necessary. Before an XMPP stream can exist, a connection must be made to an XMPP server. XMPP includes some sophisticated support for establishing connections to the right servers. Typically clients and servers utilize the domain name system (DNS) to resolve a server's domain name into an address they can connect to. Email services in particular use mail exchange (MX) records to provide a list of servers that handle mail for a given domain so that one well-known server address does not have to handle every service. Email, being an early Internet application, got special treatment in DNS. These days, service records (SRV) are used to provide a similar function for arbitrary services. The first thing an XMPP client or server does when connecting to another XMPP server is to query the appropriate SRV record at the server's domain. The response may include multiple SRV records, which can be used to load balance connections across multiple servers. If an appropriate SRV record cannot be found, the application tries to connect to the given domain directly as a fallback. Most libraries also allow you to specify a server to connect explicitly.

Long Polling

Even with AJAX, data was still being requested, or polled, at timed intervals. Servers can be crippled if too many clients poll too fast.However, to get quick updates, the polling interval needs to be quite small; the lowest latency possible is the length of the polling interval. Another issue with polling is that most poll requests do not receive new data. In order to see changes within a reasonable time frame of when they occur, the polling interval must be quite short, but the actual data may not change very often. For example, if there is new data ready on the server, the server answers immediately. If there is not new data, the server keeps the connection open, holding any reply. Once new data arrives, it finally responds to the request. If no new data arrives after some period of time, the server can send back an empty reply, so as not to hold too many open connections at once. Once a request is returned, the client immediately sends a new one, and the whole process starts over. Because each polling request is potentially open for a long period of time, this technique is called long polling. It has many advantages over normal polling.

5.4 Browser Support

In past years a Flash-based media player in more than sufficient for streaming on the Web and this technology is still necessary to support legacy browsers. But thankfully modern standards have advanced and the inclusion of HTML5 video opens doors for dozens of new opportunities.

In this guide I'd like to offer an introduction to HTML5 video for the Web. It will take some practice to understand the native in-browser player and all its functionality. When you're working with a flash video player it's all too common to associate all video formats in .flv. While this does work, most flv files cannot retain quality anywhere near the more advanced file formats/codecs. There are 3 important video types which are supported by HTML5: MP4, WebM, and Ogg/Ogv. The MPEG-4 file type is generally encoded in H.264 which allows for playback in third party Flash players. This means you don't need to keep a .flv video copy to support a fallback method! WebM and Ogg are two much newer file types related to HTML5 video. Ogg uses Theora encoding which is based on the open-source standard audio file format. These can be saved with a .ogg or .ogv extension. So which of these file types do you need for your website? Well ideally all 3 would be great as they provide the full support spectrum. Yet this isn't realistic, and in fact, you can cover all the bases with only two of them. Here is a breakdown of what works for each browser:

Mozilla Firefox – WebM, Ogg Google Chrome – WebM, Ogg Opera – WebM, Ogg Safari – MP4 Internet Explorer 9 – MP4 Internet Explorer 6-8 – No HTML5, Flash Only! Most flash video players will support MP4 files as long as they're encoded in H.264. As such, each of these browsers will embed MP4+Flash as a final resort. This means you only need to create two different video formats to support all browsers. MP4 for Safari/IE9 and a choice between WebM or Ogg for the rest.

5.5 Database Model

5.6 Use Cases

5.6.1 Frontend

5.6.2 3-tier Architecture in Software Projection

5.6.3 Use Cases Realization

5.6.4 Evaluation

5.6.5 Summary

Chapter 6

Conclusion and Outlook

6.1 Conclusion

The chapter summarizes the presented thesis providing an overview of each chapter and describing the achievement of the thesis's goals defined in the section 1.2. At the end of the chapter suggestions for the future work are made. The presented thesis consists of six chapters including the current chapter. The chapter 1 Introduction states the motivation (section 1.1) of the work arguing the necessity of creation of generic frontend. The main research questions and the goals that have to be achieved in the thesis are described in the section 1.2. The description of the thesis's structure (section 1.3) completes the first chapter.

6.2 Achieved Goals

6.3 Future work

To conclude the thesis, suggestions for the future work are made. These suggestions are devoted to improve either the developed concept or the current implementation.
Visualization and interaction metaphor for the introduced access control
Enhanced user interface for application part
User interface to support the introduced dynamic composition

6.3.1 Conceptual Aspects

This subsection describes possible improvements on the concept level.

6.4 Addressed research questions

SLA

Differentiation of SLA depending on user type/rights. Typization of SLA and filtering . Easy enhancement in case of SLA changes, then come in to a picture next questions: How to re-sign SLA with user, how to nitfy about changes frontend and user itself, that have already once accepted it, how will system react in case user will not accept new SLA. How can frontend predefine all future changes that can appear according to SLA changes?

Privacy and security

Cookies stored on a mobile device can be easily be sotollen via hacking attack on browser or account in browser. To avoid this should be in detail researched another possibility to encrypt data or to make authorisation process more secure.

Introducing the interaction awareness

In principle, when a user open first time an application, different information can be interesting to him. Therefore, to introduce a convenient awareness, the following research questions should be investigated:

- What kind of interaction awareness information end users are really need?
- Does a user want to configure the received awareness information?
- In case of providing a configuration, what an appropriate visualization and interaction metaphor can be provided?

