FACULTY OF MECHANICAL ENGINEERING

Institute of Automation and Computer Science

PHP Library for Graphical Implementations of Data

MASTER'S THESIS

Author: Ali Sarp Uyandiran

Supervisor: doc. Jan Roupec Ph.D.

BRNO 2020

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Specification Master's Thesis

Department:	Institute of Automation and Computer Science
Student:	**Bc. Ali Sarp Uyandiran**
Study programme:	Mechanical Engineering
Study branch:	Applied Computer Science and Control
Supervisor:	**doc. Ing. Jan Roupec, Ph.D.**
Academic year:	2019/20

Pursuant to Act no. 111/1998 concerning universities and the BUT study and examination rules, you have been assigned the following topic by the institute director Master's Thesis:

PHP knihovna pro generování grafů

Concise characteristic of the task:

Mnoho webovských aplikací vyžaduje výstup ve formě grafů. Na straně serveru bývají tyto www stránky řešeny obvykle jako dynamické s využitím jazyka php. Předpokládá se vytvoření knihovny v php, která umožní snadné generování grafů ve formátu svg.

Goals Master's Thesis:

Práce bude obsahovat pojednání o typech grafických reprezentací dat podle jejich použití. Budou vytypovány vhodné typy grafů pro jejich zobrazování v html dokumentech. Konečně bude proveden návrh a realizace objektově orientované knihovny v jazyve php pro generování grafů v dynamických www stránkách.

Recommended bibliography:

MATANGE, S., HEATH, D.: Statistical Graphics Procedures by Example: Effective Graphs Using SAS. SAS Institute, NC, USA, 2011.

TATROE, K., MacINTYRE, K.: Programming PHP: Creating Dynamic Web Pages. 3th Edition. O'Reily, CA, USA, 2013.

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2019/20

In Brno,

L. S.

doc. Ing. Radomil Matoušek, Ph.D. Director of the Institute	doc. Ing. Jaroslav Katolický, Ph.D. FME dean

ABSTRACT

The programming languages have always been a useful tool for sustainability and continuity of advancements in multilayered fields as like education and common industries. PHP is a widespread instrument which has an important place in this practices just like many other programming languages. This research is conducted to understand PHP language better and to increase academical familiarity about it. In the end of this practice a basic PHP library for further use has been produced as a representation of the research which has been conducted. This library allows the user to create multivariate basic graphs for visualization of variable different data and it's hoped that it will be used by further educational and business purposes by various individuals.

KEYWORDS:

PHP, Graphs, Library, Chart, Data

TABLE OF CONTENTS

LIST OF FIGURES

Number *Page*

ACKNOWLEDGEMENTS

Gratefully wish to express sincere appreciation to Mr. doc. Roupec Ph.D. for his assistance in the preparation of this research. In addition, would like to give a special thanks to FME faculty staff, teachers and coordinators for their all kinds of support during the study period.

DECLARATION

I would like to declare that this thesis is an original work which is prepared independently under the guidance of doc. Ing. Jan Roupec Ph.D. and sources which are listed in the references section.

06.06.2020 Ali Sarp Uyandiran

INTRODUCTION

A designed drawing, consisting of shapes and images with purpose of to display data that are complicated to be explicated only by quantitatively, can be defined as a graph. Graphs, transforms the information to more understandable outlook. This is particularly important when two or more sets of data are correlative in some aspects. Many areas in daily life is making use of the graphs as a major gadget to deliver the information to their subjects.

Web sources are important way to deliver the information fast and efficient way to multiple individuals on current information age. They can be defined as any kind of material that an individual can find in online environment. Naturally, graphs are a part of this information flow in internet because the usage of graphs are a fundamental part of human perception and communication.

This research has been conducted in the pursuit of forming a code which can visualize basic graph types in web environment as an indicative of an effort to learn a scripting language from scratch.

In this thesis, it's aimed to give primary introductions to topic related programming language, making descriptions about graph types and utilization of those graphs while explaining variety of them and giving detailed descriptions about prepared code to give information about it's functioning.

Chapter I

INTRODUCTION TO PHP

1.1 Web Technologies

Computers can't interact with each other in a unique way like humans so they need markup languages and multimedia packages to achieve these interactions. These interactions can be defined as Web Technologies. **[3]** Communication through Web Technologies can be succeeded through web browsers which are using HTML as a standard markup language for displaying items. These operations mostly occur with the aid by other technologies like CSS and JavaScript.

HTML is a tool for web browsers for explicating the substance of the page, which means it's directly affecting the behavioral structure of web pages. The interactive structure of HTML creates a beneficial environment for embedding HTML code in PHP for creating more sophisticated creations.

Together, HTML, CSS, and JavaScript form the requisite pieces of web environments. To elaborate; HTML is providing the skeleton of the page while CSS tends the appearance and JavaScript implements the administration and functionality of page.

1.2 PHP Attributes

PHP (acronym for "PHP: Hypertext Preprocessor"), is a commonly-used, free to download and utilize, open source scripting language which is granting many capabilities to it's user.

PHP files can be distinguished by their own extension ".php". Among with other capabilities, the main attributes can be specified as getting used in encrypting data, controlling user-access, collecting form data, recasting an existing data in your database, making cookie exchange, making all kinds of modification in files on the server and generating active page content.

PHP is always a good selection of choice for programmers because of it's distinct features. The basic features can be summarized as; performing system operations like open, read, create, close and etc., handling forms, gathering data, saving data, sending data through electronic mail. It can make modification in elements in the database, access cookies, encrypt data. As a security aspect, it can restrict movements of it's user by choice of the programmer, yields a flexible and effective set of safeguards as like safe mode which can bring limitations to movements of users as well as the memory usage and application time. These limitations can affect the performance in drastic ways. It should be stated that PHP can work in a coherent way with other applications which is an advantageous feature to have during designing web applications. Also PHP source code is not observable through the browser because the script is entirely parsed prior to it is shipped back to demanding user. **[4]**

1.3 PHP Libraries

PHP libraries, can be defined as cumulative formation of subroutines and classes to achieve the tasks which user desires to accomplish with minimal and straightforward effort.

A library can differ as the developer's own personal collective ideas molded together to achieve the tasks for specific purposes which are presented for PHP users to use.

1.4 Web Servers and Apache Utilization

A web server is software or hardware which operates with HTTP (Hypertext Transfer Protocol) and wider protocols to replies the wishes of the user on web. Web server software inspects and supervises the visitation to content. These content can be reached by the domain titles of webpages.

As concretely, a web server is a computer that houses the software of web server and relevant folders connected to the webpage, like HTML data and image files. This hardware is connected to network and if it's required it sends information to clients which are also connected to the network.

A web page is good environment to successfully visualize the content of PHP file. To perform this, a web server is required. Leading web servers include Apache, Microsoft's Internet Information Server (IIS) and Nginx . Other alternatives are Novell's NetWare server, Google Web Server (GWS) and IBM's family of Domino servers, Litespeed, and Cherokee. **[5]**

Apache carries the picked file, pipes it into the PHP binary, and routes the output stream from the command down the HTTP connection. Through the

environment protocols PHP code gets translated to visible images. These operations assists on any visuals coded in PHP file to be displayed on web page.

Chapter II

GRAPHICAL IMPLEMENTATIONS OF DATA

2.1 Graphs and Graph Types

Graphs are practical and extensive way to display a clear relation between several data. The goal of a graph is help to visualize a data that has many information, with introducing it to human perception with more simplistic and elegant way.

A important feature for a graph is to be undemanding and understandable. The most common used graphs in the daily life is;

Scatter plots ,**[6]** are a variety of mathematical diagrams to visualize values for two sets of data using Cartesian coordinate system. This data are displayed by point shaped figures which has their own specific values which can be determined through looking specific location between two different axes. (see Figure 1)

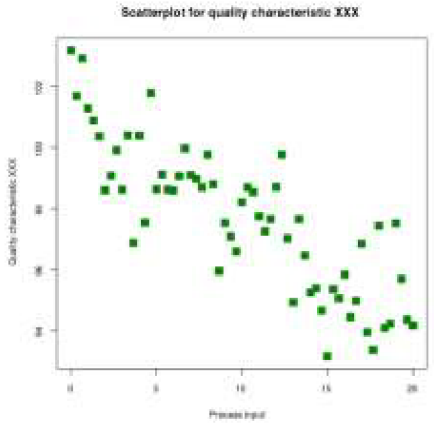

Figure 1: Scatter plot example **[6]**

Bar graphs, **[7]** are a graphical way to show data which is divided into different categories. It is displayed via bar columns which has different sizes which is differing from each other according to their quantity inside the whole group. (see Figure 2)

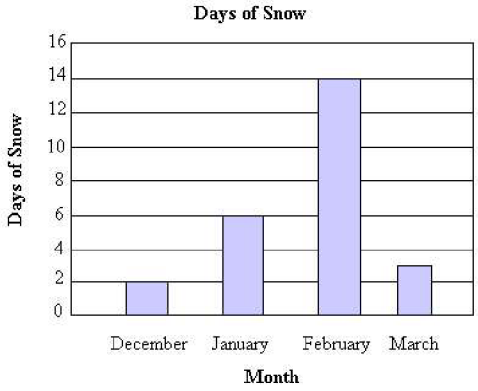

Figure 2: Bar graph example **[7]**

Line graphs, **[8]** are useful to represent the change between two states which has numerical values and to estimate behavioural predisposition. The change in values are shown with distinctive lines to create path-like visual. (see Figure 3)

