Web Content Management System Powered by Ajax

Carl-Ola Boketoft

April 8, 2010
Master's Thesis in Computing Science, 30 credits
Supervisor at CS-UmU: Håkan Gulliksson
Examiner: Per Lindström

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

Abstract

This master thesis is about the development of a web site for three restaurants in South Korea. For the maintenance and updating of the contents of the web site a CMS (Content Management System) was developed.

The major part of the work was conducted in South Korea and in close collaboration with the owners of the restaurants in order to improve communication possibilities and to be able to gather materials for the project.

A theoretical study was conducted to acquire knowledge about the web development technique Ajax, both from a programming view and how to use it for improving usability of the web site functions.

From the results of the theoretical study, methods were chosen and applied into a detailed conceptual model of the web site. This model was used as a base for the implementation of the web site pages and content management functions.

The resulting implementation of the web site is based on a common structure for the different pages, with contents available in both English and Korean. The updating of the contents is done by Ajax enhanced functions in the CMS, which also contains functions for image uploading, text editing, and layout management.

Contents

Chapter 1

Introduction

This report describes the work to develop a web site for three restaurants, located in the South Korean city Gongju, and a CMS (Content Management System) for the maintenance and updating of its contents in a user friendly way.

1.1 Task of the Project

The main task of the project was to create web pages for each of the three restaurants and a system that provides an easy way for updating all the texts and pictures on the pages. At the initiation of the project it was planned that the contents of the web site should include the restaurants' menus, directions to the restaurants, pictures related to the restaurants, and welcome messages from the owner.

The focus of the research and technical study of the project was how to use Ajax enhanced functions to improve the usability of the CMS interface on the web site. Ajax is the name for a web site programming method that use different programming languages such as HTML, JavaScript, and XML to extend the functionality of traditional web sites. An integral part of the Ajax programming method is a JavaScript object called XMLHttpRequest, which can be used to update specific contents of a web page without the necessity of reloading the whole page.

The city of Gongju is historically very important for Korea, as a former capital of one of three ancient kingdoms in the country, and has several ancient landmarks that are popular tourist attractions. In relevance to this, the owner of the restaurants wanted the web site to be available in both English and Korean in order to extend their marketing to reach outside of the local audience, and to attract the attention of both Korean as well as foreign tourists. With this web site the restaurants are for the first time presented at the web.

Since none of the restaurants have staff that are experienced in working actively with the Internet before, it was important for the owner that the maintenance functions of the web site was made to be easy to use. To satisfy the need of a web site that is fully maintained without any programmer or web designer inside the company, much effort has been laid on developing the functions and design of the web site to be intuitive and easy to use. A theoretical study was conducted for the project, about how Ajax functions can be used for improving usability aspects, and the learnings from the study were playing a big part during the implementation of the CMS functions on the web site.

In an early interview with the restaurant owner, the structure, basic functions, and main contents of the web site were agreed upon. The interview then served as a base for the creation of a requirements document that was produced to describe all of the requirements for the web site. The requirements document can be seen in Appendix A. The integral parts of the development process of the web site were to decide the range of contents, designing the aesthetic and functional parts, implementing the code, and to take the photographs that were going to be featured on the pages. The owner took on the task of producing the texts to include on the pages of the web site.

1.2 The Employing Company

The employing company for the project runs three restaurants which are all located in buildings owned by the company. The first restaurant that was opened is called "Cape Town", a reference to the South African city with the same name. Cape Town was opened in 2000, and they are serving western style food such as various pasta dishes, steak based meals, and different types of coffee drinks. The restaurant has three floors, and the tables on the third floor have a great view out over the Gongju river and surrounding mountains. The restaurant "Ye-Ga" was the second restaurant opened by the company, in 2002, and the name of the restaurant means "The Ye family", which is the family of the owner. The restaurant is located in a building right in front of Cape Town, and it consists of a three story building with seating space for about 300 guests. Ye-Ga serves a selection of the most popular Korean dishes. The third, and youngest, restaurant was opened in 2004 and is called "Ye-Ga Chon". The restaurant is located in a one story building by the Gongju river, a couple of kilometers from the other two restaurants. Ye-Ga Chon serves different types of Korean food, including dishes based on chicken, duck, and dog meat. Close to the restaurant is the a beautiful and historically very important fortress called "Gonsanseong fortress".

1.3 Sections of the Report

The report is divided into the following chapters:

Chapter 1 - Introduction
Introduction of the report, describing the task of the project and the employing company.

Chapter 2 - Problem Description
Describes the background of the project, the problem statement, and the purposes of the project from different perspectives.

Chapter 3 - Methods
Describes the development process for the project, which preparations that were made before the development, and which tools were used during the project.

Chapter 4 - Theoretical Study
Contains a theoretical study of the web developing technique Ajax and the different web programming languages it comprises. Then there is a study of different usability aspects related to web interfaces, and finally it is explored how Ajax can be used for such interfaces

with the purpose of enhancing usability.

Chapter 5 - Accomplishment
Describes the different steps of the development process of the project. The chapter contains sections about the time plan for the project, the development of the aesthetic and functional design, the implementation of the code, and the user testing during the project.

Chapter 6 - Results
Presents the finished web site and describes its implemented functions. Contains screen shots and explanatory image sequences.

Chapter 7 - Conclusions
Conclusions that can be made from the project.

Chapter 2

Problem Description

This chapter describes how the idea for the project came up, the forming and description of a problem description, and the purposes of the project both from my perspective and from the perspective of the employing company.

2.1 Background of the Project

Some years before the project, I had been to South Korea for the first time to visit the Ye family due to personal relationships. During my stay there I was a frequent visitor to the restaurants and I became fascinated with the immense popularity they enjoyed in the city. The first idea for the project arose as I was discussing with the family why the restaurants were not presented to a wider, possibly international, market of tourists and out of town visitors. A discussion came up about the importance of using the Internet as a channel of spreading information to a vast range of potential restaurant customers. The restaurant owner had earlier had thoughts about outsourcing the development of a simple web site for the restaurants to a Korean company that specializes in web development.

Some time after our discussion about presenting the restaurants on the Internet, I got the idea of me developing a web site for the restaurants as my master thesis. A proposition for the project was formed and presented to the restaurant owner, who became interested in the idea. The proposition was accepted and it was decided that the development of the web site would take place on location in South Korea, in close collaboration with the owner.

2.2 Problem Statement

Since the company did not have any employee with the competence to create a web site for the restaurants, and there was no set plan to have the project executed by another company, it suited the owner fine that the web site would be developed as my master thesis. It was agreed upon that the head responsibility for the project was laid upon me and a framework for the project was discussed and set in the initiation process of the development. During an interview with the owner, a list of key elements and contents for the web site was presented and then laid as a foundation for the creation of a requirements document for the project. A selection of requirements for the web site, from the requirements document as available in Appendix A, are presented below:

- There shall be a welcoming start page for each of the three restaurants, containing a welcome message written by the owner.
- There shall be a page with information about the restaurant, for each of the three restaurants.
- There shall be a page with the restaurant's menu, for each of the three restaurants.
- There shall be a page with directions to the restaurant and pictures from the area surrounding the restaurant, for each of the three restaurants.
- All relevant contents on the web page shall be available in both Korean and English.
- The web site shall be able to view from a guest interface and from an administrator interface.
- All content data on the web site shall be possible to update by an administrator via a CMS in the administrator interface.
- The CMS functions in the administrator interface shall appear as a layer of functionality for updating the contents, on top of the guest interface.
- Usability principles and heuristics shall be in focus when when designing the guest interface and the administrator interface for the web site.

2.3 Purposes

The main purpose of the project, from my view, was to get insights and experience from carrying out an international project, as well as getting an increased knowledge about how to design a web site interface with functionality based on Ajax techniques from a perspective of usability principles and heuristics.

The purpose for the employing company was mainly to expand the availability of information about the restaurants to people inside and outside of the city borders. With the web site available in both Korean and English it would become possible to reach a larger potential customer group, compared to only having information in Korean. Having an easy to use CMS for updating and maintaining the contents of the web site was essential since none of the employees within the company had previous experience or competence to be in charge of advanced web development or programming.

Chapter 3

Methods

The first step of the project was to come up with the main ideas and concepts for the web site, and then to present a specification of the plans to the involved persons. This was very important so that everyone would be in agreement about the framework for the project. Then a plan for the continuing development process and methods for conducting the work with the project was prepared.

The following sections of this chapter describes the development process that was followed during the work with project, which practical and theoretical preparations were made before the development, and which tools were used during the development.

3.1 Development Process

During the project, a development process described in Soren Lauesen's book "Software Requirements"[10] was used to get a good work flow and to make sure that the requirements and goals for the project were going to be fulfilled. The different steps of the development process that were taken in the project are described below:

Project inception - Initiate the development of the product. Stakeholders and goals are identified.

Elicitation - Define development goals together with the stakeholders. Commence the elicitation of requirements.

Writing the requirements - Write down and finalize the requirements in a document that specifies the requirements.

Design and programming - Development of the product, according to the requirements document.

Acceptance test and delivery - The product is installed and tested by the customer and developer to confirm that it satisfies the goals and works as described in the requirements document.

3.2 Preparations

There were substantial theoretical and practical preparations for the project, before the development of the product commenced. When the project proposition had been approved by the owner of the restaurants, it was possible to begin with the preparations for the coming development of the web site.

3.2.1 Theoretical Preparations

After it had been decided that the focus was going to be on Ajax enhanced web site development, the theoretical study about the different constituents of the Ajax technique was conducted. An extensive study including the reading of articles, learning about current web site developing trends, and exploring existing web sites, took place. Then the essential web developing languages and methods encompassed within the Ajax developing technique, including HTML, XML, DOM, and JavaScript, were explored and described in the theoretical study in Chapter 4.

3.2.2 Practical Preparations

As the project was going to be performed on location in South Korea, a big part of the practical preparations were to organize the trip there and getting settled in an apartment to conduct the work from. While getting acclimatized and organizing for the project, there were many opportunities to visit the restaurants, take photos of things related to the restaurants, trying out the food, and discussing different aspects of the project with the owner. Another important part of the preparations was to learn how to use the different programming languages that were explored in the theoretical study. I had a gained significant level of competence in PHP programming from earlier projects, but I was unfamiliar with JavaScript, XML, and DOM.

Apart from the preparations that were performed in direct relation to the project, I had already taken some photos of the restaurants, food, and the surrounding area during my previous visit to South Korea some years ago. Some of those photos were later presented on the web site.

3.3 Tools

Many different tools and programs were used during the development of the product. The following list are some of the most frequently used:

Adobe Photoshop - Image processing software that was used for photo editing, refining concept models and create graphics for the web pages.

Adobe Dreamweaver - Web development software that was used for the coding of the web site. The program is based on both coding and WYSIWYG (What You See Is What You Get) development.

Adobe Illustrator - Vector based graphics software that was used for the creation of concept models, restaurant logos, and other graphics for the web pages.

WAMP - A collection of programs that were used to set up a private web server on a computer for testing the web site while coding. WAMP stands for “Windows, Apache, MySQL, and PHP”.

Apple Safari - The web browser that the CMS interface was mainly created for. Specific features of the browser made it suitable for the Ajax functions that were implemented.

Google Docs - Web based word processing software that was used for writing documents related to the project. This tool was very useful since different computers were used during the process.

Canon EOS 50D - DSLR (Digital single-lens reflex) camera that was used for taking the photos featured on the web pages.

WinShell - $\LaTeX$ editing software that was used during the writing of this report.

Chapter 4

Theoretical Study

This chapter contains an extensive theoretical study about the web developing technique known as Ajax, web development usability principles, and how Ajax and the usability principles can be used together for the creation of user friendly Ajax enhanced functions on web sites.

4.1 Introduction

Internet has evolved from being a source of read-only information and viewable home pages into a highly interactive working place for both private persons as well as global corporations. The original building blocks and technologies that once served as the base of the contents presented on the World Wide Web are still present as a solid foundation for the online materials, but developers have desperately tried to catch up with the rapid development of new functionality and the demands from innovative web site developers. As a consequence of the vast number of web site developers and users, new platforms and techniques to meet the demands have appeared in many different forms.

As web sites move from being administered mainly by experienced and skilled moderators and developers to interactive user forums, the importance of user friendly design comes into focus. If a web site has advanced functionality that is supposed to be useful and helpful to the user but is not designed in a way that it can be understood or accessible, the functions might become obstacles instead of useful tools. With the purpose of mimicking the functional behavior of desktop applications, a new breed of web applications called RIA (Rich Internet Application) has emerged. The techniques for RIA have popped up in many different forms and appearances that have different requirements on both hardware and software. This has resulted in a problem of accessibility, since there are many different types of Internet users and a great variety of software and hardware platforms by which users can access the web.

In the first part of this theoretical study the RIA technique called Ajax, which is used for web sites with interactive user interfaces, is explored and described. The second part of the study explores different usability principles of web site development. The final section describes how Ajax functionality can be used for the purpose of enhancing web site functions from usability aspects.

4.1.1 Model of Traditional Web Applications

Traditional web applications are based on a model known as the client-server model, using HyperText Markup Language (HTML) as the main programming language. The client-server model can be described as a communication method where there is a service requester (the client) which displays information for the user and makes requests to a service provider (the server) that does all the work in terms of computing, processing, and delivering data. Web browsers are used as passive display consoles on the client side [13]. For further enhancement of web site functionality, a range of server side modules can be used for scripting together with dynamic web pages with contents reflecting who is visiting the page and what data is being passed to the page by using query strings. Considering the simplicity of this model, traditional web development is relatively uncomplicated [4].

One fundamental aspect of the server-client model is that all processing is being handled by the server, and that there is a necessity of constant reloading of contents to present updates and changes to the state of the displayed web page [6]. When the user interacts with the web application, a request is sent from the client to the server, which in turn returns a new page that is displayed by the client. While waiting for the server to respond to the client, the user can not do anything with the web application, but to wait. As a result of the nature of this model there is an abundance of data being passed from the server to the client, which increases unwanted waiting time while data is processed and page updating occurs. HTML was not constructed to be used for rich interfaces, but for passive display of information [4].

4.1.2 Rich Internet Applications

The defining aspect of RIA technologies is their ability to distribute to the browser the rendering and computing abilities of the server side scripting engine that updates displayed information and changes in the user interface on the web page [4]. RIA is used to improve browser-based user interaction by increasing client-side working power and processing capabilities, and thus reducing the amounts of requested data to be delivered from the server to the client [13].

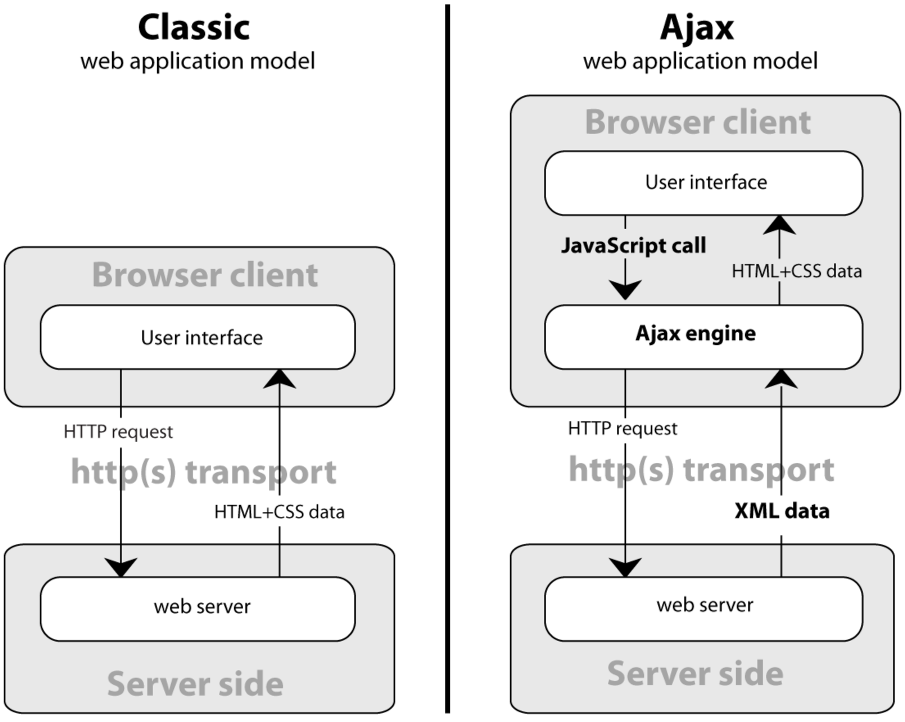
There are different types of RIA applications, but a common property is that they all have the ability to give the client side new functionality to request only specific, relevant, parts of data during a page update and to locally modify the user interface to display the changes [4]. Since the client has been enabled to request only relevant data from the server, code that has not been changed in the interface will not be sent to the client more than once, which leads to that the amounts of network traffic generated can be significantly reduced. The client still communicates with the server, but only the updated parts of the web pages are requested as opposed to repeatedly requesting an update of the entire page [4].

One of the main goals when creating RIA is to keep data between client and server synchronized and to enhance the user interface action by allowing it to update in direct response to user action [4]. The result of the introduction of this feature is that the user gets the impression that tasks are being carried out the same instant as the very initiation of the user request [4].

4.2 Ajax

From a number of possible RIA development techniques to focus on, the technique known as Ajax (Asynchronous JavaScript and XML) was chosen for the project. The main reason that Ajax was chosen was to explore its innovative uses for dynamic web contents, such as various dynamic functions at the web sites Google Maps and GMail. Another important reason is that Ajax can be used by most modern web browsers that have support for JavaScript, without the necessity of downloading a third party plug-in software to function.

Ajax is not a technology itself but rather a term for a combination of interrelated technologies that together make up a powerful tool for web development. As one of the methods of RIA web page programming, Ajax makes it possible to create web applications with improved interactivity and responsive user interfaces [6]. The client-server model is supplemented with a new intermediary; the Ajax engine. Instead of starting the web browsing session by loading a web page, an Ajax engine is started. An illustration describing the Ajax model is shown in Figure 4.1 [6]. To the left in the figure is the classic web application model, where all new information for the web browser must come directly from the server side. To the right in the figure is the Ajax web application model where the Ajax engine has been introduced [6]. The job of the Ajax engine is both to render the user interface and graphical representations as well as to handle the communication with the server side. This way, the Ajax engine allows updates of the user interface and communication with the server to occur asynchronously, independently of each other. The result is that the user never has to wait for a blank browser window that is waiting for requested information from the server to be retrieved by the client [6].



Adapted from figure by Jesse James Garrett / adaptivepath.com

Figure 4.1: *The classic web application model and the Ajax web application model [6].*

4.2.1 W3C Recommendations and Ajax

The W3C (World Wide Web Consortium) is the principal organization for developing standards for the World Wide Web. The organization was founded, and is still headed, by Tim-Berners Lee in 1994, the inventor of the World Wide Web. The mission of W3C is "To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web" [20]. To create a path for web developers and programmers towards using the World Wide Web to its full potential, allowing any web related hardware and software to work together, the W3C develops protocols and guidelines that will secure the future growth of the web. These documents are called W3C Recommendations, which are under a royalty-free patent license and thus allowing anyone to use them freely [20].

The existance of a W3C Recommendation for a technique declares that this technique has reached a certain level of conformance, which likely makes it compatible with the hardware and software common among the great mass of web developers and users.

4.2.2 Ajax Building Blocks

The following sections are dedicated to the introduction of and to some extent describe, the different web building technologies that constitute the building blocks of Ajax.

A summary of the area of use for the different technologies for web page development:

HTML and CSS are used to present contents on a web page.

XML is used to store and transport data.

XHTML is a combination of HTML and XML.

JavaScript is the scripting language that binds all the techniques together.

XMLHTTP is a JavaScript object used to enable asynchronous data exchange between client and server.

DOM serves as an interface which allows dynamic changes to be made to the contents on a web page.

4.2.3 HTML

HTML (HyperText Markup Language) is a markup language that is based on a set of markup tags that are used to describe HTML documents, also known as web pages [24]. HTML was developed and described by Tim-Berners Lee in the early 1990s and it was part of the rapid growth of the World Wide Web due to its simplicity as the main publishing language of web pages [24, 11]. With HTML, it became possible to present information to a global online audience as it was designed with the purpose of being an universally understood language, a common language for all computers to read [21]. The language gives the means to publish documents including texts, images, tables, lists, and more. The means of navigating between different HTML pages is in the form of hypertext links, activated by the click of a button [21].

HTML documents consists of HTML tags and text. The HTML tags are simple keywords within angle brackets like <html>, and they usually come in pairs, like <p>, the start of a tag, and </p>, the end of a tag. The example in code Listing 4.1 shows how a simple HTML document can look like and gives some examples what tags can be used for.

```
1 <html>
2 <body>
3   <h1>This is a heading</h1>
4   <p>This is a paragraph</p>
5 </body>
6 </html>
```

Listing 4.1: *Line 1 and 6) Text enclosed inside the tags <html> and </html> tags describe the web page.*
Line 2 and 5) Text enclosed inside the tags <body> and </body> is the contents displayed on the browser.
Line 3) Text enclosed inside the tags <h1> and </h1> is displayed as a heading.
Line 4) Text enclosed inside the tags <p> and </p> is displayed as a paragraph.

4.2.4 CSS

CSS (Cascading Style Sheets) are used to define how to format and display HTML elements [24].

When tags for formatting colors and fonts was introduced to the HTML 3.2 specification, it unintentionally became a source of complications for web developers. In web sites with large amounts of pages and contents, these new tags had to be added to every single page where the formatting had to take place, which could turn into a very long and expensive process. As a solution to the problem with this decentralized tagging, the W3C developed CSS. The introduction of CSS in the specification of HTML 4.0 made it possible to separate the appearance formatting code from the HTML document, to be stored in an external CSS document [24]. The biggest advantage of CSS was that it enabled developers to change the appearance and layout of any amount of pages in a web site, only by editing a single external CSS file.

