www.website4free.co.nz

Bachelor project IN3700

Version 2.2
June/July 2007

Project Group
Remy Alidarso 1221736
Zdravko Anguelov 1232436
Terence Bahlen 1217313
Frits de Nijs 1224263

Company
SecureSpace
Business Manager: Maria Veleva
Project Manager: Dimitar Velev

University
Delft, University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science

Commission
Project Manager: Ir. B.R. Sodoyer
Project Accompanier: J. de Vries
Preface

For the BSc Computer Science students are required to do an internship, for a minimum duration of twelve weeks, preferably with a company outside the University. We decided to try and do this internship abroad, in order to not only gain experience in developing an entire project, but also to be able to gain international experience. At a certain moment an opportunity presented itself to perform the internship (the Bachelorproject) in New Zealand. After the assignment received the OK, we went to New Zealand to start our project.

The project was developed at the company SecureSpace in Auckland, New Zealand. In short the project involved developing a system, which allowed people to subscribe to our service, where after, one of four kinds of websites would be generated for them. In the (almost) five months we stayed in New Zealand we worked for twelve weeks on the project and gained a lot of experience on how to develop a project like ours.

This document contains the result of our project in New Zealand. As said before we gained a lot of experience in developing projects and we would like to thank everybody at SecureSpace for giving us the opportunity. Not only to develop a Bachelorproject, but to develop it in a country at the other side of the world, this made the experience way better. We would also like to thank our accompaniers at the Technical University of Delft for allowing and supporting us to do the project in New Zealand.
Table of Contents

PREFACE ........................................................................................................................................... 2

1. INTRODUCTION ................................................................................................................................. 4

1.1. SECURESPACE (A COMPANY DRAFT) ......................................................................................... 4
1.2. THE PROJECT ................................................................................................................................. 4

2. PROJECT DESCRIPTION ..................................................................................................................... 5

2.1. MAIN GOAL ..................................................................................................................................... 5
2.2. REQUIREMENTS AND ANALYSIS ................................................................................................. 5
  2.2.1. Existing competitors................................................................................................................. 6
  2.2.2. Research ................................................................................................................................. 6
  2.2.3. Targeted customers .................................................................................................................. 6
2.3. USER TRAINING AND FUTURE DEVELOPMENT ............................................................................ 6

3. PLANNING .......................................................................................................................................... 8

3.1. GLOBAL PLANNING ........................................................................................................................ 8
3.2. REVISED PLANNING ..................................................................................................................... 9
3.3. HUMAN RESOURCE PLANNING .................................................................................................. 9
3.4. EVALUATION PLANNING .............................................................................................................. 10

4. TECHNICAL DOCUMENTATION ........................................................................................................ 11

5. IMPLEMENTATION ............................................................................................................................. 13

5.1. MODULES ....................................................................................................................................... 13
  5.1.1. Wizard ...................................................................................................................................... 13
  5.1.2. Article module .......................................................................................................................... 14
  5.1.3. Category Module ...................................................................................................................... 15
  5.1.4. Customer Module .................................................................................................................... 15
  5.1.5. Gallery Module ....................................................................................................................... 15
  5.1.6. Login Module .......................................................................................................................... 16
  5.1.7. Media Module .......................................................................................................................... 17
  5.1.8. Membership .............................................................................................................................. 17
  5.1.9. Menu Module ........................................................................................................................... 18
  5.1.10. Newsletter Module .................................................................................................................. 18
  5.1.11. Order Module ........................................................................................................................ 19
  5.1.12. Product Module ...................................................................................................................... 21
  5.1.13. Statistics Module ..................................................................................................................... 22
  5.1.14. Template Module ................................................................................................................... 22
  5.1.15. Shoppingcart Module ............................................................................................................. 23
5.2. LIBRARIES ..................................................................................................................................... 24
  5.2.1. Security VCL ............................................................................................................................. 24
  5.2.2. SQL _lib ................................................................................................................................... 24
  5.2.3. Main Variables ......................................................................................................................... 24
  5.2.4. Captcha ..................................................................................................................................... 25
5.3. GENERATION OF THE WEBSITE ................................................................................................. 25
5.4. MAIN WEBSITE AND ADMINISTRATION ................................................................................... 26

6. TESTING ............................................................................................................................................. 31

7. PROJECT PROGRESS ......................................................................................................................... 32

8. CONCLUSION AND RECOMMENDATIONS ....................................................................................... 38

APPENDIXES ....................................................................................................................................... 40
APPENDIX A: TECHNICAL DOCUMENTATION .................................................................................. 41
APPENDIX B: TEST REPORT .................................................................................................................. 79
APPENDIX C: STYLE A GENERATED SITE FOR DUMMIES .................................................................. 111
APPENDIX D: CLIENT MANUAL ............................................................................................................ 131
APPENDIX E: REQUIREMENTS DOCUMENT AND PLAN OF ACTION .................................................. 144
1. Introduction

This chapter provides a global introduction into our project. First we will discuss the company SecureSpace where we developed the project. Then we will give a short introduction on what the project includes. There we will also refer the reader to the most interesting parts of our report.

1.1. SecureSpace (a company draft)

SecureSpace is a company which now operates for five years, in Auckland, New Zealand. So far they have had more than 250 clients and they provide these clients with web design services. They specialize in the following functionality, in the e-commerce section:

- online shop,
- payment tracking systems,
- MYOB integration,
- and Business to Business.

This is only a small selection of the services they provide. While we were working at the company we gained a lot of insight in how to develop projects. We also assisted in the creation of a small project for a customer of SecureSpace. This was an eventful experience from which we learned a lot, not only how to develop a project, but also how to communicate with customers.

1.2. The project

During our internship we developed a system which allows users to register for a particular website and then that website would be created for them. In short the project involved gathering the requirements, doing research on various aspects and designing. After that the implementation began and the documentation and testing.

When we started one of the main requirements was to develop the project in modules, capable of operating with each other and also without one another. For more details on this aspect we refer the reader to chapter 4 and appendix A were all the major design and implementation issues are discussed. For a more indebt description on what the project includes and what it is capable of we refer the reader to chapter 5. Since we needed to gain experience on the entire process of a large project, testing was one of the key subjects we needed to develop. We did not incorporate all the test results in this report, but we included them in appendix B. For the readers who are interested in the process we followed, we refer the reader to chapter 7 and 8, here we explain what we have learned and how we see the project now we can look back at it.

In this report we will use the terms main website and generated website quite a lot. What we mean by main website is the URL www.website4free.co.nz, the site were users can register for the service. The generated site is the site which is generated for users who subscribed for the service; the URL can be something like blaat.website4free.co.nz.
2. Project Description

This chapter specifies the project in more detail. We will discuss the main goal and the future of the project; we will explain the user trainings we gave and the documents we made for future additions.

Since the requirements document and the plan of action already specified a lot of the details concerning the goal and the scope, this chapter will not go into much detail about these things. However in paragraph 2.1 and 2.2 we will give a short summary of the main topics of the plan of action and the requirements document, for more details see appendix E.

This project description is also made to specify what has changed from the original requirements. For example the main goal was to let people register for a website which they then can maintain, but during development the wizard tool became more and more specified for a complete business. So the generated website is not just an extension of their small business, but more a complete (separate) business. This is especially true for the e-commerce plus site, which has every available module that is relevant for an e-commerce website.

2.1. Main Goal

The main goal of this project is to provide a website generation and maintenance tool for small businesses that see the opportunities and benefits of having a well-working website, but who do not have the financial means (yet) to obtain a professionally custom-build website. This project is to provide those businesses with a tool that lets them build, and maintain a website that looks exactly like a professionally custom-build one, but at a far lower price.

As stated in the requirements document the scope of the project is determined by the number of modules implemented. At the end we managed to implement quite a lot of modules. Most of the modules are for an e-commerce plus site; we added a lot of modules to that particular type of website to distinguish us from competitors. Those competitors offer complete websites, but we offer a complete website plus a complete system to run your business (think of online payment, payment tracking system and statistics). To also attract customers who are not interested in an e-commerce site, the article and the media site were developed. The article site is more for people who just want a site to share information with other people. The media site can be used for the same purpose, but since it also has the capability to support memberships it can be used for a business. Letting people pay to see certain content is also a feature which distinguishes us from competitors. The final product of the project can satisfy a lot of people, for example students in New Zealand who want to share their experiences with others or business managers who want to set up an e-commerce business.

2.2. Requirements and analysis

This chapter is all about the requirements and analysis document. Because the requirements and analysis document is a separate document that can be found in the appendices (appendix E), there will be many references to it.
2.2.1. Existing competitors

Today, there are a few direct competitors, by that we mean businesses that offer approximately the same package as we do. An example competitor can be found on www.bigpixie.com. That website offers businesses a chance to build a nice website in a very fast way, but only the first 10 days are free, after that the user can choose to stop using the service, or decide to pay for the website and hosting. Because of the modular design we are able to constantly make new modules, either to raise the bar for our direct competitors by giving users new functionality to their website, or to give the same degree of functionality when competitors got some functionality we do not have yet.

2.2.2. Research

Research had to be done in order to make sure the project was going to be developed with a modular design. We got some own ideas, for instance to use the Model View Control design pattern, among some others. After some time we came across an article on the Internet explaining how to build a website in a modular way. The article opted to use the MVC design pattern so we tried to model our project using the MVC pattern and we saw that the MVC design pattern could be easily used in order to achieve our needs. The more and more we used the modular design, the more it became clear that we used the proper approach because the modular communication was very intuitive and easy to implement.

2.2.3. Targeted customers

This project targets people who are running a small business and businesses that do not yet have the financial resources to afford a professionally custom-build website.

For more details on aspects of requirements and analysis we refer the reader to appendix E.

2.3. User Training and future development

In order to get the project maintained even after we are done, we gave the maintenance team a few trainings. These trainings are very important, because creating software is one thing, maintaining it can be a whole other ballgame, especially when the maintainers did not create the piece of software themselves. Most members of the maintenance team do not share the same educational level as the development team does. So in order for them to maintain such a project with very high level implementation (abstract) decisions, we gave them a training and supplied them with enough documentation to be able to maintain the system. We explained the Model-View-Controller design pattern; how we implemented it, in order for the maintenance team to understand the general workings of our modular design.

Because of the supplied trainings and documentation, the maintenance team knows where to look for the specific code if there is an error or if something needs to be changed. Qua documentation we supplied a couple of ‘manuals’. There is a manual specifying how they can make a template for the website4free.co.nz project. This document explains how they can make a ‘framework’ of CSS around the content provided from the generated website (see appendix C). We supplied the API, generated from the comments in the code with DOxygen. The technical documentation is particularly handy if they want to know something about the modular design or the modular communication (see appendix A). There will be another
manual for end-users of this system, which consists of detailed descriptions how to use certain functionality, this is the same as the help function of the wizard (See appendix D).

Concerning the future of the project we can not say anything about the success of the project itself, however it will be very usable for later projects, because of its modular design. When new website orders come in, they can check which modules the customer needs, check which modules are already there, change them according to the users specifications, put them together and they have a working website. This assessment may be very optimistic, because there is always work to be done, because no client has the same requirements, but the start (and for some modules even more), is already there in the modules we provided.
3. Planning

This part of the document specifies the planning and the human resource planning of the project. In the beginning when we were finished gathering the requirements we did not have an idea how long certain parts were going to take, therefore we made a very global planning specifying when something needed to be finished in order to fit the entire project in twelve weeks. The global planning was specified as follows:

- Week 1: Training + documents
- Week 2: Training + documents
- Week 3: Generating admin modules
- Week 4: Generating admin modules
- Week 5: Generating admin modules
- Week 6: Generating static content for website4free.co.nz
- Week 7: Generating the layout for the generated websites (client side)
- Week 8: Generating the layout for the generated websites (client side)
- Week 9: Constructing the automatic website generator
- Week 10: Constructing the automatic website generator
- Week 11: Constructing the automatic website generator
- Week 12: Finishing up

As one can see this is a very global planning, however as we progressed with the project it became more clear how long certain parts were going to take, therefore we made a more accurate planning which can be seen in paragraph 3.1. In paragraph 3.2 one can see the adjusted planning which we made halfway the project, it is more detailed and was since that time kept up to date, so we had an overview how long everything was going to take. In paragraph 3.3 one can see the human resource planning, this is an overview for the development of the project and for the report on which person did what. In the last paragraph we made an evaluation of the planning and looked what we accomplished in time and what not and if we did, why this happened.

3.1. Global Planning
3.2. Revised planning

3.3. Human resource planning

For the human resource planning we made a distinction between the human resource planning for the project and for the report, as can be seen in figures below.
3.4. Evaluation planning

In the beginning the requirements analysis took longer than expected, because they were changing a lot. By the time we finished requirements document we managed to get an overview of what the company SecureSpace wanted for the project. After that we began to focus on the implementation of the modules, this also took a bit longer than expected, because this also required research especially on the modular design and the communication between the modules. However as the development of the modules came to an end, we started on the actual generation of the website and there we managed to gain some time, therefore in that stage we added a gallery module and an statistics module. We also made the main website and a tool to make the main website (fill in the content). For the main website we also made an extensive administration tool which has a lot of options to manage the generated websites. Then we focused on creating the appropriate documentation and also made two example templates to show that the generated website can be very professional.

In short concerning the evaluation around the sixth week we came to something of a standstill, however after that the pace went up and we managed to do what we wanted to do. We managed to finish in time, however since for some parts we did not have the things we needed, we could only finish them in the lasts weeks, which was not very convenient because it took until the last week to finish the project and this created a bit stress. However we did manage to do everything we sat out to do, just not in the form we wanted it.
4. Technical documentation

At first we incorporated the technical documentation of our project in the main report, however it became quite extensive and therefore we decided to put in an appendix, see appendix A. The appendix discusses all of the technical aspects of our project, mostly by use of sequence diagrams, but some with data flow diagrams. In the appendices the API is also included which some of the aspects of the technical documentation. Since SecureSpace values the SEO (search engine optimization) optimization of their websites we incorporated in this chapter an overview of the most significant aspects of SEO. We incorporated these aspects into the content editor of the main website4free.co.nz, (see chapter 5 for the content editor).

SEO (Search Engine Optimization)

For the main website4free.co.nz it is very important that this website is optimized for search engines, because this will be the main entry point for future customers and subscribers. If people are searching for a free website, website4free.co.nz should become one of the top results. In order for this to work, SecureSpace developed a Search Engine Optimization (SEO) manual; they also use this manual for the development of other sites. Following a list with the most important things we incorporated into this into the design of the editor for the main page.

One of the most important things for SEO are keywords. These are the words used by users to search for relevant information; therefore the keywords present in your page are relevant for SEO. Keywords can be present in the following things: the URL, the page title, the Meta tags, and the page content (all made available with content editor of the main page). To put keywords in the URL is good for search engine optimization; however it may not be the best solutions for visitors, because visitors remember short URL better, like imdb.com, instead of internetmoviedatabase.com. SecureSpace already chose the URL website4free.co.nz, which contains keywords, but not too many and it is easy to remember, because it is not too long. For the robots (Web Crawlers) who index pages for their search engine, content is very important, therefore keywords should be present in each page, however not too much, because then it is considered spamming. For a website to have a good SEO, it is therefore important that it has a rich content. For example it is a good thing to have a FAQ (Frequently Asked Questions) page which provides rich content for the website, with a lot of keywords. Other good examples are testimonials, a news section, forums, blogs, message boards.

Some HTML tags have more importance for SEO, the most important are <h1> ... <h6> and the <title> tag, the somewhat important are <b>, <i>, <img>, <meta> and <u>. Search engines do not index websites that contain only Macromedia Flash content and pages that are build with frames very well, therefore the use of them is discouraged. Another thing from which a page will benefit is a sitemap, which for some search engines you can submit yourself to their database. The benefits of doing so are obvious, the search engines' robot (WebCrawler) can crawl to more pages on the website, therefore indexing it in a better way.

Search engines also place great emphasis on links, the three main types are incoming links, outgoing links and links that link to pages on the same website. The more incoming links your site has from quality web sites the higher your site is likely to appear in search results. But the term 'quality web sites' is a bit vague, by that we mean (for Google) pages with a high PageRank or, more generally, pages that tend to appear high in search result lists. Outgoing links are also important, not only for your visitors (for relevant information), but it also helps
your site to become a part of a greater network. Therefore we incorporated the domain checker in the content of the website. The links to pages on the same website can be important as well, because the information stays relevant through the whole website. As a final step for SEO one can submit his own website to a search engine, this way your site is accessible for the public through search engines.
5. Implementation

This section of the document contains the implementation details for the project. Since the company wanted to make use of individual modules of the different parts of the system, we designed our project in a modular way in which a module is capable of working standalone and with other modules. In chapter 4 and appendix A there was already an explanation on how the modular approach is designed, therefore the focus in the first part of the report lies on the different modules and their functionality, what they can do and how they work.

The second section will discuss the libraries that have been developed. The next section will discuss the generation of the website, since this is a large script which makes the website, by making the database and by copying the folders, etc. We will also discuss the admin tools for the websiteforfree.co.nz itself, since this also works with PayPal, to track the payments and to take a site out of the air, when someone did not pay. There we will also discuss the editor for the content of the main website.

5.1. Modules

This section of the document will explain the functionality of the different modules. The following modules will be discussed:

- Article,
- Category,
- Customer,
- Gallery,
- Login,
- Media,
- Membership,
- Menu,
- Newsletter,
- Order,
- Product,
- Statistics,
- Template,
- and shopping cart.

Since the modules can be best viewed in the Wizard (administration part of the generated website), we will start with a short introduction of the Wizard.

5.1.1. Wizard

The Wizard is a sort of shell on top of all the individual modules functioning as interface for them. It is set up in a dynamic way in order to loosen up the restrictions of knowing up front what modules are going to be created. The wizard is created this way on purpose, because the generated websites will not consist of the same modules because each website type has its own modules. (However for the modules a customer does not have, there will be a html file, explaining what would be there, as sort of commercial to make the customer want more). It will load a module automatically when the main file of the module is copied into the wizard directory. Now the user has the power to use and adjust the module that was just added.
5.1.2 Article module

The article module started out as a module in which clients could edit articles and publish them on their website. However as the article module developed further and further it became more of a page module in which clients could make and edit entire pages. The article module is divided into two main parts, namely the dynamic pages and the static pages.

Static pages
The static pages consist of five general pages, with specific content. These are home, about us, contact us, a frequently asked questions page and a gallery (which has its own module). For the homepage we made use of the WYSIWYG editor, because then the user has more options to edit the view of his homepage. The about us page is a very simple page, which states what kind company the client has and what his company does. The frequently asked questions page has a separate database table, because we wanted to be able to let the client add/remove the questions in a user friendly way. We did not see any other way than doing this with a separate database table.

If a client does not want a certain static page it can set its status to not published, he can however not delete them, because he might want to add them later on again. The home page, will always be there because it is an essential part of a website. The static pages are kept very simple because, the idea is that the client makes his own sites, rather then using predefined sites, therefore the client can choose to make his own, about us and contact us and even an FAQ page (however this page will not be as dynamic as the pre-generated ones). This way the client can determine the look of his own page in the way he wants it.

Dynamic pages
The dynamic pages are the real content of the generated website. Since we do not want user to make a bad website, they can not edit the template, they can only fill in the main content by means of the page editor. The page editor consists of a page picture, a short description of the page and the general content of the page, in which they can add text and pictures. The pictures first need to be uploaded, which they also can do in the same screen. This way the user has everything he needs to make a complete page.

The article module has a close cooperation with the menu module, this is because once a page is generated it needs to be added to a menu entry, because otherwise it is not visible on the generated website. The user can select from a drop down box, under which menu item he wants to add this page. Since the menu module is loaded through the ModuleLoader, it can be the case that the menu module is not there (this scenario should not be possible, because the article module and the menu module are always installed, with every type of website someone chooses), then it is still possible for the client to make a page, however it is not visible on the homepage because there is no link to it.

When a client makes a page he makes use of a what-you-see-is-what-you-get editor (WYSIWYG editor) to fill the contents of the page. This WYSIWYG editor was not our first choice as editor, however because the company already made use of this editor we also incorporated into the website4free project. Our first choice was tinyFCK, because this one is open source and has a lot of features and works great under all browsers. SecureSpace uses the Innova editor, however there are problems with this editor under certain browsers and it is not open source. This has the advantage that you can get the manufacturers support, however its use is not very user-friendly. Incorporating the editor in the page editor was not a problem,
however since we do not want to give to much functionality to the client, because then they can make bad website design, we disabled some options. This way a client does not need to understand HTML tags, which makes the page editor understandable for everybody and easy to use.

5.1.3. Category Module

The category module is used by both the Media site type, as well as the two e-commerce site types. A product (physical or non-physical) has to be linked to a category. Categories can technically be infinite levels deep, but it is restricted to two levels for this project. This, combined with the menu module gives us a highly adjustable menu building tool that is very easy to use. For instance, users who choose an e-commerce site are able to add categories for their products. When they visit their website, they automatically have a special ‘shop’ menu consisting of all the categories. When users move their mouse over a particular category that has child categories, then the menu will fold open, showing the definitive subcategories that hold the actual products.

5.1.4. Customer Module

Two of the site types – 'media' and 'e-commerce+' provide the facility of keeping track of the registered users of the site as customers. That enables the users to use the services that those site types provide and it enables the site owner to keep track of and manage his customers. All that is possible through the Customer Module.

The customer module allows users to register on the site and become customers. In the registration process the users' data like e-mail and address are validated. The validation includes a check whether the data is potentially incorrect and whether it is potentially harmful. This is done so that only safe and useful data is saved to the system. Once they have registered, customers can also edit most of their details. They cannot change their username and a couple of other administrator related data such last login date, login count for the last 12 hours etc. That data can only be changed by an administrator as he has the rights to view and edit customers data and remove customers altogether through the wizard.

The module also offers to the administrator the possibility of searching through the customers of the site. Through the extensive search customers can be searched by id, name, username, city, country, e-mail and whether they are active or not.

All users of the application can currently be of either of two kinds – administrator or customer. To distinguish between those two an access level scheme is introduced. The access levels are what the system's authorization policy is based on. Authentication on the other hand is done in a separate module – the Login Module. In the current version only two access levels are being used, one for the administrators and one for the customers. But the design is extensible and any number of access levels can be added. The access level information is kept in a separate table that is linked to the rest of the users' information. The customer data as the data for all other user types is stored in a separate table. That data is directly managed by the CustomerModel class.

5.1.5 Gallery Module

The Gallery Module allows the administrator of a generated website to add photo or image galleries to his website. This is particularly interesting for the owner of an article site, looking
to add a large amount of photos to a story, or for the owner of a media site who wants to
display screenshots of the media available on his site. However, the gallery module is
available to all websites instantly, because just like the article module the gallery module is
considered to be necessary for basic website content.

For the customers or visitors of a generated website, the gallery module is responsible for
displaying an overview of all the categories that are a part of the gallery and have photos in
them. Each category will have a small amount of thumbnail images on display, to give the
visitor an idea of the content inside. When visitors click on one of the category names, the
gallery module will display a full list of thumbnails of images inside that category, broken up
into pages. When the visitor then clicks one of the thumbnails to view the actual image, a
slideshow is started by the gallery module, with that image as the first image. The slideshow
can be paused, and the visitor can use the previous and next buttons to manually seek for an
image. Finally, the visitor can click the current image in the slideshow to display the actual
image itself.

Administration part of the Gallery Module
The Gallery Module has administrative functions for the creation of categories, the uploading
of images, as well as for the removal of the images and categories. A category is represented
as a directory internally. The images uploaded to a directory are considered a part of that
category. The gallery module is responsible for the automatic creation of thumbnails for every
uploaded image.

The first element on the gallery page of the wizard shows a category selection form, which is
created using the gallery module. This form allows the administrator to select a category, or
add a new category. As soon as a category is selected or created, the page will also show an
image upload form and a preview of all the images inside that category. The administrator can
then use the upload form to upload multiple images at once, or use the preview to view and
delete images from the category. The preview also has a button to delete the entire category at
once.

5.1.6 Login Module

Some site's content is restricted to only particular types of users. The wizard for example is
only available to the administrators and the customers details are only available to the
particular customer himself and the administrator. As already mentioned in the Customer
Module section the system uses access levels to implement its authorization policy and
distinguish between these two kinds of users. But besides that there is also a need for an
authentication policy which will distinguish between valid users of the system and non-
authorized users. This Login Module implements the authorization on the basis of logging in
and is intended to take care of all steps of the authorization process.

The login module enables users to authenticate themselves by logging in with a username and
password. Once logged, in users can view all pages they are authorized to view. After 30
minutes of inactivity the user gets logged out automatically to make sure that there will be no
misuse of his account in case he has forgotten to logout. The users can also choose to logout
themselves when leaving. The login module also makes it possible to secure all scripts from
unauthorized execution. That way even if ill intentioned users guess the URL to an existing
script they won't be able to execute it unless they have already logged in with a valid
username and password and they are authorized to do so.
On login the supplied username and password are matched against the existing users in the database. If there is a match the Login Module uses both the access level and the user tables to retrieve the complete user information and make it available to any other module through its controller methods.

The controller uses the session control library to store a key value generated from the username and password as a hash to identify the user in further page requests. The way this key is stored is left to the library and can be either as a cookie or in the session depending on the client’s browser settings. This key is also available on the server side in the database. At the same time the php session id of the requesting user is saved to the database.

Then every time the client requests a page the second part of the authentication is done – the verification that the user has logged in as a valid user of the system. To do that the key together with the saved php session id are matched as a key and lock. The current php session id must match the session id saved to database at login and the key supplied by the client must match the key available in the database. If those checks succeed then the user is successfully authenticated. After this a user authorization is done for the particular page the user has requested.

5.1.7. Media Module

The media module is a module that users will only receive when they have chosen for the media site type. It puts people in control of uploading and showing certain media files they like. Users can upload any type of media they would like to show on their website, and the generated system is able to convert any media type to one specific media type, flash video. The Flash Video format is chosen because the necessary plug-in to show flash videos is installed on 95% of the systems that are connected to the Internet. It can be compared to Youtube, but it is not that extensive. It will not support tagging of videos for instance and it will also not have the community aspect that Youtube has. The software that converts the media files also generates a second, smaller flash video, to be used as a demo to show. Demo videos are necessary, because users who make use of the media site type are also able to accept subscriptions. This way we give artists a chance to easily setup their own media portfolio, and let them be able to accept subscriptions from people who really like their work.

However since only in the last week on the last day, the libraries for the media conversion were properly installed we did not manage to implement all the functionality of the media module. The code was there, but we did not have time to test and finish it.

5.1.8. Membership

The media site type gives users the opportunity to become a member and then to watch membership only content. To make this possible a separate membership’s module was developed. The reason that the media module and the membership module are developed separately is that although they are relatively tightly coupled they still can be deployed separately. This is of great importance as membership management and video streaming are rather common and with this separation can easily be added to any other website on their own.

The membership module allows customers to become a member of the site so that they view membership only content. They can choose from a membership for 1,2,3,6 or 12 months. In the wizard the administrator can view a list of the memberships with the following information about them: start date, expiry date, membership holder, and whether the
Every membership that is created is stored in the 'wff_membership' table. After the membership has been paid for it is set to active and can be used to view membership only content. After the expiry date of the membership has passed the membership can not be used anymore and a new one must be purchased. The expired memberships are kept in the database so that a statistical overview of them can be shown to the administrator.

5.1.9. Menu Module

The menu module is an important module because this module will be one of the backbones for the websites that are to be created. Users are able to change the order of menu items, and of course they can also add, edit and delete any existing menu items. Having that said, we need to make a side note here, because the users will not be able to delete any static (required) pages such as the homepage itself. Menu items can be nested two levels deep. So each top level menu item can have children. More on this subject can be found in the Category Module. When clicked on a top level menu item, all the available (published) articles inside a sub menu in that particular menu entry will display as small articles, with a 'read more' link to be linked to the full article. The article module is set up in a way that articles can be put under any available sub menu entry. When clicked on the 'read more' link, the full article is shown.

5.1.10. Newsletter Module

The newsletter module allows clients to send email to their clients. We divided this process into 3 steps, so it is easy for the user to send out a campaign. The first step consists of uploading the email addresses of the customers of the client, this can be done through a CSV file upload, or it can be done by hand. The uploaded email addresses can be put into groups so that it is easy for the user to maintain his database. The second step consists of making the email to be sent. A special feature in this screen is the ability to add products to the email, as an extra campaign feature. If the client does not have an e-commerce website, this feature is disabled since the ModuleLoader can not load the product module. When making a campaign the client has to fill in a HTML content and a text-only content (for the customer who use email clients that do not support HTML content). In this second step the client also has the ability to view how his email looks at the moment by pressing preview. The final step in sending a newsletter is sending the email too a certain group of customers. This final step consist of two small steps, the client first has to send out a test email, to a certain email address, before they can actually send out the campaign. This is done, because then the client knows for sure what kind of email he sends out and how it looks when customers receive it. Once the test email is sent, the client can send the campaign to all the other customers. If for some reason the client cancels the process halfway through sending the emails, this is saved and next time he can just continue with sending the emails from the point where he was. However when he sends a new campaign, this information is lost.
If someone registers on the generated website and they tick the 'subscribe for newsletter' box, than they will be added to a specific group in the newsletter module, this way the newsletter database is populated, so the client does not have to do that himself, if he does not want to.

A special feature of the newsletter module is that if a client sends out an email with pictures, the pictures get send as attachment, rather than as a link. The advantage with this is that customers who use an email client that blocks pictures do not have to press the button ‘display pictures’ in their email client. The disadvantage of this is that the email can become quite large depending in the number of pictures in the email.

Since someone who registered for a website4free.co.nz can also choose a newsletter only site, the user can edit its recipient’s database by hand. This is a simple feature in which he sees all his newsletter customers and he can edit them according to his own specifications.

