# Contents

# 1. Abstract …………………………………………………………….2

# 2. Introduction………………………………………………………...3

# 3. Background………………………………………………………...4

# 4. Design……………………………………………………………….6

# 3. Implementation…………………………………………………….11

# 4. System in Action……………………………………………………22

# 5. Future Work………………………………………………………..28

# 6. Conclusion……………………………………………………….....28

# Abstract

The main goal of this project is to build a simple and easy to use CMS .

A content management system (CMS) is a software package specifically designed to manage a website.

It provides you with a simple, non-technical way of updating your content. This is typically done via a web-based interface.

Just point-and-click, type in the new words, and hit save. Your site is instantly created and updated.

Equally easy is adding new pages, deleting old ones, or restructuring the site to match your new business model.

Along with the many other administrative tools, this leaves you to concentrate on the words, and not on the technology.

**Chapter1:**

**Introduction**

Content management systems are one of our most popular requests these days.  Many people want one so that they can update their site on their own. This CMS will allow any user to add content to a website without any knowledge of the technologies used to let it work.

This report details what we have done the last four months building a CMS. Chapter 2 starts with explaining what is CMS? What is the point of using CMS? Why did we choose such a project? And what are the tools used to achieve that. In chapter 3 we will explain steps before implementing the structure of CMS, which includes searching, deciding how to build the main structure of our CMS, do we have to build our own or to use a Framework or to code with PHP without a certain structure. To answer these questions we have to study PHP Frameworks, MVC concept, and CakePHP as a chosen Framework, and as a final choice building our own MVC.

Chapter 4 discusses our own implementation including how does our own MVC work? Directory structure, setting up the router PHP file, The CMS's private and public areas and different blocks which include Weather, News , Articles, Image Gallary,MP3 player…

In chapter 5 which has the name “System in action” gives you a user guide of how using our CMS, including front-end and back-end.

Lastly, future work includes things that can be done in future to improve our CMS. The conclusion summarizes what we learnt from this project, what skills we improved and achieved.

# Chapter2

**Background**

# 2.1 What is CMS?

A content management system (CMS) is a system used to manage the content of a website. It is a web application designed to make it easy for non-technical users to add, edit and manage a website without needing to know any HTML or Web design knowledge.

### 2.2 What are the benefits of a CMS?

Because of the user friendly interface, our content management system makes life much easier in many ways:

* You are no longer dependent on the web designers making changes for you.
* Increase flexibility of your site.
* Changes can be made any time you need, day or night.
* All the technical details are simply handled by the CMS, allowing anyone to manage and update the site.
* The CMS ensures that all the pages are consistent in design, and will build all the menus and other navigation for you; just you have to pick up a template.

**2.3 Why did we choose CMS project?**

Firstly,we thought about something helps us to master the PHP language since we had little knowledge of it, so we found our graduation project as a perfect chance to do that , by choosing a web application to implement.

Secondly, we saw a need to build a CMS that really can be used by a non-technical people and because of this we focused on the interface to be so easy.

**2.4 Languages and Tools:**

To match perfectly what we've done, we recommend using windows7 or windows xp.

Also we are using:

* Server -> WampServer2.0c
* Database -> MySQL
* Dreamweaver
* Programming language -> PHP, JavaScript, and HTML.

# Chapter3

**Design**

**3.1 Searching**

As a first step in achieving the aim of building our own CMS, we started to look around the internet for web applications that do what we want to do. As a result we found that the most three popular CMSs (complex CMSs)are Drupal, Joomla and Wordpress.

When we first came across Drupal (also Joomla)!-to get the whole idea of how does it work- we found it easy to install, but a little bit confusing and difficult to learn the administration area. Even more difficult to learn was how to customize the site as per our requirements. So we had to use tutorials to deal with that.

Also, we had a closer look to simpler CMSs to get familiar with most of its features to make something similar but in a simple way.

**3.2 Deciding how to build the main structure of our CMS?**

At this point we have one of three choices either using PHP hand coding (without MVC) or building our own MVC or using a framework that implements MVC. To make the right decision we have to make a closer look at MVC concept.

