

Developing a Web Based Content Management System Using PHP

By

Adam M. Erickson

Bachelors of Science in Information Systems Management  
Ferris State University, 2012

Associate of Applied Science Computer Information Systems Programming  
Muskegon Community College, 2005

Associate of Applied Science Computer Information Systems Networking  
Muskegon Community College, 2005

Advisor:

Dr. Douglas L. Blakemore  
Full Professor

Accounting, Finance, and Information Systems Department

Fall Semester 2012  
Ferris State University  
Big Rapids, MI

Copyright: 2012 Adam M. Erickson

All Rights Reserved

### DEDICATION

I dedicate this to my incredible family. Particularly to my children, Scarlett, Charity, Azreal, and Micah, for they have brought fulfillment to my life. I must also dedicate this to my parents, whom have helped me along during my time at Ferris, and have given me their full support. Finally, I dedicate this thesis work to the former ones I love that have passed away, they have helped me choose my actions through life by there leadership, faith, and kindness.

### ACKNOWLEDGEMENTS

I would like to thank all the people who have helped make this thesis possible. I want to thank my advisors: Professor Dr. Barbara Ciaramitaro, Professor Gerald Emerick, and Professor Dr. Doug Blakemore. They have given a fantastic deal of guidance, encouragement, support, and patience during my coursework. Their genuine interests in information systems management and my education have been an absolute inspiration to me.

## Table of Contents

	Page
List of Tables .....	6
List of Figures .....	7
Abstract .....	9
Chapter 2 Background and Literature Review.....	15
Chapter 4 Details of Research.....	24
Chapter 5 Conclusion.....	42
References.....	46
Appendices.....	49

## List of Tables

	Page
Table 1 .....	21
Table 2 .....	22
Table 3 .....	31

## List of Figures

	Page
<i>Figure 1.</i> MySQL Command Line Client.....	26
<i>Figure 2.</i> PhpMyAdmin.....	27
<i>Image 3.</i> WampServer Options.....	28
<i>Image 4.</i> Eclipse for PHP Developers.....	29
<i>Figure 5.</i> MVC.....	34
<i>Figure 6.</i> User Table. ....	36
<i>Figure 7.</i> CSS ID Tags.....	37
<i>Figure 8.</i> CSS Link. ....	38
<i>Figure 9.</i> Index.php.....	38
<i>Figure 10.</i> PHP Login Form. ....	39
<i>Figure 11.</i> HTML Login Form. ....	40
<i>Figure 12.</i> PHP to Check for SQL Injection.....	40
<i>Figure 13.</i> PHP to Check if User Exist.....	41
<i>Figure 14.</i> Final Front End Display. ....	43
<i>Figure 15.</i> Index.php.....	49
<i>Figure 16.</i> Header.php. ....	49
<i>Figure 17.</i> Aside.php. ....	50
<i>Figure 18.</i> Content.php. ....	51
<i>Figure 19.</i> Footer.php. ....	52
<i>Figure 20.</i> Valid_data_fns.php. ....	53

<i>Figure 21.</i> Require_fns.php. ....	53
<i>Figure 22.</i> Database_fns.php. ....	54
<i>Figure 23.</i> Auth_user_fns.php. ....	54
<i>Figure 24.</i> Directory Hierarchy. ....	55
<i>Figure 25.</i> Create Database Script. ....	56
<i>Figure 26.</i> User Database ERD Diagram .....	59
<i>Figure 27:</i> CSS Script .....	60



### Abstract

The Content Management System (CMS) is a web based application using a Linux Server, Apache Web-server, MySQL Database, and PHP Programming Language (LAMP). The objective of managing users, and information in any given network environment can only be hindered by the creativity of an information technology professional and not by technology. The main objective of this thesis is to develop the early development steps of a CMS. By creating the building blocks for developing, and taking into consideration basic methods for creating the core platform of a CMS for further development. All information gathered, and experience gained will assist with developing and offering my own personal e-commerce business solutions in the future and to obtain additional business and practical knowledge in an open source software and e-commerce.

## Chapter 1 Introduction

There are several different commercial and Open Source Software (OSS) Content Management Systems (CMS) available on the market. A CMS can be downloaded and implemented in a test environment with only a small amount of knowledge in web development practices. Customers may not find what they are looking for in prebuilt solutions. Conforming within the limits of one CMS like Joomla, Xoops, Wordpress, Drupal, Zend Cart, and OS-Commerce can be frustrating. The process requires a large amount of knowledge in PHP programming to be able to create a personalized theme or manage personal modules as a client demands in a prebuilt CMS. Trying to change an existing application to the needs of the client can amount to a great education in the unfruitful endeavors in software implementations. When a professional tries to adjust to some other developers intended business model, the change never seems to have the exact requirements that a company is looking for. Creating a CMS may take a long time, but for the organization that has more time than money, developing may be the best option. Just about any project can be completed with enough time and knowledge of the requirements. Creating a CMS from scratch does not start out seemingly cost effective the potential of owning every aspect of a solution can be financial beneficial in the future as the application becomes malleable to an organization's requirements.

Having the architectural design and source code, allows an organization to make rapid changes that most closed source applications would never be able to offer without the source code. Source code is a very important possession to have; to a proprietary company that sells information technology source code can be their only source of

income. The endeavor of creating the solution from scratch allows the source code to be available forever within the company. The source code could allow for an organization to effectively, provide a solution to the requirements that a company is seeking. Regardless of whether the solution is for a typical business analysis report, or if the solution is allowing for precise customizable functional designs. The CMS will have a significant better chance of evolving within the organization during the development of the website that fits an organization's web based needs.

### **Background of the Problem**

Commercial and open source companies create modules and components for users to implement within their personal CMS. The availability of open source solutions is ever increasing because of the creation of new and free applications made from the many different demands of users. During a project endeavor to create the solution a client will want an excellent product that is cheap, and they want it yesterday.

Developing a web site can take a few months to a few years when there are factors in design, aesthetics, and functionality to be considered among several different groups within an organization. While development can take a lot of man hours to complete, there can be trivial to no overhead in software costs to users because licensing costs after initial development should be of little concern. A person can learn everything they need to know about web development on the Internet, but it may help to purchase education books.

### **Statement of the Problem**

The problem addressed in this study is the ability to create the core requirements of a CMS for organizations using all OSS technology with an emphasis on being open object oriented within a Model View Control/agile framework, to provide a web based application that will meet the needs of a business and be easily to moderate in the future.

### **Objectives of the Study**

The objective of this exercise is to analyze the beginning stages of development and establish the initial programming of a core CMS that is PHP driven. The CMS will be a website that will allow for additional plug-ins. By creating the foundation of the CMS, I will have the opportunity to modify the projects PHP source code and quickly make changes to the design. The final product of the core CMS may also be used for educational purposes.

### **Rationale for the Study**

To research conventional methods to design a core CMS using current application design methodology while focusing on a platform that uses Linux, Apache, MySQL, and PHP. The front end solution will use HTML in order to determine current best practices for web development and content management systems.

### **Research Question**

How to build a core CMS with best practices implemented using PHP, MySQL and HTML?

### **Nature of the Study**

The nature of this study is to complete a comparative analysis of best practices in web development when building the core system of a content management system. The study

will also be a project to establish the beginning of a core content management system using PHP, HTML, and MySQL on an Apache Web Server running on a CentOS Linux Server.

### **Significance of the Study**

The significance of the study is to continue the development of knowledge that can be leveraged to gain employment within a company that requires in depth comprehension of how open source solutions can be leveraged. Research will also establish additional educational material for future research publications in web development to create solutions for organizations.

### **Definition of Terms**

CSS - Cascading Stylesheets Sheets helps format the look and feel of an HTML document through presentation semantics.

CMS - Content Management System is a computer program that allows publishing, editing and modifying content within the application.

CRUD - Create, Reuse, Update, Delete are the four basic functions of a persistent storage implementation.

HTML - Hypertext Markup Language, the main language for displaying web content in a web browser.

IDE - Integrated Development Environment, a program to promote the complexities of developing comprehensive programs.

ISS - Internet Information Server is the web server that comes pre-packaged within windows servers.

JavaScript – A scripting language that can be considered to implement to add functionality and user enhancements in a web page.

LAMP - A term used for the platform of servers that host a web application a LAMP stack consists of a Linux, Apache, MySQL, and PHP server.

MVC - Model View Control is an architecture design for separating different interactions within a program.

OOP - Object Oriented Programming using class object to extract instances of a section of code when necessary using techniques like inheritance to different classes logically.

PHP - Hypertext Preprocessor is a server side embedded scripting language.

WAMP - A term used for the principles of servers that host a web application a WAMP stack consists of a Windows Apache, MySQL, and PHP server.

XML - Extensible Markup Language is a markup language that helps determine the rules for encoding documents.

### **Assumptions and Limitations**

While most people prefer to save money, there is sometime a need for speed or the inability to provide web developers with the physical resources needed to implement a free solution. There is a minimal amount of up to date available information in academic libraries to provide adequate research material. Most research will be obtained through the Internet, and personally purchased reading material.

## Chapter 2 Background and Literature Review

The Internet since its humble beginnings in 1969, when it was a government funded communications network called ARPAnet, is constantly evolving (McKenzie, 2011). Creating the Internet into the experience that it is today was also helped along by some very huge companies like AT&T. Personal interest groups and academic organizations also took a large role in transforming the Internet into what we see today (Crovitz, 2012). The Internet, in its infancy, was just a textual messaging system. User requirements grew for pages that use a markup language for formatting the pages instead of explicit text pages. Formatting text displayed in a browser was accomplished using Hypertext Markup Language (HTML) to create a solution to the problem of viewing clear text formats. As interests in the Internet increased, so did the need for the Internet to necessitate interaction for users with a graphical user environment. Web pages could not be static anymore as business pushed for the ability to do more than provide information and be able to offer online sales and services with interactive pages using methods like JavaScript, Adobe Flash and Dynamic-HTML. A most significant change in how we use the Internet was the beginning of server-side scripting languages. Server-side scripting languages gave developers the ability to use dynamic web pages and create transactional-oriented web pages. Production companies have used server-side scripting language extensively in the production. Some of the larger corporations that offer these scripting languages include Oracle's Java Server Pages (JSP), open source Perl, Microsoft's Active Server Pages (ASP), and the open source Hypertext Preprocessor (PHP), (Suraski, 2002). Scripting languages allow user access to data bases that opened the door for how we use

the Internet today with electronic commerce (e-commerce). The popularity of Open Source Software (OSS) when developing web pages has continued to grow. OSS solutions are not implemented quickly like proprietary software. OSS does not have as many positive paid reviews as mature software programming model. This may be the reason that many professional may no implement OSS solutions. Because of the lack of the same marketing strategy that is available to proprietary providers that invest a significant amount of capital into marketing there product (Gittens, 2007).

### **Open Source Software**

OSS is a set of practices used to collaborate with software source code that is freely available through copywriting laws. OSS can come from individuals separated by various cultural, corporate boundaries, language and other characteristics in order to work together to create complex, non-proprietary software. Software is open sourced when it is free to redistribute the source code with it as well as in compiled mode. The open source licensing creates a way to get the source code of a program readily available to anyone that request access. OSS protects the original author of the software, does not discriminate in any way on how it can be used, cannot be specific to a product or software, cannot restrict other software and has to be technology neutral (open source, n.d.).

Open Source Software (OSS), is the most viable solution for creating a CMS because of its popularity, performance, and cost effectiveness. As of September 2012, Sourceforge.net's website proudly boasts numbers of 3.4 million open source developers working on over 324,000 OSS projects with an average of 4 million OSS downloads per



day. Most popular OSS software that can be easily obtained and utilized on the Internet is the Linux OS (Operating System), Apache Web Server, MySQL Database, and the Hypertext Preprocessing Language (PHP). There are several other applications that can be used to develop websites the grouping of Linux, Apache, MySQL and PHP has grown in popularity that it now has its own acronym, LAMP. LAMP covers on virtually all popular web hosting sites. Other open source software like Java, the Tomcat Server, and proprietary software like Microsoft's ISS is not as popular because of the increase in cost to run a web based platform.

Some of the more widely known open source licenses include the GNU (Graphical Environment of Linux Servers and Desktops); Mozilla (Firefox, Thunderbird), MIT, BSD (like UNIX), and Eclipse (Eclipse IDE). Because of the lack of dependence on software vendors, open source software allows the software to transform and morph into potentially anything the users and developers lack the finances to purchase.

### **Apache web server**

Apache the HTTP web server is continuing to be one of the most popular web servers for hosting web content. Wither it is in commercial or open source terms (Open platforms, 2012). Apache Web Server is the 1995 collaborative project that offers the source code for free to the HTTP Web server (Apache, n.d.).

### **MYSQL Database**

MySQL is the world's most popular database and is powering some of the largest companies online, including Facebook, Google, Amazon, and Yahoo (Henschen, 2011). It is an outstanding combination of fast SQL queries with access to the source code for

free has made it another success like Apache and PHP with over sixty five thousand downloads per day (Mysql, n.d.).

