Webmonitor - an Online Service to Monitor Webpage Changes

Name: Kripanath Aditya
Student Id: 012162004

A Project

in

The Department

of

Computer Science and Engineering



Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Science in Computer Science and Engineering

United International University

Dhaka, Bangladesh

March 2020

© Kripanath Aditya, 2020

Approval Certificate

This project titled "Webmonitor - an Online Service to Monitor Webpage Changes" submitted by Kripanath Aditya, Student ID: 012162004, has been accepted as Satisfactory in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering on 14.03.2020.

Board of Examiners

1.

_____ Supervisor

Prof. Dr. Mohammad Nurul Huda
Professor & Director - MSCSE Program
Department of Computer Science & Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

2.

_____ Examiner

Mohammad Moniruzzaman
Assistant Professor
Department of Computer Science & Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

3.

_____ Ex-Officio

Prof. Dr. Mohammad Nurul Huda
Professor & Director - MSCSE Program
Department of Computer Science & Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

Declaration

This is to certify that the work entitled “Webmonitor - an Online Service to Monitor Webpage Changes" is the outcome of the research carried out by me under the supervision of **Prof. Dr. Mohammad Nurul Huda**.

Kripanath Aditya
Student ID: 012162004
MSCSE Program
Department of Computer Science & Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

In my capacity as supervisor of the candidate’s project, I certify that the above statements are true to the best of my knowledge.

Prof. Dr. Mohammad Nurul Huda
Professor & Director - MSCSE Program
Department of Computer Science & Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

Abstract

Many websites like newspapers, product comparison websites may need to update information regularly based on the webpages of other websites. For that they need to compare new version of the webpage with the old version of the same webpage and find the difference. One of the best ways to do this is to show the two versions side by side in two iframes and highlight the changes in different colors. Currently there are some open source solutions to highlight the changes between 2 different texts but for comparing webpages these programs are not useful at all because for webpages we need to ignore the HTML, CSS, JavaScript code first. This fact inspired me to create this application, which will act as a webservice. It also has subscribing, login and user administration facility. So, multiple users can create accounts and also multiple admins can use this program to manage the user accounts. This program also lists the changes of many websites on a page like a summary after sorting the result. Users can add webpages or remove webpages or entire websites from monitoring. Finally, it has functions which can be used with cron job on a php web-server. When cron job is used, it will periodically monitor the webpages and list the webpages which have changes.

Acknowledgement

I pay my sincere thanks and respect to my supervisor Prof. Dr. Mohammad Nurul Huda, Director - MSCSE Program without whose supervision this project could not be completed. Also, I greatly thank my examiner, professor Mohammad Moniruzzaman, Assistant Professor for examining this project.

For this project I had to dedicate huge amount of time and research effort, passed many sleepless nights thinking about algorithms and trying to discover them. Without the support and sacrifice of my parents, my wife, and other family members it would have been immensely difficult. It will be a great injustice if I do not thank google search engine and the website stack overflow, because without these two research tools completing this project would have been almost impossible.

Table of Contents

I. List of Tables ..... **x**

II. List of Figures ..... **xi**

Introduction **1**

Background and Literature Review **3**

Methodology **5**

3.1 Index Page or Home Page **5**

3.1.1 Banner at the Top 5

3.1.2 Login Section 5

3.1.3 Registration Section 5

3.1.4 Footer at the Bottom 6

3.2 Admin Section **6**

3.2.1 Login.php 6

3.2.2 Registration.php 7

3.2.3 Logout.php 8

3.3 User Section **8**

3.3.1 Login.php 8

3.3.2 Registration.php 9

3.3.3 Logout.php 9

3.4 Admin View **9**

3.5 Font Folder **10**

3.6 Images Folder **10**

3.7 Include Folder **10**

3.7.1 Banner.php 10
3.7.2 Classes.php 11
3.7.2.1 Connect_db 12
3.7.2.2 Get_web_content_with_curl 12
3.7.2.2.1 Properties 12
3.7.2.2.2 Methods 12
3.7.2.2.3 How It Works: 13
3.7.2.3 Find_uncommon_elements_of_two_arrays (Currently Not Used) 16
3.7.2.3.1 Properties 16
3.7.2.3.2 Methods 16
3.7.2.4 Color_changes 17
3.7.2.4.1 Methods 17
3.7.2.5 Get_prebody_postbody 17
3.7.2.5.1 Methods 17
3.7.2.6 Url_relative_to_absolute 17
3.7.2.6.1 Properties 17
3.7.2.6.2 Methods 17
3.7.2.7 Main 17
3.7.2.7.1 Properties 17
3.7.2.7.2 Methods 17
3.7.2.8 Get_row_by_sql 22
3.7.2.8.1 Properties 22
3.7.2.8.2 Methods 22
3.7.2.9 Uncommon_elementKeys_in_array1_and_array2 22
3.7.2.9.1 Methods 22
3.7.3 Footer.php 28

3.7.4 Header.php:.....29
3.7.5 Scroll.js.....29
3.8 Add Webpage Website.....29
3.9 Main View.....30
3.10 Display Folder.....31
3.10.1 Display_side_by_side.php.....31
3.10.2 Display_side_by_side.css.....32
3.11 Site View.....33
3.12 More Options.....33
3.12.1 Resubmit All Pages.....33
3.12.2 Compare Versions and Save Result.....34
3.13 Database Webmonitor.....34
3.13.1 Database Tables.....35
3.13.1.1 Permanent Tables.....35
3.13.1.1.1 Admins.....35
3.13.1.1.2 Users.....35
3.13.1.1.3 Websites.....36
3.13.1.1.4 Webpages.....36
3.13.1.1.5 Versions.....36
3.13.1.1.6 Main_view_results.....36
3.13.1.2 Temporary Tables.....37
3.13.1.2.1 Versions_by_user.....37
3.13.1.2.2 Webpage_details_by_user.....37
3.13.2 Relationship Among the Tables.....38

Experimental Results and Analysis 40

4.1 Results 40

4.1.1 Side by Side View 40

4.1.2 Main View 40

4.2 Analysis 40

4.2.1 Side by Side View 40

4.2.2 Main View 44

Conclusion 46

5.1 Current Status of The Project 46

5.2 Future Works 46

References 48

Appendix A 50

6.1 PHP Function Preg_match_all 50

6.1.1 Description 50

6.1.2 Parameters 50

6.1.2.1 Pattern 50

6.1.2.2 Subject 50

6.1.2.3 Matches 50

6.1.3 Flags 50

6.1.4 Offset 53

6.1.5 Return Values 53

6.2 Password_hash [14] 53

6.2.1 Description 53

6.2.2 Parameters 54
6.2.2.1 Password 54
6.2.2.2 Caution 54
6.2.2.3 Algo 54
6.2.2.4 Options 54
6.2.3 Return Values 54
6.2.4 Password_hash() Example 55

I. List of Tables

Table 1: List of Classes and Methods 12

Table 2: List of MySQL Database Tables 35

II. List of Figures

Figure 1: Update Scanner 3

Figure 2: PageProbe Extension Screenshot 4

Figure 3: The banner of the Website 5

Figure 4: Admin and User Login Options 5

Figure 5: Admin and User Registration Options 6

Figure 6: The Footer 6

Figure 7: Admin Login Screen 6

Figure 8: How to reveal the admin registration screen 7

Figure 9: Admin Sign Up Screen 7

Figure 10: Logout Link for Admin (in Red Font) 8

Figure 11: User Login Screen 8

Figure 12: User Sign Up Screen 9

Figure 13: User Logout Link (With Red Font) 9

Figure 14: Admin View; After Admin Logged in; Admin Can Delete User. 10

Figure 15: Explanation of Regular Expression 14

Figure 16: Explanation of Regular Expression 14

Figure 17: Use of the preg_match_all Function 15

Figure 18: preg_match_all Function Output Example 15

Figure 19: Output of the get_words_array Method 18

Figure 20: Output of $all_changes_array method 19

Figure 21: $parts_devided_by_positions Array Output 20

Figure 22: Output of the $parts_colored_devided_by_positions_new Array 21

Figure 23: Output of the $array1_common_element_positions Array 23

Figure 24: $distance Array Output 24

Figure 25: $array1_common_element_positions Output 25

Figure 26: $array2_common_element_positions Output 26

Figure 27: Illustration of the Distance Pairs of Positions 27

Figure 28: Another example of the distance pairs of positions 28

Figure 29: Code Handling Static Array 28

Figure 30: The footer 28

Figure 31: Header with the Menu and Login Information 29

Figure 32: How to Add Websites in the Database 29

Figure 33: The Main View 31

Figure 34: The Side by Side view 32

Figure 35: The Site View 33

Figure 36: The More Options Page 34

Figure 37: Relationship Among the tables 38

Figure 38: Test1 41

Figure 39: Test 2.1 41

Figure 40: Test 2.2 42

Figure 41: Test 3.1 42

Figure 42: Test 3.2 43

Figure 43: Test 4 43

Figure 44: Test 5 44

Figure 45: Page 1: Webpages are Arranged by Change Time and Domain Name 44

Figure 46: Page 2: Webpages are Arranged by Change Time and Domain Name 45

Chapter 1

Introduction

There are websites where a particular page is updated regularly; may be not all the contents but some of it. For example, the price of a product. Suppose, there is a telecom company. It has different internet data packages, talk time packages. For example, in Bangladesh we have Grameenphone, Teletalk, Banglalink, Robi. They have different talk time packages, internet data packages and also combo packages. Also, often time they have different offers. In Bangladesh many phones support multiple SIMs and often a user has more than two SIM cards. So, already we have the need or want of this product comparison information in the market which can be the base of new business.

Now, if someone may wish to build a website where he can display the latest package rates and offers from different telecom companies and also to compare the packages. Now, how can this be done?

One way would be to hire a few people to check the webpages of these companies and list the information in some databases. You can guess that this is not only costly but also inefficient. What if we desire to follow a market where we have 30 to 40 providers? In such case to do this correctly and efficiently with only human labor is not only difficult but actually almost impossible.

The second way is to create an application or service (online) that will visit the pages mentioned by the users regularly (like once every 24 hours). So, if it detects any change, the page will be saved in a database table as versions. Let us name this service as 'Webmonitor'. So now we have versions of the pages in the database and by default the present version will be compared with the previous version. Text changes on the old version will be highlighted with a color like light blue; text changes on the new version will be highlighted using the color light green. Both versions can be displayed side by side in two iFrames. Let's call it 'side-by-side-view'.

Some other views like ‘side-by-side-view’ will be required. On another views, all the changes of all the webpages will be displayed as a list. The webpages will be grouped under their domains. Let us call it ‘Main-view’. After log in, users will be on this page.

On the ‘index’ or ‘home’ page users will be able to register or login; also, the admins will register or log in via the ‘index page’.

Some other pages will also be needed.

Also, we like to have a good database management system, so that we can add/remove webpages, websites or delete entire websites from the database, have some admin panel to manage the users.

So, this is in sort the design plan. This will be the ‘Webmonitor-version-01’. It can be extended in many ways. But it is better to release it first, get user feedback and requirements and then extend the optional functionalities.

Chapter 2

Background and Literature Review

At this moment there are some free paragraph compare solutions available. For example, Diff-Text [1]. This app compares only plain texts. Another example is Kompare [2]. If you supply two text files, it can compare the differences and highlight the differences. Another example is WinDiff [3]. There is a Wikipedia article [4] about this tool. This tool is from the company Microsoft. It was added in and was part of the visual Studio before Visual Studio 2010. But in practical, they are not much useful in comparing webpage versions.

Currently there is no good free or open source solution to compare webpage versions with good database management system. This factor motivated me to build this web-based solution.

For example, one free webpage version compare solution is Update Scanner [5]. It is a Firefox extension and can be found here: https://addons.mozilla.org/en-US/firefox/addon/update-scanner/. It does not have side by side view, you can view only new version or old version, there is no good database management system and JavaScript is disabled. There is no dashboard to view all recent changes from different websites, in a word, it is very limiting. Here is a screenshot off the extension:

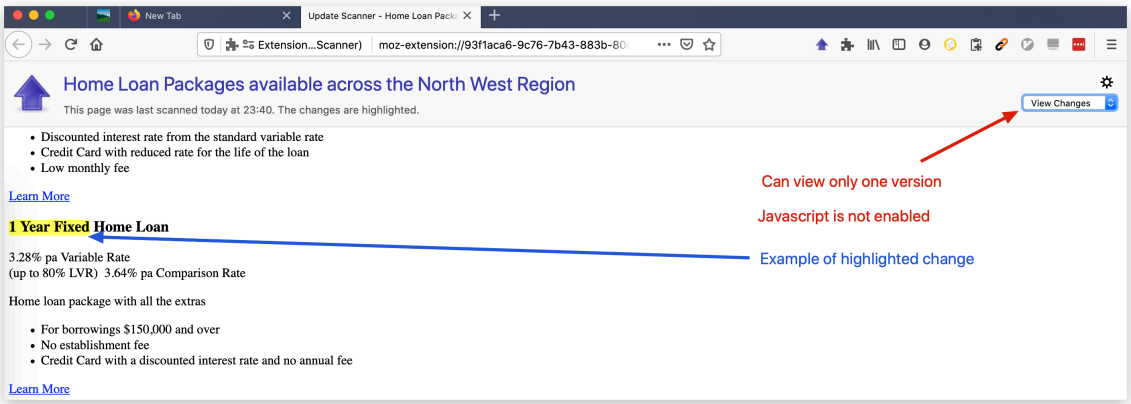


Figure 1: Update Scanner

Another free chrome extension is PageProbe [6]. It only notifies you that there are changes on the particular webpage and it list webpages with any change but there is no facility to display the highlighted changes on a side by side view with the old and the new version. So, it does not solve the objectives mentioned for the project. Here is a screenshot below.

