|  |
| --- |
| Mind Valley |
| ***Mind valley Homework Project*** |
| **Detail Design – Software Unit Specification** |
|  |
| **Naim ul Islam Prodhan** |
| **4/6/2013** |

|  |
| --- |
| The document describes the code files of the project. This is the low level design of the solution. Sometimes this document is called component level design or detail design. This document can be considered as a template for detail design for plug ins which has .php and .js files. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Creation/ Modification History Record Card** | | | | | |
| SL No. | Date | Version | Created/modified by | Approved/Reviewed by | Description |
| 1 | 6-Apr-2013 | 1.0 | Naim ul Islam Prodhan | Naim ul Islam Prodhan | Created the basic draft of LLD. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Table of Contents

[**1.** **Software Unit Objective & Description** 4](#_Toc352993302)

[**2.** **Reference Document** 4](#_Toc352993303)

[**3.** **Unit Calls** 4](#_Toc352993304)

[**4.** **File Code Amendment: header.php** 4](#_Toc352993305)

[**5.** **Unit Details** 5](#_Toc352993306)

[**5.1.** **Unit one: tabplugin.php** 5](#_Toc352993307)

[**5.2.** **Unit two: getpost.php** 5](#_Toc352993308)

[**5.3.** **Unit Three: tabs.js** 6](#_Toc352993309)

[**6. Unit Test Cases** 13](#_Toc352993310)

1. **Software Unit Objective & Description**
   1. **Objective**

In the Mindvalley Homework Project, there are 3 units or components. Each unit has one file. The overall objective of all the units is to show top 10 viewed or most popular word press posts for the last 7 and 30 days in a tab view’s second and third tab. In the first tab, all posts will be displayed.

* 1. **Units Description**
     1. First unit is tabplugin and its file name is tabplugin.php. This unit is responsible for showing the tab view in the word press site.
     2. Second unit is getpost and the file name is getpost.php. This unit is responsible for returning all posts in a word press site.
     3. Third unit is a java script file and the file name is tabs.js. This unit is responsible for getting data from Google analytics and put those data in the tabplugin.

1. **Reference Document**

|  |  |  |
| --- | --- | --- |
| **SL NO** | **File Name** | **Description** |
| 1 | tab-mindvalley\Email.docx. | All of the units are developed according to the specification described by the email sent from [jobs@mindvalley.com](mailto:jobs@mindvalley.com). |
| 2 | tab-mindvalley\Readme.docx | This file has the instructions to use unit files to get output. |

1. **Unit Calls**

|  |  |  |
| --- | --- | --- |
| **Unit Name** | **This Unit Calls** | **This Unit Called By** |
| tabplugin | tabs.js function | Word press site by putting text [mv-shortcode] |
| getpost | Calls word press functions | tabs.js |
| tabs.js | Calls Google analytics methods | Tabplugin |

1. **File Code Amendment: header.php**

|  |  |
| --- | --- |
| File Location: /www/naimul/wp-content/themes/twentytwelve web directory. | |
| SL NO | Description |
| 1 | In this file we added the following codes inside the head tag.  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.2/themes/smoothness/jquery-ui.css" />  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>  <script src="http://code.jquery.com/ui/1.10.2/jquery-ui.js"></script>  <script type="text/javascript" src="<?php echo plugins\_url(); ?>/tab mindvalley/js/tabs.js"></script>  <script src="https://apis.google.com/js/client.js"></script>  <script type="text/javascript">  var \_gaq = \_gaq || [];  \_gaq.push(['\_setAccount', 'UA-39768188-1']);  \_gaq.push(['\_trackPageview']);  (function() {  var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;  ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';  var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);  })();  </script> |

