Migrating the New Horizon Website to WordPress

Bachelor’s Thesis in Computer Systems Engineering

2012 June

Author: Yuanwen Zhu & Jia Wang
Supervisor: Wagner Ourique de Morais
Examiner: Kenneth Nilsson

School of Information Science, Computer and Electrical Engineering
Halmstad University
PO Box 823, SE-301 18 HALMSTAD, Sweden
© Copyright Yuanwen Zhu & Jia Wang, 2012. All rights reserved.
Bachelor Thesis
Report, IDE1214
School of Information Science, Computer and Electrical Engineering
Halmstad University

Description of cover page picture: Migrate the New Horizon website to WordPress
Preface

First of all, we would like to express our sincere appreciation to our supervisor Wagner Ourique de Morais for his support and feedback during the thesis work.

Furthermore, we are grateful to the company Lypson for giving us the chance to do this project, especially Michal Lysek and Tobias Persson who gave us valuable suggestions and technical support.

Finally, we want to thank Halmstad University for offering us the opportunity to do this project. It was a good chance for us to learn more.

Yuanwen Zhu & Jia Wang

Halmstad University, June 2012
Abstract

There are different problems associated with a static website, such as management and content update. To address these problems, one alternative is to use Content Management System (CMS), such as WordPress. This project describes the process of migrating a static website to WordPress and adding six additional functionalities according to requirements from a company. In the meantime, theme is used to customize the website based on company’s requirements; specific WordPress plugins and widgets are used to implement additional requirements, including submenus, user registration and login, content update, Facebook and Twitter integration, and RSS support for the new content. Besides, solutions are given to resolve problems, such as the conflict with Internet Explorer (IE) and WordPress security issues. The outcome of this work allows the company to use WordPress to manage the website, to publish automatically new content into online social networking services, to provide customized information to registered users as well as to allow web syndication.

Key words: Content Management System, Solution Stack, Static and dynamic website, WordPress
List of Figures

Figure 1. Client-server architecture for the server-side scripting technology .......... 6
Figure 2. WordPress theme interface .................................................................... 15
Figure 3. Page Attributes in the Races page ......................................................... 19
Figure 4. The submenu is delimited by the orange rectangle in the Figure .......... 20
Figure 5. Submenu is displayed when users are not logged in ......................... 23
Figure 6. User registration and login page .......................................................... 23
Figure 7. Races Page .................................................................................... 23
Figure 8. Add New Post .................................................................................. 30
Figure 9. Add the image to the left of the content .......................................... 30
Figure 10. Choose the category ....................................................................... 30
Figure 11. View the post ................................................................................ 31
Figure 12. Show the comment area ................................................................. 31
Figure 13. Administrators can upgrade a standard user to an administrator or a publisher ................................................................. 32
Figure 14. Remind the user to login ................................................................ 32
Figure 15. Login interface ............................................................................. 33
Figure 16. Display the post on the Facebook wall ......................................... 36
Figure 17. The new post is shown in the Twitter ............................................ 39
List of Tables

Table 1. Common solution stacks................................................................................................. 7
List of Acronyms

API       Application Program Interface
ASP       Active Server Pages
CALs      Client Access Licenses
CAPTCHA  Completely Automated Public Turing Test to Tell Computers and Humans Apart
CMS       Content Management System
CSS       Cascading Style Sheets
CURL      Client URL Library
FAQ       Frequently Asked Questions
FTP       File Transfer Protocol
GUI       Graphical User Interface
HTML      Hypertext Markup Language
IE        Internet Explorer
ID        Identification
IIS       Internet Information Services
IT        Information Technology
PC        Personal Computer
PHP       Hypertext Preprocessor
RPG       Role Playing Games
RSS       Really Simple Syndication
SQL       Standard Query Language
URL       Universal Resource Locater
WMS       Web Management Systems
WP        WordPress
XAMPP     Cross-platform operating system, Apache, MySQL, PHP, Perl
XML       Extensive Makeup Language
1 Introduction

The idea for this project has been proposed by the company Lypson Intelligent Systems in Halmstad [1]. The company develops software solutions for different applications as well as games, such as the New Horizon adventure game, which is a table top RPG (Role Playing Games) [2].

The New Horizon game’s website is static, containing a quantity of separate web pages that are individually created using Hypertext Markup Language (HTML) and partly with Hypertext Preprocessor (PHP)\(^1\), which are stored on the web server in respond to the clients’ request in the future [3].

Continuing up the scale, the New Horizon website is confronted with some troubles, especially time-consuming maintenances and updates. When the company wants to update the content of any web page, they need to find the page, edit it using a web page editor, and then update it on the web server. Sometimes changes might be necessary into other pages that are related to the original page. Furthermore, it perhaps involves some extra duplicated work when the company wants to change the look and layout in the required pages. Meanwhile, it is hard to keep track of all these locations when web developers copy and paste code into multiple places, so if they modify at one place and want to get the same effect at those locations, developers need to find them and update the code, which leads to delays in update.

In fact, the company has recognized that updating and maintaining the website become more complex and expensive as the number of pages and their content grow. From this point of view, a dynamic website is considered, which can generate pages in real-time when visitors request them.

1.1 Objectives

Our aim is to migrate the current New Horizon website to WordPress. WordPress is one kind of blogging platform, whose details will be discussed in the subsequent chapters. The process involves developing a WordPress theme which mimics the design of current New Horizon RPG website and adding new functionalities to extend the website according to company’s requirements. In

\(^1\) The HTML content text statically embedded in the .php files
addition, pages are made to be adapted for different screen resolutions to fit well
PC-screens, mobile phone screens, and Internet tablets.

In addition to the website migration, new features have been required by the company, such as:

- **Submenu**
  
  Submenu is introduced to replace the current menu so that administrators can easily add new items to the menu.

- **User registration and login**
  
  The website must contain user login functionality. In addition, specific pages/folders must only be accessible by users who are logged in.

  Different registration levels must exist, such as administrators, publishers, and standard users. Administrators are able to upgrade a standard user to an administrator or a publisher; publishers are able to create and publish their own posts. In the beginning, users can only register as standard users, with only access to comment on posts and to access specific pages/folders only available to registered users.

- **Content update using WordPress**
  
  Posting should be converted into WordPress so that administrators or publishers can update or modify the content dynamically.

- **Facebook integration**
  
  The post in WordPress can automatically appear on specific Facebook wall connected to the *New Horizon* website.

- **Twitter integration**
  
  The post in WordPress can automatically appear on specific Twitter connected to the *New Horizon* website.

- **RSS support for new content**
  
  RSS should be available for visitors that want to be informed about changes to the website. The RSS feed link must be put at the right side of the website.
1.2 Outline

The remainder of this report is organized as follows. Chapter 2 includes background information relevant to this work. Chapter 3 presents the main tools and methods used to develop a web application and to adapt to WordPress. Chapter 4 gives the specific solution to each objective. Chapter 5 involves how to move WordPress from one server (local) to another (live). Chapter 6 summarizes the work done in this project and describes future work.
2 Background

This chapter describes the literature review of relevant technologies, which are used to settle the matter existing in the static website. It includes dynamic websites, a solution stack, a Content Management System (CMS), and WordPress.

2.1 Dynamic Websites

In a dynamic website, a new page can be created based upon certain user interaction on the fly. The layout of the page is fixed in advance, but the content is filled dynamically from various sources such as database, which stores and retrieves the related information when requested. Then the result is sent back to the web server to insert the desired information in the specific page. Finally a customized web page is sent to the client’s web browser.