5.1.11 Order Module

The Order module is responsible for converting a customers shopping cart into an order, as well as handling integration with bank gateways. The supported gateway is Paypal. Paypal was chosen because of a few benefits:
Paypal allows customers to pay via credit card, unlike most New Zealand bank gateways.
Paypal offers free to the public documentation and support about the integration.
These benefits make integration with Paypal cover a large amount of potential customers while being easy to implement.

The first step in the order process is the ‘new order’ page. On the new order page, the order module is responsible for generating an immutable shopping cart overview, to inform the customer of his intended purchase. This overview has to display the final order items, because it will become the actual order, and customers should not be presented any surprises after paying (like items being unavailable or out-of-stock). Nor should customers be able to alter the price or amounts of items, to protect against the purchase of free products. To meet these requirements, the order module will retrieve the latest Product data of the items to purchase, and store the final shopping cart in the server side ‘session’. Also on the ‘new order’ page is a form where customers can enter their personal details, such as their contact details and delivery address. If the ‘customer module’ is available, the form allows for retrieving this data from their user accounts, speeding up the ordering process.

The second step in the order process is the ‘order overview’ page. The order overview shows the finalized order as it is stored in the database. This gives the customer a final chance to review his/her order, and possibly go back to alter any mistakes. To facilitate the correcting of mistakes, there is a back button available, linking the user back to the ‘new order’ page.

The transition between the ‘new order’ and ‘order overview’ page also provides an important event for the order module. Since the payment of orders is not processed by our system, the customer will have to be directed to another server. During this period, the order module needs to store the data in some form of persistent storage, since the session could be lost. The order is therefore stored in the database as an ‘unconfirmed’ order before displaying the ‘order overview’. This also has an advantage for the owner of the website, because he can observe the order and potentially contact the customer that placed the order, even if the order did not go through. This provides the owner with a powerful feedback mechanism.
After the ‘order overview’ page, the customer is directed through the payment gateway. There, the customer has to enter his payment details as well as confirm the payment of his/her order. This step can only be controlled in a limited way and cannot be viewed by the order module. However, it is a vital step in the order process. The order module is responsible for setting up certain ‘post’ variables, in order to integrate with the bank gateway. These include variables for the order itself (price, items and customer details), variables for identification (unique hash), and variables for the payment process (company logo, company name and gateway optional functionality).

The final step the customer will see is the ‘thank you’ page and confirmation e-mail. After the payment is processed by the bank gateway, the gateway will link the customer back to the generated website. Because there is no guarantee the customer will wait on the redirection, the only thing the order module can do is display a standard ‘thank you for placing your order’ page. From this page, returning customers can proceed browsing the generated website. Customers will also receive an order confirmation e-mail, to ensure every order is confirmed and to provide the customer with a ‘customer side’ confirmation of payment receipt.

The trigger for the confirmation e-mail is the bank server response to the website server, which is hidden from the customer and uninterruptible by the customer. This response contains the transaction details, as well as the identification variables passed through the gateway. These variables allow the order module to select the correct order from database. It is the order modules responsibility to check and confirm the correctness of the identification variables. The order module must also contact the bank server with the received variables, to confirm the bank servers own identity. This is required to make spoofing the bank response more difficult. After the identity checks, the order is updated with the provided transaction information. Depending on this information, the order is set to ‘confirmed’ or ‘cancelled’. If the order is confirmed, stock is also deduced from the ordered products. If the stock can not be reduced, for instance because the product stock has been reduced by another order, the state of the order is instead set to ‘out-of-stock’.

**Administration part of the Order Module**

The Order module also has an administration back-end. This administration area allows the owner of a generated website to view the placed orders, and deliver the ordered goods to the customer. Some parts of the administration interface and interface ideas were borrowed and developed from the SecureSpace in-house order administration system ‘Paytrack’.

Initially, the administrator is able to see an overview of all the orders that are stored in the systems database. Only the most general information about the order is displayed in the overview, such as date, total price and order status. The order module is required to display the list of available orders, as well as provide the functionality to efficiently browse the list. This includes the ability to sort the orders, and search for orders based on a number of criteria. These criteria include order type, order reference number and order date. From the list overview, the status of the order can be changed (but the possible resulting states depend on the current state of the order). A confirmed order can be turned into a delivered order, after which the customer receives an e-mail confirming the delivery of his order. Orders that are out-of-stock can be cancelled or confirmed (but only after sufficient stock is available for all the items in that order). And orders that are delivered or cancelled can be archived to make them not show up in the list of orders by default. Archived orders can still be viewed by specifying ‘show archived orders’ in the order search.
When the administrator clicks on one of the orders, a popup is displayed with all the details of that specific order. The order module has to retrieve the order from database and show it in a printable format. This allows the administrator to generate an invoice, as well as view the contact details of customers that did not complete their orders, possibly helping them to complete the order, or gaining information about the reason for not completing. On the order popup, the order module will also allow the administrator to quickly jump to another order by entering that orders unique ID (which is the second part of the order reference number).

5.1.12. Product Module

The product module is responsible for managing a web shops physical products. The user can search for products, and view the individual products that have been found.

On every page there is a product search box that allows the customer to search a product based on keywords. This form also has a link to the ‘advanced search’ page. On the advanced search page the customer can also search for products based on keywords, but the advanced search allows the user to refine the search by specifying a specific category, or limiting the price of the product.

Using either search form, the user is taken to the product list page. On the product list page, the product module is responsible for retrieving the products that match the given keywords and criteria. It is also responsible for dividing the list of products into subsets that can be displayed on one page. The amount of products per subset is adjustable. The list displays only the basic details of the products, such as name, price and image thumbnail. If the shopping cart module is available, a link for ‘add to shopping cart’ is put next to each product that is available and in stock.

When the customer clicks on one of the products, the ‘single product’ page is displayed. For this page, the product module has to retrieve all the details on a single product. This is then displayed to the user, again with a link for ‘add to shopping cart’ if a shopping cart is available. The single product page displays product name, price as well as product description and larger image.

Administration part of the Product Module

The product module also has administrative functions for adding and editing products. When the administrator visits the product area, the ‘product administrative list’ is displayed. This list shows less detail than that for the customers, but allows the administrator to edit selected properties of each individual product quickly. Properties that can be edited are stock, minimum stock, selling price and buying price. When changes are made, the administrator has to submit the changes for them by clicking on the make changes button to make them effective. The product module then has to extract the changes for each product and make them in the database. The products displayed in the list can be changed by entering search criteria in the search form that is also displayed by the product module. The products can also be ordered on the displayed properties.

When the administrator clicks on a single product in the list, the ‘edit single product’ popup is shown. This popup allows the administrator to alter every single property of a product, for instance adding to a description or uploading a different image. The product module is
responsible for loading and displaying the edit product form, as well as for populating the fields. The product module also allows quickly switching to another product to edit by entering its ID.

5.1.13 Statistics Module

An extra feature available to all the clients that subscribe for the e-commerce+ site are statistics about the sales made on their site. This functionality is implemented as a separate module - the statistics module. This module gives overviews of the product sales as three different types of charts. These chart types will be described in the next section. The second section will describe the way this module is structured and the most important aspects of the implementation.

The Statistics Module gives the following graphical overviews:

- an overview of the number of sold products for a month where either all the products in a specified category or all the products from all categories appear. This month overview is displayed as two bar charts where in chart number one the products are ordered by product name and in chart number two the products are ordered by sold quantity in descending order. The user can choose to go to the next or the previous month for which he will see the same type of overview. From this month overview the user can also select a particular product to view more detailed information as described in the second point;
- a sales overview for a particular product for a month broken down in weeks. Here the user can also choose to go to the next or previous month;
- a profit overview based on the buying and selling price of all the sold products. The difference between these two is used to build an accumulating sum to represent the profit overview. This overview is displayed as a graph with a data point per day for the period from the first to the last available sales record.

Originally it was suggested to use the already existing order data for this module. But it was decided that this module should use a separate table so that the system can record purchase information only when the module is available. Whereas purchase information always has to be recorded in the order table as it has to contain a complete record of the made purchases. Thus when statistical information is retrieved it can be only retrieved about the periods when the module was active and paid for. In this way it's prevented that clients pay for the module for a short period of time and long after it has expired pay for it again and are able to see statistical information about the whole period since the first purchase.

A generic graphical library was developed separately from the module. It uses the PHP image manipulation GD Library to generate charts. On top of that a PHP script was written that uses the graphical library and the statistical data retrieved from the Controller to return the desired statistical chart. The behaviour of the script can be influenced by passing GET parameters specifying the appearance of the generated chart image.

5.1.14. Template Module

A vital part of the website4free is the ability of the clients to easily change the look-and-feel of their website without loosing their content. This feature will enable them to easily adjust the design of their site without having to spend a lot of money and also to make website4free a more appealing option compared to the competitors. It will also allow SecureSpace to cut the costs for support as this process is automated.
To achieve this a template module was developed which uses the concept of templates. Clients can change the look-and-feel of their site by changing the template. This can be done from the wizard after they have logged in. There they can view a preview of all the templates offered and select one if they like it. The change takes place instantaneously and none of the already existing website content is lost. There is also a user friendly error handling that reports if the change can not be completed for whatever reason (template not available any more etc.).

If a client has its own template or it has a purchased one from a template supplier (e.g. templatemonster.co.nz), then they can contact SecureSpace. SecureSpace will then make the template available for the client, they will 'mold' the template in the shape required to be incorporated in the website4free.co.nz. They can do this with the manual provided to make a template. SecureSpace also accepts requests for customizations of the client's template. After a clients template has been customized, it can not be switched any more using the template module tool (in order to prevent the loss of the done work). The client must contact SecureSpace staff first if any further changes are required.

A template is a collection of a main php file, one or more css files and a couple of images. All of those are kept in a separate directory the path to which is saved in the main DB Schema along with some extra information as a preview image, the name of the template and whether the template is available to clients (see Appendix C). The reason this information is kept in the main DB schema is that this way it is centralized and accessible to all clients.

The collection of files that make up the template define the layout and the positioning of the different site elements as menu, shopping cart, login form etc. on the screen. This is done by only calling the appropriate controller functions of the different modules in the main php file which further contains only html mark-up. The controller returns the structure of the requested element as html which is then added to the template at runtime. If the particular module is not available that part is skipped and the rendering continues with the next element. In this way the functionality that the modules offer is embedded in the layout. And a ready template that defines a site's layout only needs to be extended with the calls to the right modules for the rendering of the desired elements.

This structure also makes it very easy to switch between templates without affecting the working of the modules as layout and functionality are loosely coupled. In fact on template change the existing template (php file, css etc) is overwritten with the components of the new template and the change is done. As the new template calls the same module functions to retrieve the site-specific content as the old template there will be no difference to the user except for the layout.

5.1.15. Shoppingcart Module

The Shopping cart module is only available to the e-commerce+ websites. It allows the customers of the generated website to order multiple products at once, by putting them in the shopping cart, and then ordering the entire cart at once. Because the shopping cart is only used by the customers to temporarily store products, there is no administration required for the Shopping cart Module.

On websites with the shopping cart module a shopping cart element is added to the website. This element shows the number of products in the cart and their total price at a glance. It allows the customer to access the shopping cart overview on any page, by clicking on the element. On the product views there is also a link to the shopping cart, which leads to the same overview. Clicking on the link in the product view also adds one item of that product to the shopping cart.
The actual shopping cart overview opens in a popup, unless the customer has disabled JavaScript, in which case the overview is opened in the central content area. On the overview the customer is allowed to enter a quantity for each product in the shopping cart. A quantity of 0 removes the product from the cart entirely. There is also a button to immediately clear the cart. When the customer is done shopping, he/she can click on the proceed to order button, at which point the order module takes the shopping cart data and turns it into an order.

5.2. Libraries

This Part of the document discusses some of the libraries we used in the development of the project. Emphasis lies on the libraries which are also used by the company SecureSpace in the development of their projects. We used some of their ideas and because of that we discuss the libraries they also use, so they know how we implemented them.

5.2.1. Security VCL

The class Security is a very important one, when it comes to the security of a website. The main functionality of this class is to make sure the $_POST and $_GET variables 'safe'. This is made for prevention of hack attempts of passing GET variables through the URL. The Security class has three modes, VCL_STRICT, VCL_NORMAL, VCL_SQL. VCL_STRICT only allows for a-z and 0-9 characters. VCL_NORMAL allows more characters, for example those that are necessary for an email address. VCL_SQL makes the parameters safe for insertion in the database. Using this functionality of the security class it is guaranteed that hacking the website by means of fooling around with GET variables through the URL is made impossible. VCL also makes sure that for the input fields, there are no html tags allowed because it strips those from the GET and POST variables.

5.2.2. SQL_lib

Since there would be extensive use of the database during the project, we made a library named SQL lib, which is responsible for retrieving and storing data from the database. To reach consistency this library always returns the same object, namely a Result object. This has the following parameters, the number of items returned, the query and the data (which consists of the number of items). This object is used when a SELECT query is used, for the INSERT, UPDATE or DELETE queries another object is returned, namely the ConnId object. This object has the following parameters: rows (how many rows were affected), last_insert_id (the last inserted id), state (the state of the connection), id (the id of the connection) and pID (the parent ID of the object).

The library has two functions, 'from_sql_obj' and 'sql', used for the retrieval of information (return the Result object) and the storage of data (returns the ConnId object), respectively. The library also contains a function 'err', which displays errors, but these errors are only displayed when the database is being accessed from the intranet of the company, therefore a visitor to the website will not see this debug information.

5.2.3. Main Variables

The main variables are stored in a structure we incorporated from the programming manual from SecureSpace. The main_vars.php, as the name indicates, contains the main variables (defines, constants) which are used throughout the project. It contains the following variables:

- M_DB, the database name to be used
• M_HOST, the database host (usually local host)
• M_USER, the user who can access the database
• M_PASS, the password to access the database
• These four are always used by SecureSpace and we added a couple of extra specific ones for our project, these are:
  • DOMAIN, this is used to determine the current domain name of the website. This is necessary because for functions like file uploads we need to know the exact directory tree, which has the domain name in it. For example the newsletter module requires this, to make an unsubscribe link: http://DOMAIN.websiteforfree.co.nz.
  • ROOT, this variable can be seen as the same as the BASE variable, but then from within the file system. This is done because for some functions we need to have the exact path (absolute path) to a certain file. This is used for instance in some upload functions, convert functions etc.

Because each generated website uses a different database, database user, and obviously a different password, and because the DOMAIN variable is different for every generated website, we have to generate the main variables dynamically. This is done at the registration process, just before customers are redirected to PayPal to pay for the hosting of the website. The excessive generation of sub domains and databases and database users before the sub domain is actually active (been paid for) isn't a problem, because those domains (and every reference to them, e.g. Database, database user, directory tree etc. will get deleted automatically when the domain hasn't been activated after 1 month. So there will be excessive domains on the server, which are not being used.

5.2.4. Captcha

The register page of the websiteforfree.co.nz main and the contact us page of the generated website have a security with captcha. Captcha stands for Completely Automated Public Turing test to tell Computers and Humans Apart, which simply means that an automated process can not fill in the contact us page, again and again, because there is human input needed. With these two pages this is done, with a scrambled picture, which humans can read, but computers can not. The picture consist of stripes with numbers, because of the stripes the computer cannot read it, but a human intellect can. The generation of the picture consists of two steps, namely the generation of a random number, which is trivial and from this random number a picture is created. The user has to fill in the numbers in the picture and if they are not correct he has to do it again, but then with a different random number.

5.3. Generation of the website

After we made the initial modules, we began to focus our attention on the main website. Again, focusing on reusing our work in the future, we didn’t want to build the website in a regular way, that is, to style each and every text area separately. We therefore build a special construction using a database, PHP, CSS and (X)-HTML that is capable of customizing any text area in any way we like.

With the main website being created, we now needed some basic functionality such as a registration to the service. The registration process isn’t the same as any other regular registration tool. Because our registration process uses Paypal as payment method, we had to figure out how to communicate with Paypal. Because our registration process should be able to support multiple payment methods, not only our generated websites should be dynamic, the registration process had to be dynamic too. When a subscriber likes to pay using Paypal, a special button is created on the last page of subscribing, linking directly to Paypal so users
can setup their recurring payments. Because there is no way to identify a single Paypal payment, we need to add a (possible) future subscriber to our database before the actual Paypal payment has occurred (More on this later). It is because of this exact same reason we already have to create the directory structure and a database for the possible subscriber, and setup a database user and grant certain privileges to it.

5.4. Main website and Administration

It is because of the reason mentioned at the end of section 5.3, that we came up with a special admin tool. It is built with the intention of making an administrators' job as easy as possible. The admin tool is capable of automatically deleting every reference to domains which have been created at the registration process, but which have not been confirmed yet by an initial payment. Those domains get deleted after a month when no initial payment has been received. This happens for example when a user walks through the registration process, but at the last moment (Paypal) he or she lost interest in the service. The admin tool also covers an automatic process that checks if the domain has expired (no payment received), so it will remove the link to the website. The website is now not entirely removed, because people can still make a payment for the domain, so an administrator can activate the domain again. If for some reason a site should be taken offline (eg. On request by authorities), an administrator can disable a domain with just one click on a button. Because the sources of the website are constantly being developed (enhanced, updated), the admin tool is also capable of copying the most recent source to a destination domain. Because subscribers may like to have custom-build functionalities in their website, the 'copy newest sources' function can be disabled when a site has been customized. This way the system prevents administrators from accidentally overwriting customized site sources.

The system can also be used to change user details, or domain details. The admin tool is also capable of generating a payment history overview per domain. Inserting payments can also be done manually, in case a user made a bank deposit and calls in to check if it has been received. When it is checked and it has been confirmed, an administrator can insert a payment and set the new expiry date accordingly to the payment. The admin tools can also be used to check for special statistics, for instance what the partitioning of the chosen website type is, or how much data on disk the domains are using, see figure 1.

![Figure 1: admin tool overview](image)

The ‘main website’ is the starting point for anyone interested in registering for a free website. It consists mostly of static content, with the intention of informing potential clients about the
system, persuading potential clients to take a free website and boosting search engine ranking. The main website also contains the registration script explained elsewhere.

**Design**

Although the main website is mostly static content, it was not created as static HTML. Instead, the entire website is generated from database to keep the content flexible and reusable. The design of the main website is an extension of a design used by SecureSpace, based on the idea of ‘constructors’, basic HTML patterns that can be combined to create an entire website. The website is generated using three database tables: Constructor, Page and PageElement.

**Constructor**

The constructor table contains the HTML patterns that are used on almost every page. Every constructor has an unique name, identifying the constructors appearance and most common usage. The constructor also has an HTML field, which contains the actual HTML needed to display the constructor. The HTML for a constructor also contains special tags that allow the constructor to contain nested elements, or children. An example of a very basic constructor is:

```html
<table class="grey_background">
  <tr>
    <td>
      {#1#}
    </td>
  </tr>
  <tr>
    <td>
      {#2#}
    </td>
  </tr>
</table>
```

This constructor would render two child elements positioned on top of each other in a table with a grey background (depending on the css definition of class ‘grey_background’), placing the first child above the second.

**Page**

The page table contains the general information about a single page. This information includes the pages' title, keywords and starting PageElement. Whenever a page of the main website is requested a page id is provided. This id is the unique key of the Page table, allowing the script to retrieve the requested page. First the html document type and head meta data tags are generated using data from the Page row, and then the body is generated using the PageElements tree, starting at the PageElement provided in the Page row.

**PageElement**

Every page of the main website is built using PageElements. A PageElement is identified by its unique key id. A PageElement can be one of four different types. The first type is ‘constructor’. PageElement rows that are of the type constructor contain the name of that constructor (linking the element to the constructor), and a list of child PageElements. The constructor is the only type of PageElement that allows nesting. All other types are therefore leafs of the page tree.
The second type is ‘text’. Text elements only contain text. The text can be marked up using HTML tags. An example of the text in a text element could therefore be: ‘<h1>Header</h1><p>content</p>’. The text elements contain the actual content of the website.

The third type is ‘binary’. This type of element represents an image. This type of PageElement is used to display images that are relevant to the content, such as demonstrative graphs or screenshots. The binary type is not intended for background images or decoration, since these should be a part of the constructor. Using the text property it is possible to attach alternative text to the image, as well as including the image in a link to another page.

The fourth and final type is ‘php include’. The php include type allows the page to perform dynamic actions, such as required by the registration script. PHP includes will be added on the specific location on the page. Rendering of the page will stop until the script has finished running. It is possible for the included script to render HTML, which will be inserted on the exact location the element was added.

Editor
Originally it was intended for the website content to be delivered to us by the company. This was because the focus of the development team was the development of the system, and not the writing of large amounts of content. However, because the website content was not delivered on time a page editor was developed. The page editor was intended to be usable by SecureSpace employees even after the development team had left. This meant the writing of the content could be delayed as much as needed.

The editor consists of two separate pages. The first page is the overview page. On the overview page the current page being edited is displayed on the right. On the left is the editor bar that displays the current page tree expanded to describe the type of element. There are also a few links to pages that allow the user of the editor to add new pages, edit existing page meta data and remove entire pages.

The second editor page is a pop-up that details a single PageElement. The popup shows a selection for the type of element, and input fields for the other element properties. It is also possible to jump to the parent or any of the child PageElements by clicking on the link.
The element editor also allows for another important feature, reuse of existing PageElements. By selecting an existing PageElement from the list, it can be linked to an empty constructor position. This way, by changing just one PageElement, multiple pages can be updated.

Example
This example of the use of the editor was written to demonstrate the concept of constructors and page elements in action. To start the example we have a page with only one constructor, that renders a light grey border around the contained elements.

In the empty slot we add a new constructor. This constructor renders a red header and two cells next to each other, with the left one intended to be used as an image.

Now we fill in the empty spots with text and binary elements. It is possible to go much deeper by nesting even more constructors, as has been done on the main website.
It is now possible to use this chain of page elements on other pages. For example, we could use this red header box element twice on the same page, by nesting it in a constructor displaying two separated rows.
6. Testing

To test the correctness of the modules during development we used unit testing. Unit testing is a form of automated testing, which allows for quickly discovering errors after the changing of source code, as well as immediately testing a large volume of code for errors. The general structure of a unit test consists of a ‘set up phase’, a ‘testing phase’ and a ‘tear down phase’. The ‘set up phase’ sets up the environment, like database entries and global variables, to ensure a constant and realistic situation. The ‘testing phase’ consists of the running of the actual tests itself, usually consisting of multiple assertions on the input/output of a single method or group of related methods. The ‘tear down phase’ is responsible for removing the environment created in the ‘set up phase’, as well as any data left over from the ‘testing phase’. This involves removing database entries and clearing global variables. The tear down phase ensures the next testing run is not influenced by the current one.

For the actual testing we used the Unit Test Framework for PHP5, or UT5. The UT5 framework was chosen because it is similar to JUnit, and because it has decent support for classes. The elements we tested using UT5 are the Model and the Controller (for each module). The view and interface flow were not tested using unit tests, because these produce output that can not be easily verified using automated tests. Instead, these user interface elements were tested with case tests, which have to be executed manually. Functions in the controller that only serve to generate HTML output were also not tested using unit tests, for the same reason.

We started implementing the test classes for a module as soon as that module was completed, to ensure that module was actually finished and functioning correctly. This made writing the tests easy, since the implementation of that module was still fresh in mind. It also meant that if a module had to be extended, there was already a test to ensure the previous functionality was not influenced.

The results of all the test can be found in appendix B, there will also be further explained how we tested, what we tested and we will explain how we used case test to test the various user actions and user inputs.
7. Project Progress

The progress of the project is a very important factor in the end result. It is also a useful method to check if the development is on track with the time that has been reserved for each individual task. Following an extensive progress report on a weekly basis (for the complete duration of twelve weeks).

Week 1: 30 April – 5 May
On the 26th of April we paid a visit to the company SecureSpace, where we were introduced to the crew and to our workplaces. Since SecureSpace just moved to a new office it was still a bit of a mess and our desks were not there yet. That Monday we went to the office to get started on the project, however our desks were not yet installed. This was not very convenient however we sat down and discussed all the options we wanted to have for our project and put most of it down on paper, this involved a lot of brainstorming.

Once the computers were installed that week, we could really start on the requirements analysis. We got three Macs and one PC, which some of us were very pleased with, because some wanted to gain more experience on working with Macs. We started on the requirements analysis, in this phase it became very clear that the persons (from SecureSpace) involved in the project had a lot of contradicting views on the project. Therefore in the beginning the requirements analysis started a bit slow, because they kept changing. However by the end of the week we required global overview of what the system needed in terms of functionality. The main functionality consisted of four supported types of websites for which people can register:

- a general page (article page),
- a simple e-commerce site,
- an advanced e-commerce site,
- and a media site.

Another major requirement for the project was the modular design, the building of the different parts of the system in modules, which could operate as standalones and with the other modules. This topic required more research because we did not have a good idea on how to do that. The modular design was required because the company wanted to be able to use the modules in other projects once we were finished with the project, so they needed to be build in a way, in which they could easily be incorporated into another project. At the end of the week the main requirements were assembled and we agreed to work the next week on some small things for the company, so we would get a feeling on how the company approached things.

Week 2: 7 May – 11 May
The first two days of this week we worked for the company, to see how they approached projects. We also received the programming manual of the company, in which their code style is specified and how they solve particular problems. The next day (8th of May) we received a sales training from which it became more clear what they expected from our project. So we incorporated the new knowledge of the project in the requirements documents. We were told that the company already made a business model for our project, so we asked if we could see that, so we could incorporate that in the project analysis and description. However when they went searching for the document they could not find it and so we sat down with the business manager of the company and talked about the business model. The project was intended for small companies who could not afford a professional design company for the design of their website. However these small companies
understand the need for a website. The project should provide the small business with a solution, because now they can build their site with just a few mouse clicks and they only would have to pay the hosting costs.

The project was also meant as an advertising campaign for the company, because when someone registered for a website for free and they wanted more functionality they will automatically become a 'full' client of SecureSpace. This meeting made it a bit clearer what the project should contain in terms of functionality and we incorporated this knowledge in the requirements document. This day we also started on the plan of action, in which we could easily incorporate the business model.

The next day we started searching for a way to build the project in modules, after a while we found the design pattern model-view-controller the best suitable for the modular design. We did some more research on the subject and also found a way to let the different modules communicate with each other, through means of the ModuleLoader. At the end of the week the requirements document was nearly finished, it contained the business model, the deliverables and the functional and non-functional requirements.

**Week 3: 14 May – 18 May**

This week we specified the use cases, which took quite a while because we also thought about a lot of implementation details. We went over all the use cases with the project manager, he agreed on most of them, with some minor changes. In this week we also finished the plan of action, which had a lot of similarities with the requirements documents, therefore the plan of action did not require a lot of work.

We had a couple of meetings with the project manager in which he explained how they provided security in their websites; we also used that functionality in our project. Half way through the week we asked if we could get access to the development server, in order to experiment with the modular design and to see if we could get it to work. We also started on identifying the different modules and on the database design.

By the end of the week both the Plan of Action and the Requirements document were completely finished, including the database design and the design on the different modules. We also made a global planning, which we kept global, because we did not know how long the different modules would take to build. We let the business manager and the project manager go through the requirements document and they found it a complete solution, they proposed a few changes, which we changed. That weekend we sent the plan of action and the requirements document to the project manager of the TU Delft.

**Week 4: 21 May – 25 May**

This week we decided to begin working on the different modules and then specifically for the admin part of the generated website. We started on different modules and we encountered some problems on how to let the different modules communicate with each other, however we solved it by making some changes to the ModuleLoader. On the Monday of this week we got a programmers training on how to send and receive HTML mail with PHP. We used the structure of this training for the newsletter module. For the rest of the week we further developed the modules we were already working on, we also changed the structure of the database a couple of times to better meet the requirements of the project and to make it easier to implement. At the end of the week we also got a sales training which specified how to make a success out of your live, which was a bit boring but we had fun.

**Week 5: 28 May - 1 June**

This week we got a programmers training on how the company implements a shopping cart and how to make an order from a shopping cart. We also had to make sure that the shopping
cart had integration with PayPal. The project manager explained how he wanted to see that work. For the rest of the week we worked on the modules and made a couple of agreements on how to do certain things so we achieved coherence in the implementation style. By the end of the week after a lot of hours debugging we finally found out that the development server was unable to send out email, because it was not installed on the server. So we asked the project manager if he could solve that. Since we also started working on the media module we needed to install certain codecs and other programs in order to convert uploaded media into one specific format. We sat down with the project manager to do that, we needed to install the following programs:

- ffmpeg;
- lame encoder;
- yamdi;
- and mencoder

We installed the programs; however when we tried to make it work, it did not work and we asked if we could do it ourselves, because if we did it together with the project manager it would take too much time. So we got shell access to the server but we did not have enough rights (we did not have program execution rights) to perform the actions we needed. In the mean time the project manager was busy on something else, so we could not complete that part of the media module.

Since the other modules came close to being finished we began working on the website4free itself, we needed to make a complex install scripts which makes the required folders, copies the required files and makes the database, but we did not have enough rights on the server. Therefore we chose to make the wizard (admin section of the generated website) as complete and bug free as possible. We did some work on the website4free itself, but we could not get much further, because we did not have enough rights to perform some of the actions.

**Week 6: 4 June – 8 June**

This week we discussed how the project manager wanted the design for the main website4free itself. Two weeks before we already had chosen a couple of designs from templatemonster.com. The project manager choose one of the design. Because we hit a little bit of a standstill and the company had a small project that absolutely needed to be finished before Friday we agreed to work on that on Thursday and Friday. We had all the main functionality for that project finished by the end of the week and we agreed that the next week we would sit down with the project manager and the business manager to discuss our development of the project and to specify what we needed from the company so we can finish the project, because without those things it would be impossible to finish the project.

**Week 7: 11 June – 15 June**

On the Monday of this week, we gave a programming training, about the model-view-controller design pattern and how we implemented that in the website4free.co.nz project. We gave this training so that all the other programmers working at SecureSpace knew how we implemented our project, so if they wanted to make additions and/or changes they knew where to look. We specifically explained the division and what each division was for, for example the model consists mainly of queries, since it is responsible for the access/changing of data in the database.

