GEOG 8990, Spring 2018, Tobias Fimpel

Summary/Reflection Write-up 2

The objective of my work on this Add-in called was to familiarize myself with parts of the ArcGIS Pro SDK that I would likely need to incorporate into the larger tool that I would like to build towards the end of the semester:

* Working with listboxes and different kinds of buttons
* Displaying and querying map layers (filter, symbolize, label, etc.)
* Creating maps and layouts
* Accessing layout properties (that are not exposed through the SDK API) via CIM

What I did not want to do is get caught up in overall logic/implementation questions. Thus in this Add-in there are many variables hard-coded and error catching is almost non-existent. Also I am not using MVVM practices. Instead the functionality is driven by events attached to UI elements. I am quite happy with the result and feel like this has been a very worthwhile learning experience.

The following pages show screenshots and key pieces of code. The complete Add-in code is available at <https://github.com/TFimpel/ArcGIS-Pro-February-Sandbox> .

----------------------------------------------------------------------------------------------------------------------------------------------------------------

1. The add in Button is enabled only when there is an active map. This is achieved through the condition property in DAML file.

<controls>

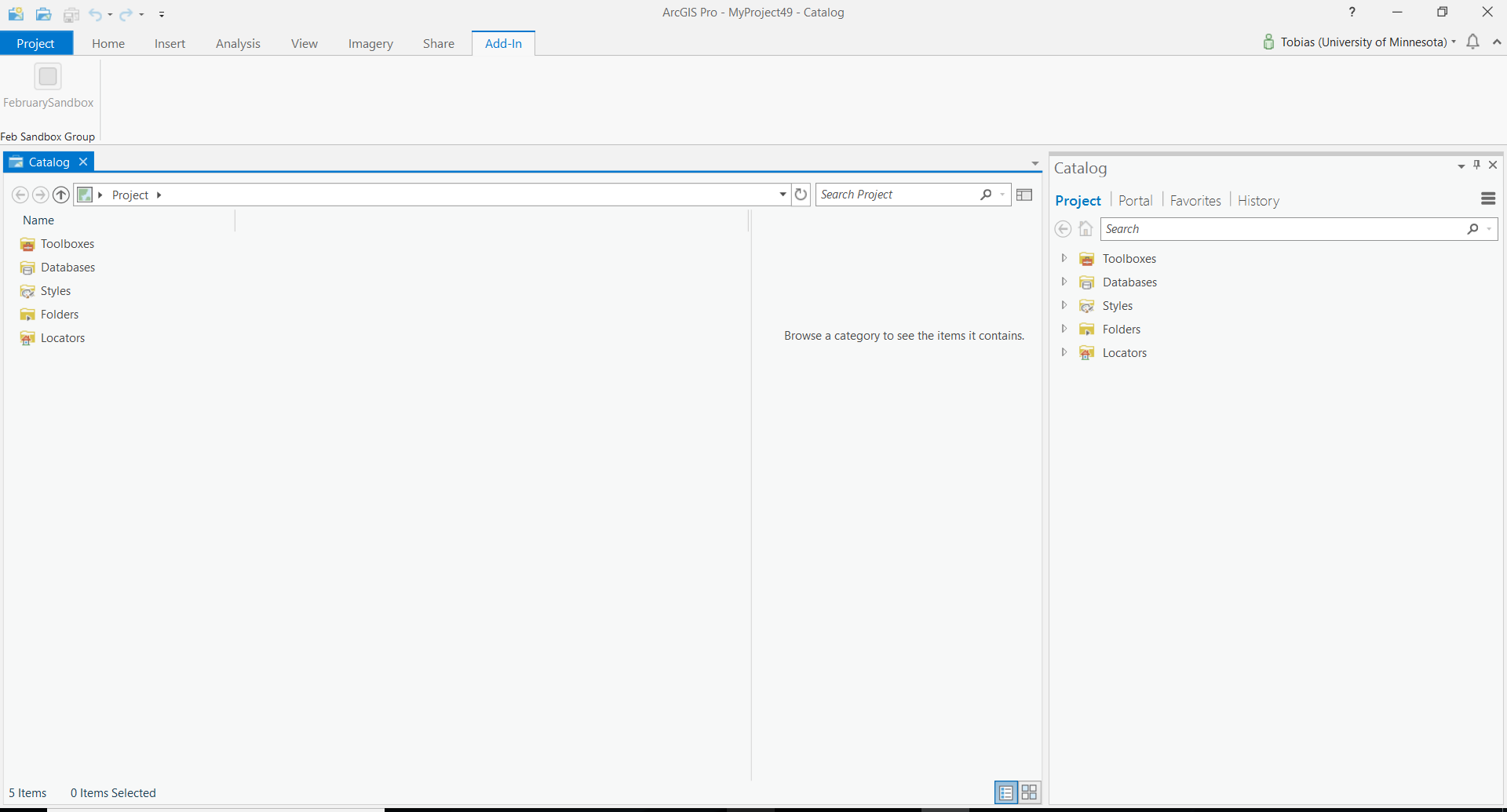
<button id="ProAppModule2\_Dockpane1\_ShowButton" caption="FebruarySandbox" condition="esri\_mapping\_mapPane"

