using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Service\_Database\_Connection

{

public class OptimalRouteAlgorithm

{

//public EntitiesContext ec;

//public OptimalRouteAlgorithm(EntitiesContext context)

//{

// ec = context;

//}

public int GetUnloadTime(decimal number, out int firstpart)

{

firstpart = (int)Math.Truncate(number);

int secondpart = (int)Math.Round(100 \* Math.Abs(number - firstpart));

int total;

if (firstpart == 0)

{

total = secondpart;

return total;

}

else

{

return total = firstpart \* 60 + secondpart;

}

}

public List<Distance\_Table> FindOptimalRoute()

{

List<Distance\_Table> originalList = new List<Distance\_Table>();

EntitiesContext ec = new EntitiesContext();

originalList = (from cust in ec.Distances

orderby cust.Id

select cust).ToList();

List<Distance\_Table> finalOptimalRoute = new List<Distance\_Table>();

List<Distance\_Table> delivery\_Address = new List<Distance\_Table>();

List<Distance\_Table> temp\_Address = new List<Distance\_Table>();

List<Distance\_Table> total\_Address = new List<Distance\_Table>();

total\_Address = originalList.GroupBy(x => x.Origin).Select(p => p.First()).Distinct().ToList();

temp\_Address = originalList.Where(p => p.Origin != p.Destination).ToList();

List<ConfigOptimalRoute> optimal\_ConfigList = new List<ConfigOptimalRoute>();

optimal\_ConfigList = (from config in ec.OptimalRoute\_Config

select config).ToList();

int maximum\_AllowedTime = optimal\_ConfigList[0].Maximum\_Hour \* 60;

decimal unload\_Totaltime = optimal\_ConfigList[0].Unload\_Time;

int unloadTime\_Firstpart;

int final\_Unloadtime =GetUnloadTime(unload\_Totaltime, out unloadTime\_Firstpart);

int temp\_total\_time = 0;

delivery\_Address = temp\_Address.Where(p => p.Origin.Contains("Zeugstraat 92, 2801 JD Gouda, Netherlands")).OrderBy(c => c.Duration).Take(1).ToList();

int final\_Totaltime = final\_Unloadtime + delivery\_Address[0].Duration ;

temp\_total\_time = final\_Totaltime;

temp\_Address = temp\_Address.Where(p => p.Destination != delivery\_Address[0].Origin).OrderBy(c => c.Duration).ToList();

delivery\_Address = originalList.Where(p => p.Origin.Contains(delivery\_Address[0].Destination) && p.Destination.Contains("Zeugstraat 92, 2801 JD Gouda, Netherlands")).Take(1).ToList();

temp\_total\_time = final\_Totaltime + delivery\_Address[0].Duration;

if (temp\_total\_time <= maximum\_AllowedTime)

{

int i = 1;

while (i <= total\_Address.Count - 2)

{

delivery\_Address = temp\_Address.Where(p => p.Origin.Contains(delivery\_Address[0].Destination)).OrderBy(c => c.Duration).Take(1).ToList();

final\_Totaltime = final\_Totaltime + delivery\_Address[0].Duration + final\_Unloadtime;

if (final\_Totaltime <= maximum\_AllowedTime)

{

finalOptimalRoute.AddRange(delivery\_Address);

temp\_Address = temp\_Address.Where(p => p.Destination != delivery\_Address[0].Origin).OrderBy(c => c.Duration).ToList();

}

else

{

delivery\_Address = originalList.Where(p => p.Origin.Contains(delivery\_Address[0].Destination) && p.Destination.Contains("Zeugstraat 92, 2801 JD Gouda, Netherlands")).Take(1).ToList();

final\_Totaltime = final\_Totaltime + delivery\_Address[0].Duration;

finalOptimalRoute.AddRange(delivery\_Address);

}

i++;

}

}

delivery\_Address = originalList.Where(p => p.Origin.Contains(delivery\_Address[0].Destination) && p.Destination.Contains("Zeugstraat 92, 2801 JD Gouda, Netherlands")).Take(1).ToList();

final\_Totaltime = final\_Totaltime + delivery\_Address[0].Duration;

finalOptimalRoute.AddRange(delivery\_Address);

return finalOptimalRoute;

}

}

}