DESIGN AND IMPLEMENTING A WEB CONTENT MANAGEMENT SYSTEM

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Xiaohui Zeng
Summer 2012
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Xiaohui Zeng:

Design and Implementing a Web Content Management System

Carl Eckberg, Chair
Department of Computer Science

Alan Riggins
Department of Computer Science

Greg Durbin
School of Theatre, Television and Film

May 14, 2012
Approval Date
Copyright © 2012
by
Xiaohui Zeng
All Rights Reserved
DEDICATION

To my loving wife Rosie and our little one.
ABSTRACT OF THE THESIS

Design and Implementing a Web Content Management System
by
Xiaohui Zeng
Master of Science in Computer Science
San Diego State University, 2012

This thesis is a report on the development of the web content management system for Professor Mark Freeman in the School of Theatre, Television and Film, San Diego State University. The goal of the project is to create a web site for Professor Freeman to showcase his film portfolio and serve as a portal for his academic works. It should also provide an easy to use, flexible framework to manage the site content.

The first part of the work focuses on the requirements, analysis, design and implementation of the system. Since the web site was completed and deployed a few years ago, and there are many advances in web technologies, the latter part of the thesis is a discussion on how the site can be done with the new frameworks, technologies and methodologies that are available today. The study will treat the project as if it’s an enterprise level application with an emphasis on how to make the site easier to maintain, extend and test.
TABLE OF CONTENTS

PAGE

ABSTRACT .......................................................................................................................... v
LIST OF TABLES .................................................................................................................. ix
LIST OF FIGURES .............................................................................................................. x
ACKNOWLEDGEMENTS ..................................................................................................... xii

CHAPTER

1 INTRODUCTION ................................................................................................................1
  1.1 Problem Statement ...................................................................................................... 1
  1.2 Thesis Organization ................................................................................................... 1

2 SYSTEM REQUIREMENTS .................................................................................................3
  2.1 Overview ..................................................................................................................... 3
  2.2 General Requirements ............................................................................................... 3
    2.2.1 Web Browser Requirements ............................................................................... 3
    2.2.2 Operating System Requirements ........................................................................ 3
    2.2.3 Hardware Requirements ..................................................................................... 3
    2.2.4 Video Streaming .................................................................................................. 4
  2.3 User Interface ............................................................................................................... 4
    2.3.1 Look and Feel ...................................................................................................... 4
    2.3.2 Navigation .......................................................................................................... 4
    2.3.3 User Input Validation .......................................................................................... 4
    2.3.4 Help .................................................................................................................... 4
  2.4 Security ....................................................................................................................... 4
  2.5 Use Cases ................................................................................................................... 5
    2.5.1 Use Case Diagram ............................................................................................... 5
    2.5.2 Web Site Use Cases ............................................................................................ 5
      2.5.2.1 Home Page .................................................................................................. 5
      2.5.2.2 View Films ................................................................................................... 7
      2.5.2.3 View Community Productions .................................................................... 7
2.5.2.4 Browse Academic Resources .......................................................... 9
2.5.2.5 Play Film Clips ................................................................................. 10
2.5.2.6 Contact Site Owner ........................................................................ 10
2.5.3 Content Management System Use Cases .............................................. 10
  2.5.3.1 Login to the Content Management System ...................................... 11
  2.5.3.2 Logout from the Content Management System ............................... 11
  2.5.3.3 Manage Users ................................................................................ 13
  2.5.3.4 Edit Front Page ............................................................................. 14
  2.5.3.5 Manage Static Contents ................................................................. 14
  2.5.3.6 Edit Theme ................................................................................... 16
  2.5.3.7 Manage Films ................................................................................ 17
  2.5.3.8 Manage Scripts ............................................................................. 17
  2.5.3.9 Manage Transcripts ...................................................................... 18
  2.5.3.10 Manage Clips ............................................................................. 20
  2.5.3.11 Manage FAQ ............................................................................... 22
3 IMPLEMENTATION DETAILS ...................................................................... 24
  3.1 Hosting Environment ............................................................................ 24
  3.2 PHP ........................................................................................................ 25
    3.2.1 Code Reuse ..................................................................................... 25
    3.2.2 File Operations .............................................................................. 25
  3.3 Javascript ............................................................................................... 26
    3.3.1 Media Preferences .......................................................................... 26
    3.3.2 Input Validation .............................................................................. 26
    3.3.3 Future Consideration ...................................................................... 27
  3.4 CSS ........................................................................................................ 27
  3.5 Security ................................................................................................... 28
    3.5.1 Access Restriction .......................................................................... 28
    3.5.2 Password Encryption ...................................................................... 28
    3.5.3 SQL Injection Prevention .................................................................. 28
  3.6 Database ................................................................................................. 28
    3.6.1 MYSQL ........................................................................................... 28
    3.6.2 Entity Relationship Modeling .......................................................... 29
LIST OF TABLES

