Database driven user friendly web application using Ajax

David Jonsson, dit03djn@cs.umu.se

June 12, 2009
Master's Thesis in Computing Science, 30 ECTS credits
Supervisor at CS-UmU: Jerry Eriksson
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

Given a set of different existing systems, a single web application has been constructed. Acting as a replacement for the systems the web application has been built up as a multiple page Ajax enabled web application. Using Ajax as technique to enhance the usability along with performance criterion such as response time for end users.

The web application follows the rules set by W3C and validates as XHTML 1.0 and CSS 2.1. Using style sheets and logic layers within the application makes it flexible for future upgrades.

Data is being stored in a SQL database. The web application will communicate with the SQL along with an existing database in order to fetch and visualize data on the GUI served by the web application.

Software used has been Microsoft Visual Web Developer 2008 Express Edition and Microsoft SQL Server 2005 Management Studio Express Edition for database creation. Importing the ASP.NET Ajax extensions framework into MS Visual Web Developer 2008 Express Edition has made it possible to add Ajax features into the web application.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the past web applications have been considered to be deficient in interactivity, responsiveness and richness when compared with traditional desktop applications [8, 16].

RIAs (Rich Internet Applications), Web 2.0 and Ajax are three popular buzz words currently used in software development. Ajax, being one of the most popular platforms along with Macromedia Flash or Java used when creating RIAs. What all RIAs have in common despite the use of platform is that they are able to deliver web applications with interactivity, responsiveness and a richness almost as high or equal to current desktop applications [16].

A web application is an application that has the ability to run on a web browser over a network, most commonly on the Internet but also over an Intranet. For the web application to execute properly it needs to follow a browser supported language (such as HTML, JavaScript, Java etc.).

One of the key benefits of switching from common desktop applications to web based ones is that it becomes easier to update and maintain the software. Instead of having to install the updated application on several computers, a single update can be made to a server. Saving a lot of time and effort is possibly one of the main reasons why web applications have grown in popularity [18].

The report will cover some basics behind web site design, how web sites can become usable and metrics that can be used when evaluating web site usability. It also covers some guidelines on how to develop web sites with regard to accessibility. It will give the reader an in depth review on how the Ajax technology works alongside its current competitors and its advantages/disadvantages compared to other technologies.

The results section covers the creation of the web application and the database. Following this the report will conclude with a discussion regarding a conclusion and final thoughts about the research along with future recommendations.

Chapter 2

Problem Description

The Danish company LNeurocom A/S is a call center that strives to improve their quality and performance. A certification can be seen as a tool helping them implement processes and monitoring systems. With the tool LNeurocom can become a more attractive partner, not only for existing but also for new customers. To be able to fulfill this certification, data needs to be accessed from several different sources in order to use it together and visualize it in a user friendly way. As an essential tool to improve monitoring performance counters and thus help in the process of becoming certified, they have decided that an SQL (Structured Query Language) database needs to be made. Data from the SQL along with an existing Informix based CMS (Call Management System) database needs to be visualized on top of a Graphical User Interface (GUI). A user friendly web application is required, acting as a GUI for the tool. The GUI must have the ability to communicate with both the SQL and Informix databases. The web applications main tasks will be to act as an input, serving insertions, deletions and updates along with retrieving and visualising data in order for them to do further analyzing. Limitations in the GUI cases of user interactions are based upon users' rights and specific roles. Currently LNeurocom have different systems, making it more difficult to combine and analyze data. Their intentions are that the combination of an SQL database along with the existing Informix based CMS database will ease the use and make data storage in different formats such as Excel sheets and MS Access databases unnecessary. A single web application within the company is ideal, with the motivation that all employees (regardless of position) are able to use it in an efficient way. Their existing Avaya system comes with the Informix database where statistics regarding all their incoming and outgoing telephony traffic is stored. Other data such as human resource information, company revenues etc. is stored in different ways making it harder to collect and visualize. The creation of the SQL database and web application is a solution to centralize the collection and monitory performance counters considered important to LNeurocom's continued efforts in improving their efficiency.

2.1 Goals

The goal of the master thesis project is to develop the user friendly web application and a database which will act as an important tool for monitoring their current performance, improvement areas and actual improvements.

The web application needs to be fully compliant with Microsoft Internet Explorer 7.

The use of CSS (Cascading Style Sheets) are also a demand where these have to follow the style sheet rules appointed out by the W3C standards. Automatic data collection will take place daily (overnight, due to less network traffic and work load). The web application will be accessed by anyone within the company making it an internal web application. The use of Microsoft Windows Login Authentication will determine access rights in terms of different user roles.

As mentioned earlier GUI limitations in terms of features in the application will be based upon user's specific role. To summarize the thesis project, goals will be listed below in an unordered list:

- Create an SQL database.
- Make a user friendly web application.
- Communication from the web application with both the new database and an existing one.
- Make a web application with a GUI that has the ability to dynamically change depending on user rights and roles.
- Automatic data collection for the database.
- Write an in-depth study on Ajax, consisting of its pros and cons along with other existing techniques.
- Based on previous studies, an appraisal of whether or not it is useful to utilize Ajax or similar techniques on the current web application.

2.2 Methods

LNeurocom base its infrastructure on Microsoft platforms and technologies wherever possible, promoting that the web application must be fully functional with the web browser MS Internet Explorer 7 and its earlier versions. LNeurocom encouraged the use of as much software as possible available free of use often referred to as freeware. Suggestions were made which ended up in using Microsoft Visual Web Developer 2008 Express Edition as software for designing the web application along with Microsoft SQL Server 2005 Management Studio Express Edition for the creation of the SQL database. With prior knowledge in Java and C-sharp the obvious choice of programming language in Visual Web Developer was C-sharp.
A web application with enhanced user friendliness and usability might need further technology in practice. Discussions regarding choice of technology ended up in Ajax as a strong alternative technology used within the web application. With little or no prior knowledge about Ajax and how it works, it seemed natural to include this as my choice of in depth study. This in-depth study of Ajax will include arguments for and against it and the existing technologies deemed as competitors.

2.3 Related Work

Samuel Nyman has conducted a master thesis aimed at creating a web application using Ajax commissioned by ICA AB. Their existing application apparently had functionality

but also some usability issues. Benefiting from an existing application he upgraded it further, using Ajax to solve the issues regarding functionality and usability. According to Nyman the major disadvantage was considered to be the non-standards used within the application of Ajax. All the characteristic technologies used in Ajax to bind it together were not standardized during his period of time, creating issues such as browsers acting differently when used. Concluding that future web applications could perform in a more advanced way this shows that earlier attempts at development would have been considered difficult or even impossible [18].

Tatiana Braescu shows that implementing an interactive web application aimed at furniture products by utilizing Ajax technology within the application gave more users information regarding products than the previous solution. By using Ajax on the principal application it had also benefited the time of information being retrieved and updated data [2].

In a bachelor's thesis named "Improving Internet systems by using AJAX techniques", Johan Petterson and Jonas Kristiansson investigated the possibility of Ajax being able to further improve usability and performance in the aspect of time for end users. Investigation showed that both aspects were successful. Giving the application not only a higher degree of usability, but also acting more efficient in performance aspects such as time [11].

Chapter 3

Designing Web Sites

When designing web sites it is important that business objectives match with the needs of the intended users. Web site design can therefore often be considered a user centered process. The intended users will have expectations and needs, which if not met by the site, the provider of the service might risk lack of credibility and a big chance of site failure. Nigel Bevan also refers to sites having the need to meet "quality of use" [1].
The design process will undergo several stages during the web site creation. The web style guide presents six major stages that most web sites will experience. Site definition and planning is considered to be the first stage and essential in the process of creating successful web sites. This is where all involved in the design process need to define the content, what functionalities are desired and which technologies are needed in order to achieve the functionalities. All of which will be required to fulfill the end users expectations.

The site also needs to appoint a site editor. Without appointing a site editor, the editorial responsibilities of the site will most likely end up in failure. With a maintenance plan defined the responsibilities are no longer loose and uncertain.

Information architecture is another stage and this is where the content is drafted. With simple prototypes that consist of small parts, it will be easy to adjust the design. The prototypes are good for two reasons; Firstly to test navigation and set up the user interface and secondly it will also help to clarify how to map graphics design and how navigating corresponds to the graphic. A key to good prototyping is to stay flexible to changes and new ideas.

Site design is the third step and where all graphics and page related layout is set. After site design the stage where the empty pages being filled with content follows, this referred to as the site construction stage. The implementation takes place here as well and when all pages are constructed with necessary database and programming comes the time for beta testing. A good way of conducting beta testing is to let users not involved in the design process become beta testers. As "fresh" users increase the discovery of faults or issues previously unnoticed and will also be able to make comments and criticisms regarded as less prejudicial or more open minded.

Finally the sixth stage called tracking, evaluation and maintenance. Most web servers have the possibility to keep track of users within the site. In storing log files a lot of useful information can be gathered about the end users site experiences. It can for instance keep track of which browser the user has, along with time of visit, for how long and number of times spent on specific pages. Keeping the records and log files can

therefore be part of determining the rate of success for a web site. During the end of the process the web site needs to undergo evaluation and maintenance/management in order to adopt to changes, and to ensure to meet the ever evolving and changing of users needs and expectations [13, 1].

3.1 Web site usability

Usability towards the end user is essential not only on Web sites, but also on other systems such as information systems. A system without the appropriate responsiveness towards the end user could end up in failure. Studies have shown that sites exhibiting usability higher than the average have gained better performance, see [10]. Information systems have design principles containing five key elements defined prior to the popularity of the Web. These five key elements can be considered as usable when trying to define design principles for Web sites as well and includes [19]:

- **Consistency of the interface** Consistency of navigational tools such as buttons and links are important. Without consistency the end user might be disoriented, whilst navigating a Web site.

- **Response time** The response time is the time it takes for a system to execute an action taken by an end user. Generally regarded the lower the better.

- **Mapping and metaphors** Donald A. Norman describes mapping as the relationship between two things [17]. One example is a link on a Web site. The end user needs to identify the link, and take appropriate action in order to navigate correctly. Action in this case being moving the mouse pointer to the specific link, clicking once will then trigger the browser to relocate to the link url. The metaphor in the link example being the underlining and blue color of the text making it easier for the users to interpret text links.

- **Interaction styles** Interaction styles are considered the messages a system sends to the end user upon an action taken.

- **Multimedia and audiovisual** Within system design different degrees of multimedia and audiovisual can be implemented.

Nielsen and Schneidermann have extended the information systems design principles to fit the Web more accordingly with [19]:

- **Navigation** Navigating within a Web site should come naturally and impose no major difficulties.

- **Response time** The download time of different objects, such as pictures etc. should be kept at a minimum.

- **Credibility** Credibility is regarding the relevance of the end users.

- **Content** The quality of the content.

3.2 Web content accessibility

With a very large amount of people using the Internet every day, the need for accessibility guidelines might prove invaluable for designers when creating web sites. Accessibility guidelines should not only be taken into consideration for those suffering from physical disabilities such as color blindness, hearing problems etc. Accessibility guidelines might also provide "regular" users a higher extent of access.

Limitations in terms of users are in a noisy environment, making it more difficult to comprehend or hear certain auditory feedback. Users currently residing in an environment where luminance is low could be another limiting factor. By following the Web Content Guidelines 1.0 can therefore be one step closer to making web sites that are more comprehensible for all potential users. The itemized list below will highlight some useful steps in order to strive for higher accessibility [3].

- **Provide equivalent alternatives to auditory and visual content.** The content should be presented in such a way that even users without auditory or visual perception is able to comprehend.
- **Do not rely on color alone.** Colors can be a good way of presenting something in a specific way. Nevertheless, disabling the possibility of colors should gracefully degrade a site without any large implications. Although almost every device with a display now manufactured manages to display high depths of colors, there are still some that might not.
- **Use markup and style sheets in a proper manner.** Improper markup and misuse of style sheets might confuse users relying on speech based web browsers. By using standard markup language and proper style sheets the user no matter what software used will easier access different pages.
- **Clarify natural language.** Clarifying the language will ease the use of translations, abbreviations etc. when translated from one language to another.
- **Create tables that transform gracefully.** Users residing on screen readers might get confused if tables are used to express other than tabular data. Confusion might especially arise if tables are being used to specify components and their placement.
- **Ensure that pages featuring new technologies transform gracefully.** If Ajax, Flash, Java or similar technology is being used within a web site it is essential that it has the ability to degrade in a graceful manner. A user with a browser not supporting e.g. JavaScript will most likely fail to access an Ajax enabled web site where no consideration to graceful transformation has been applied.
- **Ensure user control of time sensitive content changes.** Users with disabilities might have difficulties comprehending a certain time sensitive user control. It could be anything from moving text to certain flashing effects. By providing the user a possibility to change the duration in time, if not disable the content change itself will ease the use for those.
- **Ensure direct accessibility of embedded user interfaces.** Create a user interface that is not specified to work with a specific device. With device independence the user interface has a greater access rate.

- **Design for device-independence.** Design with consideration that all or as many devices as possible can work satisfyingly within the environment. Devices include mouse, keyboard and voice etc.
- **Use interim solutions.** Make sure that interfaces will work in environments that are no longer being used in a large scale. Old browser support is an essential part of accessibility.
- **Use W3C technologies and guidelines.** By using W3C guidelines the user is assured no further plug-ins or stand alone applications are needed in order to be viewed correctly.
- **Provide context and orientation information.** Providing context or orientation information on parts where pages can be seen as complex might produce a better comprehension for users both with and without disabilities.
- **Provide clear navigation mechanisms.** Clear navigation structures provides the user a greater chance of reaching its goal. No matter what kind of situation it is, whether it is a site map, menu bar etc.
- **Ensure that documents are clear and simple.** Consistency and recognition are keywords when designing for insurance of clearness and simplicity [3].

Chapter 4

Ajax

The term Ajax was first introduced by Jesse James Garrett in 2005 and it is short for Asynchronous JavaScript And XML. Ajax itself is nothing new, by combining several already existing technologies it enables developers to create RIAs with high interactivity, responsiveness and richness [8]. One of the core strengths of Ajax compared to the classic web application model is that it does not use the same send and retrieve technique. It therefore eliminates the request and wait time often occurring when interactions demand for pages to update. According to Jesse James Garrett "Ajax applications are more responsive to user actions and the programs do not experience page-reloading interruptions" [20]. With the possibility to dynamically change content on a page an Ajax application is able to stay in display while sending and retrieving data from a web server. Additionally asynchronous communication makes it possible for the web application to stay fully functional while waiting for a web server response [10].

As seen in Figure 4.1, the normal web application model works in the following way: a user interacts with the UI which will trigger a HTTP request to the web server. The web server handles the request and the UI waits for a response. When a response returns HTML and CSS data, the page will update. During that moment the user has to wait a varied amount of time, locking up the system until the new page is in display. The Ajax web application model injects an Ajax engine between the UI and the web server. The engine unlocks the system, making it possible for a user to interact with the UI without having to wait for a response. The engine works simultaneously updating the UI and if necessary, sends data to the server and returns it to the page updating only necessary parts when retrieved [8, 20].

4.1 Technologies behind Ajax

To be able to utilize Ajax web applications it is necessary to use what has been known as five characteristic technologies. XHTML+CSS, DOM, XML, XMLHttpRequest and JavaScript. Those five make it possible for web applications to dynamically change its appearance without any high latency interruptions for the end-user [8, 20, 21, 14].