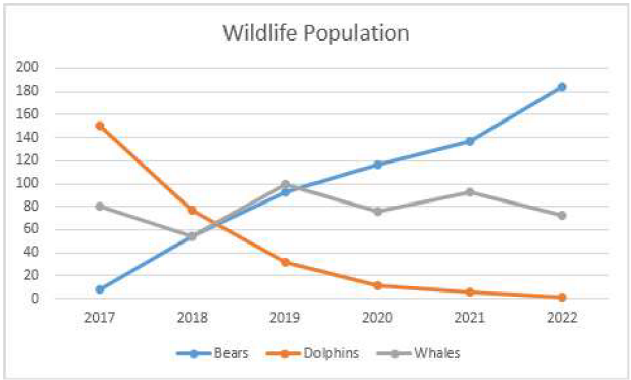

Figure 3: Line graph example **[8]**

Pie charts, **[9]** are circular graphs which are split into individual sections to represent the percentage of an element in a group which has multivariate pieces. It is named after it's cunning resemblance of a sliced pie. (see Figure 4)

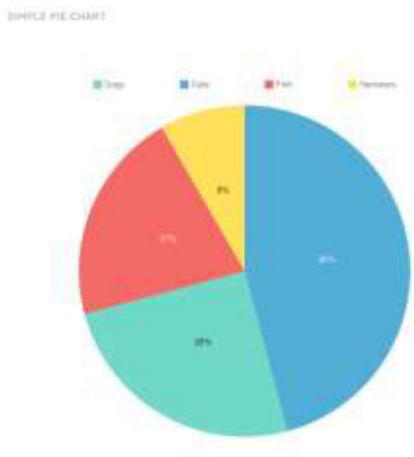

Figure 4: Pie chart example **[9]**

2.2 The Importance of Graphs

Graphs are always been an important asset in areas which has educational and occupational purposes. This is the result of the need of simplistic approaches in any establishment which is aiming to achieve sustainable growth and stability.

It's a part of human nature to avoid complicated instances especially if it's abstract contexts. To understand and explicate these concepts better, graphs play an important role in historical course of mankind and usage of it has been common for a really in long time in human history and will continue to keep it's worth for the foreseeable future.

Chapter III

THE LIBRARY DESIGN AND DESCRIPTION

3.1 Functionality and Method

To create a practical library the functions built in a class to provide convenience in feasibility and testing of the code.

The code is constructed to operate by creating an object and addressing the certain functions while specifying their attributes or values to create a graph which has the users desired features. Input attributes or values can differ in different graph types because all graphs have distinct features which are representing distinct data.

3.2 Running The Code

The code becomes operational when Apache server is activated from XAMMP software and the necessary file is specified in the page bar of browser which is fetched from the folder which contains the PHP file(in this case it's http://localhost/sarpproject/test2.php) (see Figure 5 and Figure 6)

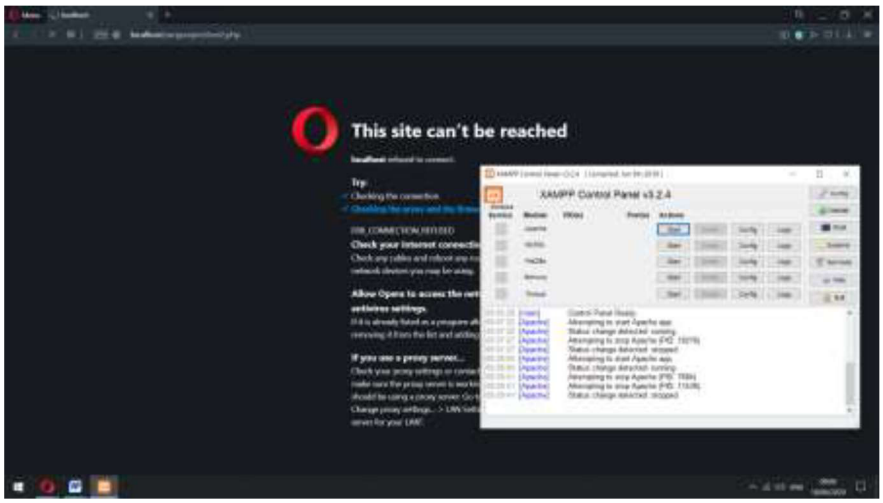

Figure 5: Screenshot of the operation (before starting the Apache server)

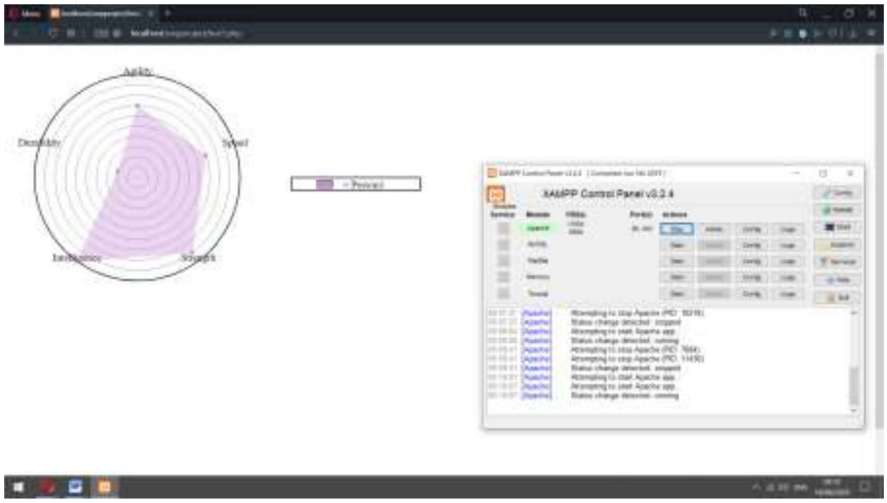

Figure 6: Screenshot of the operation (after starting the Apache server)

3.3 Intro and Functions to Call Ahead

```
1   <?php
2
3   class mysvg{ //using class method to execute
4   orders easily
```

```
5   function randomcolor(){ //function for
6   getting a random color from 7*7*24 choices
7       $an = rand(0,23); //there will be 24
8   different spectrum
9       $bn = rand(2,8); //there will be 7
10  different spectrum
11      $cn = rand(2,8); //there will be 7
12  different spectrum
13      $color =
14  'hsl('.($an*15).','.($bn*10).'%,'.($cn*10).'%
15  )'; //creates a color from different
16  spectrums
17      return $color;
18  }
```

The **Randomcolor** function is used when a unique color is needed during the process of the code. The outcome is a particular color from wide color palette which will be used to mark out an individual shape or illustration.

```
19  function roundn($x){ //rounding numbers
20  according to their decimals
20      $c1 = strlen((int)$x);
21      $c2 = ceil($x/pow(10, $c))*pow(10, $c);
22      echo $c2;
23  }
```

The **roundn** function will be used to round some specific number to their decimals.

```
24  function ifd($x, $y, $a){ //this function
25  pinpoints the location of input in
26  coordinates system
27    if($x >= 0){
28      if($y >= 0){
29        $xx= abs($x)+$a[1];
30        $yy= abs($y)+$a[0]-2*abs($y);
31      }else if($y <= 0){
32        $xx= abs($x)+$a[1];
33        $yy= abs($y)+$a[0];
34       }
35    }else if($x <= 0){
36      if($y >= 0){
37          $xx= abs($x)+$a[1]-2*abs($x);
38          $yy= abs($y)+$a[0]-2*abs($y);
39      }else if($y <= 0){
40          $xx= abs($x)+$a[1]-2*abs($x);
41          $yy= abs($y)+$a[0];
42       }
43     }
44   return array($xx, $yy);
45  }
```

The **ifd** function works as a sort of coordinate correction. It corrects the location of a specific point according to the chosen center of svg from the north-west corner of it which is the original center(0,0).

```
46  function drawRDR($a, $m){ //this function is
47  used to to create skeleton for drawing radar
48  chart
49      $txt =  "<svg height=\"".($a[0]*2+10)."\"
50  width=\"".($a[1]*2+10)."\">"; //creating the
51  environment
52      $txt .=  "<circle cx=\"".$m[0]."\"
53  cy=\"".$m[1]."\" r=\"200\" stroke=\"black\"
```

```
54  stroke-width=\"2\" fill=\"white\"/>";
55      for($i = 1; $i <= 9; $i++) { //for loop for
56  creating circles in chart
57          $txt .=  "<circle cx=\"".$m[0]."\"
58  cy=\"".$m[1]."\" r=\"".(200-($i*20))."\"
59  stroke=\"black\" stroke-width=\"0.5\"
60  fill=\"white\"/>";
61      }
62      $txt .=  "<circle cx=\"".$m[0]."\"
63  cy=\"".$m[1]."\" r=\"1\" stroke=\"black\"
64  stroke-width=\"0.5\" fill=\"white\"/>"; //for
65  creating the smallest circle in the middle
66      return $txt;
67  }
```

The **drawRDR** function will be used while creating radar charts. It will create the circles as a skeleton and the shapes will be added after as elements.