A web application is required in developing a dynamic website. It is an application that is stored on a web server beforehand and accessible over the Internet [4]. When clients send requests to the server via the web browser, the server checks request, execute the server side scripts such as PHP, and then send the result to the browser.

2.1.1 Solution Stack

In order to develop a web application, different software components are required on the server: an operating system, a web server, a database, and a scripting language. These four components are commonly called solution stack [5].

A server operating system is the platform where other three components run. The most common ones are: Linux, Windows Server, Solaris, Mac OS X Server, and so on. Linux uses open source licenses, which allows the products to be installed on multiple servers. Windows Server uses user licenses, which only allow users with Client Access Licenses (CALs) to connect to the server. Solaris uses license change, which is free of charge for a single CPU server, but will charge for a multiprocessor machine. Mac OS X Server uses unlimited user license rather than CALs [6].

The web server is a program that accepts the request for pages from a browser, interprets the request, and returns the results. Apache and Internet Information Server (IIS) are widely used. Generally, Apache is considered more secure than IIS. In addition, Apache is open-source [7].
A database is a collection of data. **Database Management System (DBMS)** is a database software program, which allows structured storage of data for easy insertion, maintenance, and retrieval [8]. **Relational Database Management System (RDBMS)** is a DBMS based upon relational model, which organizes data in the form of rows and columns, such as Oracle, MySQL, and Microsoft SQL Server [9]. MySQL is commonly used, since it is an open-source database that can be run on all operating systems.

The scripting language is an advanced programming language that is interpreted by the script engine at runtime, i.e., PHP, ASP.NET, Perl, and Python [10]. The server-side scripting technology is generally selected to write a web application, due to the short load time and further communication with the server, compared with the client-side scripting. How it works is presented below (see Figure 1).

![Figure 1. Client-server architecture for the server-side scripting technology](image)

PHP is one kind of the server-side scripting technology. It is open-source, which is designed to work with the web server, usually along with the MySQL database server.

Commonly, a solution stack is represented by an acronym, which describes its four components necessary to develop web applications. Examples of these acronyms are presented in Table 1.
<table>
<thead>
<tr>
<th>Solution Stack</th>
<th>Operating System</th>
<th>Web Server</th>
<th>Database</th>
<th>Scripting Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>WISA</td>
<td>Windows Sever</td>
<td>IIS</td>
<td>SQL Server</td>
<td>ASP.NET</td>
</tr>
<tr>
<td>WAMP</td>
<td>Windows Sever</td>
<td>Apache</td>
<td>MySQL</td>
<td>Perl, PHP, or Python</td>
</tr>
<tr>
<td>LAMP</td>
<td>Linux</td>
<td>Apache</td>
<td>MySQL</td>
<td>Perl, PHP, or Python</td>
</tr>
</tbody>
</table>

Table 1. Common solution stacks

### 2.2 Content Management System

A CMS is a software package specifically designed to manage a website [11]. It separates the content from look and layout on each page, which is customized by a template. Text and images on each page are delivered via the web server as the page is requested from the browser.

Content management involves creation, management, publication, and maintenance of information in any form or medium. Once a page has been created, it is saved into a central repository in the CMS. Later, it can be published to the website. Each author has a unique role and set of responsibilities in the publication of the content, which is beneficial to the administration and maintenance of the website.

Generally, a CMS consists of the following basic roles and responsibilities [12]:

- **Administrators** can allow a variety of roles to access specific folders or files by granting certain access rights to them.
- **Creators** can create and edit the content.
- **Publishers** can release the content.
- **Visitors** can read the content after posts are published or shared.

A number of useful features in the management and maintenance of the website are provided by a CMS [13]:

- Keep track of all the changes of the website, which includes who changed, what, and when.
- Only allow each user to change the section with the limited permission.
- Incorporate existing information resources and IT systems into the new system.
Most importantly, a CMS provides a range of workflow capabilities. If the content is created by an author, it will be sent to management center automatically for the approval and review. Finally, the content will be published after getting the sign-off from the administrator.

In this project, the most common use of a CMS is focused on: manage web content, sometimes called Web Management Systems (WMS), which intends to simplify the process of creating web applications and to facilitate the editing, publishing, management, and maintenance of websites. Moreover, users can interact with the system through a web browser anywhere, control parts of the layout, and manage web pages without any technical knowledge [14].

2.2.1 Advantages and Disadvantages of a CMS

Compared with developing a website with other software, a CMS is a good alternative, since it empowers users to take full control of their website without programming skills [11],[15].

Several benefits are listed below:

- **Improve communication among users**
  Users can make changes of their website any time they want. This is increasingly important for the owner who works on business and wants to use the website as a communication channel.

- **Simplify log report writing**
  A CMS can generate all the technical details automatically so that anyone can manage the website easily. It is simple to keep track of the records.

- **Control data validity and compliance**
  A CMS can keep the design of all the pages consistent.

- **Additional functions are added via plugins**
  Website functions and features are added by plugins, so that creating a technical architecture for users’ target audiences is not normally required.
Full template supports

The content can be separated from design based on templates so that there is no interference between each other. Thus, web developers can alter the content without influencing the layout in the website.

However, a CMS also has its drawback. Security issues must be supervised by a webmaster. For example, when security patches are released, a webmaster needs to upgrade the CMS timely.

2.3 WordPress

WordPress is a free and open-source blogging platform and CMS software based on PHP and MySQL, which is used to write a web application. It has two main features: a plugin architecture and a web template system. A plugin architecture allows users or developers to extend functionalities based on common features; a web template system is used to generate web pages and for websites deployment and delivery via the Internet.

Currently, WordPress is the most popular CMS in use on the Internet [16]. It allows users to create and edit web pages via the Internet. Besides, no knowledge of HTML, PHP or other code is required [17]. A WordPress editing interface is intended to be easy to use, which works similarly to a word-processing software. In addition, administrators can grant various access levels to different users and allow them to contribute to the website.

There are three main features of WordPress: themes, plugins, and widgets [18].

• Themes

WordPress allows users to install and switch the look of the website between themes. There are two alternatives of installing the themes available. One is to use the WordPress “Dashboard” administration tool; the other one is to upload theme folders via FTP. In addition, the PHP and HTML code in themes can be modified for more advanced customizations.

• Plugins

WordPress plugins are series of software built to add functionality to the WordPress blogging platform, which allow users to do almost anything they want and can be installed in a
moment [19]. Users can use plugins to achieve some functions instead of writing the code from the scratch. They are added and activated by using the WordPress "Dashboard" administration tool. In this project, several reputable plugins are used, such as WP-Members plugin and WP to Twitter plugin. More details about these plugins are given in Chapter 4.

- Widgets

Widgets are small modules that can be added or deleted by dragging and dropping to the sidebar area for implementing plugins’ extended abilities, such as a slideshow.

2.3.1 Advantages and Disadvantages of WordPress

As WordPress becomes more and more popular, a larger number of users choose WordPress as a CMS.

Several advantages are listed below:

- Premium quality theme

  WordPress offers numerous pre-made designs themes, which have premium quality and various styles that make users change themes at will.

- Thousands of plugins

  Plugins can realize many functionalities and applications that users want.

WordPress also has some drawbacks. Firstly, security has aroused general concern and discussion. The unsecure plugins and themes might lead websites to get hacked. Thus, users need to keep up updating the website in case it is hacked. Secondly, the conflict between Internet Explorer (IE) and WordPress is a big issue as well, which will result in different effects compared with other browsers.
3 Methods

The main methods used in the project are described in this chapter. It includes development environment and some requirements aimed to adapt to WordPress.