### **PHP: Hypertext preprocessor**

The hypertext preprocessing language (PHP) is an open sourced product that lacks the marketing power from companies like Adobe Flash and Oracle's Java. It is a server-side multi-platform, scripting language that can be embedded within HTML. Unlike JSP, (Java Server Pages), and ASP (Active Server Pages), PHP is a language for a task of making web sites. Strongly acknowledged as bundled part of the Apache Webserver, PHP can be installed on other Web servers (Kevin, 2002). Thanks to companies like Red Hat and Zend, the popularity of PHP is increasing as they offer built in platforms that provide support for PHP developers (Red hat, 2012). Some of PHP's strengths are that it is extraordinarily fast and can serve millions of hits per day. PHP is free so that it can inexpensively be deployed horizontally scaling within a large number of servers. PHP can be integrated with many database systems within the PHP libraries there is also a built in libraries for common Internet tasks like parsing XML, sending email, working with cookies, generating PDF documents, and all with just a small amount of additional code. Other reasons that PHP is growing in popularity is its portability between systems, the flexibility of development approaches, the similarities between Java and C programming, and availability of supporting documentation and support (Welling & Thomson L, 2009).

### **Integrated Development Environment**

Developers use IDE (Integrated Development Environment) platforms to help developers organize their programming lines of code file structure. IDE's are not available for all languages. IDE's become more popular as the programming language matures and there is stronger need to manage the lines of code. IDE's to help structure and organize the system blocks for any given complex application. Several commercially paid, and free versions exist for PHP that can be either downloaded for free or purchased. A couple of the most popular IDE's is Zend Studio's Eclipse version and PhpStorm. Zend Studio is from the creators of PHP and ideal for developers who are already familiar with Eclipse or IBM's commercial version Rational (Wayner, 2012).

### **Model View Control**

In the classic MVC (model, view, control) methodology, the controller is primarily the mediator between the user's actions through the view of the user interface. In the case of web development, this would be the web browser, and the domain logic, calls to the database, captured in the Model (Shaw, 2012). Some of the competing software already uses the MVC methodology. Microsoft uses an ASP to create their MVC platform. The name of the platform ASP.net MVC 4 was first offered in 2010 (Krill, 2012).

### **Chapter 3 Research Materials and Methods**

The project is to implement a core content management system using PHP, HTML, and CSS with MySQL database, Linux CentOS and Apache servers. It will be a launching ground to recognize today's best practices in web development and security when developing an interactive website. The research question below is answered using the implementation of the project model.

#### **Research Question**

How to build a core CMS with best practices implemented using PHP, MySQL and HTML?

#### **Methodology**

Methodology to answer the research question will be using qualitative analysis and synthesis of other people's research. This will allow the discovery of useful information to implement a project design that can be used to analyze the components as part of the project.

In order to satisfy my stated research question I will be conducting a comprehensive search for relevant studies and literature reviews found in different stages and using the following sources:

- Google Scholar
- Google
- ABI/INFORM Global
- Current books available in web development, PHP, and CMS

Collected information on studies that examine the research question will then be analyzed for information from acceptable studies and decisions. A statement made

highlighting implications and alternative solutions when available, and analyzing popular CMS sites (Wordpress, Drupal and Joomla), to determine design factors like the directory structure on the server and development of SQL tables.

Research will include search strings like “PHP web development evaluation”, “PHP MVC templating”, and “comparison of PHP4 and PHP5”. There were several other results found, but few that were relevant because they were too general or did not cover the stated research question. Common used search strings can be found in the following Table 1.

Table 1

*Search Strings Used*

Search Strings Used
PHP5 web development
Comparison of PHP4 and PHP5
PHP MVC web development
PHP CMS web development
Web development best practices
PHP best practices
PHP framework evaluation
CMS best practices
MySQL and PHP

In order to implement a solution for creating the content management system using conventional and available technology, common development solutions, and compared against other development resources for evaluation. All development coding and resources must be able to work within the testing environment.

Once completed, checks to make sure that the solution can be implemented on a LAMP web hosting platform, and security measures are in place. The following Table 2 lists the server information requirements that will be used as a testing framework for my research project.

Table 2

*System Requirements*

System Requirements	
Name/Version	Comments
Linux x86 CentOS 5.5	Sever
Apache 2.2.21	Web server
PHP 5.3.10	Coding
MySQL 5.1.52	Database

Model View Control methodology will be a part of the process and will prove that PHP can be used for the modeling and the viewing without the need of a templating engine.

The stable release of PHP is an acceptable practice with several security options, not in previous releases. The release of PHP5 has several security and development

improvements that utilized within the CMS development stages. Comparison on programming and security can be taken from the developer's website along with publications and well established developing websites within the web development community.

To determine what applications should be used for developing websites with PHP, I will be comparing and contrasting the most commonly used methods. Conclusion based on my own experience and what is available through online discussions will allow me to come to my own conclusions.

**Milestones**

1. Determining what software, by criteria, can be leveraged to develop a PHP website
2. Directory Structuring on the server for security and manageability of code
3. Determining if there is a need to use a templating engine
4. Example SQL syntax
5. A HTML5 structuring with CSS3 template for a welcome page
6. Example PHP coding for user management

## Chapter 4 Details of Research

### **Introduction**

Running through several sites and books it took several attempts and failures to determine the appropriate methods of creating a basic content management site that uses PHP5, CSS, and HTML. Several options are available for creating a site, and the hardest steps to determine are the beginning ones. Having the correct PHP, web-server, and database settings took the longest time. It took weeks to decide on the structure of the website's directory hierarchy. The decision to develop user login requirements in the beginning was to handle the first step of a CMS, the user login form. The login form is the first step required when a user reaches a protected site. When the user logs-in they will have access to the content available.

### **Applications**

The main constraint for development of a CMS is cost, every tool that is available I want available to anyone with access to a computer and the Internet to be able to duplicate my choices. The second constraint is the popularity because the popular the tool, the more options, references, and communities and web-hosting sites that provide the same solutions I propose are available on the Internet. Keeping the solution free from commercial applications was to promote the development within a framework that can be attained by anyone. People that can not provide start up monies in order to educate themselves. Freely available software can help develop future developers who may not believe they have the tools on hand. Because most other talked about frameworks, take a lot of money or education to begin development. One of the most expensive IDE's would

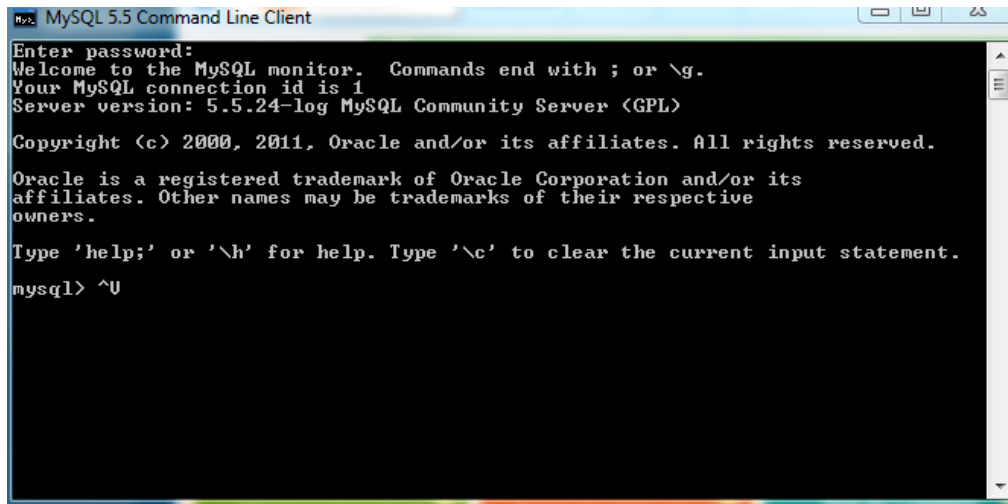


be for Microsoft Visual Basic. Availabilities of even less expensive programming languages like Java still require a considerable more amount of education and web-hosting costs, like for a Tomcat web-server. Most web hosting companies offer a LAMP stack for development web pages and some of the most popular content management systems like Joomla, Drupal, and Wordpress also use the LAMP stack because it is FOSS (free open source software). This is the reasoning that I have chosen to develop my CMS. To develop around software that is easily accessible, free to use and distribute, with a broad community base online for referencing and support.

In order to develop a PHP system, the servers present need to work with PHP. The PHP server can be installed on Apache or IIS (Internet Information Server) Web Servers, created to run strictly with Apache and when used in conjunction with IIS, usually creates create problems and further editing of configuration files are necessary for it to work correctly with IIS after installation of the a PHP server.

In order to track, monitor, and modify user's content based on what a user wants it is necessary to have some way to have a straightforward way to store and access information. This is where SQL (Structured Query Language) databases come in, because of the amount of information that can be quickly accessed as a system grows by a SQL server. Data is more easily to structure and also to secure with SQL. Just about any database can be accessed and modified through PHP including MS-SQL, Oracle, and DB2. Because MySQL is free to download, MySQL is the most popular to use in web development. MySQL is one of the few databases that offered by web-hosting companies for no additional charge, and there is a free web based graphic environment plug-in to

modify the MySQL database. The following Figure 1 is a demonstration of what MySQL access looks like from the admin console. Notice the lack of ability to accomplish anything easily from a novice's perspective.



*Figure 1.* MySQL Command Line Client.

Even experienced developers prefer to build their databases using PhpMyAdmin (<http://www.phpmyadmin.net>). PhpMyAdmin is a free tool written in PHP that helps manage the administration of a MySQL server by offering support through a wide range of operating inside a MySQL server. SQL operations that can be used through the user interface can assist in managing databases, tables, fields, relations, indexes, user permissions and even execute directly to the MySQL server SQL syntax statements. The below Figure 2 displays a few options of how phpMyAdmin can be viewed compared to the command line tool offered with MySQL.

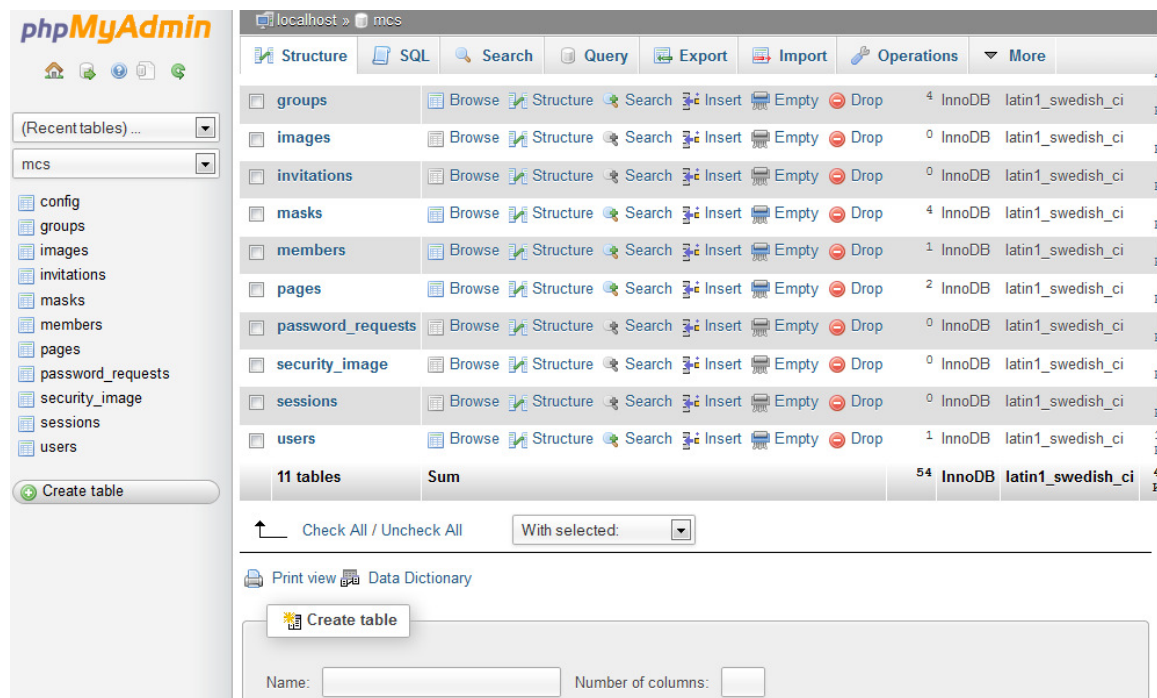


Figure 2. PhpMyAdmin.