3.4 Circular Graphs

```
68  function drawpie($names, $pointx){ //this
59  function is for drawing pie charts
70      $pointx = array_diff($pointx, [0]);
71  //removes 0 values
72  function pie($m, $angle, $mns=-1){ //function
73  to create pie slices
74      $gap=0.001; //the gap between slices
75  which in this case it should be close to zero
76      $an=($angle-90)*(M_PI/180.)-
78  $mns*asin($gap/$m); //angle will be crucial
79  to decide the slice percentage
80      return
81  sprintf('%0.2f,%0.2f',210+$m*cos($an),210+$m*
82  sin($an));
83  }
```

```
84      $txt = '<svg  width="800" height="800">';
85  //creating the environment
86      $ang1=0;
87      for($i=0; $i<= sizeof($pointx)-1;
88  $i=$i+1){
89          $n=0.001; //middle point of chart
90  which this case it should be close to zero
91          $rc=200; //radius
92          $dang = $pointx[$i] *
93  360/array_sum($pointx); // d angle
94          $laf  = $dang > 180? 1 : 0; // large
95  Arc Flag
96          $ang2 = $ang1 + $dang;      // second
97  angle
98          $cc[$i] = $this->randomcolor();
99  //calling a randomcolor function for the
100 color for a slice
101         $txt .= '<path
102 d="M'.pie($rc,$ang1).'L'.pie($n, $ang1)." A
103 $n,$n, 0,$laf,1 "  .pie($n,$ang2,1).
104                  'L'.pie($rc,$ang2,1)."A
105 $rc,$rc, 0,$laf,0, ".pie($rc,$ang1).'"
106 style="fill:'.$cc[$i].'" />'."\n";
107         $ang1=$ang2;
108         $dd[$i] =
109 100*($pointx[$i]/array_sum($pointx)); //with
110 this object the percentage of slice can be
111 stated
112         if(sizeof($pointx)==1){
113             $txt .= "<circle cx=\"210\"
114 cy=\"210\" r=\"200\" fill=\"".$cc[$i]."\"
115 />";
116             }
117             $txt .= "<rect x=\"400\"
118 y=\"300\" width=\"250\"
119 height=\"".(25+(sizeof($pointx)-1)*30)."\"
120 style=\"fill:none;stroke-
121 width:2;stroke:rgb(0,0,0)\" />"; //for
122 creating the big rentangle in explanation box
123             $txt .= "<rect x=\"450\"
124 y=\"".(305+($i)*30)."\" width=\"30\"
125 height=\"15\"
```

```
126 style=\"fill:".$cc[$i].";stroke-
127 width:1;stroke:rgb(0,0,0)\" />"; //for
128 creating the small rentangle in explanation
129 box
130             $txt.= "<text x=\"490\"
131 y=\"".(290+($i+1)*30)."\"
132 fill=\"rgb(0,0,0)\"\">= %".round($dd[$i])."
133 ".$names[$i]."</text>"; //for creating the
134 text in explanation box
135             }
136     echo $txt;
137 }
```

Function **drawpie** function is to create pie charts with various colors. It is using a path method which consists of creating multiple circles on top of each other and masking itself to create pie-like shape.

If the user applies the code below,

```
$sv = new mysvg();
$pointx = array(50,50,50);
$names = array('Apples', 'Oranges', 'Bananas');
$sv->drawpie($names, $pointx);
```

The following graph can be expected: (see Figure 7)

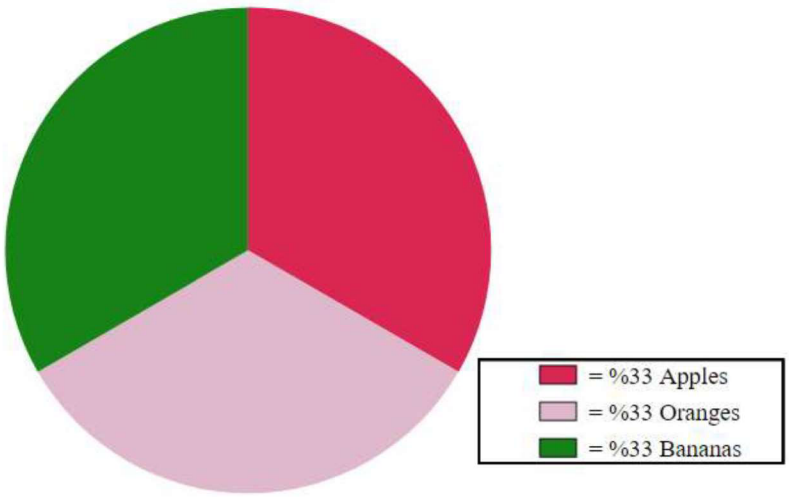

Figure 7: Pie chart

```
function drawdonut($names, $pointx){ //this
function is for drawing doughnut charts
    $pointx = array_diff($pointx, [0]);
//removes 0 values
function pie($pointx, $m, $angle, $mns=-1){
    if(sizeof($pointx)==1){
      $gap=0.005; //the gap between slices
      }else{
        $gap=3;}
    $an=($angle-90)*(M_PI/180.)-
$mns*asin($gap/$m);
    return
sprintf('%0.2f,%0.2f',210+$m*cos($an),210+$m*
sin($an));
    }
    $txt = '<svg  width="800" height="800">';
    $ang1=0;
    for($i=0; $i<= sizeof($pointx)-1;
$i=$i+1){
      $n=90;
      $rc=200;
      $dang = $pointx[$i] *
360/array_sum($pointx); // d angle
      $laf  = $dang > 180? 1 : 0; // large
Arc Flag
      $ang2 = $ang1 + $dang;      // second
angle
      $cc[$i] = $this->randomcolor();
//calling randomcolor function for each slice
      $txt .= '<path
d="M'.pie($pointx,$rc,$ang1).'L'.pie($pointx,
$n, $ang1)." A $n,$n, 0,$laf,1 "
.pie($pointx,$n,$ang2,1).

'L'.pie($pointx,$rc,$ang2,1)."A $rc,$rc,
0,$laf,0, ".pie($pointx,$rc,$ang1).'"
style="fill:'.$cc[$i].'" />'."\n";
      $ang1=$ang2;
      $dd[$i] =
```

```
177 100*($pointx[$i]/array_sum($pointx)); //with
178 this object the percentage of slice can be
179 stated
180       $txt .= "<rect x=\"400\" y=\"300\"
181 width=\"250\"
182 height=\"".(25+(sizeof($pointx)-1)*30)."\"
183 style=\"fill:none;stroke-
184 width:2;stroke:rgb(0,0,0)\" />"; //for
185 creating the big rentangle in explanation box
186       $txt .= "<rect x=\"450\"
187 y=\"".(305+($i)*30)."\" width=\"30\"
188 height=\"15\"
189 style=\"fill:".$cc[$i].";stroke-
190 width:1;stroke:rgb(0,0,0)\" />"; //for
191 creating the small rentangle in explanation
192 box
193       $txt .= "<text x=\"490\"
194 y=\"".(290+($i+1)*30)."\"
195 fill=\"rgb(0,0,0)\"\">= %".round($dd[$i])."
196 ".$names[$i]."</text>"; //for creating the
197 texts in explanation box
198       $txt .= "<text text-anchor=\"middle\"
199 x=\"210\" y=\"200\"
200 fill=\"rgb(0,0,0)\"\">TOTAL</text>"; //TOTAL
201 text
202       $txt .= "<text text-anchor=\"middle\"
203 x=\"210\" y=\"225\"
204 fill=\"rgb(0,0,0)\"\">".array_sum($pointx)."<
205 /text>";//total number text
206     }
207 echo $txt;
208 }
```

The **drawdonut** function is to create doughnut charts. It has the same principle with **drawpie** function but the spaces between slices and center point has been increased as can be seen on the *line 144* to create a different visualization.

The text in the middle is representing the total number of subjects which allows this type of chart to be used for more quantitative purposes.

If the user applies the code below,

```
$sv = new mysvg();
$pointx = array(50,50,50);
$names = array('Apples', 'Oranges', 'Bananas');
$sv->drawdonut($names, $pointx);
```

The following graph can be expected: (see Figure 8)