3.1 Development Environment

3.1.1 XAMPP as Solution Stack

According to Section 2.1.1, the development of a web application requires a set of software components on the server. One of the many possible solution stacks available is XAMPP, which is a free and open-source cross-platform web server solution stack package. X means cross-platform that contains any of four different operating systems, i.e., Microsoft Windows Server, Linux, Solaris, and Mac OS X Server; A is Apache (web server); M is MySQL (database); the two P stand for PHP (scripting language) and Perl [20].

In this project, Windows is chosen as operation system. Thus, XAMPP 1.7.7 for windows has been used [21]. In the meanwhile, PHP is chosen as server scripting language; Apache is chosen as web server; MySQL is chosen as database.

Installing XAMPP takes less time than installing each of its components separately. XAMPP can be installed on a single computer and any given instance can be copied from one computer to another [20]. Thus, it is ease of use. Users just need to download, extract, and start it.

3.1.2 Environment Establishment

Before using WordPress, an environment needs to be set up. The main steps of establishing XAMPP are described below [22]:

- **Step 1: Installing XAMPP**

  Follow the instructions in the screen. Meanwhile, check all the checkboxes.

- **Step 2: Running XAMPP**

  Open up the XAMPP control panel. Turn on Apache and MySQL by pressing the "Start" button.
Step 3: Creating the WordPress database

Open a browser and input http://localhost/phpmyadmin/. Click the “Privileges” tab and select the “Add a new User” link. Input the user name, password, and other fields. Note that the Host field must be localhost. In the Global Privileges area, check all the options, and then press the “Create User” button. Finally, press the “Go” button.

Click the “Databases” tab. In the Create new database field, input newhorizon_pcg as name and select utf8_general_ci as collation. Finally, press “Create” button.

Step 4: Configuring WordPress

Download the latest version of WordPress in the WordPress.org. Unzip the WordPress zip file to the “htdocs” folder of the XAMPP installation directory.

Rename wp-config-sample.php file to wp-config.php. Replace the default information of DB_NAME, DB_USER, and DB_PASSWORD with the information set in Step 3.

Go to https://api.wordpress.org/secret-key/1.1/salt/, copy the text generated by the link automatically to replace the default text in the wp-config.php file.

Finally, save the wp-config.php file.

Step 5: Installing WordPress

Log in the admin panel via http://localhost/wordpress/. Fill out the information in the Site Title, Username, Password, and Your E-mail area. Finally, press the “Install WordPress” button.

3.2 WordPress Adaptation

First, add the CSS file extracted from the original CSS file provided by the company into the themes folder. It is usually called style.css, which is intended to control the style of the web page.

Then, WordPress requires programmers to split the complete .php files into several different parts: header.php, index.php, sidebar.php, footer.php, and functions.php. It is crucial to put these .php files under one folder which the
theme works in. Finally, a WordPress theme is built. Details related to the .php files are listed below:

- **header.php**

  This file is constituted by the code for the header section of the theme [23].

  At the top of the header.php file, it is filled with HTML `<!DOCTYPE>` tag. Then, the information about page title, meta, and stylesheet information should be written.

- **index.php**

  This is the main file for the theme. It consists of the code for the main area and simultaneously specifies where the other files will be included [23].

  The index.php file needs to combine two main .php files’ contents. One is to include the header.php file’s content by writing the code `<?php get_header(); ?>` on the top; the other one is to include the footer.php file’s content by writing the code `<?php get_footer(); ?>` near the bottom. Inside the file, any code related to WordPress, PHP, and HTML can be put to realize specific functions.

- **sidebar.php**

  This file includes the information about the sidebar [23].

- **footer.php**

  This file contains the information about the company such as the copyright, some blogrolls like Reptid Forums, Fenix, Scribd, and so on. The relative source code should be put at the bottom.

Such structure is visually represented in Figure 2.
Moreover, specific .php files can be added under the theme folder when developers want to achieve more applications.
4 Implementation and Results

This chapter presents the achievements in six sub-objectives. They are stated separately by description concerning the existed problem, implementation on how to achieve the goal, resulting Graphical User Interface (GUI) with figures, manual about how to use it, and tests with respect to result whether is successful or not.

4.1 Submenu

Nowadays, no submenu exists in the current website; perhaps there is no space to insert more items in one line in the future. Therefore, submenu is introduced, so that administrators can add new items.

There are two main methods used to implement submenu functionality, which are WP functions and WP plugin.

WP functions used here are functions that are intended to include the header.php template file from the current theme’s directory. They are presented below:

- **get_header()**
  
  This function is used to include the header.php template file from the current theme’s directory [24].

- **wp_footer()**
  
  This function is one of the most essential theme hooks. It should be put before </body> tag in a theme template [25].

- **wp_head() and wp_footer()**
  
  They are placed correctly in your template, which makes sure that there is no trouble with a plugin. Because **wp_head** and **wp_footer** as action hooks are essential to how plugins interact with the WordPress theme.

WP plugin used here is Multi-level Navigation plugin which is intended to create the horizontal submenu [25].
4.1.1 Implementation

To display the submenu, in the index.php file, `<?php if (function_exists('pixopoint_menu')) {pixopoint_menu();} ?> was used to replace original code associated with the title and written with HTML.

To convert current web pages into WordPress, the source code of the home page was split into two basic parts, which are index.php and footer.php. The content of the body was put in the index.php file; the information about the blogroll and the copyright were put in the footer.php file. Meanwhile, `<? get_header(); ?> was added at the top of the index.php file; `<? get_footer(); ?> was added at the bottom of the index.php file. `<?php wp_footer(); ?> was added before `</body>` tag in the footer.php file.

The .php files were extracted from the legacy New Horizon website provided by the company, which included body.php, header.php, menu.php, title.php, submenu_newhorizon.php aff_links.php, footer.php, submenu_races.php, newhorizon.php, and races.php files.

The file names were modified in order to be more understandable and no conflict with the existing .php file name. Meanwhile, the name of the footer.php file was changed to footerplanet.php; the name of the header.php file was changed to headerplanet.php; the name of the newhorizon.php file was changed to planet_use.php.

Besides, `<?php bloginfo('template_directory'); ?> was added before the code of each position of the image insertion; `<? get_header(); ?> was added at the top of each main page (planet_use.php and races.php). They were intended to adapt to WordPress.

In addition, three main pages were used as the template. Meanwhile, `<?php /*Template Name:races*/ ?> was added at the top of the races.php file; `<?php /*Template Name:planet*/ ?> was added at the top of the planet.php file; `<?php /*Template Name: Homesite*/ ?> was added at the top of the homesite.php file; `<?php header('Location:http://localhost/wordpress'); die(); ?> was added inside the homesite.php file.

In the planet_use.php file, `<?php wpmem_login_status(); ?> was added below the template declaration, which was used to display user's current status. Inside the planet.php file, `if` statement was used to judge whether the user has already logged in or not. If the user has logged in, the planet_use.php file will be
invoked, which can display the planet page to the registered user; if the user has not logged in, two options (Existing users Login and New Users Registration) will be provided with the user.

For further details please refer to the Appendix A - Mark 1.

In the dashboard, three pages were added. These were called World, Planet, and Races.

In the World page, select no parent as Parent and Homesite as Template, and set Order as 1; In the Planet page, World was selected as the Parent and planet was selected as Template. The Races page’s setting was similar, as shown in Figure 3.

![Page Attributes](image)

Figure 3. Page Attributes in the Races page

In the Settings > Multi-level Navigation > Menu contents, select Home as Menu Item #1 and Pages as Menu Item #2 in the Main menu contents area; select None as other menu items. The configuration in the Second menu contents area was same as this.