1. **Unit Details**
   1. **Unit one: tabplugin.php**

|  |  |  |  |
| --- | --- | --- | --- |
| File Location: tab-mindvalley\tabplugin.php | | | |
| This file is our plugiin. When we paste the short code in any of our word press page, the tab will be created. This plugin will only create the Jquery tabs. | | | |
| **SL No** | **Function Name** | **Description** | **Code/pseudo code** |
| 1 | mv\_init() | Creating a shortcode, if we paste the short code as [mv-shortcode] in any wordpress page, mv\_function will be called. | function mv\_init() {  add\_shortcode('mv-shortcode', 'mv\_function');  } |
| 2 | mv\_function($type='mv\_function') | This function will create the tabs. In the second tab, handleClientLoad() function is called. This function is located in the tabs.js | function mv\_function($type='mv\_function') {  $result='<div id="tabs">';  $result.='<ul>';  $result.='<li><a href="#tabs-1" onclick="jobOne(this)">All Posts</a></li>';  $result.='<li><a href="#tabs-2" onclick="handleClientLoad()">Last 7 days</a></li>';  $result.='<li><a href="#tabs-3" onclick="handleClientLoad()">Last 30 days</a></li>';  $result.='</ul>';  $result.='<div id="tabs-1">';  $result.='<p>Tab 1 content.</p>';  $result.='</div>';  $result.='<div id="tabs-2">';  $result.='<p>Tab 2 content</p>';  $result.='</div>';  $result.='<div id="tabs-3">';  $result.='<p>Tab 3 content</p>';  $result.='</div>';  $result.='</div>';  return $result;  } |

* 1. **Unit two: getpost.php**

|  |  |  |
| --- | --- | --- |
| File Location: tab-mindvalley\getpost.php | | |
| This is a php page is called by ajax from the tabs.js file’s ready method. This page will return all the posts in the site. The return of this page is showed in the tabs-1 div in the plugin. | | |
| **SL NO** | **Description** | **Code/pseudo code** |
| 1 | In this page, the following two lines are added to bring wordpress functionality to this file.  require\_once('../../../wp-blog-header.php');  require\_once('../../../wp-load.php');  We are creating the global whole post object and $wpdb is the global instance of wpdb class.  global $wpdb;  global $post;  Then we are querying from the post table to return all the posts, then we are giving the response to the ajax call as html. | <?php  require\_once('../../../wp-blog-header.php');  require\_once('../../../wp-load.php');  //$post is the whole post object  //$wpdb is the global instance of wpdb class  global $wpdb;  global $post;  //query all the posts and keep the results in the $result variable  $str = "SELECT $wpdb->posts.\* FROM $wpdb->posts WHERE post\_type = 'post' AND post\_status = 'publish'";  $result = $wpdb->get\_results($str);  //process the result  echo "<table><tr><th>LIST OF ALL POSTS</th></tr>";  foreach($result as $post):  setup\_postdata($post);  echo "<tr><td>";  echo "<a href='";  echo the\_permalink();  echo "'>";  echo the\_title();  echo "</a></td></tr>";  endforeach;  echo "</table>";  ?> |