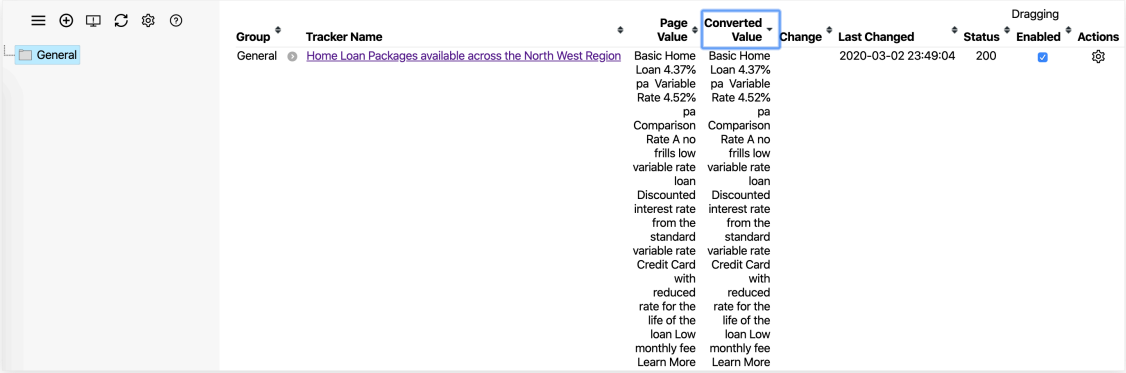


Figure 2: PageProbe Extension Screenshot

I looked for ready-made code on the internet. Several code examples are available to compare only two paragraphs but not webpages. Comparing webpage versions based on text change is much more complex task. You have to ignore any HTML, CSS or JavaScript code in the source code to get the text only and then compare. So, I had to build this entire project from scratch. I only used the scroll.js file's code from a source on the internet which helps scrolling two iFrames simultaneously on side by side view.

Chapter 3

Methodology

3.1 Index Page or Home Page

It is the welcome page. It is controlled by index.php It has 4 sections.

3.1.1 Banner at the Top

It is not clickable on index.php but on the other pages it is clickable and brings the user back on the index.php. Every page has the banner, so it will not be mentioned here again for the other pages.



Figure 3: The banner of the Website

3.1.2 Login Section

In login section the user or the admin can log in using his username or password.

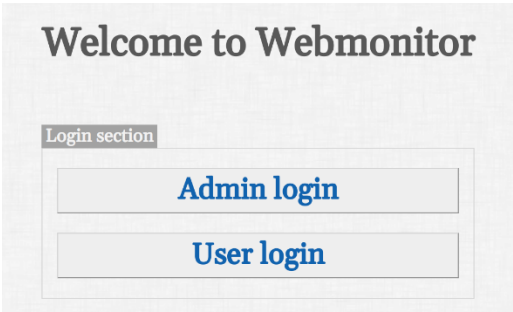


Figure 4: Admin and User Login Options

3.1.3 Registration Section

In registration section the user or the admin can register.

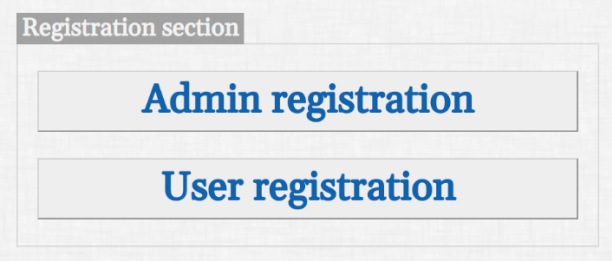


Figure 5: Admin and User Registration Options

3.1.4 Footer at the Bottom

It has copyright information. Every page has the Footer.

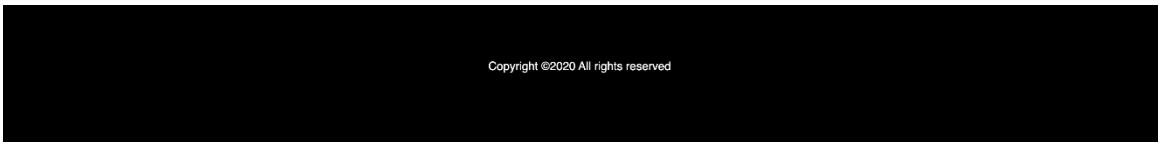


Figure 6: The Footer

3.2 Admin Section

3.2.1 Login.php

It is under the admin-login-registration folder. It helps admin to login the app using admin name and password. It can protect sql injection using prepared statement method. It has data validation facility, like username or password field cannot be empty. There is also a link for user registration to registration.php. After admin is successfully logged in admin is taken to admin_view.php.

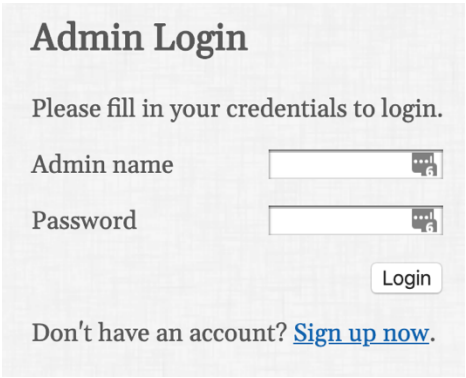


Figure 7: Admin Login Screen

3.2.2 Registration.php

It is under the admin-login-registration folder. It has the facility to register new user but you have to add a password to activate the hidden registration section.

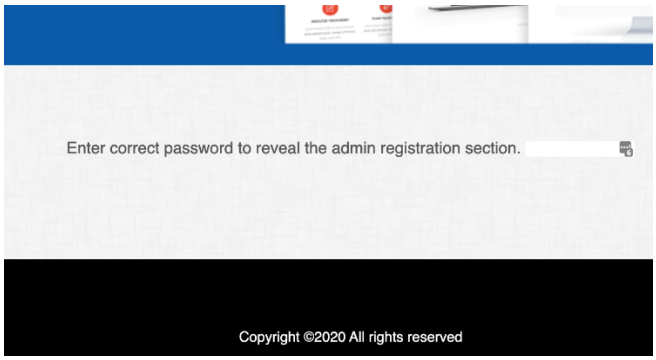


Figure 8: How to reveal the admin registration screen

After supplying the correct password, it will reveal the registration section. Some JavaScript code was used for this functionality.

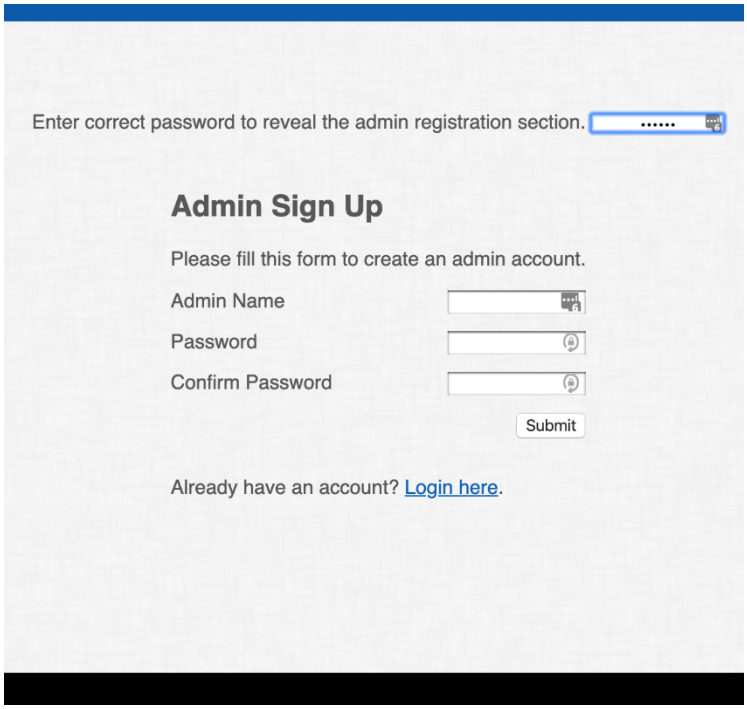


Figure 9: Admin Sign Up Screen

The registration section has 3 fields

1. Admin Name
2. Password
3. Confirm Password

It can protect sql injection using prepared statement method. It has data validation facility, like Admin Name or Password field cannot be empty. There is also a link to admin login view for if the admin already is registered. After login is successful by admin he is taken to admin_view.php.

3.2.3 Logout.php

On admin_view.php admin can delete user. Here there is a link for logout at the top right corner. After clicking the link, the admin is logged out and redirected to admin login view.

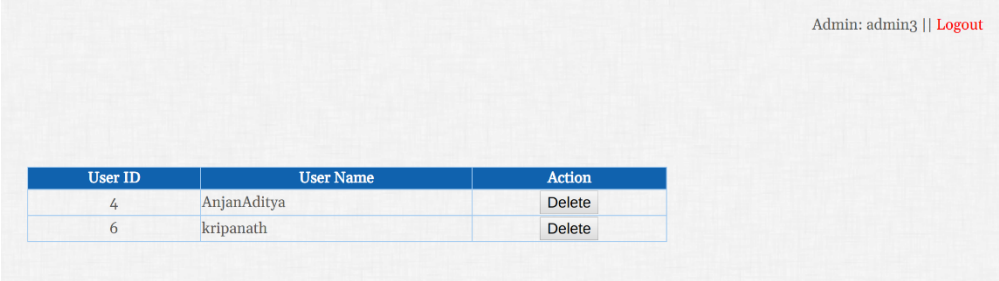


Figure 10: Logout Link for Admin (in Red Font)

3.3 User Section

3.3.1 Login.php

It is under the login-registration folder. It helps user to login the app using username and password. It can protect sql injection using prepared statement method. It has data validation facility, like username or password field cannot be empty. There is also a link for user registration to registration.php.

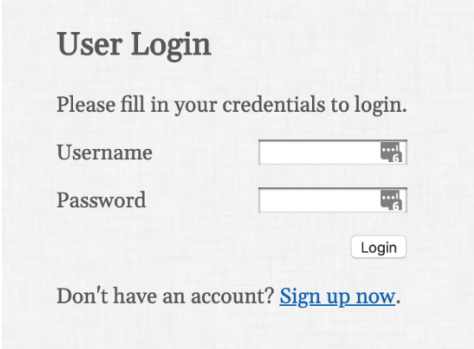


Figure 11: User Login Screen

3.3.2 Registration.php

It is under the login-registration folder. It has the facility to register new user. There are three fields.

1. Username
2. Password
3. Confirm Password

It can protect sql injection using prepared statement method. It has data validation facility, like username or password field cannot be empty.

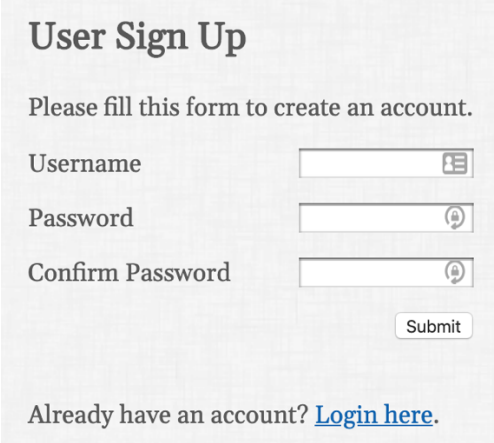


Figure 12: User Sign Up Screen

3.3.3 Logout.php

On most of the pages there is a 'Logout' link above the menu, if the user is already logged in. The user can logout clicking the link and after clicking 'Logout' it redirects to login.php under login-registration folder.



Figure 13: User Logout Link (With Red Font)

3.4 Admin View

It is controlled by admin_view.php. When an admin is logged in he is taken to this page. Here the admin can delete users.

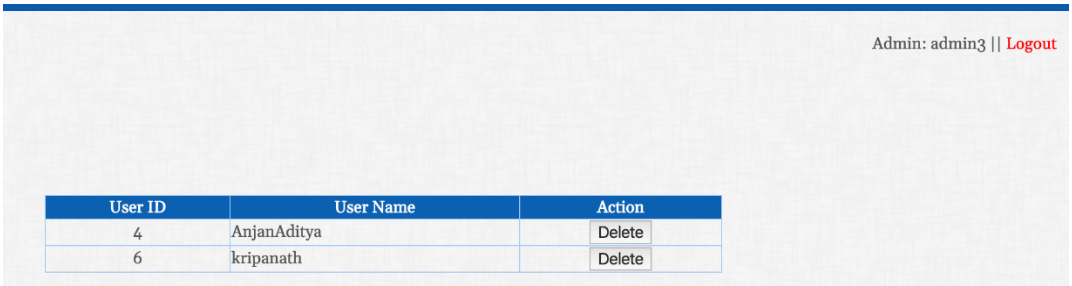


Figure 14: Admin View; After Admin Logged in; Admin Can Delete User.

3.5 Font Folder

The font folder contains the custom font for the application.

3.6 Images Folder

The image folder has all the images used for this application. It has 3 images

1. background.png
2. banner.png
3. logo_webmonitor.png

3.7 Include Folder

Almost on every page of this website there are few common parts like the banner or header. Those parts could be added on each pages individually. But there are some disadvantages for this approach. They are following:

1. They will increase lines of codes.
2. It will take more development time.
3. Maintenance will be difficult as if any part e.g. footer is needed to be updated, we will need to change code on every page where the footer is mentioned.

So, in order to solve this problem this include files in this folder having the repeating code. These codes can be added in any file using the php include statement like: include 'filename';
Now the include folder have the following files. Description will be given below.

3.7.1 Banner.php

It has code for adding the banner. This file is added on almost every page.

3.7.2 Classes.php

Below is the list of classes

Sl	Class name	Methods	Properties
1	Connect_db	__construct	$conn
2	Get_web_content_with_curl	__construct	$webpage_content
3	Get_webpage_text	__construct, string_pos_array_to_words_pos_array, get_body, get_words_from_webpage, array_empty_elements_removed_including_space	$body, $array_strings_offset_and_value
4	Find_uncommon_elements_of_two_arrays	__construct	$array1_uncommon
5	Color_changes	get_parts_devided_by_positions, the_part_up_to_the_1st_position	n/a
6	Get_prebody_postbody	get_pre_body, get_post_body	n/a
7	Url_relative_to_absolute	__construct	$content
8	Main	get_words_array, all_changes_array,	$get_webpage_text_old,

		webmonitor	$get_webpage_text _new
9	Get_row_by_sql	__construct	$column_detail
10	Uncommon_element Keys_in_array1_and _array2	detect_uncommon_elementKeys_in_ array1_and_array2	n/a

Table 1: List of Classes and Methods

3.7.2.1 Connect_db

It connects the database. We need to supply 4 parameters.

1. $host : Name of the host
2. $user_name: Name of the database user
3. $password: Password
4. $database_name: Database name

If all information given above is correct, database will be connected successfully else there will be an error message.

