APPENDIX A

Sample Source Code

(Following is the code used in the implementation. The source code is also available in the library and with my supervisor)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class Default4 : System.Web.UI.Page
{
   SqlConnection con = new SqlConnection();
   protected void Page_Load(object sender, EventArgs e)
   {
      con.ConnectionString =
ConfigurationManager.ConnectionStrings["con"].ConnectionString;
      SqlDataSource2.ConnectionString = con.ConnectionString;
      //SqlDataSource1.ConnectionString = con.ConnectionString;
      if (con.State == ConnectionState.Closed)
      {
         con.Open();
      } }
   protected void Button1_Click(object sender, EventArgs e)
   {
      SqlCommand cmd = new SqlCommand("insert into Systemdb
values(@name,@ram,@pro)", con);
```

```
        cmd.Parameters.Add("@name", SqlDbType.VarChar, 500).Value =
TextBox1.Text;
        cmd.Parameters.Add("@ram", SqlDbType.VarChar, 500).Value =
DropDownList1.SelectedItem.Text;
        cmd.Parameters.Add("@pro", SqlDbType.VarChar, 500).Value =
DropDownList2.SelectedItem.Text;
        cmd.ExecuteNonQuery();
      Label1.Text=" Systems created .....";
      GridView1.DataBind();
    }}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
public partial class Default2 : System.Web.UI.Page
{
    SqlConnection con = new SqlConnection();
    protected void Page_Load(object sender, EventArgs e)
    {
        con.ConnectionString =
ConfigurationManager.ConnectionStrings["con"].ConnectionString;
        if (con.State == ConnectionState.Closed)
        {
            con.Open();
        }    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        Random getdetails = new Random();
        int number_of_tasks = Convert.ToInt32(TextBox1.Text);
```

```
        SqlCommand cmd1 = new SqlCommand("delete from task", con);
        cmd1.ExecuteNonQuery();
        cmd1.Dispose();
        for (int i = 0; i < number_of_tasks; i++)
        {
            SqlCommand cmd = new SqlCommand("insert into task
values(@name,@ram,@pro,@er,@time,@ideal)", con);
            cmd.Parameters.Add("@name", SqlDbType.VarChar, 500).Value = "T" + (i +
1);
            cmd.Parameters.Add("@ram", SqlDbType.VarChar, 500).Value =
getdetails.Next(1, 4);
            cmd.Parameters.Add("@pro", SqlDbType.VarChar, 500).Value =
getdetails.Next(1, 2);
            cmd.Parameters.Add("@er", SqlDbType.VarChar, 500).Value =
getdetails.Next(5, 34);
            cmd.Parameters.Add("@time", SqlDbType.VarChar, 500).Value =
getdetails.Next(10, 20);
            cmd.Parameters.Add("@ideal", SqlDbType.VarChar, 500).Value =
getdetails.Next(1, 4);
            cmd.ExecuteNonQuery();
            cmd.Dispose();
        }
        Label1.Visible = true;
        GridView1.DataBind();
    }}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
```

```
using System.Web.UI.DataVisualization.Charting;
using MicrosoftACO;
public partial class Default3 : System.Web.UI.Page
{
    Series srr = new Series("chart1");
    Series srr2 = new Series("chart2");
    SqlConnection con = new SqlConnection();
   MicrosoftACO.ACOptimization TasksOptimization = new ACOptimization();
    Random generate_temp = new Random();
    protected void Page_Load(object sender, EventArgs e)
    {
        con.ConnectionString =
ConfigurationManager.ConnectionStrings["con"].ConnectionString;
        if (con.State == ConnectionState.Closed)
        {
            con.Open();
        }
        if (Page.IsPostBack == false)
        {
            SqlCommand cmd = new SqlCommand("select tname from task", con);
            SqlDataReader dr = cmd.ExecuteReader();
            CheckBoxList1.DataSource = dr;
            CheckBoxList1.DataTextField = "tname";
            CheckBoxList1.DataBind();
            cmd.Dispose();
            dr.Close();
            SqlCommand cmd11 = new SqlCommand("select count( tname) as cont from
task ", con);
            SqlDataReader dr1 = cmd11.ExecuteReader();
            dr1.Read();
            Label2.Text = dr1["cont"].ToString().Trim();
            dr1.Close();
            cmd11.Dispose();
```

```
        SqlCommand cmd111 = new SqlCommand("select count( sysname) as cont
from Systemdb ", con);
        SqlDataReader dr111 = cmd111.ExecuteReader();
        dr111.Read();
        Label1.Text = dr111["cont"].ToString().Trim();
        dr111.Close();
        cmd111.Dispose();
      }
      for (int i = 0; i < CheckBoxList1.Items.Count; i++)
      {
        CheckBoxList1.Items[i].Selected = true;
      }
      CheckBoxList1.Enabled = false;
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
      Label3.Text = "MSQ Results";
      String[,] taskmatrix;
      String[] taskname = new String[100];
      String[,] systemmatrix;
      int task_for_execution = 0, system_in_network = 0;
      for (int selectqueue = 0; selectqueue < CheckBoxList1.Items.Count;
selectqueue++)
      {
        if (CheckBoxList1.Items[selectqueue].Selected == true)
        {
          taskname[task_for_execution] = CheckBoxList1.Items[selectqueue].Text;
          task_for_execution++;
        }
      }
Queue prepaired with task selection
      int total_time = 0;
      taskmatrix = new String[task_for_execution, 6];
      int[] temptask = new int[task_for_execution];
```