* 1. **Unit Three: tabs.js**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| File Location: tab-mindvalley\js\tabs.js | | | | |
| **SL NO** | **Function Name** | **Description** | **Code/pseudo code** | **Parameters** |
|  | Ready | This method is called when a page is ready. In this method, we first send Google analytic a message that when any links in our word press site, which are post link are clicked, it will send the title of the post, along with the processed link of the post. So, google analytics will keep track number of click is done in a post.  Then, we make an ajax call to get all the posts in our word press site and we have attached the response in the tabs-1 div of the plugin. | jQuery(document).ready(function($) {  $('#tabs').tabs();  //alert("called");  var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");    //use jQuery to call the Google Analytics JavaScript  $.getScript(gaJsHost + "google-analytics.com/ga.js", function(){    //tell Analytics about the current page load using standard \_trackPageview method  try {  var \_gaq = \_gaq || [];  \_gaq.push(['\_setAccount', 'UA-39996168-1']);  \_gaq.push(['\_trackPageview']);  //var pageTracker = \_gat.\_getTracker("UA-39996168-1");  //pageTracker.\_trackPageview();  } catch(err) {alert("g");}    //loop though each anchor element  $('a').each(function(){    var href = $(this).attr('href');    //check for links starting with http or https, making sure that links are to our own domain  if ((href.match(/^https?\:/i)) && (href.match(document.domain))){  $(this).click(function() {    var extLink = href.replace(/^https?\:\/\//i, '');  //check weather it is a post link or not  //post links has a sub string ?p=  //put the ancor text as category, this is actually the title of the post  if (extLink.toLowerCase().indexOf("?p=") >= 0){  //pageTracker.\_trackEvent($(this).text(), 'Click', extLink);  \_gaq.push(['\_trackEvent', $(this).text(), 'Click', extLink]);  }  });  }  });    });  //ajax request the getPost php page  var request = $.ajax({  url: "../TestApp/wp-content/plugins/tab-mindvalley/getpost.php",  type: "GET",  dataType: "html"  });    request.done(function(msg) {  //msg has the list of all posts  $("#tabs-1").html(msg);  });    request.fail(function(jqXHR, textStatus) {  //error message if request is failed  alert( "Request failed: " + textStatus );  });  });//end of ready method |  |
| 2 | Javascript global variables  (GLOBAL DATA USED IN THIS UNIT) | Setting the client id.  Setting the url of the site.  Set the API key that we get from the Google Develoepr Console.  Set scope for analttics api. | var clientId = '946430426079-9849kq3qntjqeu7eca7d2r8gsoi67sbd.apps.googleusercontent.com';    var str ='www.kartapp.com/naimul/';  // https://kartapp.com/naimul  var apiKey = 'AIzaSyD1ONDX81rv0HYzIt352h-RoV7pbQxfG7A';    var scopes = 'https://www.googleapis.com/auth/analytics'; |  |
| 3 | handleClientLoad() | Setting the API key.  For simple access after setting the api key, call the gepi.client.load method. It will bypass authorization. | function handleClientLoad() {  gapi.client.setApiKey(apiKey);  //simple access using api key  //no need to call window.setTimeout(checkAuth,1);  ` //rather directly call the following line.  //gapi.client.load('analytics', 'v3', makeApiCall);  window.setTimeout(checkAuth,1);  } |  |
| 4 | checkAuth() | When immediate is false, the google authentication popup will come. **[So far I found in Google that if immediate is set true, pop up will not come. I failed to check because filezilla cannot connect to the ftp. Server may be down from Friday morning. It is giving authentication error. I have tested that ftp can be accessed from Bangladesh but from Friday, I cannot connect ftp. For now to test I am giving, my google account credential, User:naimul.prodhan@gmail.com Pass:apiitucti1234 But be etical, don’t see my emails.]** | function checkAuth() {  gapi.auth.authorize({client\_id: clientId, scope: scopes, immediate: true}, handleAuthResult);  } |  |
| 5 | handleAuthResult(authResult) | If successfully authenticate, then it is calling the makeApiCall method. | function handleAuthResult(authResult) {  if (authResult) {  // The user has authorized access  // Load the Analytics Client.  gapi.client.load('analytics', 'v3', makeApiCall);  } else {  // User has not Authenticated and Authorized  alert("User don't have access.");  }  } | Getting the authentication result. |
