HAAGA-HELIA
University of Applied Sciences

Website development project with Joomla 3 Content Management System

Jesper Ruuth

Bachelor's Thesis
Business Information Technology
October 2014

HAAGA-HELIA
University of Applied Sciences

Abstract

31.10.2014

Author
Jesper Ruuth

Degree programme
Business Information Technology

Thesis title
Website development project with Joomla 3 – Content Management System

Number of pages and appendix pages
41 + 2

Thesis advisor
Sauli Isonikkilä

Content management systems (CMS) are probably the most popular frameworks to build content rich websites that require extensive editorial tools. Joomla is currently the second most popular CMS and over the years it has become an extremely robust and secure platform thanks to its active and thriving developer community.

This thesis describes re-development process of a website called “Freedom for Sale”. The website was built with Joomla and the implementation included the entire development life-cycle including some special activities such as data cleansing and data migration. The project was commissioned by Art Films production Oy and it was carried out between September 2013 and April 2014.

The goal of Freedom for Sale is to promote human rights and freedom of speech by highlighting grievances and monitoring governments and multinational corporations, whose actions support or ignore violations of human rights and free speech. By providing videos, articles and reports the website provides a medium for the mistreated to get their voices heard and by so help the world’s development.

The website has a troublesome past. It was initially launched in 2007, but it was shut down after becoming a victim of harmful cyber-attacks, which corrupted majority of data and content in the process. Valid backups were not available to be used.

Besides describing the development process, this report describes the core concepts of CMSs, Joomla, information architecture and data cleansing. The thesis will also suggest a method how data cleansing and migration could be applied to Joomla with Excel or similar spreadsheet application.

Keywords
CMS, Data cleansing, Data migration, Information architecture, Joomla

Table of Contents

Abbreviations

Apache – Open source HTTP server application

CMS – Content management system

cPanel – Control panel application for web hosting

CSS – Style sheet language for markup documents (Cascading Style Sheets)

HTML – Standardized web page markup language (HyperText Markup Language)

LESS – Dynamic style sheet language

MVC – Software design pattern (Model–view–controller)

MySQL – Open source relational database management system

PHP – Server-side scripting language (PHP: Hypertext Preprocessor)

phpMyAdmin – Open source web application for MySQL database management

SQL – Standardized query language for databases (Structured Query Language)

XAMPP – Cross-platform Apache distribution containing Apache, MySQL, PHP and Perl

WAMP – Web development stack (Windows, Apache, MySQL and PHP)

WYSIWYG – Text editor that mimics final appereance (What You See is What You Get)

1 Introduction

1.1 Background

Freedom for Sale is a non-profit private organization and a website implementation project commissioned by a Finnish film production company, Art Films production Oy. Freedom for Sale's purpose is to focus attention on human rights and freedom of speech abuses by highlighting grievances and monitoring governments and multinational corporations, whose actions support or ignore these violations. The intention is to raise awareness and encourage open discussion, in an effort to have an impact on broad issues of common interest and ultimately the world's development. Freedom for Sale website provides hundreds of eye opening articles and other media content, which have been produced by numerous human rights activists and Art Films production itself.

The project's roots originate from 2007 when it was initially launched along with documentary film, Shadow of the Holy Book. Soon after being released, the website was forced to be put offline after becoming victim of regular and harmful cyber-attacks. According to previous developers, this was due to unstable and insecure commenting and community modules that allowed malicious scripts to bypass website's information security measures. It was also assessed, that the controversial content on the website made it more intriguing target for these attacks.

1.2 Objectives and tasks

The purpose of this thesis was to re-design, re-implement and re-launch Freedom for Sale website for the commissioning party. The main objective was to produce a dynamic and content rich website that would be easy to use and been built on top of secure and robust content management system. In addition, it was requested that new website should include all the previously produced content, which meant that the old content needed to be gathered and prepared for this implementation as well.

The tasks included in this thesis implementation were:

- Analyzing and specifying website's functional and non-functional requirements
- Installing and configuring the development environment and software
- Implementing data cleansing for the legacy data
- Designing website's information architecture and layout mock-ups
- Implementing data migration into Joomla CMS
- Building the website with Joomla CMS

- Testing the usability and functionality of components and layout
- Deploying website to a web server

1.3 Research topics

In addition to describing the website's implementation process, this report includes a short theory part, which covers the topics that were required during the implementation. These topics have been divided into separate chapters and they are:

- Content management systems (CMS) – What they are and how they function?
- Joomla CMS – Which are its core features and components?
- Information architecture – What it means, why it is required and how it is done?
- Data cleansing – What it means, why it is required and how it is done?

Moreover, some technical terms and topics that were relevant during the implementation (such as data migration) are covered in chapter 6.

1.4 Timing and scheduling

The development tasks described in this report were carried out between September 2013 and April 2014 and were implemented along with other studies and work. This report was written along with the development tasks and it was finished in October 2014.

2 Content Management Systems (CMS)

This chapter describes web based content management systems and their core features and components. The content is based on literature and online sources as well as author's personal experience with them.

According to US registered patent (US 6356903 B1), content management system (CMS) is an information delivery system for web based implementations that organize content of the information separately from the appearance of the presented information and by so allowing content creation and management to be done in a native format with familiar software tools (Baxter & Vogt 2002).

Simply put, CMSs are installable web applications that include a database and extensive set of tools dedicated for content creation, management and publication on internet. Their main purpose is to ease content management and publishing tasks on the web by providing tools that do not require extensive knowledge of the web markup languages or its protocols. (Shreves 2010, 3–4; TechTarget 2011a.)

Originally, CMSs were designed as tools for organizations to dispose static HTML websites and to simplify web publishing that required knowledge of HTML and other web standards. The traditional method was found impractical and expensive as publishing content required constant co-operation between the content contributors and web developers. A solution was achieved by developing a system that integrated all the elements of web publishing under single implementation. This innovation breached the barriers that had existed in traditional web publishing allowing it to become less technical and more streamlined. (Shereves 2010, 3–4.)

2.1 Key features and components

In software engineering, one will eventually come across with the terms front end and back end. Front end and back end are terms' which main purpose is to distinguishing the tasks related to presentation layers and data access layers. The front end is more concerned about the presentation logics while the back end is more concerned about the data access and business logics. Technically speaking, front end development involves mostly client-side coding, while the back end development involves mostly server-side coding.

When talked about CMSs, the terms front end and back end may refer to separate application areas as the CMS distributions basically contain two different applications. In

this case, the front end refers to the publicly visible website that serves as the medium for regular users while the back end refers to the administration application that is used for managing content and application's preferences by the system admins and editors. Nevertheless, both the public and administrator application contain front-end and back-end components as they both contain presentation and data access layers.

For the sake of clarity, this and the next chapters use the following naming conventions to distinguish these terms:

- Front end component refers to the presentation layer (Client-side code)
- Back end component refers to the data access layer (Server-side code)
- Administrator application refers to the back end application of CMSs (Administrative view)
- Public application refers to the front end application of CMSs (Public view)

2.1.1 Content Management Application (CMA)

Content Management Application (CMA) is the front end component of the administrator application. CMA is basically an administrative control panel with a graphical user-interface (GUI) that is used for managing content and assets, users, access rights, templates, extensions and other system preferences of websites. Back-end application is typically equipped with efficient and user-friendly tools such as WYSIWYG text editors that allow editing and publishing task to be done without having extensive technical knowledge of web standards and its protocols. Access to the CMA and its features are usually controlled with login and access control to ensure the integrity of content and security of the entire system. (TechTarget 2011b; Goodrich 2013.)

2.1.2 Content Delivery Application (CDA)

Content Delivery Application (CDA) is the back-end component of the administrator application. CDA is basically the component that handles all background operations while transacting data between the content repository and view by compiling user inputs made via CMA. (TechTarget 2011a; Goodrich 2013).

2.1.3 Template engine

Template engine is the CMS component dedicated for presenting information to users by attaching the content from data repository to a pre-defined layout template in order to generate the output for users. Basically, templates are layout definers and user-interface

element placeholders that do not hold actual content, but rather provide framework for presenting it. With the help of CMA, editors with minimal technical knowledge are able to edit and manage templates to some extent but more advanced editing or building one from scratch requires advanced knowledge from HTML, CSS and various other web standards. (Shreves 2010. 497–498.)

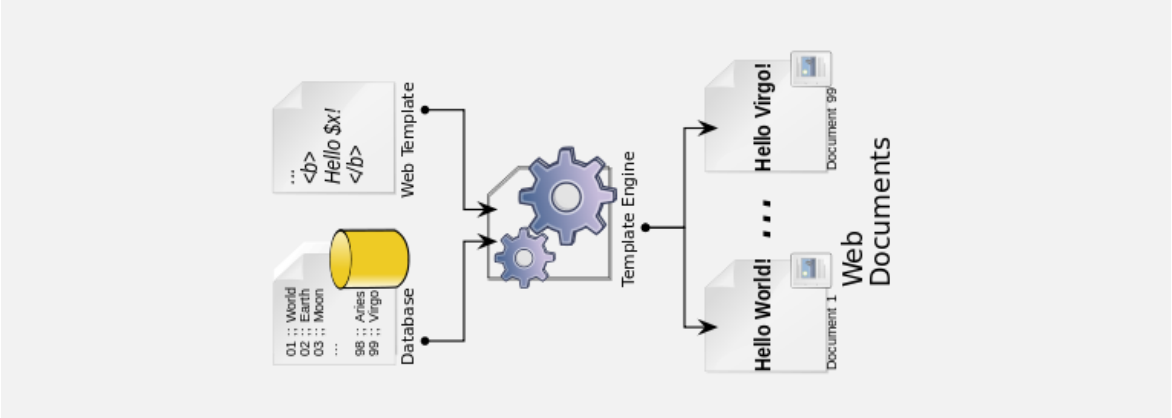


Diagram 1: Illustration of template engine's functionality (Wikipedia 2006).

3 Joomla

This chapter introduces Joomla CMS and its core features and components. The chapter does not describe Joomla's technical implementation in high-detail or compare Joomla to other CMSs. The content is based on the official Joomla documentation and other online sources as well as author's personal experience.