There are lots of technologies and tools available for building dynamic web applications. These include ASP and .Net from Microsoft, J2EE from Sun Microsystems, PHP – the list goes on. Regardless of which of these technologies/frameworks one selects, there are certain common patterns that one finds show up over and over again. One of the most talked about is the MVC or Model-View-Controller.

**3.2.1 MVC Concept:**

Model–View–Controller (MVC) is an architectural pattern used in software engineering. Successful use of the pattern isolates business logic from user interface considerations, resulting in an application where it is easier to modify either the visual appearance of the application or the underlying business rules without affecting the other. In MVC, the model represents the information (the data) of the application; the view corresponds to elements of the user interface such as text, checkbox items, and so forth; and the controller manages the communication of data and the business rules used to manipulate the data to and from the model.

**Model -** This component contains the application data which is represented by the view. Those data which is part of persistent state must reside in model object. When the state of data changes it notify the view. The controller access the model object data for effecting their state change. It represents and maintains the application state.

**View -** The view represents the state of the model to the user. It is actually represents the application data to the user and also takes the data from the user and send to the controller. There is no any business logic, flow logic inside the view it contains tags. The view renders the model data for presenting to the user. An application can contain many views.

**Controller -** The controller is responsible for controlling entire application. It accepts the input coming from the view, translates into the action to be performed by the view. The controller is only responsible for accessing model and rendering it to the various user interfaces. Simply you can say a controller accepts the data from the client, performs the business operation, and return it to the client. The flow of data in the application is controlled by the controller. It is responsible for forwarding the request to the appropriate handler.

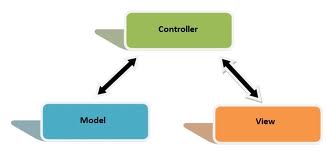


Figure 1

In simpler words:

1. Model handles all our database logic. Using the model we connect to our database and provide an abstraction layer.

2. Controller represents all our business logic i.e. all our ifs and else.

3. View represents our presentation logic i.e. our HTML code.

## 3.2.2 PHP Frameworks

The idea behind a framework is to offer a design you can use across multiple applications. All applications have a number of basic things in common. A framework is designed to provide a structure for those common elements (database interaction, presentation layer, application logic) so you spend less time writing up database interface code or presentation-layer interfaces and more time writing the application itself. The architecture represented by breaking an application up in this fashion is referred to as Model-View-Controller (MVC) which breaks apart your domain logic from your user interface (as we mentioned above).

After conducting some research, and reading some relative feedback on which frameworks "give" the developer more in terms of usability, speed, and stability, we found the most popular frameworks:

1. CakePHP Framework
2. [Zend Framework](http://framework.zend.com/)
3. [Symphony](http://www.symfony-project.org/) Framework

Now according to what we have mentioned above, we ruled out the first choice, because we would definitely have a messy PHP code, and that’s really our greatest fear. The second choice also was ruled out, because building MVC from scratch is not an easy choice. As a result we decided to choose the third option which is using a framework that implements MVC concept to make things easier to us.

Now we have to choose a framework to suit our needs.

The situation was not easy at all, because we did not find an exact answer to our question: “Which framework to use?” Do we have to choose Zend or CakePHP or Symphony?

The comparison limited between Zend and CakePHP according to our reading!

As a summary of what we had read, see below:

Use CakePHP framework if you want development to be fast and easy and do not have to think about implementation much as long as you follow convention. In other words use it if you are a beginner.   
  
 Use Zend framework if you want full control and want configuration over convention as well as strict OOP setup. Takes longer to implement, but can be more powerful. In other words use it if you are skilled enough.

### According to this, and to the fact that we are really a beginner programmers

, and according to our limited time (4 months) to deliver the project, **CakePHP was our choice as a framework**.

**3.2.3 Getting started with CakePHP Framework!**

CakePHP is a free open-source PHP development framework, based on an MVC-like architecture that is powerful. The CakePHP framework represents a foundational structure for programmers to create web applications. Using it, the developers are enabled to work in a structured manner, without losing flexibility.