| 6 | makeApiCall() | This function has three date variable, one has the current date, the second one has 7 days back date and the last one has 30 days back date.  Then using the Analytics service object we executing two queries, one will return the event categories with action and event count in descending order and will return 10 rows for last 7 days and the second query will return same for last 30 days. | // Use the Analytics Service Object to query the Core Reporting API  function makeApiCall() {  var currentDate = new Date();  var currentMonth = currentDate.getMonth()+1;  var currentDay = currentDate.getDate();  var currentYear = currentDate.getFullYear();  var modCurrentMonth = (currentMonth<10 ? '0' : '') + currentMonth;  var modCurrentDay = (currentDay<10 ? '0' : '') + currentDay;  //this is current date  var curFullDate = currentYear + "-" + modCurrentMonth + "-" + modCurrentDay;  var difCurDate = new Date(currentYear , modCurrentMonth, modCurrentDay);  var curDate = new Date(difCurDate);  curDate.setDate(curDate.getDate() - 7);  var newDate = new Date(curDate);  var newDay = newDate.getDate();  var newMonth = newDate.getMonth();  var newYear = newDate.getFullYear();  var modNewMonth = (newMonth<10 ? '0' : '') + newMonth;  var modNewDay = (newDay<10 ? '0' : '') + newDay;  //this is a date seven days before the current date, means for last 7 days query  var newFullDate = newYear + "-" + modNewMonth + "-" + modNewDay;  var curDate = new Date(difCurDate);  curDate.setDate(curDate.getDate() - 30);  var newDate = new Date(curDate);  var newDay = newDate.getDate();  var newMonth = newDate.getMonth();  var newYear = newDate.getFullYear();  var modNewMonth = (newMonth<10 ? '0' : '') + newMonth;  var modNewDay = (newDay<10 ? '0' : '') + newDay;  //this date is 30 days before, means for last 30 days query  var newFullDateThirty = newYear + "-" + modNewMonth + "-" + modNewDay;  //query for the last seven days top posts means those  //posts that have highest clicks  var apiQuery = gapi.client.analytics.data.ga.get({  'ids': 'ga:70949199',//this is the profile id from where fetching data  'start-date': newFullDate,  'end-date': curFullDate,  'metrics': 'ga:totalEvents',  'dimensions': 'ga:eventCategory,ga:eventAction,ga:eventLabel',  'sort': '-ga:totalEvents',  'max-results': 10  });  //executing the query  apiQuery.execute(handleCoreReportingResults);  //query for the last thirty days top posts means those  //posts that have highest clicks  var apiQueryNext = gapi.client.analytics.data.ga.get({  'ids': 'ga:70949199', //this is the profile id from where fetching data  'start-date': newFullDateThirty,  'end-date': curFullDate,  'metrics': 'ga:totalEvents',  'dimensions': 'ga:eventCategory,ga:eventAction,ga:eventLabel',  'sort': '-ga:totalEvents',  'max-results': 10  });  //executing the query  apiQueryNext.execute(handleCoreReportingResultsThirtyDays);  } |  |
| 7 | handleCoreReportingResultsThirtyDays(results) | If the query successfully returned without error, it will call the printRowsThirtyDays(results) method. The method itself is called for last 30 days query. | function handleCoreReportingResultsThirtyDays(results) {  if (!results.error) {  // success. call to show results  printRowsThirtyDays(results);  } else {  alert('There was an error: ' + results.message);  }  } | Result of the query. |
| 8 | handleCoreReportingResults(results) | If the query successfully returned without error, it will call the printRows(results) method. The method itself is called for last 7 days query. | function handleCoreReportingResults(results) {  if (!results.error) {  // success. call to show results  printRows(results);  } else {  alert('There was an error: ' + results.message);  }  } | Results of the query. |
| 9 | printRows(results) | This method will process the results and add in the tabs-2 div of the plugin. | function printRows(results) {  var title;  var link;  var ancor;  if (results.rows && results.rows.length) {  var table = '<table>';  // put headers in table.  table += '<tr>';  for (var i = 0, header; header = results.columnHeaders[i]; ++i) {  if(i==0 )  table += '<th>Post Titles</th>';  if(i==3)  table += '<th>Hits per Post</th>';  }  table += '</tr>';  // put cells in table.  for (var i = 0, row; row = results.rows[i]; ++i) {  table += '<tr>';  $.each(row, function(j, val) {  if(j==0){  title = val;  }  if(j==2){  link = val;  link = link.replace(str,"");  ancor = '<a href="' + link + '">' + title + '</a>';  table += '<td>' + ancor + '</td>'  }  if(j==3)  table += '<td>' + val + '</td>';  });  table += '</tr>';  }  table += '</table>';  } else {  output.push('<p>No Results Found</p>');  }  $("#tabs-2").html(table);  } | Results of the query. |
| 10 |  | This method will process the results and add in the tabs-3 div of the plugin. | function printRowsThirtyDays(results) {  var title;  var link;  var ancor;  if (results.rows && results.rows.length) {  var table = '<table>';  // put headers in table.  table += '<tr>';  for (var i = 0, header; header = results.columnHeaders[i]; ++i) {  if(i==0 )  table += '<th>Post Titles</th>';  if(i==3)  table += '<th>Hits per Post</th>';  }  table += '</tr>';  // put cells in table.  for (var i = 0, row; row = results.rows[i]; ++i) {  table += '<tr>';  $.each(row, function(j, val) {  if(j==0){  title = val;  }  if(j==2){  link = val;  link = link.replace(str,"");  ancor = '<a href="' + link + '">' + title + '</a>';  table += '<td>' + ancor + '</td>'  }  if(j==3)  table += '<td>' + val + '</td>';  });  table += '</tr>';  }  table += '</table>';  } else {  output.push('<p>No Results Found</p>');  }  $("#tabs-3").html(table);  } | Results of the query. |

**6. Unit Test Cases**

No unit test cases were produced for any of the units but a series of black box testing were done from personal computer in firefox browser and from iphone4 in safari browser. Test results were passed.