CHAPTER V

IMPLEMENTATION AND TESTING

5. 1 Implementation

First, we need to activate XAMPP. Log into the terminal, then type: **/ opt / lampp / lampp start**. Once XAMPP running successfully, then we need to make sure that the internet connection on our computers already exist (as it will be used for the Google API).

Then open a browser and type in: localhost / (foldername) TA / index.php. Then select how many cities you want. Here is the Google API (charted map and get distance) and process enumeration method:

5.1.1 Step 1 - Show The Map and Select It

User can look the map and select cities to be passed. It can use by Google API maps.

```
var myOptions = {
        center: new google.maps.LatLng(-7.0798379 , 110.3742433),
        zoom: 8,
        mapTypeId: google.maps.MapTypeId.ROADMAP
      };
```

Google API program code used to display a map (Java exact).

```
document.getElementById('lat').value=location.lat();
document.getElementById('lng').value=location.lng();
getAddress(location);
```

Google API program code used to select cities on the map from latitude and longitude for get a location.

Function to show the map and the user can select it. This function need a connected to internet.

5.1.2 Step 2 - Get Distance From The Cities Selected

Function can get distance from the cities selected. This is useful for data calculation process further.

```
var glatlng1 = new GLatLng(location1.lat, location1.lon);
var glatlng2 = new GLatLng(location2.lat, location2.lon);
var miledistance = glatlng1.distanceFrom(glatlng2, 3959).toFixed(1);
var kmdistance = (miledistance * 1.609344).toFixed(1);
```

Program code that used the Google API to get the distance of each city selected. From latitude and longitude each city we can get a distance inter-city.

Function to get distance from the selected city on the map.

5.1.3 Step 3 - Displaying data in matrix form

This Function aims to display data in the form of a matrix.

I use a two-dimensional array to help map in the form of a matrix so that the data can be easily filled and easier to read / called.

here I put the data into the inter-city distance matrix / array of two-dimensional.

```
$y=0;
FOR($i=1 ; $i <= $banyakkota ; $i++)
{
FOR($j=1 ; $j <= $banyakkota ; $j++)
{
        IF($i == $j)
        {
                $objek->set_kota($i,$j,"X");
        }
        ELSE
        {
                $y=$y+1;
                $objek->set_kota($i,$j,$jarak[$y]);
        }
}
}
```

The above program code above is useful for entering data into the distance between cities 2 dimensional array. $ object-> set_kota ($ i, $ j, "X") -> $ object-> set_kota (row, column, DATA).

to retrieve data :

```
echo "<table border=3 bgcolor='lightblue' bordercolor='red'>";
FOR($i=1 ; $i <= $banyakkota ; $i++)
{
   echo "<tr>";
   FOR($j=1 ; $j <= $banyakkota ; $j++)
   {
                IF($i == $j)
                {
```

```
                    echo "<td><font color='red'><center>    <h2>".$objek-
>get_kota($i,$j)."</h2></center></font></td>";
                }
                ELSE
                {
                    echo "<td><font color='blue'><center>    <h2>".$objek-
>get_kota($i,$j)."</h2></center></font></td>";
                }
    }
    echo "</tr>";
}
echo "</table>";
```

The above code will display the data in the form of inter-city distance matrix table.

$objek->get_kota($i,$j) -> $objek->get_kota(baris,kolom)

This Function aims to display data in the form of a matrix.

5.1.4 Step 4 - Random Cities That Passed

This Function aims to random cities that passed. With this function we can comparing city and finding the city passed its minimum point.

eg 4 cities: 1 2 3 4.

the end of the initial 1 1. Calculated by chance

(N-1)! = (4-1)! = 6 opportunities.

1 2 3 4 1 1 3 2 4 1 1 4 2 3 1

1 2 4 3 1 1 3 4 2 1 1 4 3 2 1

after that find the distance of each route and find the shortest distance. The process is complete.

```
$a=0;   $b=0;

$random = rand(2,$banyakkota);
while($a != $banyak)
        {
                $a=0;
                $random = rand(2,$banyakkota);
                FOR($x=1 ; $x <= $banyak ; $x++)
                {
                                $tampung = $rdm[$x];

                        IF($tampung != $random)
                        {
                                                        $a=$a+1;
                                }
                                }
                        }
                        $rdm[$j-1]=$random;
                        $objek->set_kotaakhir($i,$j,$random);
```

These randomized, but the city limits to the city 2 to city last (because the city has become one beginning and end of the route). For further randomized but the city should not be the same as the previous route, that route does not pass through the city that same city.