The CakePHP framework represents a robust base for handling every aspect of a web application, from the user's initial request to the final rendering of a web page. Following the principles of MVC (model, view, controller), the framework allows you to easily customize and extend most aspects of your application. It also offers a basic organizational structure, from filenames to database table names, keeping your entire application consistent and logical.

We started learning Cakephp framework; we installed many tutorials and started implementing our CMS.

After one month of working, the CakePHP started to slow us down and we reached a point that we cannot continue any more. The reason for that was:Till that moment we did not have the whole picture of cake in our minds, also troubleshooting bugs was really a nightmare, in addition our CMS got a mess and the magic that Cake does, forces us to often fight against the framework’s built-in behavior.

**3.3 Summary:**

Unfortunately, luck was not on our side when we had chosen to program with CakePHP. So we had to step back and think again and again.

Now, there is no **choice but building our own MVC from scratch** even though this choice was not acceptable before a month , but at this moment it’s OK to spend more efforts coding, because we convinced that it’s faster than mastering CakePHP , and we will teach a lot about architecture , design and more.

**Chapter 4**

**Implementation**

**4.1 Building our own MVC**

After working for a month on the MVC under CakePHP, we really had the ability to start building our own MVC. Although it seems hard to build something from scratch but it was the only solution.

## 4.1.1 Getting Started:

The "core" of a CMS is its architecture. The CMS core is the "backbone" of the CMS. It's what connects the database, browser interactions, and blocks together.

A CMS consists of the management area (admin area), and the publicly visible area (front-end).

**4.1.2 Directory structure**

In our CMS, we will keep the admin area in a directory called admin.

The directory structure has the following folders :

* Admin -> Admin area.
* Images ->CMS images.
* Layout -> CSS styles, templates for the admin area.
* Site layout -> CSS styles, templates for the site we want to build .
* Includes -> Database configuration and main router.
* Models -> deals directly with database.
* Views -> provides a user interface that is separate from the logic.
* Controllers ->have actions to be executed.
* Languages -> Arabic or English Settings.
* Components -> contains the design of the site that the user want to create.

There are only few files kept in the web root. All others are kept in whichever directory makes the most sense for them.

The most important file in the web root is index.php this file is the front-end controller. All page requests are passed through this file.

Also there is an important file which is in the includes directory called router.php this file contains the rules that tell the web server how to parse HTTP requests, redirecting through index.php (or other controllers, as we'll see later).

The application hierarchy should now look something like this:

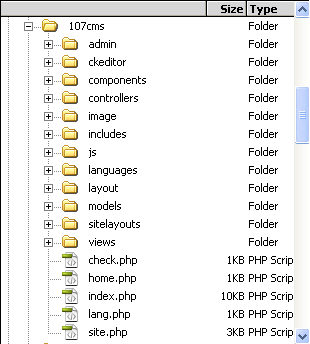


Figure 2

## 4.1.3 Setting up the Router

router.php is included in index.php. This file is the handler for all web page requests. It is like the wireless router in your house, it takes connections from the cable or satellite modem, and routes internet to every computer in the house. Router.php is going to take web page requests passed to index.php and route the request to different files (controllers) in our CMS.

This is going to grab the request string passed to the application. The request string is everything following the '?' in the URL. For example <http://localhost/index.php?page1>.

**4.2 Our Model-View-Controller**

MVC structures are useful because they allow you to separate the different processes of the web site. When needing to change or add new form fields, for instance, we need only to locate the appropriate view file and make the change. Instead of sifting through PHP output functions or scripts, we know that all the views are contained in the views folder. The same is true of controllers and models. Certain functions are available across the whole application (actions in controllers).

Our CMS can become difficult as the program grows, so the MVC architecture helps keep our application more agile.

The view is the presentation layer. It is the portion of our application that users will be most familiar with. Earlier we mentioned that the purpose of the view is to provide a user interface that is separate from the logic. There are many ways to go about this. We can use a template engine such as Smarty or something similar, or we can build our own, which what we have done. So, why do we need template engine?

Template engine is used to facilitate a manageable way to separate application logic and content from its presentation (HTML/CSS). In other words we want to isolate code from design in the view so that if we change design we do not have to go back and change code. By this we are separating the view layer into two layers.