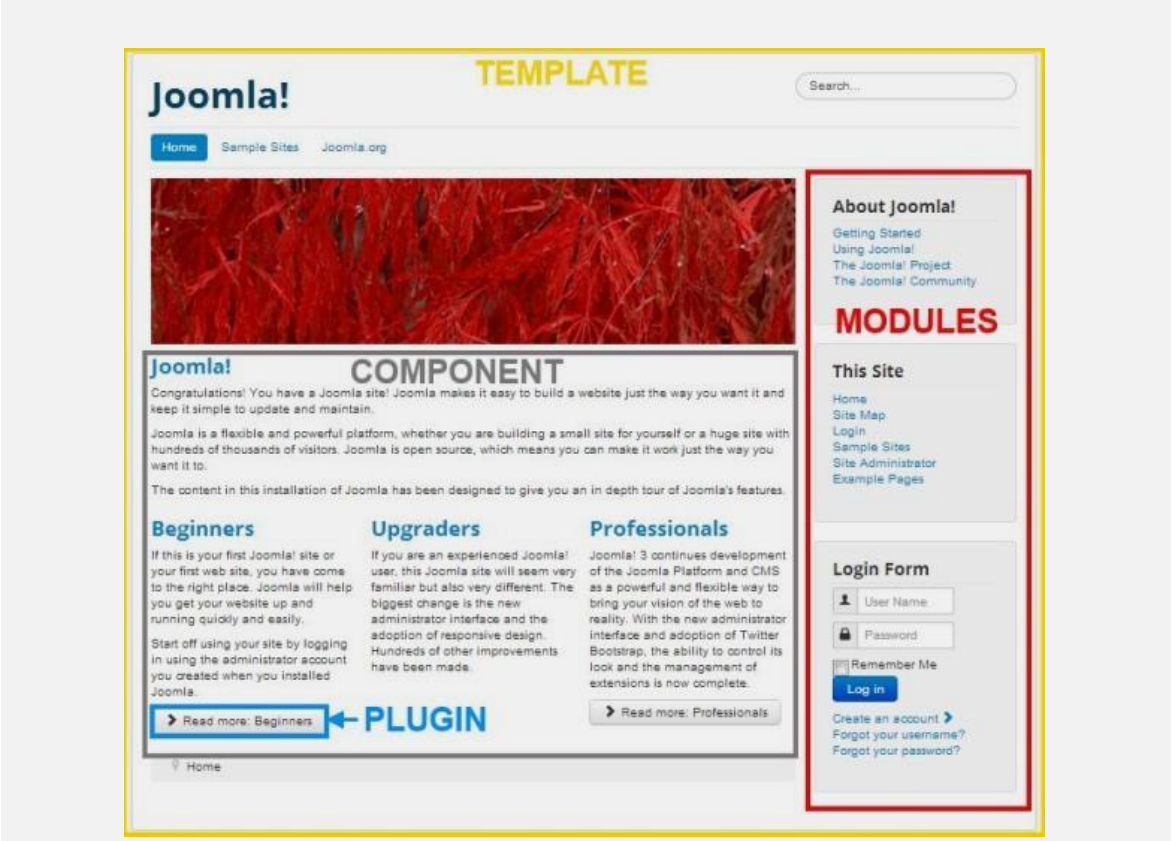
Joomla is a user-friendly open-source and community driven CMS that is built on top robust model-view-controller (MVC) framework. It is developed and maintained by The Joomla Project Team, a global developer community with contributors all over the world. (Joomla 2014a; Joomla 2014b.)

It is considered one of the best and most sophisticated CMS's available and according to a survey of W3Techs (2014), it is currently ranked as the second most popular CMS platform in the world.

3.1 Framework & Extensions

As mentioned, Joomla has been built on top of MVC framework which can be used to build stand-alone applications. Besides the framework, Joomla CMS is composed from a set of extensions which each have differing functions. Some of the extensions are part of the CMS core and are essential for proper functionality and stability of the system and thus come along with the default installation. Moreover, the core can be extended with various other extensions which can be purchased or downloaded for free via Joomla Extension Directory which is a centralized distribution channel for all types of Joomla extensions. (Joomla! Docs 2013a.)

Extensions are software packages that extend the default Joomla installation in some way. It is relevant to understand that extension term is generic and that it means all types of extensions such as components, modules, plugins and templates which each are meant for different purposes. (Joomla! Docs 2014a.)



Picture 1: Joomla web page visualizing the different extension types (Joomla 2013d).

3.1.1 Components

Components are the most sophisticated extensions that provide the main functionality to Joomla system. They can be referred as mini-applications that generate output for different parts such as the main content section of each web page. (Joomla! Docs 2014b; Shreves 2010, 547.)

Components are triggered by HTTP requests and they execute series of operations within the framework which ultimately leads to generating the output. For example, an article view component, com_content, performs all the actions required from fetching the data from database to rendering article's HTML document. The framework provides abstract classes for model (JModel), view (JView) and controller (JController) which the components extends to in order to have standardized architecture. (Joomla! Docs 2014b; Shreves 2010, 547–548.)

Joomla components are composed from site part (front-end application) and administrator part (back-end application). The site part is used for rendering page content for the front-end application while the administrator part provides a user-interface for the back-end

application for configuring components' preferences. (Joomla! Docs 2014b; Shreves 2010, 547)

3.1.2 Modules

Modules are customizable page rendering blocks that are used for displaying specific information in a specific area. Modules can be associated to components when they inherit their functionality or they can be independent blocks that render content the way specified within the module. (Joomla! Docs 2014c.)

Modules are assigned to pages via navigation menu items so they can be shown or hidden depending on the page in question. The positioning of the module is entirely dependent on the used template's module placeholders but it is possible to position a module into any placeholder that template in question provides. Modules are also divided into front end and back end parts where the back-end part provides an user-interface via Module Manager where they can be freely configured. (Joomla! Docs 2014c.)

3.1.3 Plugins

Plugins are extensions that serve as event handlers and helper applications. They are triggered by certain events and they respond to it with procedures that can be visible or esoteric. Plugins are mainly used for extending the functionality of components and they are built with observer design pattern where the dispatcher notifies all the associated plugins allowing them to be executed in sequence. (Joomla! Docs 2014d; Joomla! Docs 2013b; Shreves 2010, 543, 569.)

3.1.4 Templates

Templates are extensions that control the overall layout and visual presentation of content. They define how various page elements should be structured and how the content should be rendered within them although some components and modules may contain their own styling rules. The purpose is to separate the visual presentation from the actual content allowing more efficient website maintenance with possibility to apply consistent layout throughout the website. (Joomla! Docs 2013c; Shreves 2010, 497)

Joomla provides separate templates for both front-end and back-end applications. Templates can be installed or built from scratch and they can also be configured via back-

end application using the Template Manager. (Joomla! Docs 2014e; Joomla! Docs 2013c.)

3.2 Security

Joomla is considered to be a very secure CMS. Nevertheless, as being an open-source platform it is also a common target for various cyber-attacks and breaching attempts. (Joomla Security Info 2014; SiteGround 2014.)

“Keeping your site patched and up to date is one of the keys to maintaining your site’s integrity and protecting it against hackers” (Shreves 2010, 685).

Joomla documentation (Joomla! Docs 2014f; Joomla! Docs 2012) suggests few general guidelines on how to improve the security of Joomla installation. These guidelines include the following procedures:

- Back up early and often
- Install Joomla updates when they are released
- Use a trustworthy and secure hosting provider
- Use strong usernames and passwords for login
- Do not trust third party extensions
- Use an offline environment to test extensions before applying them to live site
- Do not use the default security settings
- Use the community to get help in security matters

In addition to above mentioned, it is considered as a good practice to use the SEF (Search Engine Friendly) component to rewrite URLs, to use proper file permissions and ownerships on server side and to use some safe third party security extensions that protect the site from different types of attacks. (SiteGround 2014; Joomla! Docs 2014f.)

3.3 Technical requirements

By the time of writing this report, the latest stable release of Joomla is 3.3 and it requires PHP version 5.3 or higher, MySQL, SQL Server or Postgre SQL database and Apache 2.0, Nginx 1.0 or Microsoft IIS 7 as the webserver (Joomla.org 2014c).

4 Information Architecture

This chapter describes what is meant with an information architecture and what kind of methods can be applied at its design process. The content is based on online sources.

According to Morville and Rosenfeld (2006, 4), information architecture is:

- A structural design of shared information environments.
- The combination of organization, labeling, search, and navigation systems within web sites and intranets.
- The art and science of shaping information products and experiences to support usability and findability.
- An emerging discipline and community of practice focused on bringing principles of design and architecture to the digital landscape.

Basically, information architecture (IA) is information system's structural solution for structuring, organizing and categorizing content in effective and sustainable way. The aim of it is to establish logical, clear and consistent structures and paths towards content so that the information can be found easily by any user. (Usability.gov 2014a.)

Information architecture is a broad concept including multiple aspects, tasks and purposes. In order to fully understand it, it is crucial to realize the overall structure in a big picture and how the pieces of information are connected to form a network of information and how they are related to each other. Designing information architecture is a constant balancing between the needs of independent users, diversity of content and context. In addition to users' needs, the content should also be structured for search engines and web crawlers making the task even more challenging. (Usability.gov 2014a & Morville 2012.)

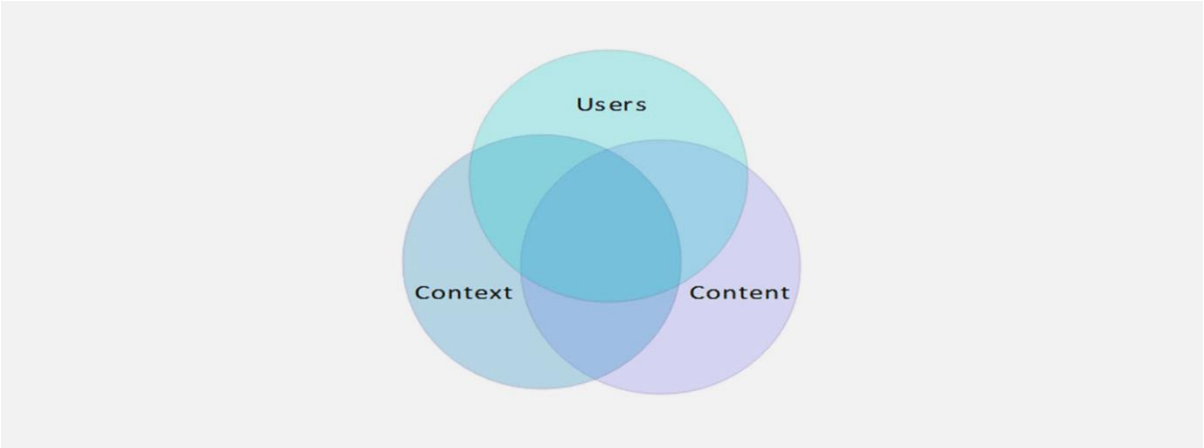


Diagram 2. The three circles of information architecture according to Rosenfeld and Morville (Usability.gov 2014a).

4.1 Methods and techniques

Information architecture contains elements from various specialty areas. It involves many activities that are more or less related to topics such as usability, user experience (UX), layout and user-interface design. To be successful it requires a good understanding of industry and organizational standards as well as recognizing how users navigate and use information systems in general. (Usability.gov 2014a.)