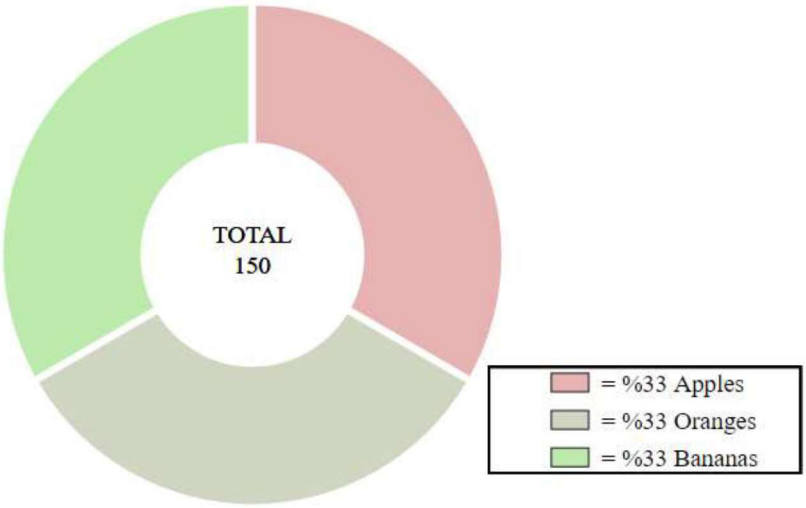

Figure 8: Doughnut chart

3.5 Visualizing Graphs and Equations in Coordinate System

```
209 function drawCS($a, $range){//draw a
210 coordinate system
211
212 function setrange($a, $range, $txt){ //set
213 range of graph
214         for($i = 0; $i <= ($a[1]/$range);
215 $i++) { //dotted lines - little lines + text
216             $backup1 = $i*$range;
217              $txt .= "<polyline
218 points=\"".($a[1]+$backup1).",".($a[0]-
219 $a[0])." ".($a[1]+$backup1).",".(2*$a[0])."\"
```

```
220
221 style=\"fill:none;stroke:rgb(0,0,0);stroke-
222 width:1\" stroke-dasharray=\"15,15\" d=\"M5
223 20 l215 0\" />
224
225             <polyline points=\"".($a[1]-
226 $backup1).",".($a[0]-$a[0])." ".($a[1]-
227 $backup1).",".(2*$a[0])."\"
228
229 style=\"fill:none;stroke:rgb(0,0,0);stroke-
230 width:1\" stroke-dasharray=\"15,15\" d=\"M5
231 20 l215 0\" />"; //dotted lines
232
233             $txt .= "<polyline
234 points=\"".($a[1]+$backup1).",".($a[0]-5)."
235 ".($a[1]+$backup1).",".($a[0]+5)."\"
236
237 style=\"fill:none;stroke:rgb(0,0,0);stroke-
238 width:3\" />
239
240             <polyline points=\"".($a[1]-
241 $backup1).",".($a[0]-5)." ".($a[1]-
242 $backup1).",".($a[0]+5)."\"
243
244 style=\"fill:none;stroke:rgb(0,0,0);stroke-
245 width:3\" />"; //little lines
246
247             $txt .= "<text text-
248 anchor=\"middle\"
249 x=\"".($a[1]+$backup1+5)."\"
250 y=\"".($a[0]+20)."\" font-size=\"10\"
251 fill=\"rgb(0,0,0)\">".$backup1."</text>
252             <text text-anchor=\"middle\"
253 x=\"".($a[1]-$backup1+5)."\"
254 y=\"".($a[0]+20)."\" font-size=\"10\"
255 fill=\"rgb(0,0,0)\">".-$backup1."</text>";
256 //text
257         }
258         for($i = 0; $i <= ($a[0]/$range);
259 $i++) { //dotted lines - little lines + text
260             $backup1 = $i*$range;
261              $txt .= "<polyline
```

```
262 points=\"".($a[1]-
263 $a[1]).",".($a[0]+$backup1)."
264 ".(2*$a[1]).",".($a[0]+$backup1)."\"
265
266 style=\"fill:none;stroke:rgb(0,0,0);stroke-
267 width:1\" stroke-dasharray=\"15,15\" d=\"M5
268 20 l215 0\" />
269
270             <polyline points=\"".($a[1]-
271 $a[1]).",".($a[0]-$backup1)."
272 ".(2*$a[1]).",".($a[0]-$backup1)."\"
273
274 style=\"fill:none;stroke:rgb(0,0,0);stroke-
275 width:1\" stroke-dasharray=\"15,15\" d=\"M5
276 20 l215 0\" />"; //dotted lines
278
279             $txt .= "<polyline
280 points=\"".($a[1]-5).",".($a[0]+$backup1)."
281 ".($a[1]+5).",".($a[0]+$backup1)."\"
282
283 style=\"fill:none;stroke:rgb(0,0,0);stroke-
284 width:3\"  />
285
286             <polyline points=\"".($a[1]-
287 5).",".($a[0]-$backup1)."
288 ".($a[1]+5).",".($a[0]-$backup1)."\"
289
290 style=\"fill:none;stroke:rgb(0,0,0);stroke-
291 width:3\"  />"; //little lines
292
293             $txt .= "<text text-
294 anchor=\"end\" x=\"".($a[1]-10)."\"
295 y=\"".($a[0]+$backup1+5)."\" font-size=\"10\"
296 fill=\"rgb(0,0,0)\">".-$backup1."</text>
297             <text text-anchor=\"end\"
298 x=\"".($a[1]-10)."\" y=\"".($a[0]-
299 $backup1+5)."\" font-size=\"10\"
300 fill=\"rgb(0,0,0)\">".$backup1."</text>";
301 //text
302             }
303     return $txt;
304 }
```

```
$txt = "<svg height=\"".($a[0]*2+10)."\"
width=\"".($a[1]*2+10)."\"\>";
$txt .=  "<polyline
points=\"".$a[1].",".($a[0]-$a[0]+10)."
".$a[1].",".(2*$a[0]+10)."\"

style=\"fill:none;stroke:rgb(0,0,0);stroke-
width:3\" />
  <polyline points=\"".($a[1]-
$a[1]).",".$a[0]." ".(2*$a[1]).",".$a[0]."\"

style=\"fill:none;stroke:rgb(0,0,0);stroke-
width:3\" />";  //+
$txt .=  "<polygon
points=\"".$a[1].",".($a[0]-$a[0])."
".($a[1]-10).",".($a[0]-$a[0]+10)."
".($a[1]+10).",".($a[0]-$a[0]+10)."\"

style=\"fill:rgb(0,0,0);stroke:rgb(0,0,0);str
oke-width:1\" />

  <polygon points=\"".$a[1].",".(2*$a[0])."
".($a[1]-10).",".(2*$a[0]-10)."
".($a[1]+10).",".(2*$a[0]-10)."\"

style=\"fill:rgb(0,0,0);stroke:rgb(0,0,0);str
oke-width:1\" />

  <polygon points=\"".(2*$a[1]).",".$a[0]."
".(2*$a[1]-10).",".($a[0]-10)." ".(2*$a[1]-
10).",".($a[0]+10)."\"

style=\"fill:rgb(0,0,0);stroke:rgb(0,0,0);str
oke-width:1\" />

  <polygon points=\"".(2*$a[1]-
2*$a[1]).",".$a[0]." ".($a[1]-
$a[1]+10).",".($a[0]-10)." ".($a[1]-
$a[1]+10).",".($a[0]+10)."\"

style=\"fill:rgb(0,0,0);stroke:rgb(0,0,0);str
oke-width:1\"/>"; //arrows
```

```
347 $txt = setrange($a, $range, $txt);
348 return $txt;
349 }
```

The **drawCS** function is to create a coordinate system which can have any desired rate. This can be arranged through the **setrange** function inside the **drawCS** function in *line 212*. The range can be manipulated to have a wider or narrow view for a specific data.

```
350 function drawfunction(){ //this function is
351 for drawing any fucntion. Right now it's set
352 to y=x*x
353     $a = array(500, 500);
354     $range = (50);
355     $txt = $this->drawCS($a, $range);
356 //drawing coordinate system
357     $elements = null;
358     for($i = -$a[0]; $i <= $a[0]; $i=$i+0.1){
359 //draw a function with this for loop
360         $x[$i]= $i;
361         $y[$i]= $i*$i/10; //user can draw
362 anything you want by changing this line.
363 Right now it's set to y=x*x/10
364         $r = $this->ifd($x[$i], $y[$i], $a);
365 //using ifd function
366         $storex[$i] = $r[0];
367         $storey[$i] = $r[1];
368         $elements .= "<circle
369 cx=\"".$storex[$i]."\" cy=\"".$storey[$i]."\"
370 r=\"1\" stroke=\"rgb(0,0,0)\" stroke-
371 width=\"2\" fill=\"rgb(0,0,0)\" />";
372 //drawing the circles in store and creating a
373 shape
374 }
375     echo $txt;
376     echo $elements;
377 }
```

The **drawfunction** function is to visualize equations in coordinate system. It calls **drawCS** function as seen in *line 355* and ads an equation on the coordinate system as an element. This process happens by creating many small circles narrowly to conceive a linear shape on the coordinate system.

If the user applies the code below,

```
$sv = new mysvg();
$sv->drawfunction();
```

The following graph can be expected which represents (y=$x^2/10$ -300): (see Figure 9)

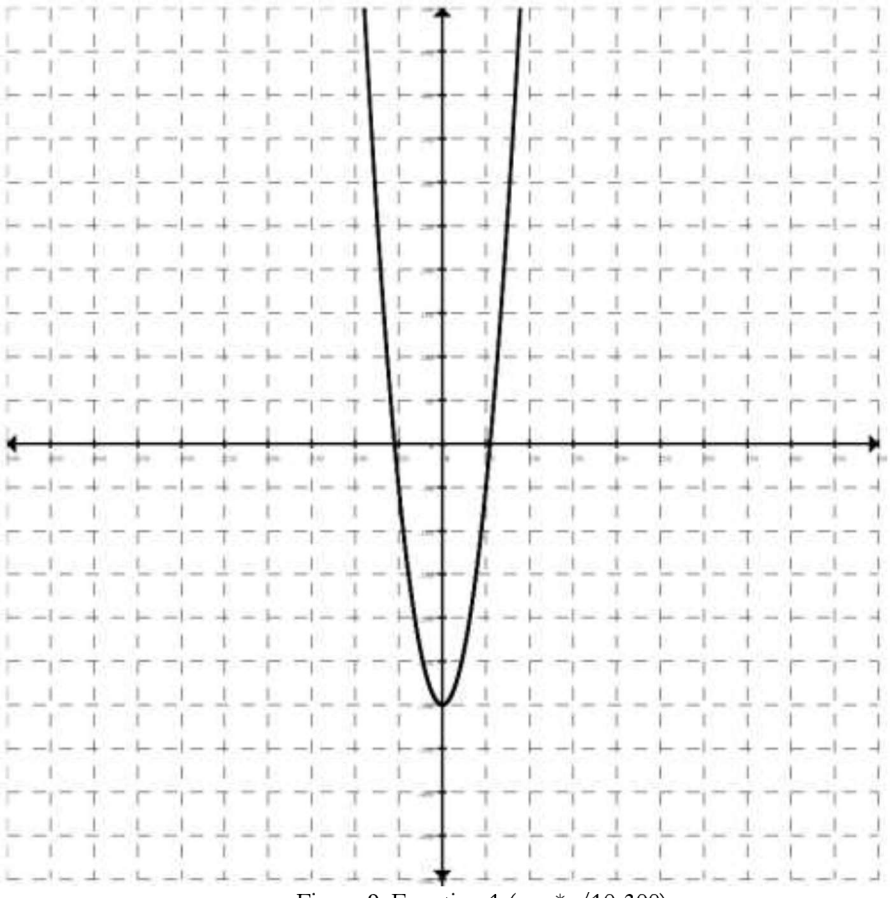

Figure 9: Function 1 (y=x*x/10-300)