This week we sat down with the business manager and with the project manager to discuss our progress with the project. We explained that we needed a lot of things from them, like the encoding of videos and a working mail server and a couple of other things. We also discussed the website4free.co.nz itself, specifically what kind of content they wanted and how it should look. After this meeting we went further on the other small project we agreed to do and made
a few changes. We also sat down with the client of this project and discussed what he thought of it, we agreed to a couple of small changes and made them. However the client came back several times this week and tried to get us to add more functionality outside the proposal, we explained that most of it was not possible, because we wanted to close this project so we could continue with the website4free.co.nz. On the Friday of the week we tried to upload the project of the client to his server, so we could be finished with that. However his server had some settings which were not compatible, so he needed to change that. Therefore he would return on Monday when the changes had been made. This was a bit annoying because we wanted to be done with the project, so we could continue with our own. This week we did not accomplish much for our own project, because the other project took longer than expected, because the client was not easily satisfied.

Week 8: 17 June – 22 June
In this week we explained to the project manager that we did not want to continue with the other project we agreed to do, because it took much longer than originally planned, because the client kept asking for changes. The project manager agreed and made sure we did not have to work any more on that project. During the week we fixed some errors of that small project and made sure it was working for the client and after that we continued with our own project. The project manager said this week, that he wanted to finish our project as soon as possible and said that the next week he would be available for us, so we could get along with the project and finish the main functionality. We were very pleased with this news, because now we could get the things we needed to finish the project. This week we focused on the website4free.co.nz itself the design and the content. The previous week we agreed with the business manager that she would provide the content for the website4free.co.nz this week, however she did not. So we finished the design and decided that they themselves had to fill in the content, because they did not make time free to do it. This week we also focused on the generation of the website, i.e. making the database, copying the files, integrating with the template. Because we used the model-view-controller-design pattern this could be done in quite an easy way. If a module was not loaded, it would simple not show in the template and this worked very nice. This week we also made an admin tool for SecureSpace in which they could see, when someone did not pay, was due to pay, etc. This admin tool can disable a site and put in back in the air and has some more similar features. Although this week we still had to wrap up bits and pieces of the other small project, we did manage to do a lot of work on our own project.

Week 9: 25 June – 29 June
The previous week the project manager asked us to make a global planning for the next week, because he wanted to know where we were with the project. He also asked if we needed something to finish the project, so we explained all the things we needed, like the privileges to create a database and the privileges to create users. Since the project manager now had given us almost everything we needed to finish the project, we managed to do a lot of work this week. We also fixed the media module, now the videos can be converted to flv format only the sound did not work because the lame encoder wasn't properly installed. On the Thursday of this week we gave a presentation of what we could do so far and the project manager was quite pleased with our work and had only a few remarks, which we fixed the next day. This week we also started with the testing of several modules which were nearly finished. Another major thing we fixed this week was the security in terms of logging in for the wizard and for the generated website (for registered customers). Since the project now was coming along good we also added an extra module, namely the gallery module, which allows clients to upload pictures and display them on his website using a slideshow. Since we were working on
security issues, we also made a captcha for the register page on the website4free main and for
the contact us page of the generated page, this way the sites cannot be used for spamming.
Everything considered, we managed to do quite a lot of work on the project and also the
report was getting along good and we would send that weekend a copy to the TU Delft.

**Week 10: 2 July – 6 July**
This week we finished all the testing of all the modules and we started on the testing report,
especially the case tests, since they could discover some bugs which still needed to be fixed.
The case tests became quite extensive and we would run through all the case tests to make
sure we did not have any bugs (or features). Since we did not have any content for the
website4free main page and we did want it to look very nice, we made a tool which allows
someone with html experience to edit the contents of the website. Since we did not receive
any content for the main page, we made this tool so SecureSpace could do it themselves at
any time they want.

We did receive some comments of the project manager but they were mostly for the design of
the website4free main page. The newsletter module was working; however for it to really
work we needed to make the project live, so that this module could be tested on an actual
generated website, because it required specific features which could only be tested once the
project was live. This week we asked if the project could be made live, however the project
manager he did not really have the time to do it. If the project was live we could also make
symbolic links, so when someone actually went to test.website4free.co.nz he would get
redirected to his own page. The project manager said in week 9 that he wanted to tweak the
wizard, so it would become more user-friendly and better looking. But since he did not make
clear how or what he wanted we did it ourselves and chose for the same design as the main
page so they would look the same. This week we also added a lot of extra functionality to the
admin tool. This was quite handy when it came to testing because we now could easily
administrate a website and change it by means of this admin tool (like making it active,
deleting any reference to it when we did no longer need it), add payments manually, change
user and domain details etc.

**Week 11: 9 July – 13 July**
This week our top priority was to make the project go live, so we could thoroughly test it to
fix any bugs we may come across. We asked several times for this, however we received little
response. Therefore we tweaked the wizard, making sure that all the modules were working in
a consistent way, so they would be more user-friendly and we finished the design for the
wizard. We also gave the admin tool the same design and we added the functionality that
someone could also register for a newsletter system only. This week we also tested the
various libraries we had. We went over the register process and the use of the wizard process,
so we would find any bugs and fixed them. The project was coming to an end, but we still did
not have some sufficient privileges to do some stuff. For example the video conversion
worked, but only from a shell (console, terminal), not when called from PHP. The newsletter
module was working, but we did not know if it would work once the project was live because
we still had to change a couple of things so it actually works. Since there were still a few
bugs, but we did not see the things we needed to fix them happen any time soon, we focused
more on the report and wanted to make sure that the report was as detailed as possible. One of
the major bugs was that the login functionality was build in the customer module, and an
article site does not come standard with a customer module, therefore when someone made an
article site, he could not login in the back end of the website (wizard) because he did not have
that functionality. Therefore we incorporated this functionality in a special login module,
which every type of website has, so they could all login to the wizard.
Week 12: 16 July – 20 July

This week the project went live, this required a bit of testing to fix some bugs, like absolute paths we used in some modules. Since the project now was live we managed to get the newsletter module working. We also made two designs from which people could choose once they had registered. We also worked hard on the report which came along nicely.

By the end of last week we finally got access to a working conversion tool, mencoder and the 'lame mp3' encoder. To be able to convert a media file to Flash Video, we need these two tools, but we need another tool to inject the metadata into the flash format. We searched for a long time on Google, and we found several options, flvmdi (flv metadata injector), flvtool2, and yamdi (yet another metadata injector). The first tool, flvmdi, makes use of Ruby on Rails, a fairly new player in the server side scripting area. Because SecureSpace did not want to install Ruby on Rails specially for this purpose, this tool fell off as a contender. Then there was the flvtool2, which is a tool that comes with a nice graphical user interface, but it is only available for Windows. Because the tool is only a GUI, (it uses the flvmdi tool at the back-end mentioned earlier), and because of the fact that it is a Windows only tool and our servers are running linux, this tool fell off as well. Yamdi was our last hope. It is a commandline only tool, and it is written in C, so we could just compile it, and try it out. After physically converting the media to flash video, we came up with an error, mencoder did not like the conversion to flash video (this feature is considered alpha, so it not a supported feature). Because of this, we are unable to convert media files to flash video, and therefore, the whole media site is not useable. so we made the final touches to the report and we gave a final presentation of what our project could do. Tuesday the 24th we send the report to the University and we were finished!
8. Conclusion and Recommendations

The purpose of the website4free project was to build a system that would give users the ability to sign up for the service, after which they would get a website with a sophisticated tool at the back end with which they could easily administrate their site. In that, the goal has been reached. The way in which the development team reached the goal however, could be better.

In the beginning when the requirements kept changing, the development team insisted on getting properly documented requirements, because this is the way it is being taught at the university. That is why we made an extensive requirements document and why the requirements analysis took longer than expected.

Agreements were made to evaluate the work that was done at the end of each week, to keep track of the progression. Unfortunately this only happened once, and at later meetings both the senior developer, as well as the business manager (project manager) were out of the office so no meeting could take place.

Because of this, the project did not develop at a rate that it could. However, the development team did not take this as an excuse and did its utmost best to overcome these problems by organizing some stuff themselves. We planned to do things ahead, and we made lists of things that we needed for the project, otherwise we could not finish the project. Some things took a long while to get, for instance the media conversion was only given on the last day of the project and making the project live was also a bit late (this also happened in the last week).

It is absolutely clear to us that the business manager and senior developer are very ambitious persons, trying to undertake as much as possible, in any area they possibly can. But because of this we sometimes felt like we got the leftovers, because they did not spend time on our project and we wondered if they were really interested. By the end of the project, the main website still did not have any real content, which we found a pity.

The development team struggled with tight privileges as well, or no privileges at all to do certain necessary operations. This was overcome by just implementing the functionalities, but to comment them out for the time being. This is one of the main reasons why we did not get the media conversion working, because on the last day of our project it was installed and after that we did not see any change to fix it.

At the end we can only say that the project has been delivered, on time, on specifications (except for the media conversion), with the functionalities that had been agreed on, it is only a shame that both the project manager, as well as the senior developer could not make more time free for us. Therefore we recommend for a future project, to be more involved to let them know you actually care about what they are doing.

However we did learn a lot, because building a system with changing requirements is of course a challenge, but we managed to make a system that works correctly and has a lot of functionality. We also learned a lot on how to develop a project and how to communicate with clients to get the requirements you want. As for the development of the project, we came across some problems, but we managed to fix them.
As for the experience in New Zealand, we can only say that we did a lot of cool things in our free time and the whole experience was well worth it.
Appendixes
Appendix A: Technical documentation
Table of Contents

1. INTRODUCTION................................................................................................................ 44

2. OVERALL DESIGN.............................................................................................................. 45
   2.1 MODEL-VIEW-CONTROLLER PATTERN............................................................................. 45

3. GENERATED WEBSITE...................................................................................................... 50
   3.1. LOADING OF CONTENT IN A GENERATED SITE.......................................................... 50
   3.2. LOADING OF CONTENT IN A GENERATED SITE DEPENDING ON THE USERS ACTIONS .......................................................... 51
       3.2.1 Loading of content in an e-commerce site ............................................................... 54
       3.2.2 More sequence diagrams of a generated site .......................................................... 58

4. WIZARD............................................................................................................................. 61
   4.1. OVERALL DESIGN ....................................................................................................... 61
   4.2. DATABASE DESIGN ................................................................................................... 62
   4.3. SOME IMPLEMENTATION DETAILS PER MODULE ..................................................... 64

5. MAIN WEBSITE4FREE.CO.NZ............................................................................................. 68
   5.1. DATABASE DESIGN .................................................................................................. 68
   5.2. IMPLEMENTATION DETAILS .................................................................................... 70

6. OPTIMIZATION AND VALIDATION REPORT....................................................................... 72
   6.1. REPORT ON THE ADMIN TOOL: ................................................................. 72
   6.2. REPORT ON A REGULAR WIZARD: ..................................................................... 74
   6.3. REPORT ON A GENERATED WEBSITE (MAIN PAGE) ......................................... 76
1. Introduction

This document contains the technical documentation for the project websit4free.co.nz. The document is divided into three main parts:

- The generated site;
- The wizard;
- The main website4free.co.nz.

Each part will discuss the major technical aspects of that particular type of the project, this will consist of sequence diagrams, data flow diagrams and how the database is constructed. We will start however with the overall design, in which is explained how the modules are built. In chapter 6 we also added a optimization and validation report, with the results from various (on-line) tests.
2. Overall Design

This part of the technical documentation specifies the overall design of our project. As stated many times before the modular design is the most important design issue of our project, therefore we will start with that.

2.1 Model-View-Controller pattern

In this paragraph we will discuss the modular design; the way we implemented all the modules and we will also explain how the moduleLoader works, because this is the backbone through which all the modules communicate with each other.

Figure 1: module overview (per website type)
The generated website is constructed out of different modules (see figure 1). This has the following advantages:

- modules allow changing a website very quickly, simply by copying the extra modules into the folders (and setting up the accompanying database tables);
- modules allow for individual construction and testing. Modules can be easily divided among the programmers;
- modules allow for increasing or decreasing the workload. Every module is an individual product, so a module can be dropped from or added to the project quite easily;
- modules allow for expanding the free website source code easily. A module could be expanded with new functionality without affecting other modules;
- modules can be reused for other projects. Once a module is finished, any website that needs the functionality provided by that module can just use that module.

The modules are implemented according to the model-view-controller design pattern. We will now discuss how we implemented the chosen design pattern, which will be further explained with an example. The Model will be responsible for retrieving and storing data into the database. The Model will therefore consist mostly of SQL queries. The View will be responsible for turning the data from the Model into valid (X)-HTML, which can be embedded into the template. The Controller is responsible for handling user input, i.e. deciding what action needs to be taken. The Controller is also responsible for processing input from another Controller (of a different module). This way, only the Controller has to be visible to the other modules, thus allowing for loose coupling between the modules.

To illustrate how the modules will work together, we give an example showing how a shopping cart overview page might be generated. The shopping cart overview will display a list of products that are in the users' virtual shopping cart. At some point, the PHP script generating the page will come across the central content area. At this point, the script will call the ModuleLoader (we will come back to the ModuleLoader a bit further) to retrieve the Shoppingcart Controller. If the shopping cart is not loaded, the ModuleLoader will construct a new Shoppingcart Controller, store it in its list of loaded modules, and return its link. The script will then continue by asking the Shoppingcart Controller to display a shopping cart overview. The Controller will request the Model to provide a list of the unique product names in the cart. Because the Shoppingcart Controller has no access to products, it will have to request the Product Controller to return a list of the actual products belonging to the names. To do so, the Shoppingcart Controller will first have to ask the ModuleLoader to provide the Product Controller. When the products are retrieved, the Shoppingcart Controller can ask the View to display a detailed list of Products, with their respective quantities (retrieved from the model) and buttons to change the contents of the shopping cart. Upon clicking one of the generated buttons, the Controller will have to respond to the user by invoking an edit method in the Model, and then to reload the page.

In this case, the shopping cart module will require the product module to perform its required action. Still, if no product module is available, the shopping cart controller will simply ask the shopping cart view to display an empty shopping cart, thus keeping the modules capable of working individually; see figures 2 and 3.
Figure 2: modular communication (1)
Singleton design pattern

To access the modules from any point on the website, we decided to use the Singleton design pattern (see figure 4). The Singleton pattern allows for a single point of entrance to retrieve a modules controller, instead of making every controller known to every other controller. Anyone that needs a controller for performing an action can call the 'getInstance' method, and this method will retrieve the same instance of a module every time. If a module is not loaded into memory yet, it is the Singletons responsibility to load the module. If a module is not available, because it has not been added to the website, it is the Singletons responsibility to provide a suitable response, without breaking the functionality (since some modules are optional).
Figure 4: Singleton design pattern
3. Generated website

This part of the technical design document specifies how the generated websites actions are passed to the modules and how the html output is passes back to populate the site.

3.1. Loading of content in a generated site

When a user browses to the generated website the index.php is loaded and the appropriate modules are loaded for the specific website type. This only calls to the ModuleLoader to get the instances of the modules. The next step is to build the site. How the main design is loaded can be found in the document ‘Style a generated site for Dummies’. Here we will discuss, with sequence diagrams, how depending on the users actions, the appropriate content is loaded. We will start with the loading of the menu and of the category list, see figures 5 and 6.

![Figure 5: loading of a menu in a generated site](image)

```plaintext
| index.php | Menu/Controller | ModuleLoader | ArticleController | ArticleModel | MenuModel | MenuView |
```

**getMenuRealTime()**

**getInstances(Article)**

**ArticleController**

**getAllStaticPagesPublished()**

| pages |

**getAllMenus()**

**displayMenuRealTime(pages)**

**html output**

**getResult()**

**result**
Other content which is loaded when a user browses to a generated website, has the same sequence of events, some involve more modules, than others, but the idea is the same.

### 3.2. Loading of content in a generated site depending on the users actions

For the main content of the generated website, there are many options of what kind of content can be loaded, for example a static page can be loaded, an article, a product, a register page, edit details page, media list, etc. We will discuss some of them next.

When a user browses to the index, then the static page home is always loaded. The loading of a static page goes as follows (see figure 7):

![Figure 7: loading of a static page in a generated site](image)

For the loading of an article, there are a few more steps involved, because when there is a single article in a submenu, than the single view for an article is loaded. If there are more articles, then an article list view is loaded as can be seen below, see figure 8. The loading of a single article is the same as that of a static page, only the methods are different.
Figure 8: loading of an article list in a generated site
The gallery is also a static page, however the loading of the gallery content goes a little bit different then that of a 'normal' static page, see figure 9.

Figure 9: loading of the various views for the gallery in a generated site

When a user clicks on a category in a media site the media list view is loaded, this can be seen in the figure 10. This is an example of what happens when a user clicks on a category.
3.2.1 Loading of content in an e-commerce site

Here we will discuss some of the content which can be loaded into an e-commerce site. The most important modules for an e-commerce site are the product module, the shopping cart module and the order module. This will become clearer in the sequence diagrams to come.

For the e-commerce sites, the loading of a product is divided into two views, the single product view and the multiple product view, which will be discussed here. If a user clicks on a category the appropriate products are fetched, this is the same process as when the user fills in a search. This process requires the order controller to display the **buy now** button. This process also requires the shopping cart controller to display the **add to shopping cart** button, however this is not incorporated into the sequence diagram, because it involves the same process as getting the **buy now** button from the order controller (only with different controller and with different methods), see figures 11 and 12.

For a single view of a product the process has a lot of similarities as with the loading of the product list, for instance it also needs the shopping cart controller and the order controller for the appropriate buttons, see figure 12.
Here we will discuss the some of the sequence diagrams leading up to going to PayPal and paying for the products. The process of paying with PayPal is done with a DFD because then the overall sequence of steps involving the payment becomes clearer, see figure 17. We will start with the sequence diagrams leading up to the payment and then we will discuss the payment process.

An e-commerce site plus has a shopping cart mini view. If the user clicks on a add to cart button, this mini view is updated, see figure 13.

The shopping cart full overview and the order views are loaded in a popup if someone has JavaScript enabled, if not these views are loaded in the main content of the generated site.
The shopping cart overview has several options for the user, clear the shopping cart, update the quantities the order now button. These actions are displayed in figure 14.

Figure 14: shopping cart actions in a generated site

When a user proceeds to the order now view, his details get loaded into the various input fields if he is logged in, this can be seen in figure 15. How that order view is loaded is explained in figure 16.

Figure 15: loading of the details of a logged in user in a generated site
Figure 16: loading of the order view in a generated site

As stated earlier, the data flow diagram in figure 17 shows the overall process of buying a product by means of PayPal.

Figure 17: overall process of buying in a generated site (with PayPal)
3.2.2 More sequence diagrams of a generated site

Now we will discuss the various sequence diagrams, that have to do with register, edit details, login/logout and becoming a member. We will start with the register sequence diagram, see figure 18. In figures 18 and 19, there is a distinction if a user fills in wrong content, then the page is reloaded, stating which content is wrong.

**Figure 18: registering in a generated site**

The edit details page gets loaded, when the user clicks on the edit details link, first part of the sequence diagram in figure 19. The second part of the sequence diagram in figure 19 explains how the content is saved. In figure 20 the process of logging in and logging, this is same as process for logging in by the wizard.
Figure 19: loading the content of the edit details page in a generated site

Figure 20: logging in and logging out in a generated site and in the wizard
Here we will discuss some events specific for the media site, these are becoming a member and watching membership content, see figure 21 and 22. Since becoming a member involves paying with PayPal, this is done with a DFD diagram.

![Figure 21: becoming a member in a generated site (with paypal)](image)

![Figure 22: watching a media file in a generated site](image)
4. Wizard

Because the options an user has in the wizard of his generated website are numerous, we will not discuss all of them, however per module we will discuss some important technical implementation details, but before that we will discuss the overall design of the wizard and the database behind the wizard. The database behind the wizard is the same as that of a generated site.

4.1. Overall design

Here we will explain how user actions are redirected to the appropriate module. If a user is in the wizard and performs a certain action, than in the get parameter is set for which module this action is intended. In the wizard directory there is a file that links to the appropriate module. The get parameters are checked with the vcl Security library and passed along to the appropriate Controller, which then performs the required action. In this way all the variables going to the module, whether they are get or post variables are safe to be processed. This process can be seen in figure 23. There are of course variations of this sequence diagram, but all these variations are much like this sequence.

![Figure 23: handling user action/user input in the wizard](image-url)
4.2. Database design

As the reader can see clearly in figure 24, the database design for the generated website has a lot of relations. This is because some modules are capable of interacting with other modules, when they are available that is. Next we will discuss each table separately, not in too much detail, because it is very intuitive how the tables work.

**Figure 24: database design for the wizard and the generated site**

The `webshop_product` table is used by both the product module, as well as the media module. That means that the `webshop_product` table is both used for real and virtual products. Some options are only useful for the real products, and some are only useful for virtual products. The properties that are only useful for the real products are: min_stock, stock, shelf_stock, GST and bargain. Downloaded, streamed, demo, real and membership only are only useful for virtual products. The `news_inc` and `news_txt` properties are only useful if the user can send newsletters, because with these properties the user can control if a certain product is to be included in a newsletter, and which text there should be assigned to the product.

The `webshop_category` table is used to store categories. Categories can be used by many different modules, for instance the media site and both the e-commerce sites can make use of categories with the media module and the product module. Categories can go infinite many
levels deep in theory, but due to layout issues with more than a few levels deep we limited this to two.

The \textit{wff\_order} table is used to store all the orders a website has. These orders are made (for a part) from the response the website gets from PayPal. Depending on the response the orders get a particular status, for more information on this we refer the reader to the main report, chapter 5.

The \textit{wff\_stats} table, or statistics table, is used to store data that can be used to show statistics on the wizard. This way the site administrators can keep track of sales, or the effect that certain advertisement programs have on their sales, like the products on the frontpage. The statistics table keeps track of who bought which product, how many times on a certain date.

The \textit{wff\_user} table is used by both the login module and the customer module. The login module uses the user table to check if a user who wants to login on the site is an administrator for the site or not. If he or she is, then it is also possible to access the wizard of the website. The customer module uses the table to store users who registered on a generated website. This way people can login to see for instance membership videos only. There is a link from accesslevel\_id to distinguish regular users from administrators of the website.

The \textit{webshop\_accesslevel} table holds data for different kinds of access levels. The only used access levels now are a regular user or an administrator. In theory each page on the generated website can be protected so only users with access level x or higher are able to see the webpage.

The \textit{wff\_membership} table holds data for the membership module, which is used for the media site type. Data stored in this table consists of information on the users who got a subscription to a generated website.

The \textit{membership\_details} table holds data for the different kinds of memberships on offer.

The \textit{newsletter\_customers} table is used to store information about customers who want to receive the newsletter of the generated website. The reason that there is duplicate information in the database (some of the data is already stored in the subscribers table) is because the following reason. Users can also subscribe for a newsletter only site and that sites uses the newsletter tables, not the customer table. Therefore we decided to make a separate table for the newsletter service.

The \textit{newsletter} table stores the different newsletters sent by the system. This is done because some e-commerce sites like to have a recurring sale each year at the same time, so they can resend an old newsletter, but then with a couple of changes.

The \textit{wff\_main} table is used for main pages, that come standard with every website. These include Home, Contact Us, Frequently Asked Questions, About Us and Gallery.

The \textit{wff\_page} table is used for new pages that are made by the user with the article module. Because each article has to be linked to a submenu entry, there is a link to wff\_submenu\_id; the submenu that holds this article. Because some articles can be important, a special flag 'on_top' can be set, to make these articles appear on top when clicked on the appropriate submenu.
The *wff_menu* stores the menu entries of a site in. The *wff_submenu* has a link to a parent menu. The order can be changed to change the order in which the sub menu entries appear on the main page.

The *wff_help* table is used to build up the frequently asked questions page.

### 4.3. Some implementation details per module

Here we will discuss some implementation details for some of the modules, we will not discuss every module, because for some there is not much to say.

**Article Module**

The most interesting thing for the article module is the WYSIWYG editor. This editor was not our first choice, however since SecureSpace already worked with it, we also used it. To incorporate this editor into the html output is very easy, because all one has to do is replace a text area with a certain id, with the editor. However the use (friendliness) of the editor is not very high, for example it does not work properly in every browser. The editor does however have a lot of options, but we disabled a lot of options, because too many options are not that user friendly, because the user might get overwhelmed by all that. The options which are left allow for a nice page, with text and with some pictures. During the project we tried to keep every page XHTML strict, however the editor generated HTML code which is not strict. So the only thing which is not strict on a generated page is the content created with the editor.

**Category and Menu Modules**

As one could see the category and the menu have a different database structure (see paragraph 4.2.), however there use is the same. For the menu we created two database tables because, the menu only had to be two entries deep and because this was recommended by the project manager. However as we implemented the menu module, it did not really work as expected and we needed to write a lot of extra code to make it work. Therefore we decided not to use this approach for the category. In the category module we kept the database entries in one table ands used parent id’s, to distinguish between parent and child. The category also had to be able to cope with more than two levels, but this would only be used if the generated site became customized by SecureSpace. Therefore this database structure was more convenient for that purpose.

**Gallery Module**

For the gallery module, which was added later on in the project, we wanted to make this module as dynamic as possible, therefore we decided not to use a database table. Instead we only would read out folders. For the user friendliness of the gallery module we used a flash embedded object, which allows for multiple upload of pictures. To get the flash upload working was a bit tricky, because it is not easy to debug, however we managed to get it working.
Media Module
The most interesting thing for the media module is the conversion from uploaded content, to the \textit{flv} format, which can be played by the flash player. Because the project manager was very busy we just managed to get it working before the deadline, because then we had everything we needed. The process of conversion can be seen in figure 25.

\begin{center}
\includegraphics[width=\textwidth]{media_conversion_diagram.png}
\end{center}

\textit{Figure 25: media file conversion in the wizard}
Order Module
The order process is already explained in the chapter 3, however here we will explain what happens to the stock if a user orders a product by means of PayPal. Then the administrator and the user receive an email of this transaction. This can be seen in the DFD, in figure 26.

Figure 26: stock management in the wizard
Statistics Module
For the generation of the images that are being used in the statistics module, we used PHP in combination with the GD library, which is a library specialised in generating images through scripting languages like PHP, Perl, etc. The GD library uses functions similar to OpenGL functions, for the generation of the images, for instance imageLine, to generate a line in the image. The images use only a few of the many available options, but one interesting is imageFill, which uses a filling algorithm that is named flood fill algorithm. After pointing out a beginning pixel, the algorithm will recursively fill the pixel above, below, and the two pixels on either side with the same colours, until a pixel has another colour than the begin pixel. This function is used with the profit chart, to colorize the surface beneath the profit line.

Template
The template module allows the user to change its template, to another template. How to construct such a template is specified in another document. If a user clicks on a preview image in the template module his template is immediately changed to one he clicked. This is done by copying the index.php, css file and the images needed for the template to the appropriate directories. Since every template us build to cope with every specific type of website, the user can switch to any template he wants.

Newsletter
The newsletter module is build in such a way that it is a three step process for the user to send out a newsletter campaign, as can be seen in figure 27. These steps are upload subscribers, make the newsletter and send out the newsletter.

![Figure 27: newsletter campaign steps in the wizard](image-url)
5. Main website4free.co.nz

Here we will discuss the major design issues of the main website4free.co.nz. We will discuss the database design and the register process; we will not discuss the use of the editor because this is already done in the main report chapter 5.

5.1. Database Design

In figure 28 the main website4free.co.nz database design can be seen, we will discuss all the tables, but we will not go into much detail, because it is pretty straightforward.

The *admins* table is used to store administrators for the website4free project. Since no 3\textsuperscript{rd} party members are allowed to administrate the project, there is no registration page. Every administrator has to be inserted in the database manually. The hash consists of the username, concatenated with the MD5 hash of the password. The last login time and date is also stored. This login is used by for the admin tool of the website4free.co.nz main website.

The *hosting_type* table is used to store details about the available hosting types. Not every website type can make use of every hosting type, because not every website type is going to

![Figure 28: main website4free.co.nz database design](image-url)

The *templates* table is used to store details about the available templates. Not every website type can make use of every template, because not every website type is going to
make the same amount of traffic each month, an article site will generate much less traffic than for instance a media site.
The 'type' property contains a small description of the hosting type, 'data' contains the amount of megabytes this hosting type may use on disc, and 'traffic' means how much gigabytes the site which uses this hosting type may use every month. The price is stored in cents so the problems with value separators (dots or commas) can be taken care of in the PHP logics, and not in the database.

The `site_type` table contains all the different kinds of websites the website4free offers. Every website type has its own calculated necessities in terms of calculated data on disc or data traffic. It is highly unlikely that for instance an article site outruns an e-commerce+ site in terms of data on disc or data traffic. The 'min_data' and 'min_traffic' properties mean minimum values necessary for the specific site type.

The `templates` table is used to store the available templates in the database. When a user subscribes to the service, he can choose a design from these templates to let his site look completely different. When a user wants to buy a unique design, that's also possible, but the template will not come into the database, because otherwise every other subscriber could choose this template. The 'basedir' property contains the directory in which the template is stored inside the 'template' directory already in the directory tree of the whole project. 'Preview' contains a small image which is visible in the wizard, when users are viewing a listing of the available templates. This will not make a big influence on the load of the database server because the template images are very small in size.

The `subscribers` table is one of the most important tables we use in our main database; this table holds all the customers who signed up for the service. Besides the usual properties, there are a few interesting properties found in this table, for instance paypal_business_email. This is used when the subscriber has chosen for a media, or one of the e-commerce websites. When a subscriber has such a site, it is possible to buy something, and because everything is done by PayPal, we need to know the receivers email address to embed in our dynamic PayPal button generator. This is also the reason that you have to fill in a PayPal business email address when registering for the service. The database user and password are also stored in the database, because when there is a problem, we can extract the database user and password easily from the admin tool. This is done because the database user and password are dynamically generated and cannot be inferred from any data whatsoever.