Table 3.1. Rohan Hardware Specification as of 02/02/2012 ......................................................24
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Use case diagram</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Homepage</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Film page</td>
<td>8</td>
</tr>
<tr>
<td>2.4</td>
<td>Film index page</td>
<td>8</td>
</tr>
<tr>
<td>2.5</td>
<td>Community production page</td>
<td>9</td>
</tr>
<tr>
<td>2.6</td>
<td>Community production index page</td>
<td>10</td>
</tr>
<tr>
<td>2.7</td>
<td>Resources page</td>
<td>11</td>
</tr>
<tr>
<td>2.8</td>
<td>Contact page</td>
<td>12</td>
</tr>
<tr>
<td>2.9</td>
<td>Login page</td>
<td>12</td>
</tr>
<tr>
<td>2.10</td>
<td>Control panel homepage</td>
<td>13</td>
</tr>
<tr>
<td>2.11</td>
<td>User editor</td>
<td>13</td>
</tr>
<tr>
<td>2.12</td>
<td>User manager</td>
<td>14</td>
</tr>
<tr>
<td>2.13</td>
<td>Front page editor</td>
<td>15</td>
</tr>
<tr>
<td>2.14</td>
<td>Static content editor</td>
<td>15</td>
</tr>
<tr>
<td>2.15</td>
<td>Static content manager</td>
<td>16</td>
</tr>
<tr>
<td>2.16</td>
<td>Theme manager</td>
<td>16</td>
</tr>
<tr>
<td>2.17</td>
<td>Film editor</td>
<td>17</td>
</tr>
<tr>
<td>2.18</td>
<td>Film manager</td>
<td>18</td>
</tr>
<tr>
<td>2.19</td>
<td>Script editor</td>
<td>19</td>
</tr>
<tr>
<td>2.20</td>
<td>Script manager</td>
<td>19</td>
</tr>
<tr>
<td>2.21</td>
<td>Transcript editor</td>
<td>20</td>
</tr>
<tr>
<td>2.22</td>
<td>Transcript manager</td>
<td>21</td>
</tr>
<tr>
<td>2.23</td>
<td>Clip editor</td>
<td>21</td>
</tr>
<tr>
<td>2.24</td>
<td>Clip manager</td>
<td>22</td>
</tr>
<tr>
<td>2.25</td>
<td>FAQ editor</td>
<td>23</td>
</tr>
<tr>
<td>2.26</td>
<td>Help &amp; FAQ</td>
<td>23</td>
</tr>
<tr>
<td>3.1</td>
<td>Entity relationship diagram</td>
<td>30</td>
</tr>
</tbody>
</table>
Figure 3.2. Media player preferences dialog. .............................................................31
Figure 3.3. Site map. ..........................................................................................32
Figure 3.4. Content management system site map. ..........................................33
Figure 4.1. Traditional layered architecture..................................................36
Figure 4.2. ONION architecture. ....................................................................37
ACKNOWLEDGEMENTS

I am grateful to all of those who supported me during my completion of the project.
CHAPTER 1

INTRODUCTION

Mark Freeman is an award winning filmmaker with more than 30 years of experience. He is a professor of Television, Film and New Media in the School of Theatre, Television and Film at San Diego State University. His work is in collections of libraries and universities throughout the country and it has been screened at museums nationally for years. In the spring of 2005, Professor Freeman started looking into ways to bring the stories he documented to the World Wide Web so that his community portraits are available to people around the world.

Coincidentally, my then girlfriend now wife Rosie was a graduate student in the Television, Film and New Media department at the time. She introduced me to Professor Freeman after knowing about the project. After seeing some of the web sites I built in the past and a discussion on my preliminary plans for the project, Professor Freeman decided to work with me. In addition to me, Professor Freeman also worked with Mr. Steve Ladd, a local web site designer. He coordinated with us on designing the layout, look and feel of the web site and I worked on gathering requirements, analyzing and designing the system and ultimately carried out the implementation and deployment.

1.1 PROBLEM STATEMENT

The goal of the project is to create a web site presenting Professor Freeman’s documentary work and serve as a portal to his academic writings. Professor Freeman will provide the text, video clips and images for the system. A backend content management system will be created for site administration. It should be easy to use; anyone with little to no programming experience should be able to administer, and update the site with ease.

1.2 THESIS ORGANIZATION

The thesis is divided into the following sections:

- CHAPTER 1 – INTRODUCTION
• Introduction of the thesis, describing the background and intent of the project

• CHAPTER 2 – SYSTEM REQUIREMENTS
  • Describes the system requirements specification

• CHAPTER 3 – IMPLEMENTATION DETAILS
  • Describes the implementation detail of the project, including hosting environment, programming language, database schema and video streaming implementation

• CHAPTER 4 – CONCLUSION AND FUTURE WORK
  • Takes a critical look at the project and discuss what could have be done differently in hindsight, considering recent advances in design principles and methodologies used in web development
CHAPTER 2

SYSTEM REQUIREMENTS

Requirement specification is a critical aspect of the system analysis process. It describes the desired behaviors of the software system and serves as the contract for the stakeholders.