The dynamically changing web site or DHTML page differs compared to regular HTML, in a way that is has the ability to reorganize or change its order of elements within the markup based upon the user's interactions. It can for instance change text, size, or color when the user's mouse pointer is hovering over a specific link marked up in the markup language. Forcing only the link element part of the page to change its

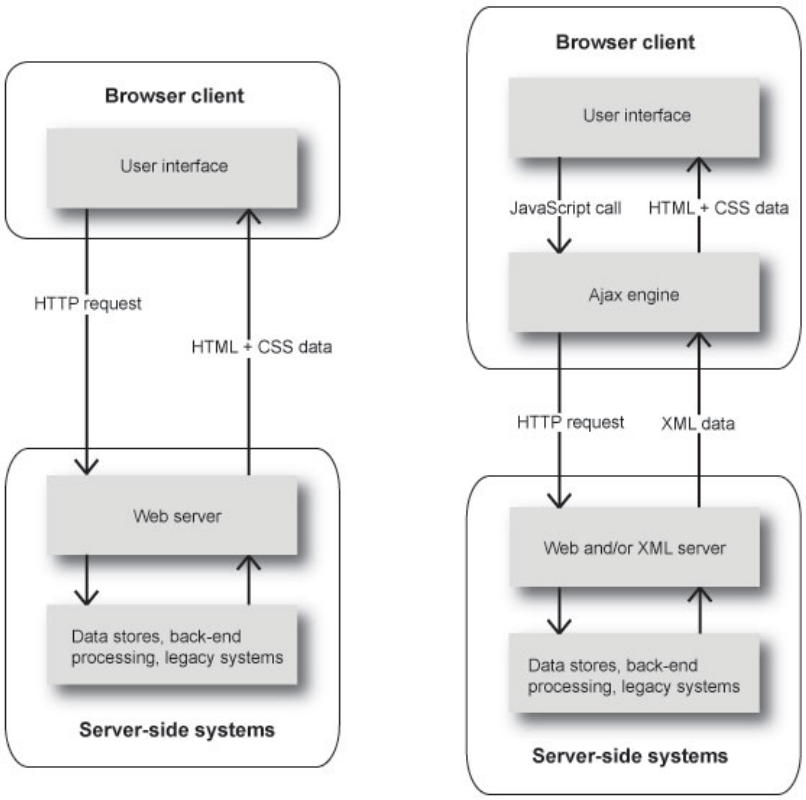


Figure 4.1: To the left showing the classic and right next side the Ajax web application model works [8, 20].

appearance when hovered [21].

4.2 (X)HTML and CSS

XHTML or Extensible HyperText Markup Language is considered to be the most commonly used markup language, and together with CSS or Cascading StyleSheets is what makes the structure and layout of the web pages within the site or application by presenting all parts, including the ones which can be changed during time. XHTML is referred to as the successor to HTML, which is considered to be a static markup language whereas XHTML is being considered dynamic.

The XHTML markup signifies the order of which all the elements (headings, paragraphs, images etc.) will be presented in the web browser. The CSS has been a W3C standard since 1996 [12]. The CSS makes it possible for web developers to have more control over how the layout should present itself on a web browser. It tells the web browser in what way the markup should present itself by defining the look and feel of every element within the page.

4.3 DOM

DOM stands short for Document Object Model and can be seen as a tree which has the possibility to rearrange and reorganize the elements within the XHTML and CSS from within a page utilizing Ajax. It has been a W3C standard since 1998 and is a programming interface/API [6].

Every unit within a (X)HTML document is represented by a node inside the tree. The DOM has the ability to traverse through the tree searching for specific nodes and manipulate and/or create new nodes with the appropriate style defined by the interface. When changes are being done within the tree the web browser will immediately refresh the page with the necessary changes made. This is the main characteristic which makes the need for full page updates unnecessary with Ajax web sites or applications.

Figures below are illustrative examples showing how the structure of the DOM tree is working.

There are several different types of nodes within a (X)HTML document but some of the most common are listed below:

- **Document:** This is the actual DOM tree and not within the main HTML document.
- **Element:** Can be many different types of elements such as the img element or p element.
- **Attribute:** The attributes within an element such as src for the img element. Where src is the actual source to the image file displayed.
- **Text:** The text within a HTML document might all have the EM or STRONG elements as their parent.

Below is a DOM tree example showing how a tree might look like. The parent node

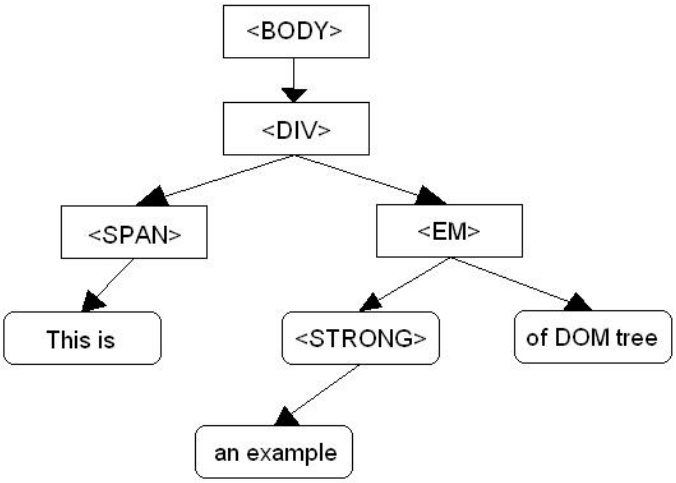


Figure 4.2: Illustrating a DOM tree where tags are used for identifying elements and ordinary text shown without any tag.

“body” contains a child node “div”. The body is parent node of div and div is therefore

child node of body. They are both nodes of type element. “This is” is a child node to element node span. It is a node of type text. A closer look at the markup language shows in which order the different nodes are constructed. The markup language is finally

```
1  <BODY>
2      <DIV>
3          <SPAN>This is</SPAN>
4          <EM>
5              <STRONG>an example</STRONG>
6              of DOM tree
7          </EM>
8      </DIV>
9  </BODY>
10
```

Figure 4.3: Shows the markup language of the example given.

interpreted by the web browser and depending on choice of elements and attributes, the developer can decide how the text should present itself.

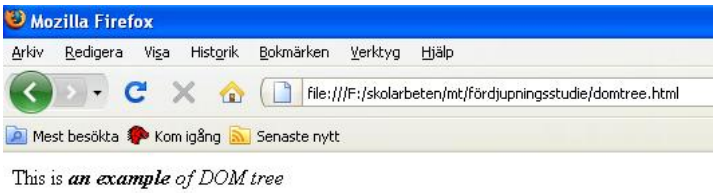


Figure 4.4: Illustrates how the web browser will interpret the markup language.

By using “strong” as a parent to “an example” makes the text present itself in bold. The “EM” element being parent to both “an example” and “of DOM tree” makes them both visualized as emphasized text.

4.4 XML

The sending and receiving of data and interpretation by the browser is often followed by XML standard. Although other techniques such as JSON (JavaScript Object Notation), HTML or simply by plain text can be used, XML is often considered the most commonly used [4].

XML is an abbreviation for Extensible Markup Language and a standard that was first accepted in 1998 by the W3C consortium as a first version [7]. It has cross-platform data operability and any developer can therefore create their own XML-based language with the appropriate tags. By following the strict standards of creating tags the web browser or application will execute reading and interpreting the data being sent or received in XML.

4.5 XMLHttpRequest

The sending and receiving of XML files from client to server can be done by in several ways, although implementing the XMLHttpRequest technique is now considered to becoming more than just a de facto standard. At their web site W3C now has a working draft [22], meaning that the W3C consortium are working on a way to standardize this characteristic technique along with others. Almost every web browser available, except Internet Explorer 6 used this technique earlier, but with Internet Explorer 7 Microsoft has claimed that they are using it as well [15].

4.6 JavaScript

Javascript is a programming language that ties all the other techniques together in the web browser. It is a programming language with a syntax quite similar to Java. Other than syntax it does not have many similarities, for instance the JavaScript code is being compiled in run time. It is a common programming language used in web pages and can be included as a part within (X)HTML documents or as a separate file linked within the (X)HTML page.

4.7 Ajax competitors

Several RIAs are competing in becoming the "ultimate" solution used when developers create applications with high responsiveness, interactivity and richness. According to Noda et al. the most commonly used platforms creating these are Ajax, Flash and Java. To make it easier for developers to decide upon platform to choose when creating RIAs Noda et. al have made up a comparison table summarizing each platforms strengths and weaknesses. Every comparison metric is given a value ranging from poor, average to good. Table 4.1 shows the difference between them.

4.7.1 Adobe Flash (Flex)

Adobe Flash has the ability to create interactive and animated web sites. It uses a script language called ActionScript. For Adobe Flash to work properly, a plugin needs to be installed on the web browser (known as Flash player).

4.7.2 Java applets

Is developed with the main purpose to support the software development over the Internet. Being platform independent Java applets have the ability to run on any platform with a browser as long as it has Java support. A Java applet has the ability to perform in a way common desktop applications normally do. Java applets, could for instance, be seen in some cases on web sites where an upload function is used. Showing a more detailed progress indication than a normal web browser would have done. The client using a site with Java applets need to have this applet installed in order to function. This along with the need of reloading itself onto the client every time a user is using the applet are considered two major set backs with the technology.

Table 4.1: Comparison of different platforms used to create RIAs [16].

Comparison table			
	Ajax	**Adobe Flash**	**Java**
Graphical Richness	Average to poor	Very Rich	Rich
Container/Engine Footprint	Good (Very light)	Average (Light)	Poor (Heavy)
Application Download	Good (Fast)	Poor (Slow)	Poor (Slow)
Audio/Video Support	Poor	Excellent	Average (OK)
Consistency on different platforms	Poor (Varies)	Good (Very consistent)	Average (Relatively consistent)
Server Requirements	Good (None or very minimal)	Poor (Yes Flex or Open Lazlo)	Average (Yes or No)
Plug-in/Runtime Req. on Client	Good (No)	Poor (Flash player)	Poor (Java Runtime Engine)
Dev. Challenge	Poor (Very complex tools, high skills req.)	Average (Rel. easy with Flex or Open Lazlo)	Average (Relatively easy w tools such as Nexaweb)
Security Concerns	Average (JavaScript codes open to public)	Good (Flash files are created)	Good (Class/Jar binary files are created)
Cost	Average (Custom Build - Free)	Average (Open Lazlo - Free)	Average (Java Web Start - Free)

4.7.3 Microsoft Silverlight

Formerly known as WPF/E or Windows Presentation Foundation Everywhere Microsoft Silverlight is another competitor to Ajax. Silverlight works in the same way as Java applets and Flash/Flex demanding software installed on a client to run properly.

4.8 Pros and cons using Ajax

With new technologies comes both advantages and disadvantages. Two itemized lists have been made with advantages listed unordered followed by disadvantages. The lists have been collected from several different sources and summarized for the reader to quickly get an overview. Considerations have been taken in order to deliver the lists as "up to date" as possible.

4.8.1 Advantages

- **Asynchronous post-backs** By eliminating the full page post-backs, web applications using Ajax are able to do partial page updates. Making the application able to run even though a call has been made to the server. Eliminating or reducing the interruption time end users will have experienced using common web applications [20].
- **Reduced network traffic** Implemented properly the Ajax web application has

the ability to send small chunks of data back and forth from client to server. Thus reducing the total amount of data being sent over the network [20, 4, 10].

- **Client side calculations** Using JavaScript some computation and work load can be handed over from server to client side within the clients web browser [20].

- **Enhanced graphical support** Modern web browsers have increased support making it possible to add richness in terms of graphics. Transparency, shading etc. are all functions which can add a new "fresh" look for web sites, making them more attractive for the end user.

- **Cross platform operability** Ajax web applications are able to run on many browsers, making them browser independent assuming JavaScript is enabled. [20, 4].

4.8.2 Disadvantages

- **JavaScript** Not all browsers have JavaScript enabled or even support making Ajax web applications inaccessible. Text based browsers such as Lynx, mobile device browsers along with screen readers for visually impaired might not be able to work unless graceful transformation is implemented.

- **Bookmarks** Bookmarking a page utilizing Ajax has proven to be difficult and confusing for users. Since the URI stays the same only changing parts of an updated page within the structure, confusion might arise when users try to bookmark a page.

- **Multiple server calls** An Ajax enabled web application sending and receiving a lot of small calls from the server might produce performance implications.

- **Unnecessary workload** Inefficient JavaScript programming can result in unnecessary workload on local machines.

- **Browser functionality** Ajax web applications will produce implications when users are trying to use a browsers back or forward function.

- **XMLHttpRequest** No current standardization method of sending and receiving XML files from client to server has yet been set. Although, W3C has announced a working draft meaning the XMLHttpRequest object could become a standard object for sending and receiving XML in the future.

- **High complexity** According to Ray Valdes there is a high level of complexity and difficulty in creating a whole site with Ajax technologies in one piece. Suggesting that Ajax enabling a whole site is easier being made in bits and pieces. "Your average developer is not going to be able to figure it out. Only a small, elite group has the ability to do it in a comprehensive and complete way." he says [20].

- **Audio and video** Both HTML and JavaScript are lacking the support for audio and video streaming making it difficult, if not impossible to use within certain areas of application development.

- **Security** Combining several existing techniques in a new and unfamiliar way making them vulnerable to security leaks. By using pre-built frameworks, developers have to rely on suppliers to keep the frameworks constantly updating and secure whenever new vulnerabilities are presented.
- **Debugging** Depending on the browser used, different debugging tools must be used in order for developers to have the ability to debug JavaScript. No cross browser debugging tool has yet presented itself nor has it been standard. Third party debugging tools might therefore need to be installed in order to debug properly.

4.9 Rich Internet Applications in the future

Nyman, S. talks about people having a vision where in the future users will be able to run all their applications over the Internet. By doing so, it will be possible to work anywhere where there is a computer or similar electronic device with web browser support along with an Internet connection [18].

Google Docs is a good example which offers users the possibility to create, share and work cooperative in real time. It supports several of the common file formats, making the need for users to have several different installed desktop applications unnecessary [9].

With sites and applications utilizing Ajax or similar technologies, the end users might get accustomed to what RIAs has to offer. Often being high degrees of interactivity, responsiveness and richness. In the end changing the expectations end users have when visiting new sites or applications for the first time. The threshold users have e.g. towards interruption times might become smaller. Making it important to strive against satisfying the ever evolving and changing “quality of use” for the end users.

Chapter 5

Existing Systems

LNeurocom have several different systems for maintaining, collecting and visualizing important data. On their Intranet site they have a telephone list where employees are listed along with their work related information. This in order for management to easily get in touch with the employees whenever something unpredictable happens.

Another system the company is currently working with is MS Excel sheets, where all the scheduling for employees take place. All statistics are pulled from predefined Excel sheets establishing connections to their Informix database making queries in order to visualize data on the sheets. The companies intentions are to try and centralize all of these along with some other data into a single web application where all procedures can be done.

5.1 Intranet homepage

On the Intranet homepage a telephone list exists along with some other sections. The list section has a search function and a grid filled with search results. A sorting method in the data grid makes it possible for users to list based upon choice of column. When searching for something that can not be found an error message saying: "Sorry, no data found" will be displayed at the page. The sorting function has ascending/descending buttons where the user can choose how the sorting will function.

5.2 Scheduling with MS Excel sheets

Excel sheets called "Vagtplan" are predefined documents where team leaders insert schedules for all their agents within a specific team. The schedules are then stored for every period, normally monthly based. Figure 5.2 shows different dates as columns and agents as rows in a grid. Inserting a new schedule for a specific agent is done by selecting a cell and inserting the time from which they will work. All cells are set with white background initially and if any changes occur after a schedule has been set, a team leader has to open the sheet and manually set the appropriate reason of absence based on a color coding prefix. They also have to type in the type of absence and during which time this absence took place. When hovering with the mouse over a cell a tool tip will appear showing the reason of absence for a specific user. The example shows a

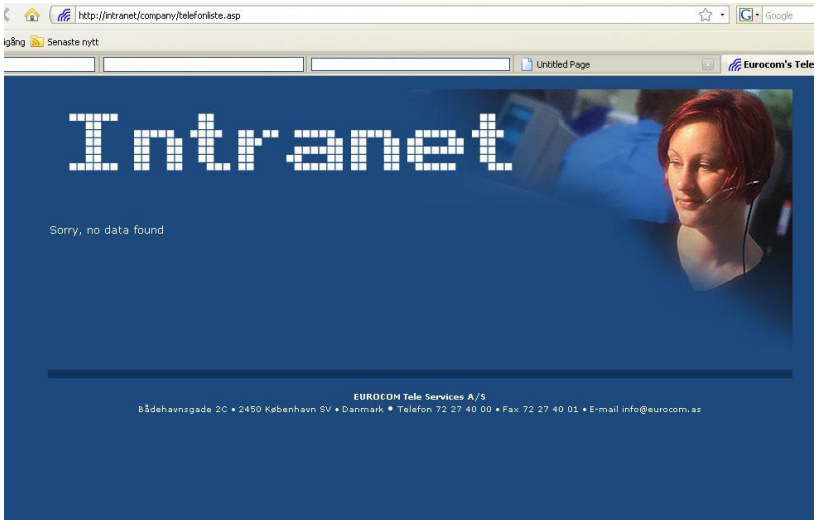


Figure 5.1: Shows when user is searching for something not found.

time period of one month ranging from the 21st September to 20th October. Scrolling horizontally will show the following dates within that time.