If user changes the *line 361* as;

```
361         $y[$i]= $i*$i*$i/10; //user can draw
362 anything you want by changing this line.
363 Right now it's set to y=x*x*x/10
```

The following graph can be expected which represents y=$x^3/10$: (see Figure 10)

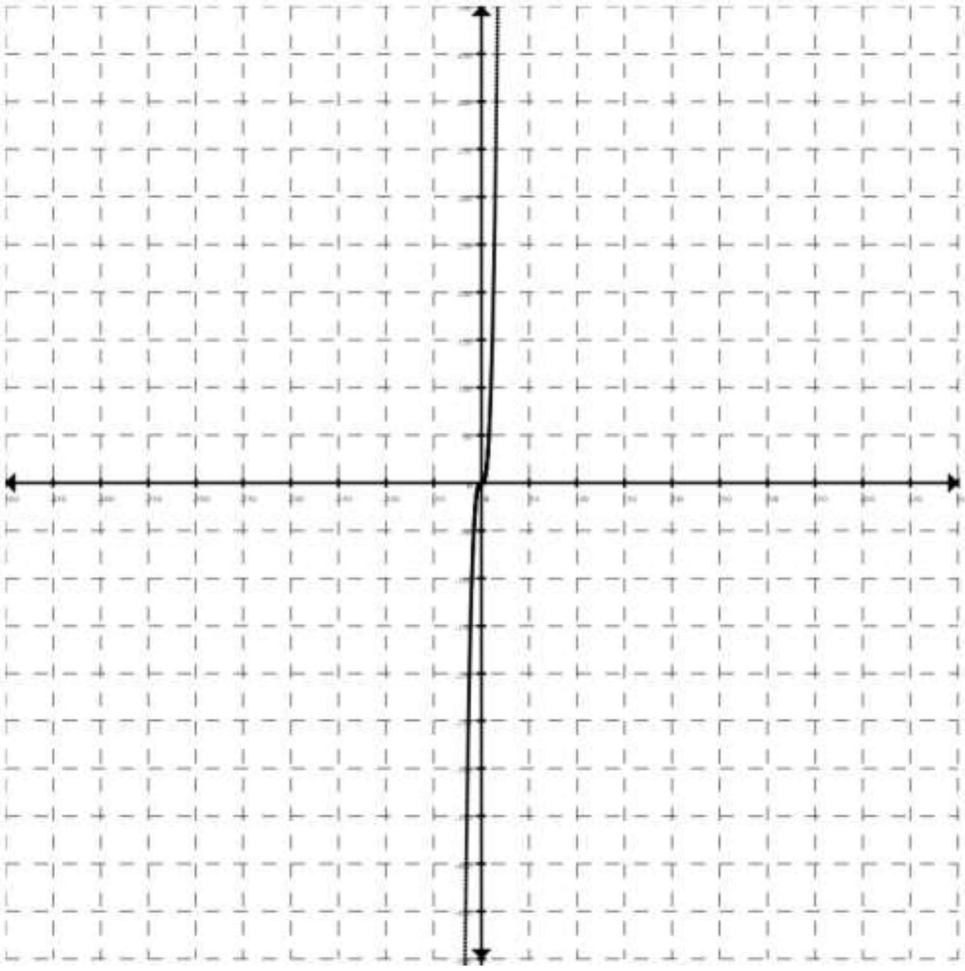

Figure 10: Function 2 (y=x*x*x/10)

```
function drawpoint($pointx, $pointy, $ln){
//this fucntion draws a point in coordinate
system
    $a = array(1000,1000);
    $elements = null;
    $elements .= "<svg
height=\"".($a[0]*2+10)."\"
width=\"".($a[1]*2+10)."\"\>"; //creaets the
svg environment
function linearregression($pointx, $pointy,
$a){ //this fucntion is to create a
additional line that shows the linear
agression of point groups.
function ifd($x, $y, $a){ //this function
pinpoints the location of input in
coordinates system
      if($x >= 0){
        if($y >= 0){
          $xx= abs($x)+$a[1];
          $yy= abs($y)+$a[0]-2*abs($y);
        }else if($y <= 0){
          $xx= abs($x)+$a[1];
          $yy= abs($y)+$a[0];
        }
       }
     else if($x <= 0){
        if($y >= 0){
            $xx= abs($x)+$a[1]-2*abs($x);
            $yy= abs($y)+$a[0]-2*abs($y);
        }else if($y <= 0){
            $xx= abs($x)+$a[1]-2*abs($x);
            $yy= abs($y)+$a[0];
     }
        }
     return array($xx, $yy);
    }

    for($i=0; $i <= sizeof($pointx)-1;
$i=$i+1){
        $xy[$i] = $pointx[$i]*$pointy[$i];
        $x2[$i] = $pointx[$i]*$pointx[$i];
        $y2[$i] = $pointy[$i]*$pointy[$i];
```

```
420     }
421         $sumx = array_sum($pointx);
422         $sumy = array_sum($pointy);
423         $sumxy = array_sum($xy);
424         $sumx2 = array_sum($x2);
425         $sumy2 = array_sum($y2);
426         $num1= ($sumy*$sumx2 - $sumx*$sumxy)
427 / (sizeof($pointx)*$sumx2 - $sumx*$sumx);
428         $num2= (sizeof($pointx)*$sumxy -
429 $sumx*$sumy) / (sizeof($pointx)*$sumx2 -
430 $sumx*$sumx);
431         $x0 = $pointx[0];
432         $x1 = $pointx[sizeof($pointx)-1];
433         $y0 = $num1+$x0*$num2;
434         $y1 = $num1+$x1*$num2;
435         $mm0 = ifd($x0, $y0, $a); //calling
436 the ifd function which has been stated
437 earlier
438         $mm1 = ifd($x1, $y1, $a); //calling
439 the ifd function which has been stated
440 earlier
441         $elements = "<polyline
442 points=\"".$mm0[0].",".$mm0[1]."
443 ".$mm1[0].",".$mm1[1]."\"
444
445 style=\"fill:red;stroke:red;stroke-width:4\"
446 />"; //creates a red line
447         return $elements;
448     }
449         $range =50;
450         $txt = $this->drawCS($a, $range);
451         for($i = 0; $i <= sizeof($pointx)-1;
452 $i=$i+1){ //for drawing multiple points
453             $x[$i]= $pointx[$i];
454             $y[$i]= $pointy[$i]; //you can
455 draw anything you want by changing this
456
457             $r = $this->ifd($x[$i], $y[$i],
458 $a);
459             $storex[$i] = $r[0];
460             $storey[$i] = $r[1];
461             $elements .= "<circle
```

```
462 cx=\"".$storex[$i]."\" cy=\"".$storey[$i]."\"
463 r=\"6\"
464 onmouseover=\"evt.target.setAttribute('r',
465 '7');\"
466 onmouseout=\"evt.target.setAttribute('r',
467 '6');\" stroke=\"rgb(0,0,0)\" stroke-
468 width=\"1\" fill=\"rgb(0,0,0)\" />"; //with
469 this line the program creates intereactive
470 dots which is increase with a mouse hover
471         }
472             if($ln==1){
473             $elements .=
474 linearregression($pointx, $pointy, $a); //if
475 user sets $ln as 1 or 0 the linear agression
476 function kicks in
477             }
478     echo $txt;
479     echo $elements;
480 }
```

The **drawpoint** function is to create scatter plots within desired range. The x and y values should presented to code with different arrays which contains them separately to perform a successful operation. An additional operation which can be achieved with this function is, visualizing linear regression of scatter plots with the **linearagression** function on *line 387*. Also the the size of points are shifting by cursor movement of user for creating more interactive environment. This operation is carried out by **onmouseover** command on *line 464*.