2.1 OVERVIEW

Upon completion of the project, the web site shall present Professor Freeman’s work to users with an attractive and intuitive user interface. The content management system should be easy to use and secure.

2.2 GENERAL REQUIREMENTS

Minimum required software version, operating system, hardware of the software system are described as follows.

2.2.1 Web Browser Requirements

Following web browsers should be supported:

- Internet Explorer 6.0 or later
- Firefox 3.6 or later
- Safari 3 or later

2.2.2 Operating System Requirements

Following operating systems shall be supported:

- Windows 98 and above
- Mac OS X

2.2.3 Hardware Requirements

Minimum requirements on client machines are:

- Pentium II processor or equivalent
- 1 GHz of processor speed
- 256 MB of RAM

2.2.4 Video Streaming

Video clips should support Real Player, QuickTime Player and Windows Media Player. Users with broadband connection or dialup connections should be able to view video clips on the web site.

2.3 User Interface

User interface of a software system is where interaction between humans and machines occurs. The user interface of this web site shall be easy to use. Any user, even with little to no technical background should be able to use the entire software with ease.

2.3.1 Look and Feel

The look and feel of the web site should be consistent throughout.

2.3.2 Navigation

Navigating in the system should be easy and consistent. A navigation panel should be present in the system at all times.

2.3.3 User Input Validation

Information entered into the content management system shall be validated before committing changes to the database. The system should return clear error messages indicating what failed validation.

2.3.4 Help

There should be a FAQ section in the content management system to assist less technical savvy users on how to use the system.

2.4 Security

The backend content management system should be accessible to authorized users only.
2.5 **Use Cases**

In software system modeling, the use case view captures the behavior of a system, subsystem, class or component as it appears to an outside user. It partitions the system functionality into transactions meaningful to actors – idealized users of a system. A use case describes an interaction with actors as a sequence of messages between the system and one or more actors [1].

This project has two subsystems – the web site and the backend content management system. Actors of the system can be categorized into three types: anonymous users, registered users and administrators. In terms of access to functionalities of the system, anonymous users have the least access as they can only interact with the public facing web site. In additional to the rights of anonymous users, registered users have access to most of the backend content management system. Administrators have access to all aspects of the system.

The following sections will describe the use cases of the project. They are illustrated as a use case diagram (See Appendix) in Section 3.1.1.

2.5.1 **Use Case Diagram**

Use case diagrams are one of the diagrams in the UML for modeling the dynamic aspects of systems. They are central to modeling the behavior of a system, a subsystem, or a class. Each one shows a set of use cases and actors and their relationships [2]. Figure 2.1 is the use case diagram for the system described in this paper.

2.5.2 **Web Site Use Cases**

The use cases of the public facing web site are described as below.

2.5.2.1 **Home Page**

Anonymous users can access the site via a homepage where links to all sections of the web site should be available on the page. Figure 2.2 is a screenshot of the homepage.
Figure 2.1. Use case diagram.
2.5.2.2 View Films

Anonymous users can view individual films. In addition to full details of the film, anonymous users can also obtain information on where to buy the film, and a film guide of links and archives. Anonymous users can also access the film script and download its PDF version from the page for the film. Figure 2.3 is a screenshot of a page for individual films.

There will be an index page where all films can be accessed at the same time. The films index page will be accessible on all pages of the web site. Figure 2.4 is a screenshot of the film index page.

2.5.2.3 View Community Productions

A community production is a film by professor Freeman in conjunction with student collaborators. Anonymous users can view individual community productions. In addition to full details of the community production, anonymous users can also obtain information on
Figure 2.3. Film page.

Figure 2.4. Film index page.
where to buy the film, and a film guide of links and archives. Figure 2.5 is a screenshot of a page of an individual community production.

![Community Production Page](image)

**Figure 2.5. Community production page.**

There will be an index page where an introduction to community productions will be provided and all community productions can be accessed at the same time. The community productions index page will be accessible on all pages of the web site. Figure 2.6 is a screenshot of the community production index page.

### 2.5.2.4 Browse Academic Resources

Anonymous users can browse syllabi and writings by Professor Freeman. Each resource will be available on web pages and a PDF format will be available as a download link.

There will be an index page where all writings and syllabi can be accessed at the same time. The index page for academic resources will be accessible on all pages of the web site. Figure 2.7 is a screenshot of the index page of academic resources.
2.5.2.5 **Play Film Clips**

On pages for films with video clips, users should be able to stream video clips.

2.5.2.6 **Contact Site Owner**

Anonymous users should be able to leave their name, email address and a message to the site owner. An email should be generated and sent to the site owner as soon as the user submits the contact form.