5.3 Reporting with MS Excel sheets

There are several different excel sheets handling the reporting of statistics. For instance LNeurocom have one sheet visualizing the agent performance statistics. This excel sheet has predefined SQL queries invoked to retrieve data to calculate the metrics needed for visualization in Excel. Figure 5.3 illustrates the agent performance chart below with data fetched from within a week set to 39, showing all agents within a specific team and how they have performed. The metrics are shown in different formats such as percentage, time in hh:mm:ss, mm:ss and by simply showing a number. The final row shows the team total summarized for every specific metric. By striving to always improve their quality and performance, LNeurocom has to analyze this data. To see what needs to be done in order to perform their very best in order to fulfill the goals set with all partners and customers.

	A	D	G	J	M	P	S	V
1	21/8 - 20/9	torsdag 21-aug Schedule	fredag 22-aug Schedule	lördag 23-aug Schedule	söndag 24-aug Schedule	måndag 25-aug Schedule	tisdag 26-aug Schedule	onsdag 27-aug Schedule
3								
4		0800-1600	0800-1600			0800-1600	0800-1600	0800-1600
5		0730-1530	0730-1530			0730-1530	0730-1530	0730-1530
6								
7								
8								
9		0930-1730	0930-1730			0930-1730	0930-1730	0930-1730
10		0730-1530	0730-1530			0730-1530	0730-1530	0730-1530
11		0700-1500	0700-1500			0700-1500	0700-1500	0700-1500
12		0700-1400	0700-1300			0700-1700	0700-1600	0700-1500
13								
14								
15		0800-1600	0800-1600			0800-1600	0800-1600	0800-1600
16								
17								
18		0900-1700	0900-1700			0900-1700	0900-1700	0900-1700
19		0900-1700				0950-1700	1000-1700	0900-1700
20		0900-1700	0900-1700			0900-1600	0900-1700	0915-1700
21								
22		1000-1700	0900-1700		0800-2100	1300-1700	1300-1700	1300-1700
23			0900-1700			0900-1200		0900-1200
24								
25								
26						0845-1800	0845-2000	0845-1800
27		0900-1700	0900-1700		1000-1800	0800-1600	0800-1600	
28		0800-1600	0815-1600			1200-2000	0805-1600	0800-1600
29		0800-1600	0800-1600			1200-2000	1200-2000	1200-2000
30		0800-1600	0800-1530			0800-1600	0800-1600	0800-1600

Figure 5.2: Excel sheet used for scheduling agents.

	Data																	
e	ACD Calls	Avg Talktime Incalls	Avg ACW	Avg handle time	Call Backs	Sum of Calls In+Out	Staff Time	Avg IN Calls / Hour	Avg OUT / Hour	Avg IN & Out / Hour	Sum of Avg Talktime Out	Sum of Outbound time	Produktivit et.	Sum of ACD+ACW+Avail time	(ACD+Availtime) /Stafftime	Sum of Total Talktime
	187	05:29	00:12	05:41	5	192	35,6	5,3	0,1	5,5	04:02	00:09:26	51,3%	30:42:43	87,7%	17:17:37
	2	09:03	00:02	09:05	0	2	0,3	6,5	0,0	6,5	00:00	00:00:00	97,9%	00:18:07	97,6%	00:18:07
	11	03:14	00:22	03:36	4	15	8,0	1,4	0,5	1,9	01:02	00:04:07	9,1%	07:42:07	96,2%	00:39:45
	59	08:29	06:08	14:37	30	89	36,3	1,6	0,8	2,5	20:54	01:38:24	44,2%	16:05:47	44,3%	09:59:35
	53	05:10	04:58	10:08	48	101	37,1	1,4	1,3	2,7	38:33	01:52:01	29,2%	06:51:14	18,5%	06:26:38
	65	07:57	03:57	11:54	41	106	110,2	0,6	0,4	1,0	26:12	02:29:59	14,0%	18:41:14	17,0%	11:06:46
	103	07:57	01:53	09:50	25	128	44,6	2,3	0,6	2,9	36:17	01:19:52	40,9%	26:51:32	60,2%	15:00:02
	3	00:10	00:03	00:13	0	3	8,0	0,4	0,0	0,4	00:00	00:00:00	0,1%	07:47:01	97,2%	00:00:31
	98	07:55	02:49	10:44	22	120	39,7	2,5	0,6	3,0	59:00	01:57:41	49,2%	20:56:47	52,8%	14:54:11
	65	05:35	00:37	06:12	6	71	18,9	3,4	0,3	3,7	15:49	00:47:26	39,7%	13:37:52	72,0%	06:51:01
	199	03:25	00:35	04:00	6	205	40,1	5,0	0,1	5,1	08:59	00:12:23	33,8%	32:22:23	80,8%	11:35:16
	82	03:15	01:22	04:37	9	91	69,3	1,2	0,1	1,3	09:39	00:13:00	9,5%	66:07:17	95,4%	04:40:11
	103	03:30	00:49	04:19	24	127	27,3	3,8	0,9	4,6	04:46	00:50:06	30,3%	20:39:21	75,5%	06:52:01
	17	05:33	03:34	09:07	3	20	8,0	2,1	0,4	2,5	05:05	00:15:16	35,7%	03:27:26	43,4%	01:49:52
	64	09:03	03:01	12:04	98	162	39,8	1,6	2,5	4,1	22:15	05:49:15	47,0%	15:29:42	38,9%	15:29:00
otal	1111	05:41	01:48	07:29	321	1432	522,6	2,1	0,6	2,7	12:33	17:38:56	29,9%	287:40:33	55,0%	123:00:33

Figure 5.3: Agent performance chart showing a specific team within a week.

Chapter 6

Accomplishment

Knowledge retrieval about their existing systems and what data would be needed to store in order for the SQL database to be created was first needed. An overview of how the organisation was built up was also needed. How procedures in which from when a call was being received to the call center, how the call then worked its way through their systems to finally end up ringing at an employees telephone headset.

A call flow chart was given to further retrieve knowledge of how a call made its way to finally reach an agent. See Appendix A for more detailed information.

LNeurocom's telephone systems had some special features, which we were given a introduction appointed to understand how it works. How the most common features of the telephone system worked in terms of different dial codes etc. The different positions within the company also made it more clear how the database needed to be constructed in order to be able to handle all different scenarios that might arise when up and running.

Documentation regarding their existing Informix database was available as both printed and digital media. Further knowledge about their Informix database was necessary in order to understand how data was going to be fetched in the appropriate way (being able to visualize it in the web UI). For more information regarding Informix see: http://support.avaya.com/edoc/docs/cms/cms6rpt2.pdf.

The SQL database was created in Microsoft SQL Server Management Studio Express, running locally on a laptop computer during the implementing period. To assure compatibility against their existing system the level of the database had to be set to SQL Server 2000. Creating all the necessary tables and attributes within them was being done in database diagram mode. The database diagram mode is a GUI which makes it easy to create and edit tables and attributes within each table. When the database was created it consisted of twelve tables with several attributes and relationships from one to another (see Appendix B for more detailed description of the database, its tables and their relations).

When the database had been created we were given test data for insertion in the database. This in order to make sure the database would be able to work properly once "real data" and connections were established to the web UI.

The next step was to create the web UI and make sure it was able to communicate with the SQL database along with their existing Informix. The web UI was developed as an ASP.NET web site in Visual Web Developer 2008 Express Edition with C-sharp as programming language. The design process started out with quick sketches being drawn on paper to visualize how the GUI might look like. Several different approaches

was considered and drawn. When suggestions were approved, implementing the GUI started taking place. Utilizing a master page and separate CSS made handling layout of different pages, content and manipulation easier.

Construction of the database along with the web UI has taken up a large part of the thesis project. After a couple of weeks of the thesis my motherboard and hard drive both crashed, resulting in a lot of data loss. Fortunately due to good documentation and physical memory I was able to get everything back the way it was in less than a week. After the recreation of the database and web UI I learned to do backups on a more regular basis.

Different areas were given different pages within the UI. This made it easy to determine when an area had been implemented and working in a way LNeurocom aimed it to. For instance scheduling tasks were implemented in a page called "Schedule.aspx". When the functionality within that page was fulfilled I could proceed with another area. It also made implementing several parts side by side possible. Whenever I ran into problems that needed some guidance or help, I directed my attention to other pages and functionalities reducing time spent waiting to solve problems.

Meetings were held with my external supervisors at the company on a weekly basis. During the meetings we sat down and discussed what had been done in the past week and what needed to be done the next.

The most challenging tasks during the thesis have been to implement a working schedule and visualizing statistic reports. Making a user friendly UI for the input/edit page has also been a stimulating task.

Chapter 7

Results

Below is a system description of the implemented components and how they relate to each other.

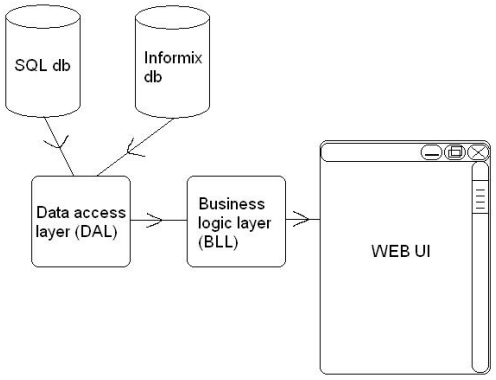


Figure 7.1: Description of all included parts and how they relate.

Using different layers the applications becomes flexible, secure and maintainable when used and further implementing needs to be done. A common way of implementing web applications is to separate the data access logic with the presentation layer. The separated layer/layers are called Data Access Layer (DAL) and Business Logic Layer (BLL). By separating the layers from the presentation all relevant code will be set into these. The DAL will hold all code that is connected to the data source such as creating the connection to the database along with common commands such as: Select, Insert, Update and Delete. The methods within the DAL will, when invoked connect to a database and make the appropriate query and return the results. As DAL separates the code from the presentation layer it does not involve any specific rules involved in the application. For example, not every user will have the possibility to delete certain data from the database. This is where the BLL is used, containing all the necessary code which for instance will execute and make sure authorization is required for certain procedures. Assuring the level of authorization has been acquired and the delete method can be invoked from the DAL. A good way of implementing presentation layers in ASP.NET is to have a part called master page. Master page's creates a way of implementing/enforcing a common design/layout throughout all pages within a web

site. The master page can contain static or dynamic content that must appear on all pages e.g. company logo, specific graphic, menus etc. Once a master page has been set the developer can focus on developing user controls, web parts and content. These controls go into the placeholders of the master page and thereby inheriting CSS from the master page. Future changes being made can in some occasions be done site wide by simply modifying the master page.

7.1 Web application

7.1.1 Master page

The master page is made up of a logotype created in Adobe Illustrator and is visible at the top left, see Figure 7.2. On the top right side there is a modified ASP.NET calendar with the possibility to show week numbers. Below the logotype is a welcome text with a label dynamically changing with users' login name depending on the user currently logged in to the application. At the bottom of the master page is a horizontal navigational menu containing all the available pages with text links associated to each individual part. Hovering over a link will trigger highlighting in terms of background changing color, to further make the user aware of selections available, see Figure 7.3. The master page will present itself on all sub pages regardless of URI in the hierarchy. Within the master page an ordinary ASP.NET calendar control has been implemented and modified with the ability to show week numbers along with actual dates in a month view with possibility to go previous and next. For more information, see Figure 7.4. Having this sort of calendar at the top of the web application making it always visible, and especially for scheduling tasks had been a wish that only the future will predict if successful or not. Using JavaScript the calendar is able to build up a column named "week" showing the numbers next to the dates with the correct weeks. It is supposed to work following the rules set by ISO 8601 for week numbering.

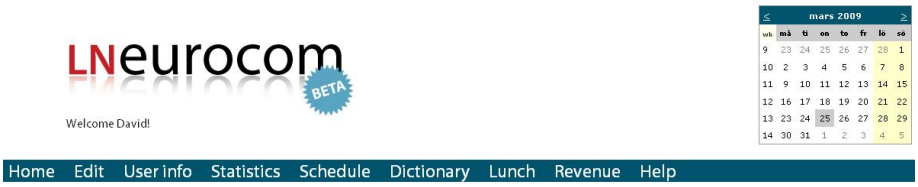


Figure 7.2: Showing the logotype made in Illustrator.

Figure 7.3: Showing the hovering effect over a link in the horizontal menu.

Figure 7.4: An ordinary ASP.NET calendar modified with JavaScript to show week numbers.

7.1.2 Home

The Home section has not yet been implemented, but will have different data displayed depending on user authorization.

7.1.3 Edit

The edit page works as an UI for inserting, deleting and editing data from the SQL database. Users without proper authorization level will not be able to see or reach this page, nor will they be able to use it. The horizontal navigation bar in the master page will dynamically change depending on the users level of authorization. The page is divided into three parts where every part has its own tab panel. This is implemented in order for users to easily switch between several views without taking up too much space in the browser. TabContainer is an ASP.NET Ajax control which creates a certain amount of tabs and structures up the contents of these. Using tabs will reduce the space required only showing the data from the current tab. This interaction style is being run by several large web browsers such as Internet Explorer, Mozilla Firefox etc. With many users having some knowledge about web browsers and how they function this implementation will most likely not be unfamiliar for most. Using master/detail view the user can edit certain data in the database from the web UI. Selecting a certain field from a grid view in “master” will make a detail view appear as “detail” where the edit, new and delete will appear. Figure 7.5 illustrates this. With client side validation the edit option chosen will help the user create or edit data in the appropriate way. Figure 7.6 illustrates the edit mode. When a change has occurred the master view updates automatically confirming the end user to see the actual change has been saved within the database.

7.1.4 User info

Information regarding employees such as work related information is stored in the newly created SQL database in a table called employee. The web application has a page called “User info” which will be a replacement of their existing telephone list from within their Intranet site. The page is built up with the use of cascading dropdown lists where the user gets the possibility to choose from within which “site” he/she wants to search. Next to the dropdown list is a “Show all” button which is implemented in a way to show all employees given that a site has been chosen. Underneath the site dropdown list is a

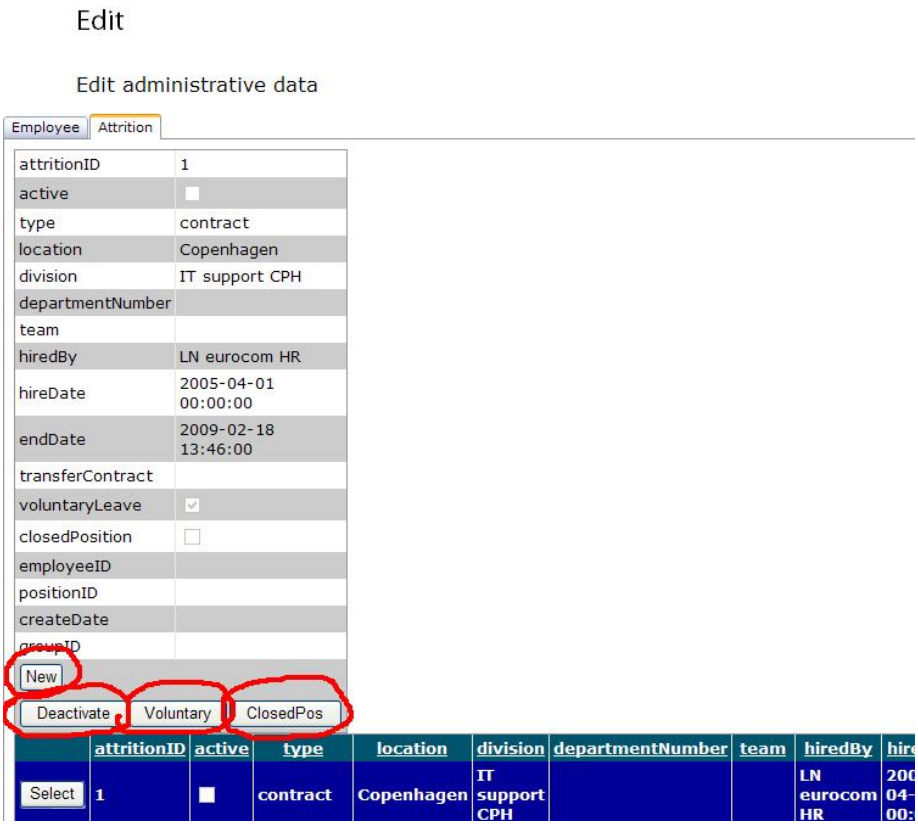


Figure 7.5: Illustrating when select has been pressed on a specific row in the grid view. Showing a details view above with edit functions. In this case from the "Attrition" table.