Integration with other application via Internet

The best opportunity to make application widely-used is to enhance list of supported hardware and software sensors. But a lot of system already propose their own sensors and corresponding app to it. How possible will be to create additional module for enhancement that will play arole of retranslator or proxy between two different systems?

Resource limitations: energy, bandwidth and computation

Since mobile devices face internal and external resource limitations, the need of differentiation of connection properties is important. For example, location data can be provided using GPS, WiFi, and GSM, with decreasing levels of accuracy. Compared to WiFi and GSM, continuous GPS location sampling drains the battery faster. One approach to this problem uses low duty cycling to reduce energy consumption of high-quality sensors (i.e., GPS), and alternates between high- and low-quality sensors depending on the energy levels of the device (e.g., sample WiFi often when battery level is less than 70 percent). This approach trades off ata quality and accuracy for energy.

6.5 Implementation Aspects

Due to the provided implemented background that supports statically defined applications and due to the lack of the time to extend this basic implementation, thedeveloped concept has been implemented partly. The future implementation work can be split up into the following tasks:

List of Figures

List of Tables

Bibliography

[1] S. Suakanto, S.H. Supangkat, Suhardi, and R. Saragih. Smart city dashboard for integrating various data of sensor networks. In *ICT for Smart Society (ICISS), 2013 International Conference on*, pages 1–5, 2013.

[2] Schuster Schill Ackermann Ameling Bendel, Springer. A service infrastructure for the internet of things based on xmpp. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 385–388. IEEE, 2013.

[3] Xianfeng Song, Chaoliang Wang, Masakazu Kagawa, and Venkatesh Raghavan. Real-time monitoring portal for urban environment using sensor web technology. In *Geoinformatics, 2010 18th International Conference on*, pages 1–5. IEEE, 2010.

[4] Michael Eggert, Roger Häußling, Martin Henze, Lars Hermerschmidt, René Hummen, Daniel Kerpen, Antonio Navarro Pérez, Bernhard Rumpe, Dirk Thißen, and Klaus Wehrle. Sensorcloud: Towards the interdisciplinary development of a trustworthy platform for globally interconnected sensors and actuators. *arXiv preprint arXiv:1310.6542*, 2013.

[5] Wikipedia. Front and back ends — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Front_and_back_ends&oldid=572495173`, 2013. [Online; accessed 15-November-2013].

[6] Suman Nath, Jie Liu, and Feng Zhao. Sensormap for wide-area sensor webs. *Computer*, 40(7):90–93, 2007.

[7] Xiaogang Yang, Wenzhan Song, and Debraj De. Liveweb: A sensorweb portal for sensing the world in real-time. *Tsinghua Science & Technology*, 16(5):491–504, 2011.

[8] Josef Spillner, Marius Feldmann, Iris Braun, Thomas Springer, and Alexander Schill. Ad-hoc usage of web services with dynvoker. In *Towards a Service-Based Internet*, pages 208–219. Springer, 2008.

[9] Inc. Open Geospatial Consortium. A cloud design for user-controlled storage and processing of sensor data. In *Sensor Web Enablement Architecture*, pages 13–30. Open Geospatial Consortium, Inc., 2008.

[10] Institute for Software and Multimedia-Technology Technical University of Dresden. Vicci, 2012.

[11] M. Franke, C. Seidl, and T. Schlegel. A seamless integration, semantic middleware for cyber-physical systems. In *Networking, Sensing and Control (ICNSC), 2013 10th IEEE International Conference on*, pages 627–632, 2013.

[12] René Hummen, Martin Henze, Daniel Catrein, and Klaus Wehrle. A cloud design for user-controlled storage and processing of sensor data. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 232–240. IEEE, 2012.

[13] Josef Spillner, Johannes Schad, and Stephan Zepezauer. Personal and federated cloud management cockpit. *Praxis der Informationsverarbeitung und Kommunikation*, 36(1):44–44, 2013.

[14] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. big'web services: making the right architectural decision. In *Proceedings of the 17th international conference on World Wide Web*, pages 805–814. ACM, 2008.

[15] Daniel Su Kuen Seong. Usability guidelines for designing mobile learning portals. In *Proceedings of the 3rd international conference on Mobile technology, applications & systems*, page 25. ACM, 2006.

[16] Michael Michael Thomas Bolin. *End-user programming for the web.* PhD thesis, Massachusetts Institute of Technology, 2005.

[17] Ian Hickson and David Hyatt. Html5: A vocabulary and associated apis for html and xhtml. *W3C Working Draft edition*, 2011.

[18] Gordon Baxter. Human factors and ergonomics in consumer product design: methods and techniques. *Ergonomics*, 55(9):1124–1125, 2012.

[19] JAKOB NIELSEN. Usability 101: Introduction to usability, 2012.

[20] U.S. Department of Health and Human Services. Visual design, 2013.

[21] Jack Moffit. *Professional XMPP programming with JavaScript and jQuery.* Wiley Publishing, Inc., Indianapolis, Indiana, 2010.

[22] XMPP Standards Foundation. Xmpp extensions, 1999.

[23] Irma Syarlina Hj Che Ilias, Sri Banu Munisamy, and Nandang Azryman Ab Rahman. A study of video performance analysis between flash video and html 5 video. In *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, page 30. ACM, 2013.