3.7.2.2 Get_web_content_with_curl

Currently not used; but when used it can get the content of a webpage.

Get_webpage_text:

It ignores html, css, javascript and get only the text from the webpage. It has following properties and methods.

3.7.2.2.1 Properties

1. $body
2. $array_strings_offset_and_value

3.7.2.2.2 Methods

1. _construct

2. string_pos_array_to_words_pos_array
3. get_body
4. get_words_from_webpage
5. array_empty_elements_removed_including_space

3.7.2.2.3 How It Works:

- First in the _construct method we use the **get_body** method to get only the body part of the source code of the webpage and remove all comments.

- In the **get_body** method we use preg_match_all function and regular expression to get the part within the <body></body> tags. The regular expression also considers that the starting body tag may have different attributes. We save the extracted part in the **$body** property.

- We use explode and implode [7] php functions to remove all html comments (<!-- *comments* -->) from the body and update **$body** by assigning this result to it.

- We use preg-replace [8] function and regular expression to minify the body. We assign the minified body in **$body** to replace the previous value.

- After getting the body we used **preg_match_all** function on the extracted $body. **Also, it is very important to note that the preg_match_all function uses the flag PREG_OFFSET_CAPTURE.** On PHP manual [9] it says that **'**If this flag is passed, for every occurring match the appendant string offset will also be returned. Note that this changes the value of matches into an array of arrays where every element is an array consisting of the matched string at offset 0 and its string offset into subject at offset 1.**'**

preg_match_all function returns the number of full pattern matches.

preg_match_all function example

The statement used in class.php is as following:

```
preg_match_all('/(<[^<>]+>|<\/[^<>]+>)([^<>]*)/', $body, $result,
PREG_OFFSET_CAPTURE);
```

Below is an illustration of how the above code works-

On regexr.com [10] when the regular expression is used it captures 4 matches for below html code

<h3 class="title">User Contributed Notes 31 notes</h3>

Each match has one full match and other 2 group matches. Here are screenshots of 2 matches among the 4 matches:

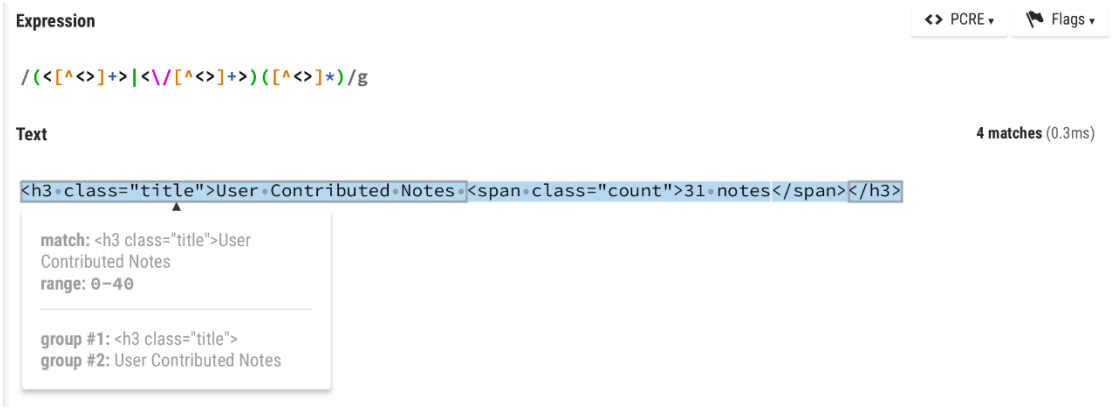


Figure 15: Explanation of Regular Expression

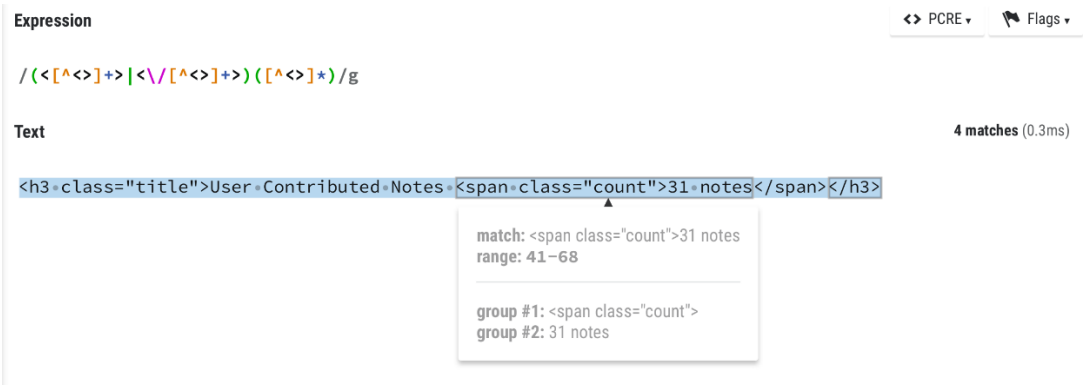


Figure 16: Explanation of Regular Expression

Please note that group#2 captures the text content between html tags.

Also, another thing to note that preg_match_all function capture also positions. Here I am mentioning an example from the PHP manual [9] :

```php
<?php
  $pattern = '<h3 class="title">User Contributed Notes</h3>';
  preg_match_all('/(<[^<>]+>|<\/[^<>]+>)([^<>]*)/', $pattern, $result, PREG_OFFSET_CAPTURE);
  echo'<pre><b><span style = "color: red;">$result</span></b><br/><br/>'; print_r($result); echo'</pre>';
?>
```

Figure 17: Use of the preg_match_all Function

The above example will output:

```
<pre><b><span style = "color: red;">$result</span></b><br/><br/>Array
(
    [0] => Array
        (
            [0] => Array
                (
                    [0] => <h3 class="title">User Contributed Notes
                    [1] => 0
                )

            [1] => Array
                (
                    [0] => </h3>
                    [1] => 40
                )

        )

    [1] => Array
        (
            [0] => Array
                (
                    [0] => <h3 class="title">
                    [1] => 0
                )

            [1] => Array
                (
                    [0] => </h3>
                    [1] => 40
                )

        )

    [2] => Array
        (
            [0] => Array
                (
                    [0] => User Contributed Notes
                    [1] => 18
                )

            [1] => Array
                (
                    [0] =>
                    [1] => 45
                )

        )

)
</pre>
```

Figure 18: preg_match_all Function Output Example

- In the above fig we can see that we are getting some arrays with end tags. Here we got </h3> tag. In the same process we can get </scripts> tags and </style> tags. We wish to remove the css and javascript codes from the $body. So, we use unset [11] function on these arrays.

- Then we remove whitespace, , newline, tab using php functions str_replace [12], trim [13], unset functions.

- **String arrays to word arrays:** We already have string arrays with the strings and their positions. We are going to convert that from string arrays to word arrays with recalculated positions of the words. For that we use the method named **string_pos_array_to_words_pos_array**.

- **string_pos_array_to_words_pos_array**: It takes $string_pos_array as parameter. First array element here is a string and the second element is an array. The $string_pos_array[0] is broken into words array using the php **explode** [14] function. Then position of each word is calculated from initial string position. Then this method returns word and position arrays.

- After applying the **string_pos_array_to_words_pos_array** we can derive the property **$array_strings_offset_and_value**. This array has word positions as keys and words are as values.

- **get_words_from_webpage method:** It returns the property $**array_strings_offset_and_value.**

- **array_empty_elements_removed_including_space:** This method if applied on an array removes elements having only a) only space or b) empty value

3.7.2.3 Find_uncommon_elements_of_two_arrays (Currently Not Used)

3.7.2.3.1 Properties

1. array1_uncommon

3.7.2.3.2 Methods

1. _construct

3.7.2.4 Color_changes

This class takes a string and an array with some positions. The string is broken in those positions.

3.7.2.4.1 Methods

1. get_parts_devided_by_positions : This method takes a string and an array with some positions. The string is broken in those positions.

2. the_part_up_to_the_1st_position: Take a string and a position array and returns substring up to the first position.

3.7.2.5 Get_prebody_postbody

3.7.2.5.1 Methods

1. get_pre_body: Get part of the webpage source up to the body tag (<body>).
2. get_post_body: Get part of the webpage source after the body tag (</body>).

3.7.2.6 Url_relative_to_absolute

3.7.2.6.1 Properties

1. content: This property has the webpage source code.

3.7.2.6.2 Methods

1. _construct: Make absolute URLs from relative URLs.

3.7.2.7 Main

3.7.2.7.1 Properties

1. **get_webpage_text_old**
2. **get_webpage_text_new**

3.7.2.7.2 Methods

1. **get_words_array:** This method returns an array containing two arrays. The first array contains all words in the body of old version as values and the position of each word as its key. The second array contains all words in the body of the new version as values and the position of each word as its key.

2. **all_changes_array:** This method take old version and new version and returns new changes in an array. It has two parameters - $old_page and $new-page.

$old_page: has old version webpage source code.

$new_page: has new version webpage source code.

Get words position and words array: Using get_words_array method we get both webpage versions' words' positions and the words as keys and values. They are assigned to a variable called $webpage_words. Please check the figure:

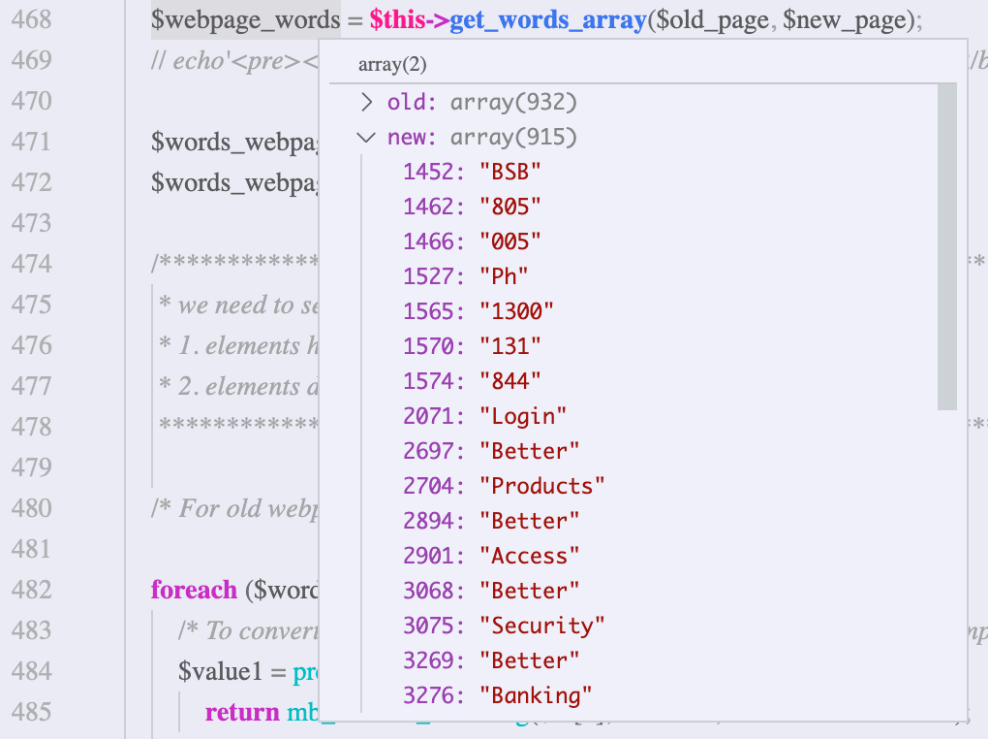


Figure 19: Output of the get_words_array Method

In the figure we can see that $webpage_words array has two array elements named $old and $new. Each of the two arrays has keys as word positions and values as the related words. Then from the array $webpage-words we get the following arrays

$words_webpage_old: keeps old versions keys and positions.

$words_webpage_new: keeps new versions keys and positions.

Next we convert &#[0-9]+ entities to UTF-8 for the above 2 array elements. So, the converted values and other values from the above 2 arrays are saved in the arrays named $array1 and $array2 respectively.

Next we get all uncommon element (or word) keys comparing $array1 and $array2; for this we use the method named detect_uncommon_elementKeys_in_array1_and_array2.

3. webmonitor

Firstly, we get all the changes applying all_changes_array method on the old version ($old_page) and the new version ($new_page). The figure below illustrates the $all_changes array.

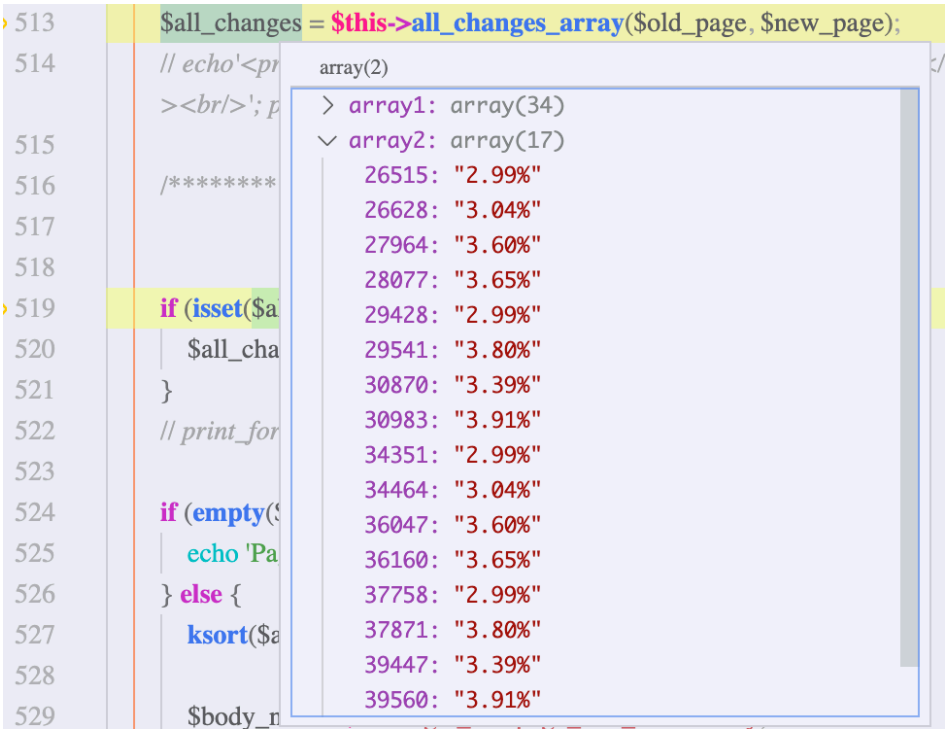


Figure 20: Output of $all_changes_array method

Here we have word positions as keys and the uncommon elements of the two webpage versions as values for old and new webpage versions (in $array1 and $array2).