In the Settings > Multi-level Navigation > Appearance, copy and paste the style code to the text area, as shown in the Appendix A - Mark 2. Then, press the “Update Options” button.

Due to the conflict between IE and WordPress, the particular code was written for IE, which contains one function intended to judge the type of the browser and special code, which is applicable to IE and same effect in other browsers.

4.1.2 Resulting GUI

Once the mouse cursor hover around World, the submenu will drop down and the Planet and Races will be shown (see Figure 4).
4.1.3 Manual

If administrators want to add the new menu after the menu *World*, they need to add a new page in the dashboard, and set the *Order* as 2.

4.1.4 Tests

In this section, the main test is performed to check the usability.

When mouse cursor hovers around the menu *World*, the submenu consisting of the menu *Planet* and the menu *Races* appears vertically below it. The corresponding pages are displayed when the menu is clicked; when the menu *World* is clicked, the original page still remains.

4.2 User Registration and Login

Nowadays, there is no user login interface, so everyone can visit all the pages. However, the company wants to enable visitors to register themselves into the website, so that these kinds of visitors can get access to more content than unregistered users. Moreover, different registration levels must exist. Therefore, user registration and login functionality must be added to the current website.

There are two main methods used to implement user registration and login functionality, which are WP functions and WP plugin.

The relative widget as the plugin is required to be added to the sidebar. Thence, some source code related to the sidebar should be added to the sidebar-right.php and function.php files. How it looks is defined in a *Cascading Style Sheets (CSS)* file [26].
Some related functions are listed below:

- **function_exists ($function_name)** is used to check whether given function has been defined or not.
  
  `$function_name` is the name of the function.

- **dynamic_sidebar($index)** is used to get the list of the sidebars and their widgets.
  
  `$index` is the sidebar's ID or name argument.

- **register_sidebar($args)** is used to define the single sidebar.
  
  - **name** - the name of the sidebar (same with the file name of the sidebar).
  - **before_widget** - HTML to place before every widget.
  - **after_widget** - HTML to place after every widget.
  - **before_title** - HTML to place before every title.
  - **after_title** - HTML to place after every title.

*WP-Members* plugin used here is intended to achieve the Login/Registration functions [27].

In most cases, a dynamic sidebar is registered for users to load some widgets. Where the widget is displayed depends on the sidebar template of the working theme.

### 4.2.1 Implementation

In the style.css file, some code was added to indicate the location (right) of the widget, as shown in the Appendix B - Mark 1.

In the *sidebar-right.php* file, `<?php if ( !function_exists('dynamic_sidebar') || ! dynamic_sidebar('Sidebar Right') ) : endif; ?>` was added between the `<div id="content-rightside">` and `</div>`, which was used to check whether the sidebar was registered with widgets or not. If they were active, they would be displayed on the specific position.

For further details please refer to the Appendix B - Mark 2.

Furthermore, `function_exists('register_sidebar')` was added to judge whether `register_sidebar`'s function was defined or not. If it existed,
sidebar can be registered. Array was used to pass the parameter to the function. The name of the sidebar was \texttt{sidebar-right}, same with the file name of the sidebar (\texttt{sidebar-right.php}).

For further details please refer to the Appendix B - Mark 3.

\texttt{<?php get\_sidebar('right'); ?>} was added to load sidebar templates to the index.php file.

In the \texttt{Settings > WP-Members > Options}, input \url{http://localhost/wordpress/wp-admin/profile.php} in the \texttt{Members Area URL} area; input \url{http://localhost/wordpress/wp-login.php?action=register} in the \texttt{Register Page URL} area.

\textit{Anyone can register} was checked under \texttt{Settings > General} of the dashboard.

In the \texttt{Settings > WP-Members > Options}, \texttt{Block Posts by default} was unchecked to use WordPress built-in login interface and \texttt{Block Pages by default} was checked to use \texttt{WP-Members} login interface.

In the \texttt{Settings > Discussion}, before a comment appears, \textit{An Administrator must always approve the comment} was checked.

In the \texttt{Appearance > Widgets}, drag and drop the \texttt{WP-Members Login} widgets to the \texttt{sidebar-right} area, and type the title in the \texttt{Title} text area.

\textbf{4.2.2 Resulting GUI}

The register and login box is on the top right corner of the home page. When a visitor click the menu to access the \textit{Planet} page, the register and login page will come up no matter whether he has not logged in or not. If he has already registered, he just needs to type the username and password to log in, and then he can visit the specific page; if not, he needs to register to get the right to browse the specific web page.

When the user (not logged in) moves the cursor to the \textit{World}, the submenu will appear as well (see Figure 5). But, when he still want to browse the \textit{Planet} page, the login and register page will come up (see Figure 6).
However, the *RACES* page can be accessed to all the users, no matter whether they logged in or not (see Figure 7).
4.2.3 Manual

On the right side of the home page, there is the login interface. There are three options provided with users. Login their accounts by typing the correct username and password; register by selecting the "Register" link; retrieve the password by selecting the "forget?" link.

In WordPress, authors (publishers) can publish and manage their own posts; subscribers (standard users) can only manage their profile; administrators can access to all the administration features. Thus, subscribers can just comment on blogs and access specific pages/folders; authors can create and post their own blogs; administrators can upgrade a standard user to an administrator or a publisher.

In the dashboard, administrators can choose which user they want to edit. Then they can press the “Edit” button below the username. In the name range, they can modify the role of the user (author or administrator).

4.2.4 Tests

In this section, the main test is performed to check user registration and login functionality is successful or not.

At the home page, users with a correct username and the corresponding password can be logged in via the login interface; once the “Register” link is selected, a register page is shown and requires users to input the username and email; once the “Forgot?” link is selected, users can retrieve the password by typing the correct username or email.

Once the Planet submenu is clicked, the register and login page appears in the unlogged in condition; the Planet page is shown as long as username and password are provided correctly. Otherwise, login will fail; the password can be reset, after the “Click here to reset” link is selected. The username and email are required to register a new account. An error message is gotten when the username is already occupied, the email is illegal, or any of them is missing.

The Races page is available to all the users, no matter whether they have registered or not, or they have logged in or not.

4.3 Content Update using WordPress

Currently, administrators or publishers need to know HTML, CSS, and FTP, when updating or modifying the content in the website. This implies that the more pages appear on the website, the more work is required. In this section, some
parts associated with post can apply WordPress built-in post functions plus some categories to enable administrators or publishers to update the content without technical knowledge though the admin dashboard.

There are two main methods used to implement content update using WordPress functionality, which are WP functions and WP plugin.

WordPress provides built-in WP functions to achieve publishing the post dynamically. The following functions have been used:

- **have_posts()**
  This function is used after *if* statement and *while* loop to check if the current website have more different posts to loop over [28].
  The result type is *Boolean*. If there are more posts, it returns *TRUE*, otherwise it returns *FALSE*.

- **the_title()**
  This function is used to display the title of the current post [29].
  The Loop must be used.

- **the_content()**
  This function is used to display the contents of the current post [30].
  The Loop must be used.

- **in_category( $category )**
  This function is used to judge if the current post belong to the specific category [31].
  $category represents the category ID (integer), name or slug (string), or an array.

- **paginate_links( $args )**
  This function is used to create paginated link list [32].
  « Previous, Next » and specific pagination can be shown on the website.
  $args means some parameters should be added.
• **Base**
  Used to reference the URL, which will be used to create the paginated links.

• **Format**
  Used for Pagination structure.

• **Current**
  Used to record the current page number

• **Total**
  Used to record the total amount of pages.