Morville and Rosenfeld state that information architecture is composed from organization systems, navigation systems, search systems and labeling systems (Morville & Rosenfeld 2006. 43).

Organization systems describe how the content is organized and categorized. Usability.gov suggests that organization systems can be further divided into schemes and structures where schemes are used for categorizing content and structures for forming relationships and hierarchies between them. (Usability.gov 2014b.)

4.1.1 Organization Schemes

Organization schemes can be exact or subjective depending on the approach. Exact schemes mean organizing content objectively in a way that it is commonly known and can be easily sorted. Such means include alphabetical and chronological ordering. Subjective schemes mean the opposite as it focus on organizing content into subjective categories defined by organizations. These categories can be for example a certain topic, a tag word or audience. The advantage of subjective mean is that the user is more likely to find related content of the subject while the exact schemes on the other hand are more predictable and so the user is likely to understand the logic of organization scheme more quickly. (Usability.gov 2014c .)

4.1.2 Organization Structures

Organization structures is a about defining relationships between contents and schemes. Well implemented structures are predictable and help users to clearly understand the overall structuring and categorization and so make navigation a lot easier. The organizational structures can be hierarchical, sequential or in a form of matrix. (Usability.gov 2014d.)

The purpose of hierarchical structures is to establish a clear hierarchy among content. The essence of hierarchical structure is in parent-child relationships where parents cover broader sections while the children provide more detailed and targeted content. The hierarchical structure is often visualized with a tree chart involving a root and numerous branches. (Usability.gov 2014d.)

Sequential structures are line like structures that include continuity of procedures until reaching the end. Sequential structures are typically used when providing step-by-step guidance such as filling registration form or conducting online shopping. Sequential structure assume that the next piece of content is somehow dependent on the previous content and so there is clear ordering among them. (Usability.gov 2014d.)

When designing organization structures, it is important to design them flexible. Information architecture is usually established with a long term perspective so it is likely that the amount of content will grow or the organizational needs change. The structures should neither be too broad nor confined as the content might get lost if there are too few or too many levels between the user and content. (Usability.gov 2014d.)

4.1.3 Content inventory

Content inventory is an inventory of all content that is included within the information system. Content inventories are tools for designers and information architects to understand the content in a larger scale and by so provide insights whether all relevant content is gathered and properly organized. In addition, content inventories give insights whether the essential metadata has been assigned to each content item. (Usability.gov 2014e.)

4.1.4 Wireframing

Wireframing is technique that can be used for layout design as well as for information architecture. Wireframe is a two-dimensional mock-up of the layout and by so it visualizes the final implementation. The benefits of wireframing are that it demonstrates early on how the users will see the content and how the used information architecture fits in. It basically gives insights whether it is functional or not. Wireframe also demonstrates the space allocation and how the content can be positioned within the limits of displays providing valuable information for making the final design decisions. (Usability.gov 2014f.)

5 Data Cleansing

This chapter defines what is meant with data cleansing, why it is required and what procedures are involved in it. The content is based on the source material and personal experience.

Data cleansing is a multi-stage process where inaccurate, inconsistent or irrelevant data records, also known as dirty data, are corrected, sanitized or removed from data storage to improve consistency and overall quality of data. It is required when dirty data affects the functionality, performance or usability of the system or degrades user experience in any way. It is procedure that should not be neglected and it is considered as an important aspect of any system maintenance and its results benefit both system administrators and end users. (Chapman 2005. 1–2.)

Dirty data is common and expected. It is cause from human mistakes such as typos or application errors existing in the source code and so can never be avoided. Clean data is essential since data that is not correct, representative or presentable for its purpose is practically meaningless. (Chapman 2005 & Dongre 2004.)

5.1 Methods and techniques

Data cleansing can be an independent process or part of a larger implementation such as data conversion and integration project. It usually involves several stages in which data is profiled, evaluated and cleansed with various routines and procedures. The implementation of these stages varies as they are affected by the quality and quantity of the data in question as well as the requirements specified by an organization or system. (Dongre 2004).

Data cleansing can rarely be a fully automated process and manual cleansing is practically always required to some extent. This is due to fact that all error types and inconsistencies cannot be solved with computational means. Finding and fixing these problems can make data cleansing an extremely tedious and time consuming process which nevertheless should not be ignored. (Dongre 2004.)

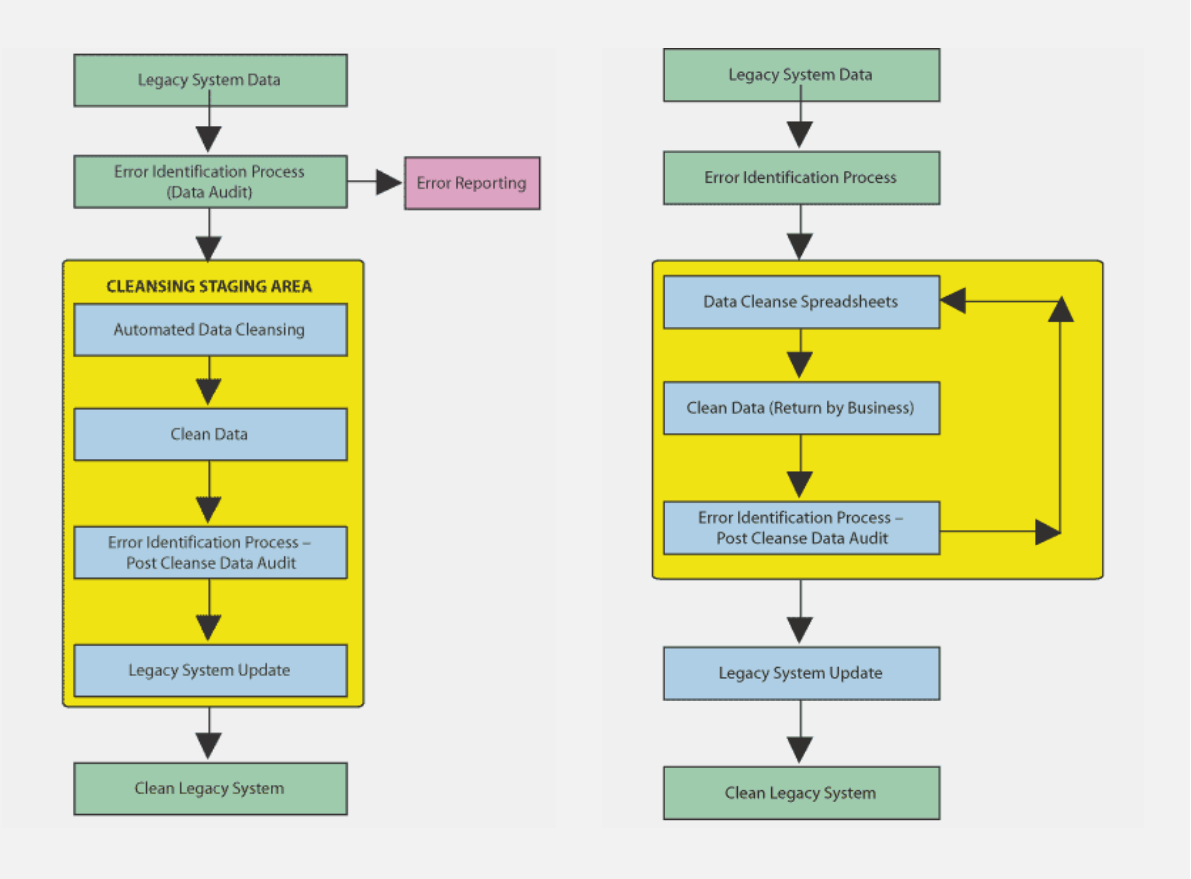


Diagram 3. An example of workflow of automated cleansing process (left) and manual cleansing process (right) (Dongre 2004).

According to Dongre, the quality of data can be measured objectively or subjectively and it pertains to issues such as accuracy, integrity, cleanliness, correctness, completeness and consistency (Dongre 2004).

Before actual cleansing activities are started, it is important to define certain quality standards for the clean data to help in future evaluation. In case of new system implementation, it is also important to consider the requirements of the target system while establishing these standards. Especially, the target system's database solution and data modeling practices should be thoroughly analyzed before defining any standards. (Dongre 2004.)

5.1.1 Staging

After the data quality requirements have been specified, a good practice is to implement so called data staging area for the subsequent cleansing process. This done in order to avoid irreversible, accidental, incorrect changes that could not be rolled back .The staging area can be a separate database or spreadsheet file where all data is centralized, isolated

and organized into easily manageable form for further analysis and processing. (Dongre 2004.)

5.1.2 Auditing

Next work phase is data evaluation and error inspection or so called data audit. The purpose of data audit is to evaluate the overall quality of data against previously specified quality requirements and to find any errors or inconsistencies among the data set. All errors and error types should be documented so that they can be traced independently later on. (Dongre 2004.)

After all data has been profiled and evaluated, the data can be rearranged according to found errors and inconsistencies. With this method the dirty data can be isolated from other clean data to avoid the risk of accidentally processing already cleansed data. (Dongre 2004.)

Data audit can be conducted by using computational methods such as using search functions to find certain condition within data and counting its volume or by simply reviewing them manually. There are also various software suites and solutions designed for data auditing and cleansing but their applicability was not tested in this thesis. In addition, data auditing requires certain logical thinking, deep understanding of the data in question and skills of creative problem solving.

5.1.3 Cleansing

The actual data cleansing is done by executing series of automated and manual routines and procedures to clean as many data inconsistences as possible. The computational routines applied for data auditing can also be utilized in data cleansing using them reversely. Data may often require several cleansing procedures and iterations before it fulfills the quality standards defined for it. For this reason, it is a good practice to split the cleansing process into several phases, making it more efficient and manageable. When the cleansing process is completed, it is recommended to re-evaluate and audit the data in case further processing is required. (Dongre 2004.)

6 Project Implementation

This chapter describes the implementation process of the Freedom for Sale website by describing the work phases, used methods and their appliance. In addition, the outcomes of each phase are analyzed and reviewed in regard of their successfulness and impact for this project.

6.1 Requirements analysis and specification

The project was started with requirements analysis and specification. The purpose of this work phase was to define website's functional and non-functional requirements and to decide the most essential guidelines that should be used in subsequent design and development work phases.

Requirements analysis was conducted by having discussions with the commissioning party about project's origins, goals and future plans, as well as analyzing the shortcomings of previous implementation. The analysis also contained a short research, where websites of similar topic were reviewed and analyzed in order to discover commonly applied design practices and to spark inspiration.