The contact page will be accessible on all pages of the web site. Figure 2.8 is a screenshot of the contact page.

2.5.3 **Content Management System Use Cases**

The use cases for the content management system are shown below.
2.5.3.1 Login to the Content Management System

Users of the content management system are classified into administrators and regular users. After entering their user name and password in the log in screen, users will be redirected to the administrative control panel and perform permitted functionalities. Figure 2.9 is a screenshot of the login screen of the system.

2.5.3.2 Logout from the Content Management System

After logging into the system, the user should be able to logout from the content management system from any part of the control panel. Upon logging out, the control panel should no longer be accessible without requiring the user to login again. There is also a
Figure 2.8. Contact page.

Figure 2.9. Login page.
session timeout that will log the user out automatically when the timeout expires. Figure 2.10 is a screenshot of the control panel homepage. The logout link is on top of the page.

![Control Panel](image)

Figure 2.10. Control panel homepage.

### 2.5.3.3 Manage Users

Administrators of the system can create, modify and delete users from the system. They can also promote a user to an administrator or downgrade another administrator to a regular user. The system will allow users to change their passwords and contact information. There is no sign up page in the system. An administrator will create all users in the user manager. Figure 2.11 is a screenshot of the user editor and Figure 2.12 is a screenshot of the user manager.

![User Manager](image)

Figure 2.11. User editor.
Registered users can change the images of the slide shows on the homepage and they should also be able to change the slogan image. Figure 2.13 is a screenshot of the front page editor.

2.5.3.5 MANAGE STATIC CONTENTS

Registered users can create static contents in the control panel. When creating the static content, users can choose from 5 different content types: CV; Filmography; Other; Syllabi and Writings. Users should be able to specify the title, type, body, keywords and description of the static content. Static contents should also have corresponding image and PDF optionally. When submitting the static content, the system will store newly created static content in the system and make it available on the web site immediately. Figure 2.14 is a screenshot of the static content editor.

All static content can be edited in the control panel after their creation. All fields of the static contents should also be editable and any changes made should be reflected on the web site as soon as changes are saved in the system.
Figure 2.13. Front page editor.

Figure 2.14. Static content editor.
Users should be able to delete static contents that are no longer needed. Figure 2.15 is a screenshot of the static content manager.

![Static Content Manager](image)

**Figure 2.15.** Static content manager.

### 2.5.3.6 Edit Theme

Users should be able to specify the color theme of the entire system. Figure 2.16 is a screenshot of the theme manager.

![Theme Manager](image)

**Figure 2.16.** Theme manager.
2.5.3.7 Manage Films

From the control panel, registered users can create films. When creating the film, users can specify its name, type, runtime, release date, short description, full description, credits, cast, reviews, awards and screenings, ordering information, guide, archives, links and keywords. Additionally, users should be able to upload posters and screenshots for the film. Figure 2.17 is a screenshot of the film editor.

![Film Manager](image)

Figure 2.17. Film editor.

After films have been added to the system, they can be edited in the control panel. All aspects of the film can be editable and any changes made should be saved to the system after submitting the edit form.

Users should be able to delete obsolete films from the system. Figure 2.18 is a screenshot of the film manager.

2.5.3.8 Manage Scripts

A script is also called a screenplay. It is a story told with pictures, in dialogue and description, and placed within the context of dramatic structure [3]. New scripts can be added in the script managing area of the control panel. Users can specify the name and text of the
script when creating the script. Optionally, a PDF can be uploaded for the script. Scripts should belong to one and only one film in the system and they should be accessible in the film they belong to on the web site. Figure 2.19 is a screenshot of the script editor.

Existing scripts should be edited in the control panel. All aspects of the scripts can be editable and any changes made should be saved to the system after submitting the edit form.

Users should be able to delete scripts no longer needed from the system. Deleting films should also delete the scripts for the film from the system. Figure 2.20 is a screenshot of the script manager.

2.5.3.9 **Manage Transcripts**

Transcripts are written transcription of all dialogs in films. Users can add new transcripts for existing films in the transcript managing area of the control panel. Users can specify the name and text of the transcript when creating the transcript. Optionally, a PDF can be uploaded for the transcript. All transcripts should be accessible in the film they belong to on the web site. Figure 2.21 is a screenshot of the transcript editor.

Existing transcripts should be editable in the control panel. All aspects of the transcripts can be editable and any changes made should be saved to the system after submitting the edit form.
Figure 2.19. Script editor.

Figure 2.20. Script manager.
Users should be able to delete transcripts no longer needed from the system. Deleting films should also delete their transcripts from the system. Figure 2.22 is a screenshot of the transcript manager.

2.5.3.10 MANAGE CLIPS

Clips are short trailers of the film or community production. From the control panel, registered users can create clips. When creating clips, users can specify the associated film, bitrate and file type. Additionally, users should be able to mark clips active and inactive. Active clips will be available to anonymous users and inactive clips will not. Figure 2.23 is a screenshot of the clip editor.