className="Dockpane1\_ShowButton" loadOnClick="true" largeImage="Images\GenericButtonPurple32.png">

<tooltip heading="February Sandbox Tooltip">February Sandbox Add-In<disabledText /></tooltip>

</button>

</controls>



1. Once a new map has been created the Add-in button can be clicked. A dockpane opens. The dockpane contains a series of textboxes, buttons, a listbox, and radiobuttons. Only small portion of the code is shown below.

[…]

<TextBlock Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" TextWrapping="Wrap">

1) Start by adding a layer of US Rivers to your map</TextBlock>

<Button Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" Name="BtnGetLayer"

IsEnabled="True"

Content="Add layer to Map"

Click="btnAddLayer1\_ClickAsync"

Style="{DynamicResource Esri\_Button}"/>

<TextBlock Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="3" TextWrapping="Wrap">

2) Next add the map to a new layout with a legend and a title element

</TextBlock>

<Button Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="3" Name="BtnAddMapToNewLayout"

IsEnabled="True"

Content="Make Layout"

Click="btnAddMapToNewLayout"

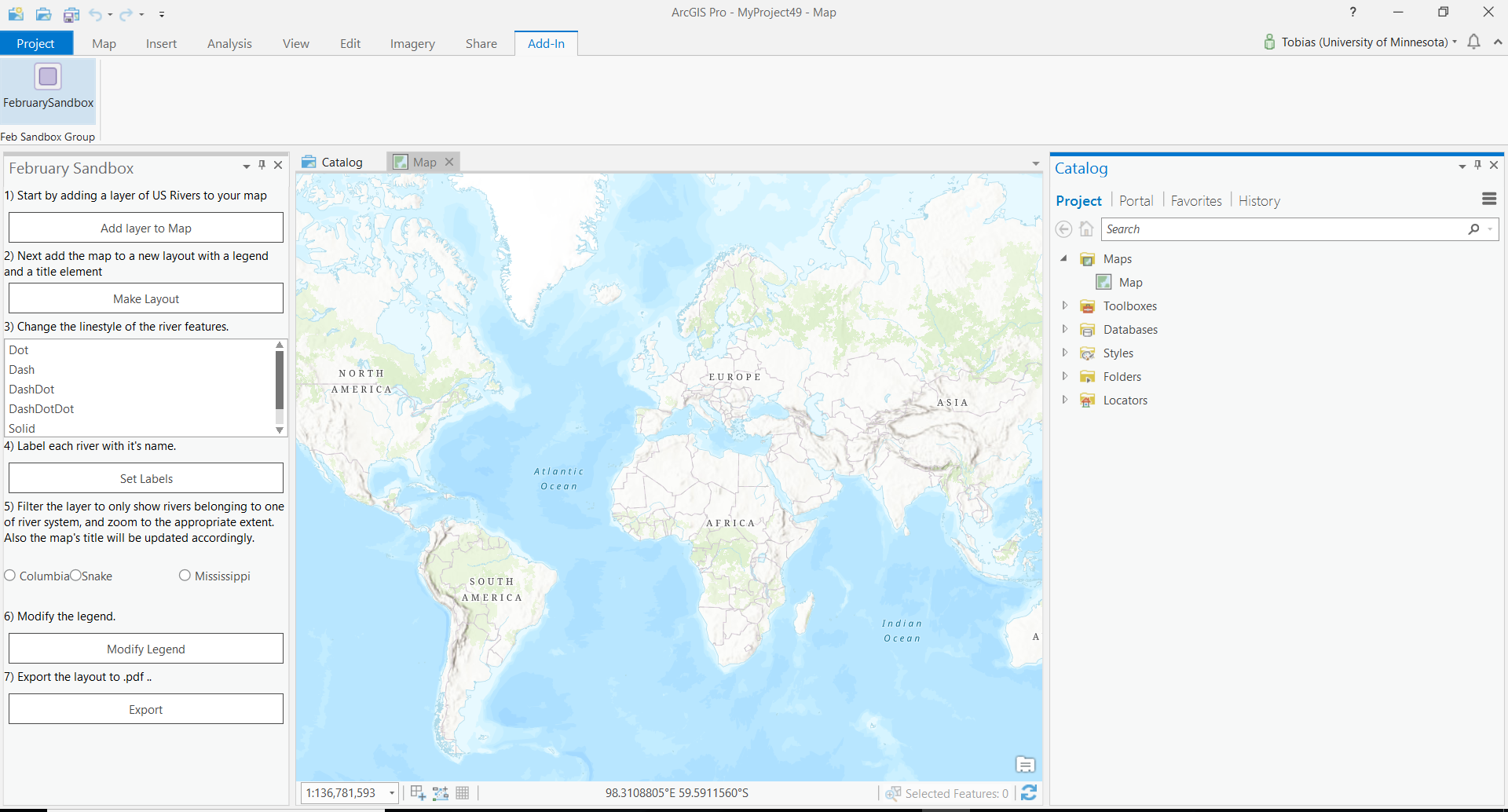
Style="{DynamicResource Esri\_Button}"/>