Because the commissioning party did not have any specific requests for the technical or aesthetical implementation, the requirements were specified mainly independently without commissioning party's active participation or contribution.

In the beginning, the requirements were specified in high-level, because the knowledge about the project and its content was too narrow at the time. Because of this, requirements were kept under ongoing inspection throughout the project implementation and were updated whenever they required changing.

6.2 Preparing the development environment

6.2.1 Why Joomla was chosen?

While specifying the requirements, it was agreed that the website should be built with a CMS. This was due to fact that the website would ultimately require various content management and publishing tools which usage should not require extensive technical knowhow. Joomla CMS in particular was chosen among other potential candidates simply because the commissioning party's previous websites had been built with it. This meant

that the commissioning party's staff was already familiar with Joomla's features and functionality and by so, did not require further consultation.

6.2.2 Installing XAMPP for Windows

Since Joomla requires a web server that supports PHP and MySQL, one was required to be configured in order to start the development tasks. During the development phase, Joomla website and database was hosted on local computer that was integrated with a WAMP (Windows, Apache, MySQL and PHP) development stack.

This environment was prepared by installing XAMPP, a cross-platform Apache distribution that contains all the components (Apache, MySQL, PHP and PERL) that are required to run a Joomla website. XAMPP was chosen, because it was suggested by the Joomla documentation and it was considered the fastest way to set up the environment for a Windows operating system. (Joomla! Docs 2013e)

The installation of XAMPP was rather easy and straight-forward process as it was done with a step-by-step installer wizard. After the installation, Apache's configuration took some effort, since the default port number required changing. By default XAMPP installation for Windows assigns Apache's port number to 80, which in this case was already reserved by some other application. The port number was changed to 81 via Apache's httpd.conf file that was residing beneath the "xampp/apache/conf/" directory path. Finally, the functionality of XAMPP installation was tested by accessing localhost via web browser.

In this implementation, XAMPP version 1.8.1 was used containing Apache 2.4.3, MySQL 5.5.27 and PHP 5.4.7. All installations were done for Windows 8 operating system.

6.2.3 Installing Joomla on localhost

After the localhost had been configured, the installation of Joomla was started. The installation was done in two phases. In the first phase, Joomla's installation package was downloaded via Joomla's official website and the package's contents were extracted into a new directory on localhost. In XAMPP setup, a proper directory path for websites is "xampp/htdocs/", where each website should be assigned to separate directory. All websites residing under "htdocs" can then be accessed via browser by entering the URL address of the localhost or alternatively using the computer's IP address. The localhost's URL is formatted as "http://localhost:81/website/", where "81" refers to Apache's web

server's port number and "website" to the namespace of website's directory. A good convention is to use short and easy to remember namespaces, because it is just temporary and it is typed often during the development.

In the next phase, the initialization of Joomla CMS was done with Joomla's browser based installation application, which was launched, when the website was accessed for the very first time. The installation was a step-by-step process, where the website's basic information and settings such as website's name, description and database preferences where chosen and configured. After the installation process was completed, Joomla deleted the contents of the installation application automatically due to security reasons and after that, Joomla CMS was set up and ready to be used.

In this implementation, the website was built for Joomla 3.2.1 CMS.

6.2.4 Other software installations

In addition to XAMPP and Joomla, also the following software was installed and/or used in order to be able to complete this project:

- Microsoft Expression Design 4 – Graphics design software that was used for designing the layout mock-ups
- Microsoft Word 2010 – Word processor software that was used for all documentation tasks
- Microsoft Excel 2010 – Spreadsheet software that was used for data cleansing and data migration tasks
- Microsoft Visio 2010 – Flowchart drawing software that was used for designing the information architecture and workflow documents
- Microsoft Project 2010 – Project management software that was used for project management activities
- MySQL Workbench 6.0 – MySQL database management system that was used for managing the Joomla's MySQL database
- NetBeans IDE 7.3 – Integrated development environment that was used for debugging Joomla and various PHP, HTML and CSS coding activities
- Notepad++ 6.3 – Simple text editor that was used for small coding and editing tasks
- FileZilla 3.7 – FTP client application that was used for deploying the website to web server

6.3 Data Cleansing with Excel

Before the previous website was put offline, the developers had managed to export backup of the database as a dump file. This dump file included a structure of the articles table containing article texts and their metadata. By analyzing the contents of this dump file, it was discovered that majority of the data had been corrupted and contained irregularities and invalid HTML notation. These errors and inconsistencies were assumed to be cause from malicious SQL injections caused by the cyber-attacks, but this could not be confirmed from the developers.

The specified data quality requirements included the following conditions:

- All articles should be relevant for the website and present time
- All articles should be properly formatted and have a consistent presentation structure
- All articles should be associated with a relevant categories, tag words and other metadata
- All articles should contain an acknowledgement paragraph to original source
- All articles should contain a hyperlink directing to original source
- All articles should have an accurate publication dates
- All articles should include a proper and validated HTML notation
- All articles should contain proper line breaks and paragraph spacing
- All articles should not contain useless HTML markups such as excessive
 tags

By validating the legacy data against these requirements, it was confirmed that extensive data cleansing procedures were required, in order to have accurate and solid data available for the new implementation.

6.3.1 Choosing and preparing the data staging environment

Data cleansing was started by creating an isolated data staging environment, where data could be freely manipulated without risking any other implementations or original dump file. The staging process was started by transforming the contents of the database dump file into more practical format. This was required because the dump file was a SQL script file, where the long article texts and metadata were embedded inside SQL insert statements, which caused them to be in impractical format for proper reading and analyzing.

At first, MySQL database and MySQL Workbench database manager was tested as a staging solution, since the contents of the dump file could be imported there easily. The importation was done by copy-pasting and executing the SQL statements into an empty database via query window. An alternate solution would have been importing the script file directly via Workbench's file import feature. After the data had been successfully imported, it was possible to read and analyze it more practically and intuitively.

While analyzing the data, it was discovered that the volume of errors and inconsistencies within the data was larger than expected. In addition, it was discovered that there was lots of variance between error types, where some would require manual cleansing procedures. For these tasks, MySQL Workbench was considered a bit impractical, so another staging environment was decided to be tested.

After pondering and testing different options, it was decided that Microsoft Excel 2010 would be used as final staging solution, since it contained several efficient functions for automated data cleansing as well as an intuitive user interface for rapid manual editing tasks. Also the support for numerous file formats and the possibility to use multiple spreadsheets tabs inside a single document was considered as a benefit.

First, the data needed to be transformed once more and this time it was achieved by simply copy-pasting the data rows from MySQL Workbench's table view into an empty spreadsheet. Before continuing, it was also checked that copy-pasting would not crop out any text parts. After this was confirmed, a new spreadsheet tab was created in which the structure of Joomla database's "#__content" table ("**#__**" refers to table prefix), containing all articles, was mimicked in order to have the similar table structure within the spreadsheet as well.

6.3.2 Web scraping new articles with copy-pasting technique

Since it had been several years since the last content update, new articles were also required in order to fulfill the requirement which specified that articles should relevant for present time. Because these new articles had not been gathered, new articles were collected by applying manual copy-paste web scraping technique.

Web scraping means the process of collecting information from third party web pages. Web scraping can be done by applying different methods, but usually it is done automatically by a scraping application or user defined collection scripts (Techopedia).

Before the scraping was initialized, once again a new spreadsheet tab was created to clearly distinct old articles from newer ones. This spreadsheet was also structured according to Joomla database's "#__content" table.

The articles were scraped from co-operative organizations' websites and it was conducted manually by copy-pasting the body text, title, lead paragraph, author, publication date, source and URL address into separate columns. In the beginning, the method was rather laborious and slow, but it turned faster once it became routine. A good practice was to arrange two screen displays side-by-side, where the source and spreadsheet were organized as parallel windows. This method significantly reduced the time of switching views on each copy-paste.

Automated scraping tools or techniques were not used in this implementation, because it was essential to analyze the relevancy of article's content before the copying was done. Also the fact that several information sources were used meant that each source would have required an individual scraping routine due to having different DOM structures, and by so it was considered rather impractical considering the amount of articles to be scraped from one source.

6.3.3 Conducting the data audit and error inspection with Excel

Before starting any cleansing procedures, it was important to assess and document the location, volume and variance of errors and error types. This so called data audit was done so that the errors and inconsistencies could be identified, prioritized and categorized in order to help the process of planning and applying cleansing procedures.

Data audit was conducted by reviewing data items against the data quality requirements by applying both manual and automated error inspection methods. At first, all data items were browsed through and each time a certain error or inconsistency occurred, it was documented and that data item flagged so that it could be traced back later on. Finally, the occurrence of specific error types were confirmed by using Excel's search formulas, which used the error as a search parameter and returned the value if the error was found. In addition, by using the calculation formulas it was possible to count the occurrences of each error type allowing them to be prioritized according to volume.

During the data audit, the following errors were identified:

- Irregular paragraph spacing and line break notation

- Missing <p> tags for article paragraphs
- Missing sources, lead paragraphs and publication dates
- Missing <ul>, <ol> and <li> tags for lists
- Excessive and useless line break notations:
, `\r and \n`
- Invalid hyperlink notation and link rot: @start_link@, @end_link@
- Invalid bold text notation: @start_bold@, @end_bold@
- Invalid heading notation: @start_title@, @end_title@ and @start_header@, @end_header@
- Invalid line-break notation: @break@
- Invalid use of special and non-western characters
- Other irregularities that require manual cleansing

By identifying these errors and error types, it was possible to invent the automated cleansing formulas, in order to minimize the necessity for manual cleansing. Also by identifying the errors for each data item, it was possible to run cleansing formulas as a batch process for similar error types.

6.3.4 Applying cleansing procedures by using combined method

The actual data cleansing was conducted by applying a combined method, which included both automated and manual cleansing routines. The goal of this work phase was to improve the overall quality of data, so that it would match the specified data quality requirements and be ready for later data migration. A personal goal for this phase was to be able to clean as many errors and inconsistencies as possible by using automated means and applying manual routines only when an automated routine could not be applied or if the cleansing case was isolated.

Before the cleansing was started, it was important to plan the order of cleansing procedures to help in management of this process. This was mainly because data cleansing can involve several stages and require multiple iterations before being completed. Thus, it is also important to monitor the progress of the process and to establish clear ordering between cleansing procedures to avoid overlaps and ensure correct outcomes. The order of procedures was done by creating a flowchart of cleansing procedures with Microsoft Visio 2010.

6.3.4.1 Running automated cleansing procedures

The automated data cleansing was conducted by applying Excel formulas and cell references. In the beginning, the spreadsheet was organized so that the data column containing the uncleansed articles was specified as the data source. From this column, the data was transferred into next column by using cell references. During this transfer, the data was partly cleansed with pre-defined cleansing formulas, which were associated to all data cells in that column. The partly cleansed data was next transferred into next column, which contained some other formula and this process was repeated until all cleansing procedures had been applied for all data rows. By using this method, the original data was never overwritten which allowed reversal to previous cleansing stages.