After clips have been added to the system, they can be edited in the control panel. All aspects of the clip can be editable and any changes made should be saved to the system after submitting the edit form.

Users should be able to delete obsolete clips from the system. Figure 2.24 is a screenshot of the clip manager.
Figure 2.22. Transcript manager.

Figure 2.23. Clip editor.
An area in the control panel should be dedicated to the Help & FAQ to provide user guide and help information for the content management system. Users should be able to create new FAQ by specifying the question and the answer. Figure 2.25 is a screenshot of the FAQ editor.

After FAQs have been added to the system, they can be edited in the control panel. All aspects of the FAQ can be editable and any changes made should be saved to the system after submitting the edit form.

Users should be able to delete obsolete FAQs from the system. Figure 2.26 is a screenshot of the FAQ manager.
Figure 2.25. FAQ editor.

Figure 2.26. Help & FAQ.
CHAPTER 3

IMPLEMENTATION DETAILS

3.1 HOSTING ENVIRONMENT

The hosting environment dictates many aspects of the project, especially the choice of programming language, underlying database technology and video streaming mechanism. So it was one of the first issues addressed. Several options were considered, but Rohan – the academic hosting environment provided by the Information Technologies Department in San Diego State University became the obvious choice quickly. Detailed description is listed in Table 3.1 [4]. Not only can we host a database driven site on Rohan, it also provides a video streaming service, which is an important requirement for the project that other hosting options had a hard time meeting at the time.

Table 3.1. Rohan Hardware Specification as of 02/02/2012

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Type</td>
<td>SunFire 4900</td>
</tr>
<tr>
<td>Operating System</td>
<td>Solaris 10</td>
</tr>
<tr>
<td>Processor Type</td>
<td>1.5 GHz Ultra SPARC-IV+, 32 MB Cache</td>
</tr>
<tr>
<td>Number Of Processors</td>
<td>8 Dual Core</td>
</tr>
<tr>
<td>Installed Memory</td>
<td>32 GB RAM</td>
</tr>
<tr>
<td>Mass Storage</td>
<td>5 x 36 GB Drives</td>
</tr>
<tr>
<td></td>
<td>A3510 Raid Array (1.75 TB)</td>
</tr>
<tr>
<td></td>
<td>FusionNAS Array (5.25 TB)</td>
</tr>
<tr>
<td>Backups</td>
<td>Legato Networker (daily)</td>
</tr>
<tr>
<td></td>
<td>LTO-3 for off site backups (weekly)</td>
</tr>
<tr>
<td>Network Connection</td>
<td>1 GB Ethernet duplex to campus</td>
</tr>
<tr>
<td>Number Of User Accounts</td>
<td>36,000 (Subject to seasonal fluctuation, with upward growth trend)</td>
</tr>
</tbody>
</table>

3.2 PHP

PHP (recursive acronym for PHP: Hypertext Preprocessor; See Appendix) is a very popular open source general-purpose scripting language that’s capable of generating HTML on web servers. One who speaks C, Java and Perl will find its syntax very familiar as the designer of the language aimed at borrowing syntax from those popular languages to ease the learning curve for ones with existing programming skills. One of the characteristics of PHP that made it incredibly popular on the web is that it is extremely easy for an inexperienced developer, but offers lots of advanced features for an advanced programmer. Aside from typical features such as generating HTML dynamically, processing form data, sending and receiving cookies, managing sessions, etc., there are quite a few other notable PHP features that have been utilized in the project.

3.2.1 Code Reuse

Similar to any other programming language, PHP supports reusing code snippets, whether it’s blocks of HTML or PHP. With PHP, there are a few different functions that help in reusing code; include() and require() are the most frequently used in the project. include() provides a warning and tries to continue execution of the program if possible if a resource cannot be retrieved. require() on the other hand stops processing the page when retrieval of a resource fails. Typically include() is used on static markups and require() is geared towards dynamic contents. The usage of partial pages in this project can be categorized as such. Static markups include common sections on every page. Examples are banner, menus, footer, copyright, etc. Pages can continue to load if any of the static content fails to load. Dynamic contents include functions to open/close database connection string, common string helper functions, common file operation functions etc. It is important to note that I did not write all of the modules. Some of them are open source snippets found on the Internet. The string tokenizer class and the HTML form classes are 2 such modules.

3.2.2 File Operations

One of the core requirements for the project is the ability to upload film clips, images, scripts and academic writings. Thankfully file input and output is rather easy with PHP. It
uses file pointers to reference files and one can operate on files by calling functions on the file pointers. Usually allowing users to upload files to your server can open severe security vulnerabilities if not handled with care, this is not an issue with the project because the functionality is not exposed in the public facing web site. Only registered users of the content management system can upload files to the system and we can place a reasonable trust on their good intentions.