If the user applies the code below,

```
$sv = new mysvg();
$pointx1 = array(-800, -500, 700);
$pointy1 = array(-350, 300, 550);
$sv->drawpoint($pointx1, $pointy1, $ln=0);
```

The following graph can be expected: (see Figure 11)

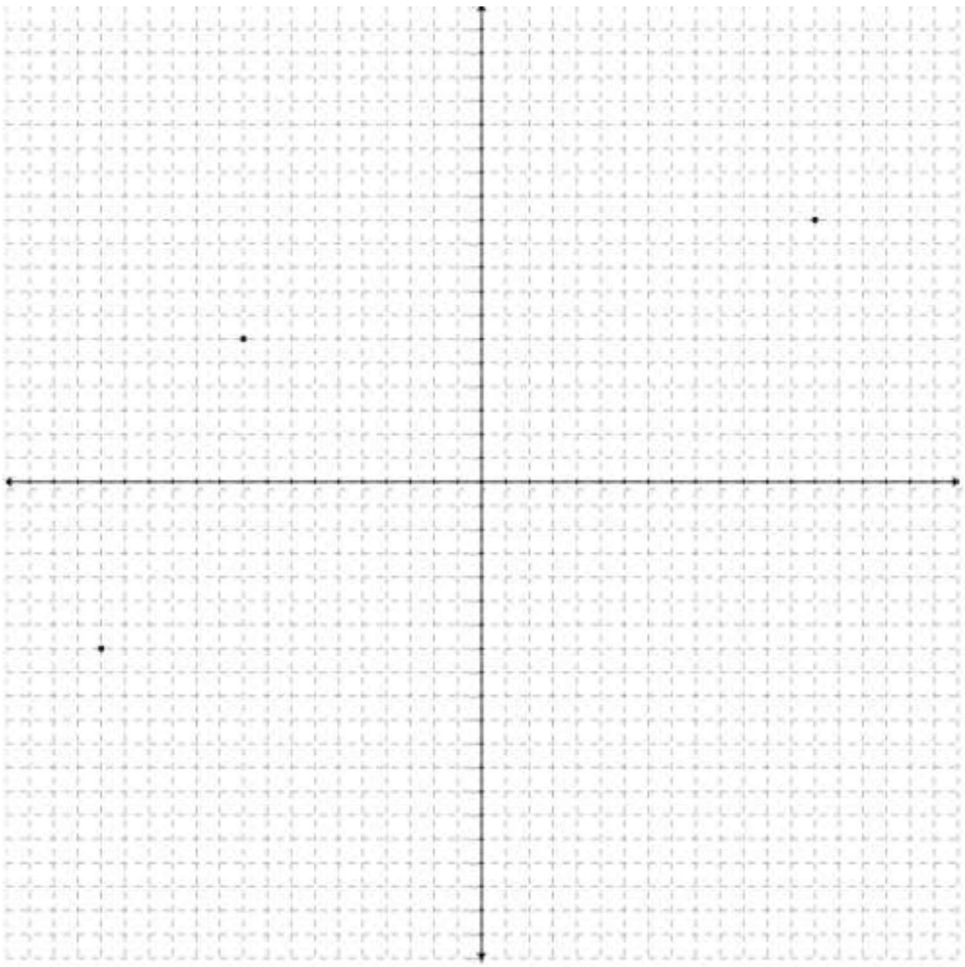

Figure 11: Scatter plot

If the user applies the code below where $ln value is set as equal to 1,

```
$sv = new mysvg();
$pointx1 = array(-800, -500, 700);
$pointy1 = array(-350, 300, 550);
$sv->drawpoint($pointx1, $pointy1, $ln=1);
```

The following graph can be expected: (see Figure 12)

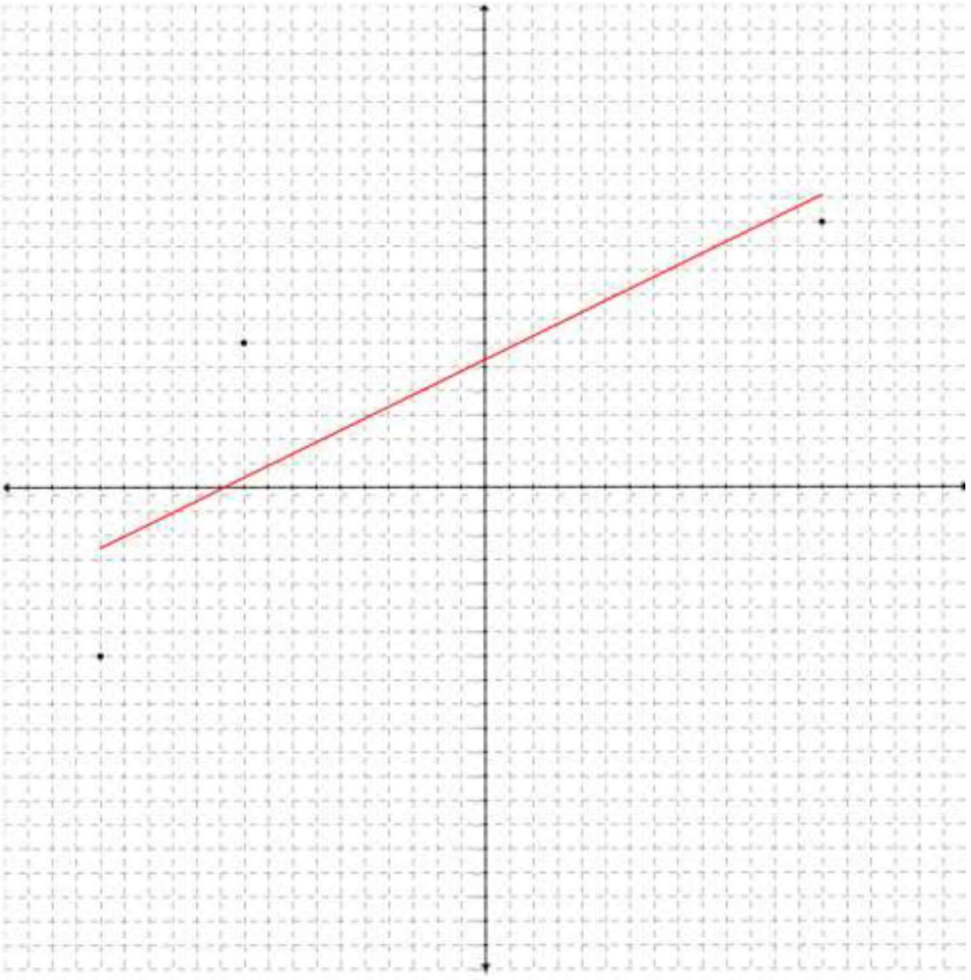

Figure 12: Scatter plot with linear regression

```
481 function drawline($pointx, $pointy){ //this
482 function draw lines from specific points in
483 coordinate system
484     static $txt = null;
485     $a = array(500, 500);
486     $range = (50);
487     if ($txt === null) {
488      $txt = $this->drawCS($a, $range);
```

```
489 //drawing the coordinate system
490      echo $txt;
491     }
492     $color = $this->randomcolor(); //calling
493 the randomcolor function
494     $elements = null;
495     $elements .= "<svg
496 height=\"".($a[0]*2+10)."\"
497 width=\"".($a[1]*2+10)."\"\>";
498     for($i = 0; $i <= sizeof($pointx)-1;
499 $i=$i+1){ //draw
500         $x[$i]= $pointx[$i];
501         $y[$i]= $pointy[$i];
502         $r = $this->ifd($x[$i], $y[$i], $a);
503         $storex[$i] = $r[0];
504         $storey[$i] = $r[1];
505         if($i>=1){
506             $elements .= "<polyline
507 points=\"".$storex[$i].",".$storey[$i]."
508 ".$storex[$i-1].",".$storey[$i-1]."\"
509
510 style=\"fill:".$color.";stroke:".$color.";stro
511 ke-width:3\" />";}
512             $elements .= "<circle
513 cx=\"".$storex[$i]."\" cy=\"".$storey[$i]."\"
514 r=\"4\"
515 onmouseover=\"evt.target.setAttribute('r',
516 '5');\"
517 onmouseout=\"evt.target.setAttribute('r',
518 '4');\" stroke=black stroke-width=\"0.5\"
519 fill= ".$color." />";
520         }
521     echo $elements;
522 }
```

The **drawline** function is to create line charts. Multiple lines can be added as elements on the same plane for making comparison through a **if** command on *line 487*, and dots that's representing the specific points on the lines are interactive like the points on **drawpoint** function.

If the user applies the code below,

```
$sv = new mysvg();
$pointx1 = array(50, 100, 200, 350, 450);
$pointy1 = array(50, 150, 100, 150, 80);
$pointx2 = array(100, 200, 250, 300, 350);
$pointy2 = array(50, 200, 100, 200, 250);
$sv->drawline($pointx1, $pointy1);
$sv->drawline($pointx2, $pointy2);
```

The following graph can be expected: (see Figure 13)