```
        for (int ll = 0; ll < task_for_execution; ll++)
        {
            SqlCommand cmd1 = new SqlCommand("select * from Task where
tname=@tskname ", con);
            cmd1.Parameters.AddWithValue("@tskname", taskname[ll]);
            SqlDataReader dr1 = cmd1.ExecuteReader();
            while (dr1.Read())
            {
                taskmatrix[ll, 0] = dr1["tname"].ToString();
                taskmatrix[ll, 1] = dr1["tram"].ToString();
                taskmatrix[ll, 2] = dr1["tpro"].ToString();
                taskmatrix[ll, 3] = dr1["tenergy"].ToString();
                taskmatrix[ll, 4] = dr1["ttime"].ToString();
                taskmatrix[ll, 5] = dr1["tideal"].ToString();
                temptask[ll] = generate_temp.Next(5, 10);
                total_time += Convert.ToInt32(dr1["ttime"].ToString());
            }
            dr1.Close();
            cmd1.Dispose();
        }
            try
        {
            for (int j = 0; j < task_for_execution; j++)
            {
                for (int jj = j + 1; jj < task_for_execution; jj++)
                {
                    if (Convert.ToInt32(taskmatrix[j, 3]) > Convert.ToInt32(taskmatrix[jj,
3]))
                    {
                        String preority = taskmatrix[j, 3];
                        taskmatrix[j, 3] = taskmatrix[jj, 3];
                        taskmatrix[jj, 3] = preority;
                        String temp2 = taskmatrix[j, 2];
                        taskmatrix[j, 2] = taskmatrix[jj, 2];
```

```
                taskmatrix[jj, 2] = temp2;
                temp2 = taskmatrix[j, 1];
                taskmatrix[j, 1] = taskmatrix[jj, 1];
                taskmatrix[jj, 1] = temp2;
                String taskname1 = taskmatrix[j, 0];
                taskmatrix[j, 0] = taskmatrix[jj, 0];
                taskmatrix[jj, 0] = taskname1;
                temp2 = taskmatrix[j, 4];
                taskmatrix[j, 4] = taskmatrix[jj, 4];
                taskmatrix[jj, 4] = temp2;
                temp2 = taskmatrix[j, 5];
                taskmatrix[j, 5] = taskmatrix[jj, 5];
               taskmatrix[jj, 5] = temp2;
            } } } }
catch
{
}
      String[] crashedsystem = new String[10];
system_in_network = Convert.ToInt32(Label1.Text);
systemmatrix = new String[system_in_network, 3];
int[] timer = new int[system_in_network];
int rowmgt = 0;
int[] temp = new int[system_in_network];
SqlCommand cmd = new SqlCommand("select * from Systemdb", con);
SqlDataReader dr = cmd.ExecuteReader();
while (dr.Read())
{
   systemmatrix[rowmgt, 0] = dr["sysname"].ToString();
   systemmatrix[rowmgt, 1] = dr["sysram"].ToString();
   systemmatrix[rowmgt, 2] = dr["syspro"].ToString();
   temp[rowmgt] = generate_temp.Next(500, 1000);
   rowmgt++;
}
dr.Close();
```

```
        cmd.Dispose();
        int[] systemload = new int[rowmgt];
        for (int j = 0; j < system_in_network; j++)
        {
           for (int jj = j + 1; jj < system_in_network; jj++)
           {
              if (Convert.ToInt32(systemmatrix[j, 1]) < Convert.ToInt32(systemmatrix[jj,
1]))
              {
                 String preority = systemmatrix[j, 0];
                 systemmatrix[j, 0] = systemmatrix[jj, 0];
                 systemmatrix[jj, 0] = preority;
                 String temp1 = systemmatrix[j, 1];
                 systemmatrix[j, 1] = systemmatrix[jj, 1];
                 systemmatrix[jj, 1] = temp1;
                 temp1 = systemmatrix[j, 2];
                 systemmatrix[j, 2] = systemmatrix[jj, 2];
                 systemmatrix[jj, 2] = temp1;
              }}}
        int[] exetimetask = new int[task_for_execution];
        int[] sysloadmgt = new int[system_in_network];
        int[] waitmgt = new int[system_in_network];
        int totalEnergyConsumption = 0, timeofexe = 0, unex = 0, dublbp =
task_for_execution;
        String[] unexe = new String[task_for_execution];
        int aa = 0, executedjobs = 0;
        int[] taskmatrix2 = new int[task_for_execution];
        int[] ctemp = new int[system_in_network];
          int t40 =    (task_for_execution *40)/100;
        int t20 = task_for_execution - (t40 * 2);
       String[,] taskmatrix1 = new String[t40, 6];// first 40%
       String[,] taskmatrix_40_2 = new String[t40, 6]; // mid 40%
       String[,] taskmatrix_20 = new String[t20, 6];  // Last 20%
       int tti_1 = 0;
```