The URL requests would always take this form “**Index.php?controller/action”**

To illustrate how things get done in our MVC -according to figure3:

1. The web server sends the request to our CMS in this form **Index.php?controller/action**.
2. Router.php tells the web server how to parse HTTP requests, redirecting through index.php. Breaking the request down to determine the target controller.
3. The controller according to the action sent, sends a find request to the model asking whether the submitted information matches anything in the database.
4. The model generates the query and runs it through the database.
5. Based on the response in step 4, the model returns either a true result or a false result to the controller.
6. The controller processes the result and fetches the appropriate view to be sent to the client.

6. The final output view is displayed to the client.

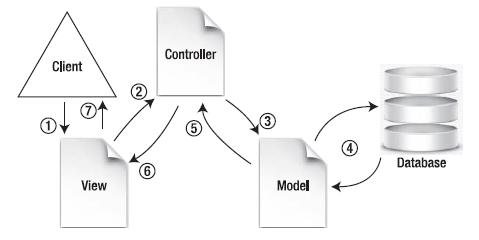


Figure 3

**4.3 The CMS's private and public areas**

Our CMS consists of the management area (admin area or back-end), and the publicly visible area (front-end).

**4.3.1 CMS Back-end:**

Simply, this is the admin area of our CMS, and this can be accessed from admin folder. From here you (as an admin) can edit the front-end, you can create, update front-end site contents. Just point-and-click, type in the new words, and hit save. Your site is instantly created and updated.



Figure 4

**4.3.2 CMS Front-end:**

This is a site generated from Managing the backend (admin area), you can have more than one site, and also you can easily change the site template.



Figure 5

**4.4 Features work:**

Our CMS supports Arabic, English Languages, it also uses AJAX where it needed, and also it is user friendly web application.

Dynamically installed templates-Admin can easily install new templates.

**4.5 Blocks**

Blocks are the real power behind how a CMS does its thing; they give CMS extra capabilities. So, the way we handle differences between sites is by using a core, and extending this with blocks.

There are a number of types of blocks that a site can use.

An example of how this might be changed with a block is if you have an “image gallery" block, where you choose images in the admin area, and those images are displayed nicely on the front-end.

In this case, the admin area should look very different from the front end.

How this case is handled in the admin area is that you open up the gallery page, the CMS sees that the page type is "gallery" and knows that the gallery block has an admin form which can be used for this page, and so that form is displayed instead of the normal page form.

On the front-end, similarly, the CMS sees that the page requested is a "gallery" type page, and the gallery block has a handler for showing page data a certain way, and so instead of simply printing the normal body text, the CMS asks the block what to do and does that instead (which then displays a nice gallery of images).

Here are the blocks that our CMS has(figure 6).

C:\Documents and Settings\ACER\Desktop\Graduation Report\video.png C:\Documents and Settings\ACER\My Documents\التّنزيلات\oie_11231111Qopio9dk.png **C:\Documents and Settings\ACER\Desktop\Graduation Report\users.png** C:\Documents and Settings\ACER\Desktop\Graduation Report\image gallary.png C:\Documents and Settings\ACER\Desktop\Graduation Report\clock.png C:\Documents and Settings\ACER\Desktop\Graduation Report\calender.png

Video Weather Users Image Gallery Clocks Calendar

C:\Documents and Settings\ACER\Desktop\Graduation Report\tools.png C:\Documents and Settings\ACER\Desktop\Graduation Report\news.gif C:\Documents and Settings\ACER\Desktop\Graduation Report\mp3.png C:\Documents and Settings\ACER\Desktop\Graduation Report\flicker.png  C:\Documents and Settings\ACER\Desktop\Graduation Report\contactus.png

Tools News MP3 Flicker Currency Contact us

C:\Documents and Settings\ACER\Desktop\Graduation Report\article.png

Articles

**Figure 6**

* **Articles**

**We are using Rich-text editing using CKeditor (figure 7):**

Anything entered into the body text area will be displayed directly on the front-end.

