protected override void OnLaunched(LaunchActivatedEventArgs e)

      {

{…}

            // Do not repeat app initialization when the Window already has content,

            // just ensure that the window is active

            if ( rootFrame == null )

            {

                {…}

            }

            if (!TryToNavigateToTileReference(rootFrame, e))

            {

                if (rootFrame.Content == null)

                {

                    // When the navigation stack isn't restored navigate to the first page,

                    // configuring the new page by passing required information as a navigation

                    // parameter

                    rootFrame.Navigate(typeof(ContosoCookbook.MainPage), e.Arguments);

                }

            }

         // Ensure the current window is active

         Window.Current.Activate();

            //[WP8SL\_TO\_UWP] The following code was added to emulate the default behavior of

            // the back button on WP8SL

            rootFrame.Navigated += (s, a) =>

              {

                  if(rootFrame.CanGoBack)

                  {

                      Windows.UI.Core.SystemNavigationManager.GetForCurrentView().AppViewBackButtonVisibility = Windows.UI.Core.AppViewBackButtonVisibility.Visible;

                  }

                  else

                      Windows.UI.Core.SystemNavigationManager.GetForCurrentView().AppViewBackButtonVisibility = Windows.UI.Core.AppViewBackButtonVisibility.Collapsed;

              };

          {…}

bool TryToNavigateToTileReference(Frame rootFrame, LaunchActivatedEventArgs e)

     {

           if (e.TileId.Contains("GroupDetailPage.xaml"))

           {

               rootFrame.Navigate(typeof(ContosoCookbook.GroupDetailPage), e.Arguments);

               return true;

           }

           else if (e.TileId.Contains("MainPage.xaml"))

           {

               rootFrame.Navigate(typeof(ContosoCookbook.MainPage), e.Arguments);

               return true;

           }

           else if (e.TileId.Contains("RecipeDetailPage.xaml"))

           {

               rootFrame.Navigate(typeof(ContosoCookbook.RecipeDetailPage), e.Arguments);

               return true;

           }

           else

               return false;

     }

protected async override void OnNavigatedTo(Windows.UI.Xaml.Navigation.NavigationEventArgs e)

      {

         if ( !App.Recipes.IsLoaded )

         {

                ProgressPanel.Visibility = Windows.UI.Xaml.Visibility.Visible;

                await Task.Delay(5000);

                await App.Recipes.LoadLocalDataAsync();

                ProgressPanel.Visibility = Windows.UI.Xaml.Visibility.Collapsed;

            }

         base.OnNavigatedTo(e);

      }

<Grid.RowDefinitions>

            <RowDefinition

                Height="Auto" />

            <RowDefinition

                Height="Auto" />

            <RowDefinition

                Height="\*" />

        </Grid.RowDefinitions>

        <StackPanel Visibility="Collapsed"

            x:Name="progressIndicator" Orientation="Vertical" HorizontalAlignment="Stretch">

            <ProgressBar

                IsIndeterminate="True"

                Height="10"

                HorizontalAlignment="Stretch">

            </ProgressBar>

            <TextBlock

                Text="Loading data, please wait..." Foreground="Red" />

        </StackPanel>

public async System.Threading.Tasks.Task LoadLocalDataAsync()

      {

            var sri = await Windows.ApplicationModel.Package.Current.InstalledLocation.GetFileAsync("Data\\Recipes.txt");

            List<Type> types = new List<Type>();

            types.Add(typeof(RecipeDataItem));

            types.Add(typeof(RecipeDataGroup));

            types.Add(typeof(RecipeDataCommon));

            DataContractJsonSerializer deserializer = new DataContractJsonSerializer(typeof(IEnumerable<RecipeDataItem>), types);

            var fileStream =  await sri.OpenStreamForReadAsync();

            var fileString =await new StreamReader(fileStream).ReadToEndAsync();

            var mStream = new MemoryStream(Encoding.UTF8.GetBytes(fileString));

            IEnumerable<RecipeDataItem> data = (IEnumerable<RecipeDataItem>)deserializer.ReadObject(mStream);

            await LoadUserImagesLocalDataAsync();

             {…}

        }

protected async override void OnNavigatedTo(Windows.UI.Xaml.Navigation.NavigationEventArgs e)

      {

            uri = this.BaseUri.AbsolutePath + "?" + e.Parameter;

         var navigationArguments = (e.Parameter ?? "").ToString().Split('&').Where(t => !string.IsNullOrWhiteSpace(t)).Select(t => t.Split('=')).ToDictionary(t => t[0], t => t[1]);

         string UniqueId = "";