Rohan provides limited storage quota on the root of the virtual directory, this is a problem to the project because of the rather big size of the scripts and papers. The issue was addressed by leveraging some of the remaining space in the media-streaming quota, which is a lot more generous.

### 3.3 JavaScript

JavaScript (See Appendix) is an interpreted programming language with object-oriented (OO) capabilities. Although JavaScript is a general-purpose language that can be used in different context, it is most commonly used in web browsers to interact with the user, control the web browser, and alter the document content that appears within the web browser window [5].

#### 3.3.1 Media Preferences

One of the most notable JavaScript usages in the project is the detection of media players in the browser. The algorithm looks at the plugins array of the navigator object for media player plugins on the client machine and displays the media player preference dialog accordingly. Internet Explorer has to be treated differently because it does not expose an API for plugins in JavaScript. The algorithm was rewritten in VBScript to handle media player detection.

#### 3.3.2 Input Validation

Client-side JavaScript validation has been implemented on the contact page. User has to provide input for all required fields on the form, a pop-up window will warn users if they submit a form without meeting the requirement.
3.3.3 Future Consideration

In 2005, programming in JavaScript was quite difficult. One has to struggle with making code compatible with multiple browsers on different operating environments. As mentioned earlier, different algorithm has to be written to detect media players in Internet Explorer versus other browsers.

In recent years, many JavaScript libraries have become popular by putting an abstraction on top of the core JavaScript language. jQuery for instance is one of the most prominent JavaScript libraries of the day. Not only does it shield the pain of cross-browser compatibility from developers, it also has an extensive plugin library that provides toolsets to make web sites more interactive and attractive at the same time. An obvious plugin choice for this project would be jquery.validate. It provides a very simple yet extensible framework to do client-side input validation.

3.4 CSS

Cascading Style Sheets (CSS) (See Appendix) is a simple mechanism for adding style (e.g., fonts, colors, and spacing) to Web documents [6]. The language can be applied to any XML document, but the most common application is to style web pages written in HTML and XHTML. CSS is designed to separate the content of the document from its presentation. By creating the separation, one can create consistent look and feel across multiple pages. It also makes it easy to change presentation without changing content. The styling of the frontend web site and the backend content management system are completely different in the project because they use different style sheets.

Similar to JavaScript, cross-browser compatibility is difficult, especially a few years ago when web standard adoption varies in major browsers. Internet Explorer once again is the worst offender in not complying with standards. One has to write special code to workaround corner cases to make styling look right on Internet Explorer, which is exactly the case with this project. Luckily like JavaScript, new CSS frameworks have been created recently to address this issue. Blueprint is one such framework (For more info, see http://www.blueprintcss.org).
3.5 Security

Several measures have been taken to ensure the software system is secure from outside attacks. They are describes below.

3.5.1 Access Restriction

The backend content management system is accessible to authorized users only. Authorized users gain access via a login page where they’re asked for username and password. A session variable will be set upon a successful login, and it will be destroyed upon logout. Before processing any further requests, all pages in the content management system check for the existence of the session variable. Users will be redirected to the login page if it’s not found.

3.5.2 Password Encryption

Instead of storing passwords in the database as clear text, they are stored as MD5 hash. PHP’s built-in md5() function is used here.

3.5.3 SQL Injection Prevention

SQL injection (See Appendix) is a technique that exploits systems that do not filter or pre-process user input properly and thereby executes maliciously embedded SQL statements unexpectedly. Care has been put in place to prevent such attacks in the project. The technique employed here is parameterized statements using the PHP MySQL ODBC driver. Instead of simply concatenating strings to compose SQL statements, each input argument is passed in as an SQL parameter, thus fixating the SQL statement by bounding user input with parameters.

3.6 Database

The content management system is a data driven web site. A robust database has been designed to store/retrieve information about the system. The technology choices and design is described below.

3.6.1 MySQL

MySQL (See Appendix) is the most used relational database management system (See Appendix) in the world. One of the reasons that made the system popular is its ease of
use that helps small development teams to create a data-driven site in relatively short time. The fact that there is no licensing fee and it is already installed on Rohan and ready to be used also influenced the decision to use it.

It is worth noting that MySQL also scales well. Many high-profile web sites including Google, Facebook and Twitter are powered by it.

3.6.2 Entity Relationship Modeling

Entity relationship modeling is a high-level conceptual data model developed to facilitate database design. A conceptual data model is a set of concepts that describe the structure of a database and the associated retrieval and update transactions on the database [7]. There are four main concepts in ER modeling:

- Entities – Objects or concepts the data represents
- Relationships – Associations among one or more entities
- Attributes – Properties of entities or relationships
- Constraints – Restrictions that limit the possible combinations of entities that may participate in the corresponding relationship set [7].

3.6.3 Entity Relationship Diagram

An entity relationship diagram (See Appendix) is usually used to describe an ER model. The entity relationship diagram for this project is illustrated in Figure 3.1.