Because I do not want my trial and errors seen by anyone on the Internet I have postponed moving my development to the Internet until next year., If all things go well, I will move all my Wordpress documents to my own personally built CMS. Until then, I need to have a system ready that can reproduce the options available with a web hosting site like Hostgator.com and GoDaddy.com. This means selecting software that can run on my computer. A WAMP (Windows, Apache, MySQL, and PHP) server that could run on a Windows 7 platform and give me all the options that I would expect from an LAMP web-serving host would need to be chosen. Two of the most popular software that is available for free downloads are XAMPP, and WampServer. XAMPP for windows offers several options including an FTP server, Tomcat server for Java, and Perl. WampServer offers just the basic functions I was looking for; an updated version of PHP, Apache and

MySQL. XAMMP was not as easy to configure. XAMMP tries to install Apache and MySQL as services on the computer which was extremely volatile because of the security settings in Windows 7. The configurable, settings seen in the following Image 3 for WampServer, worked flawlessly and was remarkably straightforward to configure. A version to download can be found at this link: <http://www.wampserver.com/en/>.

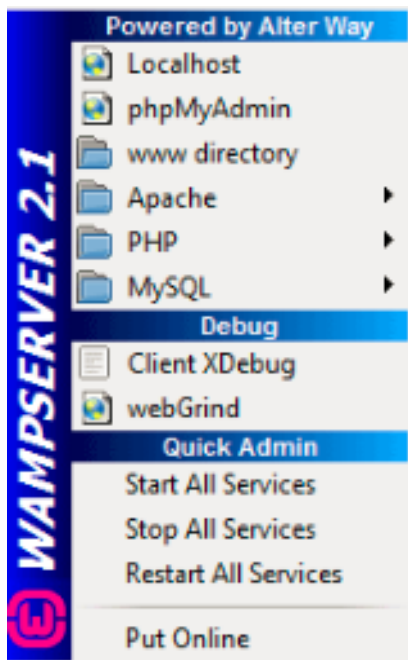


Image 3. WampServer Options.

With financial limitations of various potential developers in the world including this author, I have chosen to go with a free Integrated Developing Environment Platform (IDE) because developing a system as comprehensive as a CMS, would be extremely complicated to develop using only text files, an IDE can be particularly useful. One of the most popular IDE's that is also extensively used for Android development, Java, XML, and PHP is Eclipse. Eclipse, shown below in Image 4, is the free version of IBM's

Rational IDE, released to the public as open source software to further promoted the development of Rational. In the several years since Eclipse's release, there have been several different versions released. With the right plug-ins, it allows users to view syntax highlighting, can help with code completion, navigation, errors and warning highlighting, debugging, refactoring and code generation. It can be used by individuals and companies world wide.

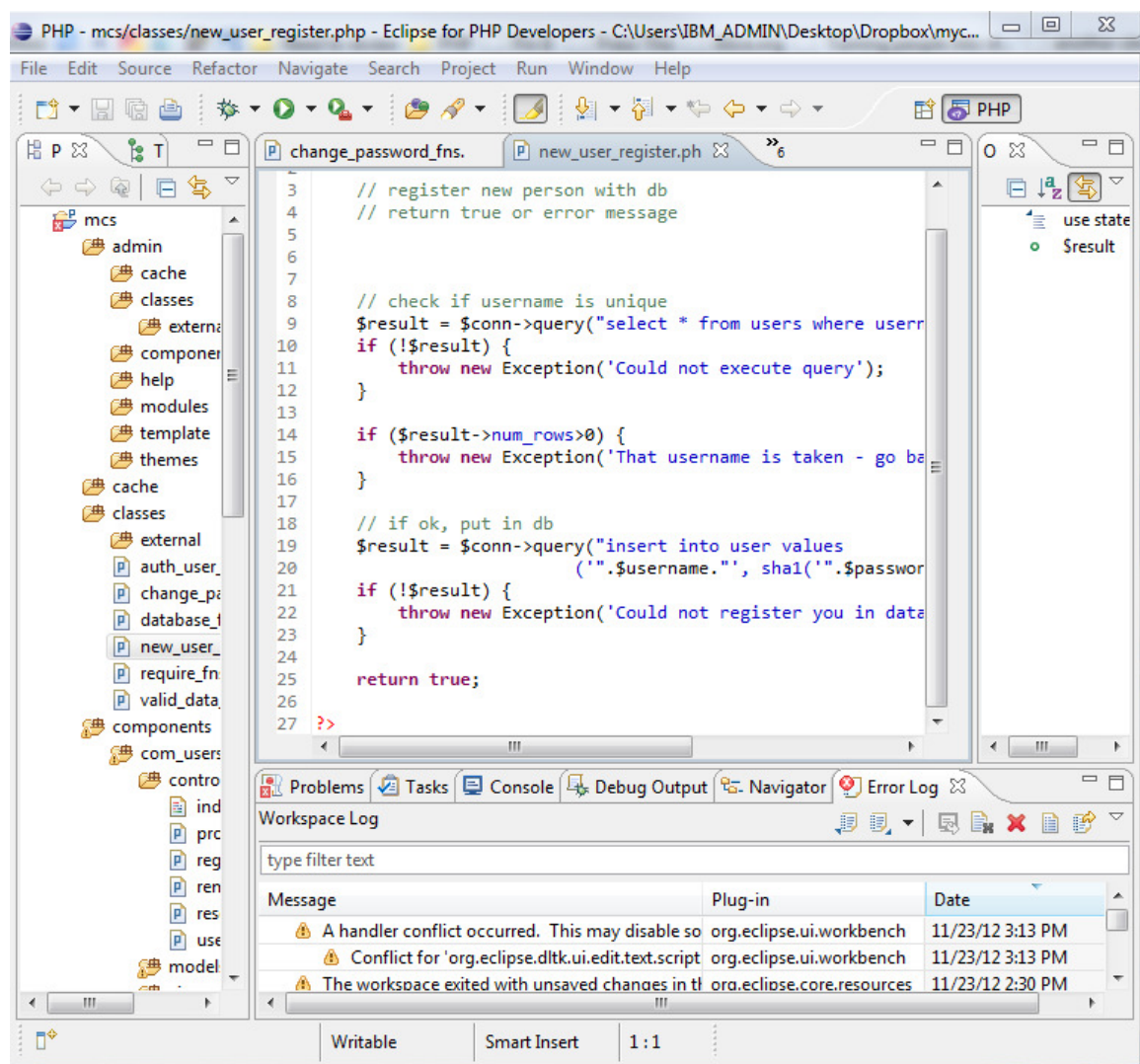


Image 4. Eclipse for PHP Developers.

The other popular IDE for PHP development is NetBeans by Oracle. NetBeans and Eclipse are both downloadable for free, but Eclipse is what I have been training on for years. In the end, it is primarily a decision that can be made personally. Both products provide the same functionality. The free download of Eclipse with Zend allows for PHP debugging, compiling warnings, and syntax highlighting (Zend.com). NetBeans can be downloaded for free at: [www.netbeans.org](http://www.netbeans.org).

### **Folder Hierarchy**

After choosing the software to use, there is still a considerable amount of options to consider programming can begin. A great amount of time has to go into understanding the amount of possible information that may be presented as the solution matures. Having a straightforward way to manage all the different files that are within an organization is also necessary. Building a directory structure that makes browsing and storing information easier for the developer is particularly useful. In order to determine how to partitioned information into a directory structure, it can help to imagine ways that it may be useful. One way is creating user rights that require security checks on folders in a CMS on the server side. Knowing that everything an administrator can do is all located within the administrator section. Creates less of a problem for determining what should be changed to what should not. It also allows determining what rights certain files should be had if they should be read, write or execute only. For a designer on a CMS project, it helps determine what files should be worked on from ones that should be left alone.

A directory structure can be highly subjective but should make sense to everyone that will be using it, the programmers, and designers. In the below table, is the basic folders that can be designated for use when building the CMS. All folders may not be used in the beginning, but forethought on the growth and complexity of the application in the future should be taken into consideration. Table 3 below illustrates the folders with the logical explanation for choosing these folders.

Table 3

*Directory in Web Server*

<b>Folder Name</b>	<b>Folder definition/Reasoning</b>
admin	The place to configure all website maintenance
cache	Cache is a place to store previously used views so that PHP does not have to build the code from scratch again and database queries.
classes	When implementing personal classes for OOP, this is internal classes would be stored.
components	Major extension, an elaborate arrangement that can be broken up into less complex components. This is how MVC can be broken up.
configs	Stores additional configuration settings
editor	This folder can be used to store an external WYSIWYG editor.
extClasses	Classes created by others that may be added to the CMS.

help	A location to keep all help files. Proper documentation here can help a lot in the future.
images	A place to store uploaded images.
includes	Functions and other files that handle the arrangement and management of information besides the classes and components.
media	Like the editor folder, media is for editing and management media.
modules	Minor extensions that create small screen boxes with useful information like who is online, but not necessary like components.
params	Static parameters for some functions and classes can be stored here for easy access.
templates	Besides for the admin templates this folder will hold all the other templates for changing the look of the CMS through HTML, JavaScript, and CSS

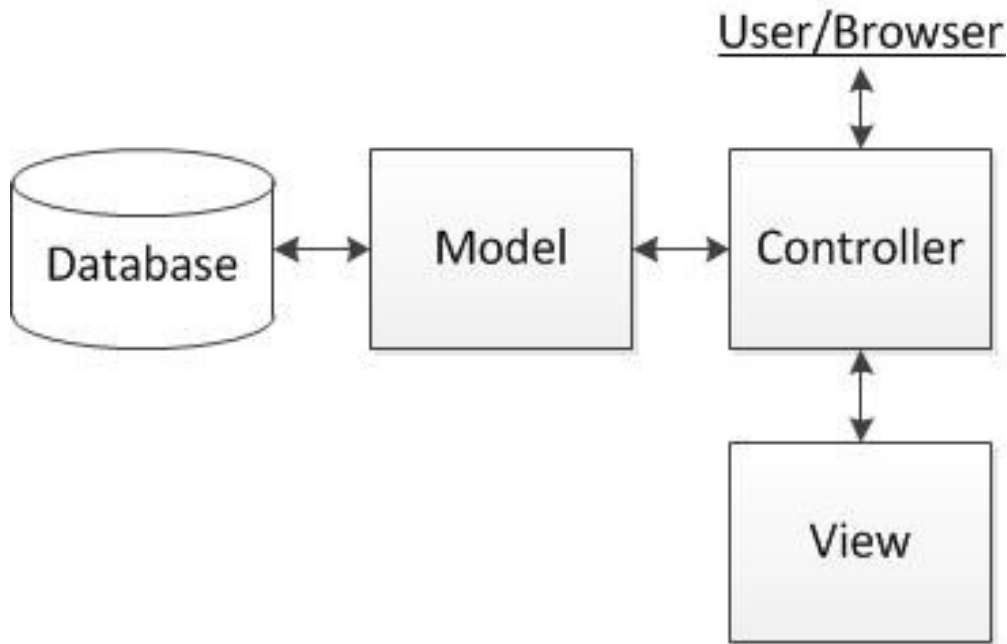
The creation of a folder hierarchy helps to differentiate certain sections from other. One example is the admin folder. The admin folder maintains separate modules, components, help files, media, editor, and template files. Separating content into other folders helps securing content for server administrators. Areas of the CMS can be differentiating between user access and admin access by folder areas. Within



components, there is a separate folder within each component. Under the component for users, there are three folders used to create a MVC that will be necessary for separating design, from the data, and the controllers that call upon the data. A visual representation of the folder hierarchy in a folder view can be found in appendix B,

### **MVC and Templating**

As discussed, because code can start to become exceedingly complex as the project grows it becomes more and more beneficial to separate business logic and presentation data separately. The Model will contain the methods used to call on the database and make add, remove update calls to the database. The Controller is responsible in the application for holding all other PHP code that does not directly manipulate data from the database. An example would be code that answer requests from the user and calls on the model to modify the stored data. The View is part of the programming that will represent the HTML. The View is commands called from the controller that requested methods from the Model to display through rendered HTML, JavaScript, CSS and/or images as seen in the below Figure 5.



*Figure 5. MVC.*

A templating engine within the View can assist in separating presentation logic and existing business logic. A good PHP programmer should be able to separate the two in separate files without needing a templating engine. The templating engine is just one more step of coding that would need to be learned in order to develop the CMS. The CMS would also be dependent on an additional layer and would cause concern about the templating engine needing updates. While a templating engine looks good in paper, I have come to the conclusion that it is an unnecessary step in creating a CMS. PHP's history started out as being a templating engine. While a templating engine could reduce errors from designers, there is no reason to begin to make another processing layer to the PHP code when PHP is still supremely capable of templating on its own. Making sure that it upholds to the MVC methodology and keeps PHP that deals with only viewing information in the Viewing folder of the systems components.

**Database**

Database creation is necessary to define before coding even begins if not done right it can hinder the ability of the programmer to write good code. Take for instance the ability of PHP to take a password from a form and have it transformed into a MD5 128-bit hash value before passing it into the database. A 128-bit hash requires 32 characters in a database field to work, if the database can be set up without the ability to store a 32 character password, it would fail in the beginning. Another to take into consideration is types of data fields that may be created in the near future. While the database can always be changed later if the fields are already available, it takes away some of the unnecessary time of re-visiting the database to make corrections. There is also the possibility of circumventing the need to fix additional mistakes that may be created when modifying the database.

