```
using MahApps.Metro.Controls;
using Microsoft.Data.Sqlite;
using Microsoft.Win32.TaskScheduler;
using PlotterDataGH.Properties;
using SendFileTo;
using System;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;

namespace WpfApp2
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : MetroWindow
    {
        int addedRows = 0;
        DataTable dataTable = new DataTable();

        public MainWindow()
        {
            InitializeComponent();
            LoadData();
        }

        //While a scan is in progress disable the scanning button
        public void DisableWhileScanning()
        {
            foreach (UserControl1 row in fillerGrid.Children)
            {
                row.btnScan.IsEnabled = false;
            }
        }

        //Load the data from the local database file
        public void LoadData()
        {
            dataTable.Clear();
            SqliteConnection cnn;
            SqliteCommand cmd = null;
            cnn = new SqliteConnection("Data Source=plotterData.db;");
            cnn.Open();

            string query = "SELECT m1.*, models.plotter_type FROM printer_data ↩
              m1 LEFT JOIN printer_data m2 ON (m1.serial_number =             ↩
             m2.serial_number AND m1.id < m2.id) INNER JOIN models on         ↩
             models.id = m1.model_id WHERE m2.id IS NULL";
            cmd = new SqliteCommand(query, cnn);

            SqliteDataReader reader = cmd.ExecuteReader();
            dataTable.Load(reader);

            foreach (DataRow row in dataTable.Rows)
            {
                RowDefinition rd = new RowDefinition();
                rd.Height = GridLength.Auto;
                fillerGrid.RowDefinitions.Add(rd);
                UserControl1 userControl1 = new UserControl1();
                userControl1.lblMeterstand.Content = string.Format(row["meters_printed"].ToString());
                userControl1.lblNaam.Content = string.Format(row["naam"].ToString());
                userControl1.plotterId = Convert.ToInt32(row["id"]);
                userControl1.lblMerk.Content = string.Format(row["plotter_type"].ToString());
                userControl1.lblMerk.Uid = string.Format(row["model_id"].ToString());
                userControl1.plotterIp = string.Format(row["ip"].ToString());
                userControl1.lblSerialNumber.Content = "S/N: " + string.Format(row["serial_number"].ToString());
                userControl1.serialnm = string.Format(row["serial_number"].ToString());

                var dt = DateTime.Parse((string)row["datetime"]);

                userControl1.lblTime.Content = dt.ToString("dd/MM/yy H:mm");
                userControl1.loadData();
                userControl1.ParentForm = this;
                fillerGrid.Children.Add(userControl1);
                Grid.SetRow(userControl1, fillerGrid.RowDefinitions.Count - 1);
                addedRows++;
            }

            //Create a CSV file for mailing
            try
            {
                StringBuilder sb = new StringBuilder();

                IEnumerable<string> columnNames = dataTable.Columns.Cast<DataColumn>().
                                                  Select(column => column.ColumnName);
                sb.AppendLine(string.Join(",", columnNames));



                DataTable cartridgeTable = new DataTable();

                foreach (DataRow row in dataTable.Rows)
                {
                    IEnumerable<string> fields = row.ItemArray.Select(field => field.ToString());
                    sb.AppendLine(string.Join(",", fields));
```

```csharp

                    SqliteConnection cnn1;
                    SqliteCommand cmd1 = null;
                    cnn1 = new SqliteConnection("Data Source=plotterData.db;");
                    cnn1.Open();

                    string query1 = string.Format("SELECT * FROM `cartridge_reading` where `parent_id` = {0}", row[0]);
                    cmd1 = new SqliteCommand(query1, cnn1);

                    SqliteDataReader reader1 = cmd1.ExecuteReader();

                    cartridgeTable.Load(reader1);
                }

                File.WriteAllText("plotterData.csv", sb.ToString());

                StringBuilder sb1 = new StringBuilder();

                IEnumerable<string> columnNames1 = cartridgeTable.Columns.Cast<DataColumn>().