The `orders` table is also one of the key tables in the database. It stores all the different orders that came in for the project. This data is derived from the response we get back when someone paid by means of PayPal. If someone paid by means of PayPal, his site is activated and he can use it. User subscribe for the service therefore they have to do regular payments, if a payment is missed the site is automatically deactivated.

The `page` table is used for the creation of the main website. For more information on how this is used we refer the reader to the main report chapter 5, were the creation of the website4free.co.nz main website is explained.

The `page_constructor` table is the heart of the dynamic website page generation. The page constructor table is used as a sort of container. Each such container can contain for instance a picture and some text, or a table with a table title, heading and body. That body can contain another container, for instance the earlier stated picture with text. There are infinite number of
options available, because everything is possible, a picture on the left and text to the right, on top, at the bottom, or two pictures, with text, the first picture on the left, the second on the right, or both on top with text beneath them, etc, (for more information see chapter 5 in the main report).

The page element table is linked with the constructor table. The page element table contains the end elements. The page constructor can be infinite levels deep, with at the end (the leaves of the generated tree) a page element (for more information see chapter 5 in the main report).

5.2. Implementation details

Here we will discuss the register process on the main website4free.co.nz, as can be seen in figure 29.

Figure 29: register process on the main website4free.co.nz
If a user registered for a certain website type, he gets all the appropriate models. Each module has its own database table. To make sure that all the database tables are created, each module has a createDatabase function in his model. If a module is copied to the generated site, this function is executed, therefore his database is created. This allows for loose coupling, because the copying of the module, automatically created the associated database tables and then the user can use his site. The generated site is populated with dummy content to make the use of the wizard and of the website clearer.
6. Optimization and Validation Report

Because SecureSpace wants to use this project for people who do not have a broadband connection, or at least not a very fast broadband connection that we are used to ourselves, it is very important that the website stays small in size. With this in the back of our minds, we developed the project in such a way, that we are using as little disk space as possible for every single page. We used special services offered on the Internet to test the user friendliness when it comes to loading speeds of our websites, and we found out that our web pages were really qualified to use even if you have a small band connection. We used services provided by www.websiteoptimization.com to test our web pages. To give the reader an idea what we are talking about, here are some tangible values for a few web pages.

6.1. Report on the Admin tool:

| URL: | http://www.websiteforfree.co.nz/admin/index.php |
| Title: | Admin Tool |
| Date: | Report run on Tue Jul 17 16:26:16CDT2007 |

### Diagnosis

| Total HTTP Requests: | 8 |
| Total Size: | 9591 bytes |

### Object Size Totals

<table>
<thead>
<tr>
<th>Object type</th>
<th>Size (bytes)</th>
<th>Download @ 56K (seconds)</th>
<th>Download @ T1 (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML:</td>
<td>1178</td>
<td>0.43</td>
<td>0.21</td>
</tr>
<tr>
<td>HTML Images:</td>
<td>3000</td>
<td>0.80</td>
<td>0.22</td>
</tr>
<tr>
<td>CSS Images:</td>
<td>1259</td>
<td>1.25</td>
<td>1.01</td>
</tr>
<tr>
<td>Total Images:</td>
<td>4259</td>
<td>2.05</td>
<td>1.23</td>
</tr>
<tr>
<td>Javascript:</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CSS:</td>
<td>4154</td>
<td>1.03</td>
<td>0.22</td>
</tr>
<tr>
<td>Multimedia:</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Other:</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### External Objects

<table>
<thead>
<tr>
<th>External Object</th>
<th>QTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total HTML:</td>
<td>1</td>
</tr>
<tr>
<td>Total HTML Images:</td>
<td>1</td>
</tr>
<tr>
<td>Total CSS Images:</td>
<td>5</td>
</tr>
<tr>
<td>Total Images:</td>
<td>6</td>
</tr>
<tr>
<td>Total Scripts:</td>
<td>0</td>
</tr>
</tbody>
</table>
Total CSS imports: 1
Total Frames: 0
Total Iframes: 0

Download Times*

<table>
<thead>
<tr>
<th>Connection Rate</th>
<th>Download Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.4K</td>
<td>9.03 seconds</td>
</tr>
<tr>
<td>28.8K</td>
<td>5.32 seconds</td>
</tr>
<tr>
<td>33.6K</td>
<td>4.79 seconds</td>
</tr>
<tr>
<td>56K</td>
<td>3.51 seconds</td>
</tr>
<tr>
<td>ISDN 128K</td>
<td>2.19 seconds</td>
</tr>
<tr>
<td>T1 1.44Mbps</td>
<td>1.65 seconds</td>
</tr>
</tbody>
</table>

*Note that these download times are based on the full connection rate for ISDN and T1 connections. Modem connections (56Kbps or less) are corrected by a packet loss factor of 0.7. All download times include delays due to round-trip latency with an average of 0.2 seconds per object. With 8 total objects for this page, that computes to a total lag time due to latency of 1.6 seconds. Note also that this download time calculation does not take into account delays due to XHTML parsing and rendering.

Analysis and Recommendations

- **TOTAL_HTML** - Congratulations, the total number of HTML files on this page (including the main HTML file) is 1 which most browsers can multithread. Minimizing HTTP requests is key for web site optimization.
- **TOTAL_OBJECTS** - Caution. You have 8 total objects on this page. Consider reducing, eliminating, and combining external objects (graphics, CSS, JavaScript) to reduce the total number of objects, and thus separate HTTP requests.
- **TOTAL_IMAGES** - Caution. You have a moderate amount of images on this page (6). Consider using fewer images on the site or try reusing the same image in multiple pages to take advantage of caching.
- **TOTAL_CSS** - Congratulations, the total number of external CSS files on this page is 1. Because external CSS files must be in the HEAD of your HTML document, they must load first before any BODY content displays. Although they are cached, CSS files slow down the initial display of your page.
- **TOTAL_SIZE** - Congratulations, the total size of this page is 9591 bytes. This page should load in less than eight seconds on a 56Kbps connection - or 3.51 seconds. But there's always room for improvement.
- **HTML_SIZE** - Congratulations, the total size of this HTML file is 1178 bytes, which less than 20K. Assuming that you specify the HEIGHT and WIDTH of your images, this size allows your page to display content in well under 8 seconds, the average time users are willing to wait for a page to display without feedback.
- **IMAGES_SIZE** - Congratulations, the total size of all your images is 4259 bytes, which is less than 15K. Ideally each image should be less than 1160 bytes, to easily fit into one TCP-IP packet.
- **CSS_SIZE** - Caution. The total size of your external CSS is 4154 bytes, which is
above 4080 bytes and less than 8K. For external files, try to keep them less than 1160 bytes to fit within one higher-speed TCP-IP packet (or an approximate multiple thereof). Consider optimizing your CSS and eliminating features to reduce this to a more reasonable size.

- MULTIM_SIZE - Congratulations, the total size of all your external multimedia files is 0 bytes, which is less than 4K.

6.2. Report on a regular Wizard:

| URL: | http://test.websiteforfree.co.nz/wizard/ |
| Title: | Wizard |
| Date: | Report run on Tue Jul 17 17:01:33CDT2007 |

**Diagnosis**

<table>
<thead>
<tr>
<th>Global Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total HTTP Requests:</td>
</tr>
<tr>
<td>Total Size:</td>
</tr>
</tbody>
</table>

**Object Size Totals**

<table>
<thead>
<tr>
<th>Object type</th>
<th>Size (bytes)</th>
<th>Download @ 56K (seconds)</th>
<th>Download @ T1 (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML:</td>
<td>1585</td>
<td>0.52</td>
<td>0.21</td>
</tr>
<tr>
<td>HTML Images:</td>
<td>3000</td>
<td>0.80</td>
<td>0.22</td>
</tr>
<tr>
<td>CSS Images:</td>
<td>1259</td>
<td>1.25</td>
<td>1.01</td>
</tr>
<tr>
<td>Total Images:</td>
<td>4259</td>
<td>2.05</td>
<td>1.23</td>
</tr>
<tr>
<td>Javascript:</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CSS:</td>
<td>5464</td>
<td>1.29</td>
<td>0.23</td>
</tr>
<tr>
<td>Multimedia:</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Other:</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**External Objects**

<table>
<thead>
<tr>
<th>External Object</th>
<th>QTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total HTML:</td>
<td>1</td>
</tr>
<tr>
<td>Total HTML Images:</td>
<td>1</td>
</tr>
<tr>
<td>Total CSS Images:</td>
<td>5</td>
</tr>
<tr>
<td>Total Images:</td>
<td>6</td>
</tr>
<tr>
<td>Total Scripts:</td>
<td>0</td>
</tr>
<tr>
<td>Total CSS imports:</td>
<td>1</td>
</tr>
<tr>
<td>Total Frames:</td>
<td>0</td>
</tr>
<tr>
<td>Total Iframes:</td>
<td>0</td>
</tr>
</tbody>
</table>
### Download Times*

<table>
<thead>
<tr>
<th>Connection Rate</th>
<th>Download Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.4K</td>
<td>10.36 seconds</td>
</tr>
<tr>
<td>28.8K</td>
<td>5.98 seconds</td>
</tr>
<tr>
<td>33.6K</td>
<td>5.36 seconds</td>
</tr>
<tr>
<td>56K</td>
<td>3.85 seconds</td>
</tr>
<tr>
<td>ISDN 128K</td>
<td>2.29 seconds</td>
</tr>
<tr>
<td>T1 1.44Mbps</td>
<td>1.66 seconds</td>
</tr>
</tbody>
</table>

*Note that these download times are based on the full connection rate for ISDN and T1 connections. Modem connections (56Kbps or less) are corrected by a packet loss factor of 0.7. All download times include delays due to round-trip latency with an average of 0.2 seconds per object. With 8 total objects for this page, that computes to a total lag time due to latency of 1.6 seconds. Note also that this download time calculation does not take into account delays due to XHTML parsing and rendering.

### Analysis and Recommendations

- **TOTAL_HTML** - Congratulations, the total number of HTML files on this page (including the main HTML file) is 1 which most browsers can multithread. Minimizing HTTP requests is key for web site optimization.
- **TOTAL_OBJECTS** - Caution. You have 8 total objects on this page. Consider reducing, eliminating, and combining external objects (graphics, CSS, JavaScript) to reduce the total number of objects, and thus separate HTTP requests.
- **TOTAL_IMAGES** - Caution. You have a moderate amount of images on this page (6). Consider using fewer images on the site or try reusing the same image in multiple pages to take advantage of caching.
- **TOTAL_CSS** - Congratulations, the total number of external CSS files on this page is 1. Because external CSS files must be in the HEAD of your HTML document, they must load first before any BODY content displays. Although they are cached, CSS files slow down the initial display of your page.
- **TOTAL_SIZE** - Congratulations, the total size of this page is 11308 bytes. This page should load in less than eight seconds on a 56Kbps connection - or 3.85 seconds. But there's always room for improvement.
- **HTML_SIZE** - Congratulations, the total size of this HTML file is 1585 bytes, which less than 20K. Assuming that you specify the HEIGHT and WIDTH of your images, this size allows your page to display content in well under 8 seconds, the average time users are willing to wait for a page to display without feedback.
- **IMAGES_SIZE** - Congratulations, the total size of all your images is 4259 bytes, which is less than 15K. Ideally each image should be less than 1160 bytes, to easily fit into one TCP-IP packet.
- **CSS_SIZE** - Caution. The total size of your external CSS is 5464 bytes, which is above 4080 bytes and less than 8K. For external files, try to keep them less than 1160 bytes to fit within one higher-speed TCP-IP packet (or an approximate multiple thereof). Consider optimizing your CSS and eliminating features to reduce this to a more reasonable size.
- **MULTIM_SIZE** - Congratulations, the total size of all your external multimedia files
is 0 bytes, which is less than 4K.

6.3. Report on a generated website (main page)

<table>
<thead>
<tr>
<th>URL:</th>
<th><a href="http://test.websiteforfree.co.nz/">http://test.websiteforfree.co.nz/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Title:</td>
<td></td>
</tr>
<tr>
<td>Date:</td>
<td>Report run on Tue Jul 17 17:05:45CDT2007</td>
</tr>
</tbody>
</table>

**Diagnosis**

**Global Statistics**

- Total HTTP Requests: 27
- Total Size: 36907 bytes

**Object Size Totals**

<table>
<thead>
<tr>
<th>Object type</th>
<th>Size (bytes)</th>
<th>Download @ 56K (seconds)</th>
<th>Download @ T1 (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML:</td>
<td>7049</td>
<td>1.60</td>
<td>0.24</td>
</tr>
<tr>
<td>HTML Images:</td>
<td>8549</td>
<td>2.30</td>
<td>0.65</td>
</tr>
<tr>
<td>CSS Images:</td>
<td>11357</td>
<td>6.26</td>
<td>4.06</td>
</tr>
<tr>
<td>Total Images:</td>
<td>19906</td>
<td>8.56</td>
<td>4.71</td>
</tr>
<tr>
<td>Javascript:</td>
<td>2528</td>
<td>0.90</td>
<td>0.41</td>
</tr>
<tr>
<td>CSS:</td>
<td>7424</td>
<td>1.68</td>
<td>0.24</td>
</tr>
<tr>
<td>Multimedia:</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Other:</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**External Objects**

<table>
<thead>
<tr>
<th>External Object</th>
<th>QTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total HTML:</td>
<td>1</td>
</tr>
<tr>
<td>Total HTML Images:</td>
<td>3</td>
</tr>
<tr>
<td>Total CSS Images:</td>
<td>20</td>
</tr>
<tr>
<td>Total Images:</td>
<td>23</td>
</tr>
<tr>
<td>Total Scripts:</td>
<td>2</td>
</tr>
<tr>
<td>Total CSS imports:</td>
<td>1</td>
</tr>
<tr>
<td>Total Frames:</td>
<td>0</td>
</tr>
<tr>
<td>Total Iframes:</td>
<td>0</td>
</tr>
<tr>
<td>Connection Rate</td>
<td>Download Time</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>14.4K</td>
<td>34.00 seconds</td>
</tr>
<tr>
<td>28.8K</td>
<td>19.70 seconds</td>
</tr>
<tr>
<td>33.6K</td>
<td>17.66 seconds</td>
</tr>
<tr>
<td>56K</td>
<td>12.76 seconds</td>
</tr>
<tr>
<td>ISDN 128K</td>
<td>7.65 seconds</td>
</tr>
<tr>
<td>T1 1.44Mbps</td>
<td>5.60 seconds</td>
</tr>
</tbody>
</table>

*Note that these download times are based on the full connection rate for ISDN and T1 connections. Modem connections (56Kbps or less) are corrected by a packet loss factor of 0.7. All download times include delays due to round-trip latency with an average of 0.2 seconds per object. With 27 total objects for this page, that computes to a total lag time due to latency of 5.4 seconds. Note also that this download time calculation does not take into account delays due to XHTML parsing and rendering.

Analysis and Recommendations

- **TOTAL_HTML** - Congratulations, the total number of HTML files on this page (including the main HTML file) is 1 which most browsers can multithread. Minimizing HTTP requests is key for web site optimization.
- **TOTAL_OBJECTS** - Warning! The total number of objects on this page is 27 - consider reducing this to a more reasonable number. Combine, refine, and optimize your external objects. Replace graphic rollovers with CSS rollovers to speed display and minimize HTTP requests.
- **TOTAL_IMAGES** - Warning! The total number of images on this page is 23, consider reducing this to a more reasonable number. Combine, refine, and optimize your graphics. Replace graphic rollovers with CSS rollovers to speed display and minimize HTTP requests.
- **TOTAL_CSS** - Congratulations, the total number of external CSS files on this page is 1. Because external CSS files must be in the HEAD of your HTML document, they must load first before any BODY content displays. Although they are cached, CSS files slow down the initial display of your page.
- **TOTAL_SIZE** - Caution. The total size of this page is 36907 bytes, which will load in over 8 seconds on a 56Kbps modem - or 12.76 seconds. Consider reducing total page size to less than 30K to achieve sub eight second response times on 56K connections. Be sure to provide feedback for pages over 30K by layering your design to display useful content within the first two seconds. Consider optimizing your site with Speed Up Your Site or contacting us about our optimization services.
- **TOTAL_SCRIPT** - Congratulations, the total number of external script files on this page is 2. External scripts are less reliably cached than CSS files so consider combining scripts into one, or even embedding them into high-traffic pages.
- **HTML_SIZE** - Congratulations, the total size of this HTML file is 7049 bytes, which less than 20K. Assuming that you specify the HEIGHT and WIDTH of your images, this size allows your page to display content in well under 8 seconds, the average time users are willing to wait for a page to display without feedback.
- **IMAGES_SIZE** - Caution. The total size of all your images is 19906 bytes, which
exceeds 15K. Consider optimizing and creatively cropping your images, and combining them where appropriate. Even better, replace graphic text and form controls with styled text to eliminate unnecessary HTTP requests. Ideally each image should be less than 1160 bytes, to easily fit into one TCP-IP packet.

- **SCRIPT_SIZE** - Congratulations, the total size of all your external scripts is 2528 bytes, which is less than 4080 bytes. This will fit into three higher-speed TCP-IP packets.

- **CSS_SIZE** - Caution. The total size of your external CSS is 7424 bytes, which is above 4080 bytes and less than 8K. For external files, try to keep them less than 1160 bytes to fit within one higher-speed TCP-IP packet (or an approximate multiple thereof). Consider optimizing your CSS and eliminating features to reduce this to a more reasonable size.

- **MULTIM_SIZE** - Congratulations, the total size of all your external multimedia files is 0 bytes, which is less than 4K.

As one can see, both the admin tools, both the one for the website4free administrators, and the one for the site owners as well, are very small in size. The generated website however is of a bigger size. There has been done a tremendous amount of work to keep the size as small as possible, for instance by paying attention to the file extensions we use for images, because the used file extensions are optimized. The result is a neat looking website, which loads fairly fast on even a 56Kbps modem. Of course when using a broadband connection there is no problem whatsoever with the size of the web pages.
Appendix B: Test Report
www.website4free.co.nz

Bachelor project IN3700

TEST RAPPORT

Version 1.5
July 2007

Project Group
Remy Alidarso 1221736
Zdravko Anguelov 1232436
Terence Bahlen 1217313
Frits de Nijs 1224263

Company
SecureSpace
Business Manager: Maria Veleva
Project Manager: Dimitar Velev

University
Delft, University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science

Commission
Project Manager: Ir. B.R. Sodoyer
Project Accompanier: J. de Vries
# Table of Contents

1. **INTRODUCTION** ............................................................................................................................... 82

2. **MODULE TESTING** ............................................................................................................................ 84
   - 2.1. ARTICLE MODULE ......................................................................................................................... 84
   - 2.2. CATEGORY MODULE ...................................................................................................................... 85
   - 2.3. CUSTOMER MODULE .................................................................................................................... 85
   - 2.4. GALLERY MODULE .......................................................................................................................... 86
   - 2.5. MEDIA MODULE .............................................................................................................................. 86
   - 2.6. MEMBERSHIP MODULE ............................................................................................................... 87
   - 2.7. MENU MODULE .............................................................................................................................. 87
   - 2.8. ORDER MODULE ............................................................................................................................ 88
   - 2.9. PRODUCT MODULE ....................................................................................................................... 88
   - 2.10. SHOPPING CART MODULE .......................................................................................................... 89
   - 2.11. TEMPLATE MODULE .................................................................................................................... 90
   - 2.12. STATISTICS MODULE .................................................................................................................. 90

3. **LIBRARY TESTING** .............................................................................................................................. 91
   - 3.1. ERROR HANDLER ............................................................................................................................. 91
   - 3.2. BUILDURL ......................................................................................................................................... 91
   - 3.3. SECURITY .......................................................................................................................................... 92
   - 3.4. SESSION CONTROL ....................................................................................................................... 92
   - 3.5. UPLOADHANDLER ........................................................................................................................... 92
   - 3.6. SQL_LIB_10 .................................................................................................................................... 93

4. **CASE TESTING** ...................................................................................................................................... 94
   - 4.1. WIZARD CASE TESTING .................................................................................................................. 94
       * General User Interface Elements ......................................................................................................... 94
       * Article User Interface Elements ......................................................................................................... 95
       * Category User Interface Elements ...................................................................................................... 98
       * Customer User Interface Elements .................................................................................................... 98
       * Media User Interface Elements ........................................................................................................... 99
       * Membership User Interface Elements ................................................................................................. 100
       * Membership overview ....................................................................................................................... 100
       * Menu User Interface Elements ........................................................................................................... 101
       * Newsletter User Interface Elements .................................................................................................... 102
       * Order User Interface Elements .......................................................................................................... 103
       * Product User Interface Elements ........................................................................................................ 104
       * Template User Interface Elements ..................................................................................................... 105
   - 4.2. GENERATED WEBSITE CASE TESTING ....................................................................................... 106
       * Case Test for Menu and Category ......................................................................................................... 106
       * Various Pages User interface elements ............................................................................................... 106
       * Customer specific Interface Elements .............................................................................................. 107
       * E-Commerce site plus specific User Interface elements ...................................................................... 109
       * Media Site specific Interface Elements ............................................................................................ 110


1. Introduction

This document contains the test report for the website4free.co.nz project. Some modules were already tested during development, but not all of them. However as soon as the project was as good as finished we began focusing on testing the software. We began with testing the individual modules, specifically the models and the controllers. As stated in the main report the views are tested with case tests. This document will mainly focus on the results we obtained from the UT5 test environment. However we will also discuss the results we obtained from testing the case tests. After we finished testing the modules, we began testing the libraries, because they are an integral part of the system (besides the modules) and therefore required extensive testing.

Here we will explain the main structure on how the test classes for the modules are built. As stated in the main report a test class consists of set up phase, a testing phase and a tear down phase. Since the global use of these phases is already explained, here we will only discuss an example which is representative for all the tested modules. We will discuss an example in detail, because then the company SecureSpace, knows how the tests are set-up and how they can easily edit/add test cases.

Example TestArticleModule

For testing the article module, we first specified all the parameters an article could have, which are:

- wff_submenu_id,
- title,
- create_date,
- content,
- picture,
- published,
- and description.

In a test class we made these parameters global, so that the test methods could easily access them. If the values of the global parameters are changed all the tests should still pass, because they all access the same global variable. In the setup phase the test articles are inserted and in the tear down phase they are deleted from the database. A test method will generally look something like this:

```php
public function test_deleteArticle()
{
    $this->articlemodel->getArticleListByMenu($this->submenu_id_1);
    $this->assertTrue($this->articlemodel->getResultCount() === 1);

    $this->articlemodel->deleteArticle($this->article_1_id);
    $this->articlemodel->getArticleListByMenu($this->submenu_id_1);
    $this->assertTrue($this->articlemodel->getResultCount() === 0);
}
```

As one can see before testing the method deleteArticle, the pre-condition is tested, namely if the article we want to delete is actually there. So the resultcount should be one. After we executed the deleteArticle method the resultcount should be zero.
This example is representative for all other test cases done with the UT5 test environment. Next we will discuss the results we obtained from testing the modules, then we will discuss the results we obtained from testing the libraries and finally we will discuss the results we obtained from the case tests.
2. Module Testing

This part of the document specifies the results we obtained from testing the modules. The results are print outs from the UT5 test environment, which consist of the total number of tests per module and the duration of each test method. UT5 also gives an output specifying the details of every test, whether or not they have passes, we did not incorporate these results in the test rapport because the - Unit Test Statistics – table some them up in a much more efficient way. For testing the modules we encountered a couple of failed test, however they could be counted on one hand and most of the failed test consists of wrongly or not according to the standards formulated queries. We of course fixed these failed tests and we arrived at a total of 790 tests for the module testing.

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestArticleModel</td>
<td>test_getArticleListByMenu</td>
<td>0.28411 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getArticleListPublished</td>
<td>0.00644 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getArticleById</td>
<td>0.00471 sec.</td>
</tr>
<tr>
<td></td>
<td>test_editArticlePublished</td>
<td>0.00882 sec.</td>
</tr>
<tr>
<td></td>
<td>test_editArticleOnTop</td>
<td>0.00884 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteArticle</td>
<td>0.00662 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteArticlesByMenu</td>
<td>0.00548 sec.</td>
</tr>
<tr>
<td></td>
<td>test_editArticle</td>
<td>0.00461 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteArticlePicture</td>
<td>0.07514 sec.</td>
</tr>
<tr>
<td></td>
<td>test_insertArticlePicture</td>
<td>0.01149 sec.</td>
</tr>
<tr>
<td></td>
<td>test_addNewArticle</td>
<td>0.10104 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getStaticPages</td>
<td>0.01837 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getStaticPagesPublished</td>
<td>0.20394 sec.</td>
</tr>
<tr>
<td></td>
<td>test_editPagePublished</td>
<td>0.23758 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getPageById</td>
<td>0.26594 sec.</td>
</tr>
<tr>
<td></td>
<td>test_savePage</td>
<td>0.24036 sec.</td>
</tr>
<tr>
<td></td>
<td>test_FAQ</td>
<td>0.01125 sec.</td>
</tr>
</tbody>
</table>

2.1. Article Module
2.2. Category Module

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestCategoryModel</td>
<td>test_getAllCategories</td>
<td>0.00393 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getCategoryByID</td>
<td>0.00315 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getAllParentCategories</td>
<td>0.00256 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getAllChildren</td>
<td>0.00241 sec.</td>
</tr>
<tr>
<td></td>
<td>test_saveCategory</td>
<td>0.00538 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteCategory</td>
<td>0.0069 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getChildrenFromParent</td>
<td>0.0024 sec.</td>
</tr>
</tbody>
</table>

2.3. Customer Module

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestCustomerModel</td>
<td>test_fetchCustomer</td>
<td>0.00454 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getPayPaEmail</td>
<td>0.00286 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getCustomer</td>
<td>0.00367 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getCustomerbyID</td>
<td>0.00357 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getUser</td>
<td>0.00305 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getAllCustomers</td>
<td>0.00447 sec.</td>
</tr>
<tr>
<td></td>
<td>test_fetchSearch</td>
<td>0.00708 sec.</td>
</tr>
<tr>
<td></td>
<td>test_registerCustomer</td>
<td>0.00578 sec.</td>
</tr>
<tr>
<td></td>
<td>test_registerAdmin</td>
<td>0.00545 sec.</td>
</tr>
<tr>
<td></td>
<td>test_editCustomer</td>
<td>0.00481 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteCustomer</td>
<td>0.00473 sec.</td>
</tr>
<tr>
<td></td>
<td>test_customerExists</td>
<td>0.00362 sec.</td>
</tr>
<tr>
<td></td>
<td>test_customerExistsID</td>
<td>0.0032 sec.</td>
</tr>
<tr>
<td></td>
<td>test_accesslevelExists</td>
<td>0.00248 sec.</td>
</tr>
<tr>
<td></td>
<td>test_login_end_session</td>
<td>0.00385 sec.</td>
</tr>
<tr>
<td></td>
<td>test_login_get_info</td>
<td>0.00247 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getAccesslevels</td>
<td>0.00219 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getAccesslevel</td>
<td>0.00265 sec.</td>
</tr>
</tbody>
</table>
2.4. Gallery Module

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestGalleryModel</td>
<td>test_createCategory</td>
<td>0.1749 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getCategories</td>
<td>0.1646 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getImages</td>
<td>0.16653 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteImage</td>
<td>0.16644 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteCategory</td>
<td>0.16381 sec.</td>
</tr>
</tbody>
</table>

**- Unit Tests Statistics -**

<table>
<thead>
<tr>
<th>Number of tests run:</th>
<th>29</th>
<th>Number of errors:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tests passed:</td>
<td>29</td>
<td>Total test time:</td>
<td>0.83657 sec.</td>
</tr>
<tr>
<td>Number of tests failed:</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.5. Media Module

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestMediaModel</td>
<td>test_getAllMedia</td>
<td>0.006 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getMediaById</td>
<td>0.00337 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getAllMediaFromCategory</td>
<td>0.00508 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteMediaById</td>
<td>0.00622 sec.</td>
</tr>
<tr>
<td></td>
<td>test_setMediaActive</td>
<td>0.00479 sec.</td>
</tr>
<tr>
<td></td>
<td>test_setMediaInactive</td>
<td>0.00529 sec.</td>
</tr>
<tr>
<td></td>
<td>test_addMediaFile</td>
<td>0.01011 sec.</td>
</tr>
<tr>
<td></td>
<td>test_editMediaFile</td>
<td>0.00452 sec.</td>
</tr>
<tr>
<td></td>
<td>test_updateCounterFromMedia</td>
<td>0.00508 sec.</td>
</tr>
<tr>
<td></td>
<td>test_setMediaNotMembershipOnly</td>
<td>0.00331 sec.</td>
</tr>
<tr>
<td></td>
<td>test_setMediaMembershipOnly</td>
<td>0.00336 sec.</td>
</tr>
<tr>
<td></td>
<td>test_checkMembershipOnly</td>
<td>0.00282 sec.</td>
</tr>
</tbody>
</table>

**- Unit Tests Statistics -**

<table>
<thead>
<tr>
<th>Number of tests run:</th>
<th>76</th>
<th>Number of errors:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tests passed:</td>
<td>76</td>
<td>Total test time:</td>
<td>0.05994 sec.</td>
</tr>
<tr>
<td>Number of tests failed:</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.6. Membership Module

<table>
<thead>
<tr>
<th>- Unit Tests Statistics -</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tests run:</td>
</tr>
<tr>
<td>Number of tests passed:</td>
</tr>
<tr>
<td>Number of tests failed:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>- Unit Tests Execution Times -</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test class</td>
</tr>
<tr>
<td>--------------------------------</td>
</tr>
<tr>
<td>TestMembershipModel</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

2.7. Menu Module
2.8. Order Module

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestMenuModel</td>
<td>test_getAllMenus</td>
<td>0.00395 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getMenuById</td>
<td>0.00305 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getSubMenuById</td>
<td>0.0031 sec.</td>
</tr>
<tr>
<td></td>
<td>test_updateTopLevelMenuItem</td>
<td>0.00384 sec.</td>
</tr>
<tr>
<td></td>
<td>test_updateSubMenuItem</td>
<td>0.00403 sec.</td>
</tr>
<tr>
<td></td>
<td>test_saveTopLevelMenuItem</td>
<td>0.00466 sec.</td>
</tr>
<tr>
<td></td>
<td>test_saveSubMenuitem</td>
<td>0.00369 sec.</td>
</tr>
<tr>
<td></td>
<td>test_saveMenuItemOrder</td>
<td>0.00434 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteMenuItem</td>
<td>0.00503 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteSubMenuitem</td>
<td>0.00368 sec.</td>
</tr>
<tr>
<td></td>
<td>test_deleteSubMenuitemsFromParent</td>
<td>0.00365 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getParentMenus</td>
<td>0.00419 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getChildMenus</td>
<td>0.00365 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getAllChildrenFromParent</td>
<td>0.00312 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getAllChildrenFromParentOrdered</td>
<td>0.00385 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getAllParentsOrdered</td>
<td>0.00313 sec.</td>
</tr>
</tbody>
</table>

2.9. Product Module

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestOrderModel</td>
<td>test_getOrder</td>
<td>0.00504 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getEnumNames</td>
<td>0.00653 sec.</td>
</tr>
<tr>
<td></td>
<td>test_searchOrders</td>
<td>0.01055 sec.</td>
</tr>
<tr>
<td></td>
<td>test_orderProcess</td>
<td>0.00773 sec.</td>
</tr>
</tbody>
</table>
2.10. Shopping Cart Module

- Unit Tests Statistics -

<table>
<thead>
<tr>
<th>Test class</th>
<th>Number of tests run:</th>
<th>Number of errors:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestProductModel</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Number of tests passed:</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Total test time:</td>
<td>0.00176 sec.</td>
</tr>
<tr>
<td></td>
<td>Number of tests failed:</td>
<td>0</td>
</tr>
</tbody>
</table>

- Unit Tests Execution Times -

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestShoppingcartModel</td>
<td>test_shoppingCartFunctions</td>
<td>0.00176 sec.</td>
</tr>
</tbody>
</table>
2.11. Template Module

- Unit Tests Statistics -

| Number of tests run: | 9 |
| Number of tests passed: | 9 |
| Number of tests failed: | 0 |
| Number of errors: | 0 |
| Total test time: | 0.02138 sec. |

- Unit Tests Execution Times -

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestTemplateModel</td>
<td>test_getTemplatePreview</td>
<td>0.00511 sec.</td>
</tr>
<tr>
<td></td>
<td>test_countTemplates</td>
<td>0.00444 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getTemplates</td>
<td>0.00425 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getBasedir</td>
<td>0.00375 sec.</td>
</tr>
<tr>
<td></td>
<td>test_templateIsCustomized</td>
<td>0.00383 sec.</td>
</tr>
</tbody>
</table>

2.12. Statistics Module

- Unit Tests Execution Times -

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestStatisticsModel</td>
<td>test_getWeekTotal</td>
<td>0.00229 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getMonthTotal</td>
<td>0.00173 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getProductsInCategory</td>
<td>0.00183 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getTotalProfitPerDay</td>
<td>0.00169 sec.</td>
</tr>
<tr>
<td></td>
<td>test_addProductRecord</td>
<td>0.00086 sec.</td>
</tr>
</tbody>
</table>

- Unit Tests Statistics -

| Number of tests run: | 7 | Number of errors: | 0 |
| Number of tests passed: | 7 | Total test time: | 0.00841 sec. |
| Number of tests failed: | 0 |
3. Library Testing

This part of the document specifies the results we obtained from testing the various libraries. The library testing was a bit different than the module testing, since it did not require database entries to test the methods (except for the sql_lib_10). Since these libraries were developed since the beginning of the project, we did not find anything wrong with them. Mainly because of their extensive use throughout the development of the project, any errors were already fixed during development. One thing that jumps out, is the duration of testing the Security library, this is a couple of seconds because, one of the methods tested, checks if a given email actually exists, which takes a while to check. We arrived at the following total of 64 tests for the library testing.