In addition to the external CSS file, there are two alternative ways to use CSS [24]. One is the internal style sheet, included in the head of the HTML document (enclosed within the <head> and </head> tags), which may be used when one page of a web site should have a unique formatting. The other way is to use an inline style to locally format a specified part of a HTML page, for example a paragraph. The inline style should normally be avoided because the advantage of separating content from style is lost.

The example in code Listing 4.2 shows how an external CSS file can be included in a HTML file, by using a specific <link> tag, to add formatting to text.

```
1 <html>
2 <head>
3   <link rel="stylesheet" type="text/css" href="mystyle.css" />
4 </head>
5 <body>
6   <p>This paragraph is blue</p>
7 </body>
8 </html>
```

Listing 4.2: *Line 3) Inside the <head> tag of the HTML document, a specific <link> tag is used to include a CSS file named "mystyle.css".*

The example in code Listing 4.3 shows how an external CSS file (called "mystyle.css" in this example) can look.

```
p {color:blue}
```

Listing 4.3: *The CSS syntax consists of three basic parts:*
Selector: usually the HTML element that you want to format. In this example it is "p" (paragraph).
Property: the attribute that will be formatted. In this example, the color will be formatted.
Value: the value of the of the property. In this example, the value of the color is blue.

When opening the HTML file in a web browser, the paragraph has successfully been formatted to contain blue text when displayed in a web browser, as shown below:

This paragraph is blue

4.2.5 XML

XML (eXtensible Markup Language) is a markup language that was designed to store and transport data. XML became a W3C Recommendation February 10, 1998 [24]. The XML files are simple text files that can be created and edited with any software that can handle plain text. The usage of XML documents is usually to carry data that can be accessed, formatted and presented by a HTML file. Therefore it is notable that XML is not a replacement, but a complement, to HTML. Compared to storing data in the HTML file, XML comes in handy when you need to display dynamic data within your HTML document. Instead of going through the procedure of opening the HTML file to edit the data locally, you can store the data in an XML file, while keeping it separate from the layout and display formatting tags in the HTML file. This way you can create separate documents for data storage and layout descriptions, achieving a clearer and logically divided organization of materials.

XML documents consists of XML tags and text. The appearance and properties of the tags are much like those of HTML documents, but with the important difference that XML tags are not predefined, you define your own, self-descriptive, tags. In XML documents, compared to those of traditional HTML, it is necessary to include both start and end tags for each tag. The contents of the XML file does not have any function on its own, it is just a piece of information wrapped in tags [24]. To make use of the stored information, the file must be read by software, such as JavaScript, to recieve and present it.

The example in code Listing 4.4 shows what a simple XML document can look like.

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <message>
3   <to>Hanui</to>
4   <from>Calle</from>
5   <heading>Lunch today?</heading>
6   <body>Shall we meet at 1pm?</body>
7   <signature>Kindly, Calle</signature>
8 </message>
```

Listing 4.4: *Line 1: Declaration that the file is a XML document of version 1.0 and that it has been encoded with (ISO-8859-1 = Latin-1/West European character set).*
Lines 2 and 8: Declares the beginning and end of the root element of the document. In this case the root element is <message>, which indicates that the elements in this document constitute a message.
Lines 3-7: These are the child elements of the root element. In this case those elements are <to>, <from>, <heading>, <body>, and <signature>.

4.2.6 XHTML

XHTML (eXtensible HyperText Markup Language) is a combination of HTML and XML, taking all the properties of the HTML 4.01 standard and adding properties from the XML markup language. Some examples of important differences compared to HTML 4.01 are that all elements must have both starting and ending tags and that tags must be in lower case [24]. The reason for introducing the XHTML language is to take one further step towards encouraging developers to create “well-formed” documents where everything has to be marked up correctly. XHTML 1.0 became a W3C Recommendation January 26, 2000 [24].

4.2.7 JavaScript

JavaScript is a scripting language, a sort of lightweight type of programming language [24]. An easily made misconception is that Java and JavaScript are similar since their names sound the same. In fact the two languages are completely different in both concept and design [24]. Java, is a powerful and more complicated programming language, while JavaScript was designed as a means to add interactivity to HTML pages [24]. JavaScript is usually embedded in the HTML code of web pages and can read and change the contents of HTML elements. The language is designed as an integrated part of the browser on the client side, and is an interpreted language, which means that JavaScripts are executed without compilation [24]. In order to make the web browser recognize and execute JavaScript commands they must be enclosed within a <script> tag.

A JavaScript can be set to be activated by an event such as when a user clicks on a HTML element, or is entering text into a text field in a form.

If a JavaScript is supposed to be activated when a web page loads, it is put in the <body> section of the HTML page. The example in code Listing 4.5 shows how a very simple JavaScript can be used for writing output to a web page, without an activating event.

```
1  <html>
2  <body>
3    <script type="text/javascript">
4      document.write("Ice cream is delicious");
5    </script>
6  </body>
7  </html>
```

Listing 4.5: *Lines 3-5: The JavaScript command, enclosed in the <script> tag, is "document.write" which is a standard command for writing output to a web page. This command will simply output the text line "Ice cream is delicous" on the web page.*

Scripts that are supposed to be activated by an event are put in the <head> section of the HTML page. Then the Script is called and executed when the activating event is occurring. An example showing how a JavaScript can be activated by a web page event is shown in code Listing 4.6.

```
1  <html>
2  <head>
3    <script type="text/javascript">
4      function message()
5      {
6        alert("Alert enabled by the onload event");
7      }
8    </script>
9  </head>
10 <body onload="message()">
11 </body>
12 </html>
```

Listing 4.6: *Lines 2-9: The JavaScript is enclosed in the <head> section of the HTML page, and is waiting to be activated by an event.*
Line 10: The activating event in this example is that the body of the web page is loaded. When the page is loaded, the JavaScript is programmed so that an alert popup will appear in the web browser.

4.2.8 XMLHttpRequest

XMLHttpRequest is a JavaScript object that makes it possible for the client to communicate directly with the web server after a web page has been loaded, without having to reload the whole page [24]. XMLHttpRequest has not yet become a W3C Recommendation, but there is a working draft of such a document, latest updated on August 20, 2009 [23]. As described above, it was previously not possible to update information on the web page without using a HTML form, in which a user clicks a "Submit" button to activate the information exchange and then wait for a new page to load with the new information from the server. During this time the user could do nothing but wait [24]. The XMLHttpRequest object is what enables the much desired functionality of making HTTP requests and recieve responses rapidly and hidden from the user, without visual interruptions [18].

4.2.9 Ajax Engine Example

The following example (adapted from W3Schools [25]) consists of a HTML document called "testAjax.htm" with a form containing two input fields: Name and Time. The user fills out the “Name” field, and then the “Time” field will automatically be filled out by an Ajax engine that works in the background. The HTML for the form is shown in code Listing 4.7.

```
1  <html>
2  <body>
3    <form name="myForm">
4      Name: <input type="text" name="username" />
5      Time: <input type="text" name="time" />
6    </form>
7  </body>
8  </html>
```

Listing 4.7: *Line 3-6: The HTML document “testAjax.htm” contains a simple form with the two fields “Name” and “Time”.*

The core of the Ajax engine is the XMLHttpRequest object. The JavaScript function ajaxFunction() in Figure 4.8 creates an XMLHttpRequest object. This creation of the XMLHttpRequest works in most new browsers (code for solving compatibility issues with older browsers have been omitted in this example).

```
1   <html>
2   <body>
3     <script type="text/javascript">
4       function ajaxFunction()
5       {
6         var xmlhttp;
7         xmlhttp=new XMLHttpRequest();
8       }
9     </script>
10
11    <form name="myForm">
12      Name: <input type="text" name="username" />
13      Time: <input type="text" name="time" />
14    </form>
15  </body>
16  </html>
```

Listing 4.8: *Line 6: A variable named xmlhttp is created to hold the XMLHttpRequest object. Line 7: The XMLHttpRequest object is created.*

When a request has been sent from the client and is being processed by the server, there must be a function that receives the data on the client side. The “onreadystatechange” property, as shown in code Listing 4.9, holds that function. The function is then called automatically when new data is sent from the server.

The readyState property contains the current status of the server’s process of the request. The onreadystatechange function will be executed every time the readyState property changes. The different values for readyState (the current state of the request) are listed below:

State	Description
0	The request is not initialized
1	The request has been set up
2	The request has been sent
3	The request is in process
4	The request is complete

The use of the properties onreadystatechange and readyState is shown in code Listing 4.9.

```
1 xmlhttp.onreadystatechange=function()
2 {
3   if(xmlhttp.readyState==4)
4   {
5     document.myForm.time.value=xmlhttp.responseText;
6   }
7 }
```

Listing 4.9: *Line 1: When new data is sent from the server, the function in onreadystatechange is called.*
Line 3: If the readyState property is complete (4),
Line 5: the responseText property is used to recieve the data that is sent back from the server.

To send a request to a server, the open() and send() methods are used. The open() method initiates the request and it takes three arguments. The first argument defines which method (GET or POST) to use when sending the request. The second argument is the address to the server side script. The third argument declares that the request should be processed asynchronously. The send() method sends the initiated request to the server. The open() and send() methods are shown in code Listing 4.10.

```
1 xmlhttp.open("GET", "time.asp", true);
2 xmlhttp.send(null);
```

Listing 4.10: *Line 1: The open() method says that the GET method should be used, that the server side script is in the file "time.asp", and that the request should be handled asynchronously.*
Line 2: The initiated request is being sent to the server by the send() method.

After inserting the code, the HTML page looks like in code Listing 4.11.

```
 <html>
 <body>
  <script type="text/javascript">
  function ajaxFunction()
  {
    var xmlhttp;
    xmlhttp=new XMLHttpRequest();
    xmlhttp.onreadystatechange=function()
    {
      if(xmlhttp.readyState==4)
      {
        document.myForm.time.value=xmlhttp.responseText;
      }
    }
    xmlhttp.open("GET","time.asp",true);
    xmlhttp.send(null);
  }
 </script>
 <form name="myForm">
  Name: <input type="text" name="username" onkeyup="ajaxFunction();" />
  Time: <input type="text" name="time" />
 </form>
</body>
```

```
24 </html>
```

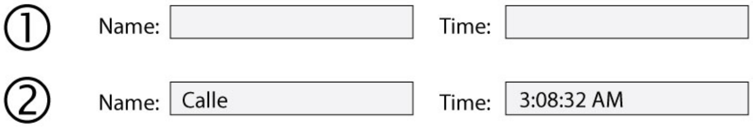
Listing 4.11: *Line 20: The onkeyup attribute within the “Name” text field, activates the ajaxFunction() when a user has entered a letter within the text field.*

The server side script in Listing 4.12 is used for returning the current time, and is stored a file called “time.asp”. The files “testAjax.htm” and “time.asp” must be placed in the same folder for the code in this example to work. The response.write(time) function in the script returns the current time when called.

```
<\%
  response.write(time)
\% >
```

Listing 4.12: *The server side ASP script, consisting of only one line of code, for returning the current time.*

When the HTML file “testAjax.htm” is run from a web server and viewed in a web browser, it will look like in Figure 4.2.



① Name: Time:

② Name: Calle Time: 3:08:32 AM

Figure 4.2: *1) Before the user has entered anything in the “Name” field, there is nothing in the “Time” field either.*
2) As soon as the user has entered text into the “Name” field, the Ajax engine retrieves the current time and displays it in the “Time” field.

4.2.10 DOM

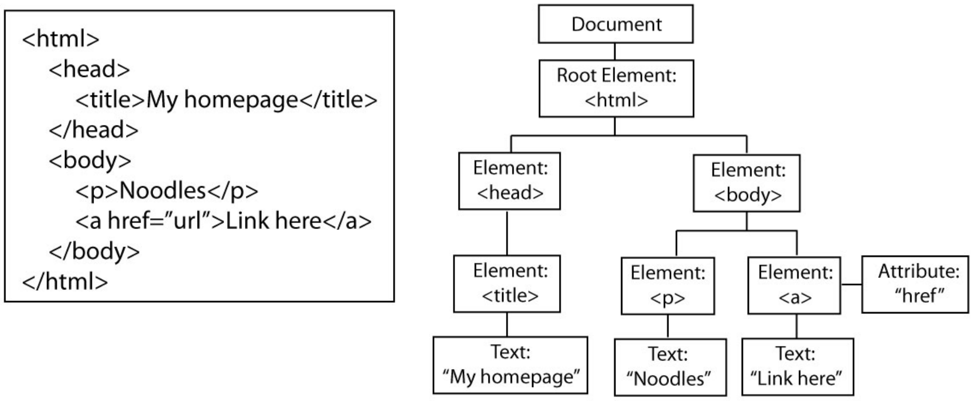
The DOM (Document Object Model) is a programming interface that describes a standard for logical structure of HTML and XML documents and how to access and modify the contents [22]. DOM Level 1 (which is concentrated on HTML and XML document models) became a W3C Recommendation October 1, 1998 [24]. Using the DOM, developers can create HTML and XML documents structured as program objects, or nodes, making it easier to develop web pages that can be dynamically modified and manipulated by users through a web browser [18]. On the web site for the W3Consortium, it is stated that:

“With the DOM, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model, with a few exceptions.” [22]

In the DOM, everything inside XML and HTML documents is defined as a node. The DOM presents the document in a tree structure, which can be visually displayed. The methods for accessing the nodes within the DOM tree can be implemented with the help of a programming language such as JavaScript. When an interface change has been made to the DOM of the web page, it will appear immediately in the web browser.

Figure 4.3 shows an example of HTML DOM. Each part in the HTML document is a node and this HTML document contains the four different types of nodes Document, Element, Attribute, and Text.

Document	The DOM tree itself, not a part of the HTML document
Element	HTML elements such as <p> and <a>
Attribute	The attribute of an element, such as 'href' in an <a> element
Text	All text



```
<html>
  <head>
    <title>My homepage</title>
  </head>
  <body>
    <p>Noodles</p>
    <a href="url">Link here</a>
  </body>
</html>
```

Figure 4.3:
The root node in the HTML is <html>, since all other nodes are located inside <html>.
The <html> node is the parent node of its two child nodes <head> and <body>.
The <head> node holds a <title> node.
The <body> node holds a <p> and an <a> node.
The <title>, <p>, and <a> elements all hold text nodes.
The <a> element holds an attribute node.

An example of how a node within the DOM can be accessed with JavaScript is shown in code Listing 4.13.

