```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
//extra
using System.Collections;
using System.Globalization;
using LOLTeamCounterPick.Classes;
using System.Resources;
using System.IO;
using System.Threading;
using LOLTeamCounterPick.Components;

namespace LOLTeamCounterPick
{
    public partial class LOLTeamCounterPick : Form
    {
        List<string> Result;
        ChampionList championlist;
        ChampionList resultlist;
        List<ChampionData> championsData;
        SelectionPanel selectionPanel;
        Thread import;
        Thread update;
        public LOLTeamCounterPick()
        {
            InitializeComponent();
            Result = new List<string>();
            selectionPanel = new SelectionPanel();
            SelectionGB.Controls.Add(selectionPanel.panel);
            championlist = new ChampionList(Properties.ChampionIcons.ResourceManager);
            selectionPanel.LoadSelectionPanel(championlist, true);

            update = new Thread(this.UpdateData);

            import = new Thread(this.ImportData);

            if (!LocalDataComplete())
            {
                MessageBox.Show("Error: Local Data Imcomplete!\r\nStarting Updata Local Data!");
                update.Start();
                while (update.IsAlive) { }
            }
            else
            {
                import.Start();
                while (import.IsAlive) { }
            }
            this.Update();
        }


        private bool LocalDataComplete()
        {
            bool isDataComplete = true;
            string currentDirectory = Directory.GetCurrentDirectory();
            string dataDirectory = currentDirectory + "\\Data";
            if (!Directory.Exists(dataDirectory)) { isDataComplete = false; }
            else
            {
                string fileName;
                foreach (string championName in GVs.championNames)
                {
                    fileName = dataDirectory + "\\" + championName + ".xls";
                    if (!File.Exists(fileName))
                    {
                        //missing at least one local data, stop detecting and go update
                        isDataComplete = false; break;
                    }
                    else { /*keep going thru the loop for all local data*/ }
                }
            }
            return isDataComplete;
        }
        private void UpdateData()
        {
            try
            {
                ProgressFRM update = new ProgressFRM();
                update.DownloadFromNet();
                championsData = (List<ChampionData>)GVs.ChampionsData;
            }
            catch (Exception error)
            {
                MessageBox.Show("Error Updating Local Data: " + error.Message + "\r\nStarting Update Again! ↙
    ");
                UpdateData();
            }
        }
        private void ImportData()
        {
            try
            {
                ProgressFRM import = new ProgressFRM();
                import.Import();
                championsData = (List<ChampionData>)GVs.ChampionsData;
            }
            catch (Exception error)
            {
                MessageBox.Show("Error Import Local Data: " + error.Message + "\r\nStarting Import Again!");
                ImportData();
            }

        }

        private void submitBTN_Click(object sender, EventArgs e)
        {
            GVs.ResetLists();
            oppoSelectPB1.Report();
            oppoSelectPB2.Report();
            oppoSelectPB3.Report();
            oppoSelectPB4.Report();
            oppoSelectPB5.Report();
            allySelectPB1.Report();
            allySelectPB2.Report();
            allySelectPB3.Report();
            allySelectPB4.Report();
            allySelectPB5.Report();
            oppoBanPB1.Report();
            oppoBanPB2.Report();
            oppoBanPB3.Report();
            allyBanPB1.Report();
            allyBanPB2.Report();
            allyBanPB3.Report();
            resultlist = GetResult();
            selectionPanel.LoadSelectionPanel(resultlist, false);
            selectionPanel.disable();
            submitBTN.Enabled = false;
            resetBTN.Enabled = true;
            copyBTN.Enabled = true;
        }

        private void resetBTN_Click(object sender, EventArgs e)
        {
            oppoSelectPB1.Reset();
            oppoSelectPB2.Reset();
            oppoSelectPB3.Reset();
            oppoSelectPB4.Reset();
            oppoSelectPB5.Reset();
            allySelectPB1.Reset();
            allySelectPB2.Reset();
            allySelectPB3.Reset();
            allySelectPB4.Reset();
            allySelectPB5.Reset();
            oppoBanPB1.Reset();
            oppoBanPB2.Reset();
            oppoBanPB3.Reset();
            allyBanPB1.Reset();
            allyBanPB2.Reset();
            allyBanPB3.Reset();

            Result.Clear();
            GVs.selectedChampion = null;
            GVs.selectedSlot = null;
            Application.DoEvents();

            selectionPanel.LoadSelectionPanel(championlist, true);
            //selectionPanel.enable();
            submitBTN.Enabled = true;
            resetBTN.Enabled = false;
            copyBTN.Enabled = false;


        }

        private void copyBTN_Click(object sender, EventArgs e)
        {
            string tmp = "Here are good choices to counter the opponents:\r\n";
            int count = 0;
            foreach (string str in Result)
            {
                if (++count % 5 == 0) { tmp += str + "\r\n"; }
                else { tmp += str + " "; }

            }
            Clipboard.SetText(tmp);
            MessageBox.Show("Copy successfully.");
        }
        private ChampionList GetResult()
        {
            Dictionary<string, double> result = new Dictionary<string, double>();
            foreach (string championName in GVs.oppoList)
            {
                foreach (ChampionData championData in championsData)
                {
                    foreach (ChampionDataRecord championDataRecord in championData.Records)
                    {
                        if (championName == championDataRecord.name)
                        {
                            if (result.Select(i => i.Key).Contains(championData.name))
                            {
                                result[championData.name] += championDataRecord.value;
                            }
                            else
                            {
                                result.Add(championData.name, championDataRecord.value);
                            }
                        }
                    }
                }
            }
            result = result.OrderByDescending(i => i.Value).ToDictionary(i=>i.Key,i=>i.Value);
            foreach (string name in GVs.banList)
            {
                result.Remove(name);
            }
            foreach (string name in GVs.oppoList)
            {
                result.Remove(name);
            }
            while (result.Count > 30) { result.Remove(result.Keys.Last()); }
            foreach (var tmp in result)
            {
                Result.Add(tmp.Key);
            }
            ChampionList resultlist = new ChampionList();
            foreach (string name in result.Keys)
            {
                resultlist.list.Add(championlist.list.Find(i => i.name == name));
            }
            return resultlist;

        }




    }
}
```