3.1. Error Handler

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestErrorHandler</td>
<td>test_addError</td>
<td>0.00077 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getErrors</td>
<td>0.00023 sec.</td>
</tr>
<tr>
<td></td>
<td>test_displayErrors</td>
<td>0.00018 sec.</td>
</tr>
</tbody>
</table>

3.2. BuildURL

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestbuildURL</td>
<td>test_buildURL</td>
<td>0.00072 sec.</td>
</tr>
</tbody>
</table>
3.3. Security

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestSecurity</td>
<td>test_vcl</td>
<td>0.00092 sec.</td>
</tr>
<tr>
<td></td>
<td>test_vcl_array</td>
<td>0.00079 sec.</td>
</tr>
<tr>
<td></td>
<td>test_email_active_ch</td>
<td>0.00144 sec.</td>
</tr>
<tr>
<td></td>
<td>test_real_email_check</td>
<td>0.31067 sec.</td>
</tr>
<tr>
<td></td>
<td>test_real_email_check</td>
<td>2.94864 sec.</td>
</tr>
</tbody>
</table>

3.4. Session Control

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testsession_control</td>
<td>test_setCookieData</td>
<td>0.00072 sec.</td>
</tr>
<tr>
<td></td>
<td>test_getCookieData</td>
<td>0.00064 sec.</td>
</tr>
</tbody>
</table>

3.5. UploadHandler

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestuploadHandler</td>
<td>test_catchErrors</td>
<td>0.00114 sec.</td>
</tr>
<tr>
<td></td>
<td>test_vcl_in</td>
<td>0.00068 sec.</td>
</tr>
<tr>
<td></td>
<td>test_pic_upload</td>
<td>0.00057 sec.</td>
</tr>
<tr>
<td></td>
<td>test_dirList</td>
<td>0.00057 sec.</td>
</tr>
</tbody>
</table>
## 3.6. SQL_lib_10

### Unit Tests Statistics

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tests run</td>
<td>9</td>
</tr>
<tr>
<td>Number of errors</td>
<td>0</td>
</tr>
<tr>
<td>Number of tests passed</td>
<td>9</td>
</tr>
<tr>
<td>Total test time</td>
<td>0.00409 sec.</td>
</tr>
</tbody>
</table>

### Unit Tests Execution Times

<table>
<thead>
<tr>
<th>Test class</th>
<th>Test method</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testsq_lib_10</td>
<td>test_from_sql_obj</td>
<td>0.00237 sec.</td>
</tr>
<tr>
<td></td>
<td>test_sql</td>
<td>0.00172 sec.</td>
</tr>
</tbody>
</table>
4. Case Testing

To test the various forms and user input fields, case testing was used. This was necessary, because it is not possible to test user interfaces using unit tests. The tested values, expected results and results of our testing rounds can be found below. Any failed tests were fixed to pass on another try. To really understand some of the test cases, one must view the appropriate view (in the wizard or in a generated website), then it becomes more clear, what really was tested. We made a distinction between the test cases in the wizard and the test cases in the generated website.

4.1. Wizard case Testing

This part of the test rapport deals with all the test cases for the wizard. The wizard was thoroughly tested, because it is the main administration tool for a user.

General User Interface Elements

There are two user interface elements that are always available from anywhere in the wizard. The first one is the menu, which has to be built correctly according to the files in the root of the wizard. The second is the JavaScript help drop-down menu.

User interface menu:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on menu element to get that modules main page.</td>
<td>Redirect to page.</td>
<td>yes</td>
</tr>
<tr>
<td>Add module main page to wizard root.</td>
<td>Menu element added.</td>
<td>yes</td>
</tr>
<tr>
<td>Remove module main page from wizard root.</td>
<td>Menu element removed.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Drop down help menu:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open page without javascript.</td>
<td>All help entries visible.</td>
<td>yes</td>
</tr>
<tr>
<td>Open page with javascript.</td>
<td>All help entries hidden.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on 'Show Help' with JavaScript.</td>
<td>All help entry titles visible.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on 'Show Help' with JavaScript when all help entry titles visible.</td>
<td>All help entries hidden.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on help 'title' when all help entry titles visible.</td>
<td>Help entry with 'title' visible.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on help 'title' when entry with 'title' visible.</td>
<td>Help entry with 'title' hidden.</td>
<td>yes</td>
</tr>
</tbody>
</table>
Article User Interface Elements
The article user interface has several pages, each with several user inputs. For the main article page there is a distinction between the dynamic pages overview and the static pages overview, because they differ in the various options they have.

Article main screen:
Dynamic Pages

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>(if article not already published)</td>
<td>Article is published.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on checkbox under published.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(if article already published)</td>
<td>Article is not published.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on checkbox under published.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(if article already on top)</td>
<td>Article no longer on top</td>
<td>yes</td>
</tr>
<tr>
<td>Click on checkbox under on top.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(if article not already on top)</td>
<td>Article is on top.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on checkbox under on top.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Click on second article on top.</td>
<td>Second article is either on top or close to the top.</td>
<td>yes</td>
</tr>
<tr>
<td>Click any link under edit.</td>
<td>Popup opens with edit page.</td>
<td>yes</td>
</tr>
<tr>
<td>Click any link under delete (after declining action).</td>
<td>Article is not deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>Click any link under delete (after confirming action).</td>
<td>Article is deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>Click any link under preview.</td>
<td>Popup opens with a preview of the article.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on link delete all (after confirming action).</td>
<td>All articles are deleted under specific menu element.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on link delete all. (after declining action).</td>
<td>All articles are not deleted under specific menu element.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Static Pages

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>(if page already published)</td>
<td>Page is no longer published</td>
<td>yes</td>
</tr>
<tr>
<td>Click on checkbox under published.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(if page not already published)</td>
<td>Page is published</td>
<td>yes</td>
</tr>
<tr>
<td>Click on checkbox under published.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Click any link under edit.</td>
<td>Popup opens with edit screen</td>
<td>yes</td>
</tr>
</tbody>
</table>
### Menu Links under Article

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on ‘Ad New Page’.</td>
<td>Popup opens with ‘new page’ screen.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on ‘Preview Page List’.</td>
<td>‘Preview Page List’ screen is loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on ‘Manage Images’.</td>
<td>‘Manage Images’ screen is loaded.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Article new article page/ Edit article page

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in all the fields and presses save.</td>
<td>Article created/edited (visible in main article screen)</td>
<td>yes</td>
</tr>
<tr>
<td>User fills in all the fields and presses close.</td>
<td>Article not created/edited</td>
<td>yes</td>
</tr>
<tr>
<td>User does not fill in a certain field (image, title, description or content).</td>
<td>Article is created/edited without information from that field</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks right mouse button and selects ‘copy image location’, user then clicks image icon in the WYSIWYG editor and ‘pastes’ the link in the field.</td>
<td>Picture is added to the content of the article.</td>
<td>yes</td>
</tr>
<tr>
<td>Browse for picture and click upload.</td>
<td>Picture is uploaded.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Article preview page list:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on drop down menu and select different menu entry</td>
<td>Show different articles belonging to that menu entry.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on article title, image or read more link</td>
<td>Redirect to full article preview.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Article manage images:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse for image and click upload.</td>
<td>Image is uploaded.</td>
<td>yes</td>
</tr>
<tr>
<td>Click delete picture.</td>
<td>Picture is deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on picture.</td>
<td>Picture shows full size picture.</td>
<td>yes</td>
</tr>
<tr>
<td>Browse for incorrect (too large, missing) image and click upload.</td>
<td>Error message, no image uploaded.</td>
<td>yes</td>
</tr>
</tbody>
</table>
### Edit Contact Us Page

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in all the fields and presses ‘Save’.</td>
<td>Contact us page is edited.</td>
<td>yes</td>
</tr>
<tr>
<td>User fills in all the fields and presses ‘Close’.</td>
<td>Contact us page is not edited.</td>
<td>yes</td>
</tr>
<tr>
<td>User fails to fill in a field and presses ‘Save’.</td>
<td>Contact us page edited, without fields information.</td>
<td>yes</td>
</tr>
<tr>
<td>User fails to fill in a field and presses ‘Close’.</td>
<td>Contact us page not edited.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Edit About Us Page

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in all the fields and presses ‘Save’.</td>
<td>About us page is edited.</td>
<td>yes</td>
</tr>
<tr>
<td>User fills in all the fields and presses ‘Close’.</td>
<td>About us page is not edited.</td>
<td>yes</td>
</tr>
<tr>
<td>User fails to fill in a field and presses ‘Save’.</td>
<td>About us page edited, without fields information.</td>
<td>yes</td>
</tr>
<tr>
<td>User fails to fill in a field and presses ‘Close’.</td>
<td>About us page not edited.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Edit Home Page

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User uses the WYSIWYG editor, to specify contents of the homepage and presses ‘Save’.</td>
<td>Home page is edited.</td>
<td>yes</td>
</tr>
<tr>
<td>User uses the WYSIWYG editor, to specify contents of the homepage and presses ‘Close’.</td>
<td>Home page is not edited.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Edit Faq Page

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User presses the 'add new Faq' link</td>
<td>Popup opens with add new Faq screen.</td>
<td>yes</td>
</tr>
<tr>
<td>User presses the edit Faq link.</td>
<td>Popup opens with edit Faq screen.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on link delete (after confirmation action).</td>
<td>Faq deleted</td>
<td>yes</td>
</tr>
<tr>
<td>Click on link delete (after declining action).</td>
<td>Faq not deleted.</td>
<td>yes</td>
</tr>
</tbody>
</table>
Add/ Edit Faq

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in al the fields and presses ‘Save’.</td>
<td>Faq is edited/created.</td>
<td>yes</td>
</tr>
<tr>
<td>User fills in al the fields and presses ‘Close’.</td>
<td>Faq is not edited/created.</td>
<td>yes</td>
</tr>
<tr>
<td>User fails to fill in a field and presses ‘Save’.</td>
<td>Faq edited/created, without fields information.</td>
<td>yes</td>
</tr>
<tr>
<td>User fails to fill in a field and presses ‘Close’.</td>
<td>Faq not edited/created.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Category User Interface Elements

The category user interface has three pages, one overview page that shows all the categories with edit and add buttons, an edit page to edit a category and an add page to add a category.

Category overview:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on any edit button.</td>
<td>Popup opens with edit screen.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on link delete.</td>
<td>Category deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>(after confirming action)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Click on link delete.</td>
<td>Category not deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>(after declining action)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Click on 'add new category' in menu</td>
<td>Popup opens with add Category screen.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Add Category:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill in category data and click save.</td>
<td>Category is added.</td>
<td>yes</td>
</tr>
<tr>
<td>Fill in category data and click close.</td>
<td>No category is added.</td>
<td>yes</td>
</tr>
<tr>
<td>Do not fill in anything and click save.</td>
<td>Error message.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Edit Category:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change populated data and click save.</td>
<td>Category is edited.</td>
<td>yes</td>
</tr>
<tr>
<td>Change populated data and click close.</td>
<td>Category stays the same.</td>
<td>yes</td>
</tr>
<tr>
<td>Remove everything and click save.</td>
<td>Error message</td>
<td>yes</td>
</tr>
</tbody>
</table>

Customer User Interface Elements

The customer user interface has four pages, one search page that allows for searching customers to be displayed in a list overview page, a view customer page that displays a single
customer and an edit customer page that allows editing of existing customers. However, customer view has no actions.

### Customer search:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click search.</td>
<td>List with customers shows on same screen.</td>
<td>yes</td>
</tr>
<tr>
<td>Enter search term and click search.</td>
<td>List with customers subset is shown on same screen.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Customer list:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click column name.</td>
<td>List is sorted on that column name.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any view link.</td>
<td>Popup opens with customer view screen.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any edit link</td>
<td>Popup opens with customer edit screen.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any delete link. (after declining action)</td>
<td>Customer is not deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any delete link. (after confirming action)</td>
<td>Customer is deleted.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Edit customer:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change populated data and click save.</td>
<td>Customer is edited.</td>
<td>yes</td>
</tr>
<tr>
<td>Change populated data and click clear.</td>
<td>Original values are restored.</td>
<td>yes</td>
</tr>
<tr>
<td>Remove everything and click save.</td>
<td>Error message.</td>
<td>yes</td>
</tr>
<tr>
<td>Remove any required field and click save.</td>
<td>Error message, remembered fields.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Media User Interface Elements

The media user interface consist of three pages, namely the add and edit page and the overview page for all uploaded content.

### Media overview:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on any media picture.</td>
<td>Redirect to media example view.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any deactivate link.</td>
<td>Media is deactivated.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any activate link.</td>
<td>Media is activated.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any edit link</td>
<td>Popup opens with media edit screen.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any delete link.</td>
<td>Show confirmation popup.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any delete link.</td>
<td>Media is not deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>(after declining action)</td>
<td>Media is deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Click on any delete link.</td>
<td>(after confirming action) Media is membership only.</td>
<td>yes</td>
</tr>
<tr>
<td>(if media not already membership only)</td>
<td>Click on checkbox under membership only.</td>
<td>Media is membership only.</td>
</tr>
<tr>
<td>(if media already membership only)</td>
<td>Click on checkbox under membership only.</td>
<td>Media is no longer membership only.</td>
</tr>
<tr>
<td>Click in view statistics in media menu</td>
<td>Redirect to media statistics.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Add media:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill in media data and click submit.</td>
<td>Media is added.</td>
<td>no</td>
</tr>
<tr>
<td>Do not fill in anything and click save.</td>
<td>Error message.</td>
<td>yes</td>
</tr>
<tr>
<td>Do not fill in any required field and click save.</td>
<td>Error message, remembered fields.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Edit media:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change populated data and click submit.</td>
<td>Media is edited.</td>
<td>yes</td>
</tr>
<tr>
<td>Remove everything and click save.</td>
<td>Error message, remembered fields.</td>
<td>yes</td>
</tr>
<tr>
<td>Remove any required field and click save.</td>
<td>Error message, remembered fields.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Membership User Interface Elements**

The membership module consists of three pages, namely the membership overview page and the statistics page, which shows the statistics for the membership and the membership set-up page, which is used to set-up the various memberships.

**Membership overview**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on a activate membership link</td>
<td>Membership activated</td>
<td>yes</td>
</tr>
<tr>
<td>Click on a deactivate membership link, if membership active and not expired</td>
<td>Membership deactivated.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on a deactivate membership link, if membership expired.</td>
<td>Membership deactivated.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on ‘Show Statistics’ in</td>
<td>Redirect to membership stats view.</td>
<td>yes</td>
</tr>
</tbody>
</table>
membership menu

Setup Memberships view

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in the the amounts per period and presses Save</td>
<td>All the amounts have been saved.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Menu User Interface Elements

The menu user interface has three pages, an overview page that shows the menu items and allows for reordering, editing and deleting, an edit page to edit a menu element and an add page to add a new menu element.

Menu overview:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on order down for parent menu entry</td>
<td>Parent menu entry one order down</td>
<td>yes</td>
</tr>
<tr>
<td>Click on order up for parent menu entry</td>
<td>Parent menu entry one order up</td>
<td>yes</td>
</tr>
<tr>
<td>Click on order up for child menu entry</td>
<td>Child menu entry in parent menu entry one order up</td>
<td>yes</td>
</tr>
<tr>
<td>Click on order down for child menu entry</td>
<td>Child menu entry in parent menu entry one order down</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any edit link</td>
<td>Popup opens with edit menu screen.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any delete link. (after confirming action)</td>
<td>Menu element is deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any delete link. (after declining action)</td>
<td>Menu element is not deleted.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Add menu:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill in menu data and click submit.</td>
<td>Menu element is added.</td>
<td>yes</td>
</tr>
<tr>
<td>Do not fill in label and click save.</td>
<td>Error message</td>
<td>yes</td>
</tr>
</tbody>
</table>

Edit menu:

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change populated data and click submit.</td>
<td>Menu element is edited.</td>
<td>yes</td>
</tr>
<tr>
<td>Remove any required field and click save.</td>
<td>Error message</td>
<td>yes</td>
</tr>
</tbody>
</table>
**Newsletter User Interface Elements**

The newsletter user interface has four pages, the introduction page, recipients database page, edit newsletter page and manage campaigns page. These pages are in that order the steps that need to be taken to make a successful campaign.

**Newsletter introduction:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse image and click upload.</td>
<td>Image is replaced with new image.</td>
<td>yes</td>
</tr>
<tr>
<td>Click upload.</td>
<td>Nothing happens.</td>
<td>yes</td>
</tr>
<tr>
<td>Browse for incorrect image.</td>
<td>Error message, header not uploaded.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Manage database:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on bulk upload to subscribers database</td>
<td>Redirect to bulk uploaded page</td>
<td>yes</td>
</tr>
<tr>
<td>Click download CSV.</td>
<td>A CSV file with all the data.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on ‘by clicking this link’ to go to manage subscribers by hand</td>
<td>Popup opens with manage subscribers overview.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Bulk upload:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paste customers, fill in group.</td>
<td>Customers added to database in group.</td>
<td>yes</td>
</tr>
<tr>
<td>Upload csv with columns indicated, fill in group.</td>
<td>Customers added to database in group.</td>
<td>yes</td>
</tr>
<tr>
<td>Paste customers, do not fill in group.</td>
<td>Error message, remembered pasted data.</td>
<td>yes</td>
</tr>
<tr>
<td>Upload csv with columns indicated, do not fill in group.</td>
<td>Error message, remembered column data.</td>
<td>yes</td>
</tr>
<tr>
<td>Upload csv without e-mail column, but with other columns and group.</td>
<td>Error message, remembered data.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Manage Subscribers by hand:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on any delete link. (after confirming action)</td>
<td>Subscriber element is deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on any delete link. (after declining action)</td>
<td>Subscriber element is not deleted.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on Edit Link</td>
<td>PopUp opens with edit subscriber view.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Manage subscriber by hand:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in the fields and presses</td>
<td>Changes have been made</td>
<td>yes</td>
</tr>
</tbody>
</table>
User fills in the fields and presses **save** | Changes declined | yes
\[
\text{(if subscriber not already active)}  \\
\text{Click on checkbox after status.}  \\
\text{(if subscriber already active)}  \\
\text{Click on checkbox after status.}
\]

**Edit newsletter:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill in template name, from, subject and content, click accept.</td>
<td>Newsletter is saved as new template.</td>
<td>yes</td>
</tr>
<tr>
<td>Click preview after previous step.</td>
<td>Newsletter is shown without products.</td>
<td>yes</td>
</tr>
<tr>
<td>Click include product list, click accept</td>
<td>Newsletter is saved under current template.</td>
<td>yes</td>
</tr>
<tr>
<td>Click preview after previous step.</td>
<td>Newsletter is shown with products.</td>
<td>yes</td>
</tr>
<tr>
<td>Do not fill in one required field.</td>
<td>Error message, remembered other fields.</td>
<td>yes</td>
</tr>
<tr>
<td>Select a template.</td>
<td>Fields are populated with that template.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Manage campaigns:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill in test e-mail address, click send test.</td>
<td>Received test e-mail.</td>
<td>yes</td>
</tr>
<tr>
<td>Fill in no e-mail address, click send test.</td>
<td>Error message.</td>
<td>yes</td>
</tr>
<tr>
<td>Fill in incorrect e-mail address, click send test.</td>
<td>Error message, remembered entry.</td>
<td>yes</td>
</tr>
<tr>
<td>Select group, click test confirmed, click send all</td>
<td>Received actual e-mail</td>
<td>yes</td>
</tr>
<tr>
<td>Select no group, click test confirmed, click send all</td>
<td>Error message.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Order User Interface Elements**

The order user interface has two pages, the multiple order search and overview page and the single order search and overview page.

**Multiple order overview:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not fill in any search fields and click list orders.</td>
<td>All non-archived orders.</td>
<td>yes</td>
</tr>
<tr>
<td>Fill in some search fields and click</td>
<td>Subset of all non-archived orders.</td>
<td>yes</td>
</tr>
<tr>
<td>Case action</td>
<td>Expected result</td>
<td>Success</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------</td>
<td>---------</td>
</tr>
<tr>
<td>Enter existing id in search field.</td>
<td>Order is displayed.</td>
<td>yes</td>
</tr>
<tr>
<td>Enter non-existing id in search field.</td>
<td>Error message.</td>
<td>yes</td>
</tr>
<tr>
<td>Click print order.</td>
<td>Order is printed neatly.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Product User Interface Elements**

The product user interface has two pages, the multiple product search and overview page and the single product edit or add page.

**Multiple product overview:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not fill in any search fields and click search.</td>
<td>All products.</td>
<td>yes</td>
</tr>
<tr>
<td>Fill in some search fields and click search.</td>
<td>Subset of all products.</td>
<td>yes</td>
</tr>
<tr>
<td>Click on column name.</td>
<td>Products are ordered by that column name.</td>
<td>yes</td>
</tr>
<tr>
<td>Change product data, click make changes.</td>
<td>Product data is changed.</td>
<td>yes</td>
</tr>
<tr>
<td>Leave product data field empty, click make changes.</td>
<td>Error message.</td>
<td>yes</td>
</tr>
<tr>
<td>Click product name.</td>
<td>Display populated product edit popup.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Single product add:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case action</td>
<td>Expected result</td>
<td>Success</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Fill in all fields, click add product.</td>
<td>Product is added.</td>
<td>yes</td>
</tr>
<tr>
<td>Do not fill in required field, click add product.</td>
<td>Error message, remembered data.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Single product edit:**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change some fields, click edit product.</td>
<td>Product is edited.</td>
<td>yes</td>
</tr>
<tr>
<td>Remove required field, click edit product.</td>
<td>Error message, remembered data.</td>
<td>yes</td>
</tr>
<tr>
<td>Enter existing product ID, click search.</td>
<td>Edit product is populated.</td>
<td>yes</td>
</tr>
<tr>
<td>Enter non-existing product ID, click search.</td>
<td>Error message.</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Statistics User Interface Elements**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on next month</td>
<td>Jump to statistics next month</td>
<td>yes</td>
</tr>
<tr>
<td>Click on previous month</td>
<td>Jump to statistics previous month</td>
<td>yes</td>
</tr>
<tr>
<td>Click on back to current month</td>
<td>Jump to statistics current month</td>
<td>yes</td>
</tr>
<tr>
<td>Click on category</td>
<td>Jump to statistics of category</td>
<td>yes</td>
</tr>
<tr>
<td>Select a product</td>
<td>Jump to Statistics of product</td>
<td>yes</td>
</tr>
<tr>
<td>Click on Back</td>
<td>Back to main screen statistics</td>
<td>yes</td>
</tr>
<tr>
<td>Click on Profit Overview</td>
<td>Profit Overview visible</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Template User Interface Elements**

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on template picture</td>
<td>Generated site changed to template view</td>
<td>yes</td>
</tr>
</tbody>
</table>
4.2. Generated Website Case testing

This part of the test rapport specifies the test cases we ran in the generated website, we did come across a couple of bugs, but we managed to fix them in time.

Case Test for Menu and Category
The structure of the menu and the categories depends in the CSS provided for the design of a website. Since the menu and the categories use a predefined structure (consisting of `<ul>` and `<li>` tags), developing a case test for this is different for each website. However here we will discuss the most common ones, for an e-commerce website template. In these kinds of templates, the menu is usually horizontal, with drop downs on a mouse over (usually placed in the top centre of the page). The categories differ, but they are usually placed vertical and some work with drop downs on a mouse over and some do not. For the category contents, different content is loaded, depending on the kind of website someone has chosen, for a media website, the media files are loaded and for an e-commerce website the products are loaded.

Menu User Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User clicks on Home.</td>
<td>Home contents loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on Contact Us.</td>
<td>Contact Us content loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks in About Us.</td>
<td>About Us content loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on FAQ.</td>
<td>FAQ content loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on Gallery.</td>
<td>Gallery content loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on main menu item (more articles under main menu).</td>
<td>List with descriptions is loaded, (from all subcategories under main menu item).</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on main menu item (one article under main menu).</td>
<td>Complete article loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on sub menu item (one article under sub menu).</td>
<td>Complete article loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on sub menu item (more articles under sub menu).</td>
<td>List with descriptions is loaded, only from that sub menu item.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Category User Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User clicks on any subcategory.</td>
<td>Subcategory content loaded.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Various Pages User Interface Elements

Here we will discuss the various options users have on different parts of the generated website. We will start with the static pages, then the articles. After that we will discuss the register page and all the options associated once a user is registered. After that we will discuss the various options a user has with an ecommerce site and then with a media site. For the ecommerce site we will focus on ordering a product and for the media site, we will focus on the use of a membership.
## Contact Us Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in at least the required fields and presses submit (all data correct).</td>
<td>Mail sent to admin and to user itself containing the message.</td>
<td>yes</td>
</tr>
<tr>
<td>User fills in at least the required fields and presses submit (not all data correct).</td>
<td>User informed of wrong data (corrects and must fill in new verification code)</td>
<td>yes</td>
</tr>
</tbody>
</table>

## FAQ Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User clicks on a specific question.</td>
<td>User is redirected on the same page to question and answer.</td>
<td>yes</td>
</tr>
</tbody>
</table>

## Gallery User Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User clicks on specific gallery.</td>
<td>Gallery content loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on specific picture.</td>
<td>Slideshow started with specific picture.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on play, pause, previous or next.</td>
<td>Appropriate action is loaded.</td>
<td>yes</td>
</tr>
</tbody>
</table>

## List of articles Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User clicks on main picture of article</td>
<td>Complete Article loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on read more button.</td>
<td>Complete Article loaded.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks on title of article</td>
<td>Complete Article loaded.</td>
<td>yes</td>
</tr>
</tbody>
</table>

## Customer specific Interface Elements

Under customer specific interface Elements, we understand the action like register, edit details and login and logout.