```
1 <html>
2 <body>
3   <p id="magicmarker">sunny weather</p>
4   <script type="text/javascript">
5     txt=document.getElementById("magicmarker").innerHTML;
6     document.write("<p>" + txt + "</p>");
7   </script>
8 </body>
9 </html>
```

Listing 4.13: *Line 3: Assign the <p> element with an id (a word of choice that is supposed to identify the element), in this case the id is "magicmarker".*
Line 5: Use the JavaScript method document.getElementByID(), which gets an element with a specific id. Tell the method to get the element with id "magicmarker" and then use the innerHTML method, which extracts the text from the retrieved element.
Line 6: Use the JavaScript command document.write to output the text on the browser.

4.2.11 Technical Limitations of Ajax

Traditional web pages link to each other through simple hypertext links and pages with different information are clearly divided from each other as they only contain passive information. Any updates on a specific page requires a full page reload, and thus can be seen as a new page. Each different page, with its unique content, has its own web address. Navigation between different pages on the web is similar to browse pages in a book, and the page numbers in the book are represented by the unique web address of each page on the web site. Going back from a page to the previously visited one is normally done by a click of the "back" button in the web browser [3]. When dynamic pages are introduced by using Ajax, the pages internal contents may change by an event, for example, the click of a button. With dynamic web pages, the metaphor of turning pages with different page numbers in a book does not apply anymore, as the content of the pages is no longer passive. When the contents change internally on a web page, the address of the page might not define what exact information is shown to the user. As a consequence of having pages with dynamic information, web browser functions that are based on the web address as main reference of navigation, such as the "back" button, or making bookmarks in the browser, might not necessarily refer to the information that the user was expecting.

Ajax is dependent on the availability of JavaScript to work, and with JavaScript turned off in the web browser, Ajax will not work. A problem with JavaScript is that it has been implemented differently in different browsers and some older browsers might not fully support modern Ajax features [18]. As neither HTML or JavaScript are designed to be used for audio or video streaming, Ajax cannot be used for such functionality.

Other problems that have brought some critique to Ajax is that printing of Ajax web pages do not always work well and that some search engines have trouble when searching Ajax pages since they do not know what page states to include when indexing the web page [16].

4.3 Other RIA Techniques

There are several techniques that compete on the market of RIA development and this section describes some popular techniques for plug-in based RIAs. One advantage for plug-in based techniques is the simplicity of development. They are called "Sandbox" techniques, as they provide the developer with stable and well formed development environments that will behave the same way across different platforms and browsers with their plug-in installed [4].

4.3.1 Adobe Flash/Flex

Flash was introduced in the market in 1996 by Macromedia, currently developed and distributed by Adobe, and has become a very popular approach to adding animated graphics, video, and interactive features to web pages [26]. The technique is often used to create animations, commercials, and RIA associated functionality. In recent versions of Flash, XML capabilities have been added to increase the functionality, which enables new possibilities to render RIA contents in the browser. This technology, which uses XMLHttpRequest, is known as Flex (Asynchronous Flash and XML) .

The programming language for Flash applications is called ActionScript. In order to run Flash applications on web pages, a Flash plug-in needs to be installed in the web browser. According to market research from Adobe Systems, an approximate of 95% of Internet-enabled personal computers had the Flash plug-in installed as of September 2008 [1].

The Flash technology has been criticized by Jakob Nielsen for being the main reason that many web developers break usability principles [14]. He stated that the use of Flash often results in the breaking of conventions associated with traditional HTML pages. Basic HTML related functions such as selecting text, form control, and right clicking contents on a Flash application act in ways different from that in typical HTML pages. He also mentions that many of these problems are possible to fix by the developers, but are often ignored as it demands more work for the developer.

4.3.2 Microsoft Silverlight

Microsoft Silverlight was first released on the market in 2007 as a web application framework for RIA applications with contents such as integrated multimedia, graphics, and interactivity [24]. The programming environment for Silverlight include the languages XAML, C#, JavaScript, and Ruby. The Silverlight plug-in must be installed in the web browser in order to run the applications. An advantage with Silverlight is that its language for describing user interfaces, XAML, is an interpreted language (which does not need compiling), which enables the contents to be searched by search engines [28].

4.3.3 JavaFX

JavaFX was introduced in the market by Sun Microsystems in 2007. It is a software platform for RIAs for many different types of devices [27]. The programming language for JavaFX applications is JavaFX Script, and the Java Runtime Environment is required to run the applications. One of the advantages with JavaFX is a feature called "Drag-to-Install" which means that a user can drag a JavaFX widget or application from the web browser window out to their desktop. The unique property of it is that the application will remember its state or context after the user has closed the web browser.

A current problem with JavaFX applications is that the Java Runtime Environment is a rather big plug-in to download, and that the performance is slower than both Flash and Silverlight [5].

4.4 Usability and Ajax

A great portion of the new web sites that appear on the web are designed with dynamic RIA functions based on techniques such as Ajax. The users are now actively interacting with the content and often even produce much of the materials themselves. To improve and develop the RIA concept, new interfaces and functions for navigating and editing materials on the web are constantly invented. There are not always standards or clear guidelines to follow when designing these new functions, as they do not have any clear predecessors. The web developers that are responsible for ensuring the usability of the functions have a task which is not always easy to master [17]. In an attempt to create a set of guidelines to help developers of user interfaces, and to have in mind when implementing new functions, Jakob Nielsen has created a set of heuristics [15]. Bruce Tognazzini has, with the same intent, created a list of principles for interaction design [19]. As new Ajax based functions are invented and developed for the web, it is desirable that the developer ensures that usability is maintained for the proposed users of the functions. Having Nielsen's heuristics and Tognazzini's principles as a reference when designing is one way for the developer to facilitate the focus on usability during the design process.

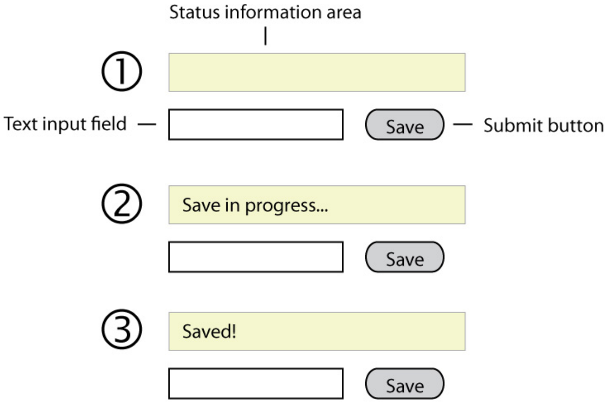
The following sections discusses a selection of heuristics and principles for interaction design from the perspective of Ajax related functions.

4.4.1 Nielsen's Heuristics

This section presents a selection of Jakob Nielsen's heuristics for User Interface Design and how they can be they can be used to improve Ajax enhanced functions.

H1 - Visibility of system status - The interface should keep the users informed of the system status all the time if possible. This could be done with feedback that lets the user know what is happening.

Figure 4.4 [12] shows an example of how a simple message box (status information area) can be used to indicate the status of a generic Ajax request being processed.

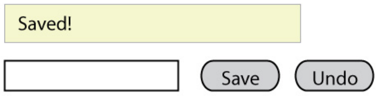


Save in progress...

Saved!

Figure 4.4:
1) The user types text in the "text input field" and presses the submit button to post the text.
2) The message "Save in progress..." is displayed while the request for posting the text is being processed by the server.
3) When the request is completed by the server, the message "Saved!" is displayed in the status information area.

H3 - User control and freedom - Web site visitors occasionally get lost in navigation and choose system functions by mistake and will need a convenient "emergency exit" to get out of unwanted state without a complicated exit procedure. Support for undo and redo also provides a greater degree of user control. Figure 4.5 shows an example of an undo button for a form.

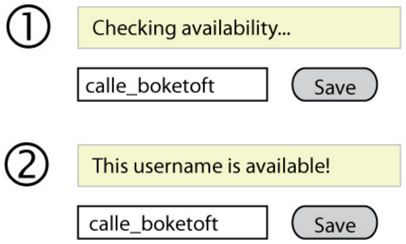


Saved!

Figure 4.5: *After the third step of the form submit process in Figure 4.4, an undo button appear next to the save button. If the user presses the undo button, the posting of the text will be undone.*

H5 - Error prevention - Helpful error messages using natural language are good in a design but even better is to create a design that prevents the problem from happening in the first place. A way to do this is by presenting a confirmation option to the user before committing an action that might lead to a problem or a conflict.

Figure 4.6 [9] shows an example of how a confirmation option is shown to the user before committing an action.



Checking availability...

calle_boketoft

This username is available!

calle_boketoft

Figure 4.6:
1) A user types a desired username in the text input area and waits for the server to check if the username is available or already taken by someone else. The message "Checking availability" is displayed in the status information area.
2) The server has answered that the username is available and the message "This username is available!" is displayed in the status information area.

H6 - Recognition rather than recall - Make a design where objects, actions and options are visible in order to minimize the user's memory load. The user should not have to remember information between different states of the site to be able to proceed within the system. Integral instructions and information for use of the system should be visible or easily accessible when they are needed by the user.

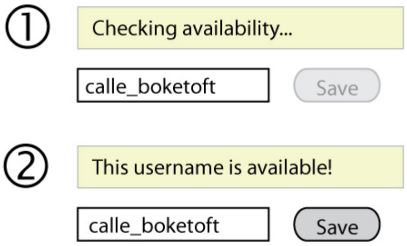
When filling out forms with multiple fields, designed in the traditional client-server model, faulty entered information is shown to the user first after the form has been submitted to the server and the web page has been reloaded with the answer to the client. The appearance of the reloaded page could mark out for the user the fields with incorrectly entered information so that the information could be corrected. Sometimes, the information could even disappear from all the fields after a page reload, even if only one field was filled out incorrectly, forcing the user redo everything. By using Ajax functions for form validation, as described in Figure 4.6, users can get continuous status information about the process and get the opportunity to correct mistakes while filling out the individual fields.

4.4.2 Tognazzini's First Principles on Interaction Design

This section lists a selection of Bruce Tognazzini's First Principles on Interaction Design and how they can be used to improve Ajax enhanced functions.

T10 Latency Reduction - Reduce latency by pushing computations and transmissions into the background. The user's perception of latency can be reduced by using visual feedback of the computation, such as a progress bar or an hourglass. Avoid possibility for multiple clicks of the same button or object, when it is not intended.

The example in Figure 4.7 (variation of Figure 4.6) shows how to avoid the possibility for multiple clicks of a button before it is confirmed that the action is possible.



Checking availability...

calle_boketoft

This username is available!

calle_boketoft

Figure 4.7:
1) Before the server has confirmed that the desired username is available, the submit button is faded and do not respond to a click by the user, thus avoiding the risk of sending information until the server has responded.
2) If the server responds that the username is available, the submit button changes to clear and it is possible to click to submit the request.

A way to reduce latency for users is to have the browser anticipate likely user actions by requesting preparatory information by the server [2]. An example of predictive fetch is implemented in Google Maps [8], where the map is fetching map graphics outside of the viewable area so that the user doesn't have to see and wait for actual loading of the map while slowly moving around the map in any direction.

Text box auto-complete is a popular way to reduce latency by examine the first few characters typed by a user, and then suggest words from a predefined list [29]. An example of this function is shown in Figure 4.8.

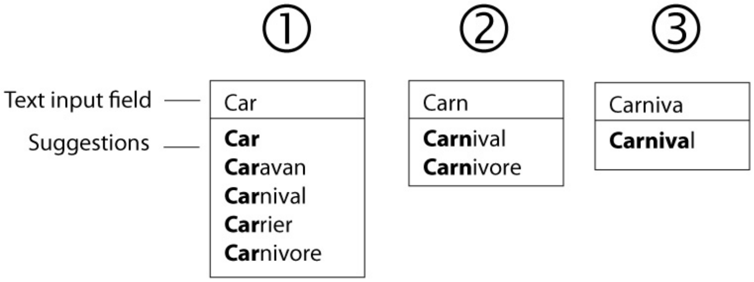


Figure 4.8:
1) A user has typed the text "Car" into the text input field, and Ajax retrieves suggestions from a list of predefined words.
2) and 3) As the user keeps typing letters into the text box, the suggestions update continuously.

T13 Protect the User's Work - Secure the work of the users by making sure that entered data does not get lost as a result of a user induced error. Any unwanted reasons for losing data are unacceptable except the unavoidable such as a power out. Continuous-save in the program architecture is a preferable solution to this hazard.

Continuous auto-save of a document without complete page reloading exists in GMail [7]. The auto save function is activated by time intervals, so that the document is continuously saved while the user is writing the mail.

Chapter 5

Accomplishment

This chapter describes the development process of the project. The time plan for the project, containing all important deadlines and major events, is described in Section 5.1. A selection of ideas and concept models for the design of the web site that was formed during the development process are presented and described in Section 5.2.1. Section 5.3 presents some of the ideas and solutions for different interesting programming related problems that arose during the implementation of the web site.

5.1 Time Plan

A time plan was set up at the beginning of the project, containing dates for the deadlines for all important steps during the development and documentation process. The full time plan can be seen in Appendix B. The different deadlines were decided from estimates about how long time and how much effort the different parts of the process would need to get finished. Some important steps were finished before the time plan was set up, such as the initiation of the project and the elicitation of the first requirements for the web site. The first four weeks of the planned time was spent with the theoretical study in Chapter 4. When the theoretical study was completed, the elicitation of requirements continued. During the elicitation, the requirements were also refined, written down, and finally checked so that they correspond to the scope and nature of the project.

When finishing the first version of the requirements document the work with the implementation of the web site begun. The programming was finished after its dedicated six weeks and a first version of the foundation for the web site was ready and presented to the restaurant owner. Only a week after the meeting with the owner, the further development of the product was delayed for about a month due to an unplanned relocation of my working place from South Korea back to Sweden. When the situation had become stable in Sweden, about one additional month was spent refining the product and writing the project report. In the end, all the deadlines for the development of the product were kept, but the deadline for the finishing of the report had to be postponed twice due to the time management complications that arose during the relocation from South Korea to Sweden.

5.2 Developing the design

Before the work with the development of the design of the web site had begun, a few factors which would come to play a big role in the design process had not yet been decided. There were no requirements saying exactly how the aesthetic design of the product should be, so there was almost total artistic freedom for me to come up with a design that I thought would work. The owner of the restaurants had not yet decided if the web site would be only for Ye-Ga, the most popular of the restaurants, or for all three restaurants.

5.2.1 Concept models

Since the factor of usability of the functions and features on the web site was very important to the project, the design of the product was a big part of the development process. The requirements in Appendix A were used as the base for choosing which functions to include in the interface and how to distribute the contents over the site. A number of concept models were drawn in the design process and were evaluated to see which one would be most suitable to continue working on as a sort of template for the actual web site. All the early models were made to be used only for Ye-Ga since it had not yet been decided if the project would include web pages for all three restaurants or just one restaurant. In order to clarify and document the thought out functionality of the different concept models, they were all provided with textual descriptions of specific functions.

One of the fist conceptual models that were candidates for the design of the site is presented in Figure 5.1. The main framework of the web site would consist of two sections with a navigation frame on top with links to the different pages of the web site. The navigation frame was made to look something like a business card with the restaurant name, address and telephone number. The bottom frame would be the main content frame for all the contents of the different pages of the web site, such as Welcome page, About page, and so on. An early idea was to have four pictures to the left in the content frame, for aesthetic purposes.

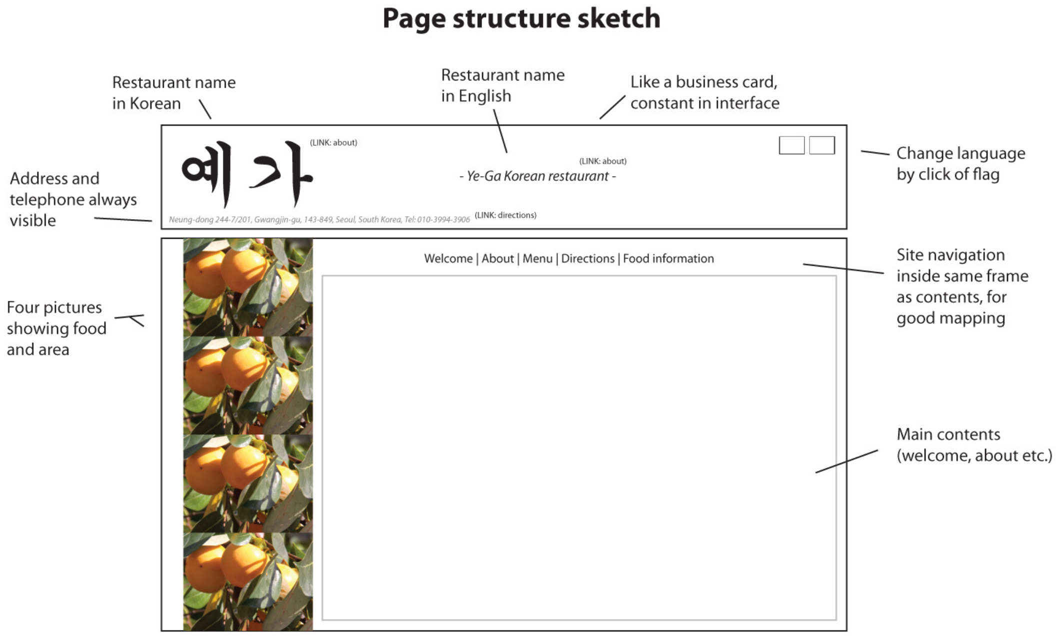
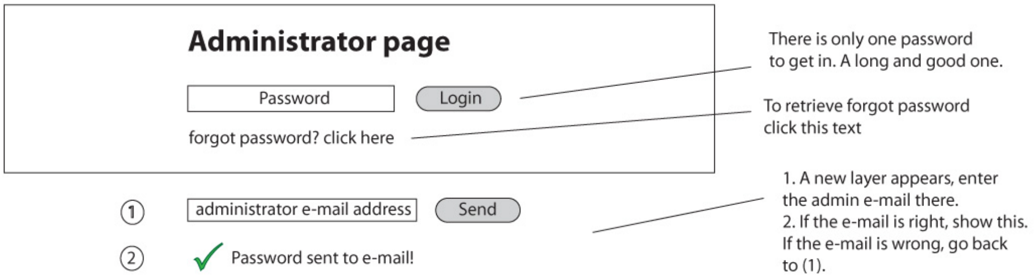

Page structure sketch
Restaurant name in Korean
Restaurant name in English
Like a business card, constant in interface
Address and telephone always visible
Change language by click of flag
예가
(LINK: about)
(LINK: about)
- Ye-Ga Korean restaurant -
Neung-dong 244-7/201, Gwangjin-gu, 143-849, Seoul, South Korea, Tel: 010-3994-3906
(LINK: directions)
Welcome | About | Menu | Directions | Food information
Site navigation inside same frame as contents, for good mapping
Four pictures showing food and area
Main contents (welcome, about etc.)

Figure 5.1: *An early conceptual model of the structure of the web site. There was a presentation frame on top, and a content frame below it with links to all the different pages of the site and where the page contents would appear.*

After deciding the main framework of the design of the web site, with a navigation frame on top and a content frame below it, the different pages and their functions were quickly drawn up to a certain level of detail. The first page to be refined and developed in more detail, including full functionality descriptions, was the administrator login page shown in Figure 5.2. Many of the functions and behaviors of the different events on the admin pages were developed in direct relation to the results of the research in the theoretical study in 4 regarding usability enhanced by Ajax techniques. For example when you click the “send” button in the example it disappears right after it has been clicked, just like described in “T10 Latency Reduction” in section 4.4.2. The login page for the administrator settings was supposed to be a very simple page. What you would see when you first go in to the page was what is shown inside the top frame in the figure. The only functionality, except actually logging in, was for retrieving the password if you lost it.


Administrator page
Password
Login
forgot password? click here
1
administrator e-mail address
Send
2
✓ Password sent to e-mail!

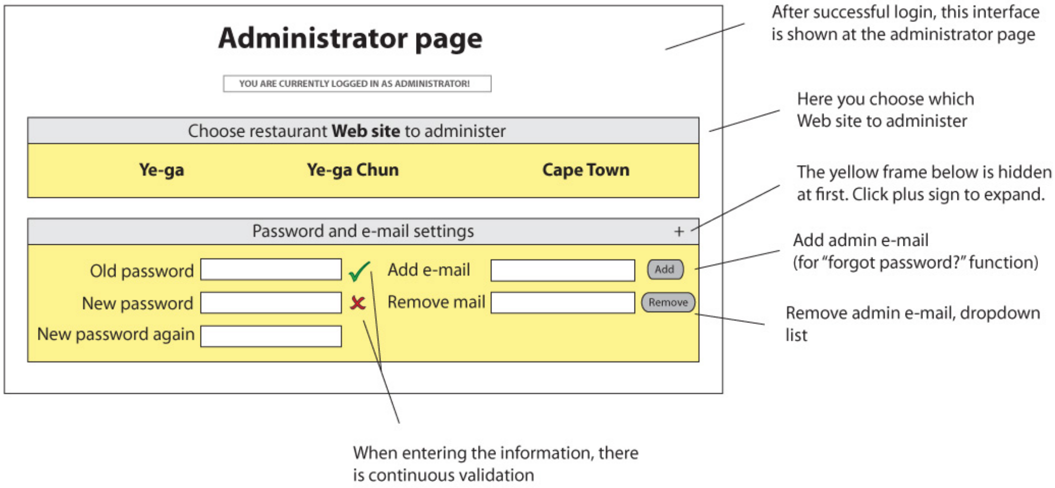
There is only one password to get in. A long and good one.

To retrieve forgot password click this text

1. A new layer appears, enter the admin e-mail there.
2. If the e-mail is right, show this. If the e-mail is wrong, go back to (1).

Figure 5.2: *The login page for the administrator settings.*

After successfully logging in to the administrator pages of the web site, the user would get to the administrator settings page. By the time the model for the administrator settings page was created, it had been agreed that a web site with pages for all three restaurants would be created. The interface for the administrator settings page is shown in 5.3. The password change form, to the left in the bottom frame in the figure, used the kind of instantaneous form verification that was explored in “H5 - Error prevention” in section 4.4.1. When a user was filling out the form for changing the password, the entered information was simultaneously validated. In the administrator settings page you could either choose to go to one of the three restaurants’ web pages to administer, or to edit the password and e-mail settings for the administrators.