```
 // divide Jobs into queues first
 for (int ti = 0; ti < t40;ti++ )
 {
    taskmatrix1[ti, 0] = taskmatrix[ti,0];
    taskmatrix1[ti, 1] = taskmatrix[ti, 1];
    taskmatrix1[ti, 2] = taskmatrix[ti, 2];
    taskmatrix1[ti, 3] = taskmatrix[ti, 3];
    taskmatrix1[ti, 4] = taskmatrix[ti, 4];
    taskmatrix1[ti, 5] = taskmatrix[ti, 5];
    tti_1++;
}
 int tti_2 = 0;
 for (int ti = t40; ti < t40*2; ti++)
 {
    taskmatrix_40_2[tti_2, 0] = taskmatrix[ti, 0];
    taskmatrix_40_2[tti_2, 1] = taskmatrix[ti, 1];
    taskmatrix_40_2[tti_2, 2] = taskmatrix[ti, 2];
    taskmatrix_40_2[tti_2, 3] = taskmatrix[ti, 3];
    taskmatrix_40_2[tti_2, 4] = taskmatrix[ti, 4];
    taskmatrix_40_2[tti_2, 5] = taskmatrix[ti, 5];
    tti_2++;
 }
int tti_3 = 0;
 for (int ti = t40 * 2; ti < task_for_execution; ti++)
 {
    taskmatrix_20[tti_3, 0] = taskmatrix[ti, 0];
    taskmatrix_20[tti_3, 1] = taskmatrix[ti, 1];
    taskmatrix_20[tti_3, 2] = taskmatrix[ti, 2];
    taskmatrix_20[tti_3, 3] = taskmatrix[ti, 3];
    taskmatrix_20[tti_3, 4] = taskmatrix[ti, 4];
    taskmatrix_20[tti_3, 5] = taskmatrix[ti, 5];
    tti_3++;
 }
 int wait = 0,ind2=0,ind3=0,ind=0;
```

```
string aaa = "";
int t = 0;
while(task_for_execution!=executedjobs)
 {
    aa = 0;
    int selected_job=1;
    if (ind < tti_1 && ind2 < tti_2 && ind3 < tti_3)
    {
       selected_job = generate_temp.Next(1, 3);
    }
    else if (ind2 < tti_2 && ind3 < tti_3 && ind == tti_1)
    {
       selected_job = generate_temp.Next(2, 3);
    }
    else
    {
       selected_job = 3;
    }
    if (ind < tti_1 && ind2 == tti_2 && ind3 == tti_3)
    {
       selected_job = 1;
   }
    if (ind ==tti_1 && ind2 < tti_2 && ind3 == tti_3)
    {
       selected_job = 2;
    }
    if (t == system_in_network)
       t = 0;
  for (int p = 0; p < system_in_network; p++)
    {
       if (selected_job == 1 && ind < tti_1)
       {
```

```
            if (Convert.ToInt32(taskmatrix1[ind, 1]) <=
Convert.ToInt32(systemmatrix[p, 1]) && Convert.ToInt32(taskmatrix1[ind, 2]) <=
Convert.ToInt32(systemmatrix[p, 2]))
                {
                    if (wait == 1)
                    {
                        timer[t] += Convert.ToInt32(taskmatrix1[ind, 4]);
                    }
                    timeofexe += Convert.ToInt32(taskmatrix1[ind, 4]);
                    totalEnergyConsumption += (timer[t] *
Convert.ToInt32(taskmatrix1[ind, 5])) + Convert.ToInt32(taskmatrix1[ind, 3]);
                    if (wait != 1)
                    {
                        timer[t] += Convert.ToInt32(taskmatrix1[ind, 4]);
                        wait = 1;
                    }
                    aaa += taskmatrix1[ind, 0] + " , ";
                    aa++;
                    t++;
                    executedjobs++;
                    ind++;
                    break;
                }                }
                    if (selected_job == 2 && ind2 < tti_2)
            {       if (Convert.ToInt32(taskmatrix_40_2[ind2, 1]) <=
Convert.ToInt32(systemmatrix[p, 1]) && Convert.ToInt32(taskmatrix_40_2[ind2, 2])
<= Convert.ToInt32(systemmatrix[p, 2]))
                {
                    if (wait == 1)
                    {
                        timer[t] += Convert.ToInt32(taskmatrix[ind2, 4]);
                    }
                    timeofexe += Convert.ToInt32(taskmatrix_40_2[ind2, 4]);
```