This Function aims to randomming cities. It can make the program determines the city passed to the random and without passing through the same city twice.

5.1.5 Step 5 - Calculates Total Distance

This Function aims to calculate the total distance of the town randomly selected to pass.

```
FOR($j=1 ; $j <= $tabel ; $j++)
{
        IF($j == $tabel)
        {
                //echo "TOTAL JARAK BARIS KE-".$i." = ".
            $jaraktotal."<br/>";
                $objek->set_kotaakhir($i,$tabel,$jaraktotal);
        }
        ELSE IF($j == $tabel-1)
        {
        }
        ELSE
        {
                $a = $j+1;
                 $tampung1 = $objek->get_kotaakhir($i,$j);
                  $tampung2 = $objek->get_kotaakhir($i,$a);

            $tampungjrk = $objek->get_kota($tampung1,$tampung2);

                 $jaraktotal = $jaraktotal + $tampungjrk;
        }
    }
```

Of functions over how to find the total distance of each iteration is to add up the distance between his city. Such as there are permutations 1 3 4 2 1. Then first thing to do is take the city distance 1-3, then 3-4 plus the distance the city, then the city plus the distance 4-2, 2-1 and then added distance of the city. So can get total distance of each iteration route.

5.1.6 Step 6 - Search The Minimum Distance

This Function aims to search for the minimum distance.

The minimum distance accommodated in a variable and if there is a minimum distance over the new variable will accommodate the new minimum distance.

```
        $jarakminimum = 1000000;
        $tampung = 0;
        $titik=0;
        FOR($i=1 ; $i <= $permutasi ; $i++)
        {
                $tampung = $objek->get_kotaakhir($i,$tabel);
        IF($tampung < $jarakminimum)
        {
                $jarakminimum = $tampung;
                $titik = $i;
        }
}
```

This Function aims to search for the minimum distance.

The trick is to first assign in variable $ jarakminimum filled with tremendous value. Then in each iteration for and viewed. value at $ jarakminimum compared with the distance values in each iteration. If the value of the distance in the iteration is less than $ jarakminimum then the value at $ jarakminimum replaced by the value of the distance in the iteration. This process is repeated until all the iterations completed. Thus be obtained with the shortest distance.

5.2 Testing

To know how far the tolerance of these applications with input different cities, the process generated by this application will see the truth. These are the testing result table:

Table 5.1 Testing Table

No	Cities Selected	Application Result	Succesfully
1	4 City = Semarang, Banjarnegara, Yogyakarta, Kudus	Semarang – Kudus – Yogyakarta – Banjarnegara -- Semarang	100%
2	5 City = Pati , Klaten, Semarang, Yogyakarta, Banjarnegara	Pati – Klaten – Yogyakarta – Banjarnegara – Semarang -- Pati	100%
3	6 City = Semarang, Yogyakarta, Sragen, Pekalongan, Blora, Ponorogo	Semarang – Blora – Ponorogo – Sragen – Yogyakarta – Pekalongan -- Semarang	100%
4	7 City = Yogyakarta, Kudus, Pekalongan, Pati, Bandung, Banjarnegara, Lembang	Yogyakarta – Pati – Kudus – Pekalongan – Lembang – Bandung – Banjarnegara -- Yogyakarta	100%
SUCCESFULLY			100%

From testing result, we can see that the application runs smoothly. The town initially randomly selected city is now obtained optimum results which will be passed in order to obtain the minimum distance. Application runs well.

5.3 Interface

Open a browser and type in: localhost / (foldername) TA / index.php. Then select how many cities you want.

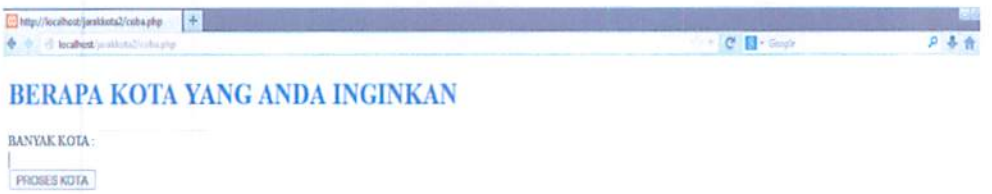


Figure 5.1 Main Menu

Type how much city want you processed.

to select multiple cities then the next process in both cities as well as the distance between the selected cities will be in accordance with the number of city-chosen early.