After successful login, this interface is shown at the administrator page

Here you choose which Web site to administer

The yellow frame below is hidden at first. Click plus sign to expand.

Add admin e-mail (for “forgot password?” function)

Remove admin e-mail, dropdown list

When entering the information, there is continuous validation

Figure 5.3: *The interface with administrator settings for the web site.*

Apart from the administrator pages, all other pages of the web site was viewable from either the guest perspective, only displaying the contents of the pages, or the CMS perspective, containing all editing functionality for the administrator of the web site. In order to see the web site from the administrator perspective, it was necessary to first log in via the administrator login page. The design of the framework of the web site from the guest perspective is shown in Figure 5.4. The navigation frame was very simplistic without any distracting elements or functions. It contained the essential contact information for the restaurant and the site navigation menu for the different pages of the restaurants. The content frame was only an empty placeholder for the main contents of the page that you could navigate to through the site navigation menu. The design was quite similar to the design in Figure 5.1, except that the site navigation menu has been moved up into the top frame.

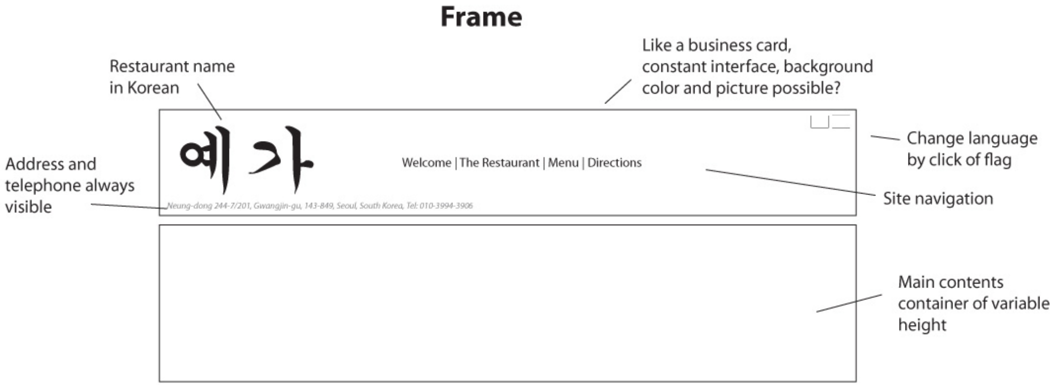

Frame
Restaurant name in Korean
Like a business card, constant interface, background color and picture possible?
예가
Welcome | The Restaurant | Menu | Directions
Change language by click of flag
Address and telephone always visible
Neung-dong 244-7/201, Gwangjin-gu, 143-849, Seoul, South Korea, Tel: 010-3994-3906
Site navigation
Main contents container of variable height

Figure 5.4: *A slightly refined design of the framework for the web site.*

When logged in as administrator, you would see the web site from the CMS perspective. Even if the navigation frame of the web site contained very little elements, the restaurant address and restaurant logo were possible to update via the CMS. The model in 5.5 shows what the navigation frame would look like from the CMS view. With this model, many of the key functions of the CMS view were introduced. Right below the navigation frame was an administrator specific frame for image updating tools. In there you could upload new background images and there was an iframe dedicated to status messages related to image uploading. The reason for including an iframe is explained in Section 5.3.6.

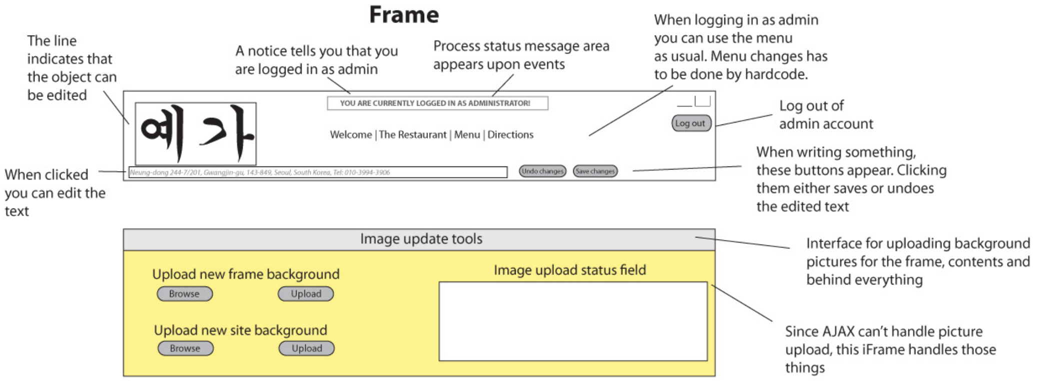

Frame
The line indicates that the object can be edited
A notice tells you that you are logged in as admin
Process status message area appears upon events
When logging in as admin you can use the menu as usual. Menu changes has to be done by hardcode.
YOU ARE CURRENTLY LOGGED IN AS ADMINISTRATOR!
예가
Welcome | The Restaurant | Menu | Directions
Log out
Log out of admin account
When clicked you can edit the text
Neung-dong 244-7/201, Gwangjin-gu, 143-849, Seoul, South Korea, Tel: 010-3994-3906
Undo changes
Save changes
When writing something, these buttons appear. Clicking them either saves or undoes the edited text
Image update tools
Interface for uploading background pictures for the frame, contents and behind everything
Upload new frame background
Browse
Upload
Image upload status field
Upload new site background
Browse
Upload
Since AJAX can't handle picture upload, this iFrame handles those things

Figure 5.5: *The navigation frame from the CMS perspective.*

Status Messages - In different locations in the CMS view there were status fields that indicated what events were being activated in the interface. In the top of the navigation frame in Figure 5.6 there was such a status field that said “YOU ARE CURRENTLY LOGGED IN AS ADMINISTRATOR”, which also worked as an event status field, where status messages were shown when different events in the top frame were activated.

Image uploading - All images on the web site were dynamic and could be uploaded via the CMS. In the upper left section of Figure 5.6 there is an image of two Korean signs that make up the logo of the restaurant. Since you were supposed to be able to upload a new image in its place, it needed some functionality for image uploading. Around the image was a line that simply indicated that it was a picture. Although it does not show in the figure, when you clicked the image a pair of buttons for uploading a new image would appear. One of the buttons said “upload” and the other button said “cancel”. If you clicked the upload button, you would get to choose a new image from your computer to upload in the place of the existing image on the web site. If you did not intend to upload a new picture or simply change your mind about uploading a new picture, you could click the cancel button and the pair of buttons would disappear again. Upon the event of uploading a new image, the status of the image upload would show in the iframe located by the background image updating tools in Figure 5.5.

Background image uploading - Not only the pictures on the web pages were possible to replace, also the backgrounds of all the different frames were possible to change. To the left in the lower of the two frames in Figure 5.6 were the functions for background image uploading. The uploading functionality was principally the same as for uploading the restaurant logo image. The only difference was that instead of clicking an image to replace to get the upload buttons, you would click the text “Upload new frame background” to make the pair of upload buttons appear.

Text editing - Nearly all the texts on the web pages, except for example the navigation menu choices and admin texts, were editable. To the left in the navigation frame in Figure 5.6 was a text field for the address and telephone number of the restaurant. Around the text field was a frame that indicated that the text was editable. If you started editing or writing text inside the field, a pair of buttons would appear. One of the buttons was a save button that saved the changes that had been made to the text, and the other button was a cancel button that reverted any current changes that had been made to the text. When one of the buttons had been clicked, they both would disappear again.

Some of the described events in Figure 5.6 were only done as a concept, and not chosen to be implemented in the final web site. An idea that emerged from “H1 - Visibility of system status” in Section 4.4.1 was to display current save status when pressing a save button, so that the user could follow the different stages of the saving process. Status updates inside the save button, as seen to the right in the figure, were not considered necessary since the saving process was quicker than was useful to be shown in the interface. The “mouse over” effect that is described to the left in the figure was implemented, but only for images, and a bit different than what is shown here.

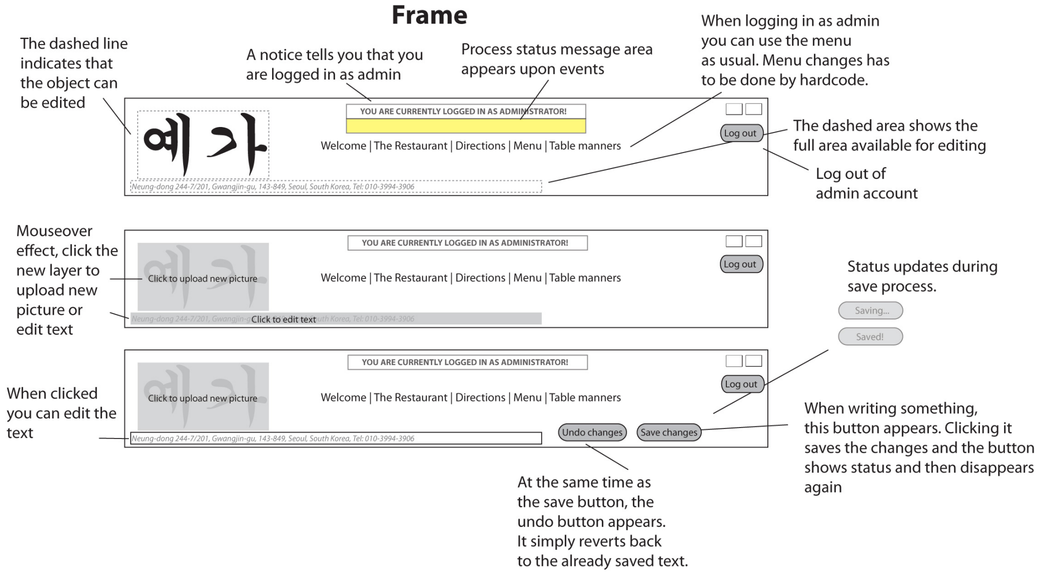

Frame
The dashed line indicates that the object can be edited
A notice tells you that you are logged in as admin
Process status message area appears upon events
When logging in as admin you can use the menu as usual. Menu changes has to be done by hardcode.
YOU ARE CURRENTLY LOGGED IN AS ADMINISTRATOR!
예가
Welcome | The Restaurant | Directions | Menu | Table manners
Log out
The dashed area shows the full area available for editing
Log out of admin account
Neung-dong 244-7/201, Gwangjin-gu, 143-849, Seoul, South Korea, Tel: 010-3994-3906
Mouseover effect, click the new layer to upload new picture or edit text
Click to upload new picture
Click to edit text
Status updates during save process.
Saving...
Saved!
When clicked you can edit the text
Undo changes
Save changes
When writing something, this button appears. Clicking it saves the changes and the button shows status and then disappears again
At the same time as the save button, the undo button appears. It simply reverts back to the already saved text.

Figure 5.6: *The top frame from the CMS perspective. Many of the usability and Ajax related functions that were later used in several different locations of the web site emerged from this conceptual model.*

When the design of the navigation frame, with site links and basic information was finished, it was time to draw up the different content pages. The set of pages that was worked on as concept models were:

Welcome - The splash screen (page that is first shown when you enter the web site) with a welcome message by the owner and a picture of the staff or the owner of the restaurant.

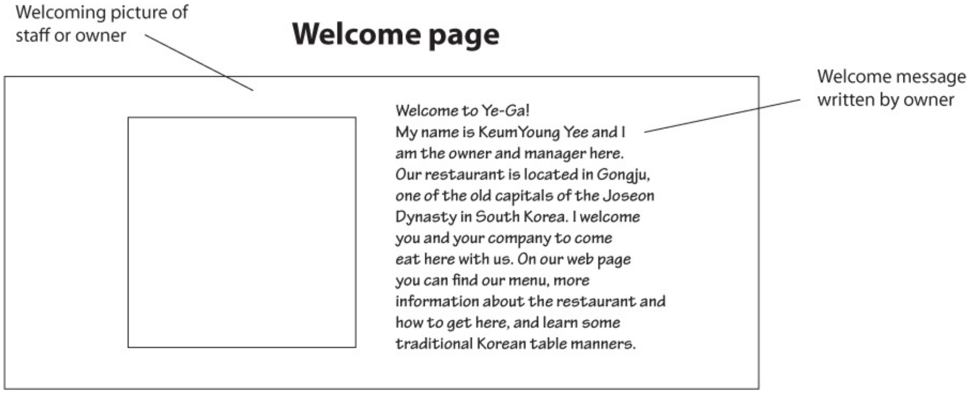
The Restaurant - The history of the restaurant and specific information about the different foods.

Directions - A page with a map and directions to the restaurant. Possibly pictures and text about the nearby area and tourist attractions.

Menu - The food menu for the restaurant, containing Korean name of the dish, Korean explanation of the dish, English explanation of the dish, price, and possibly an image of the dish.

Table Manners - A page for foreign tourists who wanted to learn about how to eat the different dishes in the restaurant.

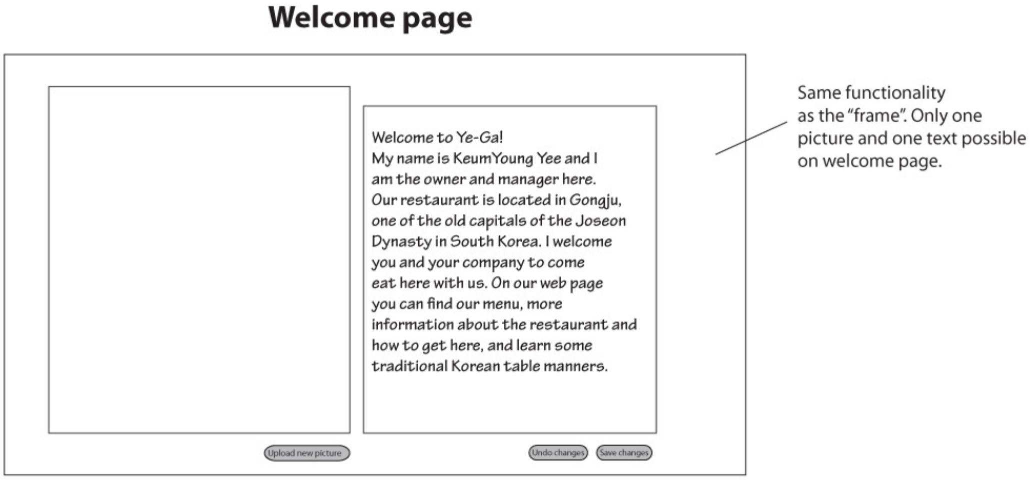
The first page that was designed was the guest view of the Welcome page, as seen in Figure 5.7. The page was only supposed to contain one picture and one text. The reason for the simple setup was to start up the designing process slowly, with an easy layout.


Welcoming picture of staff or owner
Welcome page
Welcome message written by owner

Welcome to Ye-Ga!
My name is KeumYoung Yee and I am the owner and manager here. Our restaurant is located in Gongju, one of the old capitals of the Joseon Dynasty in South Korea. I welcome you and your company to come eat here with us. On our web page you can find our menu, more information about the restaurant and how to get here, and learn some traditional Korean table manners.

Figure 5.7: *The first conceptual model of the Welcome page from the guest view. It was designed to be of extreme simplicity, to get an open starting point for the coming designs.*

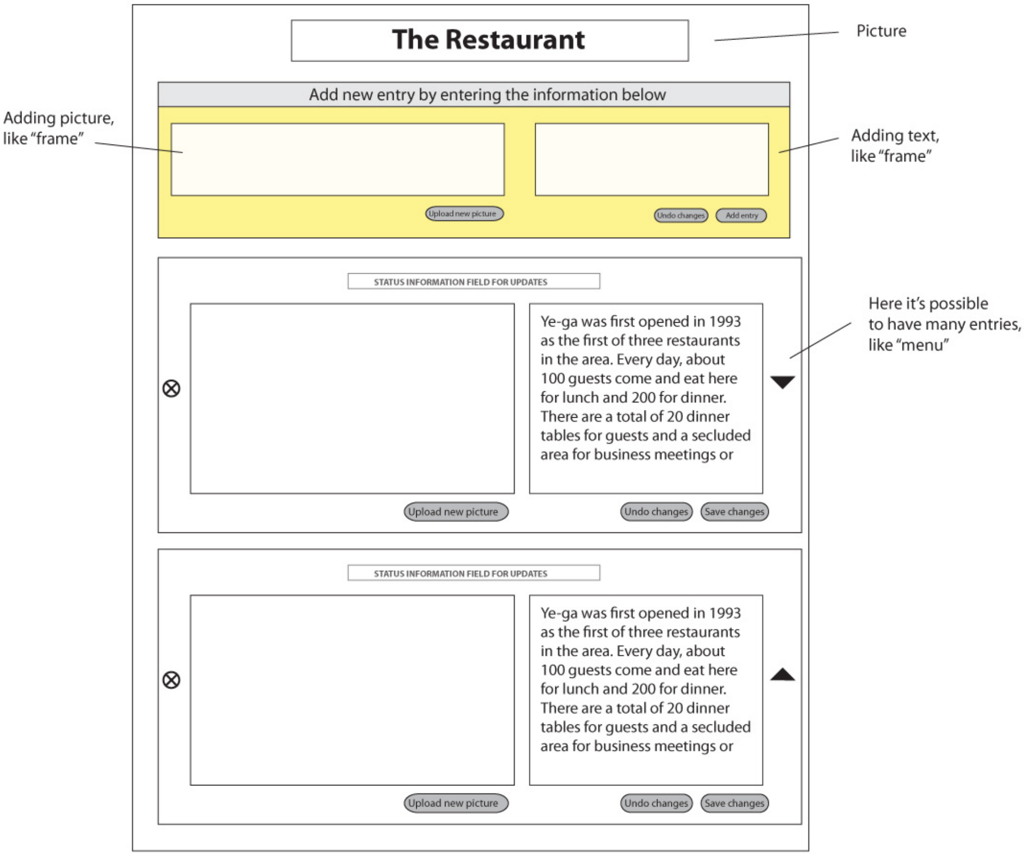
Then a concept model of the CMS view of the Welcome page was created, as shown in Figure 5.8. It maintained the simplicity of the page from the guest view, but with added functionality for updating the picture and text. The updating functions were going to work exactly like described in Figure 5.6. The frame for the picture and the text were going to be static in their positions and with fixed sizes.


Welcome page
Same functionality as the "frame". Only one picture and one text possible on welcome page.
Upload new picture
Undo changes
Save changes

Welcome to Ye-Ga!
My name is KeumYoung Yee and I am the owner and manager here. Our restaurant is located in Gongju, one of the old capitals of the Joseon Dynasty in South Korea. I welcome you and your company to come eat here with us. On our web page you can find our menu, more information about the restaurant and how to get here, and learn some traditional Korean table manners.

Figure 5.8: *The design of the concept model of the Welcome page from the CMS view was equally static and simple as the page from the guest view.*

After finishing the simple models of the Welcome page, a more dynamic model of The Restaurant page was formed, as seen in Figure 5.9. The model's main new feature was that instead of having only one pair of a picture and a text, you could now add any desired amount of pairs of pictures and texts, called entries. The adding of new entries was going to be done by uploading a new picture and writing a new text inside a dedicated, yellow, frame and then pressing the button for adding the new entry. Each new added entry was added to the page and had full possibilities for editing the text and uploading a new image. The entries were going to be added on top of each other and with possibilities to move a entry up or down in the hierarchy of entries by pushing arrow buttons.


The Restaurant
Picture
Add new entry by entering the information below
Adding picture, like "frame"
Adding text, like "frame"
Upload new picture
Undo changes
Add entry
STATUS INFORMATION FIELD FOR UPDATES
Here it's possible to have many entries, like "menu"
⊗
▼
Upload new picture
Undo changes
Save changes
STATUS INFORMATION FIELD FOR UPDATES
⊗
▲
Upload new picture
Undo changes
Save changes

Ye-ga was first opened in 1993 as the first of three restaurants in the area. Every day, about 100 guests come and eat here for lunch and 200 for dinner. There are a total of 20 dinner tables for guests and a secluded area for business meetings or

Ye-ga was first opened in 1993 as the first of three restaurants in the area. Every day, about 100 guests come and eat here for lunch and 200 for dinner. There are a total of 20 dinner tables for guests and a secluded area for business meetings or

Figure 5.9: *The concept model of The Restaurant page had the new possibility of adding pairs of pictures and texts. Any desired amount of pairs would now be possible to have in the page. Functionality for editing all texts, uploading new pictures, and moving pairs up and down in relation to each other was designed.*

To me, the design of The Restaurant page felt useful and easy to use, so it was decided that other pages were also going to be designed in a similar fashion, or even in the exact same way. The pages with the same proposed design were going to be Welcome, The Restaurant, Directions, and Table Manners. The only page with different design and functionality was the Menu page.

The Menu page from the guest view can be seen in Figure 5.10. One of the two entry types of the Menu page was the menu headline entry for the different types of meals such as "Chicken", "Beef", and "Pork". The headline would consist of a black wide banner with white text on top of it. The other constituent was the dish entry which would include all the essential information about a dish and a corresponding photo of the food. There were thoughts about having spice grading of each dish, but this feature was never implemented.

Another idea that was not implemented was to have automatic PDF document generation of the menu, for printing purposes.

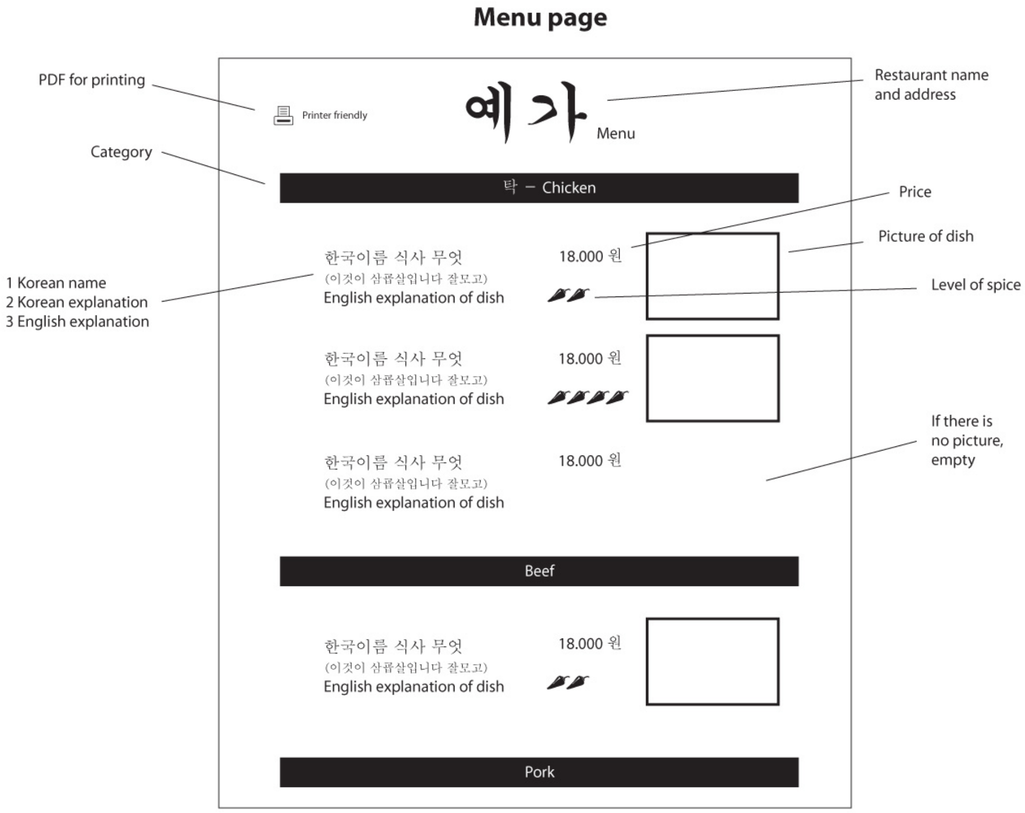


Figure 5.10: *The menu from the guest view. Two entry types made up the Menu page; the entries with the headline of the dish categories and the dish entries.*

The design of the CMS view of the Menu page, as seen in Figure 5.11, was more of a reworked and refined version of the conceptual model of The Restaurant page than a model with new ideas. Adding new dish entries and new headline entries were going to be done in a very similar fashion to that of adding new pair entries on The Restaurant page. All the different entries would also be fully editable, both text and image, and movable up and down in their hierarchy on the page. One feature that was added here and was implemented was that you could choose where in the hierarchy a new entry would appear after being added.

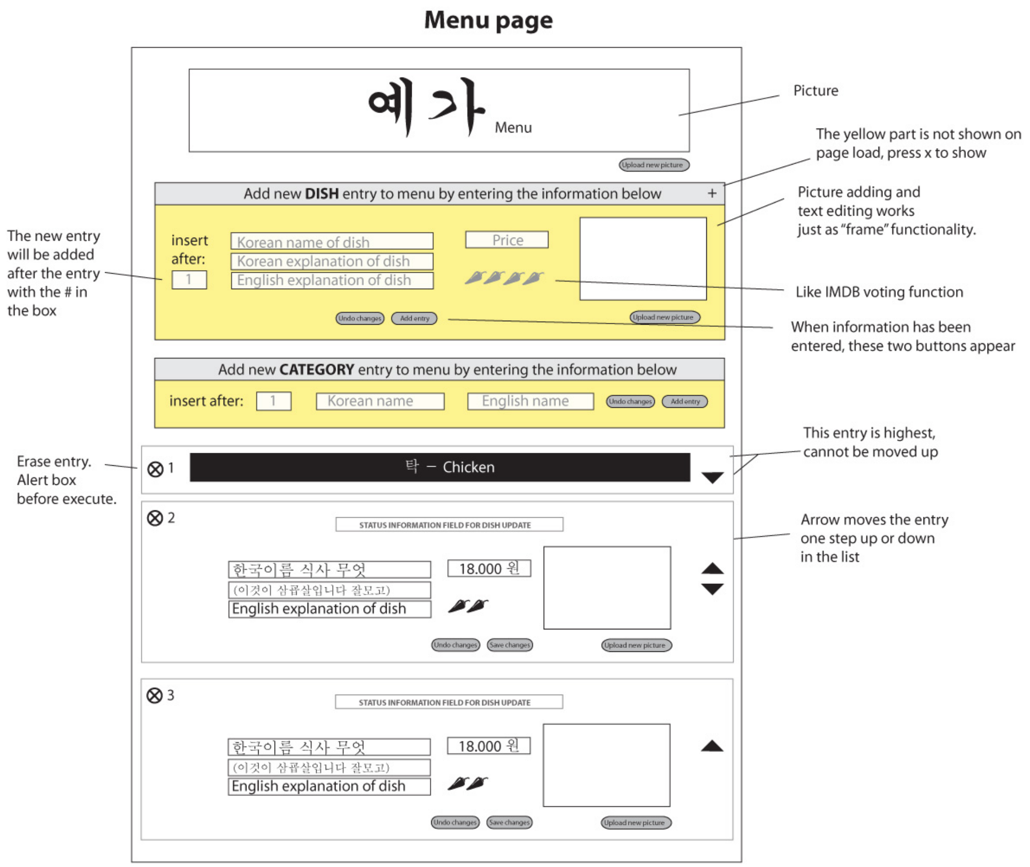

Menu page
예가
Menu
Upload new picture
Picture
The yellow part is not shown on page load, press x to show
Add new DISH entry to menu by entering the information below
+
Picture adding and text editing works just as "frame" functionality.
The new entry will be added after the entry with the # in the box
insert after:
1
Korean name of dish
Korean explanation of dish
English explanation of dish
Price
Like IMDB voting function
Undo changes
Add entry
Upload new picture
When information has been entered, these two buttons appear
Add new CATEGORY entry to menu by entering the information below
insert after:
1
Korean name
English name
Undo changes
Add entry
This entry is highest, cannot be moved up
Erase entry. Alert box before execute.
1
탁 – Chicken
2
STATUS INFORMATION FIELD FOR DISH UPDATE
Arrow moves the entry one step up or down in the list
한국이름 식사 무엇
(이것이 삼푭살입니다 잘모고)
English explanation of dish
18.000 원
Undo changes
Save changes
Upload new picture
3
STATUS INFORMATION FIELD FOR DISH UPDATE
한국이름 식사 무엇
(이것이 삼푭살입니다 잘모고)
English explanation of dish
18.000 원
Undo changes
Save changes
Upload new picture

Figure 5.11: *The Menu page from the CMS perspective. The functions were designed very similarly to the functions of the CMS version of The Restaurant page.*

The conceptual models presented in this section served as main templates for the implementation of the different pages of the web site.

5.3 Implementation

When the conceptual models of the web site had been finished to a certain level of detail, the implementation of the code for the pages began. The plan was to implement the code to be as similar to the conceptual models as possible. The first task was to set up a working web server on a local computer to be able to easily test and debug the code that was about to be implemented. A package called WAMP (Windows, Apache, MySQL, PHP) was installed on a local computer, since it contained software for running a web server with all necessary components for running the code of the web site. The choice to use PHP for server side functionality was made since I had experience in the language from a previous project.

5.3.1 Administrator Pages

The first task of the implementation was to construct the administrator login page, so that it would be possible to view the different pages of the web site from either the guest view or the CMS view. The administrator login page was implemented with the design that was proposed in the conceptual model in Figure 5.2. The password for the administrator login was stored in a separate PHP document, to keep it separated from the other code. A successful login with the correct password was only stored in the current session, there was no way of saving the login in a cookie. This was chosen since the administrator pages should be able to use from virtually any computer anywhere, and if an administrator would sit at a public computer and forget to log out after changing something, nobody else should be able to stumble upon the administrator pages due to a forgotten cookie.

When the administrator login page had been finished, it was possible to reach the administrator settings page as a logged in administrator. Since the administrator settings page was the same for all restaurants, only having functionality for changing the password and adding administrator e-mails, it was not yet necessary to divide contents based on restaurant, only based on language. The password changing function and e-mail management were implemented just as shown in the conceptual model in Figure 5.3, as a fully functional Ajax enhanced interface. In order to change the current password, it was necessary to write the old password in a field, to make sure it was actually an administrator changing the password. Then the new password had to be written in two fields, in order to ensure that the thought of new password was not misspelled. When the three fields had been successfully filled out, it was possible to click a save button that saved the new password. The function to add administrator e-mails was used for the function on the administrator login page to retrieve a lost password. If an administrator's e-mail address was part of the administrator e-mail addresses, he could write that same e-mail address in the dedicated field on the administrator login page to get the administrator password sent to his e-mail. The field for adding an administrator e-mail was programmed with a set of regular expression rules that would ensure that a well formed e-mail address was written in the field before it was possible to add it to the list of e-mails. The last function of the administrator settings page was to remove currently added administrator e-mails from a drop down list.

5.3.2 Storing the Data

The web site was going to be available in several languages and for several restaurants, and therefore the textual contents would have to be split up to be fetched depending on what language and restaurant was chosen. Very early in the process it was chosen that all textual language specific data would be saved in XML documents. There were several reasons for the choice to use XML stored data in stead of saving data in a database such as MySQL. The main reason was that data stored in XML documents would be more portable, since all data is stored in local text files, it would be very simple to move the whole web site with all contents included on, for example, a USB memory in a compressed ZIP file. It is possible to achieve the same portability of the type of data that is stored in a MySQL database, but it would require algorithms or third party software for downloading the database into local data files that would then be possible to reinsert into the database software. Another reason for choosing XML based data was that XML seemed well fit to use together with the other Ajax related techniques. For me, the choice of using and learning about XML was interesting since I had already worked with MySQL databases for earlier projects.

Most of the code for XML was written in JavaScript and therefore most of the fetching and rendering of XML contents was taking place on a browser level. Code Listing 5.1 describes one kind of XML element tag structure that was used in the implementation and code Listing 5.2 shows the corresponding tag in the HTML DOM. In this case it is a <h1> headline in the HTML DOM structure that is called “Admin page” in English and “Admin-sida” in Swedish inside the text elements in the XML structure.