In general, Excel formulas were really useful for data cleansing procedures. Especially the SUBSTITUTE -text function, that substitutes parts of some old text with some new text, was considered to be very efficient, since quite often the case was that some text part was wanted to be replaced or removed, which the SUBSTITUTE formula was able to handle efficiently. For example, the SUBSTITUTE text function allowed the correction of invalid HTML notations, removal of unnecessary line breaks and transformation of non-western and special characters into valid HTML entities. By being able to do so, majority of the data could be cleansed automatically in batch runs. Nevertheless, several cases still required manual cleansing procedures in order to be considered clean.

6.3.4.2 Running manual cleansing procedures

Since all error types and inconsistencies could not be solved with computational means, manual cleansing procedures were also required. Because the data items that required manual cleansing were predefined and flagged during the data audit phase, they did not need to be identified and the cleansing could be started right away.

Manual cleansing was extremely laborious process that required precision and constant focus to avoid typos and other additional errors. The most laborious turned out to be the hyperlinks, since they often contained broken or dead links where the URL address was often mixed with the link's title text.

6.3.4.3 Concluding the cleansing activities

Once the planned cleansing procedures had been applied, the quality of data and successfulness of cleansing procedures were tested by creating a small web application

that printed all articles on a single web page. This was done by copying the cleansed data columns into new spreadsheet and exporting it as CSV file. Based on this spreadsheet, a new database was created by using the localhost's PHP MyAdmin tool to import the contents of the spreadsheet into the database. Next, the application for printing the articles was written in PHP, which fetched the articles from the database and put the results into array, which was looped through and printed on separate section blocks on a single page. Finally, the quality of data was confirmed by browsing the page's content and reviewing the source code, and checking that each article's HTML notation worked properly.

In overall, the data cleansing was very laborious work phase, which took, along with the data migration, most of time used for this project. Off-the-shelf software suites designed for data cleansing were not used, because their functionality for this context could not be confirmed and there was no experience from their usage. In addition, it was in the interest to research how traditional spreadsheet software, such as Excel, could perform this type of data cleansing activities.

In overall, Excel was considered to be quite good solution for this implementation due to containing many efficient formulas and excellent cell reference features. Nevertheless, more illustrative and practical presentation and editing features for long article texts would have been preferred.

6.4 Designing the information architecture

After the articles and other data had been cleansed and properly organized, designing of website's information architecture was started. The goal of this work phase was to produce consistent and logical categorization conventions for contents and to decide how the website's navigation structure should be organized. This work phase included the following tasks:

- Defining categorization schemas and naming conventions
- Organizing and categorizing articles and other content
- Organizing content in hierarchical structure and producing the sitemap
- Choosing the media content for the websites (photos, videos etc.)
- Choosing the navigation system for the website

Before any categorization could be made, it was crucial to get familiar on how Joomla organizes and stores its content. Especially important was to research which database tables are essential for categorization and which components use these tables for

displaying content. This research was done by reading Joomla's documentation and by analyzing Joomla's database schema, table relations and their contents in practice. In addition to this work phase, the information gained from this research, was found extremely useful later on, when conducting the data migration.

In Joomla, content is organized into categories. Each category can then contain sub-categories, which establishes clear hierarchical structure between different categories. Organizing content into categories contains one problem though, since by default, one article can only belong to one category. For some implementations, this can make categorization with categories a bit inflexible. For this reason, in this implementation content was categorized by using tag words, which could be defined as many as liked. (Joomla! Docs 2013g.)

Content's categorization was started by organizing the content inventory. The Excel spreadsheet created during data cleansing served this purpose well, since it already contained all the content, which would be required in this implementation.

In this implementation, the content was organized under three-level structure by using subjective categorization schemes. In the beginning, all content was organized under main a category, because in Joomla, all articles must belong to some category. After this, each article was categorized under three main groups, depending on which topic they dealt with. Because these groups were organized by using tag words, each article could be assigned to more than one group. These groups also formed the second layer in main navigation system. Finally, each main group was assigned with a number of relevant tag words as sub-groups, forming the third level in navigation system.

The implementation of this work phase required extensive awareness of the content since the categorization was based on subjective schemas. In addition, the convention of good collective and accurate categorization terms was found challenging.

6.5 Designing the layout mock-ups

Designing of the layout was done in order to build an early mock-up of the website that would illustrate the final presentation by excluding all underlying functionality. This was done in order to simulate website's visual presentation in an early stage and by so help the process of building the website's template. In addition, building an early mock-up confirmed that the requirements regarding to layout would be considered in final design.

Instead of using the simplistic wireframing technique, mock-up method was chosen, because it could demonstrate the final presentation more accurately and aid the process of choosing between different design practices.

The website's layout and layout design requirements specified that:

- The website should have a home page for featured and latest articles
- The website should have subpages for an about page, categories, articles and videos
- The layout should have a static header block for displaying logo, navigation and search bar
- The layout should have a static footer block for displaying partner organizations
- The layout should have a dynamic body blocks displaying dynamic content
- The layout should be stylish that emphasizes content
- The layout should contain familiar and commonly known user-interface elements
- The layout should contain aesthetic but easy-to-read font stack
- The layout should be responsive and it should function well on every display
- The layout design should be made with modern design practices

The layout mock-up was built by using Microsoft Expression Design 4, a graphics design software, which was chosen due to previous experience. Expression Design has many similarities with Adobe's well-known Photoshop software suite, although it contains less sophisticated features. Expression Design's downsides also are that its development was discontinued in 2012 and it is only available for Windows operating systems. Nevertheless, Expression Design is still very efficient and capable graphics design software for lightweight web design purposes. (Microsoft 2014.)

Before continuing into drawing the layout elements, a proper design document was setup. First of all, it is a good practice to use pixels as the measuring unit instead of millimeters or inches, when web design is in question. This is mainly because digital media uses pixels to define and measure contents at their screen displays. (Webopedia; Web Designer Depot 2010.)

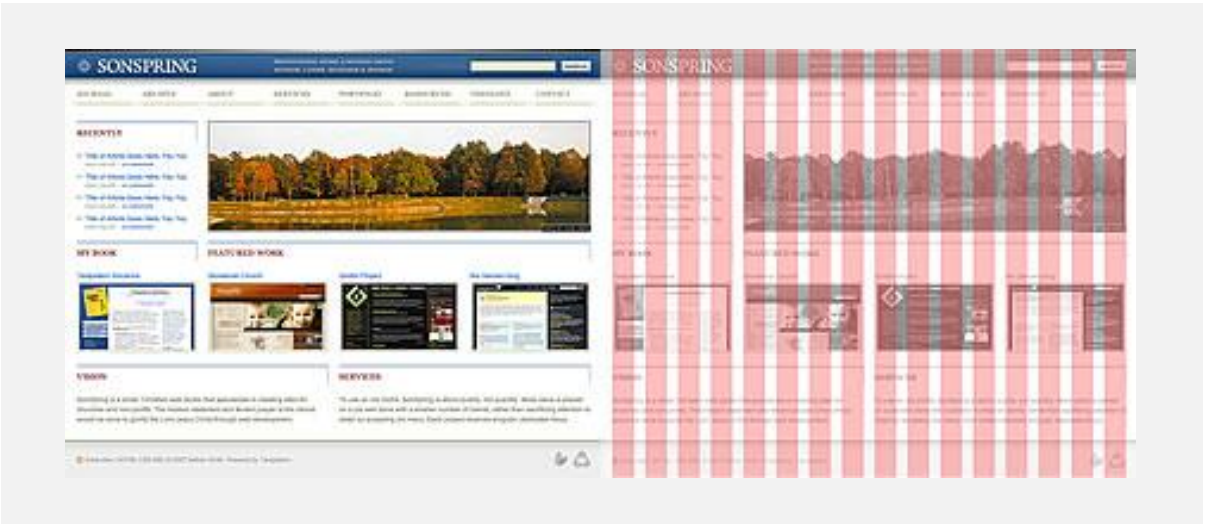
The dimensions of the design document are not absolute in Expression Design 4, since the document borders do not restrict the ability to position layout elements beyond the borderlines. In addition, document borderlines can be re-sized at any point which makes them more as guidelines for designers that aid in visualizing what the users can see at a time. That is why it is a good practice to use screen display's resolution for the document's

dimensions. In this implementation, the document was set to width of 1920 pixels and height of 1080 pixels. (Just Creative 2012; Microsoft Developer Network. 2011.)

Another good practice is to enable the grid guidelines. This helps alignment and positioning of layout elements as the grid allows elements to be snapped to their borders. Moreover, by defining 1 pixel as the document's nudge increment, which defines how much an element is moved by single stroke of an arrow key, can help in aligning elements even more precisely. This implementation was done by setting the grid size to 24 pixels, which was the initially planned as the default margin between layout elements.

Next phase in preparing the design document was setting up the grid system. Grid system is commonly applied layout design technique that helps alignment of layout elements and content sections according to relatively positioned baseline columns. Grid systems can be static when each column is given a precise width in pixels, or they can be fluid, when the widths are assigned as relative percentages based on the full width of the template. On website's, the responsiveness of grid systems is usually handled with media queries and CSS classes, that handle floating, clearing and widths of columns elements. (Bootstrap; Sonspring 2008.)

One of the most common grid systems is a 12-column grid, where the template is divided into 12 equally wide columns. The width of columns is dependent on the overall width of the page and gutter margin that is left between each column, excluding the first and last column. This implementation was done by using a 12 column grid systems where each column was assigned to a width of 69 pixels with 24 pixel gutter margins. (Bootstrap.)



Picture 2: An example of 16 column grid system (Sonspring 2008).

After the design document and the grid system had been prepared, drawing of layout elements was started. This phase was done by first including the main content sections and components that had been specified in the requirements. Next, all the additional page modules were included along with dummy texts that served as temporary placeholders to visualize the use of space. Finally, all layout elements were organized and groomed to represent the finalized mock-up of a webpage.

The final outcome was achieved by improving the layout mock-up through numerous revisions and iterations. During these iterations, different page structures, font-stacks and element positions were tested, until the layout was considered to fulfill its requirements and be generally well balanced, where content and other layout elements were aligned in a good harmony. (Appendix 1.)

6.6 Data migration

"Data migration is the process of transferring data between data storage systems, data formats or computer systems" (TechTarget 2013).