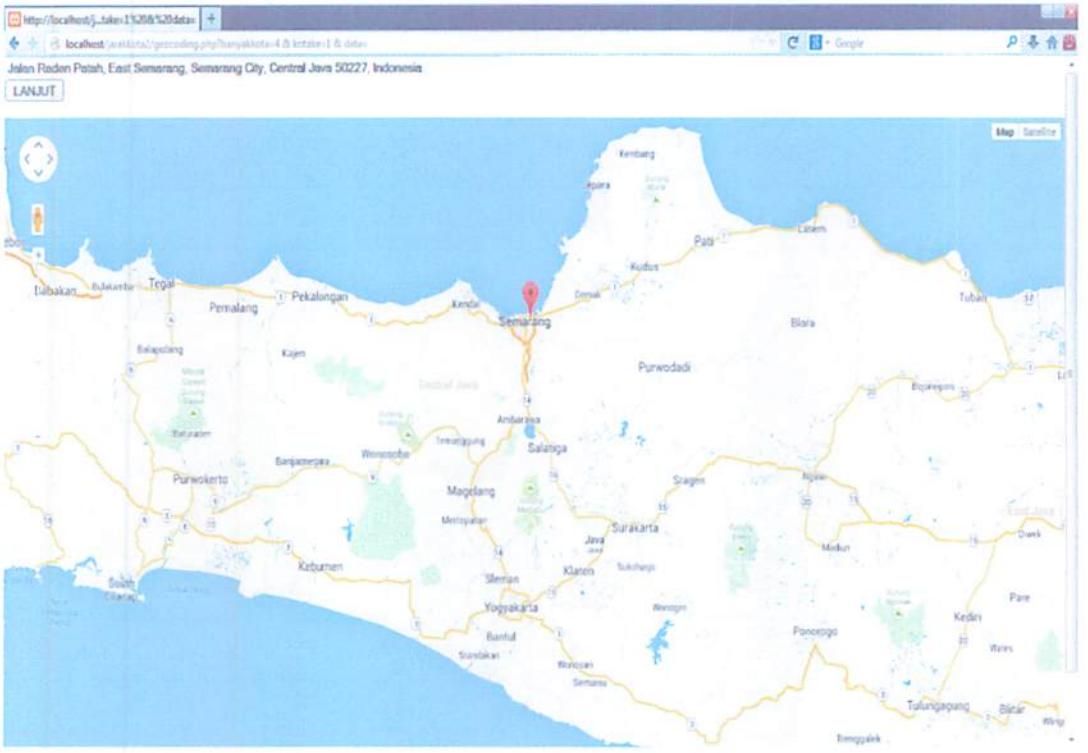


Figure 5.2 Map

Select the cities from the maps

By selecting a city in the dining google map API will automatically display the address on the map with detail. Thus, the city can be selected to be processed through the folder.