Now we assign the $array2 to a variable called $all_changes_new.

Then we sort the $all_changes_new according to the key using **ksort** [15] function. Also, we assign the keys of the above array to $all_changes_new_positions as the keys are actually are the positions of the words (the values).

Divide the body in parts on the uncommon words starting positions:

Now we use the method named get_parts_devided_by_positions of the class Color_changes to get an array (named $parts_devided_by_positions) which has parts of the HTML body divided on starting positions of the uncommon words; we have already found out the positions and the positions are the keys of the array. The array $parts_devided_by_positions is illustrated below.



Figure 21: $parts_devided_by_positions Array Output

Also, we need to find out parts up to the 1st position to reconstruct the body. We do that by using the method the_part_up_to_the_1st_position of the Color_changes class.

Code for highlighting the changes:

We create two variables to have the necessary css class as mentioned below:

$color_code_left_part = '';

$color_code_right_part = '';

Get each uncommon word length: We split each element in the previous figure by the first '<' and get the length of the word. Then we use substr_replace [16] php function twice on each element of the previous figure to add the $color_code_left_part and $color_code_right_part. Then assign the new elements to the array $parts_colored_devided_by_positions_new. Below is the figure of that array:

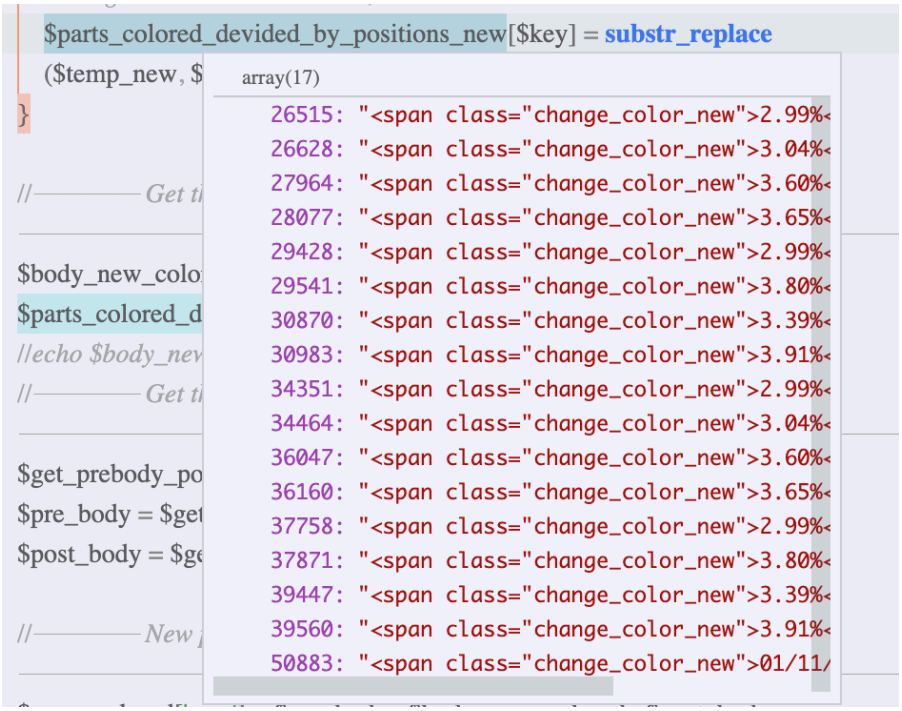


Figure 22: Output of the $parts_colored_devided_by_positions_new Array

Here value of the first element (with the key 26515) is -

2.99% P.A.

Join all the parts of the webpage version:

Now we join

- all the elements before already joined all elements, we add (join) the $the_part_up_to_the_1st_position. we call it $body_new_colored. So, our highlighted <body></body> part of the source code is complete.
- we use get_pre_body method of the Get_prebody_postbody class to get all source code before the <body> tag.
- we use get_post_body method of the Get_prebody_postbody class to get all source code after the </body> tag.

Finally we join the pre-body part, body part and the post body part to construct the whole source code of the new version.

Following similar process, we construct highlighted source code of the old version. We use different highlighting colors for the old version and the new version.

3.7.2.8 Get_row_by_sql

3.7.2.8.1 Properties

1. column_detail

3.7.2.8.2 Methods

1. __construct

After inputting $sql (variable representing sql command), it gets rows of the specific column where $sql condition meet.

3.7.2.9 Uncommon_elementKeys_in_array1_and_array2

Takes $array1 and $array2 as parameter and returns array containing array of keys of uncommon values in array1 and keys of uncommon values in array2.

3.7.2.9.1 Methods

1. **detect_uncommon_elementKeys_in_array1_and_array2**

Parameters: This method is supplied with 2 parameters; $array1 and $array2. $array1 has positions of words and the words as keys and values respectively from the old version. $array2 has positions of words and the words as keys and values respectively from the new version. Or this can be opposite $array1 can have keys and values from new version and $array2 have keys and values from old version.

Calculate relative positions of words: We calculate relative positions of each words with the following formula:

Relative position of a word = (Actual position of the element (containing the word) in the array / total number of elements in the array) x 100%

Thus we find relative positions of all words in both arrays.

Trim whitespace: Then we trim whitespace from each elements of both arrays.

Find common element keys: If $array1 values exist in $array2 find the keys of the common element in $array2 and array1 and do this step for all elements. Thus we create the new array called $common_elements_in_array1[$key1] where $key1 is the actual word key from $array1.

Find common elements with no duplicates: We use array_unique [17] function to find common elements with no duplicates between $array1 and $array2.

Common element positions: The we find out common element positions in array1 and array2. Below is the illustration of the array $array1_common_element_positions. The element 'Login' has 4 positions. First position is 2071th as it is the 96th element in array1. Relative position of the element in the array is 0.858.

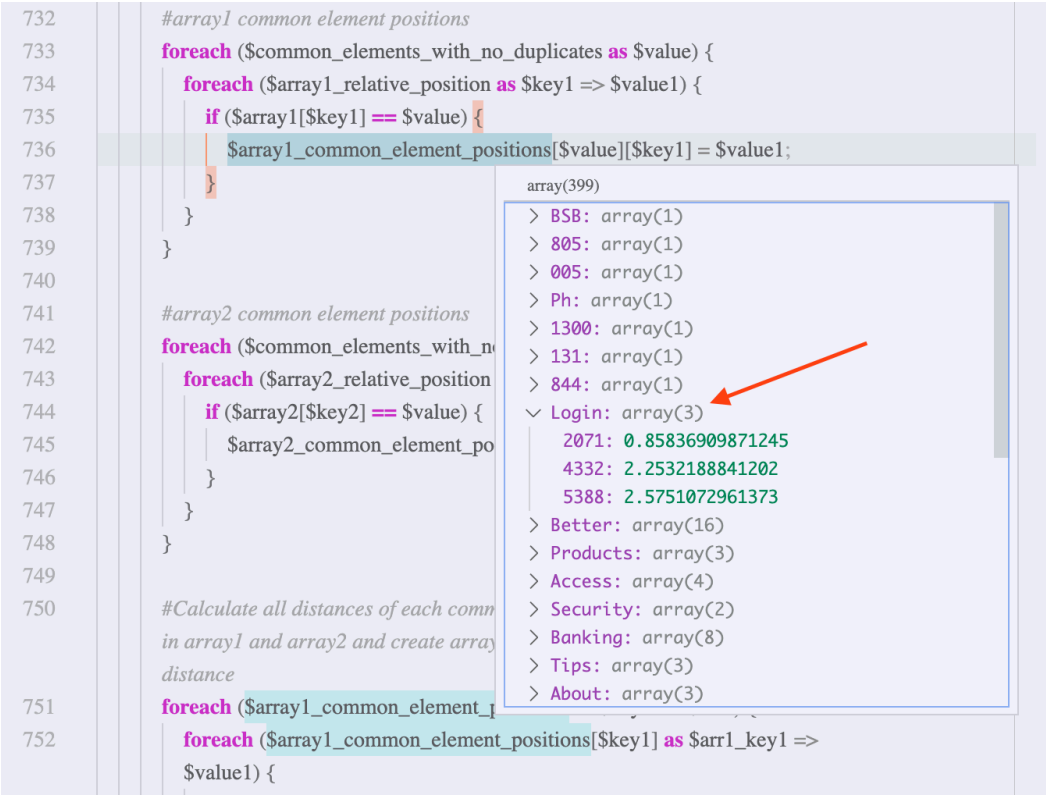


Figure 23: Output of the $array1_common_element_positions Array

Calculate all distances of each common element between its position in array1 and array2 and create array like array1_key - array2_key – distance. On the figure below please note the $distance array. For the element or word 'Login'. First element value is '2071-

2071:0.015947841178264'. The first '2071' is the position of 'Login' in array1, the second '2071' is the position of 'Login' in array2. The number after the : is the difference between relative positions of the 2071th element in array1 and 2071th element in array2.



Figure 24: $distance Array Output

Sort $distance array: Next we sort the distances from lowest to highest. In this case the $distance array figure would be same as above because it was already sorted. 'Login' was in array1 i.e., 1st version 3 times as the following figure (element no 2071, 4332, 5388):



Figure 25: $array1_common_element_positions Output

and it was in array2 i.e., 2nd version 3 times as the following figure (element no 2071, 4309, 5365):



Figure 26: $array2_common_element_positions Output

Now we need to understand which 'Login' on the 1st version is related to which 'Login' in the 2nd version. For that we needed to sort the distances.

From $distance array derive the common elements of both array1 and array2: Please note the following figure. The first foreach loop iterates for each words. In the figure it illustrates, what happens in case of the word 'Login'. We had several iterations of the foreach loop before we reach 'login' in this case. In the figure 1st column of keys are from array1 and 2nd column of keys are from array2.

Now the first element in first column is 2071 is from the first element. Now we declare a static empty array called $array1_common_keys. According to the figure the word 'Login' appeared in 2071th position in array1 and also appeared in 2071th position in array1 and these 2 positions have minimum relative distances. So these 2 'Login's are related. 2nd 'Login' appeared in 4332th position in 1st array and in 4309th position in 2nd array. As 4332th position has minimum distance with 4309th position but not 5365 or 2071th position, they are related. Now using next few lines of code we save the elements

in static array if they already does not exist in that static array. This we keep only the min distance pair values for array1 and array2.



Figure 27: Illustration of the Distance Pairs of Positions

Also, if we had on extra element 'Login' in one array (which means new word added or deleted) then it will not be added in the static common array, because

- The element (of the new position) will have highest distance with other elements and its pair will be at the bottom. And,
- The other value of this pair will be already added in the static array, it will fail to satisfy the statement at line 781; so, it will not be added in the static array. That means it is not considered common element. Below is another figure:

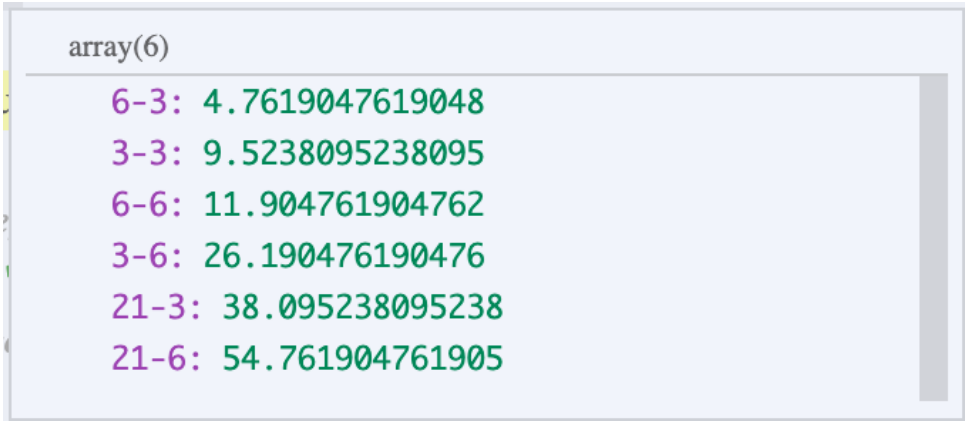


Figure 28: Another example of the distance pairs of positions

21th position is not added in the static array as the pair is 21-3 and 3 would be already added according to the following statements:

```
if (!in_array($keys_exploded[0], $array1_common_keys) &&
!in_array($keys_exploded[1], $array2_common_keys)) {
    $array1_common_keys[] = $keys_exploded[0];
    $array2_common_keys[] = $keys_exploded[1];
}
```

Figure 29: Code Handling Static Array

Get uncommon values: As we now know the common keys, we can easily get the uncommon keys using array_diff [18] function. For example, comparing array1 keys and the common array keys we can get the uncommon keys of array1. Then we can get the uncommon values using the keys.

3.7.3 Footer.php

If this file is included, it adds footer on a particular page.



Figure 30: The footer

3.7.4 Header.php:

It is included almost on all pages. It has the following parts:

1. Login information
2. Menu

It uses condition to show above mentioned parts on different pages. For example, before login, the login information will not be shown.

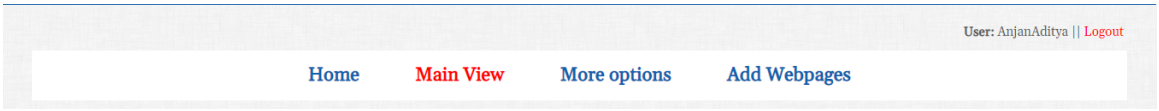


Figure 31: Header with the Menu and Login Information

3.7.5 Scroll.js

It is a JavaScript file. It is an off the shelf file [19]. The display_side_by_side.php uses this file for scrolling both iframe on side by side view at the same time.

3.8 Add Webpage Website

Controlled by add_webpage_website.php. You can visit the page by clicking the 'Add Webpages' link in the menu. Multiple webpages can be added at once. The webpages must be added on the text area line by line. Then you have to click the 'SUBMIT' button.