- **add_action** ($tag, $function_to_add)

  This function is used to hook a function on to a specific action [33].

  $tag is the name of the filter to hook the $function_to_add to.

  $function_to_add is the name of the function to be called.

- **add_filter** ($tag, $function_to_add)

  This function is used to hook a function to a specific filter action [34].

  $tag is the name of the action to which $function_to_add is hooked.

  $function_to_add is the name of the function, which is called when the filter is applied.

PHP function used here is intended to achieve passing the parameter to the WP function (paginate_links( $args )). Details about the PHP function are described below:

- **str_replace** ($find, $replace, $string, $count )

  The first three parameters are required, but the last one is optional [35].

  $find specifies the value to find in $string.

  $replace specifies the value to replace $find in $string.
$string$ specifies the value to be searched.

$count$ calculates the number of replacements.

WP plugins used here are Display Post Image and WP-PageNavi plugin, which are intended to display the image in the designated position and to create the horizontal page number. Information of the two plugins is presented below:

- **Display Post Image**
  
  Sometimes the publisher wants to insert the image on the left side of the content. Display Post Image plugin can be used to insert the image to the specific location of the post [36].

  The template tag `display_post_image()` can be put to the specific position where the publisher wants to put the image. User can insert the image to the specific place via WordPress media Upload/Insert.

- **WP-PageNavi**
  
  This plugin is used to add the paging navigation, including current page’s status, previous arrow, next arrow, and pagination [37].

WordPress Categories are used to classify and organize the user’s posts.

### 4.3.1 Implementation

To convert static posts to dynamic posts, the specific source code of HTML was extracted. Meanwhile, some corresponding PHP source code was replaced.

According the company’s requirements, there were four different layouts of the website. Custom category means that the publisher can insert the image on the left side of the content; Original category means that the publisher just can write the content; Customcomment category means that comments link will be shown at the bottom of the content based Custom category; Originalcomment category means that comments link will be shown at the bottom of the content based Original category.

First of all, source code related to the post was extracted from the current website. Then the three new .php file was created. Inside the file, the relative extracted code was put. Later, depending on the layout of current post, `<?php the_title(); ?>` was put to the location of title; `<?php the_content(); ?>` was placed to the location of the post’s content; `<?php
The method above was common approach to deal with the different layouts. The only different thing dealing with diverse layouts was whether to insert image on the left of the content or not and whether to put comments below the content. To the former, the *Display Post Image* plugin can be used to accomplish it. To the latter, PHP code can be brought in.

To the former, `function_exists('display_post_image')` was added to judge whether the `display_post_image` function was defined or not. Then `display_post_image()` function was invoked to display the image, which the publisher wants to put on the left of the content. Furthermore, the width and height of the image can be set through the function’s parameter.

For further details please refer to the Appendix C - Mark 1.

To the latter, `<a class="commentcolor" href="<?php comments_link();"><"">Comments to This Post</a>` was used to link the comments related to the post.

In the index.php file, `<?php while (have_posts()) : the_post(); ?>` and `<?php endwhile; ?>` were used to loop over all the published posts. Inside the loop, four conditions existed. Thus, `if` statement and `in_category()` function were used to judge whether the current post was assigned to the specific category or not, if it was true, then displayed the layout according to the specific .php file. At last, `paginate_links()` function or `wp_pagenavi()` template tag was invoked to achieve the pagination.

For further details please refer to the Appendix C - Mark 2.

A new category was created in the *Posts > Categories* in the dashboard.

In the editing area of the new category, name and slug were written according to diverse layouts. Then, press the “Add New Category” button.

In the dashboard, *Site Title’s* content was changed into *New Horizon* in the *Settings > General*.

Modifications in .php file are listed below:

- Login and Register Interface in WordPress

  For the logo in the login interface, inside the `functions.php` file, the new function called `custom_loginlogo()` was created to
modify the image of built-in Login and Register interface in WordPress. Inside the function, title_new.png was introduced as the background image. Then, \texttt{add_action('login\_head', 'custom\_login\_logo')} was added to hook the \texttt{custom\_login\_logo} function on to the \texttt{login\_head} action.

For further details please refer to the Appendix C - Mark 3.

For the logo link in the login interface, the new function called \texttt{custom\_login\_logo\_url($url)} was created to return the link address of the logo. So user can click the \textit{New Horizon} logo to come back to the home page in the login interface.

For further details please refer to the Appendix C - Mark 4.

- **Post**

  In the \texttt{\wordpress\wp-content\themes\NewHorizon} directory, add the code \texttt{#blogcontent{width:530px;}}, relative alignment style, and hyperlink style code into the style.css file to style the content area.

  Specific information is included in Appendix C - Mark 5.

- **Pagination**

  In the \texttt{\wordpress\wp-content\plugins\wp-pagenavi} directory, copy and paste the style code to overwrite the original \texttt{pagenavi-css.css} file to adapt to the overall design of the website, as shown in the Appendix C - Mark 6.

4.3.2 Resulting GUI

The publisher can add the new post, including the title, texts, and images in the dashboard. The interface is just similar to writing email, as shown in Figure 8.
When the publisher wants to add the picture into the left of the content, he can select “Upload/Insert” link to upload the image and set the image’s action to 1. Then, *add media* window pops up, as shown in Figure 9.

![Add Media Window](image)

**Figure 9. Add the image to the left of the content**

The publisher can also select the category. In this example, the *Customcomment* is chosen (see Figure 10).

![Choose Category](image)

**Figure 10. Choose the category**

The publisher can press the “Update” button to publish the post. Meanwhile, he can see the result via pressing “View Post” button, as shown in Figure 11.
Then the visitor can see the post, as shown in Figure 11. If the visitor wants to leave the comments, he can select the “Comments to this post” link. Later, the comment area will be shown (see Figure 12).

If the user as a standard user logged in, he can modify his profile and leave a comment; if the user as a publisher logged in, he can modify his information, edit the post, and leave a comment; if the user as an administrator logged in, he can do all the things mentioned above and also can upgrade a standard user to an administrator or a publisher (see Figure 13).
Figure 13. Administrators can upgrade a standard user to an administrator or a publisher

When the user does not log in, a hint will be shown to remind the visitor to log in, as shown in Figure 14.

Figure 14. Remind the user to login

When the user selects the “logged in” link, the new page will come up, as shown in Figure 15. After inputting the username and password, he can log in the system successfully. Likewise, he can register the new account. Certainly, he can click the New Horizon logo or click the Back to New Horizon to come back to the home page.
After the user logs in, the comment area will be displayed, as shown in Figure 12.

4.3.3 Manual

In the dashboard, the publisher can click *Add New Posts* to add the new post.

First, the publisher needs to choose which category meets his requirement via checking the specific category on the right during the editing status of the post.

Second, the publisher can write the title in the location of *Title*.

Third, the publisher can insert the image to the left of the content by selecting “Upload/Insert” link if he chooses the *Custom* category of the post. Remember the action of image must be set to 1. After the publisher adds the image from the computer, the image will appear in the gallery. Also he can edit the action of the image in the gallery.

In the content area, the publisher could write the content. Moreover, he can insert the image in the content area via selecting “Upload/Insert” link and add the hyperlink to the image in the editing area. Remember to press “insert into the post” button in the *Gallery* of the *Add Media*. Furthermore, the publisher can put the image to the right via clicking “Align Right” icon. In addition, the publisher can add the hyperlink to the specific content via clicking the “Insert/edit link” icon if it is needed. After finishing the editing the post, user can press the “Update” button to publish the post to the website.