<TextBlock Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="3" TextWrapping="Wrap">

3) Change the linestyle of the river features.

</TextBlock>

[…]



1. The “Add layer to Map” button asynchronously adds the feature layer from the url <https://services.arcgis.com/BG6nSlhZSAWtExvp/arcgis/rest/services/US_Rivers/FeatureServer/0> to the active map.

[…]

public async Task<Layer> AddLayer(string uri)

{

return await QueuedTask.Run(() =>

{

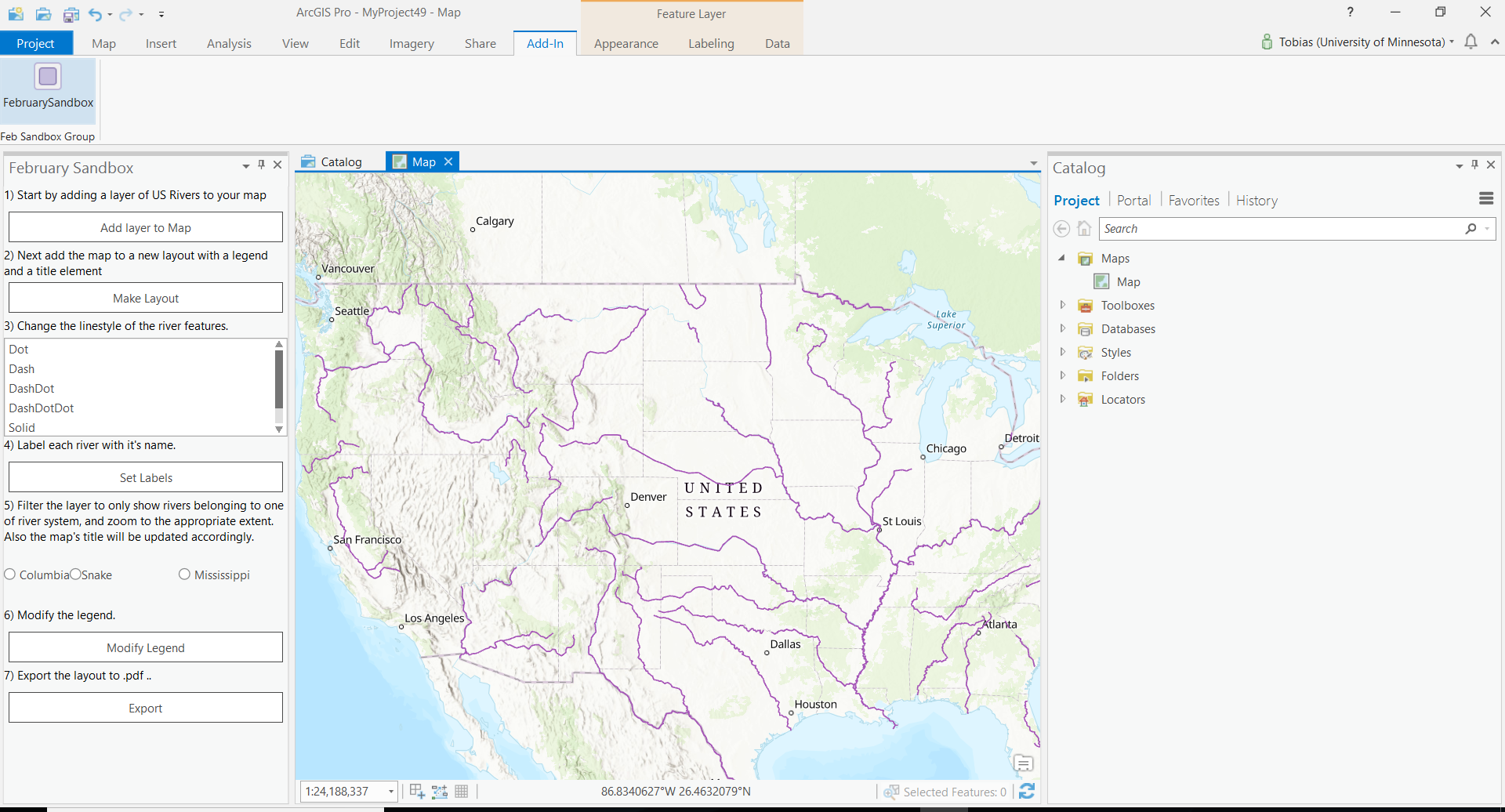
Map map = MapView.Active.Map;