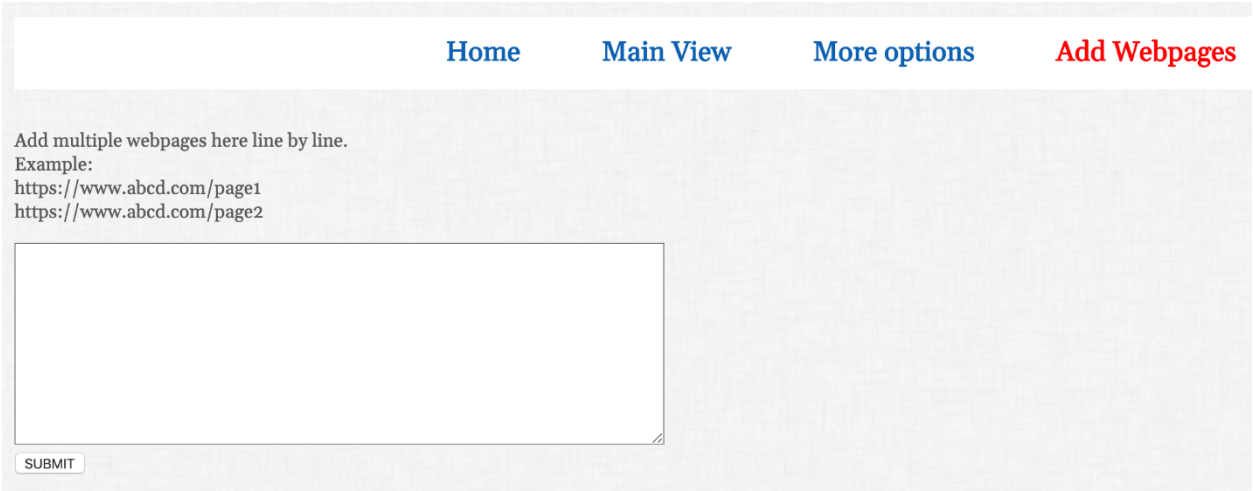


Figure 32: How to Add Websites in the Database

3.9 Main View

It is one of the most important pages of the application because it shows all the changes. The results are organized in a table. The table results are grouped under domain names. And the recent results are shown at the top. Any domain group can have multiple results. For each page the recent two versions are compared for text changes. The results are saved in a database table called main_view_results. From that table the main view table is generated very quickly. How the process works will be discussed in detail in other sections.

The main view page table has the following columns.

1. **Pages:** Shows the webpage link part after the domain name.
2. **Old Version time:** The latest two versions are compared. In this column the older version time is mentioned.
3. **New Version time:** The latest two versions are compared. In this column the new version time is mentioned.
4. **Change:** This column has change link for the webpages mentioned. Each link directs to a page called 'display_side_by_side'. Here the latest two versions of the webpage is displayed in two iframes side by side and the changes are highlighted.

Finally, there is **pagination**. The pagination is just below the menu and above the table. If the number of table rows (or number of change links) are more than a particular number of pages there are new pages links and these pages are generated are dynamically.

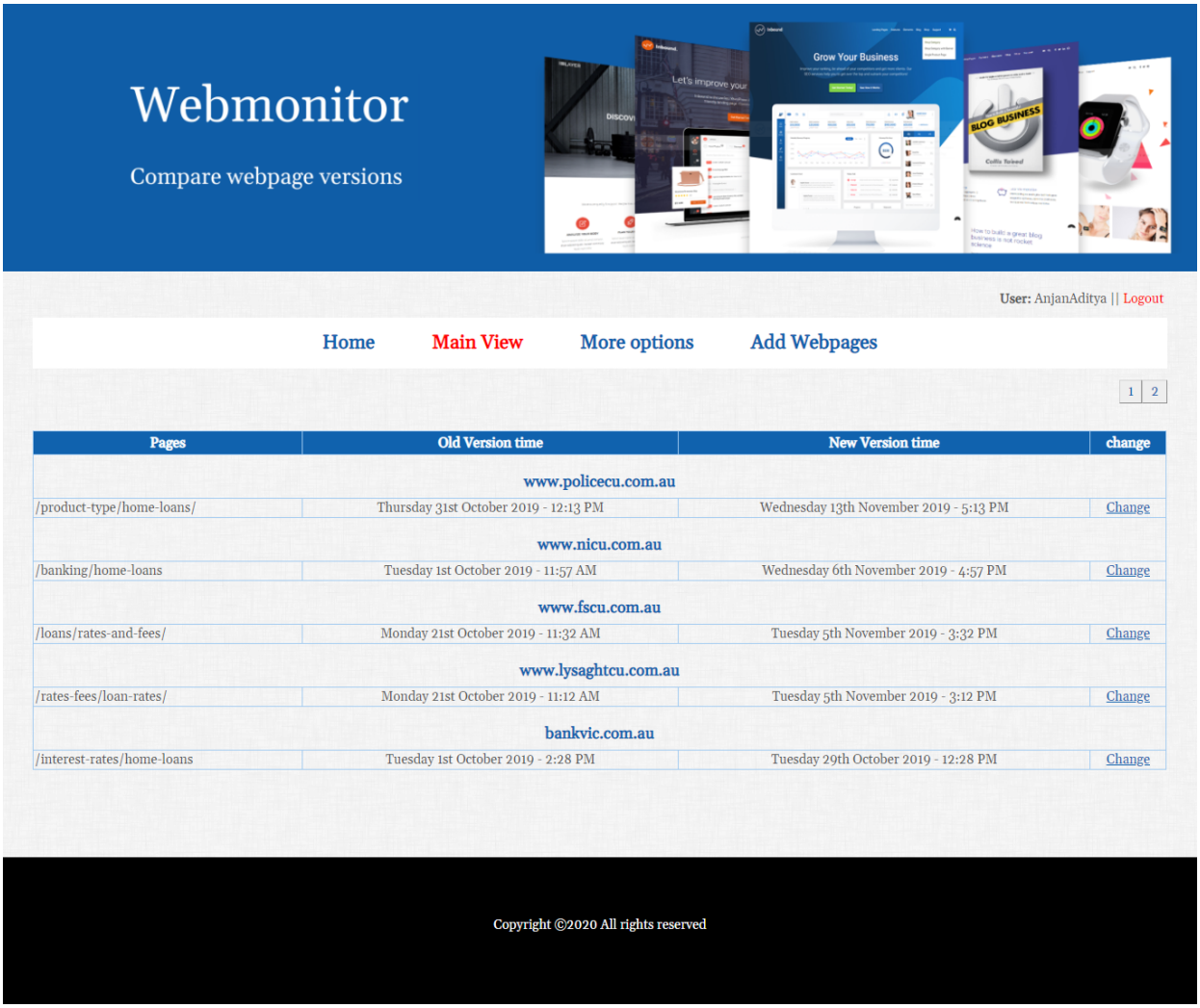


Figure 33: The Main View

3.10 Display Folder

3.10.1 Display_side_by_side.php

It creates the side by side view. It has the following parts:

1. **Left iframe**: It has the old version of the webpage. It can be noted again that the two most recent versions for the particular webpage are compared.

2. **Right iframe:** It has the new version of the webpage.

3. **Change highlights:** All the text changes on the old and new versions are highlighted. For the old version one color is used. For the new version another highlight color is used.

4. **Old version date:** At the top-right of the left iframe old version date is mentioned.

5. **New version date**: On the top-left of the right iframe new version date is mentioned.

6. **Webpage link**: On the top right of the page there is a link to the actual webpage versions of which we are comparing here.

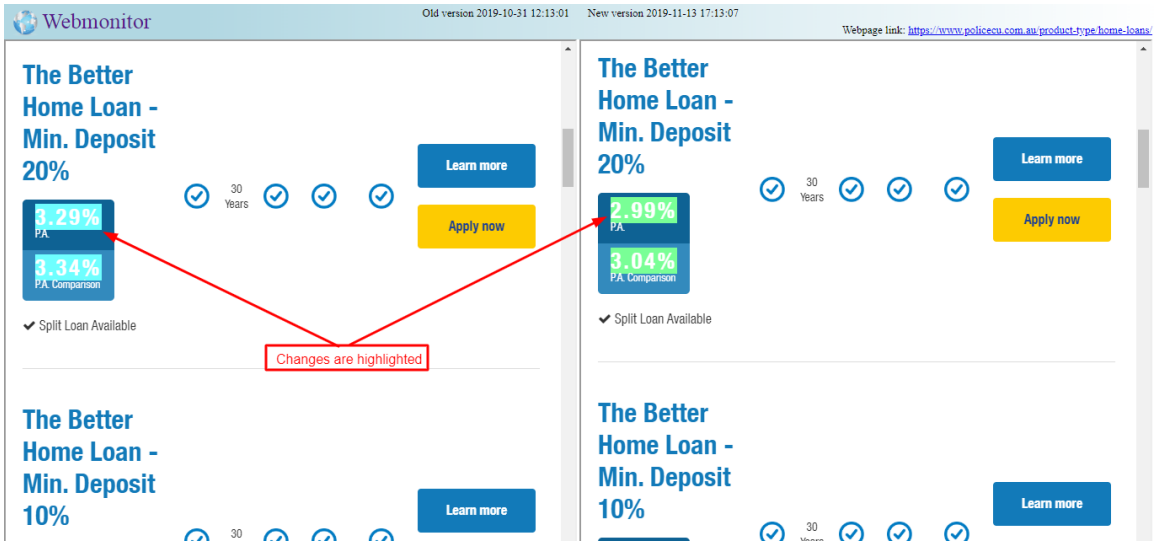


Figure 34: The Side by Side view

3.10.2 Display_side_by_side.css

This is the CSS file that controls the look of the side by side view generated by display_side_by_side.php.

1. **page_old.php:** This php file generates the page of the left iframe.
2. **page_new.php:** This php file generates the page of the right iframe.

3.11 Site View

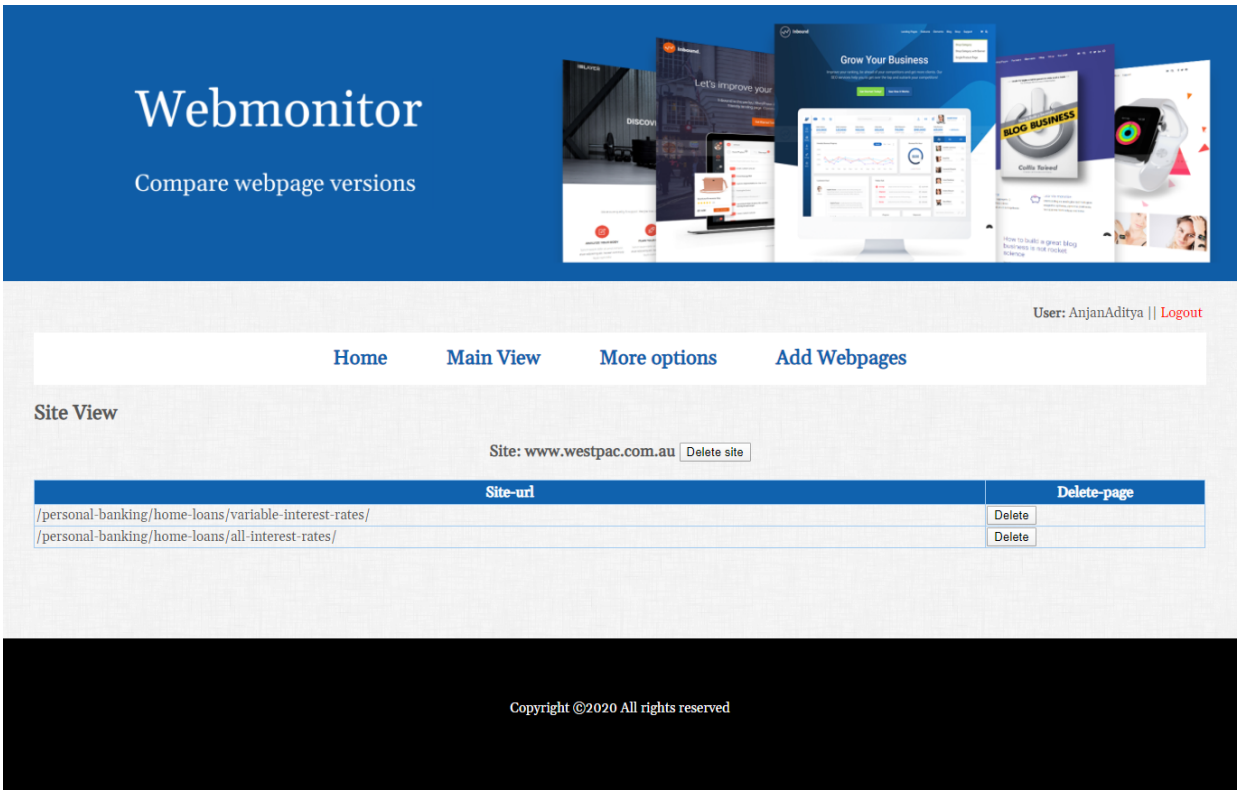


Figure 35: The Site View

3.12 More Options

On this page we have two options:

3.12.1 Resubmit All Pages

Here it is manual way for the development purposes. In final website this will be a function and will be handled by cron job in the php server. The task will be done periodically like once in every 24 hours.

We submit links with add_webpage_website.php. All the links are saved in the database. For each webpage 5 versions are kept and these versions are the latest versions. When we click the 'Resubmit all pages' link it fetches new versions of each webpages in the database and the oldest version is removed (if the number of versions is more than 5) to add the new version.

3.12.2 Compare Versions and Save Result

This function is controlled by compare_versions_and_save_result.php. For each webpage we have 5 versions. Latest 2 versions are compared. The method of comparison is only text by text. Differences between the old version and new version is found out and saved in a particular database table to show result later instantly.