The ER Diagram was generated using MySQL Workbench.

3.6.2 Table Definitions

config – Configuration information of the system.
contacts – Contact information of the site.
faq – A collection of frequently asked questions for the site administrators.
film_clips – Meta data for film video clips. film_id is a foreign key references to the film_id in the films table.
films – Essential information of the films. Such information includes title of the film, long and short description, credits, cast, reviews, links and etc. A non-clustered index was created on the film_name field to speed up frequent lookup on the field.
images – Meta data for all images used in the site.
scripts – Meta data for film scripts. *film_id* is a foreign key reference to the *film_id* in the *films* table.

slideshow – Meta data for film slideshows. *film_id* is a foreign key that references the *film_id* in the *films* table and *image_id* is a foreign key that references the *image_id* in the *images* table.

users - Stores the administrative information. As a security measure, passwords are hashed using the MD5 algorithm before saving to the database.
3.7 Video Streaming

Three of the most popular media formats at the time are supported by the system: Real Media, QuickTime and Windows Media. Two bit rates are provided to users: 384k for users with broadband connection and 56k bit rate for users with dialup connections. When a user launches the video player for the first time, he/she will be presented with a video preference dialog to specify what format and bit rate to use. Cookies will be created so that this is no longer necessary during subsequent visits. The preference dialog (Figure 3.2) also detects video players installed on the system and present the user with links to install missing video players if there are any.

This project was completed in 2005. Online video sharing was not as prominent as it is today. YouTube would be the obvious choice if it where done today. Multi-platform support, bit-rate management will all be handled automatically. More importantly, hosting videos on YouTube also provides an avenue to promote the web site, thus gaining more traffic. Figure 2.4 is the site map of the website. Figure 2.5 is the site map of the content management system.

Figure 3.2. Media player preferences dialog.
### 3.8 Site Map

Site map is a list of pages of a web site organized hierarchically. It documents the organization of the pages in the web site. Figure 3.3 is the sitemap of the public facing site and Figure 3.4 is the sitemap of the backend content management system.

![Site map](image)

**Figure 3.3. Site map.**
Figure 3.4. Content management system site map.
CHAPTER 4

CONCLUSION AND FUTURE WORK

Of all the deficiencies of the system described in this paper, the weakest link is testing. There are no unit tests, and functional tests are all done manually during development. It is not as big of a problem because of the nature of the system. It is mostly a one-off project that does not evolve a lot after the initial deployment. Such an issue becomes much more glaring if it were an enterprise level system that goes through changes continuously throughout the system’s lifecycle.

Several years after the site being deployed, web development has become much more prominent in software engineering. There have been many advances in recent years, both in terms of design philosophies and technology choices. The final chapter of this paper will discuss some of these developments, with an emphasis on making projects easier to maintain, extend and test in an enterprise environment.

4.1 DESIGN PRINCIPLES AND METHODOLOGIES

Design principles describe how elements of a design come together. There are many areas of the system that can benefit from using some of the new principles and methodologies introduced in recent years. They are described below.

4.1.1 Dependency Inversion Principle

In a conventional architecture, complex systems are decomposed into components and lower-level components are used by higher-level components to perform tasks. The dependency between low-level components and high-level components is direct and it makes the system rigid and fragile. High-level components are hard to reuse because they are hard-wired to work with the low-level components that may not work when circumstances change. This is one of the biggest issues of the system in discussion here. The user interface, business logic and data access logic are all tightly coupled with each other. Changing any of the layers will likely require a complete rewrite because of that.
Dependency inversion principle (See Appendix) is aimed at addressing this coupling issue. Robert Martin introduced the concept in 1995. He states the following:

1. High-level modules should not depend upon low-level modules. Both should depend upon abstractions.
2. Abstractions should not depend upon details. Details should depend upon abstractions [8].

In software engineering, abstraction is the ability to engage with a concept while safely ignoring some of its details [9]. The mere fact that the word “abstraction” appears three times in the definition is an indication of how much importance Martin puts on it. Abstraction leads to loose coupling because it ignores concrete implementation, which is where rigidity comes from. Dependency inversion principle requires all parts of the system to communicate in terms of contracts. As long as one satisfies the contractual requirements of the abstraction, you can change parts of the system easily, which is at the heart of the dependency inversion principle.

In a large system with multiple levels of components, if you adhere to the dependency inversion principle religiously, all parts of the system can be swapped in and out easily without affecting other parts, which makes the system more agile, stable and mobile.

4.1.2 ONION Architecture

ONION Architecture (See Appendix) is an architectural pattern introduced by Jeffery Palermo in 2008 in three blog posts. The issue Jeffery sets out to address is to create an architecture that helps creating more maintainable applications with an emphasis on separation of concerns throughout the system [10].