return LayerFactory.Instance.CreateLayer(new Uri(uri), map);

});

}

[…]



1. The “Make Layout” button creates a new layout including title and legend elements, adds the map to the layout, and makes the layout the active view tab.

[…]

public static async Task makeNewLayout()

{

Layout newLayout = await QueuedTask.Run<Layout>(() =>

{

newLayout = LayoutFactory.Instance.CreateLayout(8.5, 11, LinearUnit.Inches);

newLayout.SetName("MY NEW LAYOUT");

return newLayout;

});

await AddMapToNewLayout(newLayout);

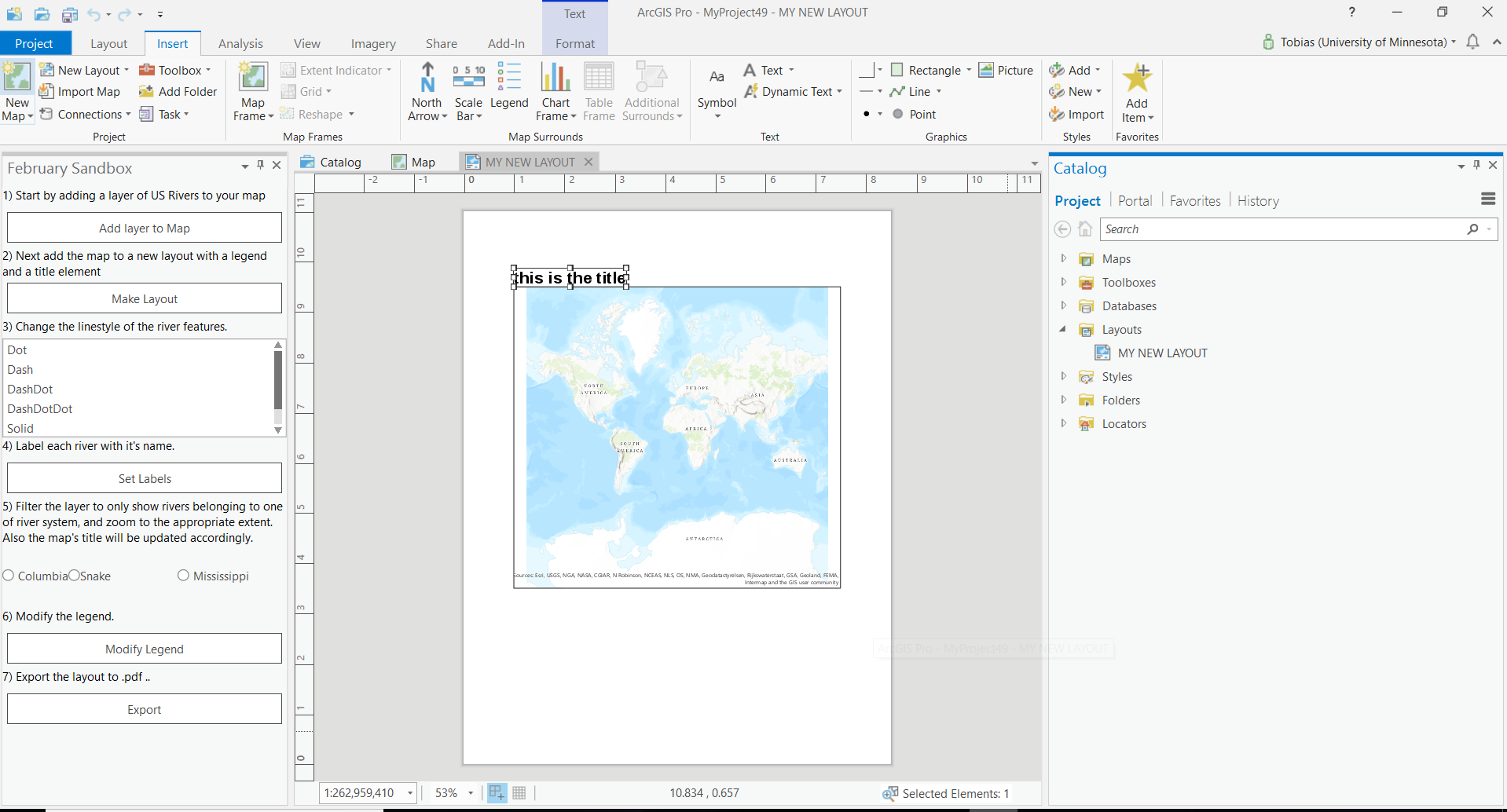
await addLegendToLayout(newLayout);