Besides, in the home page, the publisher can modify the maximum display of the post via setting the *Blog pages show at most* parameter in *Settings > Reading* in the dashboard.
4.3.4 Tests

In this section the main test is performed to check the layout of the post proposed by the company.

When a new post is published, it will always appear above the old one. Overall, four categories own the similar layout. Each category has the same style of the post title, content font, and post background.

In the content area, only text can be added if Original or Originalcomment category is chose; both text and image can be added if Custom or Customcomment category is chose; leaving comments is allowed for logged in users if Originalcomment or Customcomment category is chose.

The “Comments to This Post” link exists in Originalcomment and Customcomment category. Users can leave comments after logging in. Likewise, the comments can be replied by others.

The “Edit” link is only available to the author of the post. Once it is selected, the web page will be directed to the dashboard where the post can be modified.

4.4 Facebook Integration

Currently, the company needs to deliver the post to the Facebook wall via Facebook publishing platform manually after publishing it to the website. This means similar works are done twice. Once the company has no time to publish the post to the Facebook wall, visitors cannot get the newest information about New Horizon. However, specific WordPress plugins can simplify the operation.

The Wordbooker plugin, provided by the SteveAtty via WordPress, is used to realize that the post can automatically appear on designated Facebook wall connected to the New Horizon website [38].

There are two actual configurations in the Wordbooker, which consists of two distinct levels: Blog Level and User Level.

The Blog Level configuration fundamentally sets the default for the plugin.

The User Level configuration allows each user to customize their settings according to their requirements.

The plugin allows users to publish the post to different kinds of the walls, including Personal Wall, Application and Page.
4.4.1 Implementation

First of all, the plugin was required to be installed and activated. Afterwards, it can be configured to work with a specific Facebook account. Certainly, a Facebook account was prerequisite to associate with the *Wordbooker*.

In order to implement this functionality, three steps are required as described below:

- **Step 1: Create Facebook Page**
  
  Login in the Facebook account, and type the [http://www.facebook.com/pages/create.php](http://www.facebook.com/pages/create.php) address in a new tab to create a Facebook page. Press *Local Business or Place*, and type the related information. Press the “Get Start” button. Then upload a profile picture, provide some basic information about this page, and choose a unique Facebook web address. Finally, press the “Set Address” button.

  After the Facebook page is created, the “Invite Friends” link is selected to notify others of new Facebook page’s posts.

- **Step 2: Configure php.ini File**
  
  To enable CURL Extension in the php.ini file, search for curl and remove semicolon before it. Restart Apache after php.ini changed.

- **Step 3: Set in the Dashboard**
  
  Initially, *Connect with Facebook* was clicked under *Settings > Wordbooker*. The Facebook account was required to login to link the *Wordbooker*. Inside the relevant dialogue, *Go to App* was clicked. Then *Allow* was clicked to grant *Wordbooker* permissions on a specific Facebook account. After that, the page would automatically return back to the *Wordbooker* options page. Later, *Reload Page* was clicked. After a few seconds, a full *Wordbooker* Options screen would be shown. This meant the basic configuration was complete. Later on, the Blog Level and User Level options could be set to customize the *Wordbooker* for the website.

  Essentially, *Default Publish Post to Facebook* was checked below the Blog Level Settings; specific walls such as *Page* and *Personal*
Wall were selected to publish the post on below the User Level Settings.

4.4.2 Resulting GUI

When the new post is published via WordPress (see Figure 8), it will automatically be displayed on the specific Facebook wall (see Figure 16).

![New Horizon](image)

*Figure 16. Display the post on the Facebook wall*

4.4.3 Manual

In the dashboard, users can set corresponding configurations such as automatic publishing of the post or a specific Facebook wall based on their needs under *Settings >Wordbooker*.

In the *Posts* area, users can see *Wordbooker* options area below the content area, which is default setting users have already configured via the *Wordbooker* configuration interface. Still, users can modify the related configuration in *Wordbooker* options area before publishing the post. For example, if users do not want certain post to appear on the specific Facebook wall, they can uncheck *Publish this post to Facebook*.

If users publish the post to the *Application* automatically, the content in form of the website is displayed in front of the audiences.

If users publish the post to the *Personal Wall*, the content in form of the post is displayed in front of the audiences. This situation is similar to *Page*.

However, for the company, they can create different pages using the same Facebook account. In this way, one Facebook account can represent one

---

2 One kind of the Facebook wall

3 One kind of the Facebook wall

4 One kind of the Facebook wall
company; certain page can represent specific application developed by the company.

The relative Facebook websites are listed below:

Create a page: http://www.facebook.com/pages/create.php
Create an application: http://developers.facebook.com/
Create an account: http://www.facebook.com

4.4.4 Tests

Whether the post published in WordPress automatically appears on the Facebook wall or not is main test in this section.

Once a new post is published by WordPress, it is shown on the Facebook wall and the Facebook News Feed of the user and the friends that the user invited.

When the title of the post or “Read entire article” link is selected by the invited friends, the New Horizon web page will be shown.

4.5 Twitter Integration

The New Horizon website and the company’s Twitter account have not correlated to each other. The company publishes a new content on his website; meanwhile, this information on the Twitter requires the manual operation. This implicates duplications of effort. To simplify the work, the post published on the website should automatically appear on the Twitter.

WP to Twitter plugin is used to make the posts in WordPress tweet to Twitter automatically [39].

4.5.1 Implementation

Download the WP to Twitter plugin at Dashboard > Plugins > Add New, and then activate the plugin.

Two steps are listed below:

- Step 1: Get Twitter Application Information
  
  To configure the WP to Twitter plugin, consumer key, consumer secret, access token, and access token secret were required. The user needs to sign in with his Twitter account in the https://dev.twitter.com/apps address. First, press the “Create a new application” button. Next, fill in the application details: Name, Description, and WebSite. Meanwhile, check the “Yes, I
agree” checkbox to agree the developer rules and input the CAPTCHA. Finally, press the “Create your Twitter application” button.

After creating a new application successfully, click the “Settings” tab, and then select the “Read and Write” radio button in the Application Type area. Then press the “Update this Twitter application’s settings” button.

**Step 2: Set in the Dashboard**

Configure the plugin at Dashboard > Settings > WP to Twitter.

First, in the Connect to Twitter area, copy and paste the consumer key, consumer secret, access token, and access token secret. Then press the “Connect to Twitter” button.

Second, in the Basic Settings area, check the “Update when a post is published” checkbox and “Update when a post is edited” checkbox. Then choose Use WordPress as a URL shortener from the combobox as the short URL service. Press the “Save WP -> Twitter Options” button.

Third, check the associative categories checkbox in the Limit Updating Categories area to tweet the post pertaining to this category.

Finally, in the Get Plug-in Support area, input the email address, check the “I have read the FAQ for this plug-in.” checkbox, and input some messages in the Support Request textarea. Then press the “Send Support Request” button.

Configure the default Twitter setting in WP to Twitter box below the post content.

To make the tweet contain the post title, excerpt of the post content, and the post URL, copy shortcodes #title#, #post#, and #url# from the WP to Twitter box and paste them in the Custom Twitter Post textarea.

### 4.5.2 Resulting GUI

When the company publishes the post (see Figure 8), it will be automatically tweeted to the Twitter. The tweet includes the post’s name, the parts of post’s content, and the post link (see Figure 17).
4.5.3 Manual

The user can create a post and publish it as usual. The post’s information will be tweeted to the user's Twitter.