```
<contents>
    <site_base>
        <pagetitle language="eng">Admin page</pagetitle>
        <pagetitle language="swe">Admin-sida</pagetitle>
    </site_base>
</contents>
```

Listing 5.1: *The XML element tag structure.*

```
<h1 id="pagetitle"></h1>
```

Listing 5.2: *The HTML DOM identification tags.*

5.3.3 Rendering the Contents

The original HTML DOM structure and corresponding XML structure was only supposed to be used for pure textual contents on the web pages, such as paragraphs or headlines, not for text within buttons or place holders for images. The structure shown in code Listing 5.1 and code Listing 5.2 was used in for the original text rendering algorithm.

The algorithm for rendering text and image contents is described below:

1. Find out what language, restaurant, and specific page was chosen. This information was available in either cookies or the page address.
2. Fetch the corresponding XML file with content data for the current page.
3. Read the HTML DOM structure to find all elements with specified id tags.
4. Use the HTML tag to fetch a corresponding element tag, with the exact same name, within the XML document, and fetch the text content of the found XML element.
5. Place the text or specified image, from the XML element, inside the HTML tag and display it on the page in the browser.

As the structure of the pages got increasingly complicated, a compromise had to be made with the current XML tag format when it was going to be used for language specific images or texts inside of buttons. The XML structure that had been created was only supposed to be used for texts within text areas with, for example, <span> or <p> tags. The code with the algorithm for drawing out page specific contents was enhanced to be able to handle different types of elements of the HTML DOM, such as buttons and images. The code was never optimized to be used with a common XML format that would work for all different element types of the web site, instead the rendering algorithm was altered to be able to understand the different types of XML formats for texts, images, and texts inside of buttons.

One of the aspects that made image rendering a bit tricky was that some images were going to be the same for all languages while other images were going to be language specific. This was solved by keeping both the placeholder and image source specification for images that were the same for all languages within the hard code in the HTML documents, and not including them in the XML code.

5.3.4 Separating Contents for Different Restaurants

As the project would include web pages for three different restaurants, it was necessary to find a way to use the code in a way that all restaurants could use the same site base, only with restaurant specific content. All the code that was going to be identical for the restaurants, such as the admin login page, admin settings page, the framework for the whole restaurant web site, some of the site CSS design, and so on, was stored in the root folder of the web site. The restaurant specific contents such as page titles, textual and image contents of the different pages, were stored in a folder with the restaurant name. This way, all the contents that were shared between all restaurants was accessible from the web site root folder, and the restaurant specific contents were all stored together in one folder each.

5.3.5 Server and Browser Side Functions

The browser side functions for the web site were implemented in JavaScript and all the code was stored in one big function library file. The server side functions were implemented in PHP and all the code was stored in another function library file. It was chosen that all the PHP server side functionality was going to be written as parts of a switch statement, and not as different functions, which performed the different actions based on keywords that were sent in. Most of the page updating functions on the pages using the XMLHttpRequest object were called by the click of a button, or a keystroke, on the web site interface which in turn sent the keyword to the switch statement in the PHP document for performing the requested action.

When performing updating actions based on Ajax techniques, status messages were shown in chosen locations in the interface to indicate in which state the updating process was. Since the XMLHttpRequest object is written in JavaScript and with a dedicated variable to tell the current status of the request being performed, it was very easy to show in the browser the different stages in the updating process. Different color codes were used in combination with formulated text messages for different status notifications.

5.3.6 Image Uploading Issue

When submitting changes to texts in the web site interface, the collaboration with the XMLHttpRequest object was working seamlessly and as supposed since the object is specifically designed to handle text. A problem that occurred was that the XMLHttpRequest object only could handle text and not images at all. In order to upload an image it was necessary to find another way instead of using the the XMLHttpRequest object. There was no evident way, with the current page design, to solve the issue to upload a new image on the web site without reloading the current page entirely. Instead of accepting a full page reload when uploading an image, the problem was solved by creating a small iframe inside the CMS interface, that would handle the entire uploading process by loading the PHP functions for image uploading. The image uploading started working just fine, but now the XMLHttpRequest object was no longer involved, so it was not able to show the process of uploading the image, and status messages could not be shown in the same way for image uploading as for text editing.

A solution to the status message problem during image uploading was thought out by simply showing the image upload status inside the iframe that was created in the CMS interface. Another problem that had occurred was that the image that was uploaded, if it was supposed to replace an already existing image on the page, had to be shown in the browser after it had been uploaded. This problem was solved by setting a timer for reloading the relevant image after an interval of a couple of seconds after uploading the image inside the iframe. If the image was successfully uploaded within the time interval that was set for the reloading function, the image would reload itself to show the new image.

5.3.7 Web Site Page Types

After careful consideration it was decided that each restaurant would have four pages each, “Welcome”, “The Restaurant”, “Menu”, and “Location”, with the type of contents that were described in the concept models in section 5.2.1. According to the concept models, there were three different types of content pages for each restaurant. The first type of page was used for the Welcome and The Restaurant pages. That type of page only contained entries with pairs of a text and a picture, if no entries were added in the CMS, there were simply no contents on those pages, only the picture for the page title whose place holder was hardcoded into the HTML. Both pages were designed and worked in the exact same way. The second type of page was the Menu page which was working in a similar way to the first type of page, except that the entries did not only contain pairs of a text and an image, but dish entries with all the information for a dish. Another difference was that on the Menu page you were also able to add headline entries with, for example, categories for different dish types. The third type of page was the Location page, which was working much like the first type of page, except that there was a map hard coded in the HTML with the location of the restaurant, placed between the page title and the first entry.

5.3.8 Dynamic Page Contents

On the pages with dynamic contents, which were all four pages for the restaurants, much functionality had to be designed and implemented to make a useful CMS interface. All the page specific data for the pages was stored in XML files with a structure that is shown in code Listing 5.3. The entry in the example begins with the tag <entry type=“twin_entry”> and ends with the tag </entry>. The format showed is only one of several formats that were designed for the different types of entries.