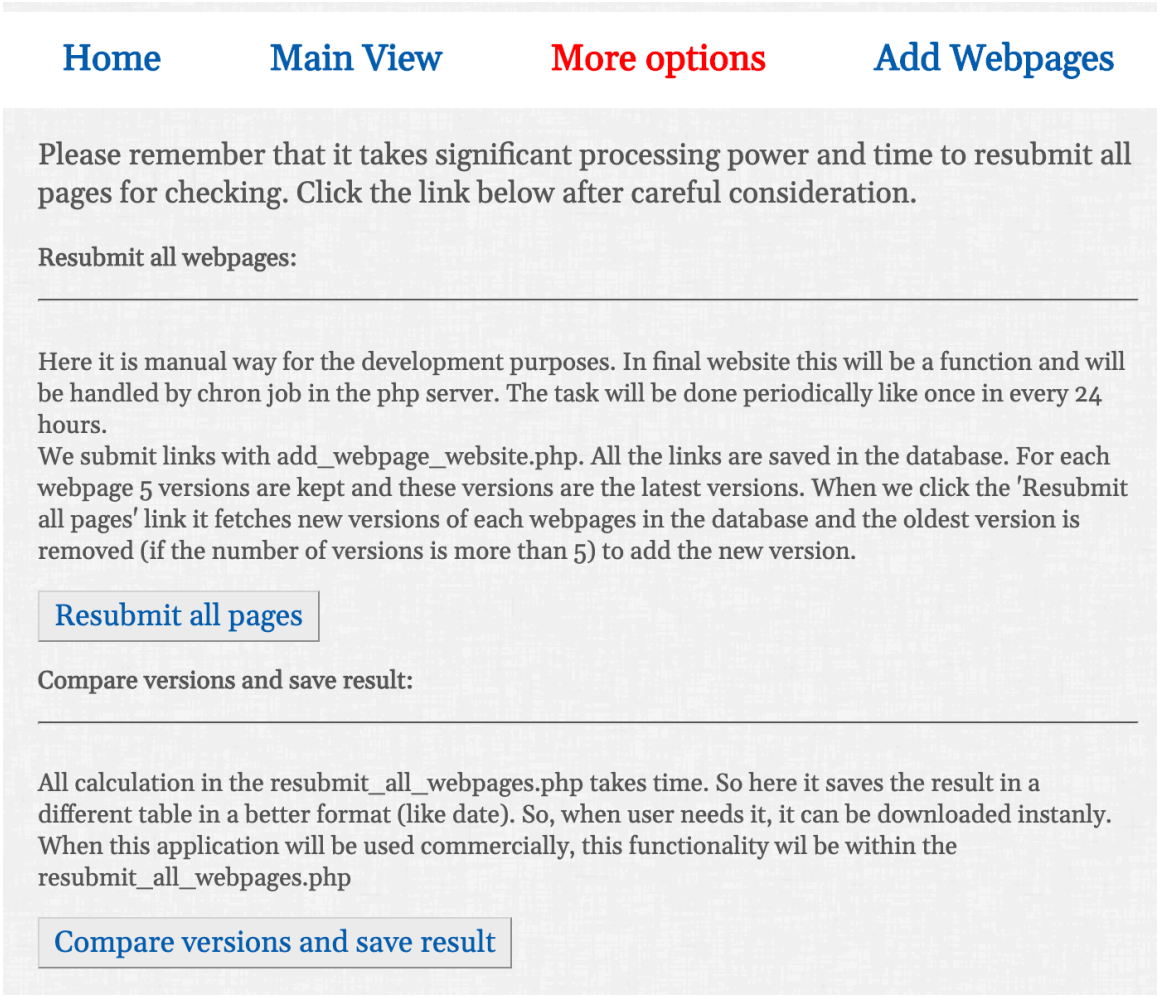


Figure 36: The More Options Page

3.13 Database Webmonitor

This application uses MYSQL database system. The name of the database is called 'webmonitor'. It has the following tables:

3.13.1 Database Tables

Here is the list of database tables:

Sl No	Table Name	Table type
1	admins	Permanent
2	users	
3	websites	
4	webpages	
5	versions	
6	main_view_results	
7	versions_by_user	Temporary
8	webpage_details_by_user	

Table 2: List of MySQL Database Tables

3.13.1.1 Permanent Tables

3.13.1.1.1 Admins

The 'admins' table stores information of the admins. It has the following columns

1. **admin_id:** It is the automated unique id number for each admin.
2. **admin_name:** This column is used to store admin names.
3. **admin_pass:** This column is used to store password created by admins. When an admin register, he/she adds the password on the form field. Then a php function called password_hash() [20] is used. This function creates a new password hash using a strong one-way hashing algorithm and then it is saved in the admin table.

3.13.1.1.2 Users

The 'users' table stores user information. It has the following columns:

1. **user_id:** It is the automated unique id number for each user.
2. **user_name:** This column is used to store user names.
3. **users_pass:** This column is used to store password created by users. When an user register, he/she adds the password on the form field. Then a php function called password_hash() is used. This function creates a new password hash using a strong one-way hashing algorithm and then it is saved in the user table.

3.13.1.1.3 Websites

1. **website_id:** It is the automated unique id number for each website.
2. **website_scheme:** It stores the http or https part of the website address.
3. **website_host:** It stores website host name. It does not include http/https part. For example: www.imb.com.au
4. **user_id:** It is a foreign key column. This column links the **users** table with the **websites** table. user_id is the primary key in the **users** table.

3.13.1.1.4 Webpages

1. **webpage_id:** It is the automated unique id number for each webpage.
2. **webpage_path:** It is the part of the web address after the domain name. For example: /loans-home-and-investment-loans.html
3. **website_id:** It is a foreign key column. This column links the **websites** table with the **webpages** table. website_id is the primary key in the **websites** table.

3.13.1.1.5 Versions

1. **version_id:** It is the automated unique id number for each version.
2. **version_date:** It is the date when the web page version is crawled
3. **version-detail:** Source code of the version. It may include html, css, javascript code.
4. **webpage_id:** It is a foreign key column. This column links the **webpages** table with the **versions** table. webpage_id is the primary key in the **webpages** table.

3.13.1.1.6 Main_view_results

The main purpose of the table is to store data that we wish to show in main_view.php page. So, we can show the data almost instantly without any complex calculation.

1. **result_id:** It is the automated unique id number for each result.
2. **site_id:** site_id column value from website table where website name is same

3. **site:** website-host column value from website table.

4. **Page:** webpage_path column value from the webpages table.

5. **old_version_date:** For the particular webpage latest two versions are compared and old_version_date column has the date of the old version (from versions table version_date column).

6. **new_version_date:** For the particular webpage latest two versions are compared and new_version_date column has the date of the new version (from versions table version_date column).

7. **old_id:** old_id column is generated from the version_id column of the version table. Latest two versions of a particular webpage is taken and the old version's id is old_id

8. **new_id:** new_id column is generated from the version_id column of the version table. Latest two versions of a particular webpage is taken and the new version's id is the new_id

9. **user_id:** user_id column of this table is generated from the user_id column of the users table

3.13.1.2 Temporary Tables

Few temporary tables were created and after calculation they were deleted. Temporary tables on computer hard-disk are required to be created because other wise if there is too much data generated, the RAM of the computer can become full.

3.13.1.2.1 Versions_by_user

In the compare_versions_and_save_result.php page a temporary table named $versions_by_user_table was created and after necessary calculation again it was deleted.

3.13.1.2.2 Webpage_details_by_user

In the resubmit_all_webpages.php page a temporary table named $webpage_details_by_user_table was created and after necessary calculation again it was deleted.

3.13.2 Relationship Among the Tables

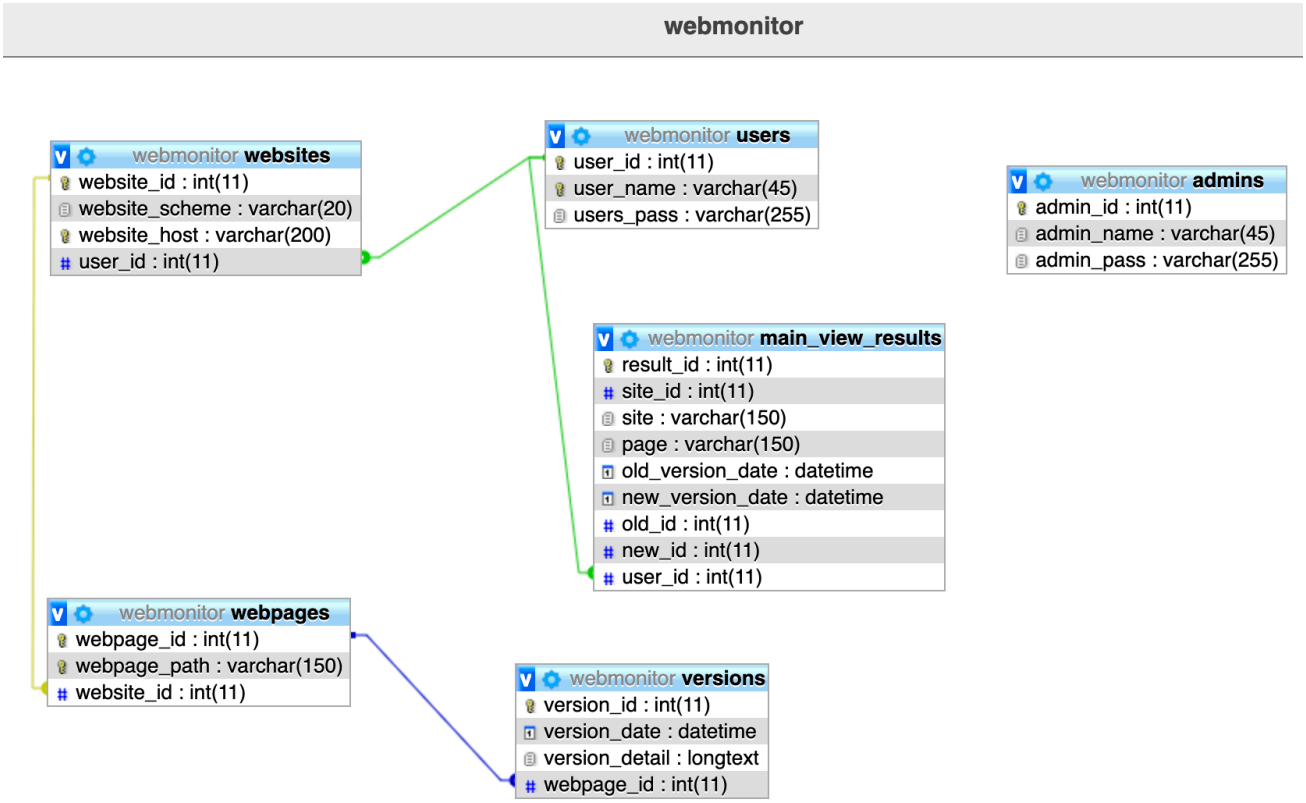


Figure 37: Relationship Among the tables

Here is a screenshot of the relationship among the tables from the phpmyadmin designer view.

admins table on the top right is an independent table. It keeps all the registration data of the admins.

The **users** table keeps all registration information of the users. It has one to many relationships with the **websites** table. Here user_id is the primary key in the **users** table and user_id is the foreign key in the **websites** table. So, one user can have many websites.

The **websites** table keeps all information of all **websites** by all users. With the foreign key column user_id we can understand which row or website belongs to which user. The **websites** table has one to many relationship with the **webpages** table. The website_id column is the primary key in the **websites** table and website_id is the foreign key in the **webpages** table. One website can have many webpages.

The **versions** table keep all versions of the **webpages**. It has the webpage_id foreign key, also **webpages** table has the webpage_id primary key. So, the **webpages** table has one to many relationships with the **versions** table.

We have another table named **main_view_results.** In this table we save data for main_view.php after long calculation. The purpose of the table is only to show data quickly when user requires and views main_view.php. This table has data of all users. But we need to delete particular user data when we want to delete the user from the database. To make that possible we have a one to many relationship with the user table where the user table is the one side and the **main_view_results** table is the many side because one user can have many records. Here user_id column is the primary key in the user table and user_id column is the foreign key in the **main_view_results** table.

Chapter 4

Experimental Results and Analysis

4.1 Results

4.1.1 Side by Side View

Generally, the detection of changes between two webpage versions is very good. In some, cases if there is too much repeating information in both version and we have changes, we may have some detection problem. In this project we have considered two aspects. The first aspect is the content of the paragraph and the second aspect is the relative position of the individual words. But again, it also depends on human interpretation because the changes in repeating information sometimes become confusing even to humans. For example, in one case, one paragraph may have same heading but change in content. In another case, content may be same but heading is different. Again, there may be multiple mentions of this paragraph on the same webpage with slight change in the heading or content. In such case, it becomes difficult to say sometimes for human also, which paragraph of the first version relates to which paragraph of the second version.

4.1.2 Main View

On the main view the application is showing the results for the webpages of which versions' changes are detected. Moreover, the result is sorted by the time of the change for the latest change of a webpage and by webpage name. Also there is pagination facility, so that not all pages are not shown in a single page; only fixed number of pages are shown per page.

4.2 Analysis

4.2.1 Side by Side View

Below are some screenshots of comparing some webpage versions. On the left side is the old version and on the right side is the new version. Changes on the old version are highlighted with turquoise (light blue) color. Changes on the new version are highlighted with light green color.