In this context, data migration refers to the process of transferring data from an Excel spreadsheet into the MySQL database of Joomla. In this implementation, the migration was done as a three stage process including data extraction, transformation and uploading activities. (Data Integration Info a; Data Integration Info b)

Diagram 4: Illustration of data migration process (Jindal. 2).

Instead of using Joomla's administrator application for uploading articles manually, migration was done by transforming the contents within the spreadsheet into SQL statements and by injecting them directly into database. This method was chosen in order to avoid tedious task of inputting each article manually.

For a successful data migration, it is essential to understand the target system's database. While analyzing Joomla's database schema and table dependencies, it was discovered that Joomla uses multiple tables for categorizing and displaying content. This meant that in addition to "#__content", also a number of other tables needed to be migrated, in order to guarantee proper functionality of Joomla's components in the final website. (TechTarget 2013)

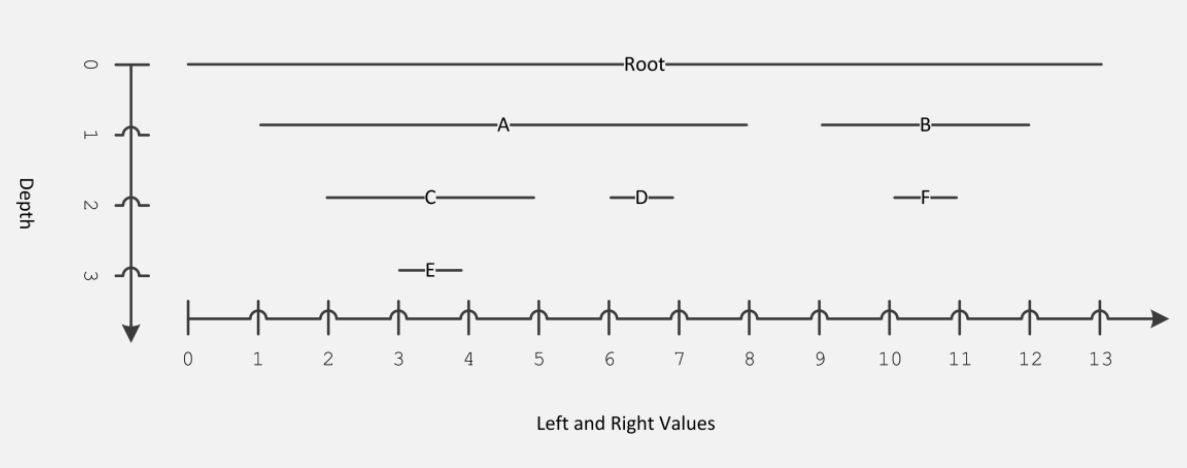
For this implementation, the following tables were identified as crucial for data migration process:

- #__assets – A table for application assets that contain access rules
- #__categories – A table for content categories
- #__content – A table for articles
- #__contentitem_tag_map – A table for mapping content items to tags
- #__content_frontpage – A table for mapping content items that can be featured on homepage
- #__tags – A table for tag words
- #__ucm_base – A base table for the content that has been defined according to unified content model (UCM)
- #__ucm_content – A table for content that has been defined according to UCM
- #__ucm_history – A table for UCM content revisions

6.6.1 Extraction

The extraction work phase was initialized by creating a new Excel spreadsheet document for the migration activities. In this document, all relevant tables were assigned into separate tabs, where each tab was organized according to database table's structure. Since the data in "#__content" had already been prepared during the data cleansing, it could be merged into the document as such. Next, rest of the remaining tables were fulfilled by inserting the missing information or using Excel's cell references whenever there were dependencies between tables. Cell referencing was done in order to avoid referential integrity errors and reduce unnecessary repetition.

The most challenging task was fulfilling the tables that contained nested sets. Nested sets are a technique in relational databases to represent hierarchical structures within tables by using left and right column values. Its main purpose is to ease the logic of fetching child nodes of a parent in a hierarchical model. The biggest challenge with nested sets is that they require each left and right value to be updated whenever new rows are added into the table. In Joomla, nested sets are used in various tables, to represent hierarchies between components and content for instance. (Joomla Docs 2013f; Petersen 2012.)



ID	Left	Right	Depth	Value
0	0	13	0	Root
1	1	8	1	A
2	2	5	2	C
4	3	4	3	E
3	6	7	2	D
5	9	12	1	B
6	10	11	2	F

Diagram 5: Illustration of nested sets with a hierarchical table structure (Petersen 2012).

6.6.2 Transformation

After all relevant data had been extracted and properly organized, the transformation process was started. The goal of this work phase was to transform the data from spreadsheets into valid SQL insert statements, which could be used later on for updating the MySQL database. The transformation was done by exporting each table as separate CSV files and converting them into SQL statements by using an online CSV to SQL conversion application. Finally, the converted statements were exported as SQL script files. (Convert CSV To SQL 2013.)

6.6.3 Load

Final phase in data migration was the process of merging the transformed data into database. This work phase was conducted within MySQL Workbench environment by executing the insert statements as database transactions in separate sessions. Transactions were used in order to provide rollback option, in case the statement contained errors or the outcome was incorrect. The order of statements was also planned beforehand in order to avoid referential integrity violations.

Finally after all SQL statements had been executed, data's validity and correctness was confirmed by running appearance tests via Joomla's public application residing on localhost. During these tests, the functionality of application's main components were tested by running scenarios such as displaying articles that only contain a specific tag word. When the functioning of these components was confirmed, data migration work phase was concluded. (Jindal. 7.)

6.7 Building the website with Joomla

After the articles and other data had been successfully migrated into database, building of the website was started. The goal of this work phase was to construct the final website by following the specified requirements, information architecture and mock-up designs.

This phase included the following main tasks:

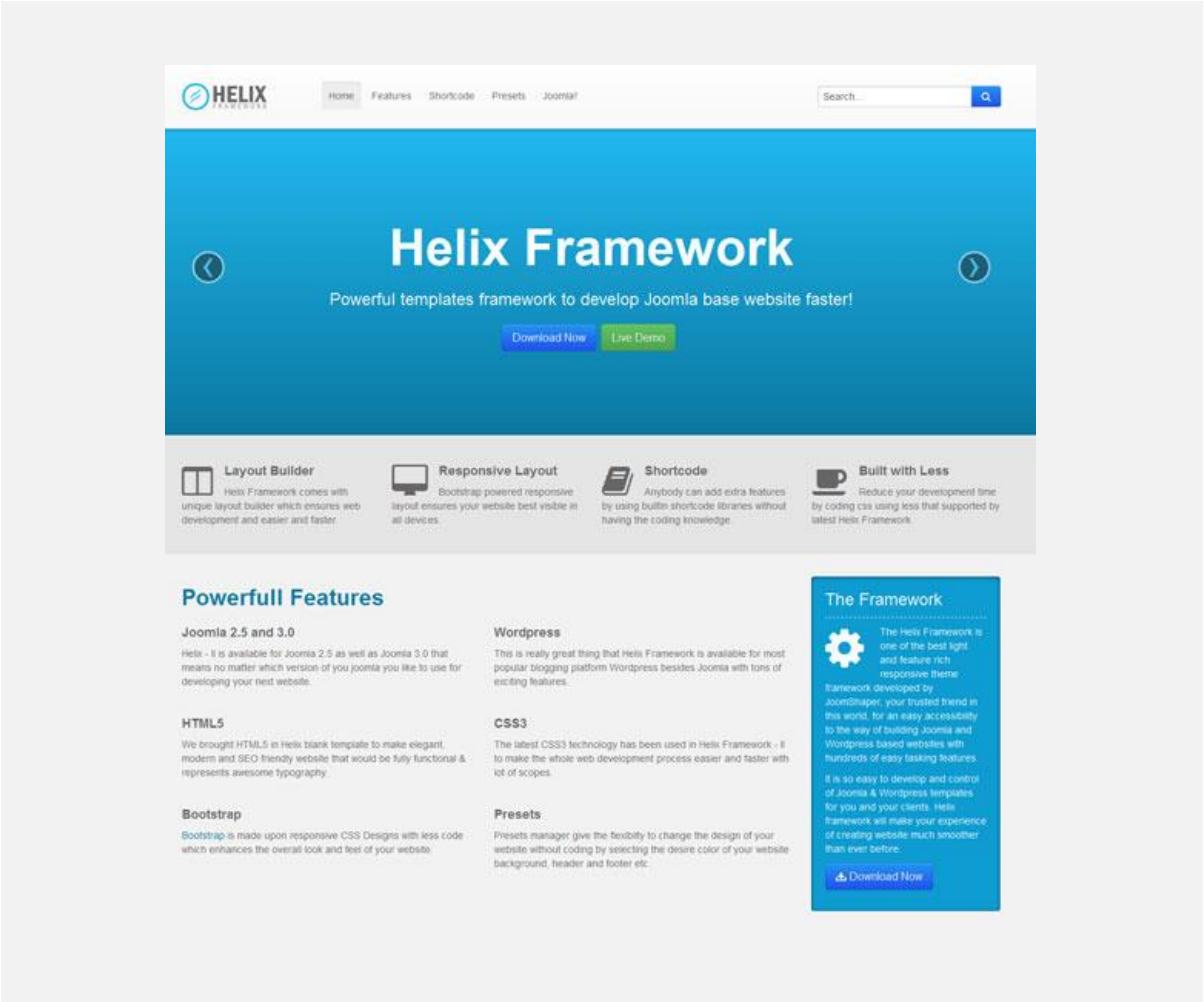
- Setting up a template that was similar to mock-up designs
- Setting up the navigation structure by creating menu items
- Assigning components to each menu item in order to provide page functionality
- Implementing additional modules and page elements
- Customizing the template according to layout mock-up
- Optimizing the layout for mobile devices

6.7.1 Choosing and customizing the template

Before stating to implement any features, a proper template for the website needed to be set up. For this implementation, Shaper Helix – II, a template developed by JoomShaper, was chosen, because it contained similar layout structuring and lots of similar features than the initially designed mock-up. By using a similar base template, less customization was required. Also, the fact that the template came along with a powerful Helix framework

that contained good customization features via administrator application, supported the decision making. (JoomShaper 2013.)

The template was installed via Joomla's back-end application by importing the template's zip file and configured via Template manager.



Picture 3: A screenshot of the Shaper Helix – II, front-end template (JoomShaper 2013).

6.7.2 Implementing navigation system and additional modules

The navigation system was created via administrator application's menu manager by creating new menu items in a hierarchical structure and assigning each item to a specific component. The components were chosen according to target page as what type of content were required to be rendered. Most of the menu items were assigned to a tag component, to fetch and display all articles that contain a specific tag word. (Joomla! Docs 2013g.)