MySQL SQL offers the same standard data types and syntax that is available in other databases. The below Figure 6, it displays the values set for the user table of my database.

```

1 CREATE TABLE IF NOT EXISTS `users` (
2   `id` int(11) NOT NULL AUTO_INCREMENT,
3   `username` varchar(255) DEFAULT '',
4   `password` varchar(255) NOT NULL,
5   `active` char(3) DEFAULT 'no',
6   `last_login` int(11) DEFAULT '0',
7   `last_session` varchar(40) DEFAULT NULL,
8   `blocked` char(3) DEFAULT 'no',
9   `tries` tinyint(2) DEFAULT '0',
10  `last_try` int(11) DEFAULT '0',
11  `email` varchar(255) DEFAULT '0',
12  `group_id` smallint(6) DEFAULT '2',
13  `activation_time` int(11) DEFAULT '0',
14  `last_action` int(11) DEFAULT '0',
15  PRIMARY KEY (`id`),
16  UNIQUE KEY `username` (`username`),
17  UNIQUE KEY `email` (`email`),
18  KEY `users_idx` (`username`),
19 ) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;

```

Figure 6. User Table.

As previously discussed the 32 character data set for the password field is missing. Instead, it is a 255 variable character data set; in case there is a change of mind on how to handle passwords in the future. It will be able to do a change without forcing a person into resetting their password the next time they visit the site. Not all fields will need to be used in the beginning; the fields used are the id, username, and password. The other fields can be used to better track users to:

See how many times a user has tried to log in

To send emails reminders/notifications

See if the user is active

Promote a user to the admin group

See how long a user has stayed on the site

Know last actions a user had on the site

## HTML5 & CSS

HTML5 is the new standard for hypertext markup language. It offers the ability to support CSS3 fully and offers new elements and attributes. In order to understand how I have created my index.php, the first page of the site displays the basics of how HTML can be used. The standard blocks used in most of an HTML5 page can be viewed in the next figure, Figure 7.

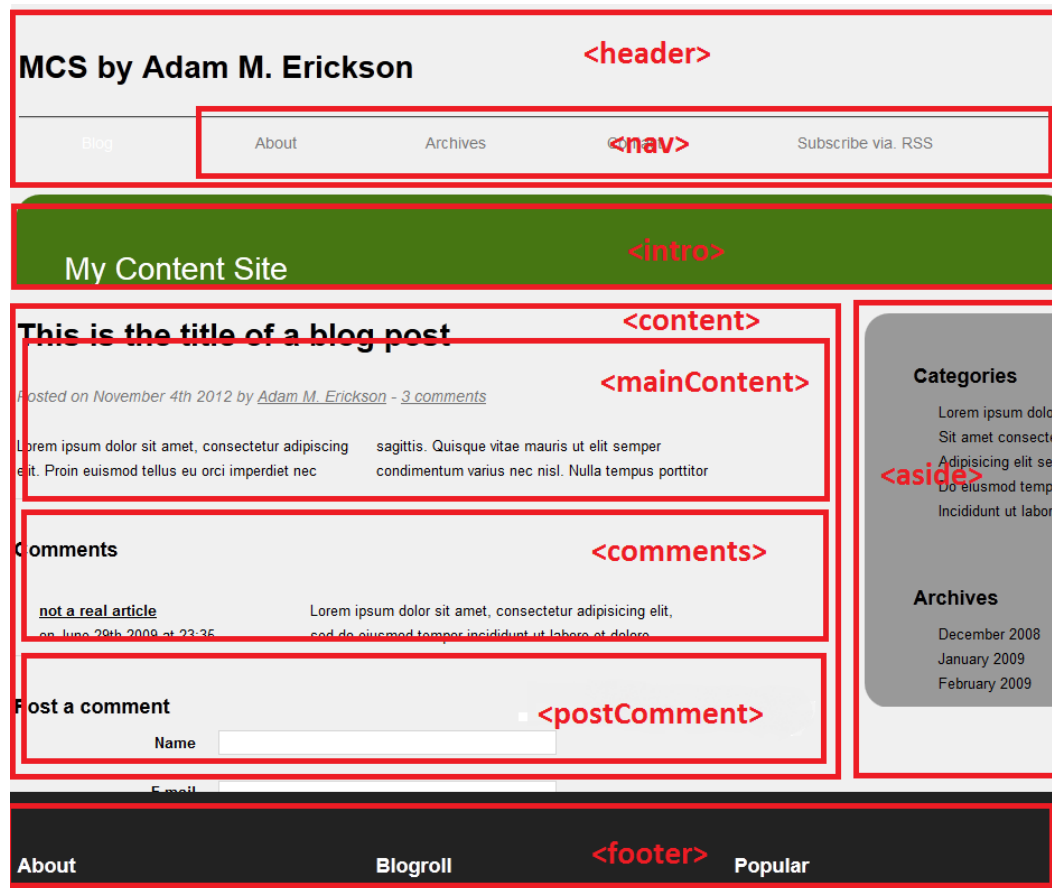


Figure 7. CSS ID Tags.

By using the `<div>` tag and assigning different id's to the `<div>` tags, it allows to structure the theme from a single file in the CSS (content style sheets) that is a link to the index page using the below HTML code example displayed in Figure 8.

```
<link rel="stylesheet" href="/themes/mcs/css/style1.css" type="text/css"
      media="screen" />
<body>
```

Figure 8. CSS Link.

## PHP Code Examples

It is indispensable to have a general example of HTML blocks that are within the CMS so that the PHP program can be better understood. Using the *include\_once* command can separate logical parts of code and HTML sections. Figure 9 below displays how the commands in index.php can be used to view several pages in one by linking several different pages together from within the CMS directory.

```
1 <!doctype html>
2 <html lang="en">
3
4 <?php
5
6 session_start();
7
8 if (!(isset($_SESSION['valid_user']) && $_SESSION['valid_user'] != '')) {
9     header ("Location: ../modules/mod_login/login.php");
10 }
11 echo ('<p>Welcome ' . $_SESSION['valid_user'] . ' :</p>');
12
13
14 include_once 'header.php';
15 include_once 'nav.php';
16 include_once 'intro.php';
17 include_once 'content.php';
18 include_once 'aside.php';
19 include_once 'footer.php';
20 ?>
```

Figure 9. Index.php.

Line 6 in index page above demonstrates how to store unique user information for each user when they authenticate on the CMS. Line 8 checks to see if the user has logged

in if they are it prints a welcome message at the top of the screen. User will have a password email reminding them of their password if forgotten. Additional options for the user are to login again or register. The login page can be broken into several different PHP pages. Shown below is the next page of CMS (*login.php*) in Figure 10 contains the form needed to login.

```

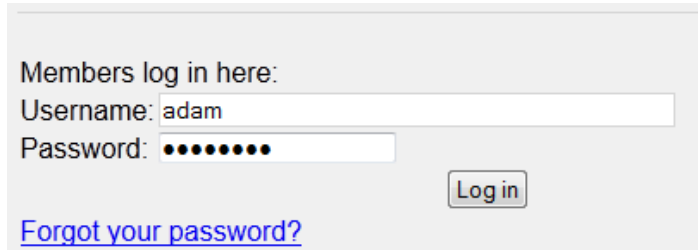
1  <?php
6  <p>
9  <form method="post" action="../../classes/auth_user_fns.php">
10 <table>
11 <tr>
12 <td colspan="2">Members log in here:</td>
13 </tr>
14 <tr>
15 <td>Username:</td>
16 <td><input type="text" name="user" /></td>
17 </tr>
18 <tr>
19 <td>Password:</td>
20 <td><input type="password" name="password" /></td>
21 </tr>
22 <tr>
23 <td colspan="2" align="center"><input type="submit"
24 value="Log in" /></td>
25 </tr>
26 <tr>
27 <td colspan="2"><a href="forgot_form.php">Forgot your
28 password?</a></td>
29 </tr>
30 </table>
31 </form>
32 <?php

```

Figure 10. PHP Login Form.

The PHP has ended, and normal constructs of a basic HTML form shown below in Figure 11, presents how the data passes to the browser with the POST function of the

HTML form. The HTML form will send to the PHP page the session variables to be used against the database for authentication.



Members log in here:

Username:

Password:

[Forgot your password?](#)

Figure 11. HTML Login Form.

The user enters in their user name and password, and select *log in*, which send the information to *auth\_user\_fns.php* script that includes the *database\_fns.php* script used to connect to the database. The commands *stripslashes()* and *mysql\_real\_escape\_string()* in the function below scrubs the user name and password. Scrubbing cleans PHP form fields that posts data to a SQL server. Scrubbing stops hackers from penetrating the database for to steal information through SQL injections. A demonstration can be seen in the following Figure 12.

```

2
3 //get database connection info
4 require_once 'database_fns.php';
5
6 // Define $myusername and $mypassword
7 $myusername=$_POST['user'];
8 $mypassword=$_POST['password'];
9
10 // To protect MySQL injection
11 $myusername = stripslashes($myusername);
12 $mypassword = stripslashes($mypassword);
13 $myusername = mysql_real_escape_string($myusername);
14 $mypassword = mysql_real_escape_string($mypassword);
15 $sql="SELECT * FROM $tbl_name WHERE username='$myusername' and password='$mypassword'";
16 $result=mysql_query($sql);
17

```

Figure 12. PHP to Check for SQL Injection.



The user name and password can then be checked against every user name in the database until it finds one match and only one match. The PHP code marks the user, as a valid user, and directs them to the index page. Next the user can begin to browse, or modify pages based on their rights. The coding displayed in the following Figure 13.

```
18 // Mysql_num_row is counting table row
19 $count=mysql_num_rows($result);
20
21 // If result matched $myusername and $mypassword, table row must be 1 row
22 if($count==1){
23     session_start();
24     $_SESSION["myusername"] = $myusername;
25     $_SESSION["password"] = $mypassword;
26     $_SESSION["valid_user"] = $myusername;
27     header("location:../index.php");
28
29 }
30 // Tell user do something else cause you're not logged in.
31 else {
32     echo "Wrong Username or Password<br />";
33     echo "You could not be logged in.<br />";
34     echo "<a href=\"forgot_form.php\">Forgot your password?</a><br />";
35     echo "<a href=\"register_form.php\">Not a member?</a><br />";
36 }
37 ?>
38
```

Figure 13. PHP to Check if User Exist.

Additional PHP examples, database entity relation database, CSS sections can be found in the appendices for further information on how I have developed the initial CMS.

## Chapter 5 Conclusion

While CMS sites are comparatively uncomplicated to use once created, I do want to note that initial installation, development, and configuration will be more difficult than just throwing a few PHP, CSS, HTML and JavaScript elements together. Most developers have focused their experience on either front-end or back-end development. Because of all the necessary steps, to build a web based information architecture that is for an application to be secure and free of bugs. Anyone that takes on the action of creating a fully functional CMS should look to have an extended network of colleagues to collect best know practices, and other additional programming knowledge. Any one who endeavors to create a CMS on the web using a commercial website hosting company. Should look for a hosting company with interfaces for easy installation and management of MySQL and PHP plug-ins. Before PHP and HTML can be fully utilized, there is a learning curve for configuring and setting up Apache, PHP, and MySQL configurations and security. Using PHP there is a possibility to design a wide range of Internet websites. The developer of an interactive website will have to make many decisions about what functionality and user interfaces to use and allow.

The following Figure 14 displays how my first build of the website will look like from the front end.

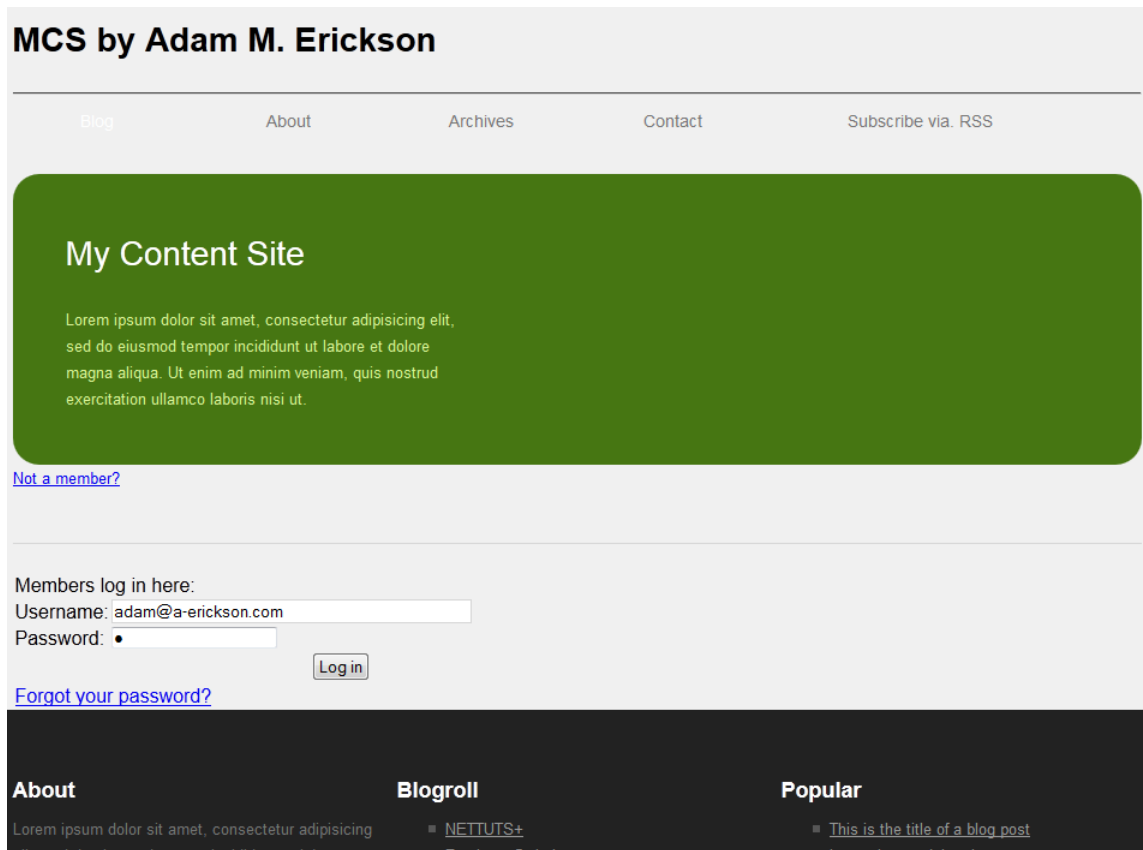


Figure 14. Final Front End Display.