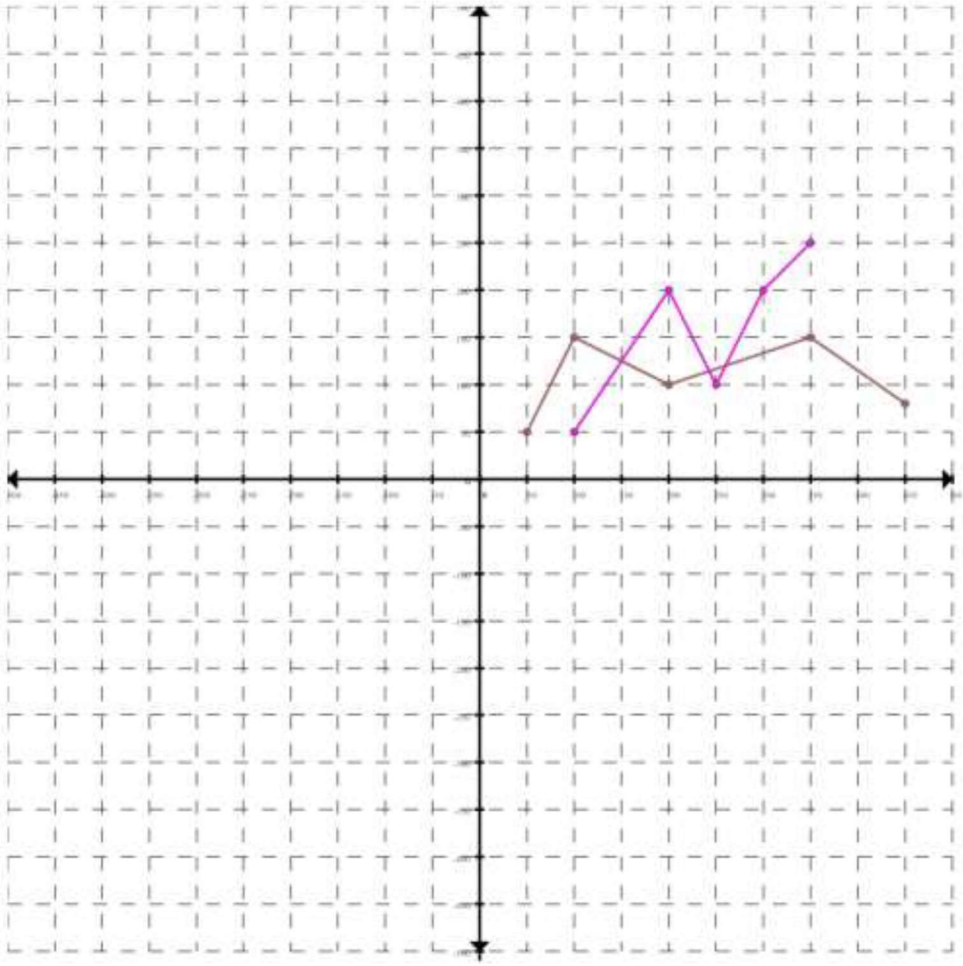

Figure 13: Line chart

```
523 function drawradarchart($names, $pointx,
524 $pointy){
525     $a = array(500, 500);
526     $m = array(250, 250); //center
527     $text = null;
528     static $txt = null;
529     if ($txt === null) { //this line helps to
530 draw multiple chart in one environment
531         $txt    =    $this->drawRDR($a,    $m);
532 //calling back RDR fucntion for skeleton of
533 radar chart
534
535         echo $txt;
536     }
537     $elements = null;
538     $elements .= "<svg
539 height=\"".($a[0]*2+10)."\"
540 width=\"".($a[1]*2+10)."\"\>";
541     $text = null;
542     $color= $this->randomcolor();
543     for($i = 0; $i <= sizeof($pointx)-1; $i++)
544 {
545         $angle = $i*2*M_PI/sizeof($pointx);
546         $sin = abs(sin($angle));
547         $cos = abs(cos($angle));
548         $numx = (200*$sin);
549         $numy = (200*$cos);
550         $ax = array();
551         $ay = array();
552         if (($angle>=0) && ($angle<=
553 M_PI/2))//every location should set
554 differently according to axis because every
555 location has a different angle
556             {
557             $elements .=  "<text text-
558 anchor=\"middle\" x=\"".($m[0]+$numx)."\"
559 y=\"".($m[1]-$numy)."\" font-size=\"20\"
560 fill=\"rgb(0,0,0)\">".$pointx[$i]." </text>";
```

```
561 //attributes
562             $ax[$i] = ($m[0]+$numx);
563             $ay[$i] = ($m[1]-$numy);
564             }
565         else if (($angle>=M_PI/2) && ($angle<=
566 M_PI))
567             {
568             $elements .=  "<text text-
569 anchor=\"middle\" x=\"".($m[0]+$numx)."\"
570 y=\"".($m[1]+$numy)."\" font-size=\"20\"
571 fill=\"rgb(0,0,0)\">".$pointx[$i]."</text>";
572             $ax[$i] = ($m[0]+$numx);
573             $ay[$i] = ($m[1]+$numy);
574             }
575         else if (($angle>=M_PI) && ($angle<=
576 3*M_PI/2))
577             {
578             $elements .=  "<text text-
579 anchor=\"middle\" x=\"".($m[0]-$numx)."\"
580 y=\"".($m[1]+$numy)."\" font-size=\"20\"
581 fill=\"rgb(0,0,0)\">".$pointx[$i]."</text>";
582             $ax[$i] = ($m[0]-$numx);
583             $ay[$i] = ($m[1]+$numy);
584             }
585         else if (($angle>=3*M_PI/4) &&
586 ($angle<= 2*M_PI))
587             {
588             $elements .=  "<text text-
589 anchor=\"middle\" x=\"".($m[0]-$numx)."\"
590 y=\"".($m[1]-$numy)."\" font-size=\"20\"
591 fill=\"rgb(0,0,0)\">".$pointx[$i]."</text>";
592             $ax[$i] = ($m[0]-$numx);
593             $ay[$i] = ($m[1]-$numy);
594             }
595         $elements .=  "<circle
596 cx=\"".($m[0]+($ax[$i]-
597 $m[0])*$pointy[$i]/10)."\"
598 cy=\"".($m[1]+($ay[$i]-
599 $m[1])*$pointy[$i]/10)."\" r=\"3\"
600 onmouseover=\"evt.target.setAttribute('r',
601 '4');\"
602 onmouseout=\"evt.target.setAttribute('r',
```

```
603 '3');\" stroke=\"black\" stroke-width=\"0.5\"
604 fill=\"".$color."\"/>";
605         $text .= "".($m[0]+($ax[$i]-
606 $m[0])*$pointy[$i]/10).",".($m[1]+($ay[$i]-
607 $m[1])*$pointy[$i]/10)." "; //to make text for
608 next line
609     }
610     $elements .= "<rect x=\"".($m[0]+300)."\"
611 y=\"".($m[1])."\" width=\"250\"
612 height=\"".(25+(sizeof($names)-1)*30)."\"
613 style=\"fill:none;stroke-
614 width:2;stroke:rgb(0,0,0)\" />"; //creating
615 big rentangle
616     for($i = 0; $i <= sizeof($names)-1; $i++)
617 {
618         $elements .= "<rect
619 x=\"".($m[0]+350)."\"
620 y=\"".(5+$m[1]+$i*30)."\" width=\"30\"
621 height=\"15\" style=\"fill:".$color.";stroke-
622 width:1;stroke:rgb(0,0,0)\" />"; //small
623 rectangle
624         $elements.= "<text
625 x=\"".($m[0]+400)."\"
626 y=\"".(20+$m[1]+($i)*30)."\"
627 fill=\"rgb(0,0,0)\"\">=
628 ".($names[$i])."</text>"; //text
629     }
630     $elements .= "<polygon
631 points=\"".$text."\"
632 style=\"fill:".$color.";opacity:0.35;stroke:".
633 $color.";stroke-width:1\" />";
634     echo $elements;
635 }
```

The **drawradar** function is to create radar charts which contains multiple subjects. The practical side of this function is variability of attributes which is used on the function. The number of attributes can be set by the user and the locations of attributes on the circle will be placed equally. This is achieved by the trigonometric placements of attributes through the operations between *lines 551 and 609*.

The dots are interactive just like in **drawpoint** function with **onmouseover** operation on *line 600*.

If the user applies the code below,

```
$sv = new mysvg();
$attributes = array('Agility', 'Speed', 'Strength',
'Intelligence', 'Durability');
$names = array('Iron Man', 'Hulk');
$sv->drawradarchart($names, $attributes, $pointx);
$sv->drawradarchart($names, $attributes, $pointy);
```

The following graph can be expected: (see Figure 14)