It's not a good idea to assume that the administrator knows HTML. Most of them

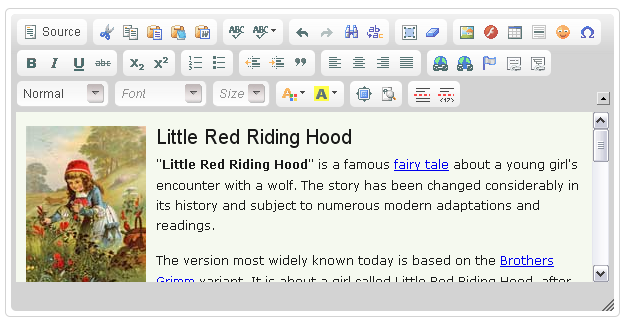
will not. Most things an admin will want to do in a web page are really simple-make something bold, insert an image or table, and so on.

An administrator can use this editor as easily as they would use any word

Processor-you can copy or paste from sources such as websites or word processor

documents and the formats will be mostly retained.

In fact, you can even copy from websites and any images in the copied text will also be copied!



**Figure 7**

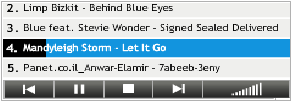
* **Tools:**

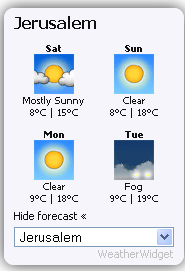
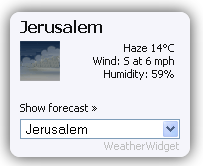
This block is designed to edit header and footer of the front-end by adding titles or images.

* **MP3 :**

Admin can browse MP3 songs to his mp3player list and listen.

It allows admin to play music on his website easy and fast. Player will automatically scan the specified folder and form a playlist. Also admin can edit his songs list.

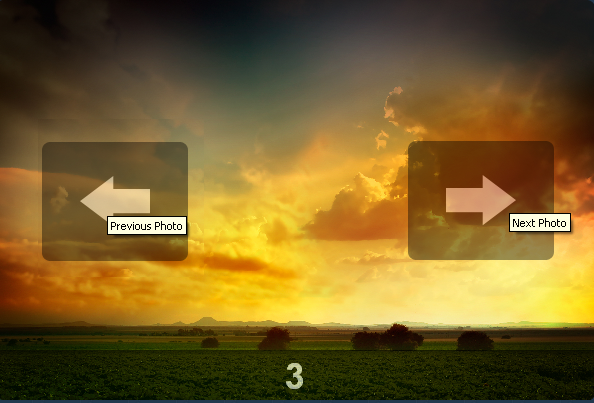
**Figure 8**

* **Weather:** 

**Figure 9 Figure 10**

As you can see in figure 9 you can get the weather for your country, just use the combo box, and if you want to see the forecast for the next 4 days just press “show forecast” .It gets the weather by **AJAX .**

* **Image Gallery:**

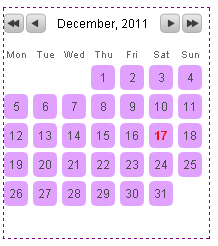
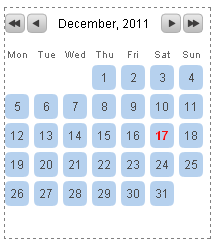
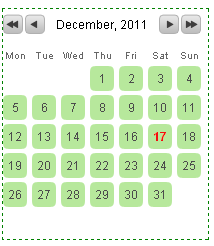
****

**Figure 11**

It’s a simple and easy to use image gallery tool written in PHP. It reads image (png, jpeg, bmp) files in your images folder, and generates thumbnails to show them in the back-end to let admin delete, add images. Then figure11 would be at the front-end as a result.