Test1: Webpage https://www.nicu.com.au/banking/home-loans

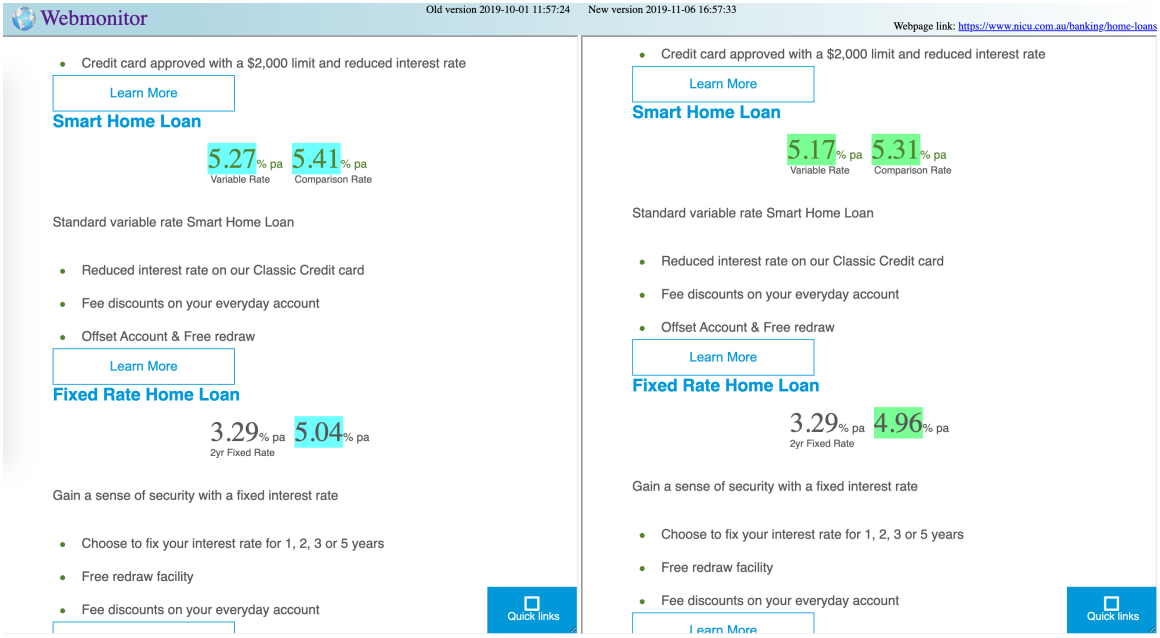


Figure 38: Test1

Test 2.1 - Webpage https://www.fscu.com.au/loans/rates-and-fees

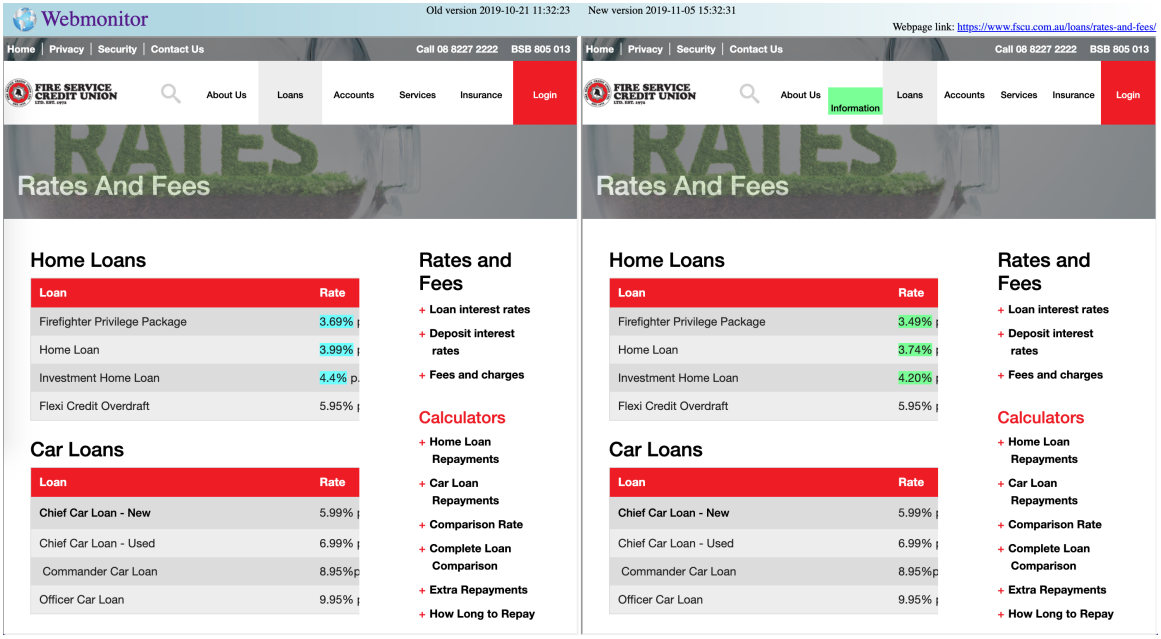


Figure 39: Test 2.1

Test 2.2 - Webpage https://www.fscu.com.au/loans/rates-and-fees

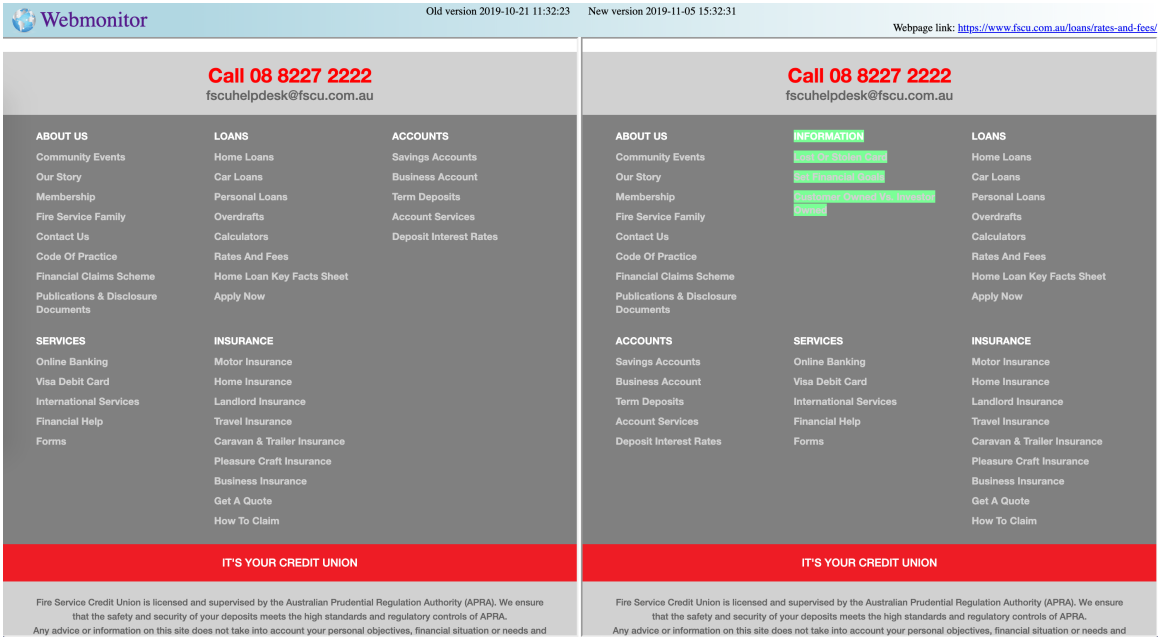


Figure 40: Test 2.2

Test 3.1 - Webpage https://www.lysaghtcu.com.au/rates-fees/loan-rates

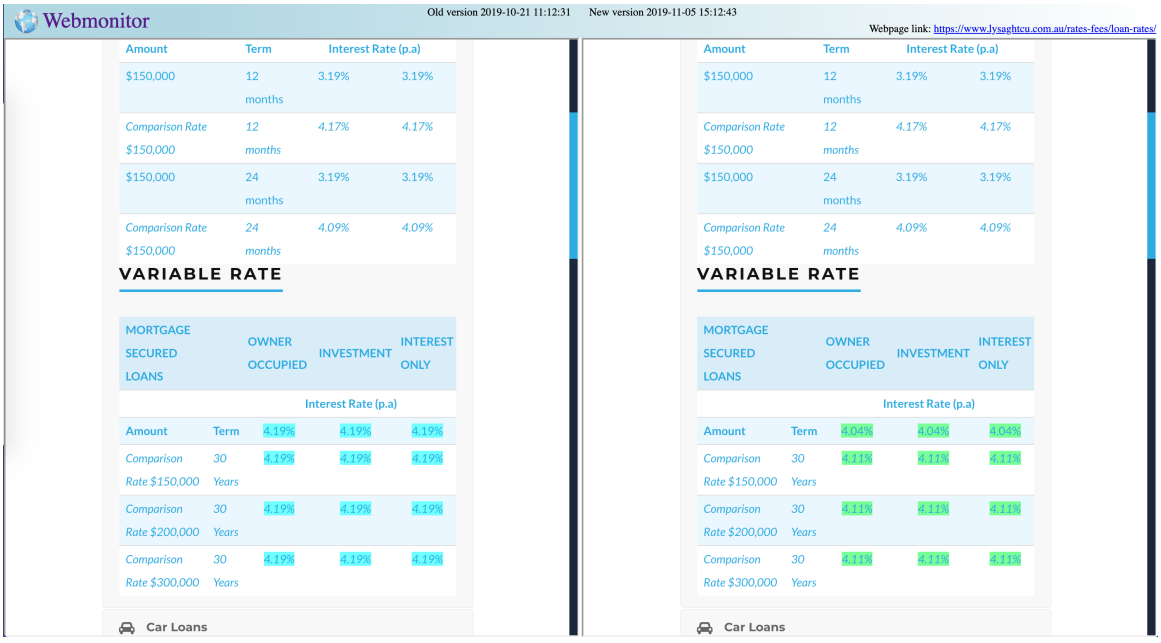


Figure 41: Test 3.1

Test 3.2 – Webpage https://www.lysaghtcu.com.au/rates-fees/loan-rates

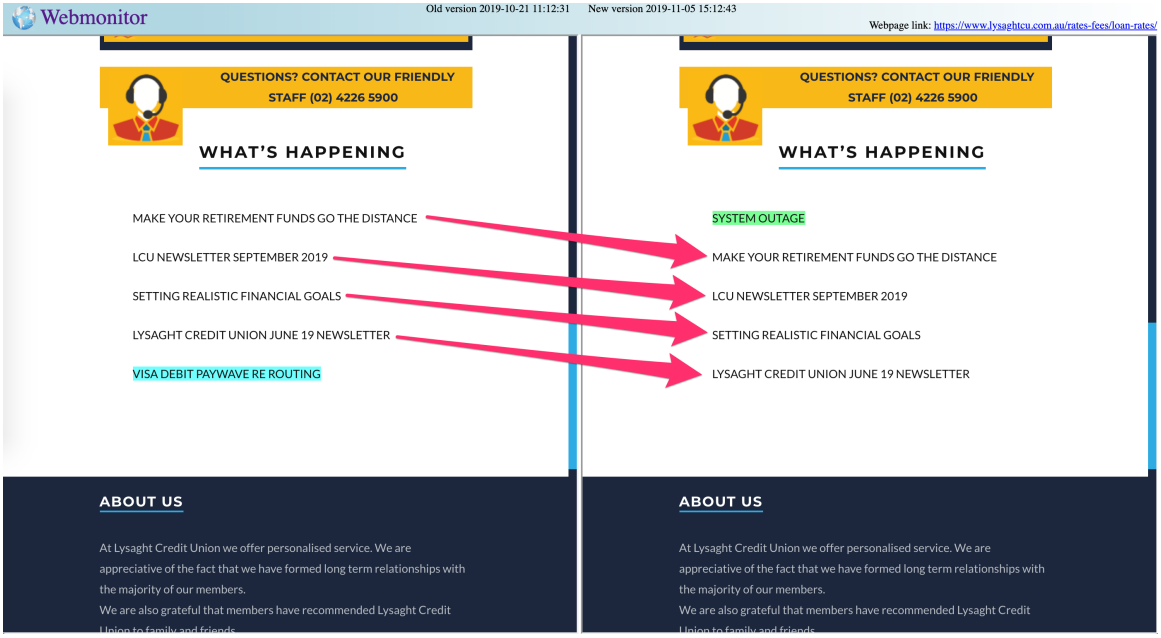


Figure 42: Test 3.2

Test 4 - https://www.westpac.com.au/personal-banking/home-loans/all-interest-rates/

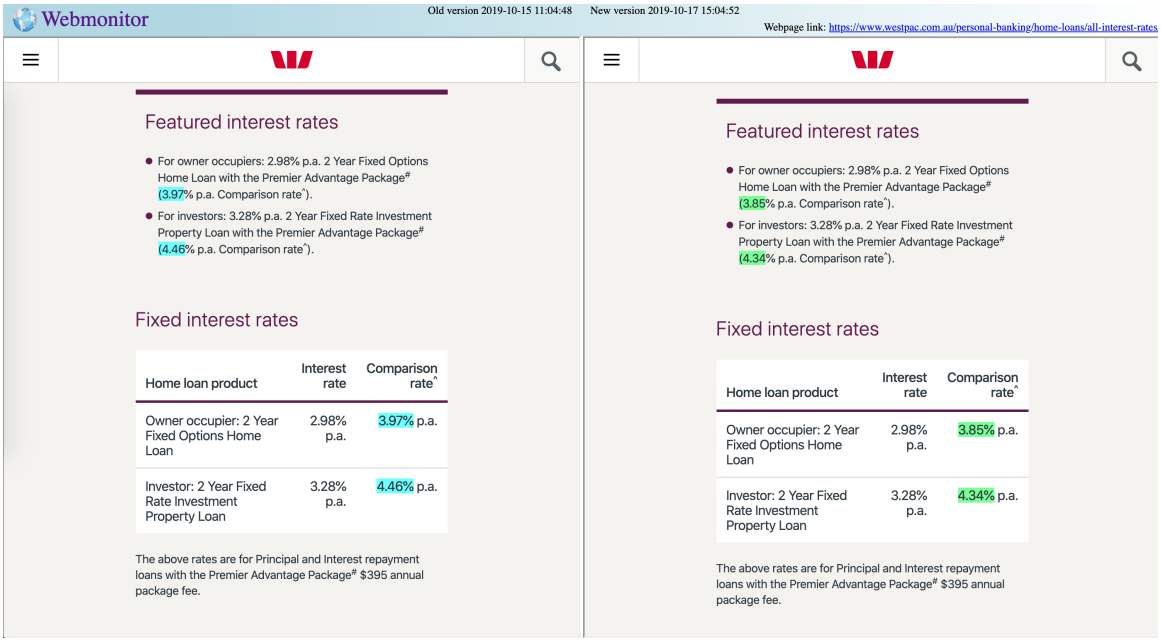


Figure 43: Test 4

Test 5 - https://www.macquariecu.com.au/loans-home-and-investment-loans.html

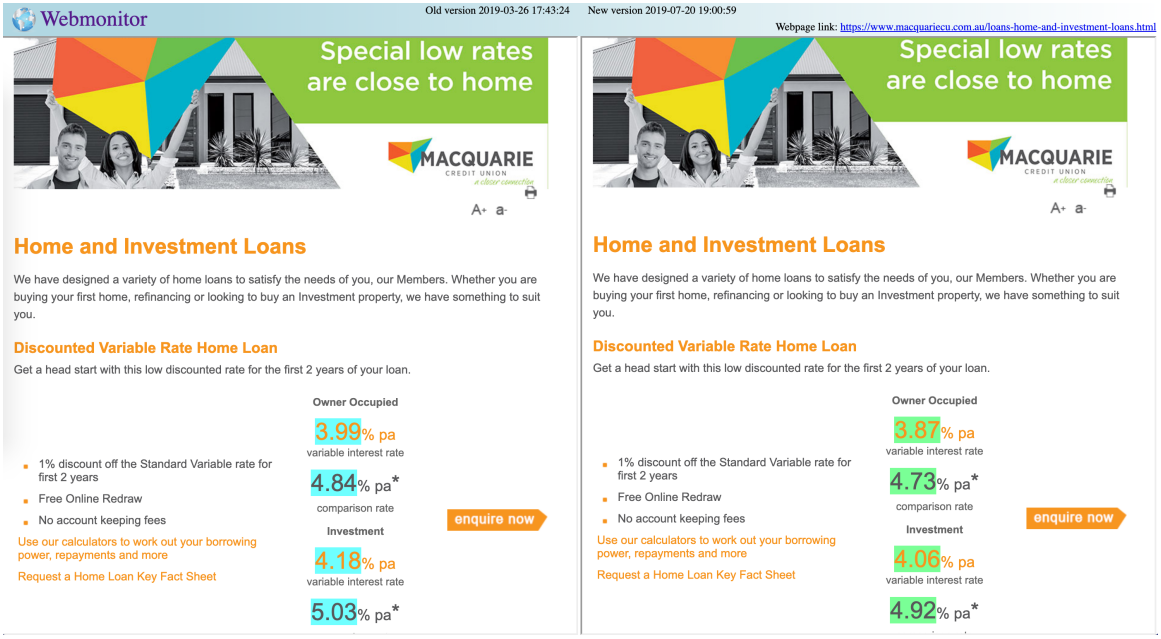


Figure 44: Test 5

4.2.2 Main View

Here we can see that on the top right corner we have pagination. For every page of the pagination, we are showing 5 changes. Also, if we check the new version times, the latest ones are at the top. Also, to note that changes are grouped by domain names.