await AddTtitleToNewLayout(newLayout);

ILayoutPane iNewLayoutPane = await ProApp.Panes.CreateLayoutPaneAsync(newLayout);

}

[…]



1. Selecting a value from the ListBox changes the line renderer to a blue line. The value determines the line style (i.e. dotted, dashed, solid, etc.)

[…]

<ListBox Grid.Row="5" Grid.Column="0" Grid.ColumnSpan="3" Grid.RowSpan="1" Name="linestyle">

<ListBoxItem Selected="btnChangeLayerSymbology" Tag="Dot" Content="Dot"></ListBoxItem>

<ListBoxItem Selected="btnChangeLayerSymbology" Tag="Dash" Content="Dash"></ListBoxItem>

<ListBoxItem Selected="btnChangeLayerSymbology" Tag="DashDot" Content="DashDot"></ListBoxItem>

<ListBoxItem Selected="btnChangeLayerSymbology" Tag="DashDotDot" Content="DashDotDot"></ListBoxItem>

<ListBoxItem Selected="btnChangeLayerSymbology" Tag="Solid" Content="Solid"></ListBoxItem>

</ListBox>

[…]

internal static Task SimpleRendererLine(FeatureLayer featureLayer, String lineStyle)

{

return QueuedTask.Run(() =>

{

if (lineStyle == "Dash") {

var lineSymbol = SymbolFactory.Instance.ConstructLineSymbol(ColorFactory.Instance.BlueRGB, 3, SimpleLineStyle.Dash);

CIMSimpleRenderer renderer = featureLayer.GetRenderer() as CIMSimpleRenderer;

renderer.Symbol = lineSymbol.MakeSymbolReference();

featureLayer.SetRenderer(renderer);

}

else if (lineStyle == "Dot")

{

var lineSymbol = SymbolFactory.Instance.ConstructLineSymbol(ColorFactory.Instance.BlueRGB, 3, SimpleLineStyle.Dot);