team listing. Disabled at first, only to get enabled when an end user has selected a site. Next to the team dropdown list is also a "Show all" button that works in the same way as the site's, except it shows all agents from within a specific team in a grid view instead. If the end user decides to see a specific user within a team he/she simply has to select from a third dropdown and by doing so triggering an update panel to update itself with the information regarding the specific user in a detail view. Figure 7.7 shows how the user info is meant to work.
With authorization methods the end users will be able to see different things in the user info page depending on level of authorization. The two different views are categorized as work related and a more detailed personal information. The "regular" user will therefore only be able to see the work related information regarding other users while users with higher authorization level, team leaders and above will be able to see all information regarding a specific user. Moreover the team leaders will only have access to agents within their team. Whereas higher authorized personnel will be able to see more detailed info regardless of a specific agents team belonging.
Cascading drop down is a function that utilizes Ajax technology. Its user-friendliness makes it easy to understand in which steps and how far the end user must go in order to achieve its goal. A web service is working in the code behind retrieving data from

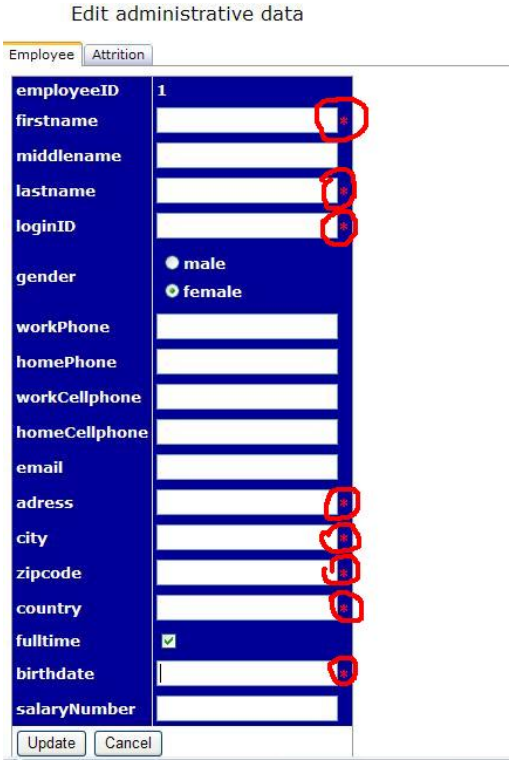


Figure 7.6: Pressing the edit button, in this case of an employee from the employee grid view which have fetched data from the employee table.

the database, which will fill the drop down lists along the way. The update panel is implemented as an area from which all the data will be presented, in doing so only requesting the necessary data that needs to be refreshed on the page from the server.

7.1.5 Statistics

The statistics is another section not yet implemented. The main layout for the statistics section has been set only missing some SQL statements in order to visualize data in table, graphic or dual mode. Figure 7.8 shows a detailed view of how the statistics page has been implemented.

7.1.6 Schedule

A page called Schedule.aspx is considered to replace the current scheduling procedure they have which uses MS Excel sheets. Their predefined MS Excel sheets for scheduling their employees will then be of abundance. By storing user schedules in a table called schedule with a relation to specific employees the web UI will be able to centralize the storing of data. Instead of having several MS Excel sheets for every scheduling period (most commonly per months) the database will store by date. This makes it more flexible to view from a specific time frame instead of static months, ranging from first to last of the month.

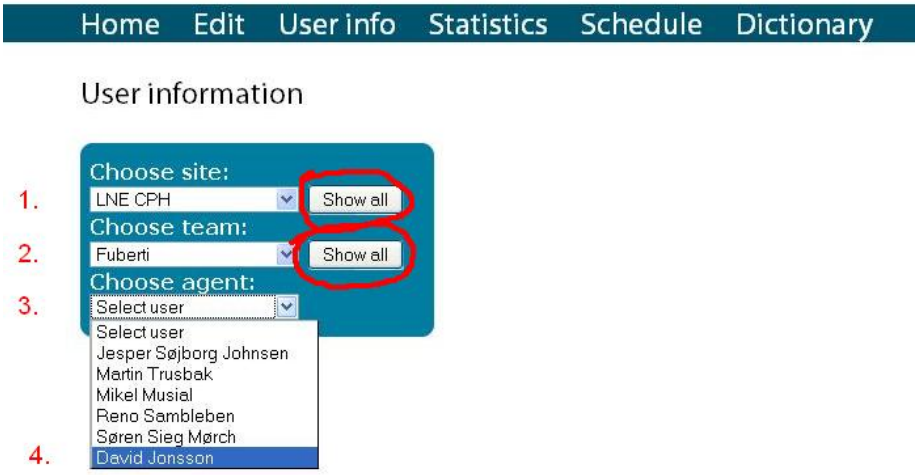


Figure 7.7: Showing when both a site and team have been selected. Once agent has been selected a details view will be displayed underneath showing an agents work related info.

The visualization in the web UI has been implemented to look very similar to their earlier MS Excel sheets. Where employee names are shown in rows and dates within a time frame are built up as columns alongside the names. The web service fills the drop down lists with active sites and teams from stored procedures. When presenting the schedule data in a grid view a stored procedure named "GetScheduleCrosstab" is called. It is a modified procedure that has been used earlier, which was supplied and approved for use by Jan Hansen (Developer/DB administrator at the company). It is a procedure that has a higher complexity than some of the other procedures. It is built up by a sub query inside a regular query making sure the date fields become columns and employee name fields become rows. A limitation forced upon the stored procedure is that the SQL Server has a maximum character amount of 4000, making it impossible to invoke if end users want to view data within a long time span. In order to avoid errors being made, client-side validation has been implemented in the date input fields. The validation makes sure the date difference between start and end is within a 31 day time frame.

The schedule page consists of two parts where one supplies the option to view data on a team detail level. The other part shows specific user data on employee detail level assuming a specific date has been chosen. The time frame from which the end user will see the team schedules has two date input fields. The two date fields have client side validation assuring the user is not trying to view e.g. with an end date before the actual start date. The validation also makes sure the user is typing the date in the right format (in this case YYYY-mm-dd).

With calendar extenders for both date input fields the users will be informed in a clear and simple way what is wrong with a suggestion to fix in order to achieve its purpose. Calendar is an ASP.NET Ajax extender attachable to any text box control. It provides client-side date picking functionality with custom date format and UI in a

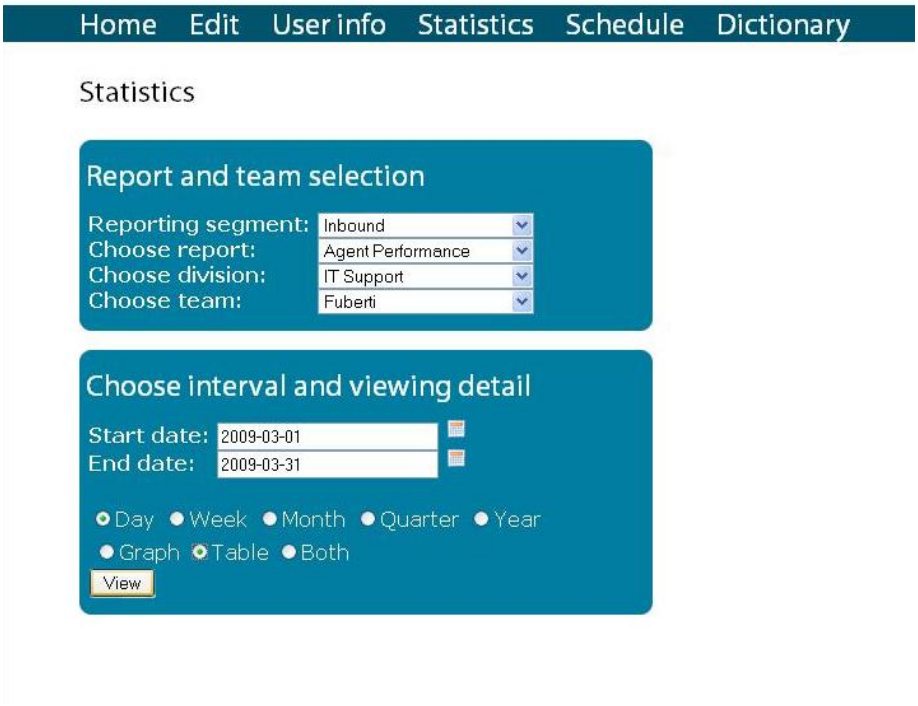


Figure 7.8: Visualizing the general layout for the statistics section.

pop up control. The user can interact with the calendar by selecting a day or switching between months/years etc. This allows the user to easily jump in time to choose dates from within the calendar.

A wish was to have the schedule view showing different colors depending on absent type within the schedule view. As seen from the Excel sheet example in fig.7.2. After discussions with my external supervisors at the company, we all agreed upon trying to have at least two different color codes in the grid view depending on absent type when changes have occurred. One color showing changes that are work related the other non related work absence. In that way the user will be able to get a quick overview of employees and the two types of absence that might have occurred during a specific time span. To view employee detail such as reason of absence, time of absence etc. the user need to supply a date and user name at the employee detail part.

7.1.7 Dictionary

A page called dictionary has been implemented to work as a digital dictionary for the user of the application. A certain amount of work related lingo with abbreviations and explanatory text is stored in the database and retrieved in the application. Since a dictionary can become quite large by time a quick search function has been constructed. The quick search function has character links which takes the user to the specific part in a quick manner. Instead of having to scroll down to the first character of letter searched. The navigation towards the desired abbreviation might prove more efficient. In order to try and minimize the information displayed to the end user thereby not forcing them

	A	D	G	J	M	P	S	V
1	21/8 - 20/9	torsdag 21-aug Schedule	fredag 22-aug Schedule	lördag 23-aug Schedule	söndag 24-aug Schedule	måndag 25-aug Schedule	tisdag 26-aug Schedule	onsdag 27-aug Schedule
3								
4		0800-1600	0800-1600			0800-1600	0800-1600	0800-1600
5		0730-1530	0730-1530			0730-1530	0730-1530	0730-1530
6								
7								
8								
9		0930-1730	0930-1730			0930-1730	0930-1730	0930-1730
10		0730-1530	0730-1530			0730-1530	0730-1530	0730-1530
11		0700-1500	0700-1500			0700-1500	0700-1500	0700-1500
12		0700-1400	0700-1300			0700-1700	0700-1600	0700-1500
13								
14								
15		0800-1600	0800-1600			0800-1600	0800-1600	0800-1600
16								
17								
18		0900-1700	0900-1700			0900-1700	0900-1700	0900-1700
19		0900-1700				0950-1700	1000-1700	0900-1700
20		0900-1700	0900-1700			0900-1600	0900-1700	0915-1700
21								
22		1000-1700	0900-1700		0800-2100	1300-1700	1300-1700	1300-1700
23			0900-1700			0900-1200		0900-1200
24								
25								
26						0845-1800	0845-2000	0845-1800
27		0900-1700	0900-1700		1000-1800	0800-1600	0800-1600	
28		0800-1600	0815-1600			1200-2000	0805-1600	0800-1600
29		0800-1600	0800-1600			1200-2000	1200-2000	1200-2000
30		0800-1600	0800-1530			0800-1600	0800-1600	0800-1600

Figure 7.9: Picture explaining how the scheduling works in their existing Excel sheets.

"information overload". Information overload being an ambiguous word, but in this case meaning the end user gets supplied with a tremendous amount of unread information on a single page [5]. The dictionary has collapsible panels implemented where every abbreviation contains of a panel with the abbreviation as heading. When a user has found the desired abbreviation a mouse click on the collapsible panel will trigger another panel to appear, showing the explanatory text connected to that specific abbreviation. The collapsible panel has a built in hide/show function which means one click will show and the next will hide the explanatory text linked to the abbreviation. By using the collapsible panels the chance of end users experiencing information overload might be minimized, the page also gets more condensed. The collapse panels can be modified by the developer however he/she wants it to look and appear. The panels themselves are post-back aware which means it is possible to configure to act in a specific way even after a full page post-back has occurred.

As mentioned earlier, validations are used in different input text boxes to assure the end user has typed "correctly". The Ajax validation extenders give the end users a high degree of feedback by a pop up which explains the errors made and can suggest how to solve them. The pop up can be closed by hitting the x button. The pop up will disappear if the x button is pressed, it will also disappear as soon as the validate accepts the input typed in by the user. Successfully implemented these extenders might direct the users' attention to the error and by giving advice supplying easy solutions.

7.1.8 Help

This section is kept simple with text informing the user of a link to a adobe .pdf document containing a user manual for the application. See Appendix C for further details.

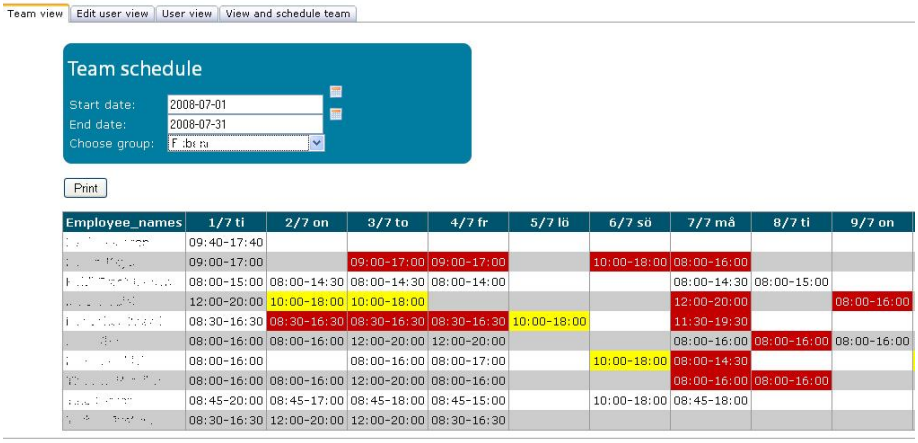


Figure 7.10: Showing the new implemented user scheduling.

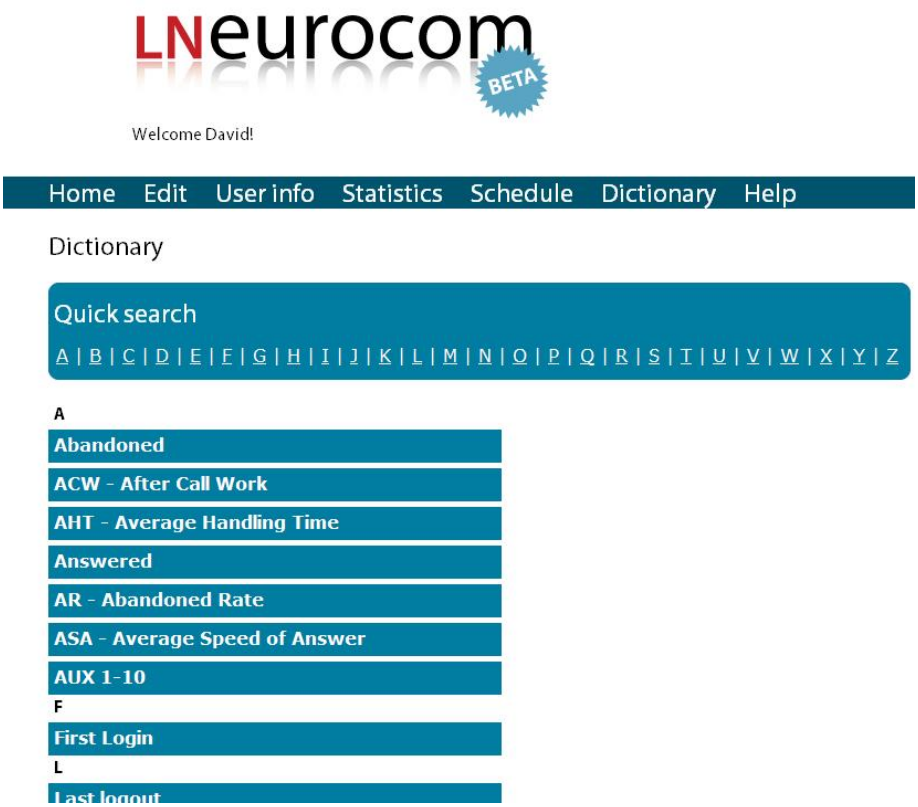


Figure 7.11: All headers fetched from the database are shown in alphabetic order.