4.1.2.1 Traditional Layered Architecture

Traditionally, the most frequently used architecture is the layered architecture illustrated in Figure 4.1. This approach improves maintainability by creating separations of components into groups based on roles and functionality. The dependency chain in this architecture points downwards with each layer depending on the layer beneath it. The biggest issue here is it creates unnecessary coupling with external infrastructure and data access.
Historically, external infrastructure and data access technologies tend to change frequently. The direct coupling in layered architecture makes it difficult to keep up with the latest and greatest without major changes to the entire code base. Applications become stale over time because tight coupling prevents easy upgrade of parts of the system. Most often than not, they get rewritten eventually. Figure 4.1 is how traditional architecture layered.

4.1.2.2 ONION ARCHITECTURE

The main goal of the ONION Architecture is to reduce coupling between layers. All coupling is toward the center with this approach. Code can depend on layers closer to the center but not the other way around. Fundamentally, it makes domain objects the center of the entire system. All layers talk in terms of interfaces, which reside in the Domain Model. External dependencies are pushed to the outer layers of the system and the likelihood of change and complexity are proportioned to the distance to the center. Data access layer for instance, gets pushed to the edge because of its tendency to refresh every few years.
The ONION Architecture, see Figure 4.2, makes use of dependency injection extensively. Objects can use objects or classes in the same layer directly, but if it needs something from another layer, it has to rely on something external to pass the dependency in. An IoC container becomes incredibly handy here as it resolves the tree of dependencies behind the scenes and makes the process seamless.

Figure 4.2. ONION architecture.

It is important to note that ONION Architecture is tailored for large enterprise applications with long-lived shelf life and complex behavior. The additional agility and maintainability do not come without a cost. It takes a relatively skilled architect quite significant effort to design the system and lay down the architecture properly. This is not what small websites/applications can afford most of the time. They have to live with the reality of short deadlines and limited resources.
4.1.3 Inversion of Control Containers and Dependency Injection

As the size of a software system grows, so does the complexity of wiring up various components in the system. Inversions of Control Containers (See Appendix) are the libraries that help creating objects for a class. Dependency injection is a common approach developers take to mitigate the proliferation of dependencies. With dependency injection (See Appendix), instead of relying on classes to construct objects themselves, they are injected externally.

Dependency injection does not come free. Especially with large systems, developers have to write quite a lot of boilerplate code for object construction and dependency resolution. To reduce these kinds of repetitive grunt work, one can use IoC (Inversion of Control) containers.

IoC containers provide a layer of abstraction with generic factory classes that instantiate classes, resolve dependencies, and manage object lifecycles for you. Most popular languages usually have multiple offerings of IoC containers. For example, Java has Spring and Guice, C# has StructureMap and Ninject, just to name a few.

There are three different techniques for dependency injection – constructor injection; setter injection and interface injection. Constructor injection requires callers to pass objects as arguments of the class constructor. Setter injection uses setter methods of properties to pass dependencies to the class. Interface injection on the other hand defines and uses interfaces for the injection. Any class that wants to use a given dependency has to implement the interface for injecting that interface into the object. It requires you to write a lot of interfaces and implementations to manage dependency resolution. Which is why most IoC containers favor constructor and setter injection.

4.2 CONCLUSION

This paper sets out to describing the background information of the project for Professor Mark Freeman’s web site and the backend content management system. It then documented the requirement specifications of the system and also implementation details of the final product. We also looked back at the project and wondered what new design principles can be used with the advances of today.
In conclusion, we have described the development of the content management system that is easy to use for one that’s not even technically savvy. The finished web site has been serving as a portal to Professor Freeman’s film productions and academic works. It has enjoyed thousands of visitors ever since its initial deployment.
REFERENCES


**Content Management System**: Software that allows publishing, editing contents and site maintenance

**CSS (Cascading Style Sheets)**: a simple mechanism for adding style (e.g., fonts, colors, and spacing) to Web documents [6]

**Dependency Injection**: To inject objects into a class, rather than relying on the class to create the object itself [11]

**Dependency Inversion Principle**: The reversal of dependencies between high-level components and low-level components for the purpose of decoupling

**Entity Relationship Diagram**: a graphic that illustrates the relationships between entities in the database

**Inversion of Control Containers**: Libraries that helps creating objects for a class

**JavaScript**: A general purpose scripting language that is commonly used in Web Browsers

**ONION Architecture**: An architectural design where code dependencies points towards layers more central only

**PHP**: Hypertext Preprocessor, an open source general-purpose scripting language that’s capable of generating HTML on web servers

**MySQL**: an open source relational database management system

**Relational Database Management System**: A database system that’s based on the relational model

**SQL Injection**: a technique used to attack databases by injecting SQL statements via form posts

**Use Cases**: caption of the behavior of a system, subsystem, class or component as it appears to an outside user

**Unified Modeling Language**: A graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system [2]

**Use Case Diagram**: a graphic that illustrates the actors, subsystems, components and use cases of the system