At Dashboard > Settings > WP to Twitter, some other settings can be done. Described below:

If the user wants to show more or other information, such as the title of the blog, a short excerpt of the post content and the post date, he can copy the shortcodes #blog#, #post# and #date# from the beginning of this plugin setting, and paste them into the textarea, when he sets the Basic Settings.

If the user wants to use other shortener, he can set it in Choose your short URL service area. For example, if he wants to choose the bit.ly shortener, he needs to input his bit.ly Username and API Key below the URL Shortener Account Settings area.

4.5.4 Tests

Whether the post published in WordPress automatically appears on the Twitter or not is main test in this section.

This test is done in a live New Horizon web page. Once a new post is published by WordPress, a tweet is shown on the Twitter. This tweet will come into other followers’ sights.

When the link is selected at the end of the tweet, the published post will be shown in a new tab; when the “New Horizon” link is selected, the New Horizon home page will be shown.

4.6 RSS Support for New Content

Visitors cannot subscribe to the New Horizon website due to no RSS feed. The visitor needs always open the website to pay attention to its update. Once the
website involves RSS and visitors subscribe to it, they will be notified of the latest information by reader or email about New Horizon automatically.

*Google FeedBurner* and WordPress build-in widgets are used to achieve RSS.

### 4.6.1 Implementation

Input the [http://feedburner.google.com/fb/a/myfeeds](http://feedburner.google.com/fb/a/myfeeds) address in the web browser. Copy and paste the blog address into the text area. Press the “Next” button [40].

In the new page, input *New Horizon PC Game* in Feed Title. Press the “Next” button. Then see the Congratulations interface, press the “Next” button. Next, check all the checkbox in claim the feed step, and press the “Next” button. After that, the feed is burned successfully, and then respectively click the “Publicize” tab and the “Chicklet Chooser” tab to reference the relative HTML code connected to RSS.

Go to the WordPress dashboard, in the Appearance > Widgets, drag and drop the Text widgets to the sidebar-right area. Paste the HTML code. Finally, press the “Save” button.

The HTML code was modified a little; for further details please refer to the Appendix D - Mark 1.

### 4.6.2 Resulting GUI

In the *New Horizon* home page with RSS icon on the right side of the page (see Figure 4), once visitors click the icon, the page will automatically switch to the RSS subscription interface.

### 4.6.3 Manual

To subscribe RSS, visitors need to click the “RSS” icon on the right side of the home page. Then they can choose subscription form including email and reader. Certainly, they can view feed XML.

### 4.6.4 Tests

Whether the *New Horizon* website can be subscribed or not via RSS is main test in this section.

This objective is tested in a live *New Horizon* web page. The RSS subscription interface is shown as long as the “RSS” icon is clicked.
Subscription can be done provided that web-based news reader is chosen, or “Get New Horizon PC Game delivered by email” is pressed.

The page writing with XML will be shown, if the “View Feed XML” button is pressed.
5 WordPress Deployment

This chapter presents an approach to transfer the local WordPress to the web server [41].


The detailed steps are described below:

1. Disable all the plugins in the local WordPress.

2. Back up the database from phpMyAdmin. Open a browser, input http://localhost/phpmyadmin/, and select the database called newhorizon_pcg.

   Navigate to the “Export” tab, save the database in SQL format, and press the “Go” button.


4. Open Filezilla and upload all the files of the local WordPress into the web server.

5. Open the wp-config.php file connected to the web server in notepad++ and update the database information, i.e., DB_NAME, DB_USER, DB_PASSWORD, and DB_HOST with the information in a live database account.

6 Discussion and Conclusion

This report presents the process of migrating the current New Horizon website into WordPress as well as the implementation of new features such as submenus, user registration and login, content update, Facebook and Twitter integration, and RSS support for the new content.

The submenu functionality has been achieved by *Multi-level Navigation* plugin. It supports hover sensitivity and keyboard accessibility from the visitors’ perspective. In addition, it is stylized to be consistent with the website design.

The user registration and login functionality has been accomplished by *WP-Members* plugin. With this functionality, a user can register to be a standard user and login his/her account. Furthermore, administrators can upgrade a standard user to an administrator or a publisher; a publisher can create and post his/her own blogs.

The content update functionality has been reached by *Display Post Image* and *WP-PageNavi* plugin. By this functionality, administrators and publishers can set the layout of the post by choosing the existed categories. In addition, posts can be commented by the user who is logged in, and the comments can be replied by others as well.

The rest functionalities related to Facebook and Twitter integration, and RSS support for the new content have been achieved. Once the post published in the website will automatically appear on a specific Facebook wall and a particular Twitter, both connected to New Horizon; RSS is available for visitors that want to be informed about changes to the website.

In this project, a number of technologies are adopted, such as PHP, MySQL, CSS, HTML, WordPress built-in functions and plugins. WordPress uses PHP for operation and MySQL for database. CSS and HTML have been used before by the other programmers from the company. After fully understanding functionalities of these two approaches, we improved the code for adapting to WordPress. WordPress built-in functions and plugins were used to add the features, i.e., submenu. Aimed at the consistency on the website and some special requirements, i.e., adapting the register and login interface to the website, some particular PHP, HTML, and CSS code were embedded into corresponding plugins files.
By means of writing the particular code for IE with same effect in other browsers and executing one function generated to judge the type of the browser, pages look and work the same in multiple browsers regardless of the conflict between IE and WordPress.

Security issues related to WordPress worth to be mentioned. Although hackers cannot login without the correct username and password, potential risk exists due to the unknowable future, i.e., robots crack the password successfully someday. Thus, register and login functionality is better to be optimized to a certain extent. For example, a user is blocked for two hours after he enters wrong password three times. Meanwhile, administrators and user will receive relevant notifications via the email. Then, the user can reset the password with the verification code provided by the email lest the password is stolen; the image verification and math test can be added to the registration process to prevent fake accounts.

As mentioned in Section 2.3, there are disadvantages in using WordPress as an alternative to address the drawbacks associated with the static website. Here, we want to discuss them further and propose some relative solutions.

Due to the default user account called the admin account, which is created with every installation of WordPress, hackers can easily launch a dictionary attack on user’s website to try to guess the password. If the user uses the admin account, the hacker can easily know his username. So it is safer to delete or change the admin account username.

Owing to WordPress default table prefix named wp_, a hacker can exploit users’ website using SQL Injection. This will facilitate him to guess the table names. So it would be better to change it. Before installing WordPress, the user can change it by changing the $table_prefix value in the wp-config.php file. After installing it, the user needs to use the WP Security Scan plugin to do so. In this way, it quite possibly keeps him from doing SQL Injection at all.

Strong password also can prevent the hacker from readily access to administrators' account to install malicious scripts that can potentially cause widespread damage on the website. It is a good idea to include at least eight characters with the minimum of one upper case letter and one figure inside the password.

As a result of no supervision for WordPress plugin developers, anyone can develop and promote a plugin. Some plugins may have security vulnerabilities.
However, we can use those plugins that have a reputable developer who regularly update them behind the scenes. They patch security risks and practical security precautions, which can help thwart or enable users to recover from potential attacks. In that way, users will be exposed to fewer risks.

Given these problems, why we still do not consider another more security blog platform instead of WordPress, such as Joomla!, Drupal?

After numerous reflecting on this question and searching on Google, we found that the problem does not lie so much in choosing which blog platform, but all PHP web CMS are vulnerable to attack. WordPress, Joomla!, and Drupal are all written in PHP and use MySQL database. Correspondingly, malicious code might be inserted into users’ database using SQL injections.

In fact, WordPress has actually enhanced its security causing it to decline the list for the 2009 Trend Mid-Year Statistics [42].