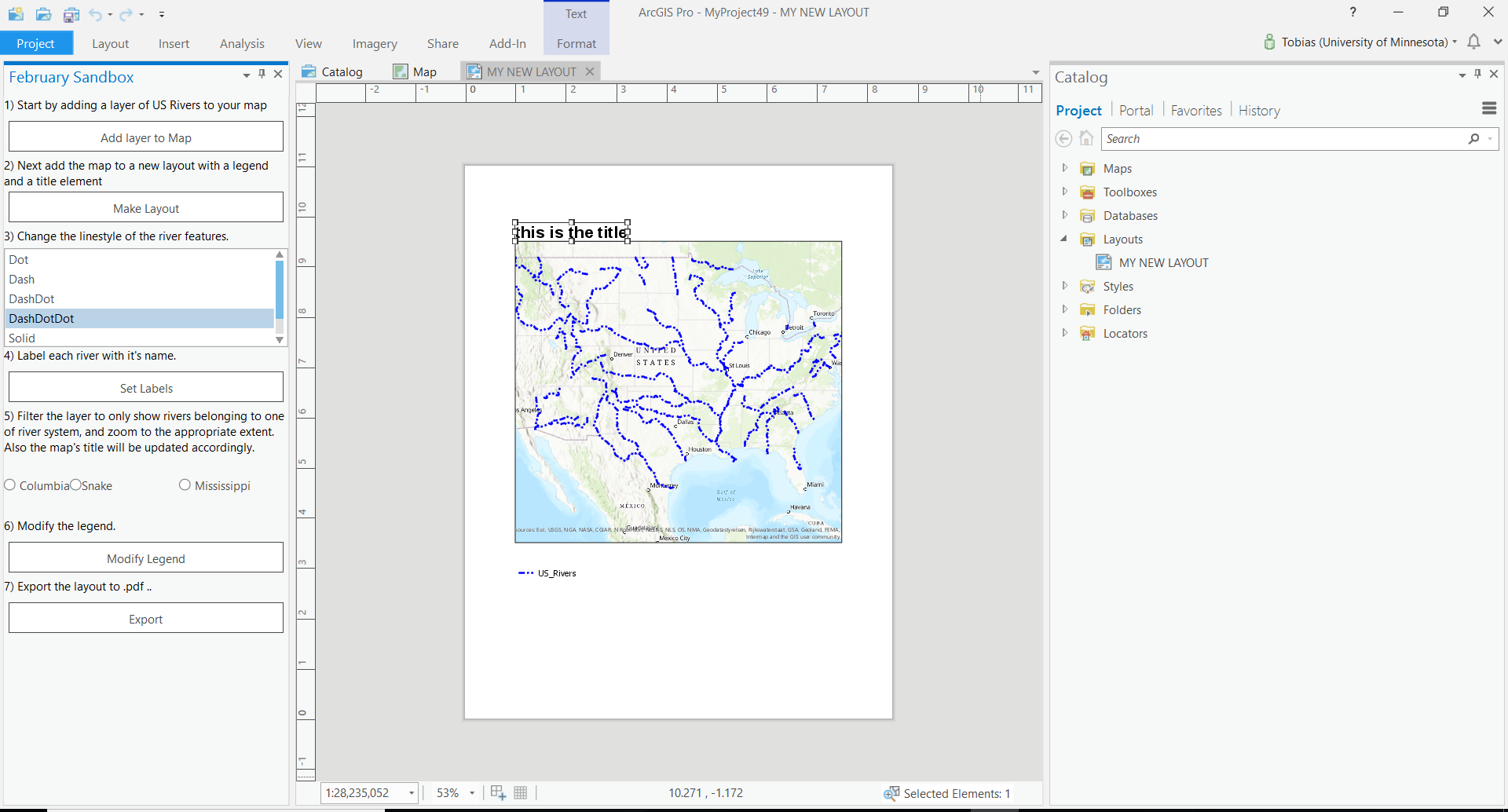
CIMSimpleRenderer renderer = featureLayer.GetRenderer() as CIMSimpleRenderer;

renderer.Symbol = lineSymbol.MakeSymbolReference();

featureLayer.SetRenderer(renderer);

}

[…]



1. Clicking the set labels button turns on labels on the feature layer.

public async Task GetFeatureLayerFromMapAndChangeLabels()

{

var riverslayer = (MapView.Active.Map.Layers.First(layer => true) as FeatureLayer);

await SimpleLabels(riverslayer);

}

[…]

internal static Task SimpleLabels(FeatureLayer featureLayer)