* **Clocks and Calendar:**

**C:\Documents and Settings\ACER\Desktop\clk2.bmp **

****

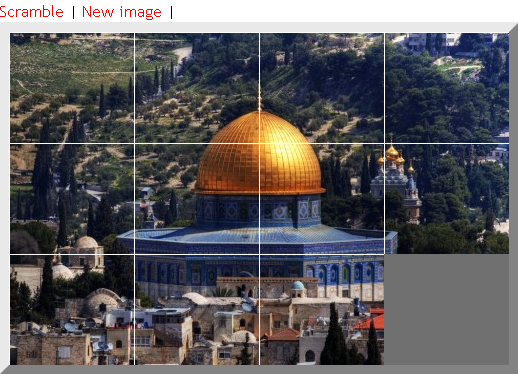
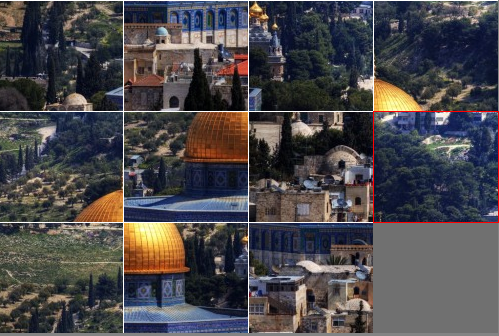
**Figure 12**

You (as an admin) can add a digital, analog clock to your front-end. Also you can add calendar with different colors as shown above.

* **Flicker slideshow:**

If you have a collection of photos that you'd like to display in a smooth, fading slide show then this block is just what you need. With it you simply pass an array of image URLs to it and the JavaScript code you need to create a slide show is returned by it. It’s being used to display a series of photographs from a Flicker image stream.

* **Image Puzzle:**

** **

**Figure 13**

## Video:

With this block you can embed YouTube video.

* **Currency, News:**

Here, you can add your own news (get news from database by AJAX) , also you can get news, currency from other sites.

* **Contact us , add comments Forms:**

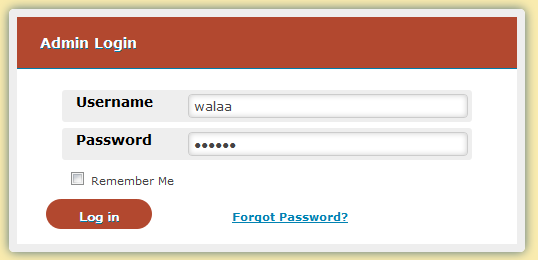
You can add comments form, and contact us form to the front-end easily.

**Chapter 5**

**System in Action**

### Accessing the admin login panel (through local host)

### Login Page

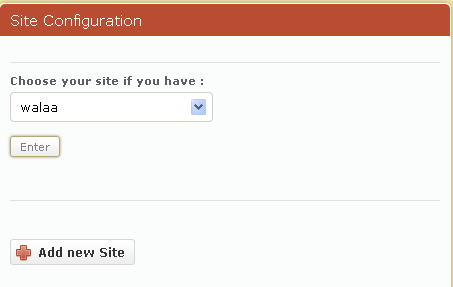


**Figure 14**

This part allows the admin who will create the site to enter control panel, and protects the control panel from unauthorized access, you can’t enter without typing a correct username and password, and if the admin forgets password it could be sent to his email.

1. **Site Configuration**

* If the admin types a correct username and password (successful login), site configuration will show up.

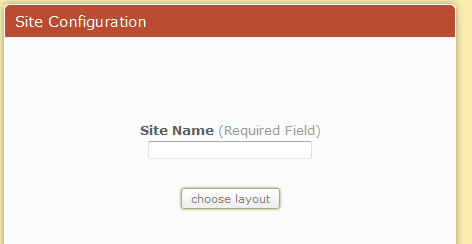


**Figure 15**

### Here, the admin has two choices either choosing to add new site (which is chosen for the first admin login) or entering an existing site.

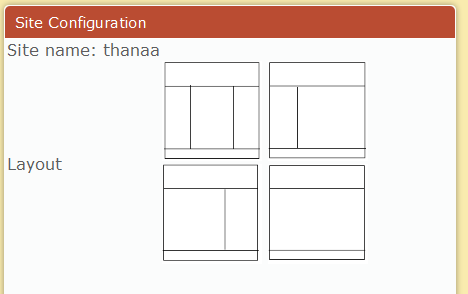
### Let’s take the first choice (add new site).

* The admin has to choose a name for his new site as in figure 16.