Finally, the additional page modules were created via administrator application's module manager. Also, the modules were associated to specific component, although in some cases, the Custom HTML component was chosen,

either included with some components functionality such as displaying a list of latest article releases or defined as customized HTML modules, when the content, functionality and element structure where defined within the module.

6.7.3 Customizing the template with LESS

The customization of the template was done by editing template's LESS files. The editing was done by extending the template's default CSS class selectors and writing additional selectors for customized modules. Joomla 3 and Helix framework support LESS, the dynamic CSS pre-processor language written in JavaScript, that extend the functionality of traditional CSS by adding support for variables, functions, nested rules and other features. These additions increase the efficiency and re-usability of CSS definitions and allow creation of more logical nested rules that mimic websites' DOM structures. (Joomla 2013; LESS)

In addition to editing the LESS files, the default views of Joomla components written in PHP and HTML were edited in order to have full control over contents' final presentation.

6.8 Testing and Deployment

Testing was done in order to confirm that the website was functioning well and fulfilled the specified requirements. Because the implementation did not include development of components or major coding tasks, testing was targeted mostly for website's usability and appearance.

During the testing, the following conditions were under a review:

- Pages and components function properly
- Back-end application's features function properly
- Layout functions properly on all display sizes
- Chosen font-stacks and font-sizes are good for readability
- All type of media content scales well on all display sizes
- Recommended information security practices has been applied

Testing was done by running scenarios, where the functioning of the above mentioned test conditions were reviewed. The testing was conducted independently and they were run along with other implementation tasks.

After the basic test scenarios had been completed and the website functioning been confirmed, website was demonstrated to the commissioning party. During this demonstration session, the commissioning party was able to test the functionality of the website and suggest improvements to parts that required more polishing. After the demonstration session, the work was approved by the commissioning party and it was ready to be released.

6.8.1 Deploying website to web server

After the website was approved by the commissioning party, it was released by deploying it to a hosting provider's web server. In order to reduce hosting expenses, it was agreed that domain registration would be excluded, so the website was deployed under commissioning party's existing domain name. The deployment process was completed in three phases, which all required use of different application tools.

In the first phase, all project files residing on localhost's directory were compressed as a zip file, that was transferred to the web server with FileZilla, FTP client application. Finally, the contents of the zip file were extracted into a new directory on the web server.

In the second phase, new MySQL database for the website was created within the web server by using hosting provider's CPanel application. Next, the localhost's MySQL database was exported as a database dump file by using localhost's PHP MyAdmin database management application. Finally, the contents within the dump file were exported into web server's newly created database by executing the dump file's insert statements with server's PHP MyAdmin application.

In the third and final phase, website's configurations were set by editing the new connection parameters to a config file residing on the website's root directory on server's side. Finally, the deployment work phase was concluded when the website could be publicly accessed by entering the website's URL address.

7 Summary

Before implementing this thesis, I didn't have any actual knowledge or experience about data cleansing or data migration. Researching these topics as well as getting firsthand experience by applying them in practice, helped me to understand their core concepts and realizing their importance for website development projects and how they fit in the big picture of development process. Especially the importance of clean and high-quality data was highly valued, as it was realized through first-hand experience that data becomes meaningless and loses its value, if it is badly formatted or contains critical errors.

Joomla CMS was a familiar platform from my previous projects, so choosing it for this implementation was a safe choice. Still, I had never conducted data migration for it, which ultimately turned out to be more challenging task than expected. Nevertheless, it was also a good experience as it helped me to understand Joomla's underlying design architecture and data storage structures better.

In overall, this project implementation was quite laborious and challenging, but also useful and rewarding and I was able to learn from each work phase something new. The final website ended up being functional and decent and it satisfied commissioning party at the end.

The project was delayed with couple of months, since data cleansing and migration activities lasted much longer than expected. This could have been avoided by analyzing the data and system's requirements more thoroughly before making the schedule estimations or if more efficient data cleansing and data migration plans and practices would have been made and applied.

7.1 Future development

Future development plans of the website were left undecided by the time of this report and its maintenance and editorial work were left for the commissioning party to decide. Since some important development areas such as search-engine-optimization were left out from this implementation, the website might require further development activities. Also, more extensive and thorough usability and information security testing could be applied in the future.

7.2 Importance of results

In addition to completing this thesis in order to complete my studies, this project was also important for the commissioning party, since the contents of the website concerned universally important topics that are also relevant for their film productions. Being forced to put the previous website offline, had been very disappointing and frustrating, since a lot of work had been put for marketing and producing the website and its content. Thus, having the previously produced material back online was very important for them as well as updating the content and modernizing the website's appearance in the process.

References

Baxter, S & Vogt L.C. 2002. Google Patents. Content Management System. US 6356903 B1. URL: http://www.google.com/patents/US6356903. Accessed: 5 Sep 2014.

Bootsrap. CSS. URL: http://getbootstrap.com/css/. Accessed. 17 Oct 2014.

Chapman, A. 2005. Principles and Methods of Data Cleaning. URL: http://imsgbif.gbif.org/CMS_ORC/?doc_id=1262&download=1. Accessed: 6 Sep 2014.

Convert CSV To SQL. 2013. Data Design Group Inc. URL: http://www.convertcsv.com/csv-to-sql.htm. Accessed: 14 Oct 2014.

Data Integration Info. a. Data Migration. URL: http://www.dataintegration.info/data-migration. Accessed: 13 Oct 2014.

Data Integration Info. b. ETL (Extract-Transform-Load). URL: http://www.dataintegration.info/etl. Accessed: 13 Oct 2014.

Dongre, K. 2004. Information Management. Data Cleansing Strategies. URL: http://www.information-management.com/infodirect/20041029/1012952-1.html. Accessed: 6 Sep 2014.

Freedom for Sale. 2014. URL: http://www.artfilmsproduction.com/freedomforsale/. Accessed: 16 Oct 2014.

Goodrich, R. 2013. Business News Daily. CMS: Content Management Systems & Software. URL: http://www.businessnewsdaily.com/5148-content-management-systems.html

Jindal, S. L&T Infotech. Effective Testing & Quality Assurance in Data Migration Projects. URL: http://www.lntinfotech.com/services/testing/documents/Testing_Quality_Assurance_in_Data_Migration_Projects.pdf. Accessed: 14 Oct 2014.

Joomla. 2013. Joomla 3: Mobile R3ADY / US3R Friendly. URL: http://www.joomla.org/3/en. Accessed: 14 Oct 2014.

Joomla! Docs. 2012. Top 10 Stupidest Administrator Tricks. URL: http://docs.joomla.org/Top_10_Stupidest_Administrator_Tricks. Accessed: 5 Sep 2014.

Joomla! Docs. 2013a. Understand How Joomla Works. URL: http://docs.joomla.org/Portal:Beginners/Understand_How_Joomla_Works. Accessed: 7 Sep 2014.

Joomla! Docs. 2013b. Administration of a Plugin in Joomla. URL: http://docs.joomla.org/Administration_of_a_Plugin_in_Joomla. Accessed: 7 Sep 2014.

Joomla! Docs. 2013c. Getting Started with Templates. URL: http://docs.joomla.org/J3.x:Getting_Started_with_Templates. Accessed: 7 Sep 2014.

Joomla! Docs. 2013d. Extension types (general definitions). URL: http://docs.joomla.org/Extension_types_(general_definitions). Accessed: 5 Sep 2014.

Joomla! Docs. 2013e. Beginners. Installing Joomla. URL: http://docs.joomla.org/Beginners#Installing_Joomla.21. Accessed: 14 Oct 2014

Joomla! Docs. 2013f. Using nested sets. URL: http://docs.joomla.org/Using_nested_sets. Accessed: 14 Oct 2014.

Joomla! Docs. 2013g. How To Use Content Tags in Joomla!. URL: http://docs.joomla.org/J3.x:How_To_Use_Content_Tags_in_Joomla. Accessed: 16 Oct 2014.

Joomla. 2014a. About Joomla. URL: http://www.joomla.org/about-joomla.html. Accessed: 7 Sep 2014.

Joomla. 2014b. About the Joomla Project. URL: http://www.joomla.org/about-joomla/the-project.html. Accessed: 7 Sep 2014.

Joomla. 2014c. Technical Requirements. URL: http://www.joomla.org/technical-requirements.html. Accessed: 5 Sep 2014.

Joomla! Docs. 2014a. Extension. URL: http://docs.joomla.org/Extension. Accessed: 7 Sep 2014.

Joomla! Docs. 2014b. Component. URL: http://docs.joomla.org/Component. Accessed: 7 Sep 2014.

Joomla! Docs. 2014c. Module. URL: http://docs.joomla.org/Module. Accessed: 7 Sep 2014.

Joomla! Docs. 2014d. Plugin. URL: http://docs.joomla.org/Plugin. Accessed: 7 Sep 2014.

Joomla! Docs. 2014e. Template. URL: http://docs.joomla.org/Template. Accessed: 7 Sep 2014.

Joomla! Docs. 2014f. Security: URL: http://docs.joomla.org/Security. Accessed: 5 Sep 2014.

Joomla Security Info. 2014. URL: http://www.joomlasecurity.org/. Accessed: 5 Sep 2014.

JoomShaper. 2013. Templates. Shaper Helix -II Joomla Templates Framework. URL: http://www.joomshaper.com/joomla-templates/helix-ii. Accessed: 17 Oct 2014.

Just Creative. 2012. The Ultimate Web Design Workspace for Photoshop. URL: http://justcreative.com/2012/04/17/web-design-workspace-photoshop/. Accessed: 17 Oct 2014.

LESS. Language Features. URL: http://lesscss.org/features/. Accessed: 14 Oct 2014.

Microsoft. 2014. Download Center. Microsoft Expression Design 4 (Free Version). URL: http://www.microsoft.com/en-us/download/details.aspx?id=36180. Accessed: 16 Oct 2014.

Microsoft Developer Network. 2011. Expression Design 4. Managing your files. Set document page options. URL: http://msdn.microsoft.com/en-us/library/cc295184(v=expression.40).aspx. Accessed: 17 Oct 2014.

Morville, P. 2012. Understanding Information Architecture. URL: http://prezi.com/aafmvya6bk7t/understanding-information-architecture/. Accessed: 6 Sep 2014.

Morville, P. & Rosenfeld L. 2006. Information Architecture for the World Wide Web. 3rd ed. O'Reilly. Sebastopol, CA.

Petersen, E. 2012. Nested Sets. URL: http://www.evanpetersen.com/item/nested-sets.html. Accessed: 14 Oct 2014.

Shreves, R. 2010. Joomla Bible. Whiley Publishing Inc. Indianapolis, CA.

SiteGround. 20014. Joomla Security Tutorial. How to Secure Joomla 3 and Protect it Against Hacker Attacks. URL: http://www.siteground.com/tutorials/joomla/joomla-security.htm. Accessed: 5 Sep 2014.