{

return QueuedTask.Run(() =>

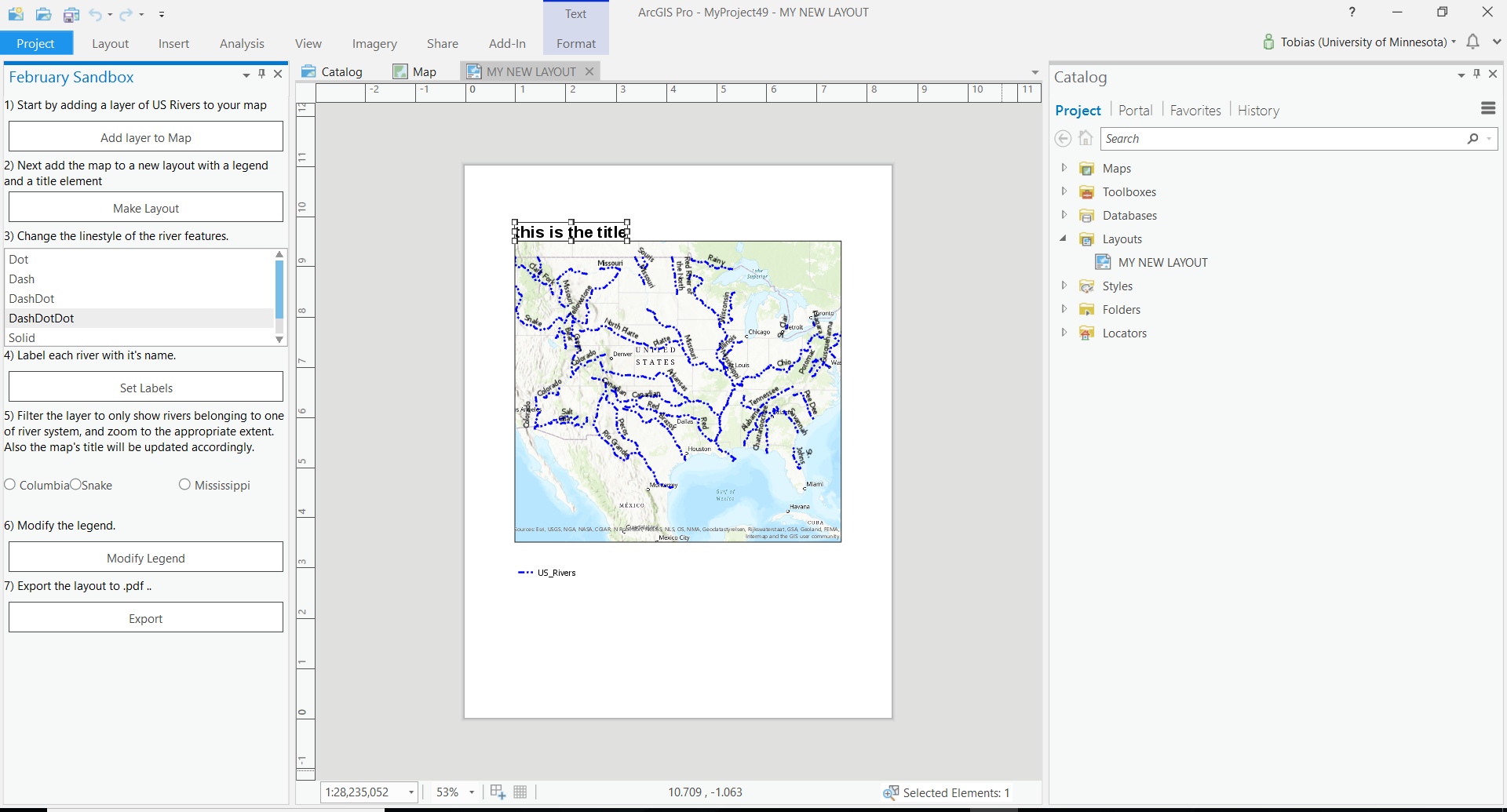
{

featureLayer.SetLabelVisibility(true);

}

);

}



1. Choosing one of the radio buttons sets a definition query on the layer, updates the title element’s text, and then zooms to the layer’s extent.

[…]

<RadioButton Grid.Row="9" Grid.Column="0" Content="Columbia" Click="btnChangeLayerFilter" Tag="Columbia" IsChecked="False"/>

<RadioButton Grid.Row="9" Grid.Column="1" Content="Snake" Click="btnChangeLayerFilter" Tag="Snake" IsChecked="False"/>

<RadioButton Grid.Row="9" Grid.Column="2" Content="Mississippi" Click="btnChangeLayerFilter" Tag="Mississippi" IsChecked="False"/>

[…]

public async void btnChangeLayerFilter(object sender, EventArgs e)

{

var featureName = ((RadioButton)sender).Tag;

await GetFeatureLayerFromMapAndChangeFilter(featureName.ToString());

}

[…]

public async Task GetFeatureLayerFromMapAndChangeFilter(String featureName)