## Register Page Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in all fields correct and presses submit</td>
<td>User is registered and can now login</td>
<td>yes</td>
</tr>
<tr>
<td>User fills in some fields incorrect and presses submit</td>
<td>Error message displayed</td>
<td>yes</td>
</tr>
</tbody>
</table>

## Login Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in username and password</td>
<td>User is logged in</td>
<td>yes</td>
</tr>
</tbody>
</table>
### Edit Your Details Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User changes details correctly and presses Save</td>
<td>User details have been changed</td>
<td>yes</td>
</tr>
<tr>
<td>User changes details incorrectly and presses Save</td>
<td>User receives error message</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Logout Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User presses logout button</td>
<td>User is logged out</td>
<td>yes</td>
</tr>
</tbody>
</table>

### E-commerce site specific Interface Elements

There is distinction between the e-commerce site and the e-commerce site plus, the plus site has a complete shopping cart system and the ‘normal’ e-commerce only has a buy now button which proceeds directly to paypal.

### Small product search

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in a keyword</td>
<td>Products which have anything to do with keyword are displayed.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Advanced product search

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in fields according to his specifications.</td>
<td>Products according to search criteria are displayed.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Small Product View Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User presses add to cart button.</td>
<td>Shopping cart view loaded and shopping cart updated.</td>
<td>yes</td>
</tr>
<tr>
<td>User presses view details button.</td>
<td>User is redirected to Product View Page.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Product View Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User presses add to cart button.</td>
<td>Shopping cart view loaded and shopping cart updated.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Order now Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User presses the order now button.</td>
<td>User is redirected to paypal page.</td>
<td>yes</td>
</tr>
</tbody>
</table>
E-Commerce site plus specific User Interface elements

As stated before, the e-commerce plus website has a shopping cart module, with a complete order module behind it. The test cases in this section are therefore specific for these modules.

### Goto Shoppingcart Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User clicks in shopping cart logo.</td>
<td>If JavaScript enabled, Popup loaded else shopping cart loaded in main screen.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Shopping cart Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User changes number of items and presses ‘Update Quantities’</td>
<td>Quantities (including prices) are updated according to user specifications</td>
<td>yes</td>
</tr>
<tr>
<td>User presses ‘Clear shopping cart’</td>
<td>The shopping cart if the user is emptied.</td>
<td>yes</td>
</tr>
<tr>
<td>User presses order now</td>
<td>User is redirected to order page.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks in shopping cart logo.</td>
<td>If JavaScript enabled, Popup loaded else shopping cart loaded in main screen.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Order Screen Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User fills in username and password and presses ‘Get Details’.</td>
<td>Details of the user are filled in the fields.</td>
<td>yes</td>
</tr>
<tr>
<td>User fills in manually all details clicks paypal payment method and presses ‘Confirm Order’.</td>
<td>If details provided are missing or wrong format user is notified else user is redirected to order overview page</td>
<td>yes</td>
</tr>
<tr>
<td>User presses ‘back to shopping cart’</td>
<td>User is redirected to shopping cart overview page.</td>
<td>yes</td>
</tr>
<tr>
<td>User clicks in shopping cart logo.</td>
<td>If JavaScript enabled, Popup loaded else shopping cart loaded in main screen.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Order Overview Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User presses the ‘Proceed to paypal’ button.</td>
<td>User is redirected to pay pal page.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Payment with Pay pal completed

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Once the payment with paypal is completed the user is redirected to the thank you page of the generated website.</td>
<td>User is in the thank you page of the generated website.</td>
<td>yes</td>
</tr>
</tbody>
</table>
Media Site specific Interface Elements

This section specifies the various actions a user has on a media website. The most important thing for a media website is the membership only content. Once a user has registered on a media website, it automatically has a membership, however he still needs to activate it by going to PayPal and paying the appropriate amount for the membership.

### Edit membership details Interface Elements

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User chooses membership duration and presses submit</td>
<td>User is redirected to PayPal.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### View media Interface details

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User presses subcategory</td>
<td>User sees list of available media</td>
<td>yes</td>
</tr>
<tr>
<td>User selects a media file (membership only)</td>
<td>If user has active membership he can view the content, otherwise he can not.</td>
<td>yes</td>
</tr>
<tr>
<td>User selects a media file (not membership only)</td>
<td>User can view the media file.</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Payment with PayPal completed

<table>
<thead>
<tr>
<th>Case action</th>
<th>Expected result</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>User now has a membership for the chosen duration.</td>
<td>User can view all membership only content.</td>
<td>yes</td>
</tr>
</tbody>
</table>
Appendix C: Style a generated site for Dummies
Style a generated site for Dummies

Version 1.0
July 2007

Project Group
Remy Alidarso   1221736
Zdravko Anguelov  1232436
Terence Bahlen   1217313
Frits de Nijs    1224263

Company
SecureSpace
Business Manager:   Maria Veleva
Project Manager:    Dimitar Velev

University
Delft, University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science

Commission
Project Manager:    Ir. B.R. Sodoyer
Project Accompanier: J. de Vries
# Table of Contents

1. INTRODUCTION................................................................................................................ 114

2. GENERATED SITE STRUCTURE............................................................................................ 115
   2.1. STRUCTURES .................................................................................................................. 116
      2.1.1 General Structure ...................................................................................................... 116
      2.1.2 Main Content Structure ............................................................................................ 117
   2.2. PREDEFINED PAGES ........................................................................................................... 126

APPENDIXES ........................................................................................................................................ 128
   APPENDIX A: EXAMPLE TEMPLATE ........................................................................................ 128
1. Introduction

This document will discuss how a template can be made for a generated website. There are a couple of things a designer needs to know to make a good design. Since the structure of a generated site is predefined, we added id or classes in certain views, so the designer can access them and put them in the CSS file to style that view.

We will mention it more in this document, but if the views of a module are changed or something else inside the code, be sure to set the website to customized in the admin tool of the webstite4free.co.nz. Then if someone wants to copy the newest sources to the site, this will not happen, because than your changes are not lost.
2. Generated site structure

For a designer to make a good template for the websiteforfree.co.nz project there are a few basic things he needs to know before he can make a good design for the project websiteforfree.co.nz. Let's start with the index.php. A non styled index.php will look like this:

```php
<?php include_once('lib/index_includes.php'); ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
    <meta http-equiv="content-type" content="application/xhtml+xml; charset=iso-8859-1" />
    <title></title>
    <link rel="stylesheet" type="text/css" href="css/style_template.css" />
    <script type="text/JavaScript" src="js/pop_v99.js"></script>
    <script type="text/JavaScript" src="js/fixLinks.js"></script>
</head>
<body onload="fixLinks()">
    <?php require_once('lib/includes/display_goto_wizard.php'); ?>
    <?php require_once('lib/includes/display_menu.php'); ?>
    <?php require_once('lib/includes/display_categories.php'); ?>
    <?php require_once('lib/includes/frontpageproducts.php'); ?>
    <?php require_once('lib/includes/customer_options.php'); ?>
    <?php require_once('lib/includes/productsimplesearch.php'); ?>
    <?php require_once('lib/includes/shoppingcartminiview.php'); ?>
    <?php require_once('lib/includes/main_content.php'); ?>
</body>
</html>
```

The first php tags (displayed in red) must always be there, because it includes essentials parts for the overall working of the index.php.

The other php tags (displayed in blue) include the different parts of the content of the index.php. In short what a designer must do is place around these php tags the CSS to style the template. What will be included by the php tags will be discussed in paragraph 2.1. In paragraph 2.1 we will also discuss how one can style certain content of the generated site with id’s already given. The last of the php tags includes all the content which should be loaded into the centre of the page (the main content). Since it specifies the main content it much be placed in such a way that it shows the main content in the right place on the site.

Some predefined pages, like contact us form and register contain inline CSS, so those are not editable to a certain level. However these pages are made in such a way that they look good in every site. It is possible to style these pages, if one goes to the appropriate module and opens modulenameView.php and edits the appropriate view. If one styles a website that way, do not forgot to go to the admin tool of the website4free.co.nz main page and set the page to customized. Then the modules will not be overwritten with new code, hence you will not loose your styled view.

Appendix A contains a complete template; we refer the reader to take a good look at that appendix, because than the overview on how to create a template will become much clearer.
2.1. Structures

This part of the document will explain the several CSS structures which are crucial for a good template design. The first thing a designer needs to know is that the CSS file should be named `style_template.css` and be placed in the `css` directory, because than the wizard knows were to look for the CSS file. We will discuss all the php tags given in the introduction and we will discuss what the content is for certain and what id’s are given to style that content.

2.1.1 General Structure

This part of the document will specify the general things which will be included into a template.

The go to wizard button

The go to wizard button is displayed on the generated site if an admin is logged in. This button makes it easy for the admin to switch between the view of the wizard (where he can specify the content of his generated site) and the view of the generated site (where he sees his content displayed).

The following php tag will include this in the design of the template: (remember it will only be visible if an admin of the generated site is logged in)

```php
<?php require_once('lib/includes/display_goto_wizard.php'); ?>
```

The structure of the go to wizard button is defined as below; we also have defined the CSS below, so the designer does not have to design that for every generated site, but can simply copy paste it into the design.

**Structure**

```html
<div class="preview">
  
  <p>
    <a href="wizard/index.php?module=home">Wizard</a>
  
</p>
</div>
```

**CSS**

```css
/* -- For the Back to wizard button if logged-in */
div.preview{
  width: 120px;
  height: 20px;
  background-color: #0076cd;
  border: 5px solid #c91805;
  position:absolute;
  float:right;
top:0em;
right:0em;
font-size:1em;
z-index:7;
/* for IE */
filter:alpha(opacity=40);
/* CSS3 standard */
opacity:0.4;
/* for Mozilla */
-moz-opacity:0.6;
}
div.preview p{
  margin: 5px 25px;
}```
The Menu Structure
The menu of a generated site can be included with the following php tag:

```php
```

This will include a structure as defined below (which can be shaped into any form with the appropriate CSS by means of the id menu list).

**Structure**

```
<ul id="menulist">
  <li><a href='index.php?spage=1'>Home</a></li>
  <li><a href='index.php?page=3716'>test</a>
      <ul>
        <li><a href='index.php?page=3716&sub=3687'>wetgweywery</a></li>
        <li><a href='index.php?page=3715'>kjgisdgsdg</a></li>
      </ul>
    </li>
  <li><a href='index.php?page=57'>Top Level Menu Item</a>
      <ul>
        <li><a href='index.php?page=57&sub=51'>kaas</a></li>
      </ul>
    </li>
</ul>
```

The Category Structure
The category of a generated site has the same structure as the menu of a generated site and it has the id category list to style it. It will be included into the generated site, if the generated site is an e-commerce site or a media site. The php tag which will include the category structure is

```php
```

**Structure**

```
<ul id="categorylist">
  <li>
    <a href=''>Testcategory</a>
    <ul>
      <li><a href='/index.php?category=2'>Subcategory One</a></li>
      <li><a href='/index.php?category=3'>Subcategory Two</a></li>
    </ul>
  </li>
</ul>
```

2.1.2 Main Content Structure

This part of the document will specify what the main content of can be and how this can be styled. This will be discussed per module, that has content on the generated site. We will not discuss the predefined pages here (those will be in 2.1.3), except for the contact us page.

**Article Module: article list and article single view (Dynamic pages)**
The article module has two views for a generated website, namely an article list, shows the description of all article belonging to a certain menu entry and a single article view. Below is the structure given of both these views and there once can also see the id’s one requires to style these looks with the CSS.

Structure article list
<table id="article_list">
<table id="article_list_single">
<tr>
<td colspan="2">
<a href="index.php?page_id=1" >Test Article 1</a>
</td>
</tr>
<tr>
<td>
<a href="/index.php?page_id=1" >
<img src="images.jpeg" alt="picture_1" width="100px"/>
</a>
</td>
<td >Lorem ipsum dolor sit </td>
</tr>
<tr>
<td colspan="2">
<a href="index.php" id="read_more_button">read more</a>
</td>
</tr>
</table>
</table>
.<.
. (repeating table with id article_list_single)
.
.</table>

Structure single article view
<table id="display_article">
<tr>
<td><h3>Test Article 1</h3></td>
</tr>
<tr>
<td>&nbsp;</td>
</tr>
<tr>
<td>
<img id="Test Article 1" src="images.jpeg" alt="Article" width="150px"/>
</td>
</tr>
<tr>
<td>&nbsp;</td>
</tr>
<tr>
<td>article content</td>
</tr>
</table>

Article Module: static pages
Here we will discuss the various static pages a generated site can have.
The Home Page
Since the homepage is being generated from the content provided by the user, with help from the WYSIWYG editor, there is nothing one can do qua styling for the homepage. This is entirely up to the user, he can make it just the way he wants it.

The Contact Us Page
The Contact Us Page consists of two parts. The first part is the part which is retrieved from the database and contains the details filled in by the user. This structure of this part will look something like this:

Structure
<table id="contact_us">
  <tr>
    <td>Postal Address:</td>
    <td>Lorem ipsum dolor sit amet</td>
  </tr>
  <tr>
    <td>Phone:</td>
    <td>0000000000</td>
  </tr>
  <tr>
    <td>Fax</td>
    <td>0000000000</td>
  </tr>
  <tr>
    <td>Address:</td>
    <td>Lorem ipsum dolor sit amet</td>
  </tr>
</table>

The second part of the contact us page consists of the contact us form. This file can be found in lib/contact_us.php. The style of this form is done with inline CSS. This page is made such that it will look good in very site, however it can be changed by going to the contact_us.php and changing the look.

The FAQ Page
The FAQ Page consists of questions and answers. On top of the page are all the questions as links to the questions and answer on the bottom of the page. This page is editable by defining the look of a link <a class="faq" href="#..." />. For the rest the FAQ consist of text, so there is not much to style.

The About Us Page
The About Us page consists of two questions and is just a simple page, to let a company introduce itself. This page only consists of text and therefore there is not much to style on this page. This page is kept simple because users of the generated website are pushed to make their own about us page.

The Gallery Page
The gallery page consists of four views:
- the category overview
- all the images in a category
- the slideshow
- and the single image view.
For the slideshow and the single image view, there is nothing to style. However for the category overview can be style with the id’s given in the structure.

Structure gallery overview

```html
<p id="gallery_overview_header">Category Overview</p>
<table id="gallery_overview">
<tr>
  <td>
    <table>
      <tr>
        <td>
          <a href="index.php?galleryfolder=Supra_new">
            <img src="065_dside.jpg" />
          </a>
        </td>
        <td colspan="1">
          <a href="index.php?galleryfolder=Supra_new">Supra_new</a>
        </td>
      </tr>
    </table>
  </td>
  <td>&nbsp;</td>
</tr>
</table>
```

The overview of all the pictures in a gallery can be styled with the id’s in the following structure:

Structure all the pictures in a gallery

```html
<p id="gallery_images_header">Images in category Supra_new</p>
<table id="gallery_images">
<tr>
  <td>
    <a href="index.php?slideshowfolder=Supra&slideshowimage=065.jpg">
      <img src="065.jpg" />
    </a>
  </td>
  <td>
    <a href="index.php?slideshowfolder=Supra&slideshowimage=060.jpg">
      <img src="060.jpg" />
    </a>
  </td>
  <td>
    <a href="index.php?slideshowfolder=Supra&slideshowimage=055.jpg">
      <img src="modules/gallery_module/images/Supra_new/thumb/055.jpg" />
    </a>
  </td>
  <td>
    <a href="index.php?slideshowfolder=Supra&slideshowimage=050.jpg">
      <img src="modules/gallery_module/images/Supra_new/thumb/050_rds.jpg" />
    </a>
  </td>
</tr>
</table>
```
Customer Module: options
The customers options consist of the following parts
- edit your details link,
- register link,
- welcome message if logged in,
- error message if not able to log in,
- and the membership details link.
These can be styled by simply styling the links in a CSS file. If one however wants this to be different go to the lib/includes/customer_options.php, there you can specify how you want this to look. The php tag which will include the customer options is:

```php
```

Product Module
The product module has several views in the generated website, here we will specify how to style these looks.

Product Module: products of front page (specials)
The following php tag will include the products a user of the wizard has specified as specials and wants them displayed on the front page:

```php
<?php require_once('lib/includes/display_goto_wizard.php'); ?>
```

Structure: products on frontpage (specials)
```html
<div id="front_page_products">
  <div class="front_page_product">
    <table cellspacing="10" cellpadding="0">
      <tr>
        <td style="width: 30%;" rowspan="4">
          <a href="index.php?id=1">
            <img style="width: 100%;" src="button1.jpg" alt=""
          </a>
        </td>
        <td style="font-weight: bold;" align="right">
          Name
        </td>
        <td>
          <a href="index.php?id=1">Product One</a>
        </td>
      </tr>
      <tr>
        <td style="font-weight: bold;" align="right">
          Price
        </td>
        <td>119.00</td>
      </tr>
    </table>
  </div>
</div>
```
Product Module: single product view
For a single product view one can use the id single product to change the look with only CSS. As one can see below the single product view is a table and it has inline CSS. Therefore if the site needs to be customized one can change this view in the product module by going to the ProductView.php and changing the appropriate view.

Structure: single product view
<table style="width: 100%;" cellspacing="0" cellpadding="0" id="single_product">
  <tr>
    <td style="width: 40%;">  
      <img style="width: 100%;" src="button1.jpg" alt="Product Preview" />
    </td>
    <td style="width: 60%;" style="padding-right: 10px;">  
      <table style="width: 100%;" cellspacing="0" cellpadding="0">
        <tr>
          <td style="font-weight: bold;" align="right">Name</td>
          <td align="left">Product One</td>
        </tr>
        <tr>
          <td style="font-weight: bold;" align="right">Price</td>
          <td align="left">119.00</td>
        </tr>
        <tr>
          <td style="font-weight: bold;" align="right">Available</td>
          <td align="left">Yes</td>
        </tr>
      </table>
    </td>
  </tr>
</table>
Product Module: product simple search
The product simple search view can be included in the design with the following PHP tag:
```php
```
The structure of the product simple view is as follows:

Structure simple product search
```html
<form action="index.php" method="get">
<table id="product_simple_search_form">
<tr>
<td><input name="keyword" type="text" value="" /></td>
<td><input type="submit" value="Search" /></td>
</tr>
<tr>
<td colspan="2"><a href="?advanced_search=yes">Advanced Search</a></td>
</tr>
</table>
</form>
```

Product Module: product advanced search
The advanced search of a product is defined with inline CSS, because then all the input fields are equal and properly aligned. It can be styled with CSS, by using the id advanced search.

Structure product advanced search
```html
<form action="index.php" method="get">
<h1>Search Products</h1>
<table style="width: 60%; margin-left: auto; margin-right: auto;"
cellspacing="0" cellpadding="3" id="advanced_search">
<tr>
<td style="width: 40%;" align="right">Keyword</td>
<td style="width: 60%;"><input style="width: 97%;" name="keyword" type="text" value="" /></td>
</tr>
</table>
</form>
```
### Product Module: product search results / product overview

When a customer searches for products or clicks on a subcategory, then the products from the search or the subcategory are displayed in groups of four. On the bottom of this group of four are the page numbers for the products, if the search or the subcategory contains more than four products than the user can click there to show the rest of the products. These groups of four can be styled with the id `product_search_results` and with the class `product_box`.

**Structure group of four products**

```html
<table id="product_search_results" cellspacing="0" cellpadding="0">
    <tr>
        <td style="width: 50%;"
            class="product_box" cellspacing="0" cellpadding="0">
            <tr>
                <td style="width: 40%;" rowspan="4">
                    <a href="index.php?id=1">
                        <img style="width: 100%;" src="/button1.jpg" alt="" />
                    </a>
                </td>
                <td style="width: 100%;"
                    name="category">
                    <select style="width: 100%;" name="category">
                        <option value="all">Any Category</option>
                        <option value="2">Test Subcategory One</option>
                        <option value="3">Test Subcategory Two</option>
                    </select>
                </td>
                <td style="width: 60%;"
                    name="minprice" type="text" value="">
                    <input style="width: 97%;" name="minprice" type="text" value=""
                </td>
                <td style="width: 60%;"
                    name="maxprice" type="text" value="">
                    <input style="width: 97%;" name="maxprice" type="text" value=""/>
                </td>
                <td style="width: 60%;"
                    name="order">
                    <select style="width: 100%;" name="order">
                        <option value="">Select one</option>
                        <option value="down">Highest price first</option>
                        <option value="up">Lowest price first</option>
                    </select>
                </td>
                <td colspan="2" align="right">
                    <input type="submit" value="Search"/>
                </td>
            </tr>
        </table>
    </tr>
</table>
```
<td style="font-weight: bold;" align="right">Name</td>
<td>Product One</td>
<td style="font-weight: bold;" align="right">Price</td>
<td>119.00</td>
</tr>
<tr>
<td colspan="2" align="center">
<a href="index.php?id=1">View details</a>
</td>
</tr>
<tr>
<td colspan="2" align="center">
<a href="index.php?cart=1&id=1" onclick="PopW('lib/shoppingcartoverview.php?cart=1&id=1&amp;js=true','Shoppingcart Overview', '500', '400', 'no'); return false;">Add to cart</a>
</td>
</tr>
</table>
.
.
.
(3 more products)
</td>
</tr>
<tr>
<td colspan="2">1</td>
</tr>
</table>
</td>
</tr>
</table>

**Media Module: media overview**

If the generated website is a media site, then if a user clicks on a category it will display all the media in that category. If one wants to style that particular view, one can use the id media list as one can see below.

**Structure media overview**

```html
<table id="medialist">
<tr>
<th>Preview</th>
<th>Title</th>
<th>Description</th>
</tr>
<tr>
<td><a href="/index.php?mid=1">"Test Media File"<img src="thumbs/" alt="Test Media File" width="100" style="border: 1px solid #000000;" /></a></td>
<td>Test Media File</td>
<td>This is only a dummy media file</td>
</tr>
</table>
```

**Media Module: single media view**
For the Single media view, there is nothing to style, because this only shows the flash player in the centre of the screen.

**Shopping Cart Module: shopping cart mini view**
If the generated website is an e-commerce plus website, then the shopping cart will be available for all the visitors to the website. This site always has a shopping cart mini view, which will display the current number of items in the shopping cart and the total price. This part can be included in the main website with

```php
```

*Structure shopping cart mini view*

```html
<table id="shopping_cart_mini_view">
  <tr>
    <td>
      <a href="index.php?cart=1" class = "popup_link2">
        <img src="cart.jpg" alt="shoppingcart" />
      </a>
    </td>
    <td>
      Your shoppingcart is currently empty
    </td>
  </tr>
</table>
```

As one can see this can be styled by means of the id shopping_cart_mini_view. If one does not want to show the image cart.jpg, one can use display:none for that picture and than your own picture can be displayed their.

### 2.2. Predefined pages

For a generated website there are a couple of predefined pages. In paragraph 2.1 we already discusses the contact us page, here we will discuss the rest of the predefined pages. We will start with predefined pages for the customer, then for the e-commerce site plus and then for the media site.

**Customer Module: register and edit your details page.**
In the customer module and then specifically the CustomerView.php file contains the views for the register page and for the edit your details page. These pages are styled with inline CSS, al input fields equals and everything aligned. These pages can be styled, but then one needs to change the CustomerView.php files, however we doubt if this is needed because these pages are very basic and only contains input fields and some text.

**Shopping Cart Module: shopping cart overview**
The shopping cart overview page does not have an id's to style it; however there is something special with this page. If the browser has JavaScript enabled than this view will be a popup, if not this view will be displayed in the centre of the page. If one does want to change this view, because the site needs to be customized, once can go to the shopping cart module and find the appropriate view in the ShoppingcartView.php.

**Order Module: order overview**
For the order overview, the same holds as for the shopping cart overview, concerning the JavaScript. If one wants to change this view, one can go to the order module, to the OrderView.php and find the appropriate view.
Membership module: membership details
The membership details page consist of some text and a few buttons to go to pay pal and pay for a membership. This view can be found in the membership module in the MembershipView.php and there one can find the appropriate view if it needs to be changed.
Appendixes

Appendix A: example template