```
                totalEnergyConsumption += (timer[t] *
Convert.ToInt32(taskmatrix_40_2[ind2, 5])) +
Convert.ToInt32(taskmatrix_40_2[ind2, 3]);
                if (wait != 1)
                {
                   timer[t] += Convert.ToInt32(taskmatrix_40_2[ind2, 4]);
                   wait = 1;
                }
                aaa += taskmatrix_40_2[ind2, 0] + " , ";
                aa++;
                executedjobs++;
                t++;
                ind2++;
                break;
             }                }
                if (selected_job == 3 && ind3 < tti_3)
          {
             if (Convert.ToInt32(taskmatrix_20[ind3, 1]) <=
Convert.ToInt32(systemmatrix[p, 1]) && Convert.ToInt32(taskmatrix_20[ind3, 2])
<= Convert.ToInt32(systemmatrix[p, 2]))
             { if (wait == 1)
                {
                   timer[t] += Convert.ToInt32(taskmatrix_20[ind3, 4]);
                 }
                timeofexe += Convert.ToInt32(taskmatrix_20[ind3, 4]);
                totalEnergyConsumption += (timer[t] *
Convert.ToInt32(taskmatrix_20[ind3, 5])) + Convert.ToInt32(taskmatrix_20[ind3,
3]);
                if (wait != 1)
                {
                   timer[t] += Convert.ToInt32(taskmatrix_20[ind3, 4]);
                   wait = 1;
                }
                aaa += taskmatrix_20[ind3, 0]+" , ";
```

```
                aa++;
                t++;
                executedjobs++;
                ind3++;
                break;
            }}}
    if (aa == 0)
    {
        if (selected_job == 1 && ind < tti_1)
        {
            unexe[unex] = taskmatrix1[tti_1, 0];
            unex++;
        }
        if (selected_job == 2 && ind2 < tti_2)
        {
            unexe[unex] = taskmatrix_40_2[ind2, 0];
            unex++;
        }
        if (selected_job == 3 && ind3 < tti_3)
        {
            unexe[unex] = taskmatrix_20[ind3, 0];
            unex++;
        }}}
CheckBoxList2.Items.Clear();
for (int chkbox = 0; chkbox < unex; chkbox++)
{
    CheckBoxList2.Items.Add(unexe[chkbox].ToString());
}
try
{
    Label5.Text = unex + " ";
    Chart1.Visible = true;
    //show parameters and plot chart
    Label4.Text = totalEnergyConsumption + "  joules  ";
```

```
            String[] xvall = { "Energy consumption", "Time Consumption", "unexecuted
jobs" };
            int[] yvall = { totalEnergyConsumption, timer[0], unex };
            Chart1.Series[0].Points.DataBindXY(xvall, yvall);
            Chart1.Series.Add(srr);
            Label10.Text = timer[0] + "ms";
        }
        catch { }
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        Label3.Text = "ACO Results";
        String[,] taskmatrix;
        String[,] taskmatrix1;
        String[] taskname = new String[100];
        String[,] systemmatrix;
        int task_for_execution = 0, system_in_network = 0;
        for (int selectqueue = 0; selectqueue < CheckBoxList1.Items.Count;
selectqueue++)
        {
            if (CheckBoxList1.Items[selectqueue].Selected == true)
            {
                taskname[task_for_execution] = CheckBoxList1.Items[selectqueue].Text;
                task_for_execution++;
            }}
        int total_time = 0;
        taskmatrix = new String[task_for_execution, 6];
        int[] temptask = new int[task_for_execution];
        for (int ll = 0; ll < task_for_execution; ll++)
        {
            SqlCommand cmd1 = new SqlCommand("select * from Task where
tname=@tskname ", con);
            cmd1.Parameters.AddWithValue("@tskname", taskname[ll]);
            SqlDataReader dr1 = cmd1.ExecuteReader();
```

```
            while (dr1.Read())
            {
                taskmatrix[ll, 0] = dr1["tname"].ToString();
                taskmatrix[ll, 1] = dr1["tram"].ToString();
                taskmatrix[ll, 2] = dr1["tpro"].ToString();
                taskmatrix[ll, 3] = dr1["tenergy"].ToString();
                taskmatrix[ll, 4] = dr1["ttime"].ToString();
                taskmatrix[ll, 5] = dr1["tideal"].ToString();
                temptask[ll] = generate_temp.Next(5, 10);
                total_time += Convert.ToInt32(dr1["ttime"].ToString());
            }
            dr1.Close();
            cmd1.Dispose();
        }
        try
        {
            for (int j = 0; j < task_for_execution; j++)
            {
                for (int jj = j + 1; jj < task_for_execution; jj++)
                {
                    if (Convert.ToInt32(taskmatrix[j, 3]) > Convert.ToInt32(taskmatrix[jj,
3]))
                    {
                        String preority = taskmatrix[j, 3];
                        taskmatrix[j, 3] = taskmatrix[jj, 3];
                        taskmatrix[jj, 3] = preority;
                        String temp2 = taskmatrix[j, 2];
                        taskmatrix[j, 2] = taskmatrix[jj, 2];
                        taskmatrix[jj, 2] = temp2;
                        temp2 = taskmatrix[j, 1];
                        taskmatrix[j, 1] = taskmatrix[jj, 1];
                        taskmatrix[jj, 1] = temp2;
                        String taskname1 = taskmatrix[j, 0];
                        taskmatrix[j, 0] = taskmatrix[jj, 0];
```