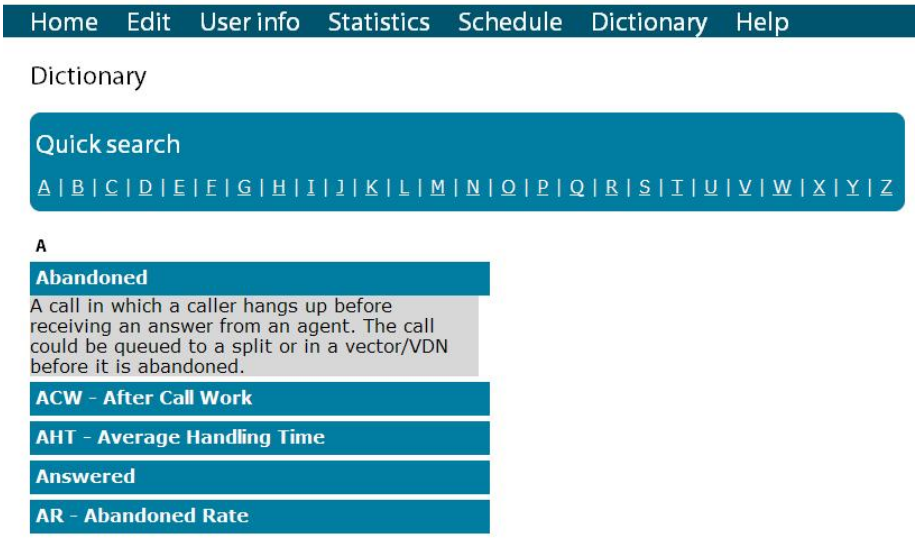


Figure 7.12: Clicking on a header will display the explanatory text connected to the header.

Chapter 8

Conclusions

Has Ajax been applicable and usable on the current web application? In this case it has been fairly advantageous since limitations have removed some issues arising when implementing Ajax. MS Internet Explorer 7 being the only browser used along with JavaScript have made it possible to discard the disadvantage's Ajax and specifically associated issues JavaScript can bring. Having no demands for audio or video streaming support, the web application has remedeed another possible disadvantage.

The impact on reduced network traffic that Ajax web applications can bring will be difficult to determine at this time, since no current web application of this type exists and thus comparisons are difficult.

Starting from scratch developing an Ajax web application has been a very challenging and insightful experience. Agreement with Ray Valdes, claiming Ajax has a disadvantage of being highly complex when trying to Ajax enable whole web sites.

Ajax has proven to gain a higher extent of usability when used compared to ordinary web applications. When implemented it has also been proven that performance gains can be achieved.

A disadvantage witnessed in earlier studies, can now be considered irrelevant if a workaround is implemented. The disadvantage here, being the inability to use the browsers back button in a correct manner, as Ajax web applications and especially single page applications have had troubles in the past utilizing the browsers back button functionality. Evidently this issue has been solved now with workarounds which might produce more implementing, on the other hand enabling the functionality many considered essential within a browser.

Security concerns have been addressed by some [23]. When using old techniques in a "new" way binding them together as an Ajax enabled web application making new security issues arise which might need to be overseen. Using pre-built frameworks the developer has to rely on framework developers to strive at keeping the security level high within, by making appropriate updates when severe security issues are exposed and published. Simply by supplying the developers with updated frameworks, in this case the ASP.NET Ajax framework has been used for MS Visual Web Developer 2008 Express. The developers implementing Ajax web applications might need to do further research, in order to compare and analyze which framework best meets with the needs for security aspects.

Utilizing cross-browser independence Ajax has the ability to run on several different web browsers making it very flexible. That said it will act and visualize all data within

the application in the same manner depending on browser used. If every single technique becomes standardized in the future, "validators" could possibly provide a more trustworthy cross-browser compatibility. This could make it unnecessary for developers to test their applications/sites on several different browsers in order for assurance.

Web site designers striving to develop applications with cross-platform independence need to consider the web content accessibility guidelines described earlier. To be able to follow them correctly, the application needs to handle graceful transformation. Meaning browsers without e.g. JavaScript support might need further implementing in order to supply these users with an alternative application not using JavaScript. Thus finally producing more work for developers.

Being able to bookmark pages within a site or application has been an issue as well. Workarounds are available here, but will add more implementing for developers in order to function.

8.1 Limitations

The schedule time frame view has been set to only visualize within a certain day time frame. This in order to fulfill the requirement forced upon the stored procedure "GetScheduleCrosstab", since SQL Server has a maximum character amount of 4000.

The use of only Windows Login Authentication as a way of certifying a user has access to use the application could possibly be considered a weakness by some. No login/logout has been implemented in the application to further secure users from "stealing" or in other ways use other users computers while logged on with a browser. To retrieve a higher level of security within the application a login page with automatic logout if a user has been inactive a certain amount of time could be a solution.

8.2 Future work

The application has not taken into consideration the possibility of several users trying to modify data at the same time. Therefore the database will store information based on time, making the last update override others. Using "optimistic concurrency" the scenario described previously can be avoided and handled by giving the users feedback through the UI in a way that end users know others are trying to modify as well. The cause of this has been that a very low possibility of two or more users are editing the same information at the same time.

Since the original specification only included MS Internet Explorer 7, no general regards towards compatibility with other browsers has been taken. Though personal considerations have been taken trying to follow the web content accessibility guidelines set by the Web style guide as much as possible. Along with successful validation towards CSS level 2.1 supplied by the W3C consortium and a level of acceptance regarding XHTML 1.0 markup language.

Without any existing web application in use the design process started with defining several pages handling different procedures. The Ajax web application itself should not be considered an Ajax enabled single page application, but a multiple page web application consisting of Ajax enabled parts. Used with the basic purpose of enhancing the end user experience in regards of usability and performance. Using update panels on parts within the application, where a number of interactions are considered high the UI

will appear to act instantly without any full page reloads disabling the user to interact while requesting post-backs.

The web application has no ability to handle gracefully transformation when used without a JavaScript enabled browser. It only informs users without JavaScript to make sure they use it in order to take full advantage of the application, thus making parts of the application inaccessible using a browser without JavaScript support. Discussing this issue with my supervisors has made it clear that this was not considered a high priority task, and therefore future work.

The application has not yet been installed within the company's Intranet for beta testing. When installed for testing, bugs and modifications might include some further implementing. Optimizations and logging might also prove invaluable.

After beta testing and possible further implementation, a survey consisting of end user satisfaction, usability and performance could be conducted in order to determine the "quality of use".

Chapter 9

Acknowledgements

I would like to specially thank Björn Sjöström and Jan Hansen, external supervisors at LNeurocom teleservices A/S for their tremendous help that has been given during the thesis project. Thanks to all of LNeurocom for giving me the opportunity to do my master's thesis in Copenhagen, Denmark. Thanks to Jerry Eriksson, my internal supervisor at Umeå University for helping me with input and feedback on the report.

References

[1] Nigel Bevan. Usability issues in web site design. In *Proceedings 6th Interactive Publishing'99*, pages 1–7, 1999.

[2] T. Braescu. Interaktiv webbapplikation för möbelprodukter, 2007. http://www.diva-portal.org/diva/getDocument?urn$_n bn_s e_h ik_d iva - 336 - 2_{fulltext.pdf}, accessed 2009 - 02 - 12$.

[3] Wendy Chisholm, Gregg Vanderheiden, and Ian Jacobs. Web content accessibility guidelines 1.0. *interactions*, 8(4):35–54, 2001.

[4] A. Jason Clark. Ajax (asynchronous javascript and xml): This isnt the web im used to, 2006. http://www.infotoday.com/online/nov06/Clark.shtml, accessed 2009-02-10.

[5] Angela Edmunds and Anne Morris. The problem of information overload in business organisations: a review of the literature. *International Journal of Information Management*, 20(1):17 – 28, 2000.

[6] Lauren Wood et. al. Document object model (dom) level 1 specification, 1998. http://www.w3.org/TR/REC-DOM-Level-1/, accessed 2009-02-10.

[7] Tim Bray et. al. Extensible markup language (xml) 1.0, 1998. http://www.w3.org/TR/1998/REC-xml-19980210, accessed 2009-02-10.

[8] J-J. Garrett. Ajax: A new approach to web applications. Technical report, http://www.adaptivepath.com, 2005.

[9] Google. Google docs, 2009. http://docs.google.com, accessed 2009-02-11.

[10] J. Kluge, F. Kargl, and M. Weber. The effects of the ajax technology on web application usabiliy, 2007.

[11] J Kristiansson and J Pettersson. Förbättra internetsystem genom implementation av ajax-teknik, 2006. http://urn.kb.se/resolve?urn=urn:nbn:se:hj:diva-436, accessed 2009-02-12.

[12] Håkon Wium Lie and Bert Bos. Cascading style sheets, level 1, 1996. http://www.w3.org/TR/CSS1/, accessed 2009-02-10.

[13] PJ Lynch and S Horton. *Web style guide*. Yale University Press; 2nd edition (March 2002), 1999.

[14] Ali Mesbah and Arie van Deursen. Migrating multi-page web applications to single-page ajax interfaces. *Software Maintenance and Reengineering, European Conference on*, 0:181–190, 2007.

[15] Microsoft. Xmlhttprequest object, 2009. http://msdn.microsoft.com/en-us/library/ms535874(VS.85).aspx, accessed 2009-05-04.

[16] T. Noda and S. Helwig. Rich internet applications. Technical report, http://www.uwebc.org, 2005.

[17] Donald A. Norman. *The Design of Everyday Things*. The MIT Press London,England, 2001.

[18] S. Nyman. Webbapplikationer med ajax, 2006. http://www.cs.umu.se/education/examina/Rapporter/SamuelNyman.pdf, accessed 2009-02-10.

[19] Jonathan W. Palmer. Web site usability, design, and performance metrics. *Info. Sys. Research*, 13(2):151–167, 2002.

[20] Linda Dailey Paulson. Building rich web applications with ajax. *Computer*, 38(10):14–17, 2005.

[21] K. Smith. Simplifying ajax-style web development. *Computer [0018-9162]*, 39(5):98– 101, 2006.

[22] Anne van Kesteren. The xmlhttprequest object, 2008. http://www.w3.org/TR/XMLHttpRequest/, accessed 2009-02-10.

[23] J West. The dark side of ajax, 2006. http://assets.en.oreilly.com/1/event/3/ThePresentation.pdf, accessed 2009-02-12.

Appendix A

Call Flow Chart

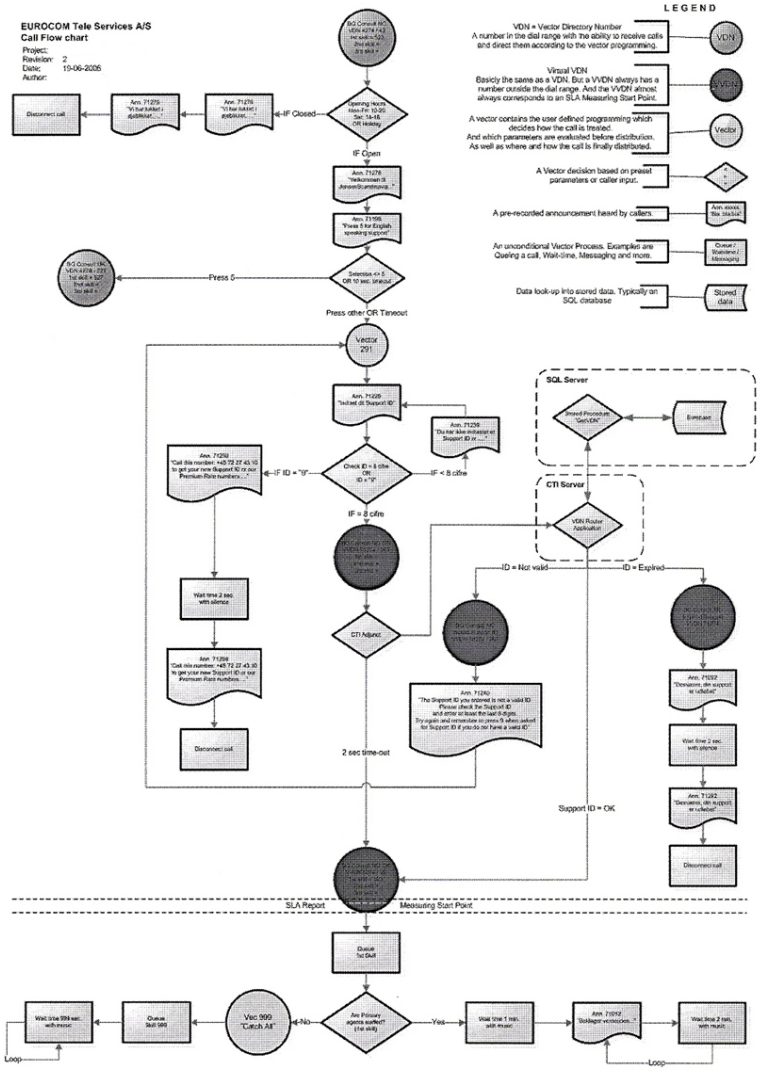


Figure A.1: Call flow chart illustrating how a call is handled.

Appendix B

SQL Database overview

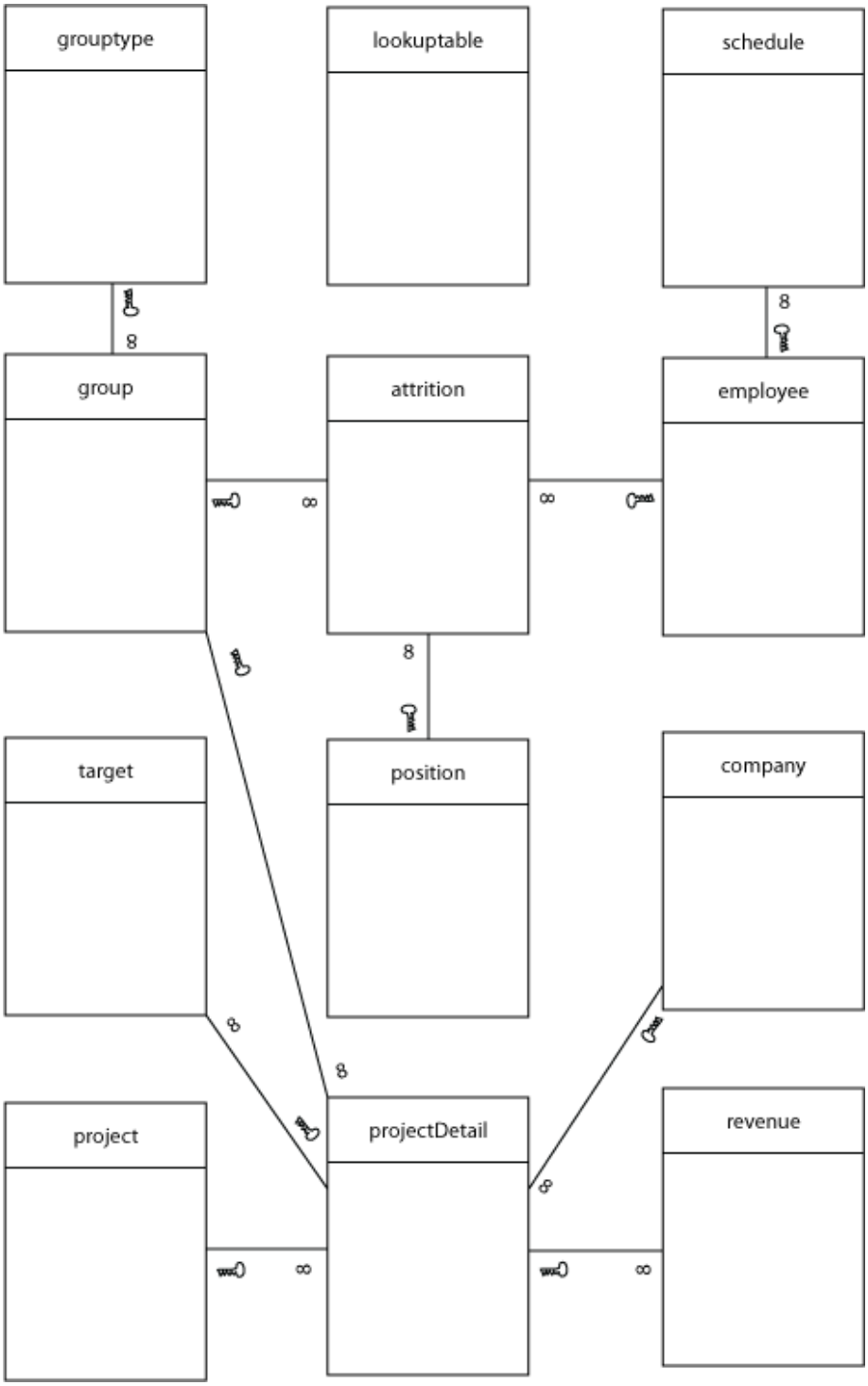


Figure B.1: Overview of the SQL Database with all tables and relationships.