However, one thing should be kept in mind is that users do need to take the proper steps to deal with security issue on a regular basis, since PHP and MySQL are vulnerable. For instance, users need to install security patches periodically.

In short, the website has been adapted to WordPress nicely. Moreover, all new features are integrated into WordPress with the help of few built-in functions, plugins and PHP code. Besides, several proposals have been put forward to secure the website. Thus, the New Horizon website has been highly customizable in WordPress based on company’s requirements.

In retrospect, this project gave us experience on how to handle a big project and how to address problems in work. Initially, we worked out the requirement analysis through discussing with the company. Afterwards, the specific schedule was designed based on requirements captured during the discussion, which contains the essential deadline for each objective. Then we drew up a "blueprint" to implement corresponding task to each goal. During that time, we did individual test on each objective, and subsequently had meeting with company to show the results, listened to their suggestions, and improved them. After all the modules were implemented, we integrated them. Then, overall test was done.

Through this project, we comprehended the difference between a static website and a dynamic website, and how and why a CMS was used as an alternative to convert a static website to a dynamic website. On the other hand, we realized that good communication skills are of great importance, including discussing with the partner, company, and supervisor. Meanwhile, our supervisor Wagner
gave us constructive ideas to deal with the project. Also we got useful advices from the company. Simultaneously, cooperation with each other between partners is also vital. All in all, this project benefits us a lot.
Reference


*Interview with Kai Seidler from the XAMPP project*. [[MySQL AB]], 2006.


Appendix A - Submenu

Mark 1:

```php
<?php
if ( is_user_logged_in() ) {
    include "planet_use.php";
} else {
    the_content();
}
?>
```

Mark 2:

```css
#suckerfishnav {
    background:#121212 url("../multi-level-navigation-plugin/images/suckerfish_black.png") repeat-x;
    font-size:17px;
    font-family:arial,sans-serif;
    width:100%;
}
#suckerfishnav, #suckerfishnav ul {
    float:left;
    list-style:none;
    line-height:40px;
    padding:0;
    margin:0;
    width:100%;
}
#suckerfishnav a {
    display:block;
    color:#dddddd;
```
text-decoration:none;
padding:0px 10px;
}
#suckerfishnav li {
float:left;
padding:0;
}
#suckerfishnav ul {
position:absolute;
left:-999em;
height:auto;
width:131px;
font-weight:normal;
margin:0;
line-height:1;
}
#suckerfishnav li li {
width:129px;
font-weight:bold;
font-family:arial,sans-serif;
}
#suckerfishnav li li a {
padding:4px 10px;
width:110px;
font-size:15px;
color:#dddddd;
}
#suckerfishnav li ul ul {
margin:-21px 0 0 150px;
} #suckerfishnav li li:hover {
background:#121212;
}

#suckerfishnav li ul li:hover a, #suckerfishnav li ul li:hover a, #suckerfishnav li ul li li:hover a, #suckerfishnav li ul li li li:hover a {
color:#ddddd;
}

#suckerfishnav li:hover a, #suckerfishnav li.sfhover a {
color:#ddddd;
}

#suckerfishnav li:hover li a, #suckerfishnav li:hover li a, #suckerfishnav li li:hover li a, #suckerfishnav li li li:hover li a {
color:#ddddd;
}

#suckerfishnav li:hover ul ul, #suckerfishnav li:hover ul ul ul, #suckerfishnav li:hover ul ul ul ul, #suckerfishnav li.sfhover ul ul, #suckerfishnav li.sfhover ul ul ul, #suckerfishnav li.sfhover ul ul ul ul {
left:-999em;
}

left:auto;
background:#444444;
}

#suckerfishnav li:hover, #suckerfishnav li.sfhover {
background:#454545;
}

}
Appendix B - User Registration and Login

Mark 1:

#content-rightside {
width:310px;
margin:0 10px;
background-color: #F4F5F3;
background-image:url(images/news_bg.jpg);
-webkit-border-radius: 8px;
-moz-border-radius: 8px;
border-radius: 8px;
text-decoration:none;
padding-left:20px;
padding-top:2px;
padding-bottom:2px;
}

#sidebar-right
{width:350px;}

Mark 2:

<div id="content-rightside">
<ul>
<?php /* Widgetized sidebar, if you have the plugin installed. */
if ( !function_exists('dynamic_sidebar') || !dynamic_sidebar('Sidebar Right') ) : ?&gt;
&lt;?php endif; ?&gt;
&lt;/ul&gt;
&lt;/div&gt;

Mark 3:

&lt;?php
if ( function_exists('register_sidebar') )

register_sidebar(array(
    'before_widget' => '<li id="%1$s" class="widget %2$s">',
    'after_widget' => '</li>',
    'before_title' => '<h2 class="widgettitle">',
    'after_title' => '</h2>', ));

register_sidebar(array('name'=>'sidebar-right')); ?>
Appendix C - Content Update using WordPress

Mark 1:

<? php if(function_exists('display_post_image')) { ?>
    <a href="<? php the_permalink(); ?>" title="<? php printf( esc_attr__( 'Permalink to %s', 'Test' ), the_title_attribute( 'echo=0' ) ); ?>" rel="bookmark">
        <? php echo display_post_image('width=120px&height=180px&css=alignleft&parent_id='.$post->ID); ?>
    </a>
<? php } ?>

Mark 2:

<?php if ( have_posts() ) : while ( have_posts() ) : the_post();
if (in_category('19'))
    include(TEMPLATEPATH.'/customsinglepost.php');
elseif (in_category('20'))
    include(TEMPLATEPATH.'/originalsinglepost.php');
elseif(in_category('21'))
    include(TEMPLATEPATH.'/customcommentsinglepost.php');
elseif(in_category('22'))
    include(TEMPLATEPATH.'/originalcommentsinglepost.php'); ?>

Mark 3:

<?php wp_pagenavi(); ?>

<?php
function custom_loginlogo() {
    echo '<style type="text/css">
    h1 a {background-image:
    url(' . get_bloginfo('template_directory') . '/images/title_new.png) !important; }
</style>'; 
}
add_action('login_head', 'custom_loginlogo');
?>

Mark 4:

<?php
add_filter( 'login_headerurl', 'custom_loginlogo_url' );
function custom_loginlogo_url($url) {
    return 'http://localhost/wordpress/';
} ?>

Mark 5:

/* Alignment */
.alignleft {
    display: inline;
    float: left;
    margin-right: 1.625em;
}
.alignright {
    display: inline;
    float: right;
    margin-left: 1.625em;
}
.aligncenter {
    clear: both;
    display: block;
Mark 6:

.wp-pagenavi {
    clear:both;
    text-align:center;
    margin-bottom:20px;
}

.wp-pagenavi a, .wp-pagenavi span {
    text-decoration:none;
    padding:3px 5px;
    margin:2px;
    color:rgb(0,175,255);
    font-family:arial, verdana, tahoma, "times new roman", sans-serif;
    font-weight:normal;
Appendix D - RSS Support for New Content

Mark 1:

<a href="http://feeds.feedburner.com/NewHorizonPcGame" title="Subscribe to my feed" rel="alternate" type="application/rss+xml">
<img src="http://www.feedburner.com/fb/images/pub/feed-icon32x32.png" alt="" width="29" style="border:0; height:20px; width:20px; vertical-align:middle;"/></a>

RSS
Presentation of the authors

Yuanwen Zhu

The author is a bachelor in Computer Systems Engineering.

Email: zhuyuanwennina@126.com
Presentation of the authors

Jia Wang

The author is a bachelor in Computer Systems Engineering.

Email: clarawang89@hotmail.com