```
                taskmatrix[jj, 0] = taskname1;
                temp2 = taskmatrix[j, 4];
                taskmatrix[j, 4] = taskmatrix[jj, 4];
                taskmatrix[jj, 4] = temp2;
                temp2 = taskmatrix[j, 5];
                taskmatrix[j, 5] = taskmatrix[jj, 5];
                taskmatrix[jj, 5] = temp2;
            } }} }
catch
{
}
String[] crashedsystem = new String[10];
system_in_network = Convert.ToInt32(Label1.Text);
systemmatrix = new String[system_in_network, 3];
int[] timer = new int[system_in_network];
int rowmgt = 0;
int[] temp = new int[system_in_network];
SqlCommand cmd = new SqlCommand("select * from Systemdb", con);
SqlDataReader dr = cmd.ExecuteReader();
while (dr.Read())
{
   systemmatrix[rowmgt, 0] = dr["sysname"].ToString();
   systemmatrix[rowmgt, 1] = dr["sysram"].ToString();
   systemmatrix[rowmgt, 2] = dr["syspro"].ToString();
   temp[rowmgt] = generate_temp.Next(500, 1000);
   rowmgt++;
}
dr.Close();
cmd.Dispose();
int[] systemload = new int[rowmgt];
for (int j = 0; j < system_in_network; j++)
{
   for (int jj = j + 1; jj < system_in_network; jj++)
   {
```

```
            if (Convert.ToInt32(systemmatrix[j, 1]) < Convert.ToInt32(systemmatrix[jj,
1]))
            {
               String preority = systemmatrix[j, 0];
               systemmatrix[j, 0] = systemmatrix[jj, 0];
               systemmatrix[jj, 0] = preority;
               String temp1 = systemmatrix[j, 1];
               systemmatrix[j, 1] = systemmatrix[jj, 1];
               systemmatrix[jj, 1] = temp1;
               temp1 = systemmatrix[j, 2];
               systemmatrix[j, 2] = systemmatrix[jj, 2];
               systemmatrix[jj, 2] = temp1;
            } }  }
      TasksOptimization.init(taskmatrix,systemmatrix,task_for_execution);
      taskmatrix1 = TasksOptimization.ACO();
      int[] exetimetask = new int[task_for_execution];
      int[] sysloadmgt = new int[system_in_network];
      int[] waitmgt = new int[system_in_network];
      int totalEnergyConsumption = 0, timeofexe = 0, unex = 0, dublbp =
task_for_execution;
      String[] unexe = new String[task_for_execution];
      int aa = 0, executedjobs = 0;
      int[] taskmatrix2 = new int[task_for_execution];
      int[] ctemp = new int[system_in_network];
      int waitingTime = 0, t=0;
      for (int jobqueues = 0; jobqueues < task_for_execution; jobqueues++)
      {
         aa = 0;
         if (system_in_network == t)
            t = 0;
         for (int p = 0; p < system_in_network; p++)
         {
```

```
            if (Convert.ToInt32(taskmatrix1[jobqueues, 1]) <=
Convert.ToInt32(systemmatrix[p, 1]) && Convert.ToInt32(taskmatrix1[jobqueues,
2]) <= Convert.ToInt32(systemmatrix[p, 2]))
                {
                    timer[t] += Convert.ToInt32(taskmatrix1[jobqueues, 4]);
                    timeofexe += Convert.ToInt32(taskmatrix1[jobqueues, 4]);
                    totalEnergyConsumption += (timer[t] *
Convert.ToInt32(taskmatrix1[jobqueues, 5]) ) +
Convert.ToInt32(taskmatrix1[jobqueues, 3]);
                    aa++;
                    executedjobs++;
                    t++;
                    break;
                } }
          if (aa == 0)
          {
             unexe[unex] = taskmatrix[jobqueues, 0];
                unex++;
           } }
        CheckBoxList2.Items.Clear();
       for (int chkbox = 0; chkbox < unex; chkbox++)
       {
          CheckBoxList2.Items.Add(unexe[chkbox].ToString());
       }

       try
       {
          Label5.Text = unex + " ";
          Chart1.Visible = true;
          //show parameters and plot chart
          Label4.Text = totalEnergyConsumption + "  joules  ";
          String[] xvall = { "Energy consumption", "Time Consumption", "unexecuted
jobs" };
          int[] yvall = { totalEnergyConsumption, timer[0], unex };
```