## Conclusion

Any web development project does take a high level of experience in many different areas. Building a large scale CMS is like building a house. It would take an extraordinarily long time and a lot of training for only one person. Most successful large project in web development may section portions off into smaller manageable groups, depending on the amount of technical experience that is necessary. Unlike other programming languages that run through a compiler before shipping to the client, PHP source code is available for the client to open up and take advantage of a programmer's

code to further their own work. This could be looked at as either an advantage for promoting sales or a disadvantage because one can not secure the code after distributing a project to a client.

PHP is primarily a web development language with many abilities within the core, and additional libraries have not been introduced because it is primarily a web development tool, a good understanding of other technologies like Bash, Cron, Perl, CSS, JavaScript, and HTML are necessary in order to deliver a commercial product.

This author would define himself as an intermediate web developer when developing PHP at the start of the project. With future study and implementation of different methods of PHP coding, security, and creating functioning sites that offer users different services I plan to become an expert there are several different reasons why I have selected this topic. Future work is to create documentation for publishing that can help to educate students on how to develop interactive web pages that follow object oriented programming (for advanced classes) using best known practices in the World Wide Web and information architecture.

After creating the base system, the CMS can be utilized to create a system available in today's market. Some of the applications that can be envisioned and created can be a document repository system, client management system, online sales management system, a company's intranet and/or Internet site.

### **Suggestions for Further Study**

Future work that will add functionality to a CMS and also incorporate further study could also include introducing the PEAR library framework of reusable PHP components

that offer the ability to for PHP that comes with extensive functionality for PHP developers like the ability to authenticate to many different types of operating system authentication servers. This would be immensely useful because companies ask that any new applications use the same single sign on for security that they are already using. Wither it is working with Active Directory, Radius, LiveUser or OpenID. Additionally the PEAR library offers prepackaged code that will manage XML, caching, event handling, file systems, logging, mail, math, numbers, and other functions that found in other programming languages like Java, yet tailored specifically for web applications and web service like Amazon.

When learning about the technology that drives a CMS, one can begin to discover all the addition work that goes into the creation. The ability to create information architecture around a large CMS is imperative for functionality of the system to work correctly. The technology and methods that drive the Internet are numerous, and there is always a new method for web security and web development presented every year that needs to be evaluated. Make sure to stay involved with discussion lists, professional associations, books and journals available online or at the library that deal with application development.

## References

Apache. (n.d.) HTTP Server Project. What is the Apache Server Project Retrieved from:

[http://httpd.apache.org/ABOUT\\_APACHE.html](http://httpd.apache.org/ABOUT_APACHE.html)

Bernstein, M. (2002). 10 tips on writing the living Web. *A list apart: For people who make websites*, 149. Retrieved from <http://www.alistapart.com/articles/writeliving>

Bonaccorsi, A., & Rossi, C. (2003). Why open source software can succeed. *Research Policy*, 32(7), 1243-1258. Retrieved October 24, 2011, from ABI/INFORM Global. (Document ID: 405243571).

Crovitz, G. (2012, July 30). Information Age: WeHelpedBuildThat.com. *Wall Street Journal* (Eastern Edition), p. A.11. Retrieved August 15, 2012, from ABI/INFORM Global. (Document ID: 2723545181).

Feldman, D. A. (2006, April). Understanding open source: Part 2. *KM World*, 15(4), 20,22,24. Retrieved October 24, 2011, from ABI/INFORM Global. (Document ID: 1021576011).

Gittens, C. (2007, March). Open source is still fighting against fear. *Computing Canada*, 33(4), 25. Retrieved August 15, 2012, from ABI/INFORM Global. (Document ID: 1252213981).

Henschen, D. (2011). Why all the HADOOP? *InformationWeek*, (1316), 19-20,22-24,26. Retrieved from <http://search.proquest.com/docview/904423979?accountid=10825>

Kevin, F. C. (2002). PHP: An open source solution for web programming and dynamic content. *Information Technology and Libraries*, 21(3), 116-120. Retrieved from

<http://search.proquest.com/docview/215832480?accountid=10825>

Krill, P. (2012). Microsoft upgrades ASP.net MVC to enable faster-loading web apps.

*InfoWorld.Com*, Retrieved from

<http://search.proquest.com/docview/923642873?accountid=10825>

McKenzie, A. (2011). INWG and the Conception of the Internet: An Eyewitness

Account. *IEEE Annals of the History of Computing*, 33(1), 66-71. Retrieved August 15, 2012, from ABI/INFORM Global. (Document ID: 2286554181).

MySQL. (n.d.) Market Share. Retrieved from: [http://www.mysql.com/why-](http://www.mysql.com/why-mysql/marketshare/)

[mysql/marketshare/](http://www.mysql.com/why-mysql/marketshare/)

Open platforms will win cloud race: Citrix. (2012, Nov 19). *The Business Times*.

Retrieved from <http://search.proquest.com/docview/1170523431?accountid=10825>

Open Source. (n.d.) The Open Source Definition. Open Source Initiative. Retrieved from

<http://www.opensource.org/docs/osd>

Red hat expands OpenShift ecosystem with Zend partnership to offer professional-grade environment for PHP developers. (2012, Oct 09). *Business Wire*. Retrieved from

<http://search.proquest.com/docview/1095224091?accountid=10825>

Shaw, M. (2012). The role of design spaces. *IEEE Software*, 29(1), 46-50. doi:

<http://dx.doi.org/10.1109/MS.2011.121>

Sourceforge.net.(2012, September). Retrieved from: <http://sourceforge.net/about>

Wayner, P. (2012). Review: 2 PHP tools rise above the rest. *InfoWorld.Com*, Retrieved from <http://search.proquest.com/docview/931169920?accountid=10825>

Welling, L. & Thomson L. (2009). PHP and MySQL web development. The Fourth edition. Developer's Library.

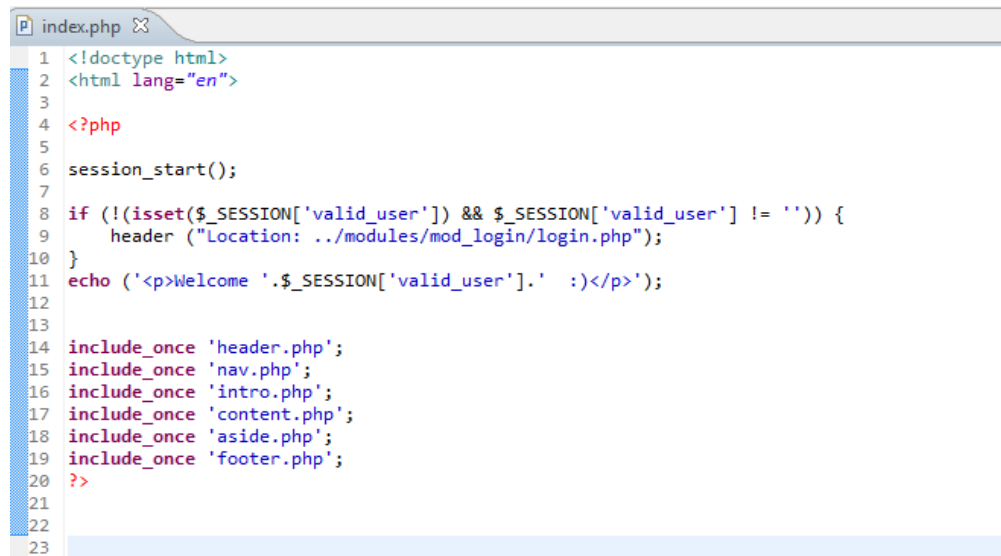
Zeev, S. (2002, June). The next HTML. *Computer Technology Review*, 22(6), 38.

Retrieved August 15, 2012, from ABI/INFORM Global. (Document ID: 140669711).