                                             Select(column => column.ColumnName);
                sb1.AppendLine(string.Join(",", columnNames1));

                foreach (DataRow row1 in cartridgeTable.Rows)
                {
                    IEnumerable<string> fields = row1.ItemArray.Select(field => field.ToString());
                    sb1.AppendLine(string.Join(",", fields));
                }

                File.WriteAllText("cartridgeData.csv", sb1.ToString());
            }
            catch (System.IO.IOException)
            {
                MessageBox.Show("Please close Excel");
            }

        }

        //Add another plotter
        private void btnAdd_Click(object sender, RoutedEventArgs e)
        {
            AddPlotter addPlotter = new AddPlotter();
            addPlotter.ParentForm = this;
            addPlotter.Show();
        }


        //Opens the settings page
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            SettingsPage sp = new SettingsPage();
            sp.Show();
            this.Close();
        }

        #region Mailer

        private void btnSendMail_Click(object sender, RoutedEventArgs e)
        {
            MAPI mapi = new MAPI();

            //mapi.AddAttachment("plotterData.db");
            mapi.AddRecipientTo("Helpdesk@goedhart-its.com");
            mapi.AddAttachment(Environment.CurrentDirectory + "\\plotterData.csv");
            mapi.AddAttachment(Environment.CurrentDirectory + "\\cartridgeData.csv");
            mapi.SendMailPopup("Testen plotter data", getMailData());
        }

        private string getMailData()
        {
            dataTable.Clear();
            SqliteConnection cnn;
            SqliteCommand cmd = null;
            cnn = new SqliteConnection("Data Source=plotterData.db;");
            cnn.Open();

            string query = "SELECT m1.*, models.plotter_type FROM printer_data m1 LEFT JOIN printer_data m2 ON (m1.serial_number = m2.serial_number AND m1.id < m2.id) INNER JOIN models on models.id = m1.model_id WHERE m2.id IS NULL";
            cmd = new SqliteCommand(query, cnn);

            string mailBody = "";

            foreach (DataRow row in dataTable.Rows)
            {
                mailBody += "Plotters: \n";
                mailBody += string.Format(row["serial_number"].ToString());
                mailBody += "\n";
                mailBody += string.Format(row["meters_printed"].ToString());
                mailBody += "\n";
                mailBody += string.Format(row["plotter_type"].ToString());
                mailBody += "\n";
                mailBody += "\n";
                mailBody += "Cartridges: \n";


                DataTable dataTableCartridge = new DataTable();
                SqliteCommand cmd1 = null;

                string query1 = string.Format("SELECT * FROM `cartridge_reading` where `parent_id` = {0}", row["id"]);
                cmd1 = new SqliteCommand(query1, cnn);

                SqliteDataReader reader1 = cmd1.ExecuteReader();
                dataTableCartridge.Load(reader1);
```

```

                    foreach (DataRow rowCartridge in dataTableCartridge.Rows)
                    {
                        mailBody += string.Format(rowCartridge
                        ["cartridge_model"].ToString());
                        mailBody += "\n";
                        mailBody += string.Format(rowCartridge["volume"].ToString
                        ());
                        mailBody += "\n";
                        mailBody += "\n";
                    }

                }

                return mailBody;
            }

            #endregion

            public void TaskCreater()
            {
                // Get the service on the local machine
                using (TaskService ts = new TaskService())
                {
                    if (ts.GetTask("Plotter Scanner") != null)
                    {
                        ts.RootFolder.DeleteTask("Plotter Scanner");
                    }
                }

                foreach (UserControl1 row in fillerGrid.Children)
                {
                    // Get the service on the local machine
                    using (TaskService ts = new TaskService())
                    {

                        var debugField = System.IO.Path.GetDirectoryName(
      Assembly.GetExecutingAssembly().GetName().CodeBase);

                        debugField = debugField.Substring(6);

                        var filename = debugField + @"/ghWebscraper.exe";

                        //Start the Converted python file and pass the paramater
                        string arguments = string.Format(@"{0} {1} {2} {3}",
                        row.lblMerk.Uid.ToString(), row.plotterIp,
                        Settings.Default.bedrijfsNaam, row.lblNaam.Content);

                        TaskDefinition td = ts.NewTask();

                        if (ts.GetTask("Plotter Scanner") != null)
                        {
                            td = ts.GetTask("Plotter Scanner").Definition;
                        }

                        // Create a new task definition and assign properties

                        td.RegistrationInfo.Description = "Scans plotter";