         UniqueId = navigationArguments["ID"];

         if ( !App.Recipes.IsLoaded )

            await App.Recipes.LoadLocalDataAsync();

         await NavigateToRecipe(UniqueId);

         base.OnNavigatedTo(e);

      }

public object IValueConverter.Convert(object value, Type targetType, object parameter, string language)

        {

            var file = value.ToString();

            StorageFolder folder = ApplicationData.Current.LocalFolder;

            Stream stream = null;

            try

            {

                var task1 = folder.OpenStreamForReadAsync(file);

                Task.WaitAll(task1);

                stream = task1.Result;

                Windows.UI.Xaml.Media.Imaging.BitmapImage bitmap = new Windows.UI.Xaml.Media.Imaging.BitmapImage();

                bitmap.CreateOptions = BitmapCreateOptions.None;

                bitmap.SetSource(stream.AsRandomAccessStream());

                stream.Dispose();

                return bitmap;

            }

            catch

            {

                return null;

            }

        }

async void btnTakePicture\_Click(object sender, Windows.UI.Xaml.RoutedEventArgs e)

        {

            try

            {

                CameraCaptureUI cameraUI = new CameraCaptureUI();

                Windows.Storage.StorageFile capturedMedia =

                    await cameraUI.CaptureFileAsync(CameraCaptureUIMode.Photo);

                if (capturedMedia != null)

                {

                    StorageFolder folder = ApplicationData.Current.LocalFolder;

                    await folder.CreateFolderAsync(item.Group.Title, CreationCollisionOption.OpenIfExists);

                    string fileName = string.Format("{0}\\{1}.jpg", item.Group.Title, DateTime.Now.ToString("dd-MM-yyyy HH-mm-ss"));

                    StorageFile file = await folder.CreateFileAsync(fileName);

                    await capturedMedia.CopyAndReplaceAsync(file);

                    if (null == item.UserImages)

                        item.UserImages = new System.Collections.ObjectModel.ObservableCollection<string>();

                    item.UserImages.Add(fileName);

                }

            }

            catch (System.InvalidOperationException ex)

            {

                await (new Windows.UI.Popups.MessageDialog("An error occurred.")).ShowAsync();

            }

        }

void btnShareShareTask\_Click(object sender, Windows.UI.Xaml.RoutedEventArgs e)

     {

           DataTransferManager dataTransferManager = DataTransferManager.GetForCurrentView();

           dataTransferManager.DataRequested += DataTransferManager\_DataRequested;

           DataTransferManager.ShowShareUI();

       }

       private async void DataTransferManager\_DataRequested(DataTransferManager sender, DataRequestedEventArgs args)

       {

           DataRequest request = args.Request;

           request.Data.Properties.Title = "Share Example";

           request.Data.Properties.Description = "A demonstration on how to share";

           if (null != item.UserImages && item.UserImages.Count > 0)

           {

               var file = await ApplicationData.Current.LocalFolder.GetFileAsync(item.UserImages[0]);

               request.Data.SetStorageItems(new List<StorageFile>() { file } );

           }

       }

public static void SetReminder(RecipeDataItem item)

           {

               var toastNotifier = Windows.UI.Notifications.ToastNotificationManager.CreateToastNotifier();

               if (!IsScheduled(item.UniqueId))

               {

                   var schedule = toastNotifier.GetScheduledToastNotifications().FirstOrDefault(x => x.Id == item.UniqueId);

                   if (null != schedule)

                       toastNotifier.RemoveFromSchedule(schedule);

                   Windows.Data.Xml.Dom.XmlDocument doc =

 ToastNotificationManager.GetTemplateContent(ToastTemplateType.ToastText02);

                   var textLines = doc.GetElementsByTagName("text");

                   textLines[0].InnerText = item.Title;

                   textLines[1].InnerText = "Have you finished cooking?";

                   ScheduledToastNotification reminder = new ScheduledToastNotification(doc, DateTime.Now.AddSeconds(10));

                   reminder.Id = item.UniqueId;

                   toastNotifier.AddToSchedule(reminder);

               }

               else

               {

                   var schedule = toastNotifier.GetScheduledToastNotifications().FirstOrDefault(x => x.Id == item.UniqueId);

                   if (null != schedule)

                       toastNotifier.RemoveFromSchedule(schedule);

               }

           }

        public static bool IsScheduled(string name)

        {

               var schedule = Windows.UI.Notifications.ToastNotificationManager.CreateToastNotifier().GetScheduledToastNotifications().FirstOrDefault(x => x.Id == name);

               if (schedule == null)

               {

                   return false;

               }

               else

               {

                   return true;

               }

           }