Appendix C

User manual

Web application user manual

This is a user manual for the web application. Currently the application is in “beta” mode and therefore can be viewed as a working beta.

The manual is divided by section specific guidance headings and will contain scenario descriptions for each of these.

Master page / header

The top of the web site consists of a master page/header shown on all other pages/sections. It consists of a logo and welcome text along with an ASP.NET modified calendar to show week numbers next to the dates at the top right of the site. At the bottom of the header is a horizontal menu making it possible to switch between pages/sections.

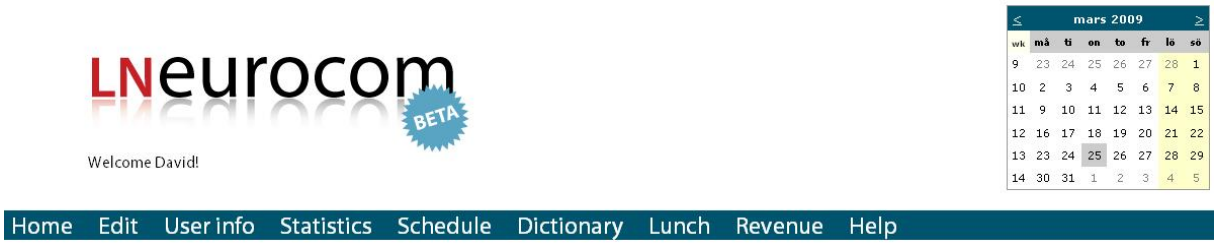


Figure 1. Shows the master/header content of the web site.

Hovering the mouse over the horizontal menu on a specific text link will highlight that specific link as shown below.

Figure 2. Hovering the mouse over a link in the horizontal menu.

Changing color of the hovered link will further aid the users to select the appropriate section.

The modified ASP.NET calendar has the possibility to show the current month and day along with week numbers. It is also possible to go ‘previous’ and ‘next’ month and view dates/weeks there.

Figure 3. Shows where the user has the next and previous options.

Home

This section is not running at the moment. The intention here is to have three bob/wob displayed. Depending on position, it will show different best of business/worst of business. An agent will be able to see their respective three best and worst of business attributes in order to percept and perform upon these measures. A team leader will see attributes non-personal on a team level instead. By clicking on the 'Home' link the user will be sent to the home section. This is also the start page within the web application always being displayed first upon start.

Edit

The edit section handles data connected with the database. Users with specific authorization will have access to this section. It is divided into administrative, management and administrative based sections.

This section consists of 'TabContainers'. One click on a tab within the 'TabContainer' will present current data for chosen tab. By selecting a row, further options will be presented. Depending on the selected tab these options will vary. Common options such as 'Edit' and 'New' will provide the user with the possibility to create new or edit existing data. Once updated, the user has the option to view that the alteration has been successful.

Figure 4 below shows when a row is selected.

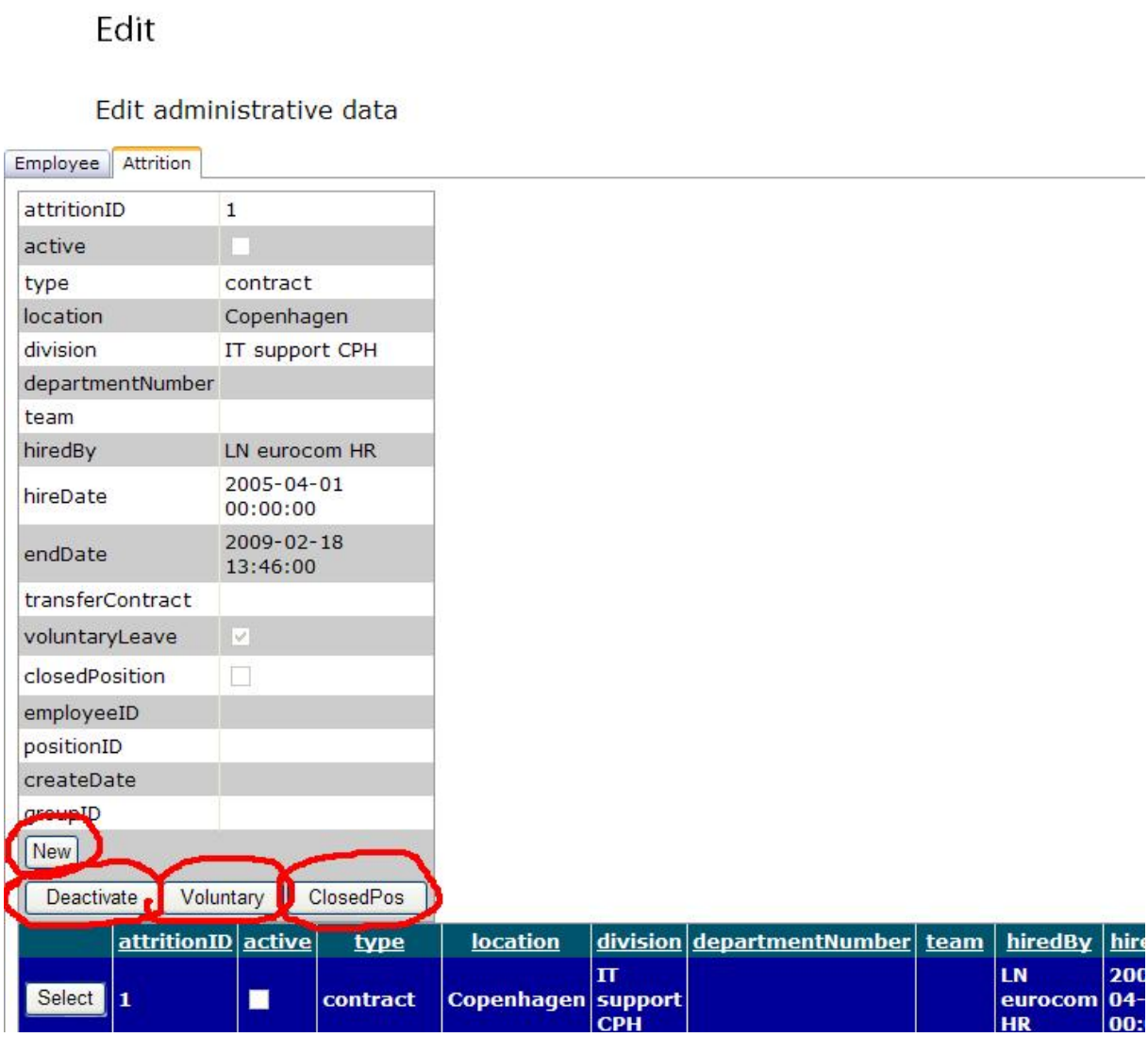


Figure 4. Shows when a user has selected the attrition tab and selected the first attrition.

Selecting an attrition gives the user four different options in this scenario. 'New', 'Deactivate', 'Voluntary' and 'ClosedPos' are the buttons available here. Pressing the new button will make it possible to create a new attrition. Pressing 'Deactivate' will deactivate the attrition selected. 'Voluntary' and 'ClosedPos' are attributes which can be ticked/unticked depending on situation.

User info

This part will replace the existing telephone listing from the intranet homepage.

The interface consists of three dropdown lists each containing available data from the database. Using cascading dropdown list function, the user has to select in a hierarchy order. Two "show all" buttons are present next to the site and team selection, making it possible to show all current data

within selection. Clicking the “show all” button or selecting a specific agent will trigger the database to present the data below.

Figure 5 shows the order of selecting and where the show all buttons are located.

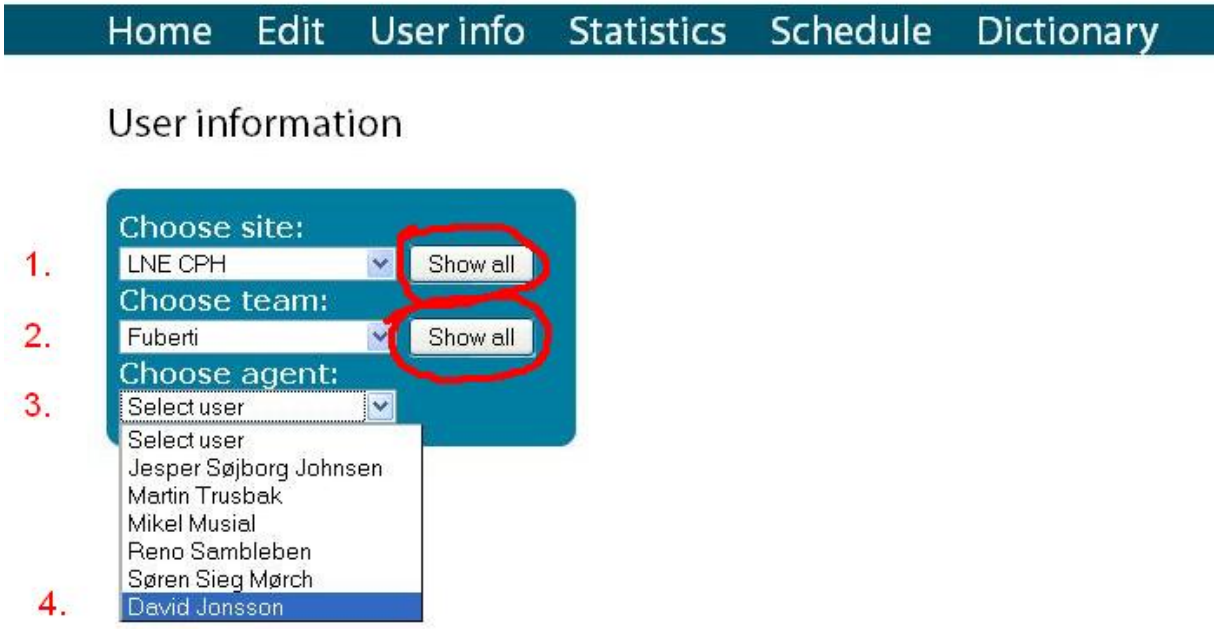


Figure 5. Following the four steps will show a specific users work information.

As seen in the figure the ‘Show all’ buttons are situated next to the site and team dropdown lists.

Statistics

Connected to the Avaya Informix CMS database, the statistics page will have the possibility to show different statistics based upon different levels. Selecting the appropriate report and team selection in combination with time and viewing detail will present data in graphical and tabular viewed data or both. This section is currently not working, but will be up and running as soon as possible.

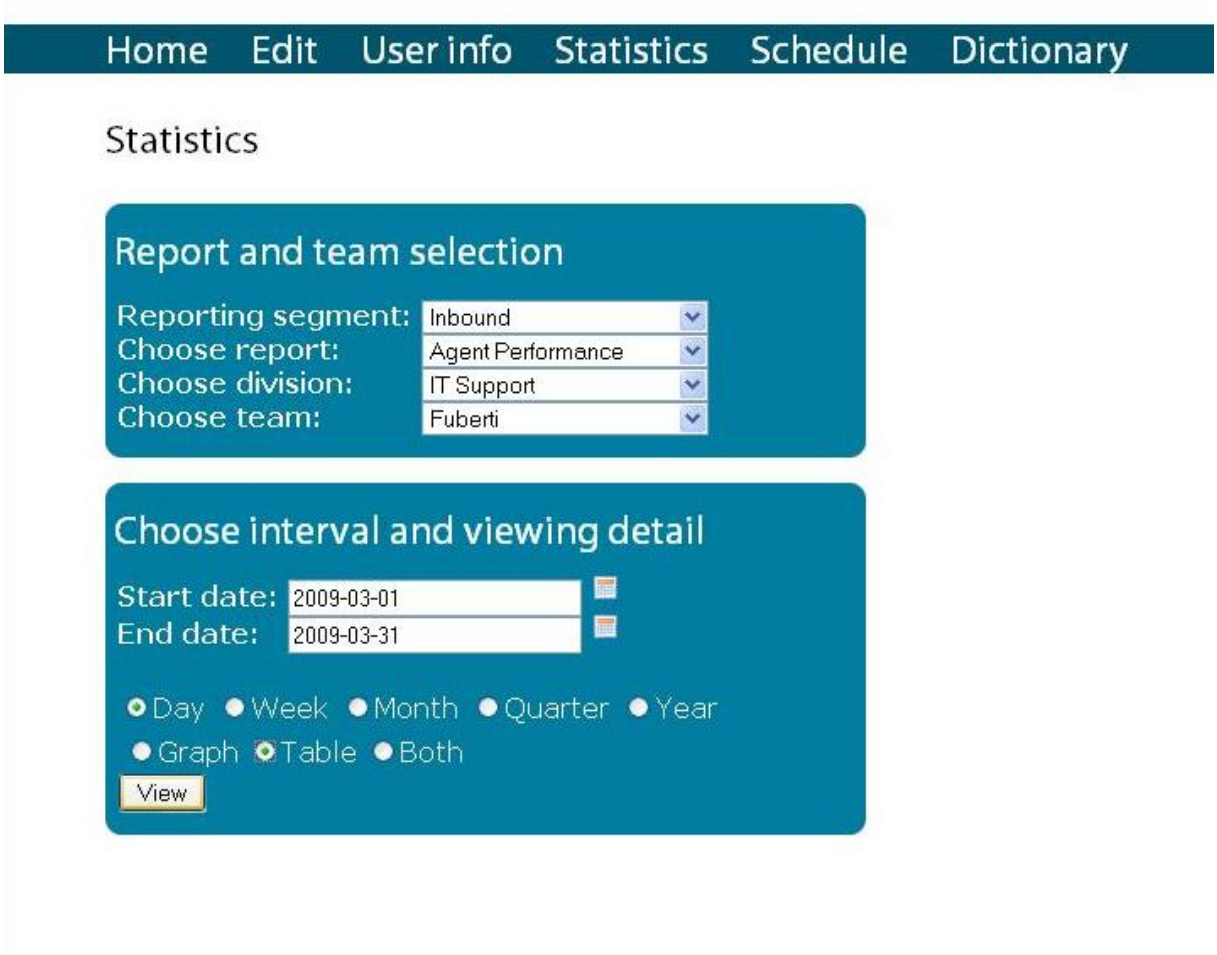


Figure 6. Showing the statistics section and how it is built up.

The user first needs to choose from the 'Report and team selection' box. At the current moment the reporting segment will only cover inbound. After choosing everything needed in that box the user has to select a time interval he/she wants to view data from within. Selecting a start and end date followed by selecting a detail view along with visualize option will present the specific data. Pressing the 'View' button will therefore trigger an update panel to show the chosen selections.

Schedule

The schedule section also consists of a 'TabContainer' with several different tabs.

View and schedule team tab

View and schedule team is a tab implemented for team leaders to insert agent schedules into the database.

If proper authorization is acquired accessing the Schedule page, the first box to appear called 'Team view' has the possibility to show existing user schedules within a time period and group selected.

The fourth tab called 'View and schedule team' has three different boxes. For inserting multiple schedules between start and end date the user can use the 'Insert multiple' box. By selecting user/users from the dropdown list pressing add the user/users will be shown underneath (as shown in the red circle). When users are set the activity type needs to be set as well along with dates and times. When all these are set, pressing the 'Insert' button will insert all users with selected reason between dates and for selected time.