                        td.RegistrationInfo.Author = "Goedhart Groep";

                        if (td.Triggers.Count == 0)
                        {
                            // Create a trigger that will fire the task at this time every day
                            td.Triggers.Add(new DailyTrigger { DaysInterval = 1 });
                        }

                        // Create an action that will launch Notepad whenever the trigger fires
                        td.Actions.Add(new ExecAction(filename, arguments, debugField));

                        // Register the task in the root folder
                        ts.RootFolder.RegisterTaskDefinition(@"Plotter Scanner", td);


                    }

                }

                using (TaskService ts = new TaskService())
                {
                    var debugField = System.IO.Path.GetDirectoryName(
        Assembly.GetExecutingAssembly().GetName().CodeBase);

                    debugField = debugField.Substring(6);

                    var filename = debugField + @"/NewWay.exe";

                    TaskDefinition td = ts.NewTask();

                    if (ts.GetTask("Plotter Scanner") != null)
                    {
                        td = ts.GetTask("Plotter Scanner").Definition;
                    }

                    // Create a new task definition and assign properties

                    td.RegistrationInfo.Description = "Scans plotter";
                    td.RegistrationInfo.Author = "Goedhart Groep";

                    if (td.Triggers.Count == 0)
                    {
                        // Create a trigger that will fire the task at this time every day
                        td.Triggers.Add(new DailyTrigger { DaysInterval = 1 });
                    }

                    // Create an action that will launch Notepad whenever the trigger fires
                    td.Actions.Add(new ExecAction(filename, null, debugField));

                    // Register the task in the root folder
                    ts.RootFolder.RegisterTaskDefinition(@"Plotter Scanner", td);
                }
            }

            #region Scanning
            public void RunScan(string Merk, string IP, string Naam)
            {
                var debugField = System.IO.Path.GetDirectoryName(
        Assembly.GetExecutingAssembly().GetName().CodeBase);

                debugField = debugField.Substring(6);

                var filename = debugField + @"/ghWebscraper.exe";

                //Start the Converted python file and pass the paramater
                string arguments = string.Format(@"{0} {1} {2} {3}", Merk, IP, Settings.Default.bedrijfsNaam, Naam);

                //Process myProcess = new Process();
                //myProcess.Exited += new EventHandler(myProcess_Exited);
                //myProcess.StartInfo.FileName = filename;
                //myProcess.StartInfo.Arguments = arguments;
                //myProcess.Start();

                doStuff(filename, arguments);
            }

            async System.Threading.Tasks.Task doStuff(string fileName, string args)
            {
                DisableWhileScanning();
                await RunProcessAsync(fileName, args);

                //MahApps.Metro.IconPacks.PackIconFontAwesome fe = btnScan.Content as MahApps.Metro.IconPacks.PackIconFontAwesome;
                //fe.Kind = MahApps.Metro.IconPacks.PackIconFontAwesomeKind.BinocularsSolid;
                //btnScan.IsEnabled = true;

                fillerGrid.RowDefinitions.Clear();
                fillerGrid.Children.Clear();
                LoadData();
            }

            public static async Task<int> RunProcessAsync(string fileName, string args)
            {
                using (var process = new Process
                {
                    StartInfo =
            {
                FileName = fileName, Arguments = args,
                UseShellExecute = false, CreateNoWindow = true,
                RedirectStandardOutput = true, RedirectStandardError = true
            },
                    EnableRaisingEvents = true
                })
                {
                    return await RunProcessAsync(process).ConfigureAwait(false);
                }
            }
            private static Task<int> RunProcessAsync(Process process)
            {
                var tcs = new TaskCompletionSource<int>();

                process.Exited += (s, ea) => tcs.SetResult(process.ExitCode);
                process.OutputDataReceived += (s, ea) => Console.WriteLine(ea.Data);
                process.ErrorDataReceived += (s, ea) => Console.WriteLine("ERR: " + ea.Data);

                bool started = process.Start();
                if (!started)
                {
                    //you may allow for the process to be re-used (started = false)
                    //but I'm not sure about the guarantees of the Exited event in such a case
                    throw new InvalidOperationException("Could not start process: " + process);
                }

                process.BeginOutputReadLine();
                process.BeginErrorReadLine();

                return tcs.Task;

            }

            #endregion
        }
    }
```