```
<contents>
    <twin_entries>
        <entry type="twin_entry">
            <id>1</id>
            <location>3</location>
            <text>
                <eng>text for entry</eng>
                <swe>text till inlägget</swe>
            </text>
        </entry>
    </twin_entries>
</contents>
```

Listing 5.3: *The structure for the entry information on one of the pages of a restaurant page.*

Since there were different types of entries, the XML format had to be able to handle different entries with different types of elements for information. Eventually, there were six different types of entries on the pages. The specific entry type was stored as an attribute called “type” in the XML entry element. A number of different entry types were included in the final version of the pages of the web site. There were two types of pages with different types of entries. The first type of page included the Welcome, the restaurant, and Location pages and contained the four following entry types:

Twin entry with the attribute “twin_entry”, consisting of a text to the left and a picture to the right.

Reverse twin entry with the attribute “twin_entry2”, consisting of a picture to the left and a text to the right.

Headline entry with the attribute “twin_headline”, consisting of a simple headline entry with one line of text of a big size.

Image entry with the attribute “twin_image”, consisting of a single image.

The second type of page was the Menu page, which had the two following entry types:

Headline entry with the attribute “menu_headline”, consisting of a simple headline entry with a single line of big white text on a black background.

Dish entry with the attribute “menu_entry”, consisting of all the necessary information and image of a dish, as explained in the concept model.

All modification of the internal contents of the different entries was done by updating the XML contents with the help of the XMLHttpRequest object and PHP server side functions. Also the moving of entries up and down the entry list on a page was done by the same means. Implementing all the functionality was not free from problems but it was completed in a consistent and organized way.

5.4 User Testing

Even if there was no focus on organized user testing during the project, there was a significant amount of user testing taking part during the development of the web site. While developing the conceptual models of the different parts of the web site, friends and family were asked to try using the different functions in order to find out if the ideas were going to be useful once implemented. Several ideas for functions of the web site were redesigned and reworked due to outcomes of the user testing and discussions with different people. While working on the implementation of the web site, users were once again asked to try out the different functions to find out if the ideas had been implemented in a good way.

5.5 Photography

During my working period in South Korea, not much time was dedicated only to photography. Two days were spent specifically for taking photos for the web site. The first day was organized to take photographies of the different dishes in the restaurants. The dishes were chosen by the owner based on which meals were most representative of the menus. The chefs in the three restaurants prepared the meals and I went to the restaurants one by one during the day to take the photographies. Those photos were then edited to fit into the menu pages for the restaurants. The second organized day was supposed to be for taking photographies of the staff in the restaurants. Unfortunately I was only able to take group photos of the staff at Ye-Ga Chon. When I arrived at the other restaurants they had already started preparing for the lunch hour, so it was not possible to gather everyone for group photos.

Instead of having more organized photography sessions, I brought the camera everywhere I went in Gongju in order to get some representative photos of motifs related to the restaurants. Since I visited the three restaurants to eat during many occasions, taking photos of cooked food was done continuously. On a few occasions, I went out to different scenic spots in Gongju to take photos of the surroundings both during day time and night time.

Chapter 6

Results

This chapter presents the finished web site design and explains the different functions and possibilities with the CMS.

6.1 Pages of the Web Site

The figures in this section are example versions of some of the pages for the three restaurants. All contents on all of the pages of the restaurants are available in both Korean and English. The page The Restaurant is not shown here since it has the exact same structural design as the Welcome page. Figure 6.1 shows the splash screen for the whole web site. The page contains navigation links to the web pages for the three restaurants and the lock image in the lower right corner links to the admin pages.

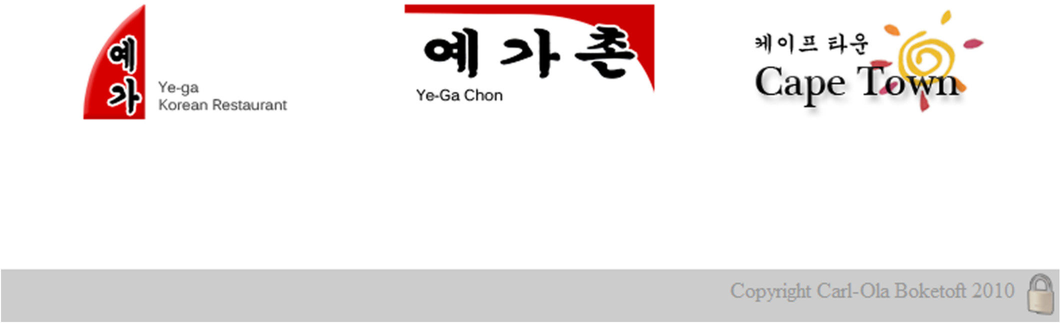
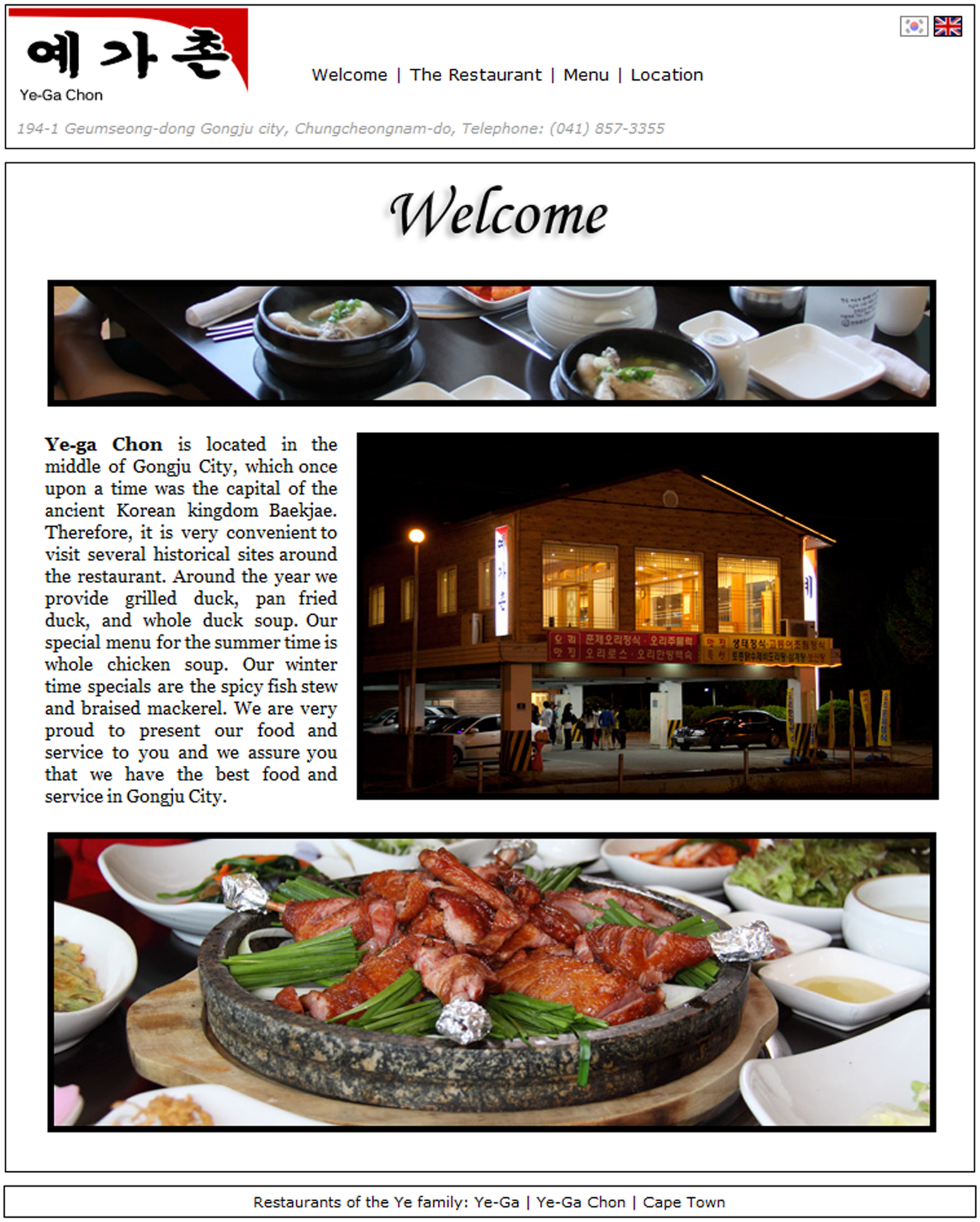


Figure 6.1: *Choose which restaurant web page to visit, or log in to the admin pages.*

An example of the Welcome page, from the guest view, for the restaurant Ye-Ga Chon is shown in Figure 6.2.

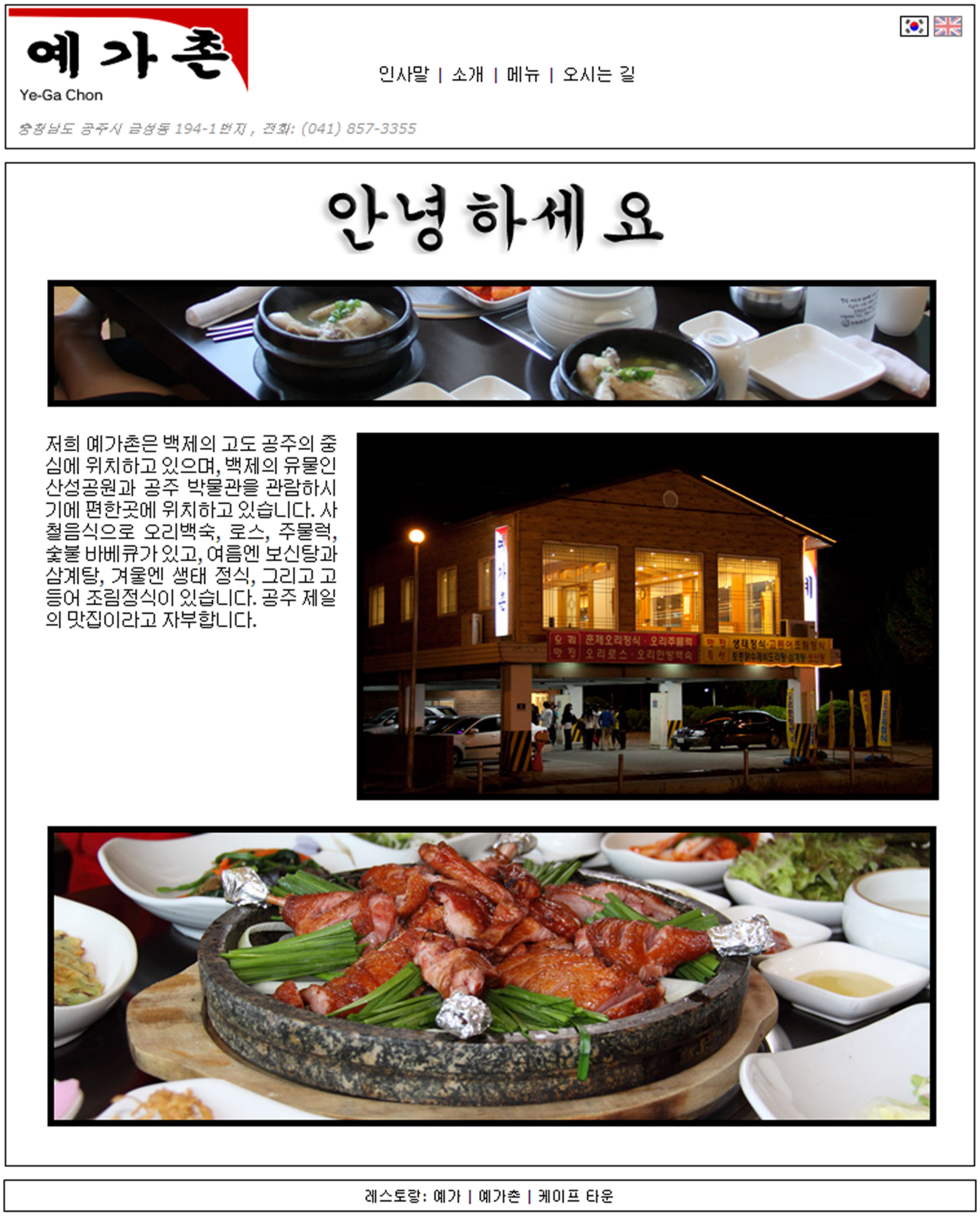

예가촌
Ye-Ga Chon
Welcome | The Restaurant | Menu | Location
Welcome
Restaurants of the Ye family: Ye-Ga | Ye-Ga Chon | Cape Town

194-1 Geumseong-dong Gongju city, Chungcheongnam-do, Telephone: (041) 857-3355

Ye-ga Chon is located in the middle of Gongju City, which once upon a time was the capital of the ancient Korean kingdom Baekjae. Therefore, it is very convenient to visit several historical sites around the restaurant. Around the year we provide grilled duck, pan fried duck, and whole duck soup. Our special menu for the summer time is whole chicken soup. Our winter time specials are the spicy fish stew and braised mackerel. We are very proud to present our food and service to you and we assure you that we have the best food and service in Gongju City.

Figure 6.2: *An example of how the English Welcome page, from the guest view, for the restaurant Ye-Ga Chon could look like.*

Figure 6.3 shows an example of the same Welcome page, but this time in Korean.


예가촌
Ye-Ga Chon
인사말 | 소개 | 메뉴 | 오시는 길

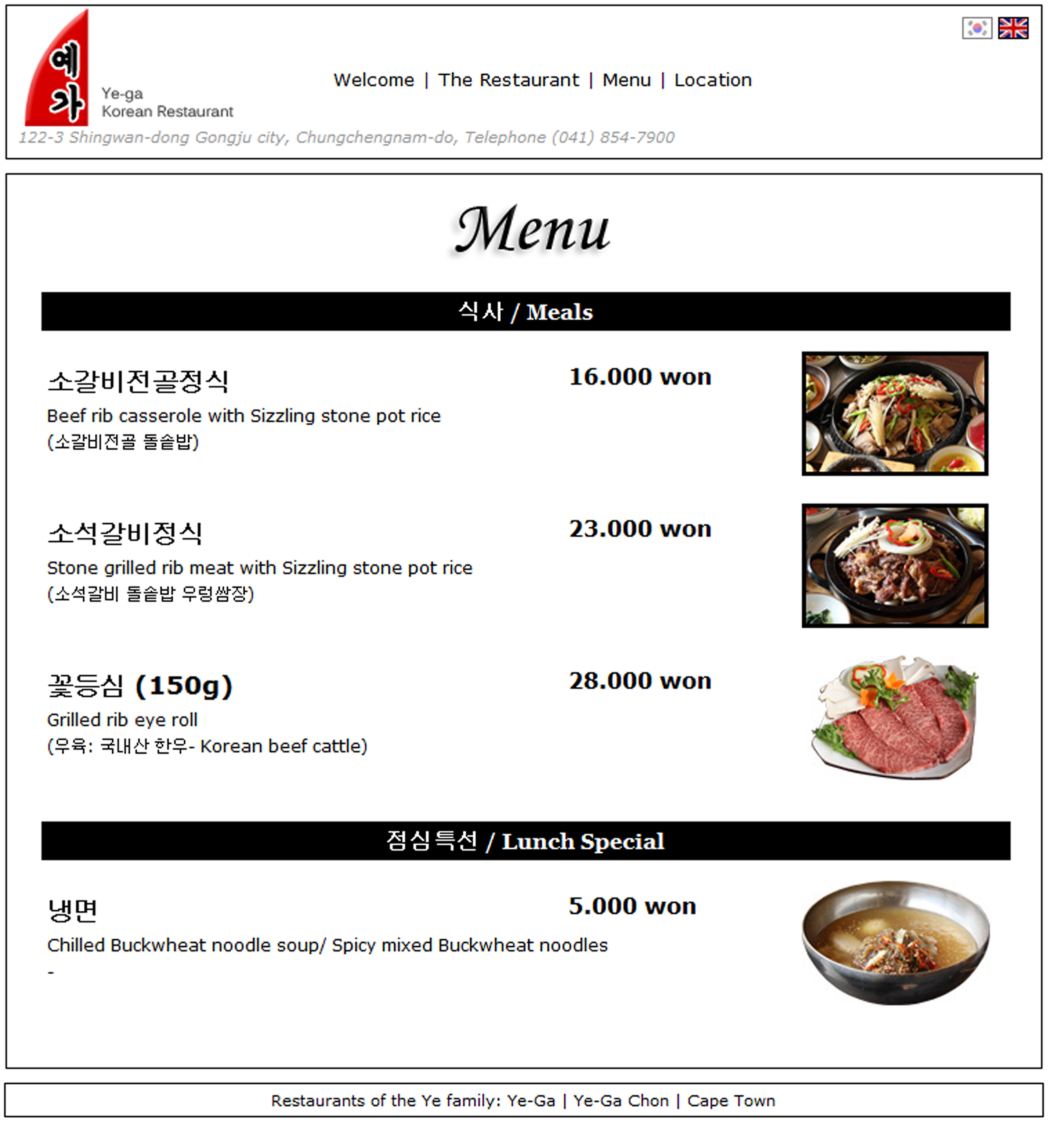
충청남도 공주시 금성동 194-1번지 , 전화: (041) 857-3355

안녕하세요

저희 예가촌은 백제의 고도 공주의 중심에 위치하고 있으며, 백제의 유물인 산성공원과 공주 박물관을 관람하시기에 편한곳에 위치하고 있습니다. 사철음식으로 오리백숙, 로스, 주물럭, 숯불 바베큐가 있고, 여름엔 보신탕과 삼계탕, 겨울엔 생태 정식, 그리고 고등어 조림정식이 있습니다. 공주 제일의 맛집이라고 자부합니다.

Figure 6.3: *An example of how the Korean Welcome page for the restaurant Ye-Ga Chon could look like.*

All three restaurants have extensive menus, and the example in Figure 6.4 shows a small selection of dishes from the menu of the restaurant Ye-Ga.



122-3 Shingwan-dong Gongju city, Chungchengnam-do, Telephone (041) 854-7900

Menu

식사 / Meals

소갈비전골정식 — 16.000 won
Beef rib casserole with Sizzling stone pot rice
(소갈비전골 돌솥밥)

소석갈비정식 — 23.000 won
Stone grilled rib meat with Sizzling stone pot rice
(소석갈비 돌솥밥 우렁쌈장)

꽃등심 (150g) — 28.000 won
Grilled rib eye roll
(우육: 국내산 한우- Korean beef cattle)

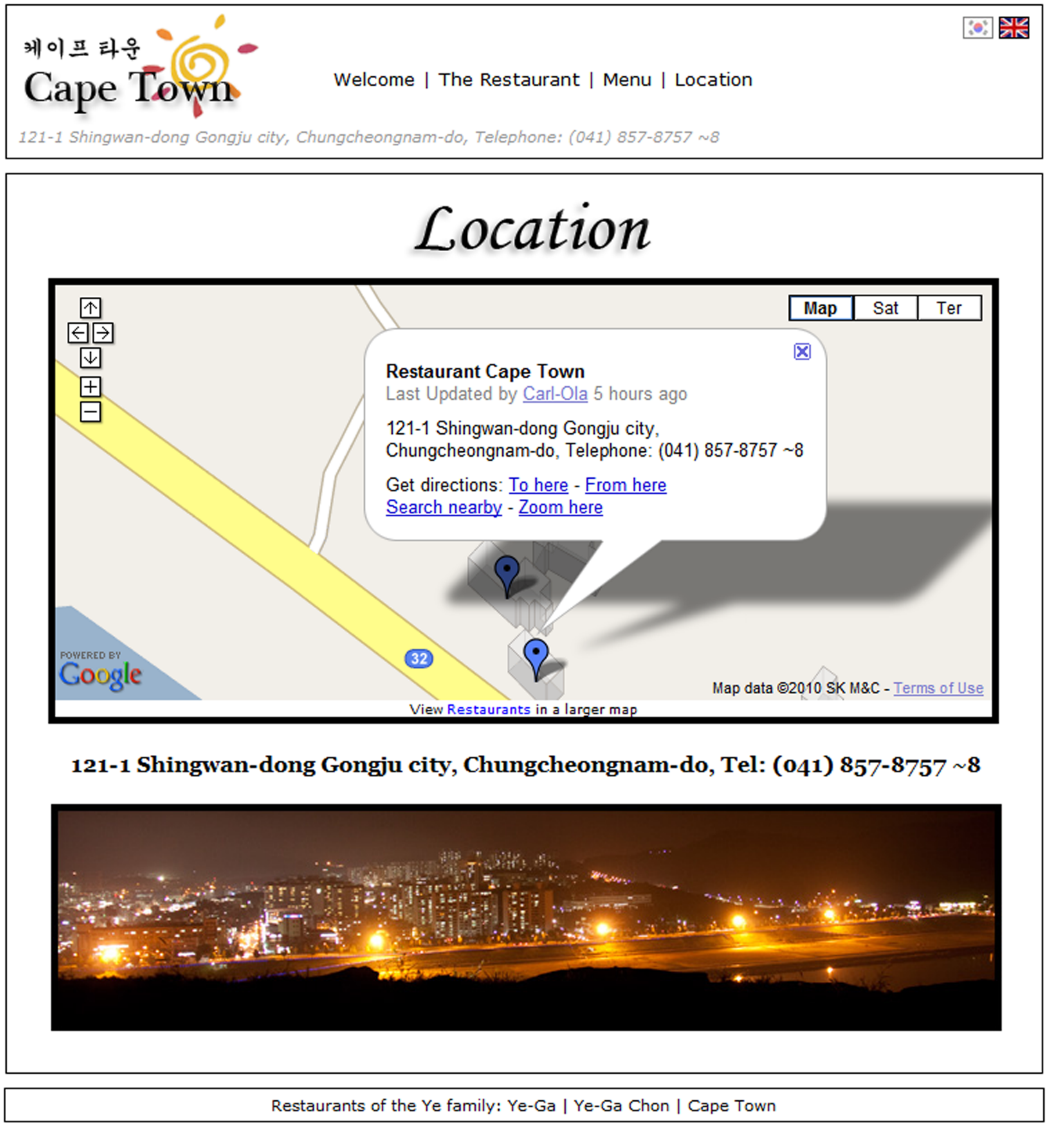
점심특선 / Lunch Special

냉면 — 5.000 won
Chilled Buckwheat noodle soup/ Spicy mixed Buckwheat noodles
-

Restaurants of the Ye family: Ye-Ga | Ye-Ga Chon | Cape Town

Figure 6.4: *An example of how the Menu page for the restaurant Ye-Ga could look like.*

The last example, in Figure 6.5, is the Location page for the restaurant Cape Town. When you first arrive at the Location page, the speech bubble shown in the example is not visible, only the map itself is showing. To see the speech bubble you have to click the building for the restaurant Cape Town, which is centered on the map. All the maps for the three restaurants are in the form of embedded maps from Google Maps. The map is always located above the entries, right under the page title, and can not be moved, resized, or edited in any other way from the CMS.


케이프 타운
Cape Town
Welcome | The Restaurant | Menu | Location
Map
Sat
Ter
POWERED BY Google
32
View Restaurants in a larger map

121-1 Shingwan-dong Gongju city, Chungcheongnam-do, Telephone: (041) 857-8757 ~8

Location

Restaurant Cape Town
Last Updated by Carl-Ola 5 hours ago

121-1 Shingwan-dong Gongju city,
Chungcheongnam-do, Telephone: (041) 857-8757 ~8

Get directions: To here - From here
Search nearby - Zoom here

Map data ©2010 SK M&C - Terms of Use

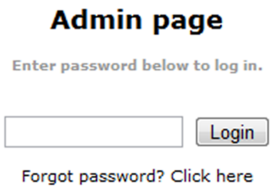
121-1 Shingwan-dong Gongju city, Chungcheongnam-do, Tel: (041) 857-8757 ~8

Restaurants of the Ye family: Ye-Ga | Ye-Ga Chon | Cape Town

Figure 6.5: *An example of how the Location page for the restaurant Cape Town could look like.*

6.2 Administrator Pages

Before entering the administrator settings page, it is necessary to log in with an administrator password. The administrator login page is shown in Figure 6.6. Everything from the login page is shown in the example, except the pictures of flags for language change. The two things that can be done on this page is either to log in with the administrator password or to retrieve the password, for a registered administrator.

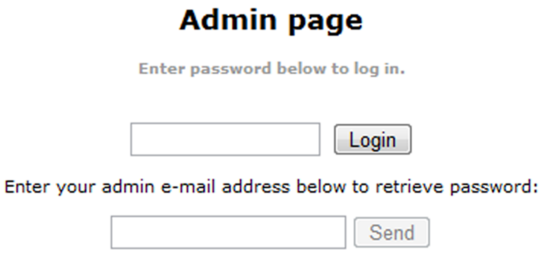


Admin page

Enter password below to log in.

Figure 6.6: *The administrator login page.*

If an administrator has lost the password, and he has a registered administrator e-mail address, he should click the text that says "Forgot password? Click here". The interface that appears after clicking the text is shown in Figure 6.7. The field that appears is where the registered administrator e-mail address is entered. The button for retrieving the password, send, is only possible to click if a well formed e-mail address is entered in the field.



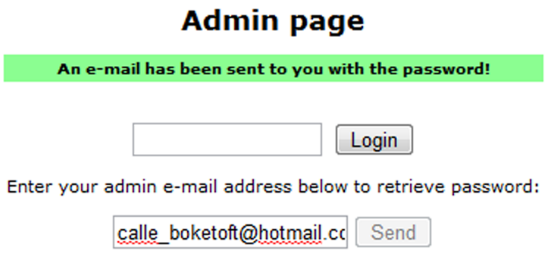
Admin page

Enter password below to log in.

Enter your admin e-mail address below to retrieve password:

Figure 6.7: *The interface for retrieving a lost password.*

If an administrator enters his registered administrator e-mail address into the field and press send, an e-mail with the password will be sent to his e-mail, like shown in Figure 6.8. The green field in the figure is a status message that is shown for three seconds before disappearing again. There are many different types of status messages for different actions on the web site. For example if someone enters an unregistered e-mail address into the field in the figure, there will be a status message that tells the user that the e-mail address is not registered.



Admin page

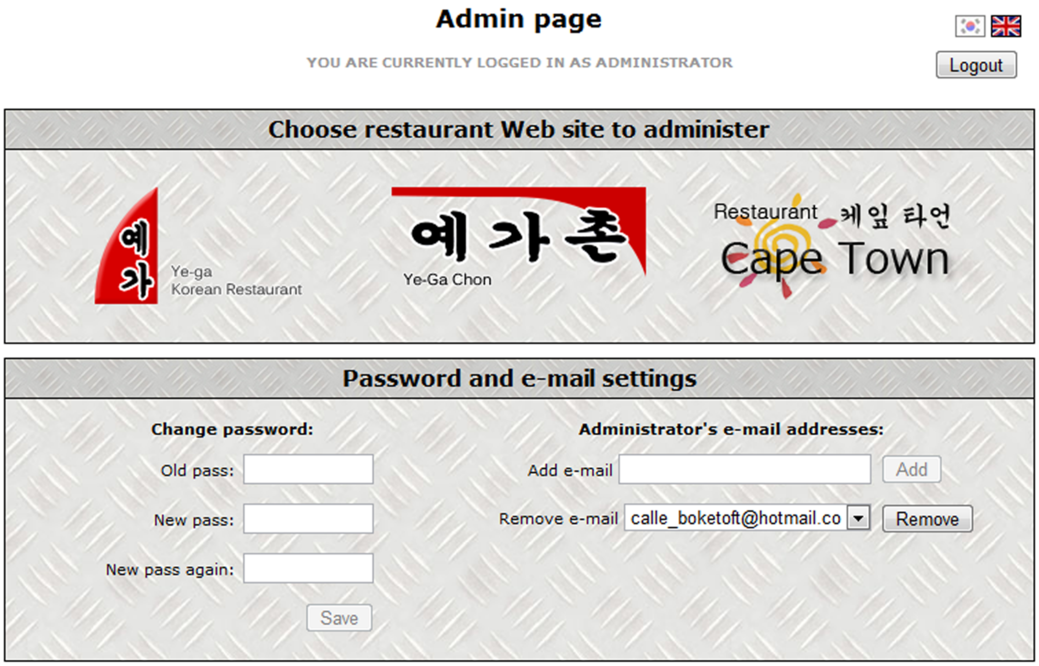
An e-mail has been sent to you with the password!

Enter your admin e-mail address below to retrieve password:

calle_boketoft@hotmail.cc

Figure 6.8: *A successful retrieval of a lost administrator password.*

After successfully logging in, the administrator gets to the administrator settings page, which is shown in Figure 6.9. The administrator settings include functions for changing the password for the web site and to add or remove administrator e-mail addresses. It is possible to distinguish administrator settings frames on the web site by their gray diamond plate backgrounds. The login function is not individual since all administrators share the same password for entering the CMS. The persons who can retrieve a lost password, by using the password retrieval function on the administrator login page, must have an e-mail address that is registered as an administrator e-mail address. If an administrator wants to go on straight to the CMS of one of the tree restaurants, he simply clicks the restaurant logo in the upper frame.