```
            Chart1.Series[0].Points.DataBindXY(xvall, yvall);
            Chart1.Series.Add(srr);
            Label10.Text = timeofexe + "ms";
        }
        catch { }
 }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Web.UI.DataVisualization.Charting;
using MicrosoftACO;
public partial class Default5 : System.Web.UI.Page
{
    Series srr = new Series("chart1");
    Series srr2 = new Series("chart2");
    SqlConnection con = new SqlConnection();
    MicrosoftACO.ACOptimization TasksOptimization = new ACOptimization();
    Random generate_temp = new Random();
    protected void Page_Load(object sender, EventArgs e)
    {
        con.ConnectionString =
ConfigurationManager.ConnectionStrings["con"].ConnectionString;
        if (con.State == ConnectionState.Closed)
        {
            con.Open();
        }
        if (Page.IsPostBack == false)
```

```
        {
            SqlCommand cmd = new SqlCommand("select tname from task", con);
            SqlDataReader dr = cmd.ExecuteReader();
            CheckBoxList1.DataSource = dr;
            CheckBoxList1.DataTextField = "tname";
            CheckBoxList1.DataBind();
            cmd.Dispose();
            dr.Close();
            SqlCommand cmd11 = new SqlCommand("select count( tname) as cont from
task ", con);
            SqlDataReader dr1 = cmd11.ExecuteReader();
            dr1.Read();
            Label2.Text = dr1["cont"].ToString().Trim();
            dr1.Close();
            cmd11.Dispose();
            SqlCommand cmd111 = new SqlCommand("select count( sysname) as cont
from Systemdb ", con);
            SqlDataReader dr111 = cmd111.ExecuteReader();
            dr111.Read();
            Label1.Text = dr111["cont"].ToString().Trim();
            dr111.Close();
            cmd111.Dispose();
        }
        for (int i = 0; i < CheckBoxList1.Items.Count; i++)
        {
            CheckBoxList1.Items[i].Selected = true;
        }
        CheckBoxList1.Enabled = false;
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        Label3.Text = "Proposed Results";
        String[,] taskmatrix;
        String[] taskname = new String[100];
```

```
        String[,] systemmatrix;
        int task_for_execution = 0, system_in_network = 0;
        for (int selectqueue = 0; selectqueue < CheckBoxList1.Items.Count;
selectqueue++)
        {
            if (CheckBoxList1.Items[selectqueue].Selected == true)
            {
                taskname[task_for_execution] = CheckBoxList1.Items[selectqueue].Text;
                task_for_execution++;
            }
        }
        int total_time = 0;
        taskmatrix = new String[task_for_execution, 7];
        int[] temptask = new int[task_for_execution];
        for (int ll = 0; ll < task_for_execution; ll++)
        {
            SqlCommand cmd1 = new SqlCommand("select * from Task where
tname=@tskname ", con);
            cmd1.Parameters.AddWithValue("@tskname", taskname[ll]);
            SqlDataReader dr1 = cmd1.ExecuteReader();
            while (dr1.Read())
            {
                taskmatrix[ll, 0] = dr1["tname"].ToString();
                taskmatrix[ll, 1] = dr1["tram"].ToString();
                taskmatrix[ll, 2] = dr1["tpro"].ToString();
                taskmatrix[ll, 3] = dr1["tenergy"].ToString();
                taskmatrix[ll, 4] = dr1["ttime"].ToString();
                taskmatrix[ll, 5] = dr1["tideal"].ToString();
                taskmatrix[ll, 6] = "0";
                temptask[ll] = generate_temp.Next(5, 10);
                total_time += Convert.ToInt32(dr1["ttime"].ToString());
            }
            dr1.Close();
            cmd1.Dispose();
```

```
        }
        try
        {
            for (int j = 0; j < task_for_execution; j++)
            {
                for (int jj = j + 1; jj < task_for_execution; jj++)
                {
                    if (Convert.ToInt32(taskmatrix[j, 3]) > Convert.ToInt32(taskmatrix[jj,
3]))
                    {
                        String preority = taskmatrix[j, 3];
                        taskmatrix[j, 3] = taskmatrix[jj, 3];
                        taskmatrix[jj, 3] = preority;
                        String temp2 = taskmatrix[j, 2];
                        taskmatrix[j, 2] = taskmatrix[jj, 2];
                        taskmatrix[jj, 2] = temp2;
                        temp2 = taskmatrix[j, 1];
                        taskmatrix[j, 1] = taskmatrix[jj, 1];
                        taskmatrix[jj, 1] = temp2;
                        String taskname1 = taskmatrix[j, 0];
                        taskmatrix[j, 0] = taskmatrix[jj, 0];
                        taskmatrix[jj, 0] = taskname1;
                        temp2 = taskmatrix[j, 4];
                        taskmatrix[j, 4] = taskmatrix[jj, 4];
                        taskmatrix[jj, 4] = temp2;
                        temp2 = taskmatrix[j, 5];
                        taskmatrix[j, 5] = taskmatrix[jj, 5];
                        taskmatrix[jj, 5] = temp2;
                    }
                }
            }
        }
        catch
        {
```