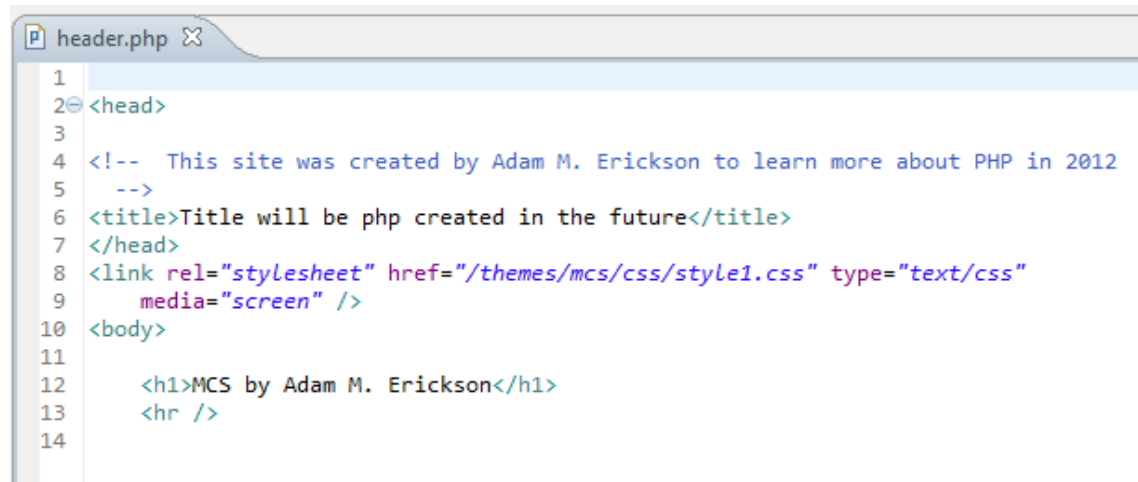


## Appendices

## Appendix A: Additional PHP Code Examples

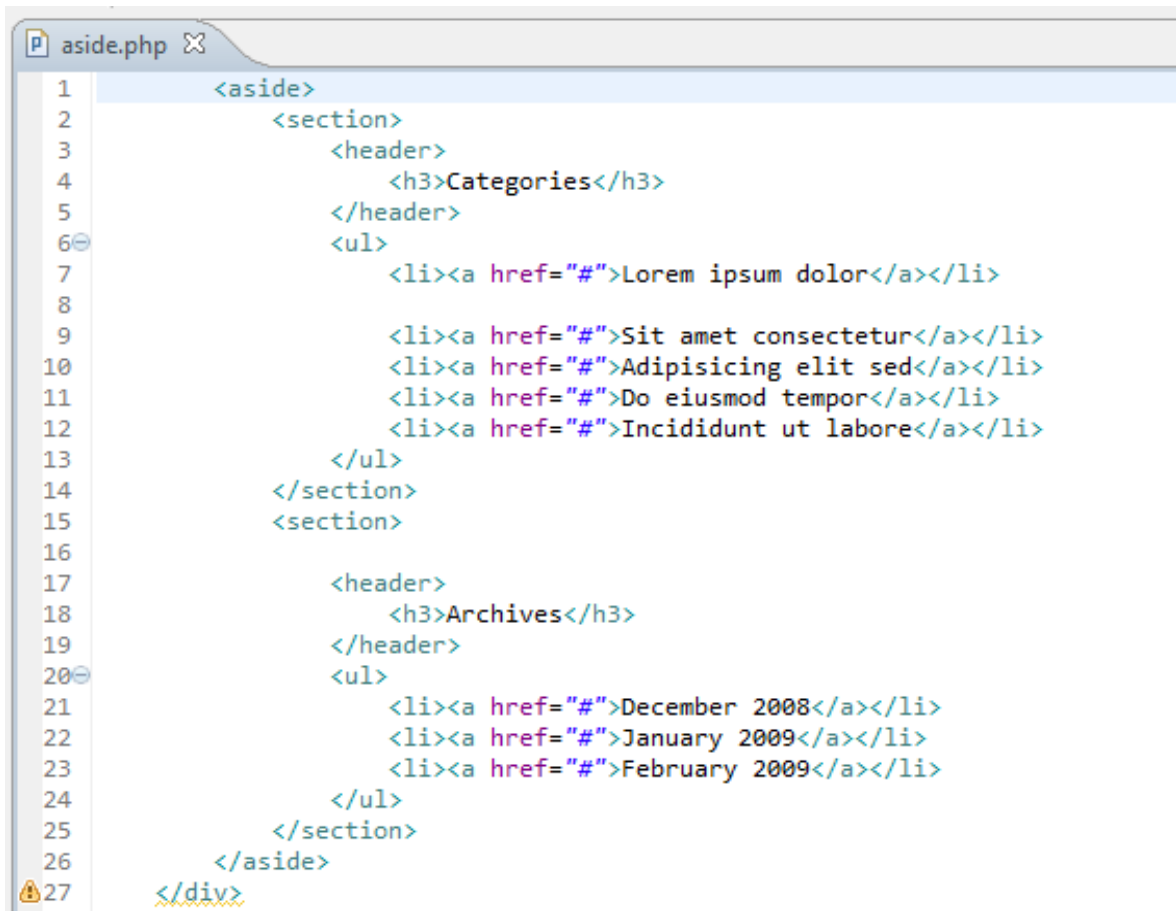
A screenshot of a code editor showing the contents of index.php. The code is a PHP script that starts with an HTML doctype and lang attribute. It then starts a session and checks if a session variable 'valid\_user' is set. If not, it redirects to a login page. If set, it echoes a welcome message. Finally, it includes several other PHP files: header.php, nav.php, intro.php, content.php, aside.php, and footer.php.

```
1 <!doctype html>
2 <html lang="en">
3
4 <?php
5
6 session_start();
7
8 if (!isset($_SESSION['valid_user']) && $_SESSION['valid_user'] != '') {
9     header ("Location: ../modules/mod_login/login.php");
10 }
11 echo ('<p>Welcome ' . $_SESSION['valid_user'] . ' :</p>');
12
13
14 include_once 'header.php';
15 include_once 'nav.php';
16 include_once 'intro.php';
17 include_once 'content.php';
18 include_once 'aside.php';
19 include_once 'footer.php';
20 ?>
21
22
23
```

*Figure 15.* Index.php.A screenshot of a code editor showing the contents of header.php. The code is an HTML header section. It includes a comment about the site's creation, a title tag, a link to a stylesheet, and a body tag containing an h1 heading and a horizontal line.

```
1
2 <head>
3
4 <!-- This site was created by Adam M. Erickson to learn more about PHP in 2012
5 -->
6 <title>Title will be php created in the future</title>
7 </head>
8 <link rel="stylesheet" href="/themes/mcs/css/style1.css" type="text/css"
9       media="screen" />
10 <body>
11
12     <h1>MCS by Adam M. Erickson</h1>
13     <hr />
14
```

*Figure 16.* Header.php.



```
1 <aside>
2   <section>
3     <header>
4       <h3>Categories</h3>
5     </header>
6     <ul>
7       <li><a href="#">Lorem ipsum dolor</a></li>
8
9       <li><a href="#">Sit amet consectetur</a></li>
10      <li><a href="#">Adipisicing elit sed</a></li>
11      <li><a href="#">Do eiusmod tempor</a></li>
12      <li><a href="#">Incididunt ut labore</a></li>
13    </ul>
14  </section>
15  <section>
16
17    <header>
18      <h3>Archives</h3>
19    </header>
20    <ul>
21      <li><a href="#">December 2008</a></li>
22      <li><a href="#">January 2009</a></li>
23      <li><a href="#">February 2009</a></li>
24    </ul>
25  </section>
26 </aside>
27 </div>
```

Figure 17. Aside.php.

```

content.php
1 <div id="content">
2   <div id="mainContent">
3     <section>
4       <article class="blogPost">
5         <header>
6           <h2>This is the title of a blog post</h2>
7           <p>Posted on <time datetime="2012-11-29T23:31+01:00">November 4th
8
9         </header>
10        <div>
11          <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin
12
13          
14
15          <p>Pellentesque ut sapien arcu, egestas mollis massa. Fusce lectus
16
17          <p>Vivamus vitae nulla dolor. Suspendisse eu lacinia justo. Vestib
18
19        </div>
20      </article>
21    </section>
22    <section id="comments">
23      <h3>Comments</h3>
24      <article>
25        <header>
26          <a href="#">not a real article</a> on <time datetime="2012-10-29T2
27
28        </header>
29        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do ei
30      </article>
31      <article>
32        <header>
33          <a href="#">FAke Article</a> on <time datetime="2012-10-29T23:40:0
34
35        </header>
36        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do ei
37      </article>
38      <article>
39        <header>
40          <a href="#">placeholder</a> on <time datetime="2012-06-29T23:59:00
41
42        </header>
43        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do ei
44      </article>
45    </section>
46    <form action="#" method="post">
47      <h3>Post a comment</h3>

```

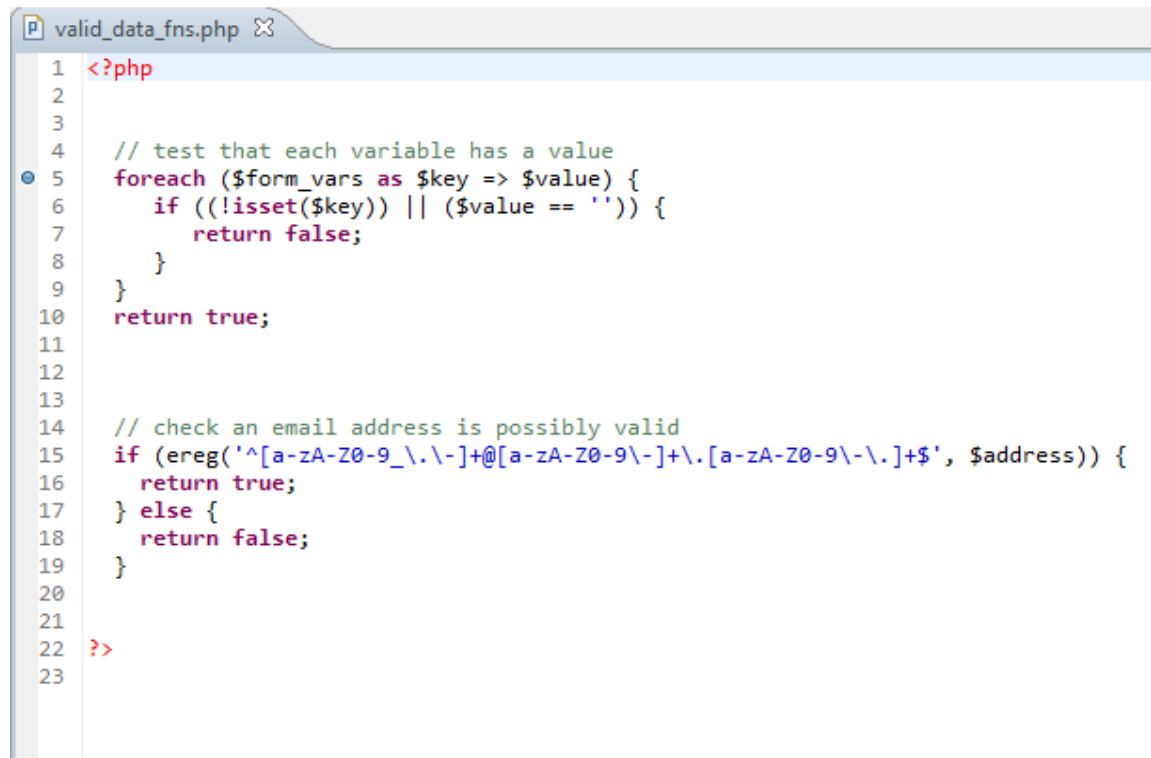
Figure 18. Content.php.

```

1 <footer>
2     <div>
3         <section id="about">
4             <header>
5                 <h3>About</h3>
6             </header>
7             <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
8
9         </section>
10        <section id="blogroll">
11            <header>
12                <h3>Blogroll</h3>
13            </header>
14            <ul>
15                <li><a href="#">NETTUTS</a></li>
16                <li><a href="#">FreelanceSwitch</a></li>
17
18                <li><a href="#">In The Woods</a></li>
19                <li><a href="#">Netsetter</a></li>
20                <li><a href="#">PSDTUTS</a></li>
21            </ul>
22        </section>
23        <section id="popular">
24            <header>
25
26                <h3>Popular</h3>
27            </header>
28            <ul>
29                <li><a href="#">This is the title of a blog post</a></li>
30                <li><a href="#">Lorem ipsum dolor sit amet</a></li>
31                <li><a href="#">Consectetur adipisicing elit, sed do eiusmod</a></li>
32                <li><a href="#">Duis aute irure dolor</a></li>
33
34                <li><a href="#">Excepteur sint occaecat cupidatat</a></li>
35                <li><a href="#">Reprehenderit in voluptate velit</a></li>
36                <li><a href="#">Officia deserunt mollit anim id est laborum</a></li>
37                <li><a href="#">Lorem ipsum dolor sit amet</a></li>
38            </ul>
39        </section>
40    </div>
41
42 </footer>
43
44 </body>
45 </html>

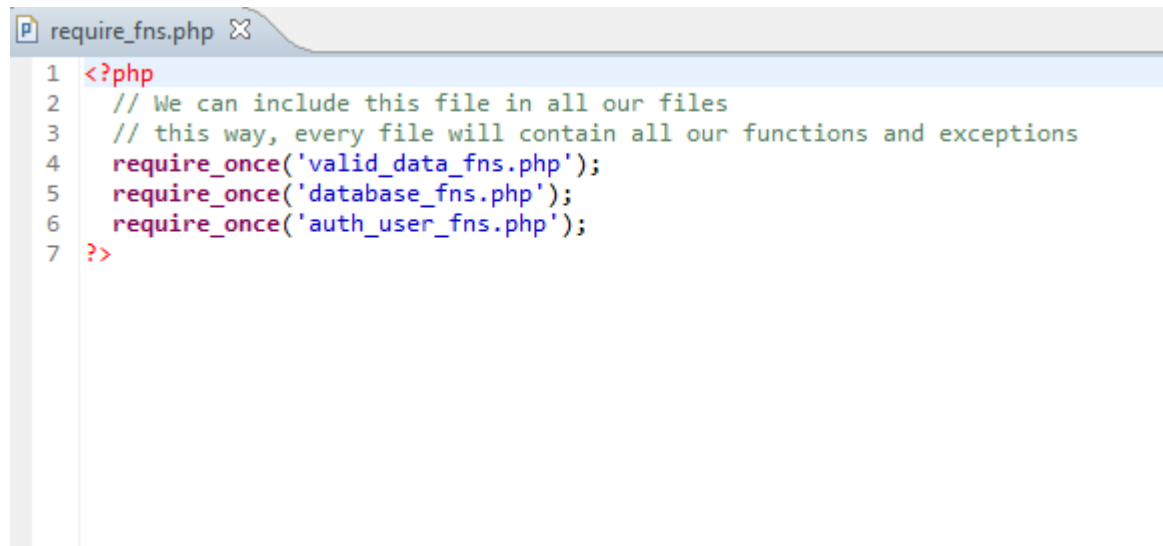
```

Figure 19. Footer.php.



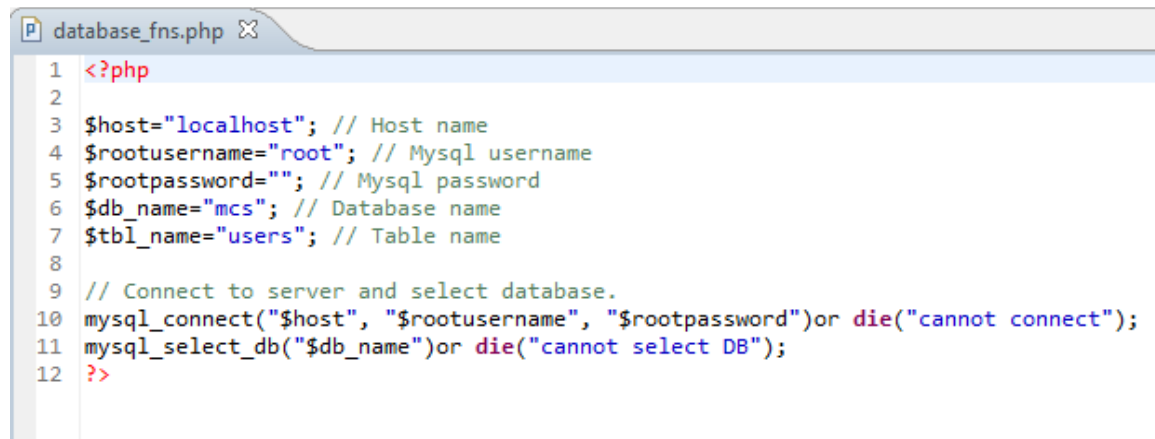
```
1 <?php
2
3
4 // test that each variable has a value
5 foreach ($form_vars as $key => $value) {
6     if ((!isset($key)) || ($value == '')) {
7         return false;
8     }
9 }
10 return true;
11
12
13
14 // check an email address is possibly valid
15 if (ereg('^[a-zA-Z0-9_\.\\-]+@[a-zA-Z0-9\\-]+\\.[a-zA-Z0-9_\\.\\-]+$', $address)) {
16     return true;
17 } else {
18     return false;
19 }
20
21
22 ?>
23
```

Figure 20. Valid\_data\_fns.php.



```
1 <?php
2 // We can include this file in all our files
3 // this way, every file will contain all our functions and exceptions
4 require_once('valid_data_fns.php');
5 require_once('database_fns.php');
6 require_once('auth_user_fns.php');
7 ?>
```

Figure 21. Require\_fns.php.