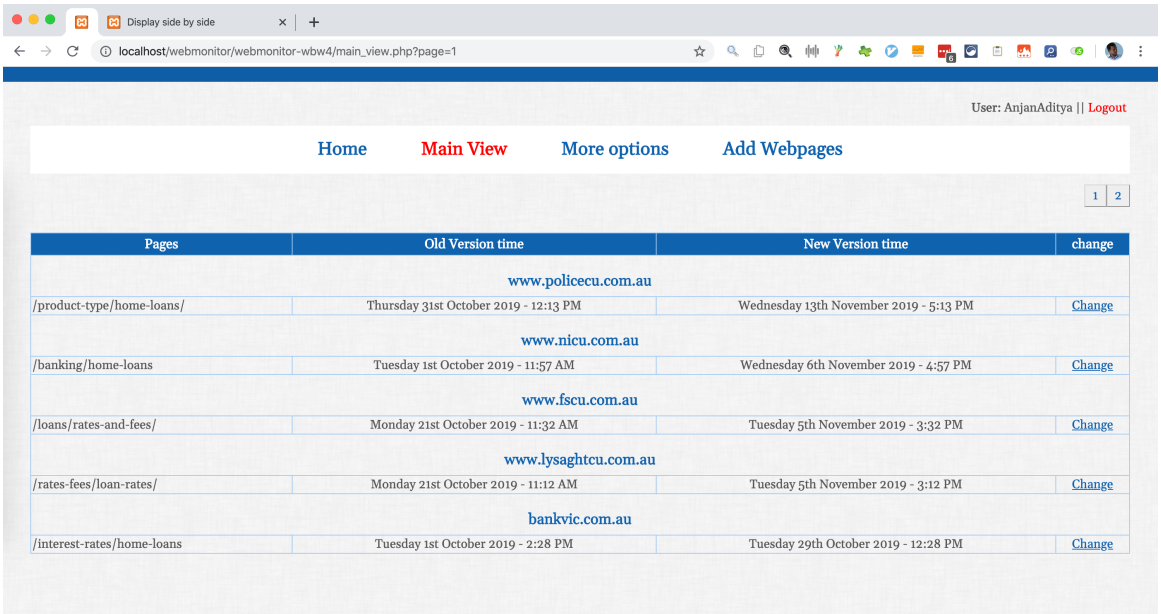


Figure 45: Page 1: Webpages are Arranged by Change Time and Domain Name

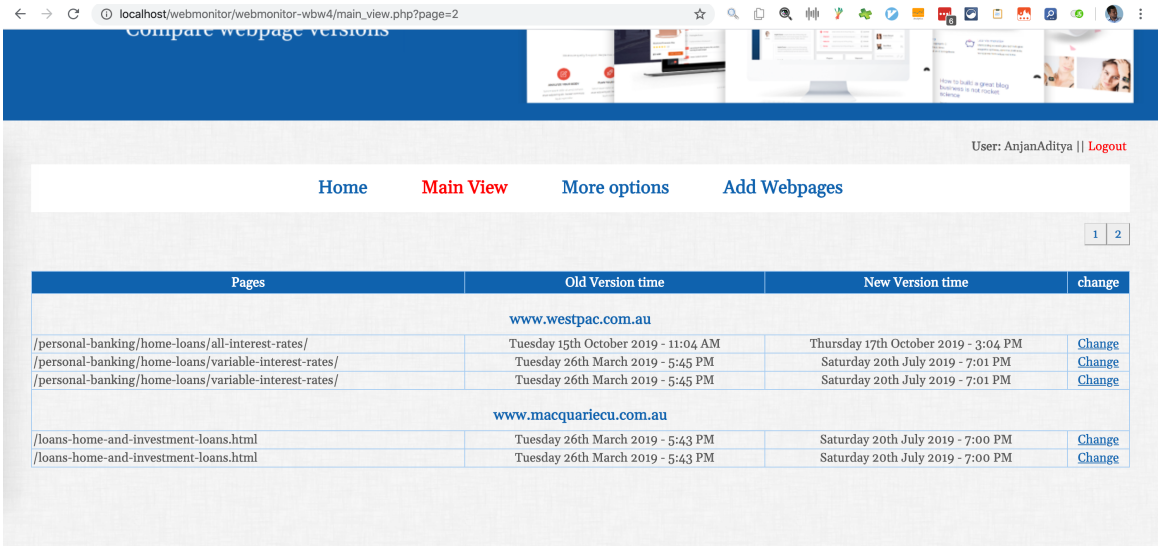


Figure 46: Page 2: Webpages are Arranged by Change Time and Domain Name

Chapter 5

Conclusion

5.1 Current Status of The Project

This project is a significant development to detect changes within versions of webpages. Currently the detection of change is based on change of text content. Quality of detection of change is very good except in some rare situations.

5.2 Future Works

In order to improve near perfect quality results, I have some plans but it will be more complex algorithm. For example, first we can rank paragraphs of both webpage versions according to similarity, so that we can compare only most similar paragraph pairs with the already existing algorithm. As a result, working scope of current algorithm will be much smaller because it will not have to work with whole webpage versions. So, the quality of detection of changes will improve. But the new algorithm will also need to take in account that there may be new or deleted paragraphs in the new webpage version.

In other ways, we have scope for future development. For example,

- Detect image changes
- Detect changes only in the HTML code
- Detect changes in PDF files
- Detect only digit changes
- Detect only non-digit changes
- Detect changes in div's with particularly mentioned class
- Ignore changes in div's with particularly mentioned class

- Categorize or tag the webpage links; so that when the user selects particular category, he can view changes in webpages only from that category.

- Filter out the changes (on the Main View) that is not checked.

Overall this project can be a great platform to launch an online service to detect change between webpage versions.

References

[1] "Diff-Text," 10 October 2019. [Online]. Available: https://en.wikipedia.org/wiki/Diff-Text.

[2] "Kompare-Wikipedia," 24 June 2019. [Online]. Available: https://en.wikipedia.org/wiki/Kompare.

[3] T. O'Reilly, T. Mott and W. J. Glenn, Windows 98 in a nutshell : a desktop quick reference, Sebastopol, California: O'Reilly Media, Inc, 1999, pp. 425-427.

[4] "WinDiff - Wikipedia," 4 March 2020. [Online]. Available: https://en.wikipedia.org/wiki/WinDiff.

[5] sneakypete81, "Update Scanner – Get this Extension for Firefox (en-US)," 24 April 2019. [Online]. Available: https://addons.mozilla.org/en-US/firefox/addon/update-scanner/.

[6] Nodetics, "PageProbe - Automatic Page Monitor - Chrome Web Store," 6 March 2020. [Online]. Available: https://chrome.google.com/webstore/detail/pageprobe-automatic-page/kpkcofgbohgcpjkeidodnmaokbjlnmac?hl=en.

[7] "PHP: implode - Manual," [Online]. Available: https://www.php.net/manual/en/function.implode.php.

[8] "PHP: preg_replace - Manual," [Online]. Available: https://www.php.net/manual/en/function.preg-replace.

[9] "PHP: preg_match_all - Manual," [Online]. Available: https://www.php.net/manual/en/function.preg-match-all.php.

[10] "regexr.com," [Online]. Available: https://regexr.com/.

[11] "PHP: unset - Manual," [Online]. Available: https://www.php.net/manual/en/function.unset.php.

[12] "PHP: str_replace - Manual," [Online]. Available: https://www.php.net/manual/en/function.str-replace.

[13] "PHP: trim - Manual," [Online]. Available: https://www.php.net/manual/en/function.trim.

[14] "PHP: explode - Manual," [Online]. Available:

https://www.php.net/manual/en/function.explode.php.

[15] "PHP: ksort - Manual," [Online]. Available: https://www.php.net/manual/en/function.ksort.php.

[16] "PHP: substr_replace - Manual," [Online]. Available: https://www.php.net/manual/en/function.substr-replace.

[17] "PHP: array_unique - Manual," [Online]. Available: https://www.php.net/manual/en/function.array-unique.

[18] "PHP: array_diff - Manual," [Online]. Available: https://www.php.net/manual/en/function.array-diff.

[19] "CloudFront," [Online]. Available: https://dj1xg6k8tv6mw.cloudfront.net/p/frame-scroll.6.min.js.

[20] "PHP: password_hash - Manual," [Online]. Available: https://www.php.net/manual/en/function.password-hash.php.

Appendix A

Here we are going to explain in detail some important function used in this application:

6.1 PHP Function Preg_match_all

preg_match_all [3] is a very important function used in this application. Here is description for it from the PHP manual:

preg_match_all — Perform a global regular expression match

6.1.1 Description

preg_match_all (string $pattern , string $subject [, array &$matches [, int $flags = PREG_PATTERN_ORDER [, int $offset = 0]]]) : int

Searches subject for all matches to the regular expression given in pattern and puts them in matches in the order specified by flags.

After the first match is found, the subsequent searches are continued on from end of the last match.

6.1.2 Parameters

6.1.2.1 Pattern

The pattern to search for, as a string.

6.1.2.2 Subject

The input string.

6.1.2.3 Matches

Array of all matches in multi-dimensional array ordered according to flags.

6.1.3 Flags

Can be a combination of the following flags

1. PREG_PATTERN_ORDER
2. PREG_SET_ORDER
3. PREG_OFFSET_CAPTURE
4. PREG_UNMATCHED_AS_NULL

In this application the PREG_OFFSET_CAPTURE was used. Here is the explanation of how it works:

If this flag is passed, for every occurring match the appendant string offset will also be returned. Note that this changes the value of matches into an array of arrays where every element is an array consisting of the matched string at offset 0 and its string offset into **subject** at offset 1.

```
<?php

preg_match_all('/(foo)(bar)(baz)/', 'foobarbaz', $matches, PREG_OFFSET_CAPTURE);

print_r($matches);

?>
```

The above example will output:

```
Array

(

   [0] => Array

      (

         [0] => Array

            (

               [0] => foobarbaz

               [1] => 0
```

```
            )

    )
[1] => Array
    (
        [0] => Array
            (
                [0] => foo
                [1] => 0
            )
    )
[2] => Array
    (
        [0] => Array
            (
                [0] => bar
                [1] => 3
            )
    )
[3] => Array
    (
        [0] => Array
```

```
        (
            [0] => baz
            [1] => 6
        )
    )
)
```

6.1.4 Offset

Normally, the search starts from the beginning of the subject string. The optional parameter offset can be used to specify the alternate place from which to start the search (in bytes).

6.1.5 Return Values

Returns the number of full pattern matches (which might be zero), or FALSE if an error occurred.

6.2 Password_hash [14]

Below description is from the php manual.

password_hash — Creates a password hash

6.2.1 Description

password_hash (string $password, int $algo [, array $options]) : string

password_hash() creates a new password hash using a strong one-way hashing algorithm. password_hash() is compatible with crypt(). Therefore, password hashes created by crypt() can be used with password_hash().

The following algorithms are currently supported:

1. PASSWORD_DEFAULT

2. PASSWORD_BCRYPT
3. PASSWORD_ARGON2I
4. PASSWORD_ARGON2ID

PASSWORD_DEFAULT was used in this application:

PASSWORD_DEFAULT - Use the bcrypt algorithm (default as of PHP 5.5.0). Note that this constant is designed to change over time as new and stronger algorithms are added to PHP. For that reason, the length of the result from using this identifier can change over time. Therefore, it is recommended to store the result in a database column that can expand beyond 60 characters (255 characters would be a good choice).

6.2.2 Parameters

6.2.2.1 Password

The user's password.

6.2.2.2 Caution

Using the PASSWORD_BCRYPT as the algorithm, will result in the password parameter being truncated to a maximum length of 72 characters.

6.2.2.3 Algo

A password algorithm constant denoting the algorithm to use when hashing the password.

6.2.2.4 Options

An associative array containing options. See the password algorithm constants for documentation on the supported options for each algorithm.

If omitted, a random salt will be created and the default cost will be used.

6.2.3 Return Values

Returns the hashed password, or FALSE on failure.

6.2.4 Password_hash() Example

```
<?php

/**

 * We just want to hash our password using the current DEFAULT algorithm.

 * This is presently BCRYPT, and will produce a 60 character result.

 * Beware that DEFAULT may change over time, so you would want to prepare

 * By allowing your storage to expand past 60 characters (255 would be good)

 */

echo password_hash("rasmuslerdorf", PASSWORD_DEFAULT);

?>
```

The above example will output something similar to:

```
$2y$10$.vGA1O9wmRjrwAVXD98HNOgsNpDczlqm3Jq7KnEd1rVAGv3Fykk1a
```