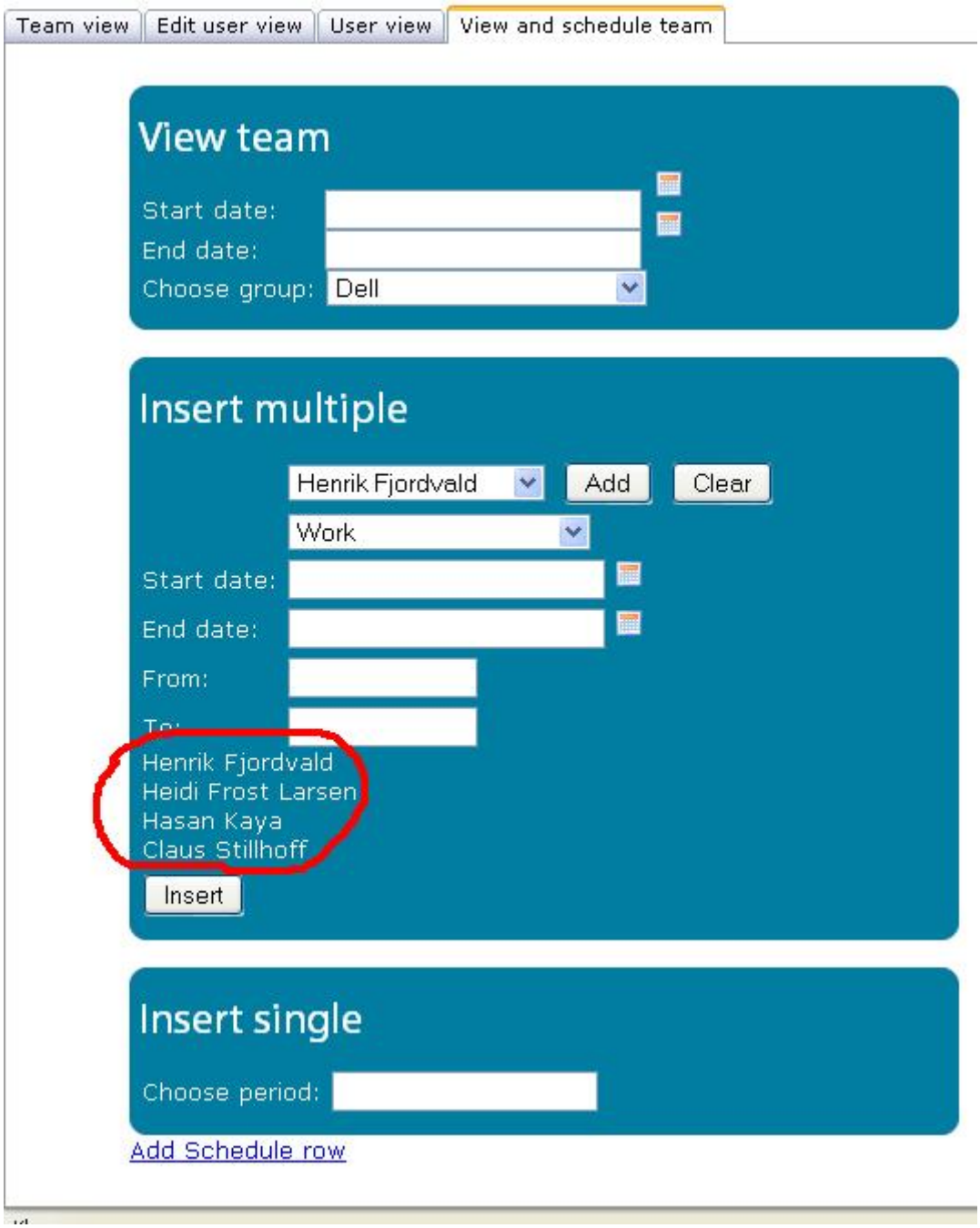


Figure 7. Shows the procedure when inserting multiple schedules.

The third box called ‘Insert single’ has the possibility to add users on a multiple level but with more work needed in order to insert every schedule. It has more flexibility and better overview than ‘Insert multiple’. Its disadvantage is that every day needs to be filled out with specific time for every user.

First step is to insert desired period in the format in *yyyy-mm*. By choosing a year along with month for scheduling makes it **monthly based**. Once a time period has been set the user has a linkbutton underneath the box called ‘Add Schedule row’ which adds a row into a monthly based insert template.

Every click on ‘Add Schedule row’ will add yet another row underneath the existing one, making it possible to insert several specific users’ combined with activity schedules at once.

Once rows have been inserted the next step is to click the empty cell of user.

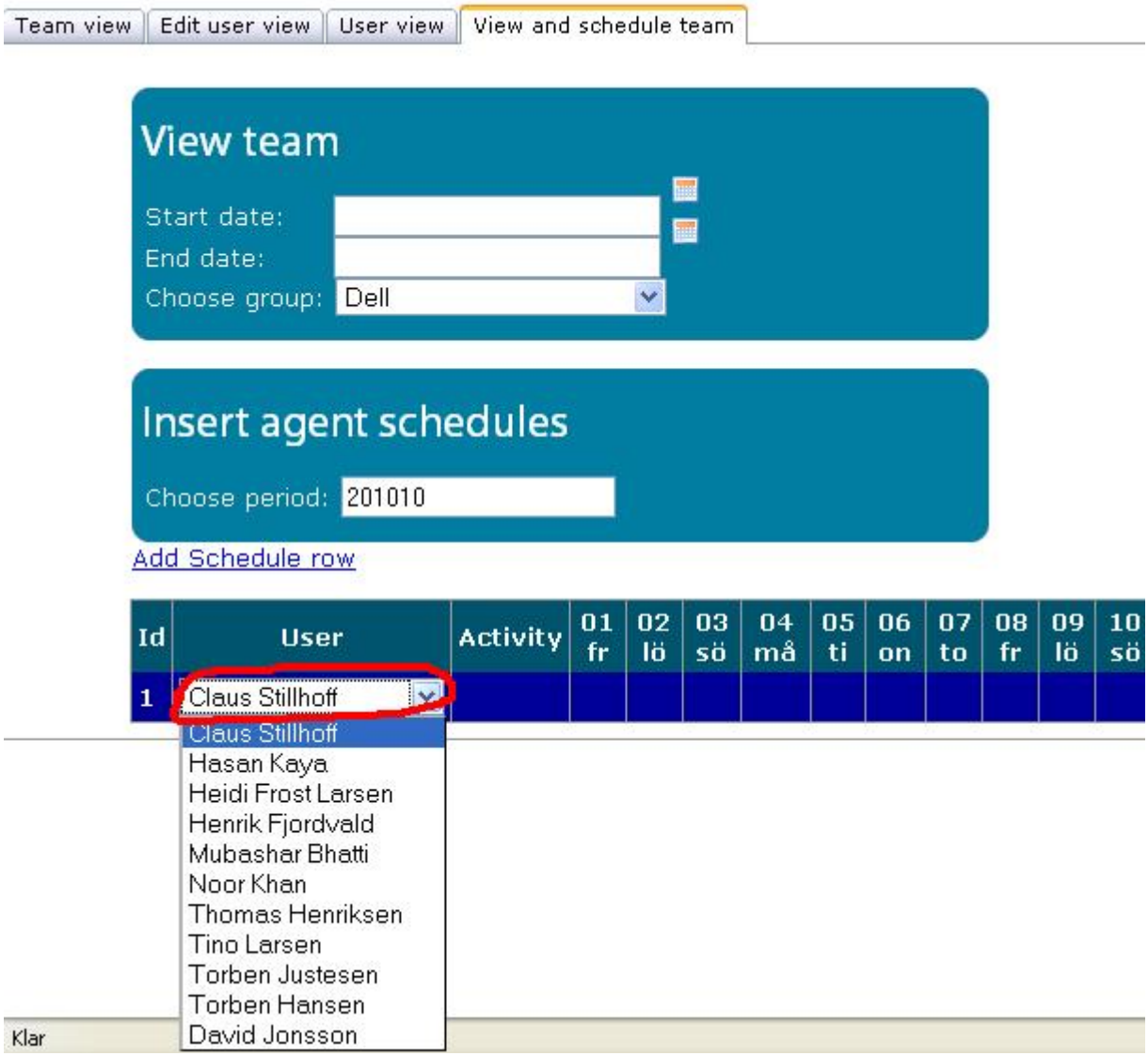


Figure 8. Shows what happens when the user cell of a row is clicked.

After selecting the specific agent to insert schedules for the next step is to select activity by clicking that empty cell. Several different activity types will then present itself almost identical to the user dropdown list with activity types instead. The final step/steps are to select dates a team leader wants to schedule the specific agent and activity within.

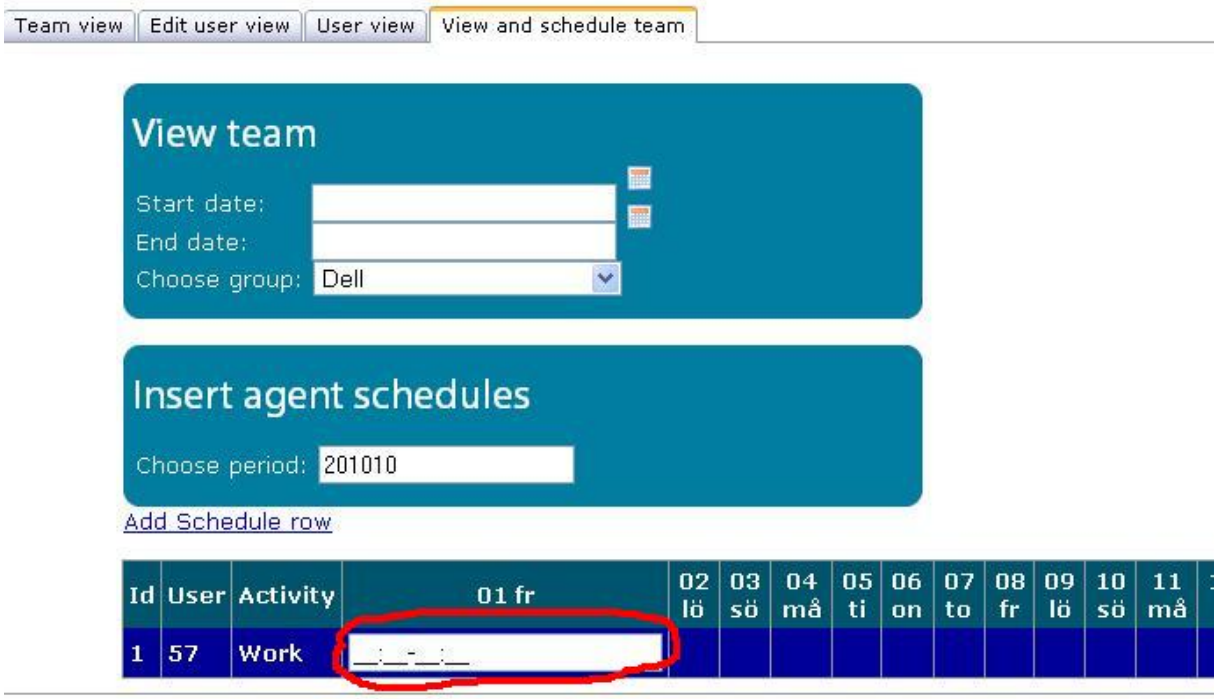


Figure 9. Inserting for a specific date will give a textbox like above.

As figure 9 shows, the team leader needs to insert from along with to time in the format e.g.:

12:00-13:00.

Once a cell has desired times the user simply needs to click the next cell to insert data into.

IMPORTANT to follow these steps in order to insert single agent schedules!

1. Select period of time for scheduling.
2. Add a schedule row.
3. Click the rows empty user cell and choose user.
4. Click the empty activity cell for that specific user.
5. Click on desired empty date inserting the from along with to time for that specific user schedule.

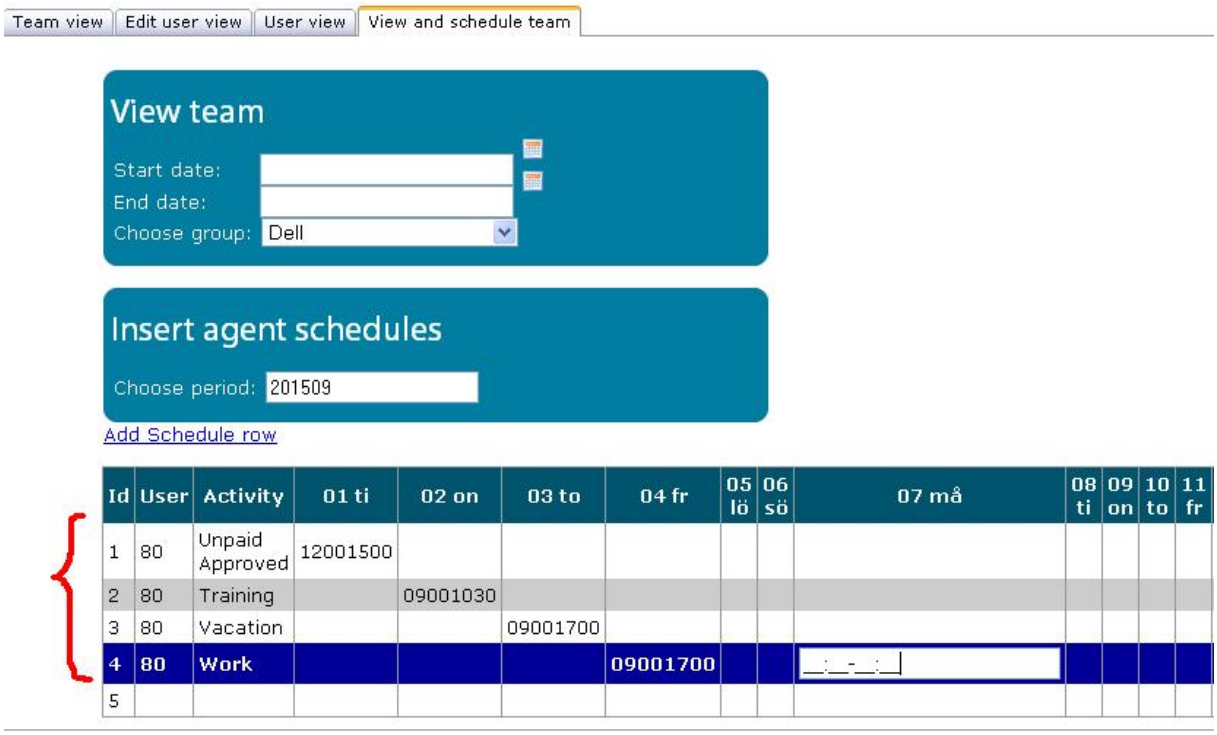
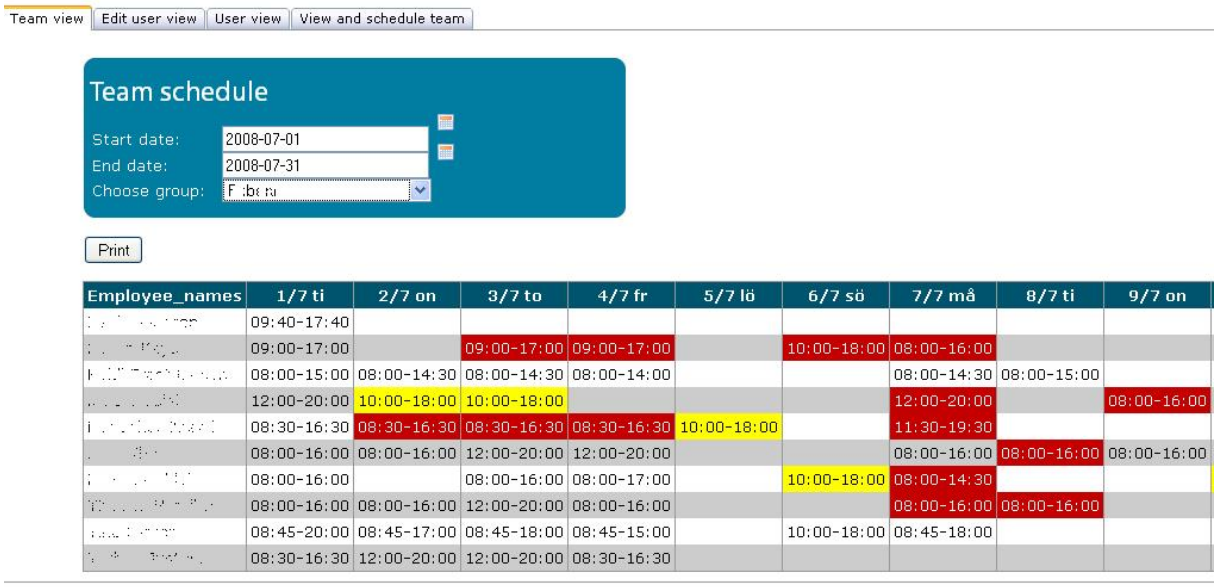


Figure 10. Example overview when same user is scheduled with specific activity for each row.

By using every row for specific user activity gives the end user a good overview over schedule while scheduling. Using the 'View team' box to assure insertions have been made.

Team view tab

Using color coding (red and yellow) the schedules will be shown either in normal, red or yellow mode depending on their status. A specific user schedule with red background will consist of a change that is associated with a non in-house work related change. E.g. the user has called in sick or in any other way is unable to attend work. Yellow background visualizes in-house work related changes. E.g. the user could be in training or other work related instances during a specific time period within a scheduled day.