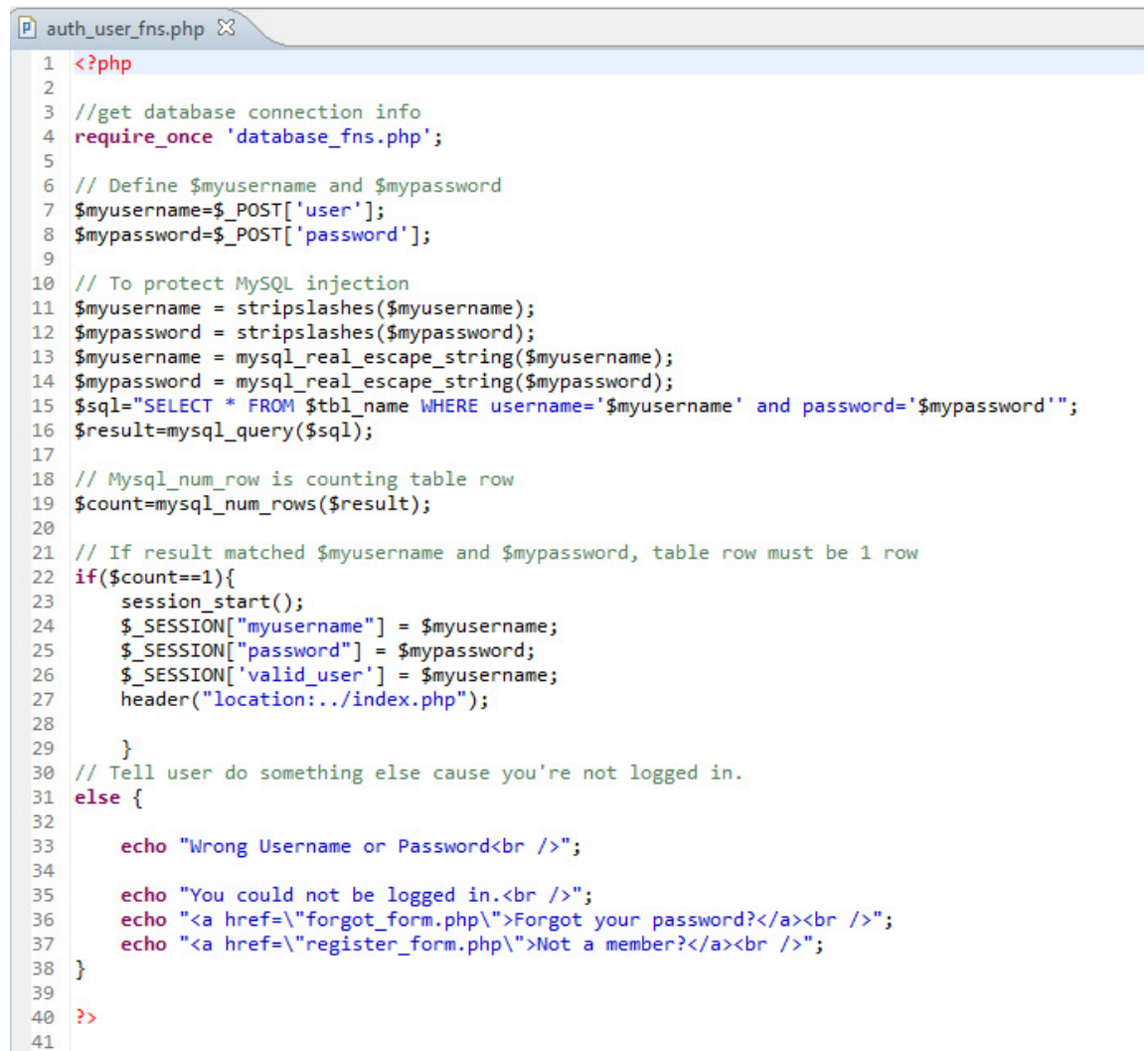


```

1  <?php
2
3  $host="localhost"; // Host name
4  $rootusername="root"; // Mysql username
5  $rootpassword=""; // Mysql password
6  $db_name="mcs"; // Database name
7  $tbl_name="users"; // Table name
8
9  // Connect to server and select database.
10 mysql_connect("$host", "$rootusername", "$rootpassword")or die("cannot connect");
11 mysql_select_db("$db_name")or die("cannot select DB");
12 ?>

```

Figure 22. Database\_fns.php.



```

1  <?php
2
3  //get database connection info
4  require_once 'database_fns.php';
5
6  // Define $myusername and $mypassword
7  $myusername=$_POST['user'];
8  $mypassword=$_POST['password'];
9
10 // To protect MySQL injection
11 $myusername = stripslashes($myusername);
12 $mypassword = stripslashes($mypassword);
13 $myusername = mysql_real_escape_string($myusername);
14 $mypassword = mysql_real_escape_string($mypassword);
15 $sql="SELECT * FROM $tbl_name WHERE username='$myusername' and password='$mypassword'";
16 $result=mysql_query($sql);
17
18 // Mysql_num_row is counting table row
19 $count=mysql_num_rows($result);
20
21 // If result matched $myusername and $mypassword, table row must be 1 row
22 if($count==1){
23     session_start();
24     $_SESSION["myusername"] = $myusername;
25     $_SESSION["password"] = $mypassword;
26     $_SESSION['valid_user'] = $myusername;
27     header("location:../index.php");
28 }
29
30 // Tell user do something else cause you're not logged in.
31 else {
32
33     echo "Wrong Username or Password<br />";
34
35     echo "You could not be logged in.<br />";
36     echo "<a href=\"forgot_form.php\">Forgot your password?</a><br />";
37     echo "<a href=\"register_form.php\">Not a member?</a><br />";
38 }
39
40 ?>
41

```

Figure 23. Auth\_user\_fns.php.

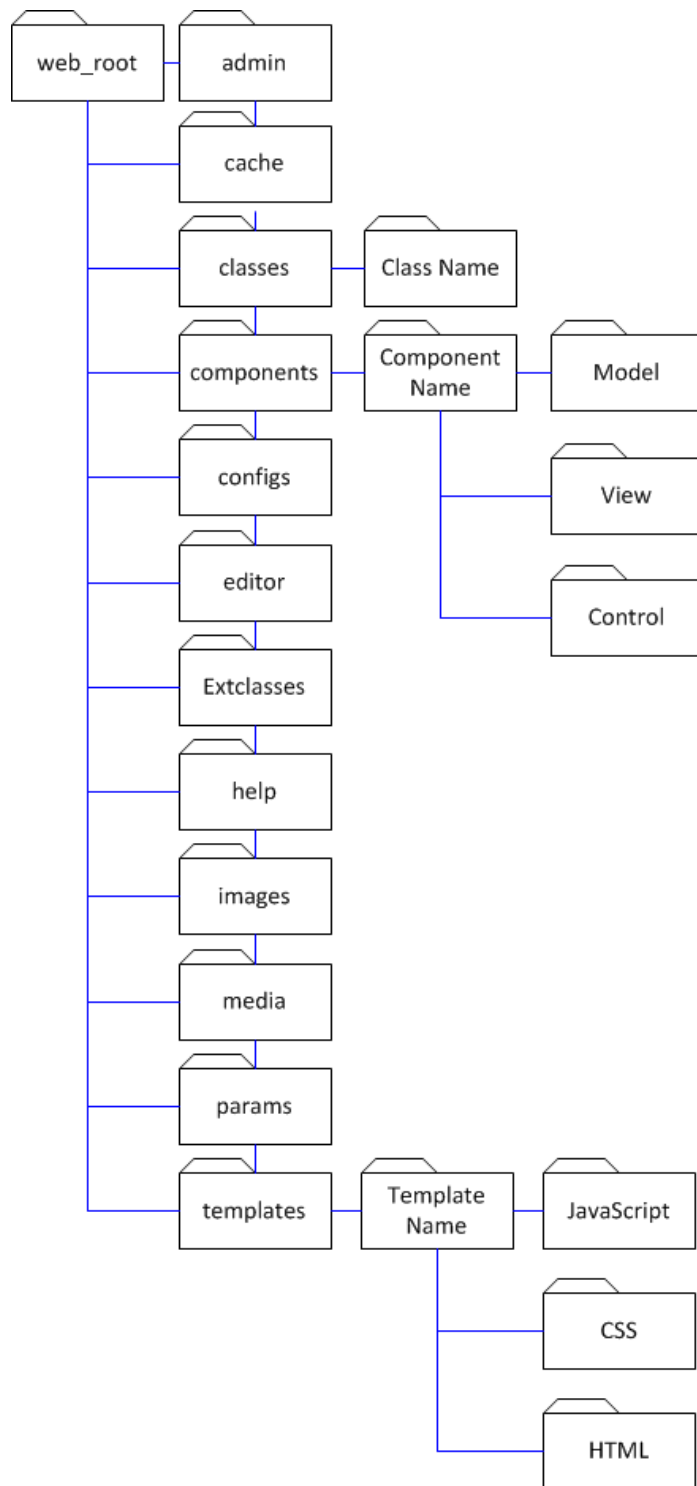
**Appendix B: CMS Directory Hierarchy**

Figure 24. Directory Hierarchy.

## Appendix C: MySQL Create Database and User Information Tables Script

Figures in appendix C are for creating the database and adding the user fields only.

```

1  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
2  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
3  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
4
5  DROP SCHEMA IF EXISTS `mydb` ;
6  CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci ;
7  USE `mydb` ;
8
9  -----
10 -- Table `mydb`.`user_usergroup_map`
11 -----
12 DROP TABLE IF EXISTS `mydb`.`user_usergroup_map` ;
13
14 CREATE TABLE IF NOT EXISTS `mydb`.`user_usergroup_map` (
15   `user_id` INT UNSIGNED NOT NULL DEFAULT '0' COMMENT 'Foreign Key to #__users.id' ,
16   `group_id` INT UNSIGNED NOT NULL DEFAULT '0' COMMENT 'Foreign Key to #__usergroups.id' ,
17   PRIMARY KEY (`user_id`, `group_id`) )
18   DEFAULT CHARACTER SET = utf8;
19
20 -----
21 -- Table `mydb`.`usergroups`
22 -----
23
24 DROP TABLE IF EXISTS `mydb`.`usergroups` ;
25
26 CREATE TABLE IF NOT EXISTS `mydb`.`usergroups` (
27   `group_id` INT UNSIGNED NOT NULL DEFAULT '0' COMMENT 'Adjacency List Reference Id' ,
28   `title` VARCHAR(100) NOT NULL DEFAULT '' ,
29   `description` VARCHAR(300) NULL ,
30   UNIQUE INDEX `idx_usergroup_parent_title_lookup` (`group_id` ASC, `title` ASC) ,
31   INDEX `idx_usergroup_title_lookup` (`title` ASC) ,
32   INDEX `idx_usergroup_adjacency_lookup` (`group_id` ASC) ,
33   PRIMARY KEY (`group_id`) ,
34   CONSTRAINT `fk_#__usergroups_#__user_usergroup_map1`
35     FOREIGN KEY (`group_id`)
36     REFERENCES `mydb`.`user_usergroup_map` (`group_id` )
37     ON DELETE NO ACTION
38     ON UPDATE NO ACTION)
39   DEFAULT CHARACTER SET = utf8;
40
41