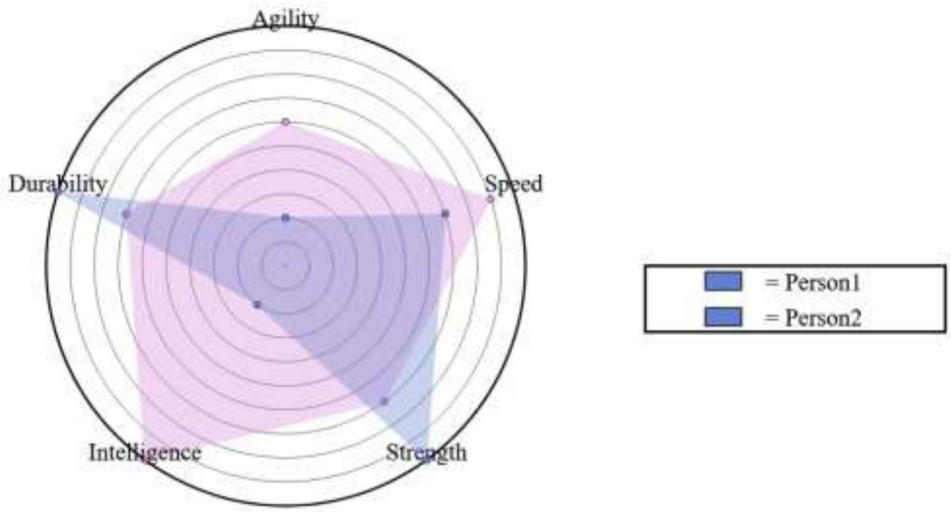

Figure 14: Radar chart

```
636 function drawbarchart($names, $pointx,
637 $pointy){
638     $a = array(500, 500);
639     $ak = ($a[0]/(sizeof($pointx)));
640     function absolute($point){ //get max of
641 abolute values in an array
```

```
642     for($i=0; $i <= sizeof($point)-1;
643 $i=$i+1){
644     $absolute[$i] = abs($point[$i]);
645     }
646     return $absolute;
647     }
648 function randomcolor(){ //getting a random
649 color from 7*7*24 types
650     $an = rand(0,23);
651     $bn = rand(2,8);
652     $cn = rand(2,8);
653     $color=
654 'hsl('.($an*15).','.($bn*10).'%,'.($cn*10).'%
655 )';
656     return $color;
657     }
658 function drawEN($a, $ak, $pointx, $pointy,
659 $names){ //coordinate system
660     $txt = "<svg height=\"".($a[1]*3)."\"
661 width=\"".($a[0]*3)."\"\>";
662     $txt .= "<polyline
663 points=\"".(100).",".($a[1]-$a[1]+10)."
664 ".(100).",".($a[1])."\"
665
666 style=\"fill:none;stroke:rgb(0,0,0);stroke-
667 width:3\" />
668                 <polyline
669 points=\"".(100).",".$a[1]."
670 ".($a[0]+100+$ak).",".$a[1]."\"
671
672 style=\"fill:none;stroke:rgb(0,0,0);stroke-
673 width:3\" />";  //+
674
675     $txt .=  "<polygon
676 points=\"".(100).",".($a[1]-$a[1])." ".(100-
677 10).",".($a[1]-$a[1]+10)."
678 ".(100+10).",".($a[1]-$a[1]+10)."\"
679
680 style=\"fill:rgb(0,0,0);stroke:rgb(0,0,0);str
681 oke-width:1\" />
682
683                 <polygon
```

```
684 points=\"".($a[0]+100+10+$ak).",".$a[1]."
685 ".($a[0]+100+$ak).",".($a[1]-10)."
686 ".($a[0]+100+$ak).",".($a[1]+10)."\"
687
688 style=\"fill:rgb(0,0,0);stroke:rgb(0,0,0);str
689 oke-width:1\" />"; //arrows
690     $m = 0;
691     for($i = 0; $i <= sizeof($pointx)-1;
692 $i=$i+1){
693         $percentage =
694 $pointy[$i]/array_sum(absolute($pointy));
695         $wow = $a[1]-
696 ($a[1]*(95/100)*$percentage); // 95 for full
697 values
698         $vv = randomcolor();
699         if($pointy[$i]<0){ //only appear if
700 theres negative values
701
702             $txt .= "<polyline
703 points=\"".(100).",".($a[1])."
704 ".(100).",".(2*$a[1])."\"
705
706 style=\"fill:none;stroke:rgb(0,0,0);stroke-
707 width:3\" />
708
709                     <polygon
710 points=\"".(100).",".(2*$a[1])." ".(100-
711 10).",".(2*$a[1]-10)."
712 ".(100+10).",".(2*$a[1]-10)."\"
713
714 style=\"fill:rgb(0,0,0);stroke:rgb(0,0,0);str
715 oke-width:1\" />";
716
717             $txt .= "<rect x=\"".(100+$ak-
718 $ak/4+$ak*$i)."\" y=\"".($a[1])."\"
719 width=\"".($ak/2)."\" height=\"".($wow-
720 $a[1])."\"
721                 style=\"fill:".$vv.";stroke-
722 width:2;stroke:rgb(0,0,0)\" />";
723         }
724
725             $txt .= "<polyline
```

```
726 points=\"".(100-5).",".$wow."
727 ".(100+5).",".$wow."\"
728
729 style=\"fill:none;stroke:rgb(0,0,0);stroke-
730 width:3\" />
731
732                     <polyline
733 points=\"".(100).",".($wow)."
734 ".($a[0]+100+$ak).",".($wow)."\"
735
736 style=\"fill:none;stroke:rgb(0,0,0);stroke-
737 width:1\" stroke-dasharray=\"15,15\" d=\"M5
738 20 l215 0\" />"; //dotted lines
739
740             $txt .= "<rect x=\"".(100+$ak-
741 $ak/4+$ak*$i)."\" y=\"".($wow)."\"
742 width=\"".($ak/2)."\" height=\"".($a[1]-
743 $wow)."\"
744
745 style=\"fill:".$vv.";stroke-
746 width:2;stroke:rgb(0,0,0)\" />";
747             $txt .= "<text text-
748 anchor=\"end\" x=\"".(100-10)."\"
749 y=\"".($wow+5)."\"
750 fill=\"rgb(0,0,0)\">".($pointy[$i])."</text>"
751 ;
752                     if($pointy[$i]>=0){
753                         $txt .= "<text text-
754 anchor=\"middle\" x=\"".(100+$ak*$i+$ak)."\"
755 y=\"".($a[1]+20)."\"
756 fill=\"rgb(0,0,0)\">".($pointx[$i])."</text>"
757 ; //text
758                     }else{ //negative
759                         $txt .= "<text text-
760 anchor=\"middle\" x=\"".(100+$ak*$i+$ak)."\"
761 y=\"".($a[1]-10)."\"
762 fill=\"rgb(0,0,0)\">".($pointx[$i])."</text>"
763 ; //text
764                     }
765             $txt .= "<rect x=\"850\"
766 y=\"300\" width=\"250\"
767 height=\"".(25+(sizeof($pointx)-1)*30)."\"
```

```
768 style=\"fill:none;stroke-
769 width:2;stroke:rgb(0,0,0)\" />"; //big
770 rentangle
771             $txt .= "<rect x=\"900\"
772 y=\"".(305+($i)*30)."\" width=\"30\"
773 height=\"15\" style=\"fill:".$vv.";stroke-
774 width:1;stroke:rgb(0,0,0)\" />"; //small
775 rextangle
776             $txt.= "<text x=\"945\"
777 y=\"".(290+($i+1)*30)."\"
778 fill=\"rgb(0,0,0)\"\">=
779 ".$names[$i]."</text>"; //text
780             }
781     echo $txt;
782     }
783     drawEN($a, $ak, $pointx, $pointy,
784 $names);
785 }
786 }
```

The **drawbarchart** function is to create simple bar charts with different individual colors. The rectangle images getting placed on special axis which is created by **drawEN** function in *line 658*.

If the user applies the code below,

```
$sv = new mysvg();
$names = array('Apples', 'Peaches', 'Bananas',
'Strawberries', 'Lemons', 'Carrots');
$pointx1 = array(50, 100, 200, 350, 450);
$pointy1 = array(50, 150, 100, 150, 80);
```

The following graph can be expected: (see Figure 15)

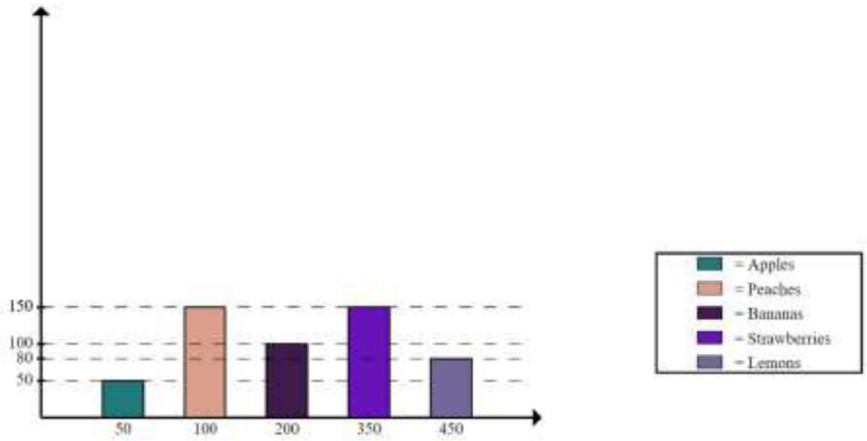

Figure 15: Bar chart

Chapter IV

EVALUATION AND DISCUSSION

4.1 Performance

We can observe that the result is a multifunctional library which is capable of creating multi-variety of graphs.

4.2 Suggestions for Further Development

It should be stated that during research stages, the intended result was mostly achieved except for more interactive visuals. Some of the intended interactive elements couldn't implemented because of time scheduling conflicts but this is a negligible deficiency because overall results are found satisfactory compared to ideal intentions. Hopefully this additions can be added in any further developments in similar practices.

Chapter V

CONCLUSION

5.1 Obtained Results

The program is capable of creating seven different visualizations which is convenient with the intended target. User can apply different sorts of data for the desired graph. With applying different data with their own attributes, it's possible to create pie chart, doughnut chart, radar chart, bar chart, line chart, a scatter plot and it's possible to create various functions on a coordinate system.

5.2 Conclusion

In the end of research, general self-experience about PHP language has been increased and the library which has ability to create various results is ready to share with other users. This library can be used for basic experimental intentions and educational reasons as well as vocational purposes.

REFERENCES

[1] Matange, S., Heath, D.: Statistical Graphics Procedures by Example: Effective Graphs Using SAS. SAS Institute, NC, USA, 2011

[2] Tatroe, K., Macintyre, K.: Programming PHP: Creating Dynamic Web Pages 3th Edition. O'Reily, CA, USA, 2013

[3] user: lsn, [online] 2020 [cit. 19.06.2020] https://www.lsnsoft.com/all-about-web-technologies/

[4] Shubham Upadhyay, [online] 2020 [cit. 19.06.2020] https://www.quora.com/What-are-the-characteristics-of-PHP-variables

[5] Paliavi Godse, [online] 2020 [cit. 19.06.2020] https://www.milesweb.com/blog/hosting/web-server-types-web-servers/

[6] user: Staticshakedown, [online] 2020 [cit. 19.06.2020] https://en.wikipedia.org/wiki/Scatter_plot

[7] N/A, [online] 2020 [cit. 19.06.2020] https://www.basic-mathematics.com/bar-graphs.html

[8] Patrick Fuller, [online] 2020 [cit. 19.06.2020] https://medium.com/@patrickbfuller/line-plot-7b4068a3a9fc

[9] N/A, [online] 2020 [cit. 19.06.2020] https://moqups.com/templates/charts-graphs/pie-donut-chart/simple-pie-chart/