```php
<?php
include_once('lib/index_includes.php');
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<meta http-equiv="content-type" content="application/xhtml+xml; charset=iso-8859-1" />
<title></title>
<link rel="stylesheet" type="text/css" href="css/style_template.css" />
<script type="text/JavaScript" src="js/pop_v99.js"></script>
<script type="text/JavaScript" src="js/fixLinks.js"></script>
</head>
<body onload="fixLinks()">
<div id="background">
<table id="main_table" cellspacing="0" cellpadding="0">
<tr>
<td id="header">
<table class="fullwidth" cellspacing="0" cellpadding="0">
<tr>
<td rowspan="2" align="left" valign="top">
<img id="logo" src="images/logo.gif" alt="Sport, the biggest choice"
width="282" height="140" />
</td>
<td align="right" valign="top">
<p class="top_links">
<?php if ($productcontroller){ ?><a href="index.php?advanced_search=yes">Advanced Search</a>
<?php}if ($customercontroller && !$logincontroller->loggedin()){?>
<a href="index.php?register=1">Register</a>
<?php } else if ($customercontroller && $logincontroller->loggedin()){?>
<a href="index.php?edit=1">Edit Details</a>
<?php}?></p>
</td>
</tr>
<tr>
<td align="right" valign="bottom">
<?php require_once('lib/includes/shoppingcartminiview.php'); ?></td>
</tr>
<tr>
<td colspan="2">
<table id="main_navigation" cellspacing="0" cellpadding="0">
<tr>
<td class="white_left">&nbsp;</td>
<td class="bluebackground" align="center" valign="bottom">
<table id="nav_container">
<tr>
<td><?php require_once('lib/includes/display_menu.php'); ?></td>
</tr>
</table>
</td>
<td class="white_right">&nbsp;</td>
</tr>
<tr>
<td class="white_round_left">&nbsp;</td>
<td class="white_bottom">&nbsp;</td>
<td class="white_round_right">&nbsp;</td>
</tr>
</table>
</td>
</tr>
</table>
</td>
</tr>
</table>
</div>
</body>
</html>
```
www.website4free.co.nz

Bachelor project IN3700

CLIENT MANUAL

Version 1.5
July 2007

Project Group
Remy Alidarso          1221736
Zdravko Anguelov      1232436
Terence Bahlen         1217313
Frits de Nijs          1224263

Company
SecureSpace
Business Manager: Maria Veleva
Project Manager: Dimitar Velev

University
Delft, University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science

Commission
Project Manager: Ir. B.R. Sodoyer
Project Accompanier: J. de Vries
# Table of Contents

1. INTRODUCTION......................................................................................................................... 134

2. CLIENT MANUAL QUESTIONS AND ANSWERS ........................................................................ 135
   2.1. ARTICLE MODULE............................................................................................................. 135
   2.2. CATEGORY MODULE........................................................................................................ 137
   2.3. CUSTOMER MODULE....................................................................................................... 137
   2.4. GALLERY MODULE.......................................................................................................... 138
   2.5. MEDIA MODULE............................................................................................................... 138
   2.6. MEMBERSHIP MODULE................................................................................................. 139
   2.7. MENU MODULE.............................................................................................................. 140
   2.8. NEWSLETTER MODULE................................................................................................. 140
   2.9. ORDER MODULE............................................................................................................ 141
   2.10. PRODUCT MODULE...................................................................................................... 142
   2.11. TEMPLATE MODULE.................................................................................................... 143
   2.12. STATISTICS MODULE.................................................................................................. 143
1. Introduction

This document contains the client manual for the project websiteforfree.co.nz. This document consists of the questions and the answers that can be found in the Wizard. The questions can be found in the centre on top of every page belonging to a module. We made this document because it can also be handy for the developers at SecureSpace, since the basic use of every module is explained in this document. This document is fairly simple; however it makes the overview of the entire project and especially the wizard clearer to user, developers and other interested people. As with many of the documents, this document will discuss all the modules separately.
2. Client manual Questions and Answers

This section of the document consists of questions and the answers, as can be found in the help of the wizard. The questions and answers give a more detailed insight into the working of the wizard.

2.1. Article Module

For what are the overall functions of the articles?

The overall function consist of
- ‘add new page’, with which you can add a new page,
- ‘preview page list’, through which you can see how your page list looks on your main site,
- and ‘manage images’ through which you can manage your pictures for your home page and your dynamic pages content.

How can I make a new article?
To make a new article simply press the ‘add new page’ and fill in the content for your page. For more details on how to fill in the content we refer to 'how can I edit an article' question and to the question 'how do I use the WYSIWYG editor?'.

How do I manage my images?
To manage your images, which you can use in your home page content and your article contents, simply press the ‘manage images’ link. There you can upload new images and delete old ones, by pressing the appropriate buttons.

For what is the preview page list?
The preview page list gives you a feeling how your pages are going to look in your main site.

What are dynamic pages?
Dynamic pages are pages, of which you can specify the content completely to your own specifications.

What are static pages?
Static pages are pages, with content already specified by us, and made especially for you, for more information on them, see the 'How do I edit my ..... page?'.

What can I do with the 'Edit Dynamic pages'?
The ‘Edit Dynamic Pages’ overview, shows all your pages in a simple overview. With this view you can change the following things:
- You can change the status of an article to published or to not published, (when the checkbox is not checked the article is not visible on your main website).
- You can change the 'on top' status of an article, if you have more than one article in a specific category, the on top status makes sure that the article with the checkbox checked is the one that will be placed on top. It will be the first article people see, when they go to that directory.
- When the edit image is clicked (the pencil), you will be redirected to the edit view, were you can change various options of an article (see the 'How can I edit an article?' question).
- The 'delete all' link, will delete all articles associated with that particular menu entry, you will not be able to retrieve them. Once you have confirmed your action all the article in that menu entry will be deleted.
- When the delete image is clicked (the red cross), you will be asked if you are sure, you want to perform the delete action, depending on you choice the article gets deleted or not.
- When you press the preview image (magnifying glass), you will be redirected to a page where you can view your article. Remember this is not the final look; it will look a bit different on your website, because of the styling.

How can I edit an article?
To edit an article, press the edit image (pencil) and you will be redirected to the edit article page. In the edit article page you can specify the following things for an article, the main picture, which will be displayed on top of your article (Press browse and then upload to upload a main picture, to delete a picture, press delete picture). You can also fill in the page title and a short description of the article. Below that one can select under which menu entry, you wants this article to be visible. The page content can be specified with the WYSIWYG editor, to use this editor go to the 'How do I use the WYSIWYG editor’ question below this one.

How do I use the WYSIWYG editor?
The use of the WYSIWYG editor is the same as any document editor program, like Word or Open Office. The most important thing you need to know is how to insert an image (see also ‘how do I manage my images' question). First right-click on an image and select 'Copy Image Location', then press the image picture in the WYSIWYG editor and paste the image location in the link input field. And an image has been added to your article.

What can I do with the Edit static pages overview?
With the edit static pages overview, you can specify the content of the following pages, home, contact us, faq and about us. To edit the content of your gallery you must go the gallery in the navigation. Per static page, you can press the edit button and you can check the published checkbox. If the published checkbox is checked, the page will be visible on your main site. To edit content, see the 'How can I edit my ....page.' questions.

How can I edit my Home page?
To edit your 'Home' page, simply press the edit image (pencil) and you will be redirected to the edit Home page page. There you can specify the look of your home page, with the WYSIWYG (What You See Is What You Get) editor, the editor works the same as with the add and edit off a dynamic page.

How can I edit my Contact Us page?
To edit your 'Contact Us' page, simple press the edit image (pencil) and you will be redirected to the edit Contact Us page. There you can fill in the fields and your 'Contact Us' page will be generated. Once you have filled in the fields, simple press 'save changes', to save your 'Contact Us' page content.

How can I edit frequently asked questions?
To edit your 'FAQ' page, simple press the edit image (pencil) and you will be redirected to your 'FAQ' overview page. There you can do the following actions, add, edit and delete a frequently asked question. To add a FAQ simple press 'Add new FAQ' link and fill in the
question and the answer and then press Save. To edit a FAQ question, simple press the edit image (pencil) and then change the question and the answer and then press the Save button. To delete a FAQ, simple press the delete image (red cross) and once you have verified your action, the question will be deleted.

How can I edit my 'About Us' page?
To edit your 'About Us' page, simple press the edit image (pencil) and you will be redirected to the edit About Us page. There you can make an introduction about Who you are and What you do. Once you have filled in these two fields, simple press 'save changes', to save your 'About Us' page content.

How can I edit my 'Gallery' page?
To edit your 'Gallery' page, go to the navigation and press 'gallery', there you will see all the options to edit you gallery content.

2.2. Category Module

How does the category module work in general?
Added categories appear on the main page; in certain other modules you can add articles or products/media to a specific category. Article and products/media can only be added to subcategories.

How can I add a category?
You can add a category by clicking on ‘Add New Category’ on the left in the menu.

How can I edit a category?
A category can be edited by clicking on the pencil beneath ‘Edit’ in the table.

How can I delete a category?
Simply click on the cross beneath ‘Delete’ in the table to delete a category. You have to confirm this action before it gets performed.

How can I change the parent of a sub category?
Just click on the pencil beneath ‘Edit’ and select a different parent category.

2.3. Customer Module

What can I do here?
- You can search through your registered customers by a couple of criteria.
- You can order the search results by customer id, customer name, customer username, location, e-mail or access level in ascending or descending order.
- You can view, edit or delete a customer from your system.

How to search?
You can search for a particular customer by filling in a customer id.
You can also search for all the customers that satisfy the criteria you specify, like name, e-mail, active non-active etc.

How to view customer details?
Fill in the information you know about the user you want to view in the search form. Press the search button. In the resulting list select 'view' on the right of the customer row.

How to change customer details?
Fill in the information you know about the user you want to view in the search form. Press the search button. In the resulting list select 'edit' on the right of the customer row. Edit the customer information in the fields and press 'save'.

How to delete customers?
Fill in the information you know about the user you want to view in the search form. Press the search button. In the resulting list select 'delete' on the right of the customer row. You will be asked for confirmation and after you click OK the customer will be removed from your system.

2.4. Gallery Module

How do I add a new gallery category?
To add a new category to your gallery, type in the category name in the 'Add New' input field. Then click on the 'Select / Add' button. Your new category will be created and selected automatically.

How do I select an existing gallery category?
To select an existing category, simply click on the 'Select Category' drop down list of categories, and select the category from the list. If you have JavaScript enabled, the category will be selected automatically. If not, click on the 'Select / Add' button to select the category.

How do I add images to the selected gallery category?
To add images to the selected category, click on the browse button under the 'Upload Images' header. Then, select all the images you wish to upload and click 'Open'. Finally, click 'UPLOAD' to upload the selected images to the current category.

How do I delete images from the selected gallery category?
To delete an image from the current category, find the image in the list of thumbnails and click the 'Delete' button under the image. This will delete the image from the list.

How do I delete the entire selected category?
To delete an entire category, click on the 'Delete Entire Category' button that is located on top of the category thumbnail overview.

2.5. Media Module

How does the media module work in general?
The media module is capable of converting media files to web-capable media files, with a smaller file size. You can use this for instance to upload a video or music file, assign the media file to a certain category, and show it to the whole world with the embedded media player.

How can I add a media file?
Click on ‘Add New Media’ on the left inside the menu structure. Follow the onscreen directions and that’s it.
How can I edit a media file?
Click on the pencil beneath ‘Edit’ in the table to edit a media file. Notice: It is not possible to edit the media file itself; you can only edit the title, description or category for the specific media file.

How can I delete a media file?
Click on the cross beneath ‘Delete’ in the table to delete the media file, you have to confirm this action before it is performed.

Why do I have to activate each media file?
Adding a media file is a very delicate process. A lot of things can happen. To be absolutely sure that you are satisfied with the outcome of the conversion process, we encourage to skip through the media file so you can see the conversion job.

How can I make sure that only users can see the media files who are actually a (paying) member of my website?
When you tick the checkbox beneath ‘Membership only’, only users who have a membership can view the media file (it has to be active too!).

How can I view a media file before I make it visible on the main page?
Click on the preview thumbnail to view a media file in the wizard.

How can I view a media file but not interfere with the ‘streamed’ statistic?
When you view a media file inside the wizard, it does not count the ‘streamed’ statistic.

How can I view statistics about my media files?
You can view the categorization partitioning of the uploaded media files when you click on ‘View Media Statistics’ in the menu on the left.

2.6. Membership Module

What can I do here?
Here you can view all the memberships that customers of your site have. You can view the username of the holder the begin date of the membership and the expiry date. You can also terminate or activate each membership separately.

How do I activate a membership?
Find the membership you want to activate in the list and press 'activate' next to it. You will be asked for confirmation after which the membership will be activated.

How do I terminate a membership?
Find the membership you want to terminate in the list and press 'terminate' next to it. You will be asked for confirmation after which the membership will be terminated.

How do I set up the memberships?
You can set up memberships for 1, 2, 3, 6 and 12 months, each with its own price. You can do this in the Setup Memberships link under the membership navigation. People can then browse to your website and buy a membership for your site. Then they will be able to access membership only content.
2.7. Menu Module

How does the menu module in general?
Added menu items appear in the main menu on the front page. You have to add an article to a subcategory, to add content to your menu entry.

How can I add a menu item?
Click ‘Add New Menu Item’ on the menu left to add a menu item.

How can I edit a menu item?
Click on the pencil beneath the column ‘Edit’ in the table to edit the menu entry. Notice: When a menu item has child menu items, a parent menu item cannot be selected.

How can I delete a menu item?
Click on the cross beneath the column ‘Delete’ to delete the particular menu item. Notice: When you delete a parent menu item, all of its children menu items will be deleted as well.

How can I change the parent of a sub menu item?
Click on the pencil beneath ‘Edit’ and then change the parent menu item.

Why can’t I add an article when I have a menu item?
You need a sub menu item to add an article.

How can I change the order in which the menu items appear on the front page?
Click either on the ‘order up’ or ‘order down’ image to change the order in which the menu items will appear on the front page.

2.8. Newsletter Module

Uploading a header for your Newsletter
The first step for a successful newsletter campaign is to upload a header for your newsletter, this will be the picture above your newsletter. For now only .jpg files can be uploaded. To upload the newsletter header, simple press ‘browse’ and then ‘upload’.

What is Manage Recipients Database?
Under the manage recipients database link you can fill in your database with email contacts, you can do this by hand in the text field or you can do this through means of a CSV file. For the CSV file upload, one can specify the columns, with the various input fields. Once you have filled the email addresses one must specify under which group you want them to be added, this can be done through a new group or an already existing group.

Download subscribers as a CSV file?
You can download your database entries of your newsletter campaign in a CSV file, if you want to use them in a different place. This can be done by clicking on ‘Manage Recipients Database’ and then by pressing ‘click here’.

What is the ‘with_register’ group?
The ‘with_register’ group contains all the customers, who subscribed to your newsletter by means of the register form.
Making/Editing a newsletter

If you want to start a new campaign, first you specify if you want to use an old campaign or if you want to start a new one by using the select template dropdown or by typing a new one. If you want a successful campaign you have to fill in from and the subject. If you have an e-commerce site plus, you can also include products in your newsletter, see the 'How do I include products in my newsletter' question. Next you can fill in the HTML content of your newsletter; the editor (which you also use in article management) works the same as a text editing program like Word or Open Office and is pretty straightforward. Next you must specify the text-only content for users who use a mail client which does not support html content.

How do I include products in my newsletter?
To include products in your newsletter tick the box include products list. To specify which products you want to show in your newsletter, click show the new products link. In the popup that opened you can specify which products you want to include in your newsletter, by selecting the checkbox. Then you must fill in the text, you want to show for the product in your newsletter. If you are done you can press accept to accept your changes or cancel to ignore them.

For what is the preview button?
With the ‘preview’ button you can see how your newsletter looks.

For what is the accept button?
By pressing the ‘accept’ button you will save your template.

The final step

The final step in your newsletter campaign is to actually send the email to all your customers. First you can select which campaign you want to send to your customers in the dropdown box on top of the screen. Then you can start sending to your customers. Before you can do this, you must send out a test email to someone, who than can verify that the newsletter looks as it is supposed to look. After you have done this, simply click 'Test Confirmed' and then press ‘send all’ and all your customers will get the newsletter. If for some reason you want to cancel your campaign halfway through sending, simply close your browser and the sending of emails will stop.

How do I manually manage subscribers?
You can manage your newsletter subscribers by going to ‘Manage Recipients Database’, there you can click the appropriate link. In a popup window an overview of all your customers is given, here you can delete and edit all your customers.

2.9. Order Module

How do I search for orders?
To search for specific orders, you can use the 'Search orders' form. On the 'Search orders' form there are five search criteria that can all be combined to narrow down the search results. The criteria are explained here:

- Username, searches orders belonging to a single customer with this username.
- Actual name, searches orders belonging to a single customer with this name.
- Order date, restricts the searched orders to a certain starting and ending date. You must specify the date in years, months, days. The second day of March 2007 would
become '2007-03-02', '2007.03.02' or '2007 03 02'.

- Order state, which restricts the orders searched to the specified state.
- Show, limits the number of orders per page to reduce the amount of scrolling required.

How do I change the state of an order?
To change the state of an order, first search for the order and find it in the list. Then click on the desired state button. The available states depend on the current state of the order. Orders that have been made 'archived' will be removed from the list, and can only be shown by searching with 'show archived' on. Archived orders can be taken out of the archive again. If an order is out-of-stock, make sure you have enough stock before trying to set that order to confirmed, because the stock for that order will be reserved when making it confirmed.

How do I see an orders full details?
To view the details of an order, first search for the order and find it in the list. When you click on the order reference number, a popup will appear showing that orders details.

2.10. Product Module

How do I search for products?
To search for specific products, you can use the 'Search Products' form. On the 'Search Products' form there are five search criteria that can all be combined to narrow down the search results. The criteria are explained here:

- Keyword, searches product based on keywords in the product name and description.
- Category, restricts the searched products to a single category.
- Selling price, restricts the searched products to a certain minimum and maximum price. The price can be entered in whole dollars or in dollars and cents. You can use either a comma or a dot to separate whole and cents, so '123.45' and '123,45' are equivalent. Show, limits the number of products per page to reduce the amount of scrolling required.
- Special Condition, which restricts the products searched to very specific conditions.

How do I add a new product?
To add a new product, click on the 'Add New Product' link, which can be found in the navigation menu on the left, under the 'Product' menu entry.

How do I edit an existing product?
To edit an existing product, search for the intended product and find it in the list. When you click on the name the edit product popup will be shown where you can alter the product details. To save the changes to this product click on the 'Edit Product' button.

How do I change multiple products at once?
Some properties of products can be edited from this screen right away. Click on one of the input fields in the 'stock' or 'price' columns and change the value to the new value. To store these changes, click on the 'Make Changes' button. Stock has to be entered in whole units, while price can be a decimal number.

How do I delete products?
To delete a product or several products from the list of products, click on the 'delete' checkbox in the first column of the product list. Then, click on the 'Make Changes' button to delete that product.
2.11. Template Module

What can I do with the template module in general?
With the template module you can easily change between templates, by simply pressing the picture and then your template is changed to that layout, if you do not like it simply press on the picture of your previous layout and it is changed back.

2.12. Statistics Module

What can I do with the statistics module in general?
The statistics module gives you insight in your weekly, monthly, or annually sales. You can browse through each week and see how sales went that week. In conjunction with advertisements you can see the actual effect they have on the sales. Not only sales in general can be inspected, but also every individual product can be inspected.

How can I browse through the calendar, or how can I browse through weeks?
Click on the 'previous' or 'next' links on the center of the screen.

How can I return to the current week after browsing through the calendar?
Click on the 'go to current week' link that is on the center of the screen between 'previous' and 'next'.

How can I check sales of a specific category?
Click on the bottom of the screen on the category you want to see in particular.

How can I inspect the total profit chart?
The profit chart can be seen when you click on 'profit overview' in the main menu.
Appendix E: Requirements document and plan of action
www.website4free.co.nz

Bachelor project IN3700

Requirements Document

Version 2.0
June/July 2007

Project Group
Remy Alidarso 1221736
Zdravko Anguelov 1232436
Terence Bahlen 1217313
Frits de Nijs 1224263

Company
SecureSpace
Business Manager: Maria Veleva
Project Manager: Dimitar Velev

University
Delft, University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science

Commission
Project Manager: Ir. B.R. Sodoyer
Project Accompanier: J. de Vries
1. Introduction

This document specifies the requirements analysis for the project `websiteforfree.co.nz`, which will be developed at SecureSpace Ltd. (Auckland, New Zealand). SecureSpace wants to give people (small businesses) the opportunity to register and build their own website based on a chosen template. Therefore this document specifies the requirements to make `websiteforfree.co.nz`.

First we will specify the current situation, in order to explain why SecureSpace wants this project to be developed. In the third paragraph, the general architecture of the templates will be further specified. This is followed by the functional and non-functional requirements, after which the use cases related to this project will be given.
2. Current situation

Currently, a few hosting providers offer the service which SecureSpace wants to offer with websiteforfree.co.nz. These providers offer quite extensive services, however SecureSpace wants to distinguish from these providers by offering a website for free in as few steps as possible. SecureSpace also wants to distinguish from the other providers by making the system as user-friendly as possible and give the clients a lot of options, to make their website a success.

There is a market for this service, because a lot of small businesses understand to need for a website, but they do not have the means to buy one from a website development company. SecureSpace wants to offer a service for this market by means of the websiteforfree.co.nz. So in short the service that needs to be build, should provide a complete and easy website creation in a few steps.
3. General Architecture

This paragraph will discuss the general architecture as indicated through means of various interviews with the business manager and the project manager.

The software package that is going to be created will be able to build websites, each with a distinct template set. These templates will be the following:

- article site;
- simple e-commerce site;
- advanced e-commerce site;
- media site.

A main feature of the system that is going to be built is the division into modules. The modules will be able to operate together and individually. So far we have identified the following modules for the default generated website:

- database module, used for the connection with the database;
- security module, used for the validation of forms, sql queries, etc;
- error module, used to capture error messages and display them on the site at a specific place;
- session module, used to let people login in a secure way, and to make sure that pages are made (un)available for people with different privileges;
- content management module, which is used to change the (static) pages of the website, this module can be divided into the following sub modules;
- WYSIWYG module, used to easily modify the content of the website
- menu module, the modify the menu of the website;
- newsletter module, used to send newsletters, however this module can also be used to only sent newsletters without having to first create a website (this can be seen as a special feature);
- package module, used to ‘pack’ the site of a customer into a zip (os something like that) in case they want to switch hosting providers. In this case they only receive the html and css files. The second option for the package module is to ‘pack’ everything, this is the case when they want to become a professional customer of SecureSpace, in this way SecureSpace can change their site, without interference form the websiteforfree.co.nz, into a professional site (they have all the files in one package).

The modules mentioned above are the main modules which all of the templates will have (the article/media/simple e-commerce site may not have a newsletter system; this is yet to be discussed). There are distinctive features for each template; the article website will have the following distinctive module:

- article management module, so the owner can control articles (add them, publish them, delete them after a certain date, etc);

(we discussed the option of adding comments to articles, but this is set on hold for the time being, if this is going to be implemented in the future we will add an comment module to the article management module).

The media site has the following extra modules:

- membership module, to give subscribers access to the full content;
- media module, to upload media, delete media, etc;
- statistics module, to see how many visitors watched what videos, etc.

We also discussed the following modules for the media site, but these are set on hold for the time being:
• (profile module, so users who are logged in can change their profile);
• (comment module, so users can post comments or give a rating on media).

The e-commerce websites have a lot of extra modules:
• product module, to keep track of the products (stock management, etc);
• category module, to add, edit or delete categories to which products can be assigned;
• customers modules, to add, edit, delete customers;
• pay track module, to allow for a financial overview and to allow multiple forms of payment (only for the advanced e-commerce site)
• shopping cart module, so people can virtually shop in your shop!
• newsletter module, as mentioned earlier.

We also discussed the following module for the e-commerce site, but it is set on hold for the time being:
• (banner module, to show banners on the website).

The above mentioned modules are subject to change, because along the way we might consider to add extra modules or to pack certain modules into one, but this depends on the course of the project.
4. Functional requirements

This paragraph will specify the purpose and the scope of the project, as well as the stakeholders and the various actors involved in the development of the websiteforfree.co.nz.

4.1. Purpose and scope of project

The purpose of this project is to create a software package that lets businesses or individual people create a simple and fast, but foremost complete website. Users can choose from four different websites that they can create. There are a few basic templates available to choose from, but users will also be able to buy a template which they think is better suitable for their website.

The scope of the project is determined by the number of modules, i.e., the number of functionalities which can be added to a website, the four template websites (article, simple/advanced e-commerce site and the media) all offer the same basic functionality and their distinctive functionality. The project involves a complete front end system for the visitors of the website and a complete back end system for the administrator of the website (which includes all sort of management tools, we refer the reader to the different modules in the paragraph 3).

For the website to be commercially viable, hosting will not be provided freely, but users are obligated to take a subscription on one of our hosting subscription packages. Users can have their site get packaged after 6 months in case they want to change hosting provider. They won't be able to package their website entirely, just the HTML + CSS (client side) files. For this requirement their will also has to build a package module, so that a complete site can be packaged and sent to the client.

4.2. Stakeholders and actors

In this paragraph we will first specify the main stakeholders in this project and then we will discuss them in detail.

website4free.co.nz involves the following main stakeholders:

- SecureSpace, the company who initially gave the order for this project;
- the development team that creates the project;
- the clients who will use the service to create a website;
- and the customers of the clients.

4.2.1. SecureSpace

SecureSpace is the company who developed the business model for the project websiteforfree.co.nz and they will be responsible for the availability of the system, so they have to make sure the system is available at all times. SecureSpace takes care of the hosting provided for the generated website and keeps track of the payments for this hosting, with the pay tracking system delivered with the websiteforfree.co.nz.

4.2.2. Development team
The development team is responsible for the implementation and the correct functionality of the created software package. They have to make sure that the main functionalities are met and that the system can be used in the future and that it is easy to add functionality, therefore the system will be build in modules (see appendix A and appendix B).

4.2.3. Clients

The clients of the provided service are the most important stakeholders when it comes to making the project a success, their (dis-) content of the system will make it a success/failure. In order to facilitate these clients the created software package should be very user-friendly. Thus success of the project depends on the content of the clients, if they can reach the audience they want, they will not cancel their subscriptions and the business model will be met.

4.2.4. Customers of clients

The customers of the clients are also an important stakeholder, because they influence the clients, if they feel comfortable using the generated site, then clients can reach their audience and they will also feel good about their site and then the business model can be met. The Customers of the clients make the site interesting for the client and than the clients will continue their subscriptions.
5. Non-functional requirements

This paragraph will specify the non-functional requirements of the project, we will mentions the most important non-functional requirements, these are, user interface (user friendliness), documentation (for future additions), testing, performance characteristics, search engine optimization, quality standards, security issues, compatibility and some general requirements.

5.1. User interface

The user interface for every screen should be very intuitive. The success of this project is partially based on the ease of use of the software. If the users don't feel comfortable in using the software and tools provided, they won't use it. Another important thing besides the management of the generated sites is the usability of the generated sites. The customers of our clients should be able to easily find products or media with certain keywords. The (nonetheless) available and extensive help references should be consulted as little as possible.

5.2. Documentation

All the software that is created for this project should be well documented. In every source file there should be a general description of what the file is used for and other purposes. The functions inside should be well commented too. Special software is to be used to generate programmers’ documentation, as well as user documentation. The software to be used is phpdoc, which has a similar functioning as javadoc. Second, a user’s manual should be written that describes how the website can be adjusted at the users wish. Included in the user’s manual are all the management tools. After this user’s manual, there should be a new users manual that focuses on the use of the websites that are to be created, e.g. How to use the client side of the web shop or media section of a generated site.

5.3. Testing

The entire source that is written needs to be thoroughly tested. The use of automated tests has many advantages when compared to manual testing. This way we can achieve a higher level of code coverage. We are also forced to make sure that every branch in the code tree is present, in case, for instance, some if clause fails. We found unit test 5 (UT5) as a suitable testing environment for php 5.

5.4. Performance characteristics

All the software code should be optimized up to a certain degree in order to achieve a fast website that doesn't take long to react to users input. The responsiveness level is also a key element in customer binding. If customers never have to wait more than a few seconds for a page to load, they will last longer on a website, so clients of this project will not be disappointed in their online customers.

5.5. Search engine optimization

The main website for the website for free project should be fully optimized for search engines in order to get more customers. All the generated source code for the websites is optimized for
search engine robot indexers. In this way, the project can guarantee the client to some extend a good hit index when people search for keywords used on their generated website.

5.6. Quality standards

The quality standards for this project are quite high because high quality is expected by our aimed client group. The aimed user group wants to create a fully functional, extended website without having to think about the technical aspect of creating one. The technical aspects of creating a website should be completely taken out of hand by the project on hand.

5.7. Security issues

Because with this project, many websites will be created, it is absolutely crucial that the website generator itself will be tested in order to minimize bugs or vulnerabilities. Next, the generated websites themselves should be tested on layout bugs, functionality bugs, and vulnerabilities. Some extra constraint is that the server should be using the best security approaches as possible, to further limit down the consequences in case of a security penetration.

5.8. Compatibility

The software that is going to be created for this project will consist of PHP code, XHTML, CSS, and JavaScript. The generated XHTML and CSS source code should comply with the w3c standards. MySQL is also going to be used in this project as database software. Because a requirement is to reach as many users as possible, the software that is going to be created should work on all web browser clients that have more than 1% usage of all web traffic. Web browsers that match that description in New Zealand are (both pc and macs):

- Internet Explorer 6.0
- Internet Explorer 7.0
- Mozilla Firefox
- Mozilla Camino
- Safari

5.9. General Constraints

The generated websites should not consume too much data traffic, so users who can show videos on their website should be made aware that videos can take up a lot of traffic, so if they exceed their standard hosting level data traffic, there is an additional fee.
6. Use Cases

This document contains the Use Cases for the 'websiteforfree.co.nz' website and all websites created using the website creation tool on the 'websiteforfree.co.nz' website. Websites that have been customized for specific needs of customers might have different Use Cases from the ones defined here. Each Use Case is presented as a list. User actions are in normal font, while system responses are in italics.

The Use Cases will be divided into smaller groups of Use Cases by their common owners. For instance, Use Cases belonging to the generated media website can be found between the generated website Use Cases, website type media.

6.1. Websiteforfree.co.nz user area

These Use Cases belong to the websiteforfree.co.nz website itself, and the things the users can do there. They involve becoming a client, creating a website and the various website administrator functions.

6.1.1. Use Case 1: Sign up for websiteforfree.co.nz

**Actors:** Visitor of websiteforfree.co.nz

**Entry Condition:** The visitor wants to create a free website.

**Flow of Events:**
1. Click 'sign up' link.
2. Display 'domain name check' page.
3. Enter desired domain name.
4. Select desired domain type, 'top level' or '.websiteforfree.co.nz'
5. Click 'continue' button
6. Display 'create account' page.
7. Enter personal details.
8. Click 'continue' button.
9. Check input for syntax errors.
10. Depending on the status of the syntax error check:
    11. If there was a syntax error, display filled 'create account' page with error(s) and go to 7.
    12. If there was no syntax error, display 'select website type' page and go to 11.
    13. Select intended website use from 'personal', 'blog/article', 'webshop', 'media'.
    14. Click 'continue' button.
    15. Display 'select hosting' page.
    16. Select expected level of hosting required.
    17. Enter payment details.
    18. Click 'continue' button.
    19. Display 'legal agreement' page.
    20. Depending on the legal agreement:
        21. If in agreement, click 'I Agree' button.
        22. If not in agreement, click 'I Decline' button.
    23. Depending on the provided input:
        24. If the user clicked 'I Decline', display 'main' page and stop.
        25. If the user clicked 'I Agree', display 'registration overview' page and go to 20.
    26. Click 'print' button.
    27. The system prints the overview with all the details.
28. Depending on the provided input (what the user clicks next):
29. If the user clicked 'cancel', display 'main' page and stop.
30. If the user clicked 'submit', go to 23.
31. Send activation email.
32. Click activation link in the activation email.
33. Activate user in the database.
34. Log in user.