```

Figure 25. Create Database Script.



```
42 -----
43 -- Table `mydb`.`user_notes`
44 -----
45 DROP TABLE IF EXISTS `mydb`.`user_notes` ;
46
47 CREATE TABLE IF NOT EXISTS `mydb`.`user_notes` (
48   `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT ,
49   `user_id` INT(10) UNSIGNED NOT NULL DEFAULT '0' ,
50   `subject` VARCHAR(100) NOT NULL DEFAULT '' ,
51   `body` TEXT NOT NULL ,
52   `created_user_id` INT(10) UNSIGNED NOT NULL DEFAULT '0' ,
53   `created_time` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00' ,
54   `modified_user_id` INT(10) UNSIGNED NOT NULL ,
55   `modified_time` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00' ,
56   `review_time` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00' ,
57   `publish_up` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00' ,
58   PRIMARY KEY (`id`) ,
59   INDEX `idx_user_id` (`user_id` ASC) )
60 DEFAULT CHARACTER SET = utf8;
61
```

```

62
63 -----
64 -- Table `mydb`.`users`
65 -----
66 DROP TABLE IF EXISTS `mydb`.`users` ;
67
68 CREATE TABLE IF NOT EXISTS `mydb`.`users` (
69   `id` INT NOT NULL AUTO_INCREMENT ,
70   `name` VARCHAR(255) NOT NULL DEFAULT '' ,
71   `username` VARCHAR(150) NOT NULL DEFAULT '' ,
72   `email` VARCHAR(100) NOT NULL DEFAULT '' ,
73   `password` VARCHAR(100) NOT NULL DEFAULT '' ,
74   `usertype` VARCHAR(25) NOT NULL DEFAULT '' ,
75   `block` TINYINT(4) NOT NULL DEFAULT '0' ,
76   `sendEmail` TINYINT(4) NULL DEFAULT '0' ,
77   `registerDate` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00' ,
78   `lastvisitDate` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00' ,
79   `activation` VARCHAR(100) NOT NULL DEFAULT '' ,
80   INDEX `usertype` (`usertype` ASC) ,
81   INDEX `idx_name` (`name` ASC) ,
82   INDEX `idx_block` (`block` ASC) ,
83   INDEX `username` (`username` ASC) ,
84   INDEX `email` (`email` ASC) ,
85   PRIMARY KEY (`id`) ,
86   CONSTRAINT `fk_#__users_#__user_usergroup_map1`
87     FOREIGN KEY (`id` )
88     REFERENCES `mydb`.`user_usergroup_map` (`user_id` )
89     ON DELETE NO ACTION
90     ON UPDATE NO ACTION,
91   CONSTRAINT `fk_#__users_#__user_notes1`
92     FOREIGN KEY (`id` )
93     REFERENCES `mydb`.`user_notes` (`user_id` )
94     ON DELETE NO ACTION
95     ON UPDATE NO ACTION)
96 DEFAULT CHARACTER SET = utf8;
97

```

```

98
99
100 SET SQL_MODE=@OLD_SQL_MODE;
101 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
102 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
103

```

## Appendix D: MySQL Entity Relational Diagram

The below figure is a visual representation of appendix C from a relational context.

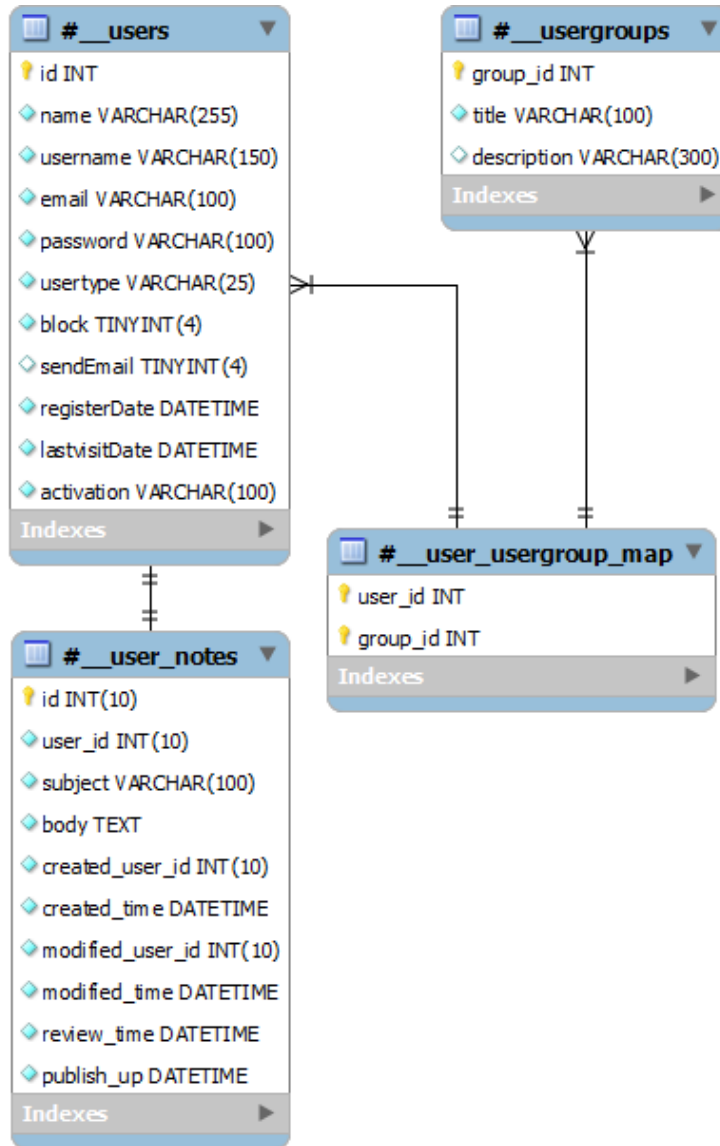


Figure 26. User Database ERD Diagram

## Appendix F: CSS Script

```
1  /*-----  
2  =BASIC SETUP  
3  -----*/  
4  
5  /* Makeshift CSS Reset */  
6  * {  
7      margin: 0;  
8      padding: 0;  
9  }  
10  
11  /* Tell the browser to render HTML 5 elements as block */  
12  header, footer, section, aside, nav, article {  
13      display: block;  
14  }  
15  
16  body {  
17      margin: 0 auto;  
18      padding: 22px 0;  
19      width: 940px;  
20      font: 13px/22px Helvetica, Arial, sans-serif;  
21      background: #F0F0F0;  
22  }  
23  
24  h1, h2 {  
25      font-size: 28px;  
26      line-height: 44px;  
27      padding: 22px 0;  
28  }  
29  
30  h3 {  
31      font-size: 18px;  
32      line-height: 22px;  
33      padding: 11px 0;  
34  }  
35  
36  p {  
37      padding-bottom: 22px;  
38  }  
39  
40
```

Figure 27: CSS Script

```
42  /*-----  
43  =NAVIGATION  
44  -----*/  
45  
46  nav {  
47      position: absolute;  
48      left: 0;  
49      width: 100%;  
50      background: url("images/nav_background.png");  
51  }  
52  
53  nav ul {  
54      margin: 0 auto;  
55      width: 940px;  
56      list-style: none;  
57  }  
58  
59  nav ul li {  
60      float: left;  
61  }  
62  
63  nav ul li a {  
64      display: block;  
65      margin-right: 20px;  
66      width: 140px;  
67      font-size: 14px;  
68      line-height: 44px;  
69      text-align: center;  
70      text-decoration: none;  
71      color: #777;  
72  }  
73  
74  nav ul li a:hover {  
75      color: #fff;  
76  }  
77  
78  nav ul li.selected a {  
79      color: #fff;  
80  }  
81  
82  nav ul li.subscribe a {
```

```

83         margin-left: 22px;
84         padding-left: 33px;
85         text-align: left;
86         background: url("images/rss.png") left center
87     }
88
89
90
91     /*-----
92     =INTRODUCTION
93     -----*/
94
95     #intro {
96         position: relative;
97         margin-top: 66px;
98         padding: 44px;
99         background: #467612 url("images/intro_background.png") re
100
101         /* Background-size not implemented yet */
102         -webkit-background-size: 100%;
103         -o-background-size: 100%;
104         -khtml-background-size: 100%;
105
106
107         /* Border-radius not implemented yet */
108         -moz-border-radius: 22px;
109         -webkit-border-radius: 22px;
110     }
111
112     #intro h2, #intro p {
113         position: relative;
114         z-index: 9999;
115         width: 336px;
116     }
117
118     #intro h2 {
119         padding: 0 0 22px 0;
120         font-weight: normal;
121         color: #fff;
122     }

```

```

122     }
123
124     #intro p {
125         padding: 0;
126         color: #d9f499;
127     }
128
129     #intro img {
130         position: absolute;
131         top: 0;
132         right: 0;
133         width: 653px;
134         height: 100%;
135
136         /* Border-radius not implemented yet */
137         -moz-border-radius: 22px;
138         -webkit-border-radius: 22px;
139     }
140
141     /*-----
142     =CONTENT AREA AND SIDEBAR LAYOUT
143     -----*/
144
145     #content {
146         display: table;
147     }
148
149     #mainContent {
150         display: table-cell;
151         width: 620px;
152         padding-right: 22px;
153     }
154
155     aside {
156         display: table-cell;
157         width: 300px;
158         background: #999999 url("") top right;
159
160         /* Border-radius not implemented yet */
161         -moz-border-radius: 22px;
162         -webkit-border-radius: 22px;

```

```

163     }
164
165
166
167  /*-----
168  =BLOG POST
169  -----*/
170  .blogPost header p, .blogPost header p a {
171      font-size: 14px;
172      font-style: italic;
173      color: #777;
174  }
175
176      .blogPost header p a:hover {
177          text-decoration: none;
178          color: #000;
179      }
180
181  .blogPost div {
182      /* Column-count not implemented yet */
183      -moz-column-count: 2;
184      -webkit-column-count: 2;
185
186      /* Column-gap not implemented yet */
187      -moz-column-gap: 22px;
188      -webkit-column-gap: 22px;
189  }
190
191  .blogPost img {
192      margin: 22px 0;
193      -webkit-box-shadow: 3px 3px 7px #777;
194  }
195
196
197
198  /*-----
199  =COMMENTS
200  -----*/
201
202  #comments {
203      margin-top: 21px;

```



```
203     margin-top: 21px;
204     padding-top: 22px;
205     border-top: 1px solid #d7d7d7;
206 }
207
208 #comments article {
209     display: table;
210     padding: 22px;
211 }
212
213 #comments article:nth-child(odd) {
214     padding: 21px;
215     background: #E3E3E3;
216     border: 1px solid #d7d7d7;
217     -moz-border-radius: 11px;
218     -webkit-border-radius: 11px;
219 }
220
221 #comments article header {
222     display: table-cell;
223     width: 220px;
224     padding-right: 22px;
225 }
226
227     #comments article header a {
228         display: block;
229         font-weight: bold;
230         color: #000;
231     }
232
233     #comments article header a:hover {
234         text-decoration: none;
235     }
236
237 #comments article p {
238     padding: 0;
239 }
240
241
242
243
```

```
244  =COMMENT FORM
245  -----*/
246  form {
247      margin-top: 21px;
248      padding-top: 22px;
249      border-top: 1px solid #d7d7d7;
250  }
251
252  form p {
253      display: table;
254      margin-bottom: 22px;
255      padding: 0 22px;
256  }
257
258  form label {
259      display: table-cell;
260      width: 140px;
261      padding-right: 20px;
262      text-align: right;
263      font-weight: bold;
264      vertical-align: top;
265  }
266
267  form input[type="text"], form input[type="email"], fo
268      display: table-cell;
269      width: 300px;
270      height: 20px;
271      border: 1px solid #d7d7d7;
272  }
273
274  form textarea {
275      width: 300px;
276      height: 100px;
277      border: 1px solid #d7d7d7;
278  }
279
280  form input[type="submit"] {
281      margin-left: 162px;
282  }
283
```

```
286  /*-----
287  =SIDEBAR
288  -----*/
289  aside section {
290      margin: 22px 0 0 22px;
291      padding: 11px 22px;
292      background: url("images/sidebar_section_background.png")
293
294      /* Border-radius not implemented yet */
295      -moz-border-radius: 11px;
296      -webkit-border-radius: 11px;
297  }
298
299  aside section ul {
300      margin: 0 0 0 22px;
301      list-style: none;
302  }
303
304  aside section ul li a {
305      display: block;
306      text-decoration: none;
307      color: #000;
308  }
309
310  aside section ul li a:hover {
311      text-decoration: underline;
312  }
313
314
315
316  /*-----
317  =FOOTER
318  -----*/
319  footer {
320      position: absolute;
321      left: 0;
322      width: 100%;
323      background: #222;
324  }
325
326  footer div {
```

```
326     footer div {
327         display: table;
328         margin: 0 auto;
329         padding: 44px 0;
330         width: 940px;
331         color: #777;
332     }
333
334     footer div section {
335         display: table-cell;
336         width: 300px;
337     }
338
339     footer div #about, footer div #blogroll {
340         padding-right: 20px;
341     }
342
343     footer h3 {
344         color: #FFF;
345     }
346
347     footer a {
348         color: #999;
349     }
350
351     footer a:hover {
352         color: #FFF;
353         text-decoration: none;
354     }
355
356     footer ul {
357         margin: 0 0 0 40px;
358         list-style: square;
359         color: #565656;
360     }
361
362     footer ul li a {
363         display: block;
364     }
```