```
        }
         String[] crashedsystem = new String[10];
        system_in_network = Convert.ToInt32(Label1.Text);
        systemmatrix = new String[system_in_network, 3];
        int[] timer = new int[system_in_network];
        int rowmgt = 0;
        int[] temp = new int[system_in_network];
        SqlCommand cmd = new SqlCommand("select * from Systemdb", con);
        SqlDataReader dr = cmd.ExecuteReader();
        while (dr.Read())
        {
           systemmatrix[rowmgt, 0] = dr["sysname"].ToString();
           systemmatrix[rowmgt, 1] = dr["sysram"].ToString();
           systemmatrix[rowmgt, 2] = dr["syspro"].ToString();
           temp[rowmgt] = generate_temp.Next(500, 1000);
           rowmgt++;
        }
        dr.Close();
        cmd.Dispose();

        int[] systemload = new int[rowmgt];
        for (int j = 0; j < system_in_network; j++)
        {
           for (int jj = j + 1; jj < system_in_network; jj++)
           {
              if (Convert.ToInt32(systemmatrix[j, 1]) < Convert.ToInt32(systemmatrix[jj,
1]))
              {
                 String preority = systemmatrix[j, 0];
                 systemmatrix[j, 0] = systemmatrix[jj, 0];
                 systemmatrix[jj, 0] = preority;
                 String temp1 = systemmatrix[j, 1];
                 systemmatrix[j, 1] = systemmatrix[jj, 1];
                 systemmatrix[jj, 1] = temp1;
```

```
                temp1 = systemmatrix[j, 2];
                systemmatrix[j, 2] = systemmatrix[jj, 2];
                systemmatrix[jj, 2] = temp1;
            }
        }
    }

    int[] exetimetask = new int[task_for_execution];
    int[] sysloadmgt = new int[system_in_network];
    int[] waitmgt = new int[system_in_network];
    int totalEnergyConsumption = 0, timeofexe = 0, unex = 0, dublbp =
task_for_execution;
    String[] unexe = new String[task_for_execution];
    int aa = 0, executedjobs = 0;
    int[] taskmatrix2 = new int[task_for_execution];
    int[] ctemp = new int[system_in_network];
    int t1 = (task_for_execution) /2;
    int t2 = task_for_execution - t1;
    String[,] taskmatrix1 = new String[t1, 7];// first
    String[,] taskmatrix_2 = new String[t2, 7]; // second
        int tti_1 = 0;
    for (int ti = 0; ti < t1; ti++)
    {
      taskmatrix1[ti, 0] = taskmatrix[ti, 0];
      taskmatrix1[ti, 1] = taskmatrix[ti, 1];
      taskmatrix1[ti, 2] = taskmatrix[ti, 2];
      taskmatrix1[ti, 3] = taskmatrix[ti, 3];
      taskmatrix1[ti, 4] = taskmatrix[ti, 4];
      taskmatrix1[ti, 5] = taskmatrix[ti, 5];
      taskmatrix1[ti, 6] ="0";
      tti_1++;
    }
    int tti_2 = 0;
    for (int ti = t1; ti < task_for_execution; ti++)
```

```
{
   taskmatrix_2[tti_2, 0] = taskmatrix[ti, 0];
   taskmatrix_2[tti_2, 1] = taskmatrix[ti, 1];
   taskmatrix_2[tti_2, 2] = taskmatrix[ti, 2];
   taskmatrix_2[tti_2, 3] = taskmatrix[ti, 3];
   taskmatrix_2[tti_2, 4] = taskmatrix[ti, 4];
   taskmatrix_2[tti_2, 5] = taskmatrix[ti, 5];
   taskmatrix_2[tti_2, 6] = "0";
   tti_2++;
}
int wait = 0, ind = 0, ind2 = 0;
string aaa = "";
int t = 0;
int selct = 0,selet=0;
while (task_for_execution != executedjobs)
{
   aa = 0;
    selct = 0;
    selet = 0;
   if (t == system_in_network)
      t = 0;
   // find free system
   for (int p = 0; p < system_in_network; p++)
   {
      if ( ind < tti_1 || ind2 < tti_2 )
      {// parameters calculation
         if (ind < tti_1 && selct==0)
         {// parameters calculation
            if (Convert.ToInt32(taskmatrix1[ind, 1]) <=
Convert.ToInt32(systemmatrix[p, 1]) && Convert.ToInt32(taskmatrix1[ind, 2]) <=
Convert.ToInt32(systemmatrix[p, 2]))
            {
               if (wait == 1)
               {
```