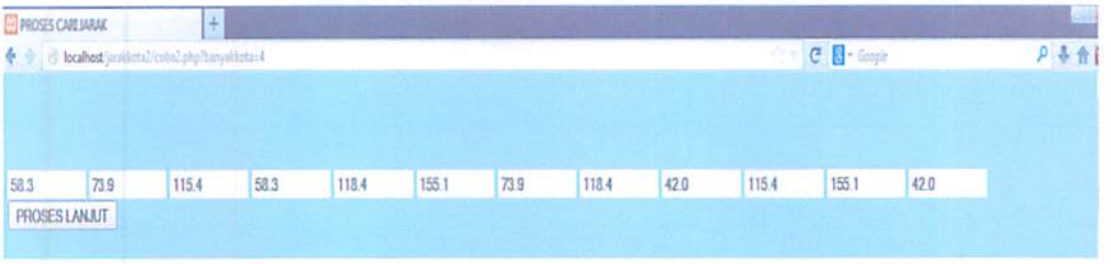


Figure 5.3 Get Distance

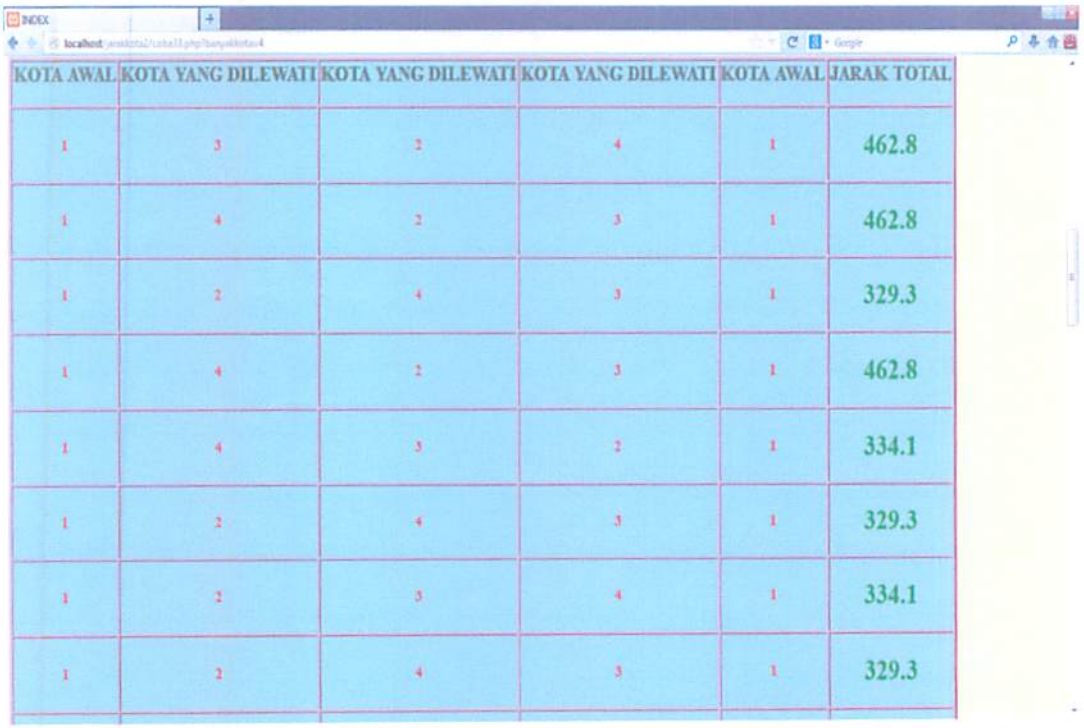
Get Distance From Google API. Program will automatically find the distance between the city so that the process can be carried out further calculations.

JARAK KOTA	Jalan Raden Patah, East Semarang, Semarang City, Central Java 50227, Indonesia	Burikan, Tegalrejo, Magelang, Indonesia	Jalan Lingkar Luar Pati, Pati, Central Java 59119, Indonesia	Jalan Jogorogo-Pancur, Lasem, Rembang 59271, Indonesia
Jalan Raden Patah, East Semarang, Semarang City, Central Java 50227, Indonesia	X	58.3	73.9	115.4
Burikan, Tegalrejo, Magelang, Indonesia	58.3	X	118.4	155.1
Jalan Lingkar Luar Pati, Pati, Central Java 59119, Indonesia	73.9	118.4	X	42.0
Jalan Jogorogo-Pancur, Lasem, Rembang 59271, Indonesia	115.4	155.1	42.0	X

Figure 5.4 Result Matrix

City and distance from Google API show in Matrix.

Of the process that is performed before selecting the city and get the distance in this process and the city distance obtained already accommodated in 2-dimensional array and shown to a table.



KOTA AWAL	KOTA YANG DILEWATI	KOTA YANG DILEWATI	KOTA YANG DILEWATI	KOTA AWAL	JARAK TOTAL
1	3	2	4	1	462.8
1	4	2	3	1	462.8
1	2	4	3	1	329.3
1	4	2	3	1	462.8
1	4	3	2	1	334.1
1	2	4	3	1	329.3
1	2	3	4	1	334.1
1	2	4	3	1	329.3

Figure 5.5 Randoming Process Enumeration

The process of random pattern city lines in order to get the minimum distance. These randomized, but the city limits to the city 2 to city last (because the city has become one beginning and end of the route). For further randomized but the city should not be the same as the previous route, that route does not pass through the city that same city.

5.3.2 Result Window

Look at the city end result which will be passed in order to get the minimum distance.

Figure 5.6 Result

This is the final result obtained from these iterations city and sought the minimum distance.