**Figure 16**

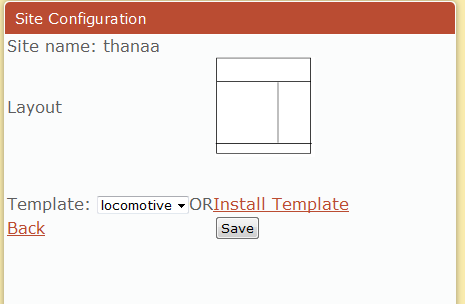
* In addition to the header and footer, most sites have the following layouts :
* Center with right and left.
* Or Center with right only.
* Or Center with left only.
* Or just Center without right ,and left



**Figure 17**

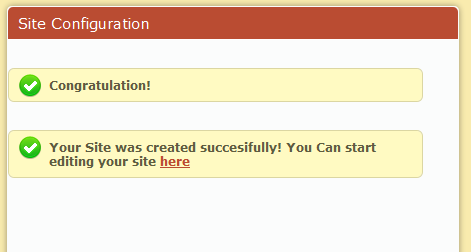
So, according to this, the admin has to choose a certain layout for his front-end as in Figure 17.

Then admin has to choose a template from a combo box which contains available templates, also he has a choice to browse a template from elsewhere.



**Figure 18**

Figure 18 was the last step to create a new site, then figure19 will show up to say that you are done with the basic site configuration.

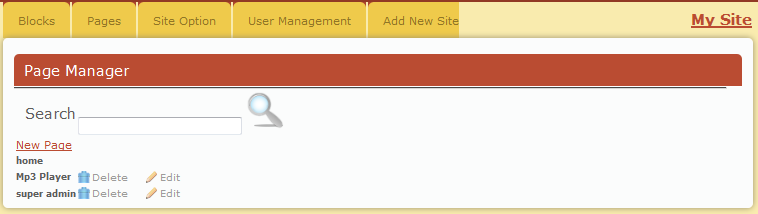


**Figure 19**

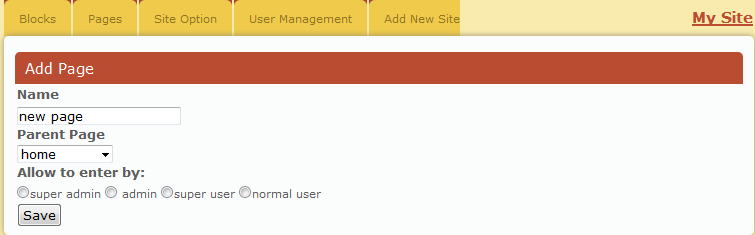
**4. Control Panel**

From here you can start editing your new site.

Admin can add, delete, edit pages (figure20,21)and blocks, also he can change site layout and template. It’s easy to do just select and edit, and that’s done through Blocks, Pages Tabs.



**Figure 20**



**Figure 21**

### 

**Figure 22**

As shown in figure22 admin can see site pages, and blocks for each page. He also can edit the blocks including it’s name and position .

* **My Site:**  Through this option -which is in top to the right-, admin can view his site, which will be opened in a new tab in the browser.



**Figure 23**

* Admin can easily move through his sites if he has more than one site, see figure 24.



**Figure 24**

5. **User Management:**

Through this option admin can easily add users to the site after choosing their types, and can edit them.

* Finally, the following figures show sites that have been done through our CMS with different styles.

** **

Figure 25 Figure 26



Figure 27

**Future Work:**

1. The system can be improved by turning it into drag and drop system.
2. Adding more options per each block.
3. Allowing more than one admin to edit the same site with different privileges.

**Conclusions:**

* Wehave done our best to keep everything smart and simple*.*
* As an advice if you already know one framework, the right one is the one you know. There is usually no great difference between what you can achieve with different frameworks, so given that you have only a short time for the project, it makes sense to use what you know already.
* When you want to do MVC the first time, you do not start from scratch. CakePHP is a really good recommendation. When you have done it many times and understand what MVC is about, then you will be able to implement an MVC without a framework (like what we did).
* Using MVC structure results in simple, clean and understandable code.
* We really achieved our goals from this project(we improved our PHP skills, we built a user friendly CMS )