35. Display 'site management' page.

**Exit Condition:** The visitor has become a client of websiteforfree.co.nz and has a free website, unless the visitor declined the agreement.

### 6.1.2. Use Case 2: Log in to websiteforfree.co.nz

**Actors:** Client of websiteforfree.co.nz

**Entry Condition:** The client wants to log in.

**Flow of Events:**
1. Fill in username and password.
2. Click 'login' button.
3. Check username and password combo.
4. Depending on the result of the check:
   5. If not correct, reload page, display error and go to 1
   6. If correct, go to 5.
7. Log in user username.
8. Display 'site management' page.

**Exit Condition:** The client has logged in.

### 6.1.3. Use Case 3: Change personal details

**Actors:** Client of websiteforfree.co.nz

**Entry Condition:** The client wants to change his personal information and is logged in.

**Flow of Events:**
1. Click 'my account' link.
2. Display 'account options' page.
3. Click 'change personal details' link.
4. Display 'personal details' page.
5. Alter contents of some fields.
6. Enter password.
7. Click 'submit' button.
9. Display 'account options' page.

**Exit Condition:** The client has changed his personal information.

### 6.1.4. Use Case 4: Change hosting level

**Actors:** Client of websiteforfree.co.nz

**Entry Condition:** The client wants to change the hosting level for his site and is logged in.

**Flow of Events:**
1. Click 'site management' link.
2. Display 'site management' page.
3. Click 'hosting details' link.
4. Display 'hosting details' page.
5. Click 'change hosting plan' link.
6. Display 'hosting options' page.
7. Select expected level of hosting required.
8. Click 'change hosting' button.
9. Alter hosting level for the website for next month.
10. Display 'site management' page.

Exit Condition: The client has changed the hosting level for his site.

6.1.5. Use Case 5: Change website type

Actors: Client of websiteforfree.co.nz
Entry Condition: The client wants to change the type of website for his site and is logged in.
Flow of Events:
1. Click 'site management' link.
2. Display 'site management' page.
3. Click 'website details' link.
4. Display 'website details' page.
5. Click 'alter website type' link.
6. Display 'alter website type' page.
7. Select new website type.
8. Click 'submit' button.
9. Display 'are you sure, data will be lost' warning.
10. Depending on the users intentions:
11. If not intended to rework site, click 'no' button.
12. If intended to rework site, click 'yes' button.
13. Depending on the user input:
14. If user clicked 'no', display 'website details' page and stop.
15. If user clicked 'yes', go to 12.
16. Change website directory structure to meet new website type demands.
17. Change website database to support new website features.
18. Display 'website details' page.

Exit Condition: The website has had its type changed, unless the no button was clicked.

6.1.6. Use Case 6: Select entire website template

Actors: Client of websiteforfree.co.nz
Entry Condition: The client wants to select a general template for his website and is logged in.
Flow of Events:
1. Click 'site management' link.
2. Display 'site management' page.
3. Click 'website template' link.
4. Display 'website templates' page.
5. Click the '<<template name>>' link.
6. Display 'template details' page in a popup screen.
7. Depending on the users preferences:
8. If not the desired template, close popup screen and go to 5.
9. If it is the desired template, close popup and go to 8.
10. Click 'use template' button.
11. Copy template to website directory.
12. Display 'site management' page.

Exit Condition: The clients website has a general template.

6.1.7. Use Case 7: Edit entire website template
**Actors:** Client of websiteforfree.co.nz

**Entry Condition:** The client wants to edit the general template of his website and is logged in.

**Flow of Events:**
1. Click 'site management' link.
2. Display 'site management' page.
3. Click 'website template' link.
4. Display 'website templates' page.
5. Click the 'edit template' link.
7. Display 'template editor' page.
8. Use editor to edit template to clients liking.
9. Click 'save' button
10. Overwrite old website template with new template.

**Exit Condition:** The client has edited the general template of his website.

---

**6.1.8. Use Case 8: Edit website static pages and menu structure**

**Actors:** Client of websiteforfree.co.nz

**Entry Condition:** The client wants to edit the static pages and menu of his website and is logged in.

**Flow of Events:**
1. Click 'site management' link.
2. Display 'site management' page.
3. Click 'website content' link.
4. Load existing pages and menu structure into editor.
5. Display 'static page editor' page.
6. Use editor to edit pages and menu to clients liking.
7. Click 'save' button
8. Edit, add and remove static page content to database, and overwrite menu structure in database.

**Exit Condition:** The client has edited the static pages and menu of his website.

---

**6.2. Websiteforfree.co.nz admin area**

These Use Cases belong to the websiteforfree.co.nz website itself, and the things the administrators can do there. They involve viewing site statistics, printing tax invoices and packaging a website for custom development.

**6.2.1. Use Case 9: Display site statistics**

**Actors:** Administrator of websiteforfree.co.nz

**Entry Condition:** The administrator wants to view website statistics and is logged in.

**Flow of Events:**
1. Click 'administrator pages' link.
2. Display 'administrator pages' page.
3. Click 'extended site statistics' link.
4. Load static page content into editor.
5. Display 'extended site statistics' page.
7. Display 'site statistics' page.

**Exit Condition:** The site statistics are displayed.

---

**6.2.2. Use Case 10: Print tax invoice**
**Actors:** Administrator of websiteforfree.co.nz

**Entry Condition:** The administrator wants to print an invoice and is logged in.

**Flow of Events:**
1. Click 'administrator pages' link.
2. Display 'administrator pages' page.
3. Click 'payment details' link.
4. Display 'payment details' page.
5. Select the user from the list, or search for the user.
6. Click the user.
7. Display 'user payment overview' page.
8. Click the payment to print invoice for.
9. Display 'user payment' page.
10. Print page.

**Exit Condition:** The administrator has printed an invoice.

**6.2.3. Use Case 11: Package entire site**

**Actors:** Administrator of websiteforfree.co.nz

**Entry Condition:** The administrator wants to get the full code of a site and is logged in.

**Flow of Events:**
1. Click 'administrator pages' link.
2. Display 'administrator pages' page.
3. Click 'website tools' link.
4. Display 'website tools' link.
5. Select the website from the list, or search for the website.
6. Click 'package website'.
7. Copy directory structure and required php files. Provide files for download.

**Exit Condition:** The administrator has packaged a website.

**6.3 Generated website user area**

These Use Cases belong to a generated website, and the things the users can do there. They involve becoming a customer, purchasing items and using membership content.

**6.3.1. Use Case 12: Registering as a member of the free website.**

**Actors:** Visitor of a free website.

**Entry Condition:** Visitor has accessed the free website.

**Flow of Events:**
1. Click the 'register' link.
2. Display the 'register page'.
3. Enter the required fields.
4. Click 'continue' button.
5. Depending on the content of the free website:
6. If a legal section exists, display the legal section and go to 6.
7. If no legal section exists, go to 7.
8. Depending on the contents of the legal document:
9. If in agreement, click 'I agree' button and go to 7.
10. If not in agreement, click 'I decline' button and go to 11.
11. Display the user overview.
12. Depending on the data entered:
13. If the entered data is correct, click 'Submit' button and go to 9.
14. If the entered data is not correct, click 'Cancel' button and stop.
15. Create a new ‘wff_user’ with provided data in the database.
16. Record the visitor as 'logged in'.
17. Display the 'main page'.

**Exit Condition:** The Visitor is logged in as a User and is back at the main page.

### 6.3.2. Use Case 13: Browsing for an item from the web shop.
**Actors:** Customer of a free website.
**Entry Condition:** Customer has accessed the free website.
**Flow of Events:**
1. Click the 'category <<name>>' link.
2. Display the 'product list page' with all 'products' in that 'category'.
3. Depending on whether the item was found:
4. If the item was not found, go back to 1.
5. If the item was found, done browsing.

**Exit Condition:** The Customer has browsed for a (list of) product(s).

### 6.3.3. Use Case 14: Searching for an item from the web shop.
**Actors:** Customer of a free website.
**Entry Condition:** Customer has accessed the free website.
**Flow of Events:**
1. Click the 'search product' link.
2. Display the 'search page'.
3. Enter the search criteria in at least one field.
4. Click 'search' button.
5. Select the 'products' matching the entered criteria.
6. Display the 'product list page' with the matching 'products'.
7. Depending on whether the item was found:
8. If the item was not found, go back to 1.
9. If the item was found, done searching.

**Exit Condition:** The Customer has searched for a (list of) product(s).

### 6.3.4. Use Case 15: Logging in to the clients website.
**Actors:** Visitor of a free website.
**Entry Condition:** Visitor has accessed the free website.
**Flow of Events:**
1. Enter his 'username' and 'password' in the corresponding fields.
2. Click 'login' button.
3. Check the provided username and password combo.
4. Depending on the check result:
5. If the username and password combo exists, log in user 'username'.
6. If the username and password combo does not exist, display an error indicating incorrect password and go to 1.

**Exit Condition:** The Visitor is logged in as a User and is back at the main page.

### 6.3.5. Use Case 16: Entering a single item into the shopping cart.
**Actors:** Customer of a free website.
**Entry Condition:** Customer has accessed the free website.
**Flow of Events:**
1. The customer 'searches'(14) or 'browses'(13) for the product.
2. Clicks the desired product '<name>'.
3. Display the single product page for the selected product.
4. Enter desired amount.
5. Click 'add to shopping cart' link.
6. Record the selected product in the users shopping cart cookie.
7. Display the 'shopping cart' page.

**Exit Condition:** The Customer has added an amount of an item to his shopping cart.

### 6.3.6. Use Case 17: Entering multiple items into the shopping cart.

**Actors:** Customer of a free website.

**Entry Condition:** Customer has accessed the free website.

**Flow of Events:**
1. The customer 'searches' or 'browses' for a list of products.
2. Select multiple products.
3. Click 'add selected items to shopping cart'.
4. Record the selected products in the users shopping cart cookie.
5. Display the 'shopping cart' page.

**Exit Condition:** The Customer has added amounts of several items to his shopping cart.

### 6.3.7. Use Case 18: Ordering a shopping cart.

**Actors:** Customer of a free website.

**Entry Condition:** Customer has accessed the free website and has put items in his shopping cart.

**Flow of Events:**
1. Click the 'order items' button.
2. Check if the client is logged in.
3. Depending on the login state:
4. If the user is not logged in, display 'login first page' and go to 4.
5. If the user is logged in, display the 'order overview page' and go to 5.
6. Perform 'log in' use case (15).
7. View order details and check amounts.
8. Fill in delivery and bill address details, if required.
10. Click 'order items' button.
11. Create 'order' in the database.
12. Display 'order overview page'.
13. View accepted order and pay for order. Continue using website.

**Exit Condition:** The Customer has ordered the contents of his shopping cart.

### 6.3.8. Use Case 19: Viewing membership content.

**Actors:** Customer of a free website.

**Entry Condition:** Customer has accessed the free website and has a membership.

**Flow of Events:**
1. Perform the 'log in' use case (15) if not logged in.
2. The customer 'searches' or 'browses' for membership content.
3. The customer clicks the membership content details.
4. The server checks the membership expiry date of the user.
5. Depending on the membership status:
6. If membership expired, display 'membership expired page' and go to 6.
7. If membership not expired, display 'membership content page' and go to 7.
8. Decide whether or not to get new membership.
9. Use membership content.

**Exit Condition:** The Customer has gained access to membership content.

### 6.3.9. Use Case 20: Posting a comment

**Actors:** Registered user of a free website.

**Entry Condition:** User has visited and read the article or item commented about.

**Flow of Events:**
1. Perform the 'log in' use case (15) if not logged in.
2. Click 'write a comment'.
3. Display the 'add comment page'.
4. Enter comment title and text.
5. Click 'submit comment' button.
6. Add the comment to the database.
7. Display 'article page'.

**Exit Condition:** The User has posted a comment to an article.

### 6.3.10. Use Case 21: Getting a membership

**Actors:** Registered user of a free website.

**Entry Condition:** User is registered at the free website.

**Flow of Events:**
1. Click 'subscribe to membership content' link.
2. Display 'membership registration page'.
3. Fill in payment details.
4. Click 'become a member' button.
5. Display 'membership details page'.

**Exit Condition:** The user has a membership for viewing membership content.

### 6.3.11. Use Case 22: Ending a membership

**Actors:** Registered user and member of a free website.

**Entry Condition:** User is registered at the free website and has an active membership.

**Flow of Events:**
1. Click 'membership details' link.
2. Display 'membership details page'.
3. Click 'end membership'.
4. Display 'thank you page'.
5. Click 'continue'.
6. Display 'membership details page'.

**Exit Condition:** The user has ended his membership.

### 6.4. Generated website admin area

These Use Cases belong to a generated website, and the things the admin can do there. They involve editing products, categories and customers, as well as adding articles and managing comments.

#### 6.4.1. Use Case 23: Adding a product

**Actors:** Administrator of a free website of the webshop type.

**Entry Condition:** Administrator has logged in.

**Flow of Events:**
1. Click the 'webshop products' link.
2. Display the 'webshop products' page.
3. Click the 'add product' link.
4. Display the 'new product' page.
5. Enter the required fields.
6. Click 'add product' button.
7. Add the product to the database.
8. Display the 'webshop products' page.

**Exit Condition:** The Administrator has added a product.

### 6.4.2. Use Case 24: Editing a product

**Actors:** Administrator of a free website of the webshop type.

**Entry Condition:** Administrator has logged in and knows which product to edit.

**Flow of Events:**
1. Click the 'webshop products' link.
2. Display the 'webshop products' page.
3. Search for the product.
4. Select the product from the list.
5. Depending on the intended changes of the administrator
6. If the administrator wants to edit min stock, stock, buying price, selling price and offer he can do so in the list of products.
7. Administrator edits small details of the product(s) and presses save all.
8. Edit the product in the database go to 11.
9. If the administrator wants to edit more details, goto 6.
10. Click 'edit product' button.
11. Display the 'edit product' page.
12. Change the intended fields.
13. Click 'edit product' button.
14. Edit the product in the database.
15. Display the 'webshop products' page.

**Exit Condition:** The Administrator has edited a product.

### 6.4.3. Use Case 25: Adding a category

**Actors:** Administrator of a free website of the webshop type.

**Entry Condition:** Administrator has logged in.

**Flow of Events:**
1. Click the 'webshop categories' link.
2. Display the 'webshop categories' page.
3. Click the 'add category' link.
4. Display the 'new category' page.
5. Enter the required fields.
6. Click 'add category' button.
7. Add the category to the database.
8. Display the 'webshop categories' page.

**Exit Condition:** The Administrator has added a category.

### 6.4.4. Use Case 26: Editing a category

**Actors:** Administrator of a free website of the webshop type.

**Entry Condition:** Administrator has logged in and knows which category to edit.

**Flow of Events:**
1. Click the 'webshop categories' link.
2. Display the 'webshop categories' page.
3. Search for the category.
4. Select the category from the list.
5. Click 'edit category' button.
6. Display the 'edit category' page.
7. Change the intended fields.
8. Click 'edit category' button.
9. Edit the category in the database.
10. Display the 'webshop categories' page.

**Exit Condition:** The Administrator has edited a category.

### 6.4.5. Use Case 27: Editing a customer

**Actors:** Administrator of a free website of the webshop type.

**Entry Condition:** Administrator has logged in and knows which customer to edit.

**Flow of Events:**
1. Click the 'webshop customers' link.
2. Display the 'webshop customers' page.
3. Search for the customer.
4. Select the customer from the list.
5. Click 'edit customer' button.
6. Display the 'edit customer' page.
7. Change the intended fields.
8. Click 'edit customer' button.
9. Edit the customer in the database.
10. Display the 'webshop customers' page.

**Exit Condition:** The Administrator has edited a category.

### 6.4.6. Use Case 28: Editing an order

**Actors:** Administrator of a free website of the webshop type.

**Entry Condition:** Administrator has logged in and knows which order to edit.

**Flow of Events:**
1. Click the 'webshop orders' link.
2. Display the 'webshop orders' page.
3. Search for the order.
4. Select the order from the list.
5. Click 'edit order' button.
6. Depending on the state of the order:
   7. If the order is not sent or completed, display the 'edit order' page and go to 7.
   8. If the order is sent or completed, display 'error page' and stop.
7. Change the intended fields.
8. Click 'edit order' button.
9. Edit the order in the database.
10. Display the 'webshop orders' page.

**Exit Condition:** The Administrator has edited an order, unless that order was already locked.

### 6.4.7. Use Case 29: Adding an article

**Actors:** Administrator of a free website of the article/blog type.

**Entry Condition:** Administrator has logged in.

**Flow of Events:**
1. Click the 'new article' link.
2. Display the 'new article' page.
3. Change the intended fields.
4. Click 'add article' button.
5. Add the article in the database.
6. Display the 'article <<name>>' page.

**Exit Condition:** The Administrator has added an article.

### 6.4.8. Use Case 30: Moderating a comment

**Actors:** Administrator of a free website of the article/blog type.

**Entry Condition:** Administrator has logged in.

**Flow of Events:**
1. Click the 'article <<name>>' link.
2. Display the 'article <<name>>' page.
3. Scroll to find the comment.
4. Click 'edit comment' button.
5. Display the 'edit comment' page.
6. Change the comment.
7. Click 'edit comment' button.
8. Edit the comment in the database.
9. Display the 'article <<name>>' page.

**Exit Condition:** The Administrator has edited a comment.
Appendixes

A: The model-view pattern

The generated free website will be constructed out of different individual modules. This modular approach has several big advantages:

- modules allow changing a website very quickly, simply by copying the extra modules into the folders;
- modules allow for individual construction and testing. Modules can be easily divided among the programmers;
- modules allow for increasing or decreasing the workload. Every module is an individual product, so a module can be dropped from or added to the project quite easily;
- modules allow for expanding the free website source code easily. A module could be expanded with new functionality without affecting other modules;
- modules can be reused for other projects. Once a module is finished, any website that needs the functionality provided by that module can just use that module.

For the construction of the modules, we decided to use the Model-View-Controller design pattern. This decision was made because every module has to have a central access point, has to generate HTML, respond to user input and change / use their own data. The View will be responsible for turning the data from the model into HTML that can be embedded into a larger HTML template. When a button or link on the generated HTML is clicked, the Controller has to respond to that user input, and decide the course of action to follow. The Controller is also responsible for processing input from other modules constructors. And if some data has to be changed or retrieved, it is the models responsibility to perform that action. This way, only the Controller has to be visible to the other modules, allowing for a loose coupling between the modules.

To access the modules from any point on the website, we decided to use the Singleton design pattern. The Singleton pattern allows for a single point of entrance to retrieve a modules controller, instead of making every controller known to every other controller. Anyone that needs a controller for performing an action can call the get instance method, and this method will retrieve the same instance of a module every time. If a module is not loaded into memory yet, it is the Singletons responsibility to load the module. If a module is not available, because it has not been added to the website, it is the Singletons responsibility to provide a suitable response, without breaking the functionality (since every module is optional).

To illustrate how the modules will work together, we give an example showing how a shopping cart overview page might be generated. The shopping cart overview will display a list of products that are in the users virtual shopping cart. At some point, the PHP script generating the page will come across the central content area. At this point, the script will call the ModuleLoader to retrieve the ShoppingcartController. If the shopping cart is not loaded, the ModuleLoader will construct a new ShoppingcartController, store it in its list of loaded modules, and return its link. The script will then continue by asking the ShoppingcartController to display a shopping cart overview. The Controller will request the Model to provide a list of the unique product names in the cart. Because the ShoppingcartController has no access to Products, it will have to request the ProductController to return a list of the actual Products belonging to the names. To do so, the ShoppingcartController will first have to ask the ModuleLoader to provide the
When the Products are retrieved, the ShoppingcartController can ask the View to display a detailed list of Products, with their respective quantities (retrieved from the model) and buttons to change the contents of the shopping cart. Upon clicking one of the generated buttons, the Controller will have to respond to the user by calling an edit method in the Model, and reloading the page.

In this case, the shopping cart module will require the product module to perform its required action. Still, if no product module is available, the shopping cart controller will simply ask the shopping cart view to display an empty shopping cart, thus keeping the modules capable of working individually.

---

**B: Class Diagram**

This class diagrams specifies how the different modules will communicate with each other. This class diagram is for communication purposes only, because the system will be build in models instead of classes, so this diagram only shows how the different modules will communicate with each other.
www.website4free.co.nz

Bachelor project IN3700

Plan of Action

Version 2.0
June/July 2007

Project Group
Remy Alidarso  1221736
Zdravko Anguelov  1232436
Terence Bahlen  1217313
Frits de Nijs  1224263

Company
SecureSpace
Business Manager:   Maria Veleva
Project Manager:   Dimitar Velev

University
Delft, University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science

Commission
Project Manager:   Ir. B.R. Sodoyer
Project Accompagnier:   J. de Vries
# Table of Contents

1. INTRODUCTION .................................................................................................................. 171

2. PROJECT DESCRIPTION .................................................................................................. 172  
   2.1. THE CLIENT ................................................................................................................... 172  
   2.2. CONTACT PERSONS ....................................................................................................... 172  
   2.3. PROJECT SCOPE ........................................................................................................... 173  
   2.4. PROJECT DESCRIPTION .............................................................................................. 173  
   2.5. TASK FORMULATION .................................................................................................. 173  
   2.6. DELIVERABLES ............................................................................................................ 174  
   2.7. BOUNDARY CONDITIONS .......................................................................................... 174  
   2.8. RISK FACTORS ........................................................................................................... 174  

3. APPROACH .......................................................................................................................... 175  
   3.1. METHODOLOGY ............................................................................................................ 175  
   3.2. USED TECHNIQUES ..................................................................................................... 175  
   3.3. ACTIVITIES .................................................................................................................. 175  
   3.4. PLANNING .................................................................................................................... 175  

4. PROJECT INSTITUTION AND CONDITIONS ................................................................... 176  
   4.1. INVOLVED PARTIES .................................................................................................... 176  
   4.2. INFORMATION ............................................................................................................ 176  
   4.3. FACILITIES ................................................................................................................... 176  

5. QUALITY CONTROL ......................................................................................................... 177  
   5.1. THE QUALITY .............................................................................................................. 177  
   5.2.1. Documentation ......................................................................................................... 177  
   5.2.2. Version control .......................................................................................................... 177  
   5.2.3. Evaluation ............................................................................................................... 177  

APPENDIXES ......................................................................................................................... 178  

APPENDIX A: PLANNING .................................................................................................... 178
1. Introduction

This document contains the plan of action for the Bachelor project (IN3700) for the Technical University Delft, carried out at SecureSpace in Auckland, New Zealand. This plan of action will specify how the development of the Bachelor project (websiteforfree.co.nz), will proceed.

SecureSpace (a company draft)

SecureSpace is a company which operates for five years, in Auckland, New Zealand, so far they have had 250 clients and they provided these clients with web design services. They specialize in the following functionality, in the e-commerce section:

- Online Shop,
- Payment Tracking Systems,
- MYOB Integration,
- and Business to Business.

They also specialize in a lot of other areas of which here are only named a few

- E-Mail Marketing,
- Content Management System,
- Banner Management System,
- Search Engine Optimization,
- Website Security,
- Mobile Phone Solutions (WAP),
- and Video Streaming.

So far the company consists of about ten employees, all specialist in different fields of web design and implementation. One of their main specialities is their Search Engine Optimization, as they describe this service:

“Having a beautiful and functional website is good. Making people visit it is what brings you results and puts CASH in YOUR pocket. This is what the Search Engine Optimization is all about”.

The Plan of Action

The reason why SecureSpace wanted this project is because of the current situation of the market on web design. There is a niche market, in which a few companies operate, that lets people or small businesses build their own website (for free) with tools provided by the companies. The customers in this corner of the market are individuals and/or small businesses who do not have the means to buy a complete web design from a web design company. However these individuals and small companies recognize the need for a website, therefore SecureSpace wants to offer them this opportunity without involving all the costs. The Technical University of Delft approved this assignment to be carried out by four students, for their Bachelor project.

The plan of action is split into five main sections. The project description will be presented in the second paragraph. The third paragraph will specify what course of action will be taken for the project to come to a successful development. The fourth paragraph will specify the management aspects of the project. The fifth paragraph will specify the project institution. The final paragraph will specify how the quality of the delivered project will be measured.
2. Project Description

This paragraph specifies the project description and how the system will be build. This is done (for a part) in a separate document, namely the Requirements Document in which all of these aspects are addressed and all the proposed design structures are explained. Therefore in this paragraph we will only specify, what is not already done in the Requirements Document.

2.1. The Client

The client of the project will be SecureSpace, because they are going to use the system on websiteforfree.co.nz, and the different modules (as specified in the Requirements Document) can in the future be used with the development of other websites. The Technical University of Delft can also be seen as a client since they are going to follow the development and grade the project once it is finished.

2.2. Contact Persons

Four main contact persons can be identified with the development of this project, namely the business manager of SecureSpace, the project manager at SecureSpace and the for the Technical University of Delft, the project manager and the project accompanier for abroad projects:

Business manager (SecureSpace):
Msc Maria Veleva
Phone: 094417027
Fax: 094417028
E-mail: admin@securespace.co.nz

Project Manager (SecureSpace):
MEng. Dimitar Velev
Phone: 094417027
Fax: 094417028
E-mail: admin@securespace.co.nz

Project Manager (TU Delft):
Ir. B.R. Sodoyer
Phone: +31 (0)15 27 86346
E-mail: B.R.Sodoyer@tudelft.nl

Project Accompanier (TU Delft abroad projects)
J. de Vries
Phone: +31 (0)15 27 85952
E-mail: J.deVries@tudelft.nl
2.3. Project scope

The formal project description of the project states the following: 
“to develop a system that will allow the users to register and build their own website base on a chosen template. Websiteforfree.co.nz will be the ultimate tool for easy and smart website creation.”

As the formal description specifies it should be very easy to build a website in as few steps as possible, this will be a crucial success factor, which will be explained further in section three. The supported chosen templates will be:

- article site;
- a simple e-commerce site;
- an advanced e-commerce site (has all the functionality of the simple e-commerce site, but with the following added functionality:
  - newsletter system
  - pay tracking system);
- and a media site.

These four templates identify the scope of the project as they are the four main goals which need to be developed, these four templates need to be developed in a way that more functionality can be added in a later stage and therefore the project will be developed in modules. (The modules will be build according to the model-view pattern, for more details see the Requirements Document).

2.4. Project Description

Most of the functionality is described in the Requirements Document, however in this paragraph the main features will be described again. These main features are:

- each template will have its own content management tool;
- each template has its own embedded WYSIWYG editor to edit the content of the template;
- depending on the chosen template the content management tool will be extended with several features (like newsletter system, pay tracking system).

This specifies the generated website main functionality; further specification is explained in the Requirements Document.

For the websiteforfree.co.nz it there is the functionality of the “state of the art administration and accounts system, with integrated credit card payment”. This is also further explained in the Requirements Document.

2.5. Task Formulation

For the University it will be an effort obligation, but for SecureSpace it will be a result obligation because they actually want to use the system, but they can not find the time to build it themselves. Therefore they approved that the 'internship' will set up the project and starts developing it. The most important thing for SecureSpace is that the project will be developed in modules which each can operate separately, so that they easily can add functionality in the future for other projects. The main responsibility of the 'internship' is to make sure that at the
end all of the four templates have the basic functionality and that it is build in modules. The four templates will first be developed with the following basic functionality:

- article site, which consists of a couple of pages (with static content);
- simple e-commerce site, which allows for products and a shopping cart;
- advanced e-commerce-site, which has the functionality of the previous site, but with added functionality like a newsletter and a payment tracking system;
- media site, which allows for posting of media and has a membership function.

2.6. Deliverables

The deliverables for the project for SecureSpace can be divided into two parts, each with different subparts. The two main parts are the functionality of the generated website and the websiteforfree.co.nz itself, which has as subparts the integrated payment system and the generating of the website. The generated website has a lot of subparts which will be the different modules, as specified in the Requirements Document. Most of the modules will specify the administration section of the generated website, from which they can control their entire website. The entire website will be build from what they are able to do in the administration section.

So to sum it up there are two main deliverables, namely the websiteforfree.co.nz (with the payment system) and the generated website (with all the different modules).

For the Technical University of Delft there are also a few deliverables, namely the plan of action and the main documentation. For starters we will start with the requirements document and the plan of action.

2.7. Boundary Conditions

For all the boundary conditions (on the requirements) we refer the reader to the Requirements Document.

2.8. Risk factors

At this stage of the development of the project we have identified the following risk factor, namely the time constraint; therefore some of the modules are set on hold (as can be seen in the Requirements Document) until there is time left to develop these modules.
3. Approach

This section of the plan of action specifies the methodology, together with the techniques and the activities which are going to be used in the approach for the project. Most of the approach paragraph is already specified in the Requirements Document; therefore this paragraph will not add many to what is already specified in the Requirements Document.

3.1. Methodology

As specified earlier the end result of the project should consist of modules, operating individually and being able to work as a unit (if put together). Therefore we will specify the methodology of the project by the different modules, first we will develop the views for the models and then we will implement the working of the modules. The focus in the early stages of the project will be placed on the following modules:

• generating a website, with a chosen ready-to-use template;
• WYISWYG content editor for the generated website;
• management tool for the article page;
• management tool for the products;
• management tool for the shopping cart;
• management tool for the newsletter service;
• management tool for the pay tracking service
• and a media module, which allows for uploading and downloading of media.

3.2. Used techniques

The techniques which are going to be used are completely specified in the Requirements Document.

3.3. Activities

The activities are specified in the Requirements Document.

3.4. Planning

For the planning we refer the reader to appendix A: the planning.
4. Project institution and conditions

This section of the plan of action specifies the project institution, by determining the involved parties for the development and the information stream for the project and the facilities which are going to be used.

4.1. Involved parties

The involved parties for the development of the project are the development team and the project manager (SecureSpace). The business manager also plays a role in the development of the project, however, the role of the business manager will be limited, since the business manager only wants the business model to be achieved.

The condition which applies to the project manager is the accompaniment of the development team. He is responsible to steer the development team in the right direction, especially in the beginning, so the goals can be achieved and the risk factors can be minimized.

The conditions which apply to the development team are quite extensive, because they are the core of the project, since they make the planning, implementation and the documentation. Most of the requirements for the are specified in the Requirements Document, but not the planning, this is specified in appendix A of this document.

4.2. Information

The information of the project will consist of two parts namely the part for university which will consist of three-weekly updates, which will specify the progress made and the deliverables met.

The other part of the information will consist of the documentation of the project for the company SecureSpace, this part is very important, because the students will not be available for further questioning once the project is finished. In order to achieve this, the project and especially the written code will be documented thoroughly so SecureSpace can make easy adjustments and additions in the future. Also after the software package a manual is made in which all functionality is explained and in which a user guide is included.

4.3. Facilities

The organisation regarding the project will consist of five members, namely the project manager and the development team. The development team has access to three macs and one pc, all with the required software installed. The software and hardware specifications are specified in the Requirements Document. For clarification we will mention them also in this paragraph.

The developed software package should work on macs as well as pc, therefore the project will be developed and tested on macs and on pc's. This specifies the hardware requirements for the project. The software considered in this project consists of MYSQL (the database) php (the main functionality will be made in php), JavaScript (which will specify additional functionality) and some flash functionality.
5. Quality Control

This section of the plan of action specifies the quality control of the project the emphasis will be on the documentation, version control and the evaluation of the project.

5.1. The Quality

This section will specify how the quality of the software package is going to be guaranteed, especially how SecureSpace can use it, since the development team will not be around anymore after the completion of project.

5.2.1. Documentation

The documentation consists of two parts, namely the documentation of the written code and the documentation of the design. The development team has identified these two documentation parts, since the code (and especially) the modules should be easy to use in the future and use to add adjustments. More details on the documentation can be found in the Requirements Document.

5.2.2. Version control

The development team discussed the use of version control with the project manager, but since the company SecureSpace does not use it, they make regular back-ups of their servers, from which the development take place, they decided not to use it, since they can make a roll-back with the back-ups from the servers.

5.2.3. Evaluation

The evaluation of the project is determined by the deliverables, (i.e. the different modules) and since the one of the main constraint is the user friendliness, this will be a main factor in the evaluation of the project. Some modules have more priority than others, therefore in the evaluation phase some modules have a higher priority than others. The modules mentioned in paragraph 3.2 will have a high priority since they are the core of the project.
Appendixes

Appendix A: Planning

The planning for the project consists of the full 12 week for which the project is scheduled. The basic planning is as follows:

In the first few weeks (1 or maybe 2) we setup the requirements document, action plan, and try to get an overview over the whole project. In this period we also get a lot of special trainings from the company we work at. This is to get used of the coding standards that the company currently uses so our project can be used after we have delivered it.

After this period, we begin with the different admin tools that are going to be used by the users of the service. There are quite a lot different modules that need to be developed, so this period can consist of several weeks of work.

When these modules for the generated websites are done, we begin with the main website, where people can subscribe to the service. This will not take a long time, because it is basically a static website with the addition of a payment tracking system, which we are going to build for the generated website as well.

Then the functionality for the actual generation of a website when users subscribe is going to be build. When this is done, the system as a whole is finished, and if there is time left we can improve the administration tools, or we can make other modules, that basically can be plugged into the system, because the whole system is being created in a modular way.

The planning per week is roughly the following:

- Week 1: Training + documents
- Week 2: Training + documents
- Week 3: Generating admin modules
- Week 4: Generating admin modules
- Week 5: Generating admin modules
- Week 6: Generating static content for websiteforfree.co.nz
- Week 7: Generating the layout for the generated websites (client side)
- Week 8: Generating the layout for the generated websites (client side)
- Week 9: Constructing the automatic website generator
- Week 10: Constructing the automatic website generator
- Week 11: Constructing the automatic website generator
- Week 12: Finishing up