Sonspring. 2008. 960 Grid System. URL: http://sonspring.com/journal/960-grid-system. Accessed: 17 Oct 2014.

Techopedia. Web Scraping. URL: http://www.techopedia.com/definition/5212/web-scraping. Accessed: 16 Oct 2014.

TechTarget. 2011a. Content Management System (CMS). URL: http://searchsoa.techtarget.com/definition/content-management-system. Accessed: 6 Sep 2014.

TechTarget. 2011b. Content Management Application (CMA). URL: http://searchcontentmanagement.techtarget.com/definition/content-management-application-CMA. Accessed: 7 Sep 2014.

TechTarget. 2013. Data migration. URL: http://searchstorage.techtarget.com/definition/data-migration. Accessed: 13 Oct 2014

Usability.gov. 2014a. Information Architecture Basics. URL: http://www.usability.gov/what-and-why/information-architecture.html. Accessed: 6 Sep 2014.

Usability.gov. 2014b. Information Architecture Methods. URL: http://www.usability.gov/how-to-and-tools/methods/information-architecture/index.html. Accessed: 6 Sep 2014.

Usability.gov 2014c. Organization Schemes. URL: http://www.usability.gov/how-to-and-tools/methods/organization-schemes.html. Accessed: 6 Sep 2014.

Usability.gov 2014d. Organization Structures. URL: http://www.usability.gov/how-to-and-tools/methods/organization-structures.html. Accessed: 6 Sep 2014.

Usability.gov 2014e. Content Inventory. URL: http://www.usability.gov/how-to-and-tools/methods/content-inventory.html. Accessed: 6 Sep 2014.

Usability.gov 2014f. Wireframing. URL: http://www.usability.gov/how-to-and-tools/methods/wireframing.html. Accessed: 6 Sep 2014.

Verens, K. 2010. CMS Design Using PHP and jQuery. Packt Publishing Ltd. Birmingham, UK.

W3Techs. 2014. Usage of content management systems for websites. URL: http://w3techs.com/technologies/overview/content_management/all. Accessed: 7 Sep

2014.

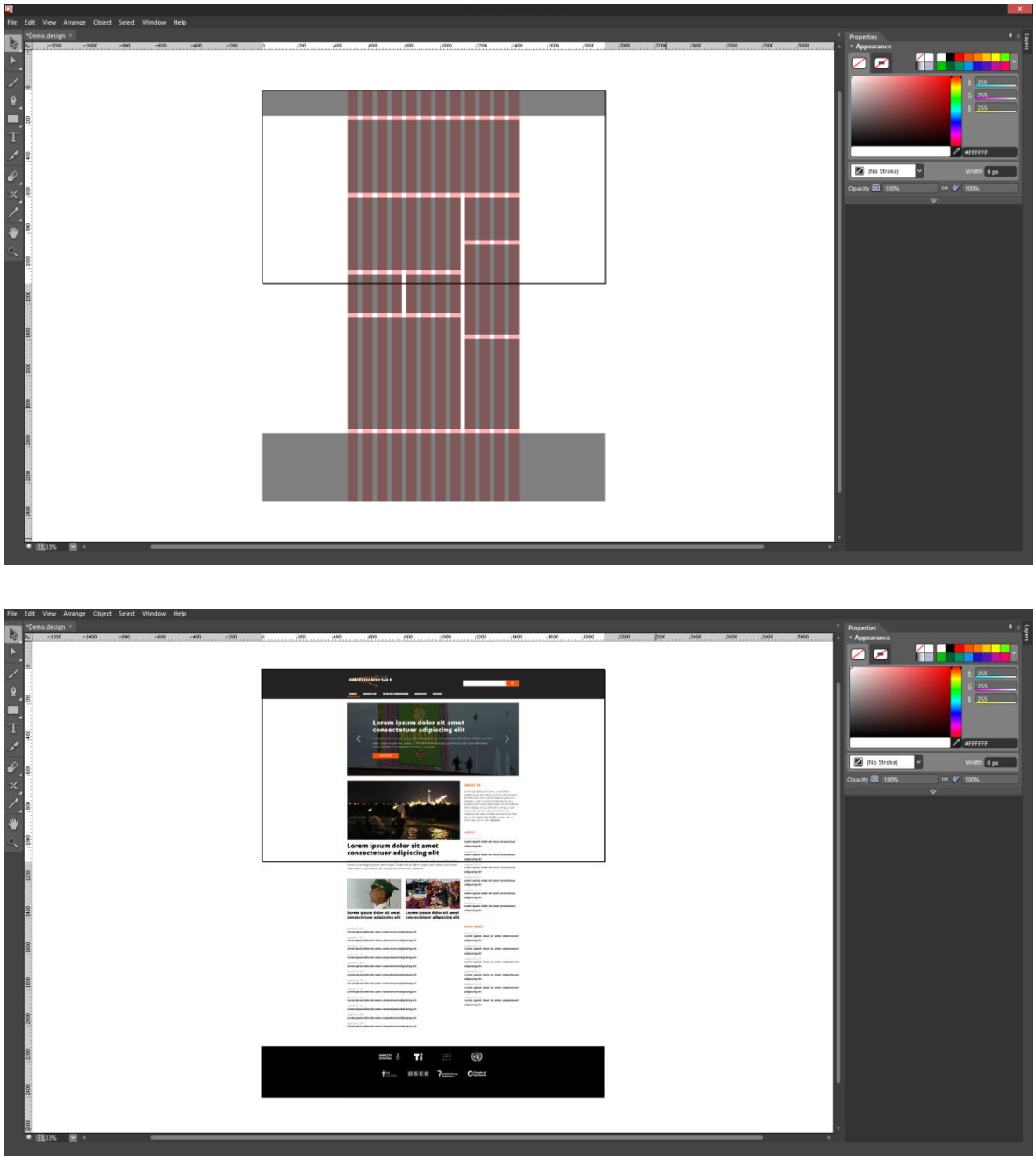
Web Designer Depot. 2010. The Myth of DPI. URL: http://www.webdesignerdepot.com/2010/02/the-myth-of-dpi/. Accessed: 17 Oct 2014.

Webopedia. Pixel. URL: http://www.webopedia.com/TERM/P/pixel.html. Accessed: 17 Oct 2014.

Wikipedia. 2006. Template Engine. URL: http://en.wikipedia.org/wiki/File:TempEngWeb016.svg. Accessed: 6 Sep 2014.

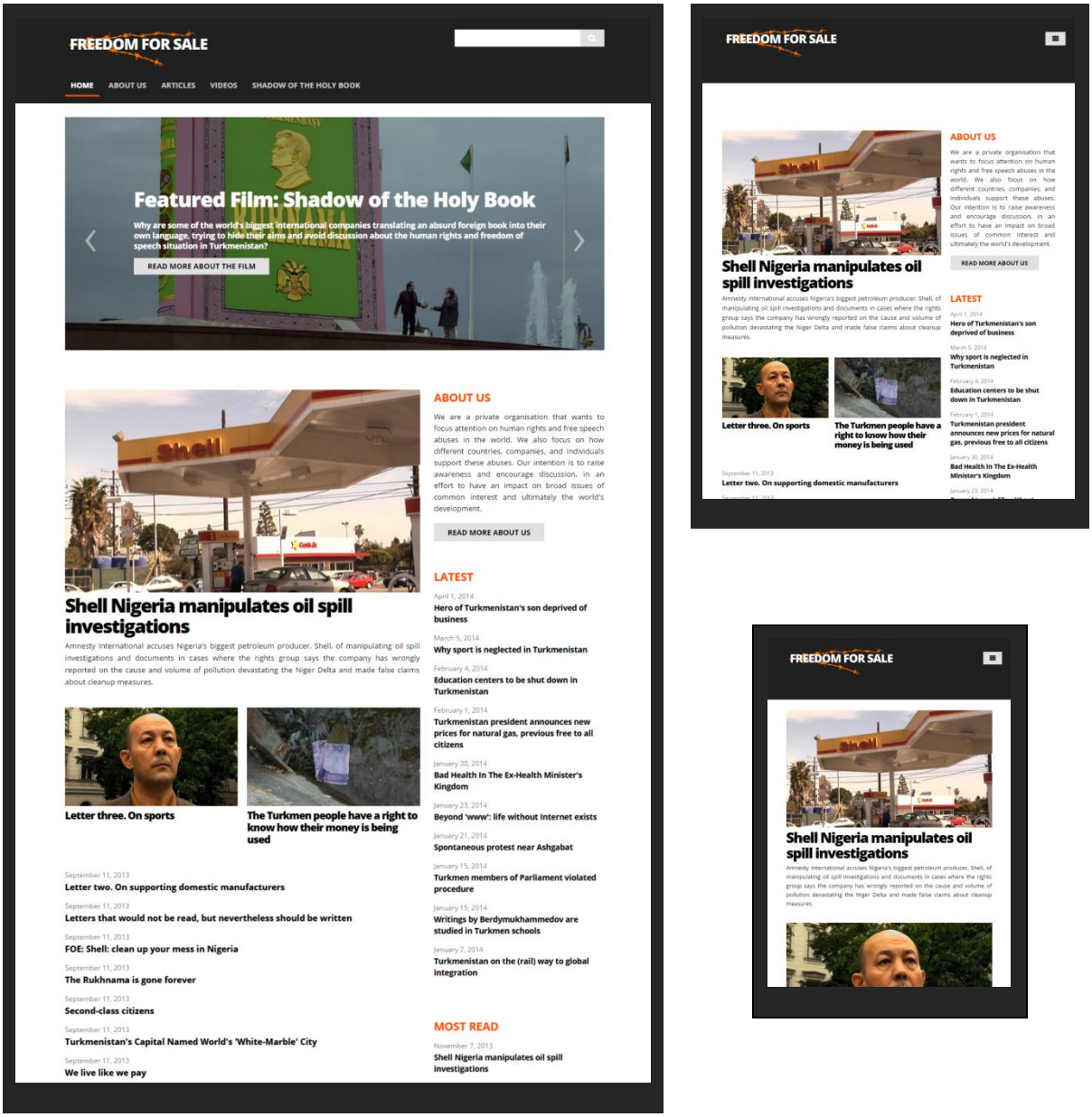
Appendices

Appendix 1: Screenshots of the homepage mock-ups



Picture 4. The mock-up designs made with Microsoft Expression Design 4, showing the initially planned content section and module positions (top) and the final homepage mock-up (bottom).

Appendix 2: Screenshots of the finished website's homepage view



Picture 5. Website's layout for desktop displays (left), tablet displays (top-right) and mobile phone displays (bottom-right) (Freedom for Sale 2014).