Admin page

YOU ARE CURRENTLY LOGGED IN AS ADMINISTRATOR

Choose restaurant Web site to administer

Password and e-mail settings

Change password:

Old pass:

New pass:

New pass again:

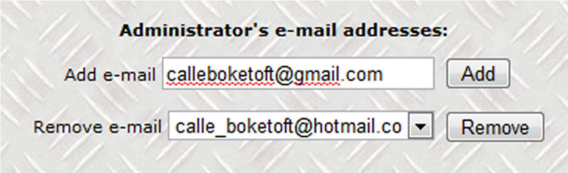
Administrator's e-mail addresses:

Add e-mail

Remove e-mail calle_boketoft@hotmail.co

Figure 6.9: *The administrator settings page. The administrator can choose either to click one of the restaurant logos to go to the CMS for the restaurant pages or to manage the administrator password and e-mail addresses. Frames that have gray diamond plate backgrounds are only available in the CMS view.*

The administrator e-mail managing interface has functions for adding and removing administrator e-mail addresses. In the example in Figure 6.10, the e-mail address "calle_boketoft@hotmail.com" is already a registered administrator e-mail address, and thus appears in the pop up menu with registered e-mail addresses. To remove an administrator e-mail address, select the address in the pop up menu and click the remove button. The address "calleboketoft@gmail.com" has been entered in the field for adding administrator e-mail addresses. To add the address to the list, just push the add button.



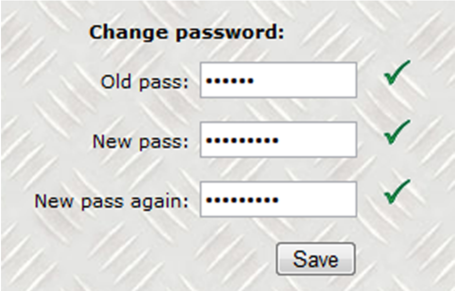
Administrator's e-mail addresses:

Add e-mail calleboketoft@gmail.com

Remove e-mail calle_boketoft@hotmail.co

Figure 6.10: *On the administrator settings page there are possibilities for managing the administrator e-mail addresses.*

It is also possible to change the administrator password for the web site. The interface for password change is shown in Figure 6.11. There are three fields that need to be filled out in order to change the password. Before the fields have been filled out correctly, the save button is disabled and thus not possible to click. The first field is for the old password, and if the old password is typed correctly, a green check mark will appear next to the field. The two other fields are for the new password. If those two fields are filled out in the exact same way, green check marks will appear next to those fields as well. When all three fields have been filled out correctly, and there are green check marks next to all fields, the save button will become enabled and possible to click in order to save the changes.



Change password:

Old pass: •••••• ✓

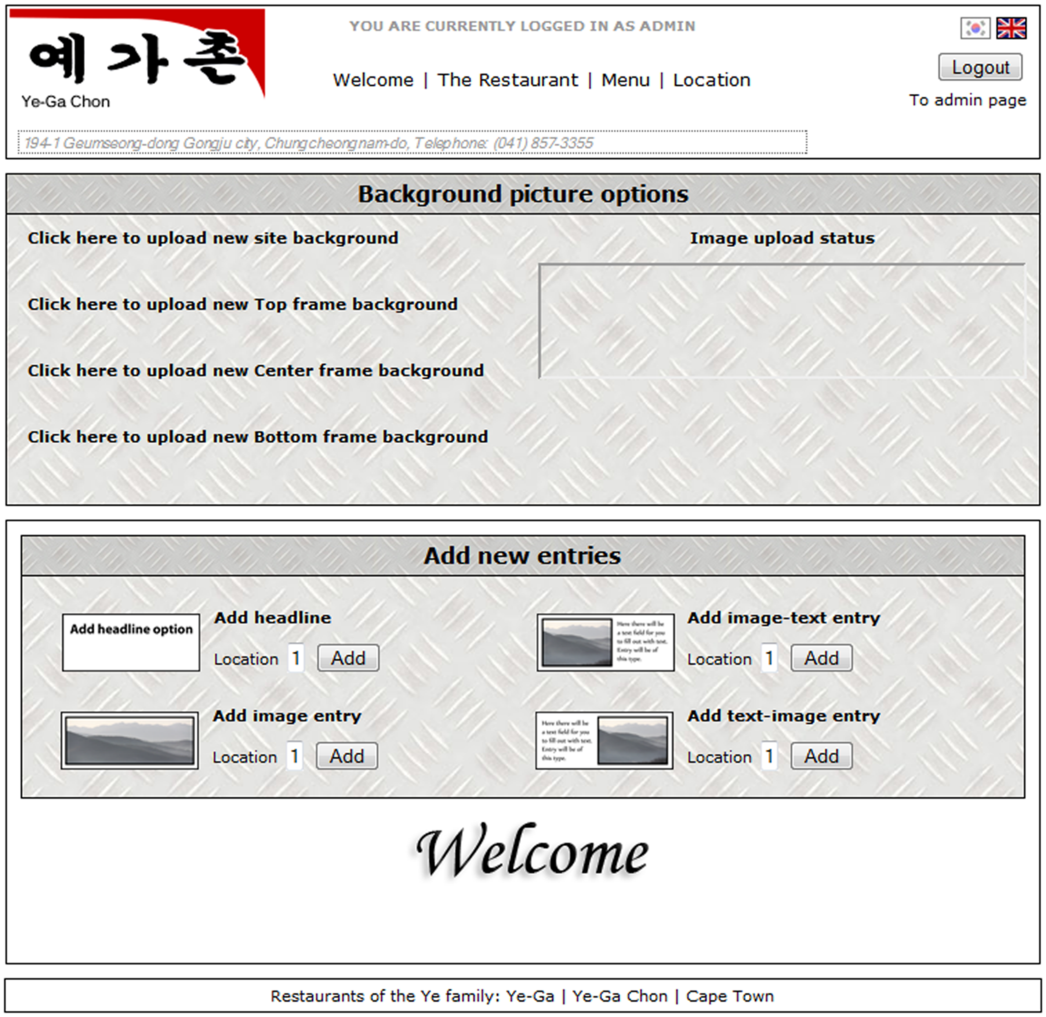
New pass: ••••••••• ✓

New pass again: ••••••••• ✓

Figure 6.11: *The interface for changing the administrator password. It is necessary to fill out the fields correctly in order to save the new password.*

6.3 CMS Functions

The CMS view of the web site contains quite many functions, but all functions that are used for similar purposes are designed to work in very similar ways. The two main functions of the CMS is to upload pictures and write texts directly in the web browser. In Chapter 5, most of the concepts for the functionality were explained, but since the implemented functions did not turn out to work in the exact same way as thought out in the concept models, they will be explained in detail here as well. In Figure 6.12 an example with the Welcome page for the restaurant Ye-Ga Chon, from the CMS view, is shown. The page is empty from entries, so there is nothing except the page title that says "Welcome" on the page. The bottom frame, with the names of all the three restaurants, is a navigation bar for going to the different restaurants' pages. The frame is available both in the CMS and the guest view.


YOU ARE CURRENTLY LOGGED IN AS ADMIN

194-1 Geumseong-dong Gongju city, Chungcheongnam-do, Telephone: (041) 857-3355

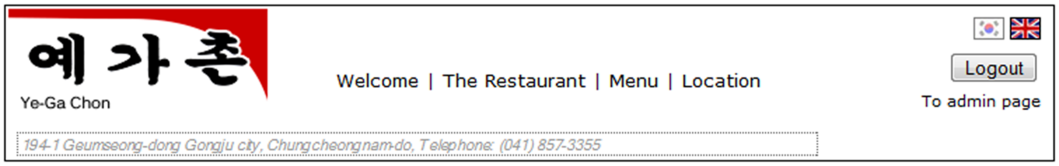
예가촌
Ye-Ga Chon
Welcome | The Restaurant | Menu | Location
Logout
To admin page
Background picture options
Click here to upload new site background
Click here to upload new Top frame background
Click here to upload new Center frame background
Click here to upload new Bottom frame background
Image upload status
Add new entries
Add headline option
Add headline
Location 1 Add
Add image-text entry
Location 1 Add
Add image entry
Location 1 Add
Add text-image entry
Location 1 Add
Welcome
Restaurants of the Ye family: Ye-Ga | Ye-Ga Chon | Cape Town

Figure 6.12: *An example of the CMS view of the Welcome page for the restaurant Ye-Ga Chon.*

6.3.1 Image Uploading

The following figures, Figure 6.13 to Figure 6.17, illustrates an example of the procedure for uploading a new image on the web site. The image in the example is the restaurant logo for the restaurant Ye-Ga Chon but the procedure for image uploading is the same for all dynamic images on the site.

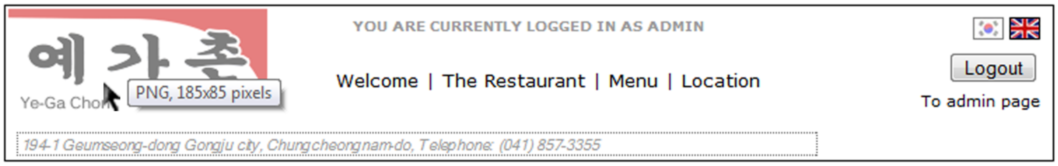
Figure 6.13 shows what the interface looks like before anything has been done.



194-1 Geumseong-dong Gongju city, Chungcheongnam-do, Telephone: (041) 857-3355

Figure 6.13: *The CMS interface for the navigation frame.*

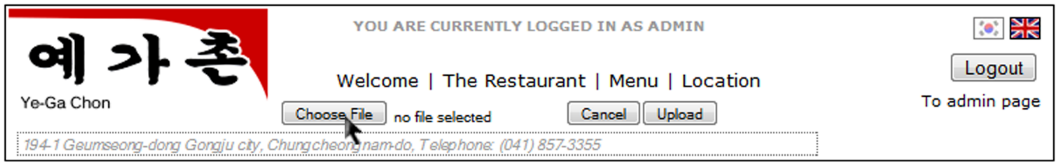
Figure 6.14 shows what it will look like when the administrator has moved the cursor over the image. The image fades out a bit and a tool tip appears with the text "PNG, 185x85 pixels". The tool tip contains all the information needed for creating and uploading a new image in the location. The file format in this example should be PNG, and the image should have the size dimensions 185 pixels by 85 pixels to fit in the spot. In other locations on the page where image uploading is possible, the necessary information for the image to be uploaded is shown in the tool tip, just as in this example.


YOU ARE CURRENTLY LOGGED IN AS ADMIN

194-1 Geumseong-dong Gongju city, Chungcheongnam-do, Telephone: (041) 857-3355

Figure 6.14: *The administrator has moved the cursor over the image.*

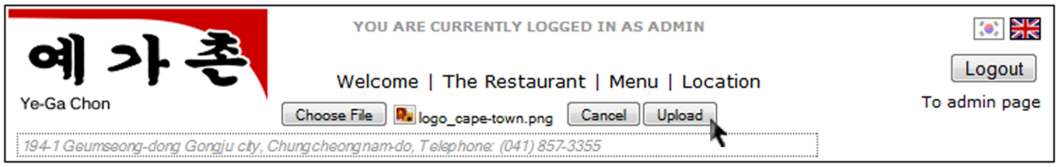
In Figure 6.15, the administrator has clicked the image. Three buttons appear on the screen, close to the picture, that say "choose file", "cancel", and "upload". If he wants to upload an image file, he should click the "choose file" button and select an image from his computer.


YOU ARE CURRENTLY LOGGED IN AS ADMIN

194-1 Geumseong-dong Gongju city, Chungcheongnam-do, Telephone: (041) 857-3355

Figure 6.15: *The administrator has clicked the image.*

In Figure 6.16, an image called "logo_cape-town.png" has been chosen from the computer. It is important that the image to be uploaded conforms to the rules that were shown in the tool tip. If the administrator tries to upload an image that does not correspond to the rules, either it will not fit well into the spot or it will not be uploaded.


YOU ARE CURRENTLY LOGGED IN AS ADMIN

194-1 Geumseong-dong Gongju city, Chungcheongnam-do, Telephone: (041) 857-3355

Figure 6.16: *A new image has been chosen.*

In Figure 6.17 the user has pressed the upload button and the image upload function has been activated. The image was successfully uploaded and refreshed to show on the page.

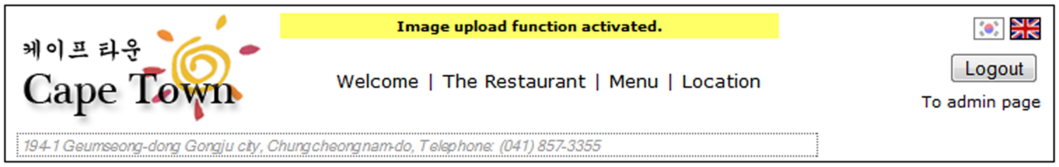
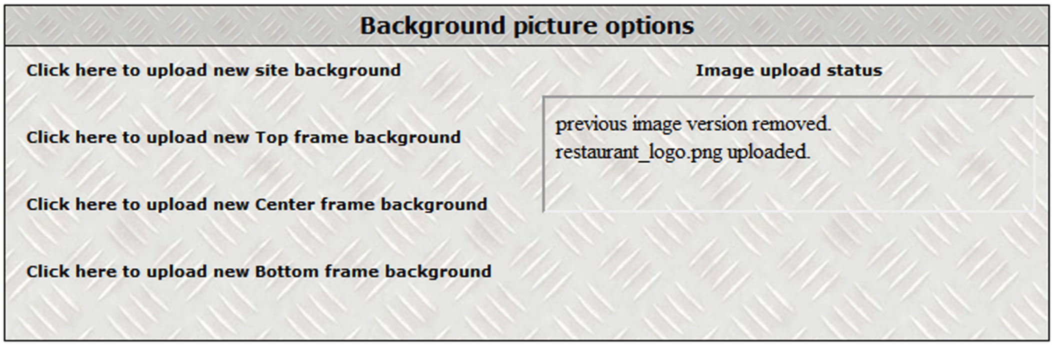


Image upload function activated.

194-1 Geumseong-dong Gongju cty, Chungcheongnam-do, Telephone: (041) 857-3355

Figure 6.17: *The new image has been successfully uploaded.*

Whenever an image upload button has been pressed, a yellow status message notifies the administrator that the image upload function has been activated. Since the image uploading had to be dealt with through an iframe, the status messages for all image uploading results are shown to the right in the frame for background picture options. In Figure 6.18 the status message for a successfully uploaded restaurant logo image is shown to the right. The message says that the previous image was removed and that a new image was uploaded in its place. The four texts to the left in the frame are for uploading backgrounds for the different sections of the framework for the web site. When clicking one of the texts, the three buttons for uploading an image appears under the text. They work in the same way as the buttons in the example above.


Background picture options
Click here to upload new site background
Click here to upload new Top frame background
Click here to upload new Center frame background
Click here to upload new Bottom frame background
Image upload status

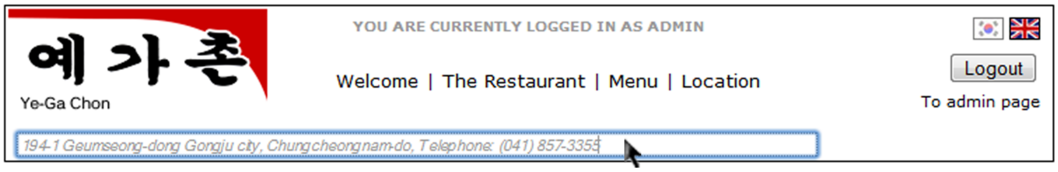
previous image version removed.
restaurant_logo.png uploaded.

Figure 6.18: *The administrator interface for background image uploading and image upload status messages.*

6.3.2 Text Editing

The second major part of the CMS is the text editing. In Figures 6.19 to 6.21 is an example of how text editing in the CMS works.

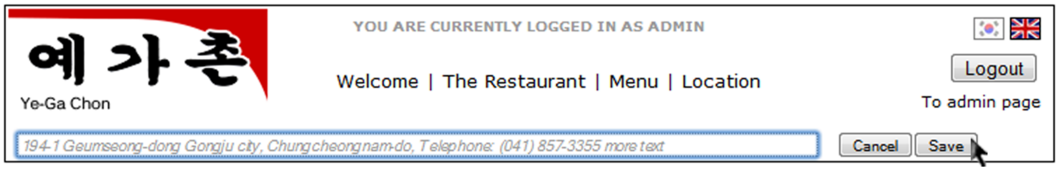
Figure 6.19 shows how the address field for the restaurant Ye-Ga Chon looks like after it has been selected by a mouse click by the administrator (the blue line that shows around the text field look differently in different web browsers).


예가촌
Ye-Ga Chon
YOU ARE CURRENTLY LOGGED IN AS ADMIN
Welcome | The Restaurant | Menu | Location
Logout
To admin page

194-1 Geumseong-dong Gongju city, Chungcheongnam-do, Telephone: (041) 857-3355

Figure 6.19: *The text field has been selected.*

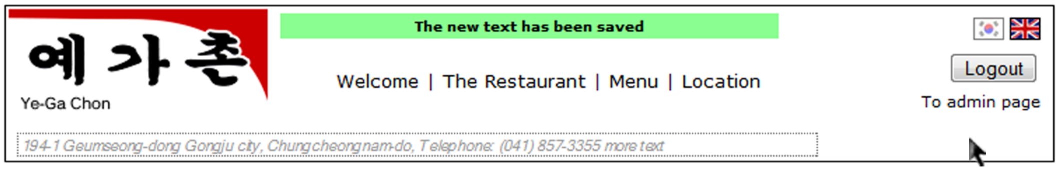
In Figure 6.20 the administrator has typed the text “more text” after the telephone number in the address. A pair of buttons appear, “cancel” and ”save”. If the cancel button is pressed, the text in the field will revert to the text that was last saved. Pressing the save button will save the text that is currently in the text field into the corresponding XML document.


예가촌
Ye-Ga Chon
YOU ARE CURRENTLY LOGGED IN AS ADMIN
Welcome | The Restaurant | Menu | Location
Logout
To admin page
Cancel
Save

194-1 Geumseong-dong Gongju city, Chungcheongnam-do, Telephone: (041) 857-3355 more text

Figure 6.20: *The administrator has written more text into the text field.*

After pressing the save button, like shown in Figure 6.21, the two buttons will disappear again, and a status message will appear for three seconds.



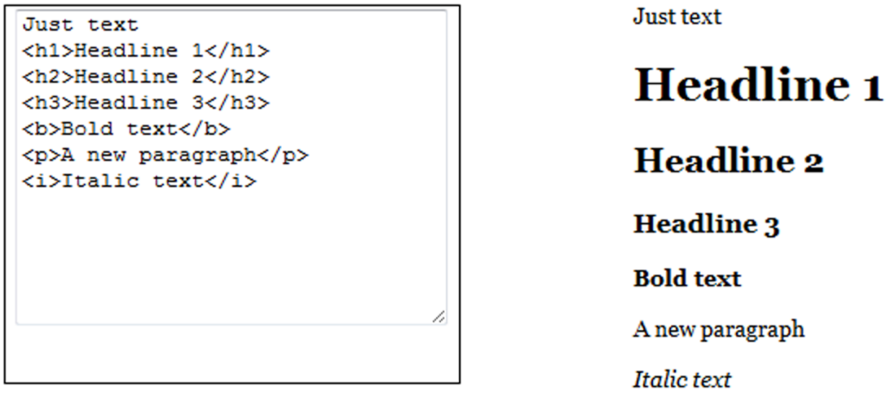
The new text has been saved

194-1 Geumseong-dong Gongju city, Chungcheongnam-do, Telephone: (041) 857-3355 more text

Figure 6.21: *The administrator has pressed the save button and the text has been saved.*

6.3.3 HTML Formatting Within Text Boxes

When text is written inside a text box in the CMS view, the formatting of the text when viewed in the guest view will be as predefined in hardcode for each entry type. For example, when normal text (without HTML tags) is written inside the headline entries on the Welcome page it will become bold in the guest view. The implementation of the web site was done this way since the CMS functions are supposed to be used by persons who are not used to any type of programming, so knowing HTML text formatting should not be necessary. The predefined formatting is not mandatory to use, though. If HTML tags are written within the text boxes of the different entry types, they will format the text accordingly as they are shown in the guest view. An example of this feature is shown in Figure 6.22.