Employee_names	1/7 ti	2/7 on	3/7 to	4/7 fr	5/7 lö	6/7 sö	7/7 må	8/7 ti	9/7 on
[redacted]	09:40-17:40								
[redacted]	09:00-17:00		09:00-17:00	09:00-17:00		10:00-18:00	08:00-16:00		
[redacted]	08:00-15:00	08:00-14:30	08:00-14:30	08:00-14:00			08:00-14:30	08:00-15:00	
[redacted]	12:00-20:00	10:00-18:00	10:00-18:00				12:00-20:00		08:00-16:00
[redacted]	08:30-16:30	08:30-16:30	08:30-16:30	08:30-16:30	10:00-18:00		11:30-19:30		
[redacted]	08:00-16:00	08:00-16:00	12:00-20:00	12:00-20:00			08:00-16:00	08:00-16:00	08:00-16:00
[redacted]	08:00-16:00		08:00-16:00	08:00-17:00		10:00-18:00	08:00-14:30		
[redacted]	08:00-16:00	08:00-16:00	12:00-20:00	08:00-16:00			08:00-16:00	08:00-16:00	
[redacted]	08:45-20:00	08:45-17:00	08:45-18:00	08:45-15:00		10:00-18:00	08:45-18:00		
[redacted]	08:30-16:30	12:00-20:00	12:00-20:00	08:30-16:30					

Figure 11. Shows when a user has selected a time period of one month along with team.

The user will see if present, data for that specific time period along with team selected. The color coding shows different changes that might have occurred during that time period.

The interval from start to end date can only handle an amount of approximately 31 days, making it impossible to view schedules more than around a month if lots of data is present.

Printing a schedule using Internet Explorer 7

- Press the print button under the box in the web app.
- A new pop-up window with the specific schedule will present itself.
- Right click somewhere in the new window and select ‘Print preview’.
- Press ctrl+l or the sideway printing icon, making the page becomes printed sideways (landscape).
- Select printer and press the print button.

User view tab

This tab makes it possible for the user to select and view a users' specific schedule.

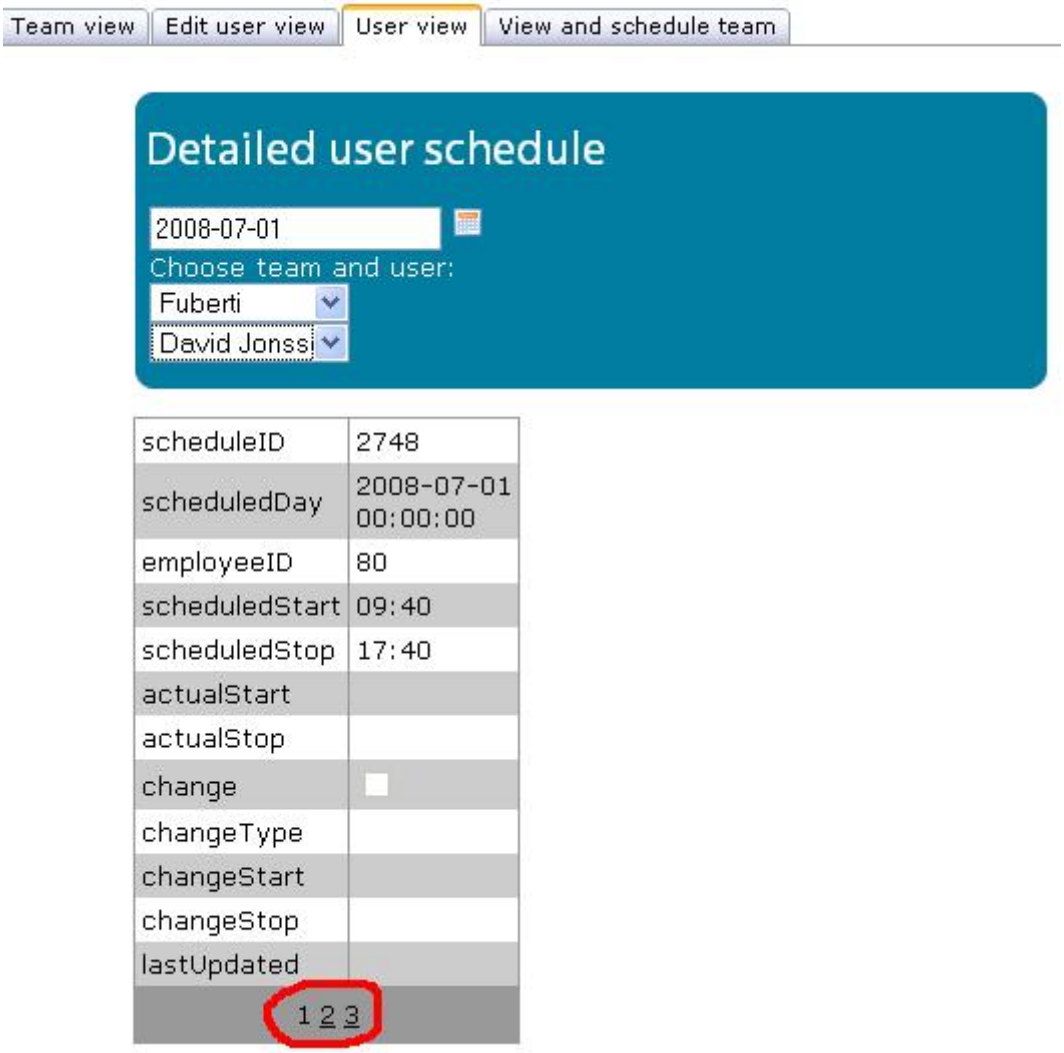


Figure 12. Shows a detailed user schedule.

Selecting a specific date and user will if exists show a detailed user schedule for that user. Note the red circle showing if more than one schedule is present. The user can for instance have been scheduled to work, but had to take training during the day finally got sick and went home earlier than expected. This scenario might generate more than just one user schedule. By clicking the different numbers will show the info regarding changes and reason along with changeStart/changeStop.

Edit user view tab

The view and edit user tab has identical possibilities as the view user tab with the exception that this tab makes it possible for team leaders to search for and edit specific user schedules.

Dictionary

The database stores a specific dictionary where specific authorization is required to add or edit within the dictionary.

All users will be able to view the dictionary page and quickly get information regarding specific work related lingo used, by using the 'quick links' option the user will be able to reach a specific 'lookup word' in a timely manner. Clicking the specific word will trigger the explanation and present itself underneath. The word headers have show/hide functionality making it possible to view several lookup words at the same time. Clicking once shows and yet another to hide.

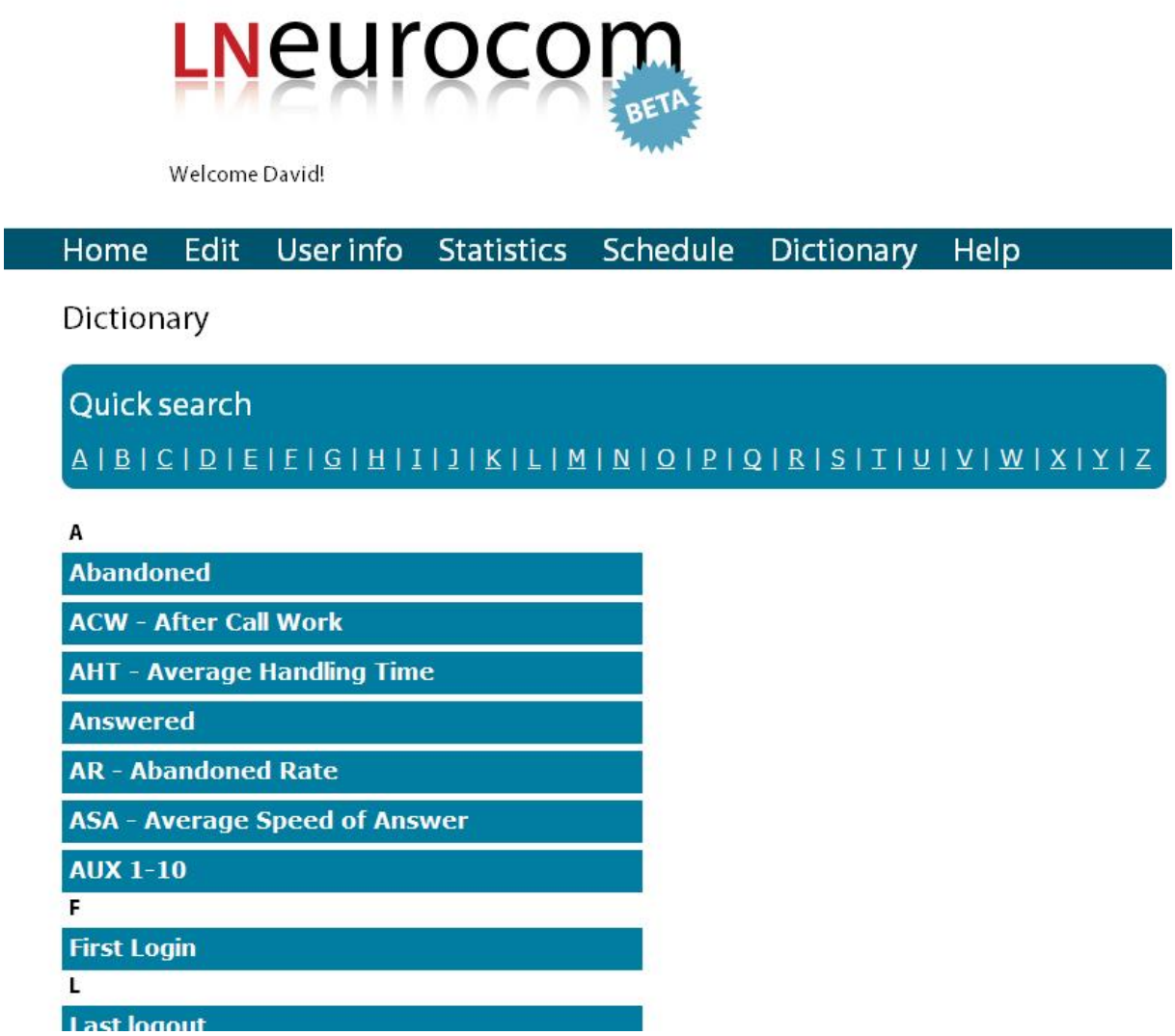


Figure 13. When the user presses the 'Dictionary' link from the horizontal menu, this will be visualized.

Clicking e.g. the 'Abandoned' rectangle will trigger a panel to display the lookup text connected to the abandoned header.

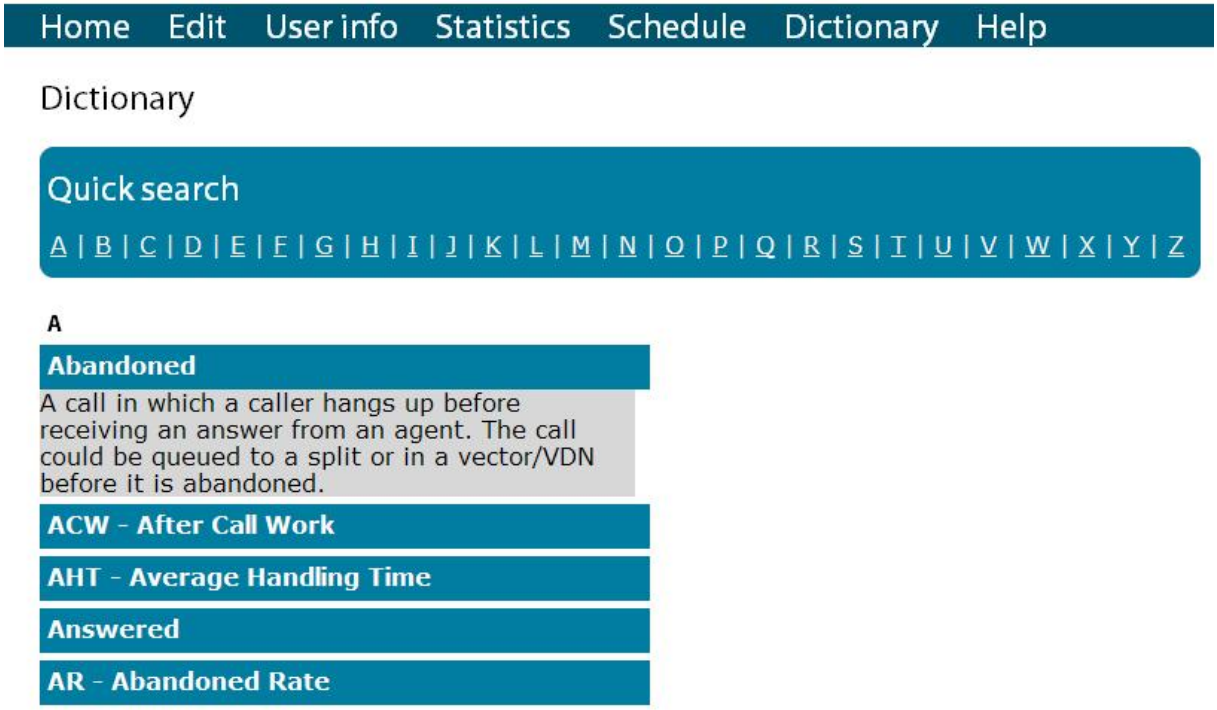


Figure 14. User has clicked the abandoned header and the explanatory text appears underneath.

Lunch

The lunch section has two tabs, one for lunch administration setting the weekly menus and one for users viewing the lunch menu.

When a user visits the lunch section it will automatically fetch the current weeks' menu. It is also possible to look forward/back in time to see what has been served/will be served by selecting year and week from two dropdown lists.

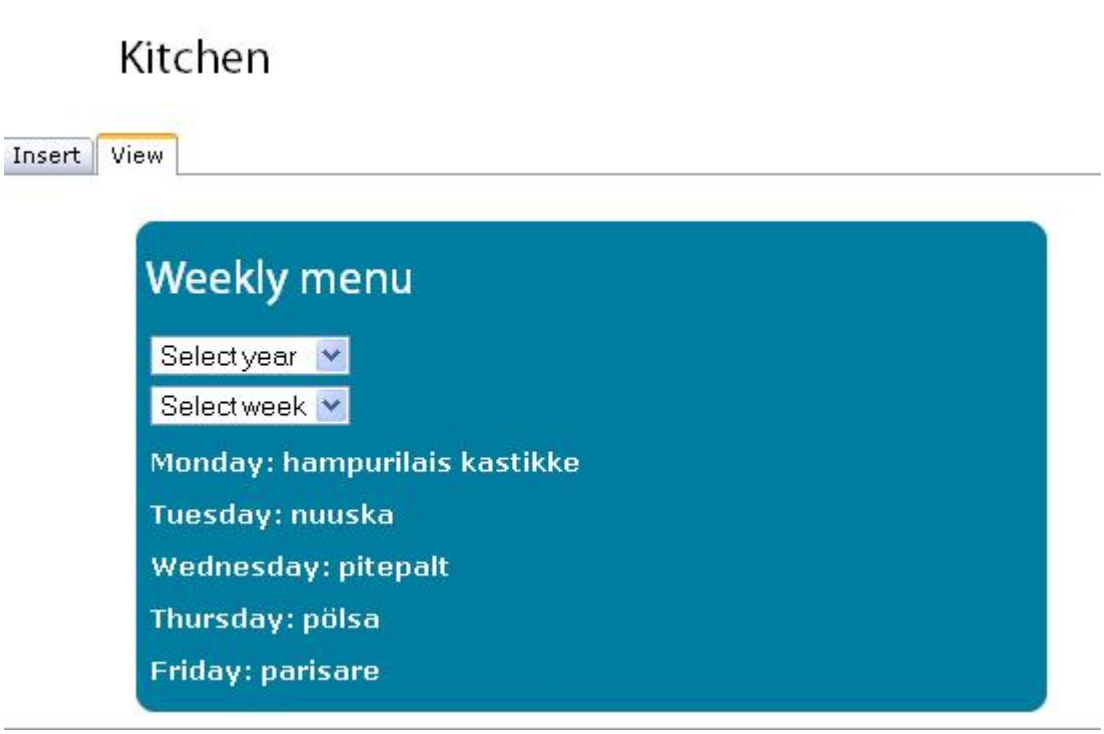


Figure 15. Shows how the view part of the lunch menu looks like.

Help

This is the section where this document is published in pdf format. ☺