```
                    timer[t] += Convert.ToInt32(taskmatrix1[ind, 4]);
                }
                timeofexe += Convert.ToInt32(taskmatrix1[ind, 4]);
                totalEnergyConsumption += (timer[t] *
Convert.ToInt32(taskmatrix1[ind, 5])) + Convert.ToInt32(taskmatrix1[ind, 3]);
                if (wait != 1)
                {
                    timer[t] += Convert.ToInt32(taskmatrix1[ind, 4]);
                }
                taskmatrix1[ind, 6] = "1";
                aaa += taskmatrix1[ind, 0] + " , ";
                aa++;
                executedjobs++;
                ind++;
                 selct = 1;
            }
        }
        if (ind2 < tti_2 && selet==0)
            if (Convert.ToInt32(taskmatrix_2[ind2, 1]) <=
Convert.ToInt32(systemmatrix[p, 1]) && Convert.ToInt32(taskmatrix_2[ind2, 2]) <=
Convert.ToInt32(systemmatrix[p, 2]))
            {
                if (wait == 1)
                {
                    timer[t] += Convert.ToInt32(taskmatrix_2[ind2, 4]);
                }
                timeofexe += Convert.ToInt32(taskmatrix_2[ind2, 4]);
                totalEnergyConsumption += Convert.ToInt32(taskmatrix_2[ind2,
3]);
                executedjobs++;
                ind2++;
                selet = 1;
            }
        }
```

```
                if (selct == 1 && selet == 1)
                {
                    t++;
                    break;
                }
               }
          Label4.Text = totalEnergyConsumption + "  joules  ";
          String[] xvall = { "Energy consumption", "Time Consumption", "unexecuted
jobs" };
          int[] yvall = { totalEnergyConsumption, timer[1], unex };
          Chart1.Series[0].Points.DataBindXY(xvall, yvall);
          Chart1.Series.Add(srr);
          Label10.Text = timer[1] + "ms";
       }
       catch { }
   }
}
<?xml version="1.0"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
          <appSettings>
                    <add key="ChartImageHandler"
value="storage=file;timeout=20;dir=c:\TempImageFiles\;"/>
          </appSettings>
          <connectionStrings>
                    <add name="con" connectionString="Server=JASPREET-
PC\SQLEXPRESS;database=scheduling;uid=sa;pwd=123;"/>
                    <add name="schedulingConnectionString" connectionString="Data
Source=JASPREET-PC\SQLEXPRESS;Initial Catalog=scheduling;Persist Security
Info=True;User ID=sa;Password=123" providerName="System.Data.SqlClient"/>
          </connectionStrings>
```

```
<system.web>
        <httpHandlers>
            <add path="ChartImg.axd" verb="GET,HEAD,POST"
type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler,
System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" validate="false"/>
        </httpHandlers>
        <pages>
            <controls>
                <add tagPrefix="asp"
namespace="System.Web.UI.DataVisualization.Charting"
assembly="System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"/>
            </controls>
        </pages>
        <compilation debug="true" targetFramework="4.0">
            <assemblies>
                <add assembly="System.Web.DataVisualization,
Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/></assemblies></compilation>
        <authentication mode="Forms">
            <forms loginUrl="~/Account/Login.aspx" timeout="2880"/>
        </authentication>
        <membership>
            <providers>
                <clear/>
                <add name="AspNetSqlMembershipProvider"
type="System.Web.Security.SqlMembershipProvider"
connectionStringName="ApplicationServices" enablePasswordRetrieval="false"
enablePasswordReset="true" requiresQuestionAndAnswer="false"
requiresUniqueEmail="false" maxInvalidPasswordAttempts="5"
minRequiredPasswordLength="6" minRequiredNonalphanumericCharacters="0"
passwordAttemptWindow="10" applicationName="/"/>
            </providers>
```

```
            </membership>
            <profile>
                <providers>
                    <clear/>
                    <add name="AspNetSqlProfileProvider"
type="System.Web.Profile.SqlProfileProvider"
connectionStringName="ApplicationServices" applicationName="/"/>
                </providers>
            </profile>
            <roleManager enabled="false">
                <providers>
                    <clear/>
                    <add name="AspNetSqlRoleProvider"
type="System.Web.Security.SqlRoleProvider"
connectionStringName="ApplicationServices" applicationName="/"/>
                    <add name="AspNetWindowsTokenRoleProvider"
type="System.Web.Security.WindowsTokenRoleProvider" applicationName="/"/>
                </providers>
            </roleManager>
        </system.web>
        <system.webServer>
            <modules runAllManagedModulesForAllRequests="true"/>
            <validation validateIntegratedModeConfiguration="false"/>
            <handlers>
                <remove name="ChartImageHandler"/>
                <add name="ChartImageHandler"
preCondition="integratedMode" verb="GET,HEAD,POST" path="ChartImg.axd"
type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler,
System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"/>
            </handlers>
        </system.webServer>
</configuration>
```