```
Just text
<h1>Headline 1</h1>
<h2>Headline 2</h2>
<h3>Headline 3</h3>
<b>Bold text</b>
<p>A new paragraph</p>
<i>Italic text</i>
```

Just text

Headline 1

Headline 2

Headline 3

Bold text

A new paragraph

Italic text

Figure 6.22: *To the left in the image is an entry from the CMS view where HTML tags have been written inside the text box. To the right in the image is the same entry from the guest view. The text has been formatted by the HTML tags that were written in the CMS view.*

6.3.4 Working with Entries

Since all pages of the web site are initially empty, except for the page title and the map on the Location page, it is necessary to add different types of entries on the pages. In the concept models, there was only one type of entry, the pair with a picture to the left and a text to the right. In an early version of the implementation it was only possible to add that type of entry. The width of the image was limited to a maximum width and a minimum width in order to ensure that the pair would be balanced in size between the text and the image. After trying to add a couple of entries with pairs it became obvious that it was necessary to have more types of entries to get a more varying layout. Three new types of entries were created to be used on the pages Welcome, The Restaurant, and Location. The CMS functions for adding the entries are shown in Figure 6.23. The first entry type is the headline, which is an entry with one line of text with a bold font. The second new type of entry is the image entry with a centered image, and no text. The width of the image is limited to a maximum width to fit the width of the page, but the height is free to choose for the administrator. The third new entry type is the pair with a text to the left and image to the right. The image in the pair has a minimum and maximum width.

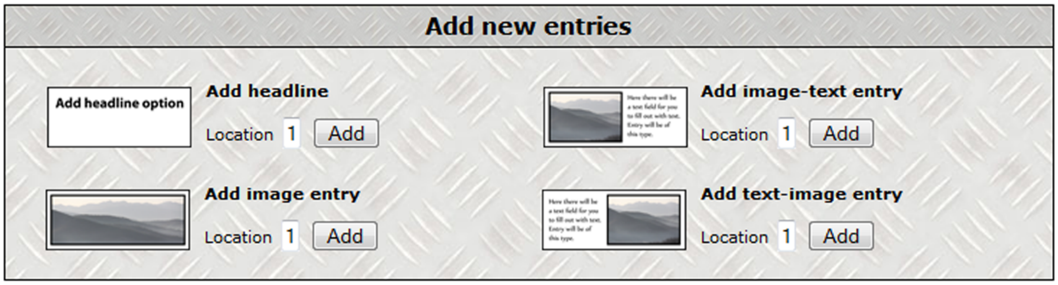
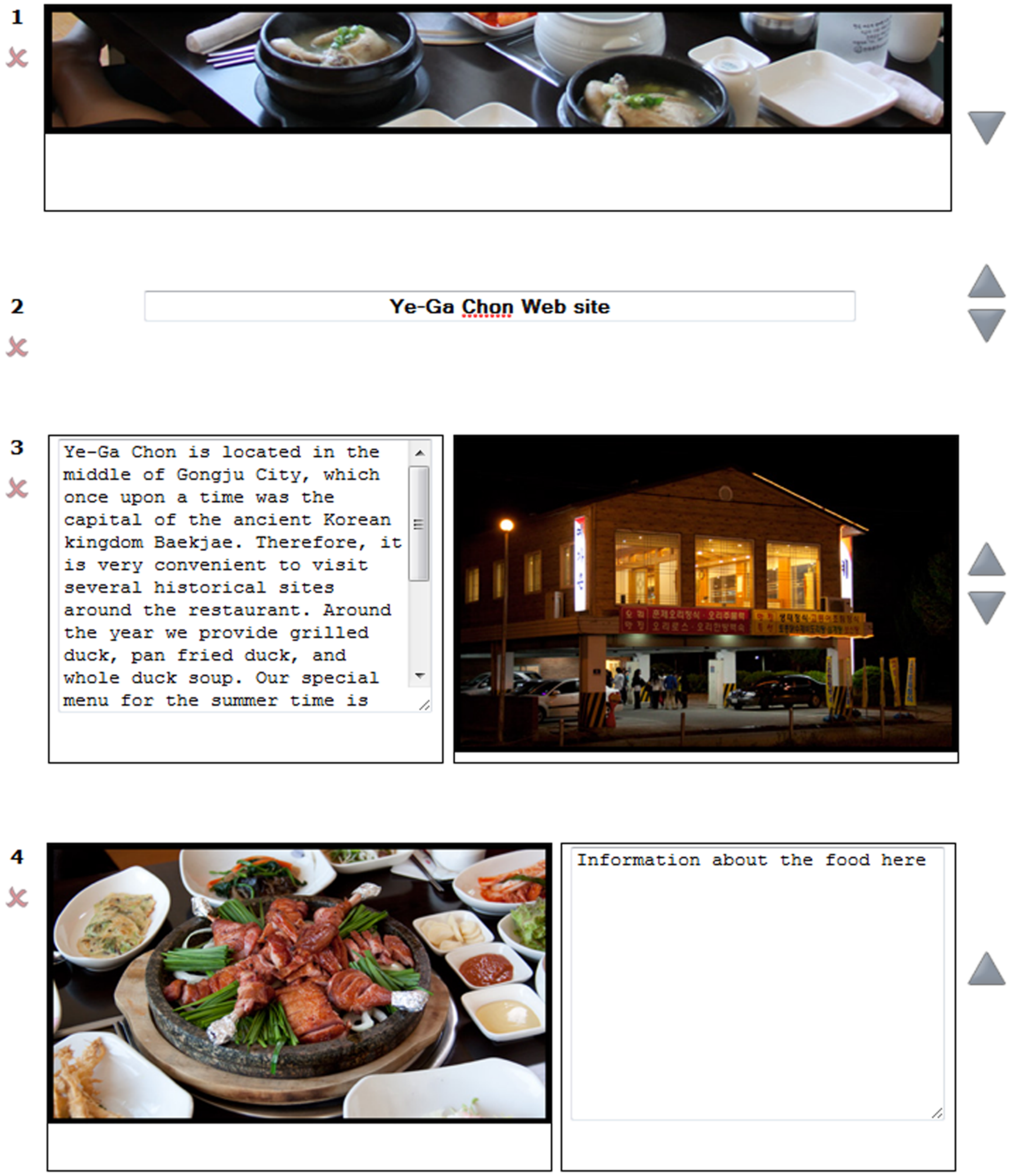

Add new entries
Add headline option
Add headline
Location 1 Add
Add image-text entry
Here there will be a text field for you to fill out with text. Entry will be of this type.
Location 1 Add
Add image entry
Location 1 Add
Add text-image entry
Here there will be a text field for you to fill out with text. Entry will be of this type.
Location 1 Add

Figure 6.23: *The CMS functions for adding new entries on the pages Welcome, The Restaurant, and Location.*

When adding entries and looking at them from the CMS, the size of the containers for the entries are of constant height and width in order to get an organized CMS view. An example of added entries of all four types can be seen in Figure 6.24. The First entry from the top is the image entry, which consists of a single image. The second entry type is the headline, consisting of one line of bold text. The third entry is a pair with a text to the left and image to the right. The fourth entry at the bottom is the pair with an image to the left and a text to the right.



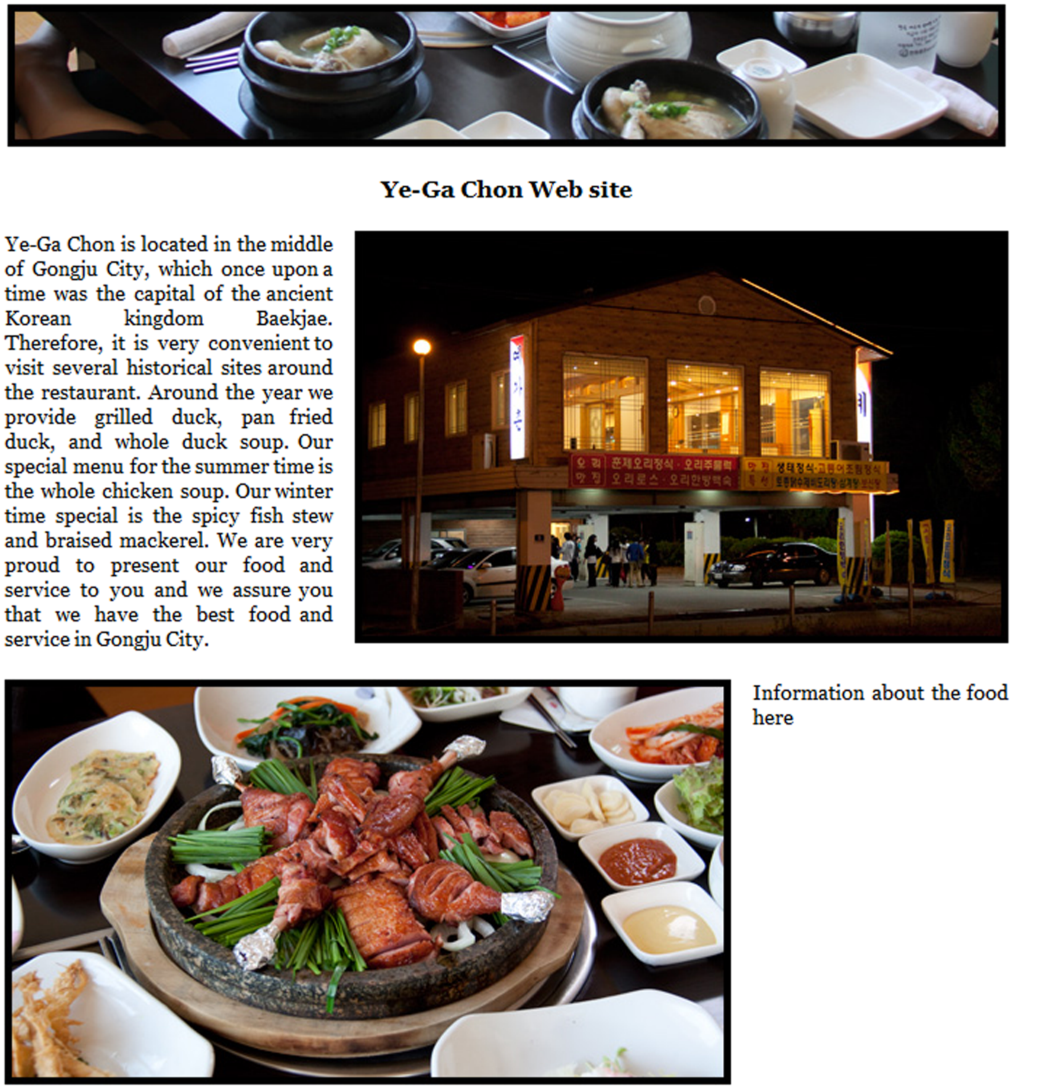
Ye-Ga Chon Web site

Ye-Ga Chon is located in the middle of Gongju City, which once upon a time was the capital of the ancient Korean kingdom Baekjae. Therefore, it is very convenient to visit several historical sites around the restaurant. Around the year we provide grilled duck, pan fried duck, and whole duck soup. Our special menu for the summer time is

Information about the food here

Figure 6.24: *All the four entry types from the CMS view.*

When the same page is shown from the guest view, it looks like in Figure 6.25. There are no longer the thin frames from the CMS view around the entries (the thick black frame around the images are inside the images), the editing functions are gone, and the images and texts get their proper sizes.



Ye-Ga Chon Web site

Ye-Ga Chon is located in the middle of Gongju City, which once upon a time was the capital of the ancient Korean kingdom Baekjae. Therefore, it is very convenient to visit several historical sites around the restaurant. Around the year we provide grilled duck, pan fried duck, and whole duck soup. Our special menu for the summer time is the whole chicken soup. Our winter time special is the spicy fish stew and braised mackerel. We are very proud to present our food and service to you and we assure you that we have the best food and service in Gongju City.

Information about the food here

Figure 6.25: *The four entry types from the guest view.*

The Menu page was implemented with the two entry types that were described in the concept model in Chapter 5. The first type of entry is the headline, with a single line of white bold text on a black background. The second type of entry is the dish entry, with all the necessary information for one dish in a restaurant. The CMS functions for adding the two entry types is shown in 6.26.

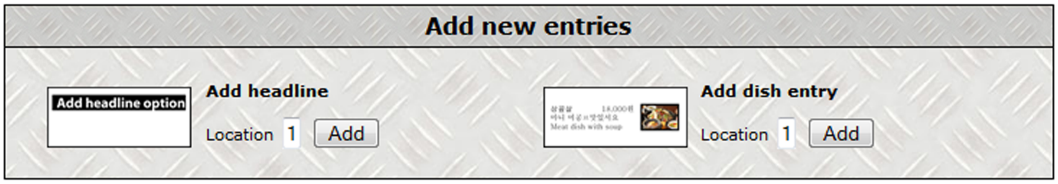
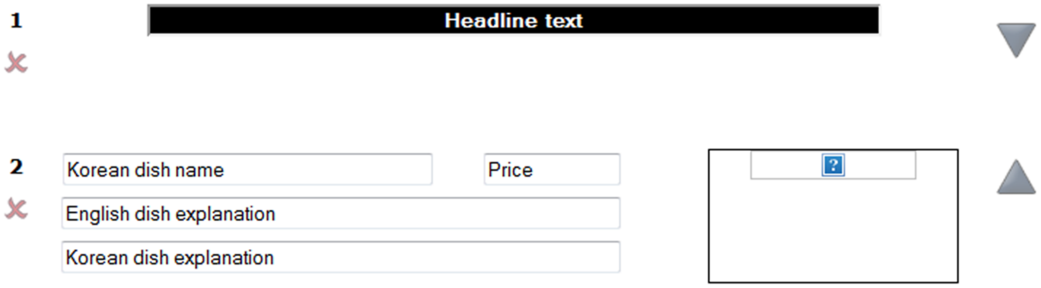


Figure 6.26: *The CMS functions for adding new entries on the Menu page.*

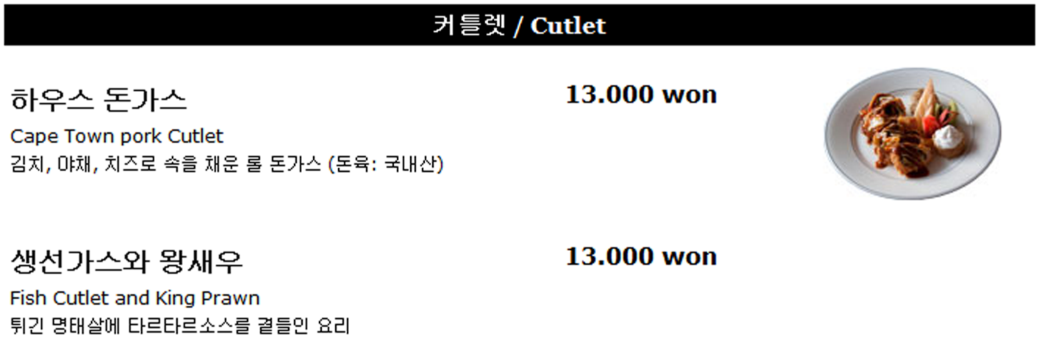
When adding the Menu page entries in the CMS view, it will look like in Figure 6.27. In this example nothing has been written into the entries and no image was uploaded for the dish. Upon adding a new entry, the fields are filled out with a standard text that explains what to type in the fields.


1 Headline text

2 Korean dish name Price
English dish explanation
Korean dish explanation

Figure 6.27: *The two entry types on the Menu page, from the CMS view.*

One thing that is special for the menu entries is that they look exactly the same irrespective of choice of language. This because they have texts in both languages. In Figure 6.28, with three menu entries shown (a headline and two dishes) from the guest view, the first dish has been provided with an image while the second dish has not.


커틀렛 / Cutlet

하우스 돈가스 13.000 won
Cape Town pork Cutlet
김치, 야채, 치즈로 속을 채운 롤 돈가스 (돈육: 국내산)

생선가스와 왕새우 13.000 won
Fish Cutlet and King Prawn
튀긴 명태살에 타르타르소스를 곁들인 요리

Figure 6.28: *Three entries from the guest view on the Menu page. A headline and two dishes.*

Chapter 7

Conclusions

This chapter summarizes the project and draws conclusions from the different parts of the working process.

7.1 Working in South Korea

The plan for the project was defined with clearly separated and solid steps. Having the steps for the different parts of the work process facilitated the work flow and took away confusion about where to work and what was coming next in the project. Another advantage with a well made plan was also that it described what was needed and what was needed to discuss with involved stakeholders and how to properly document the different parts of the work.

The initial time plan for the project was sketched up quickly by evaluating the magnitude of the steps of the process as well as based on my experience of how much time the different parts of the work would take and how much time I thought I would need for logistical matters. The decision to place a big part of the work in South Korea, close to the restaurants, turned out to be beneficial for the project in several ways. An important advantage of working in direct collaboration with the owner of the restaurants as well as being on location was that it gave me the possibility to get a good picture of the restaurant business that I was working for. It was also very practical to be there for the reason of taking photographies and getting other materials for the pages of the web site.

Since only one of the persons related to the restaurants spoke English, I was depending on her as my interpreter during all communication with the other people. Fortunately, she was available most of the time during the work.

For me personally, an important reason to conduct the work in South Korea instead of home in Sweden was to get the experience of living and working in a far away part of the world and to get a glimpse of how different things could work in different countries. As I did not know much about social structures or working conditions in South Korea before my journey, I learned first hand about how big the differences between Sweden and South Korea are.

7.2 Learning Process

Before the project, programming of Ajax functions was unfamiliar to me, and once I figured out how to make the constituents work in symbiosis, it turned out to be an invaluable tool for implementing many useful types of functions. The work to program Ajax functions, while following the usability aspects, went as intended and the results turned out very satisfactory.

Early in the project it became evident that it was a great help to have the usability aspects as guidelines whenever new functions for different purposes were going to be implemented for the web site.

7.3 Design and Implementation

Since there were no requests from the owners about the layout of the web site, it turned out to be both interesting and challenging to have almost complete freedom in the design process. The experiences from the theoretical study about Ajax and usability principles was a great help during the development of the conceptual models.

In the first models, only one type of entry was planned for the different pages. When the choice was made that different entry types should be implemented, it turned out to improve the layout significantly. Apart from design details and small modifications in the layout, the basic structure of the web site was maintained from the first models all the way to the finished product.

When the first pages of the web site had been implemented, it became evident that the XML tree structure used was not going to work for all different elements on the pages. A big work started with adjusting the code and in the end the result was successful. All the functions worked as supposed and not much code adjustments were needed.

An expected difficulty with the implementation was how to solve the encoding of text on the pages since all text was going to be available in both English and Korean. Fortunately the XML format turned out to be very good at handling different types of languages and making it work with English and Korean was no problem at all.

7.4 Goals and Purposes

The purpose from my view, to carry out an international project and learn more about web site design, was well achieved during the work. It was a great journey from first getting the idea for the project to actually seeing how the finished product turned out. In the end, I believe that the finished product became just as good as I had hoped from the beginning, if not better.

The company's goal to spread information about the restaurant to an international audience on the web has not yet been fulfilled, but it is planned that the web site soon will be made available on a Korean web server and exposed on the Internet.

The requirements for the web site were made to be on a reasonable level for the scope of the project and the ones that did not get realized were pushed aside since the web site turned out to be implemented for all three restaurants, and not just Ye-Ga as initially proposed.

7.5 Limitations

An important limitation to be aware of is that the current CMS interface for the web site works best in the web browser Apple Safari. Some of the functions may not work as wanted in other web browsers. This limitation was not seen as serious since the browser Safari is possible to download and use for free for both Mac and PC.

The XML element structure for textual data on the web site was designed to be processed quite fast with the different algorithms, but as compromises with the structure had to be made during the implementation, the algorithm tended to become a bit slow. The speed issue mainly concerns working with entries on the different pages.

7.6 Future work

Future work could include implementing some of the requirements that were discarded when it was decided to develop the web site for all three restaurants instead of just one, such as a function for PDF generation of the menu and a function for seasonal, automatic, updating of the images and texts on the pages depending on the time of year.

At the time of writing, the web pages have not yet been officially published in South Korea nor put up on a Korean web server. This will be done in a near future, in agreement with the owners of the restaurants. At the time of publishing, it is also planned to optimize the web site for search engines and online tourist guides.

A desirable future work would be to improve the implementation of the CMS to make it function properly under more web browsers than only Apple Safari.

Chapter 8

Acknowledgements

I would like to begin by thanking Håkan Gulliksson, my internal supervisor at Umeå University, for all helpful feedback and supervision during the project. Thanks also to Per Lindström, examiner for this project. Special thanks to the owners of the restaurants, Kumeyoung Ye and Eunjun Han, and their family for all support and help during the project while I was in South Korea. Thanks to my girlfriend Hanui Ye for her invaluable help with everything related to our time in South Korea, and especially for her assistance as interpreter during all communication with the people there. Finally I would like to thank my father, Åke Boketoft, for being helpful and supportive during all the different stages of the project and not the least for helping me stay on track with my work through the whole journey.

References

[1] Adobe. Flash content reaches 99% of internet viewers. http://www.adobe.com/products/player_census/flashplayer/, accessed 2009-01-04, 2010.

[2] AjaxPatterns.org. Predictive fetch. http://ajaxpatterns.org/Predictive_Fetch, accessed 2009-09-13, 2010.

[3] Field Expert. Ajax best practices: Don't break the back button. http://www.fieldexpert.com/journal/2006/01/03/ajax-best-practices-dont-break-back//, accessed 2009-09-05, 2010.

[4] Jason Farrell and George S. Nezlek. Rich internet applications the next stage of application development. *Proceedings of the ITI 2007 29th Int. Conf. on Information Technology Interfaces*, 2007.

[5] Jacek Furmankiewicz. Javafx script in action: Lacking for ria, promising for mobile. http://www.devx.com/RichInternetApps/Article/35174, accessed 2009-09-11, 2010.

[6] Jesse James Garrett. Ajax: A new approach to web applications. http://www.adaptivepath.com/ideas/essays/archives/000385.php, accessed 2009-09-03, 2010.

[7] Gmail. Gmail. http://www.gmail.com/, accessed 2010-01-24, 2010.

[8] Google. Google maps. http://maps.google.com/, accessed 2010-01-24, 2010.

[9] jQuery for Designers. Ajax form validation example. http://jqueryfordesigners.com/demo/ajax-validation.php, accessed 2009-09-13, 2010.

[10] Soren Lauesen. *Software Requirements - Styles and Techniques*. Addison-Wesley, 2002.

[11] Tim Berners Lee. Re: status. re: X11 browser for www. http://lists.w3.org/Archives/Public/www-talk/1991SepOct/0003.html, accessed 2009-09-07, 2010.

[12] Antonio Lupetti. Nice ajax effect for message box using mootools. http://woork.blogspot.com/2008/03/nice-ajax-effect-for-message-box-using.html, accessed 2009-09-13, 2010.

[13] NetScript. Netscript whitepaper. http://netscript.com/site/loc/en/profile/NetScript_RIA_WhitePaper.pdf, accessed 2009-09-03, 2010.

[14] Jakob Nielsen. Flash: 99% bad. http://www.useit.com/alertbox/20001029.html, accessed 2009-09-11, 2010.

[15] Jakob Nielsen. Ten usability heuristics. http://www.useit.com/papers/heuristic/heuristic_list.html, accessed 2009-09-05, 2010.

[16] Jakob Nielsen. Why ajax sucks (most of the time). http://www.usabilityviews.com/ajaxsucks.html, accessed 2009-09-14, 2010.

[17] Tim O'Reilly. What is web 2.0, design patterns and business models for the next generation of software. http://oreilly.com/web2/archive/what-is-web-20.html, accessed 2009-01-05, 2010.

[18] Linda Dailey Paulsson. Building rich internet applications with ajax. *Computer*, 10:14–17, 2005.

[19] Bruce Tognazzini. First principles of interaction design. http://www.asktog.com/basics/firstPrinciples.html, accessed 2009-09-07, 2010.

[20] W3Consortium. About w3c. http://www.w3.org/Consortium/, 2010.

[21] W3Consortium. Introduction to html 4. http://www.w3.org/TR/html401/intro/intro.html, accessed 2009-09-05, 2010.

[22] W3Consortium. What is the document object model? http://www.w3.org/TR/DOM-Level-2-Core/introduction.html, accessed 2009-09-05, 2010.

[23] W3Consortium. Xmlhttprequest, editor's draft 24 december 2009. http://dev.w3.org/2006/webapi/XMLHttpRequest/, accessed 2009-09-02, 2010.

[24] W3Schools. W3schools. http://www.w3schools.com/, accessed 2009-09-05, 2010.

[25] W3Schools. The xmlhttprequest. http://www.w3schools.com/ajax/ajax_browsers.asp, accessed 2009-09-05, 2010.

[26] Wikipedia. Adobe flash. http://en.wikipedia.org/wiki/Adobe_Flash, accessed 2009-09-11, 2010.

[27] Wikipedia. Javafx. http://en.wikipedia.org/wiki/JavaFX, accessed 2009-09-11, 2010.

[28] Wikipedia. Microsoft silverlight. http://en.wikipedia.org/wiki/Microsoft_Silverlight, accessed 2009-09-11, 2010.

[29] Nicholas C. Zakas. Make life easy with autocomplete textboxes. http://articles.sitepoint.com/article/life-autocomplete-textboxes, accessed 2009-09-13, 2010.

Appendix A

Requirements document

1. Elicitation of Requirements

1.1 Brainstorming
Brainstorming was used in the beginning of elicitation to get some ideas started.

1.2 Interviews with customer
The following interviews have been held with the owner of the restaurants.

Interview September 17, 2009: During the first interview I proposed the structure of the web site and what basic functions and contents that should be included in the web pages. The customer expressed that he liked the ideas and that he thought that it was OK to start the project.

Interview October 21, 2009: During the interview, the customer expressed the following thoughts and requirements:
- There shall be web sites for all three restaurants.
- Serious photography of all essential meals in the restaurants shall be done.
- An introduction text to the web pages, personally written by the owner, shall be displayed on the first page of the web sites.
- There shall be photos of the employees welcoming guests into the restaurants.

1.3 Looking at other web sites
Inspiration for functions and appearance have come from looking at other web sites.
www.fellini.se
www.tgifridays.com

2. Requirements Specification

R: There shall be web pages for all three restaurants Ye-Ga, Ye-Ga Chon, and Cape Town.

2.1 Feature requirements
R: There shall be a welcoming start page for each of the three restaurants, containing a welcome message written by the owner.

R: There shall be a page with information about the restaurant, for each of the three restaurants.

R: There shall be a page with the restaurant's menu, for each of the three restaurants.

R: There shall be a page with directions to the restaurant and pictures from the area surrounding the restaurant, for each of the three restaurants.

R: All relevant contents on the web page shall be available in both Korean and English.

R: There shall be photos of the restaurant and pictures of the restaurants and staff welcoming guests.

R: There shall be information about Korean food and dinner table manners.

R: There shall be photos of chosen meals and information about them.

~~R: The menu shall be printable through automatic PDF generation.~~

~~R: There shall be automatic appearance update according to season time of year.~~

2.2 Screens
The screens are the the same as the conceptual models in Chapter 5.

2.3 Quality Requirements
R: The web sites shall be built up on a modular foundation so that it will be possible to use the same foundation for all three restaurants' web sites.

R: The web sites shall be able to view from a guest interface and from an administrator interface.

R: All content data on the web site shall be possible to update by an administrator via a CMS in the administrator interface.

R: The CMS functions shall be implemented including Ajax technology, where useful.

R: All restaurant-specific data shall be stored in XML files.

R: All aesthetic layout formatting settings for the web page shall be read from a CSS file.

R: The change between languages shall be done by clicking a flag.

R: The web site shall work in browsers where Ajax functionality is available.

~~R: The web page shall be made available in search engines, tourist guides, etc.~~

2.4 Usability requirements
R: Usability principles and heuristics shall be in focus when when designing the guest inter-

face and the administrator interface for the web site.

R: The administrator interface shall be as similar to the guest interface as possible, in appearance.

R: The CMS functions in the administrator interface shall appear as a layer of functionality for updating the contents, on top of the guest interface.

~~R: Textual contents of the web site shall be kept to a minimum to keep maintenance and user input load low.~~

2.5 Security requirements

R: The CMS will be password protected.

R: There shall be guests, who see the guest interface, and an administrator, who see the CMS interface.

Appendix B

Time Plan

Before 22 September 2009
- *Project inception*: Initiate the development of the product. Stakeholders and goals are identified.
- *Elicitation*: Define development goals together with the stakeholders. Commence elicitation of requirements.

22 September
- Theoretical study

18 October
- *Elicitation*: Continue the elicitation of requirements
- *Writing the requirements*: Write down and finalize the requirements.
- *Checking and validation*: Check and validate the requirements.

26 October
- *Design and programming*: Develpment of the product, according to the requirements document.
- Continuous report writing

5 December
- *Acceptance test and delivery*: The product is installed and tested by the customer and developer to confirm that it satisfies the goals and works as described in the requirements.
- Move back to Sweden

11 January 2010
- Report writing
- Prepare for presentation

22 February
- Report done

After 22 February
- Have presentation and opposition