{

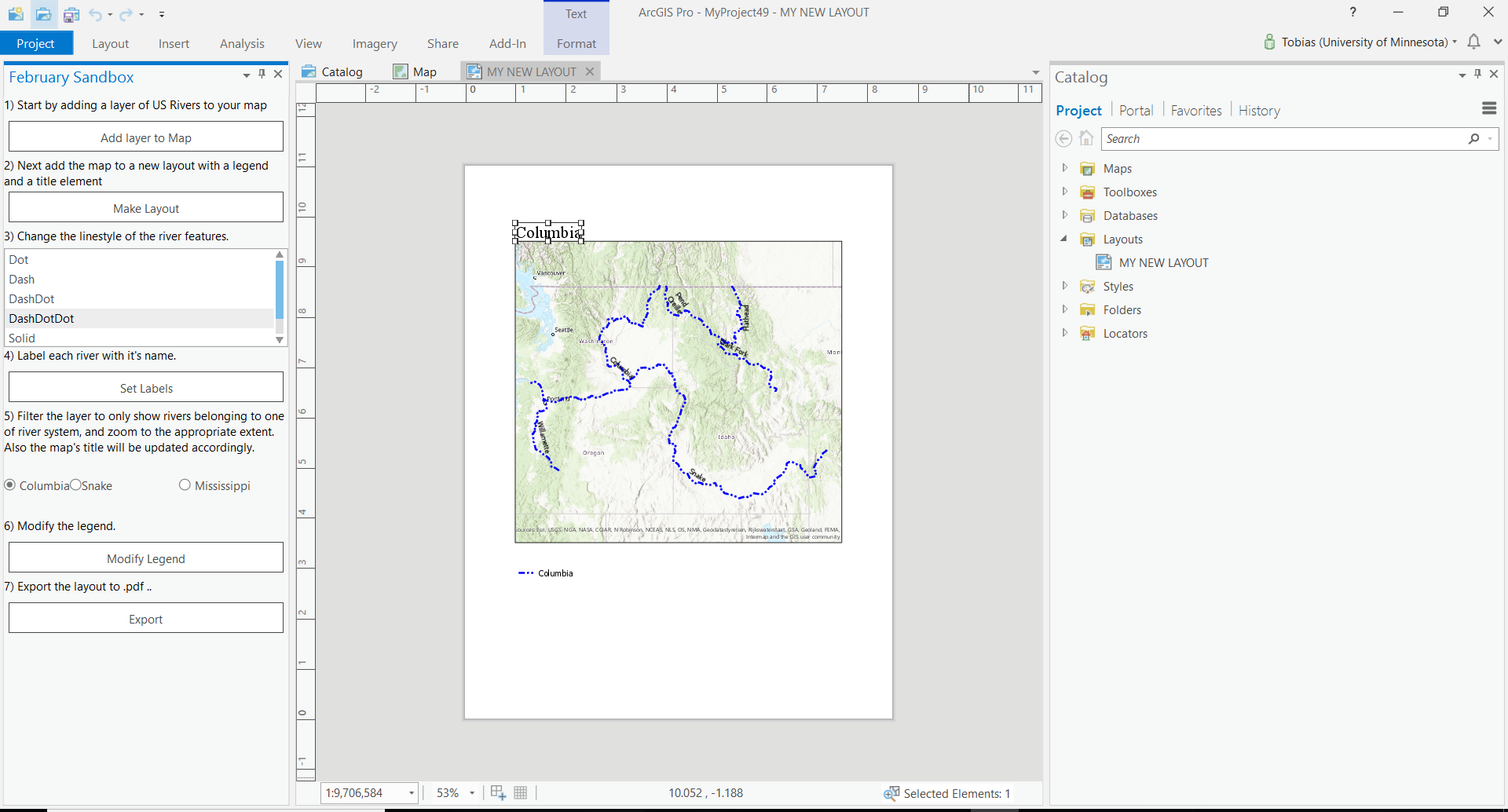
var riverslayer = (MapView.Active.Map.Layers.First(layer => true) as FeatureLayer);

await SimpleFilter(riverslayer, featureName);

await UpdateTitle(featureName);

await ZoomToLayer(riverslayer);

}



1. Clicking the “Modify Legend” button changes the font color of the layer name in the legend to blue.

public async Task modifyLegend()

{

await QueuedTask.Run(() =>

{

LayoutProjectItem layoutItem = Project.Current.GetItems<LayoutProjectItem>().FirstOrDefault(item => item.Name.Equals("MY NEW LAYOUT"));

Layout layout = layoutItem.GetLayout();

if (layout != null)

{

Element element = layout.FindElement("New Legend");

if (element != null)

{

CIMLegend CIMLegend = element.GetDefinition() as CIMLegend;

CIMLegend.Locked = true;

foreach (var legendItem in CIMLegend.Items)

{

var itemLabel = legendItem.LabelSymbol.Symbol as CIMTextSymbol;

foreach (var symlayer in ((CIMPolygonSymbol)itemLabel.Symbol).SymbolLayers)

{

if (symlayer is ArcGIS.Core.CIM.CIMSolidFill)

{

var itemLabelSym = (CIMSolidFill)symlayer;

itemLabelSym.Color.Values[0] = 0;

itemLabelSym.Color.Values[1] = 0;

itemLabelSym.Color.Values[2] = 255;

}

}

}

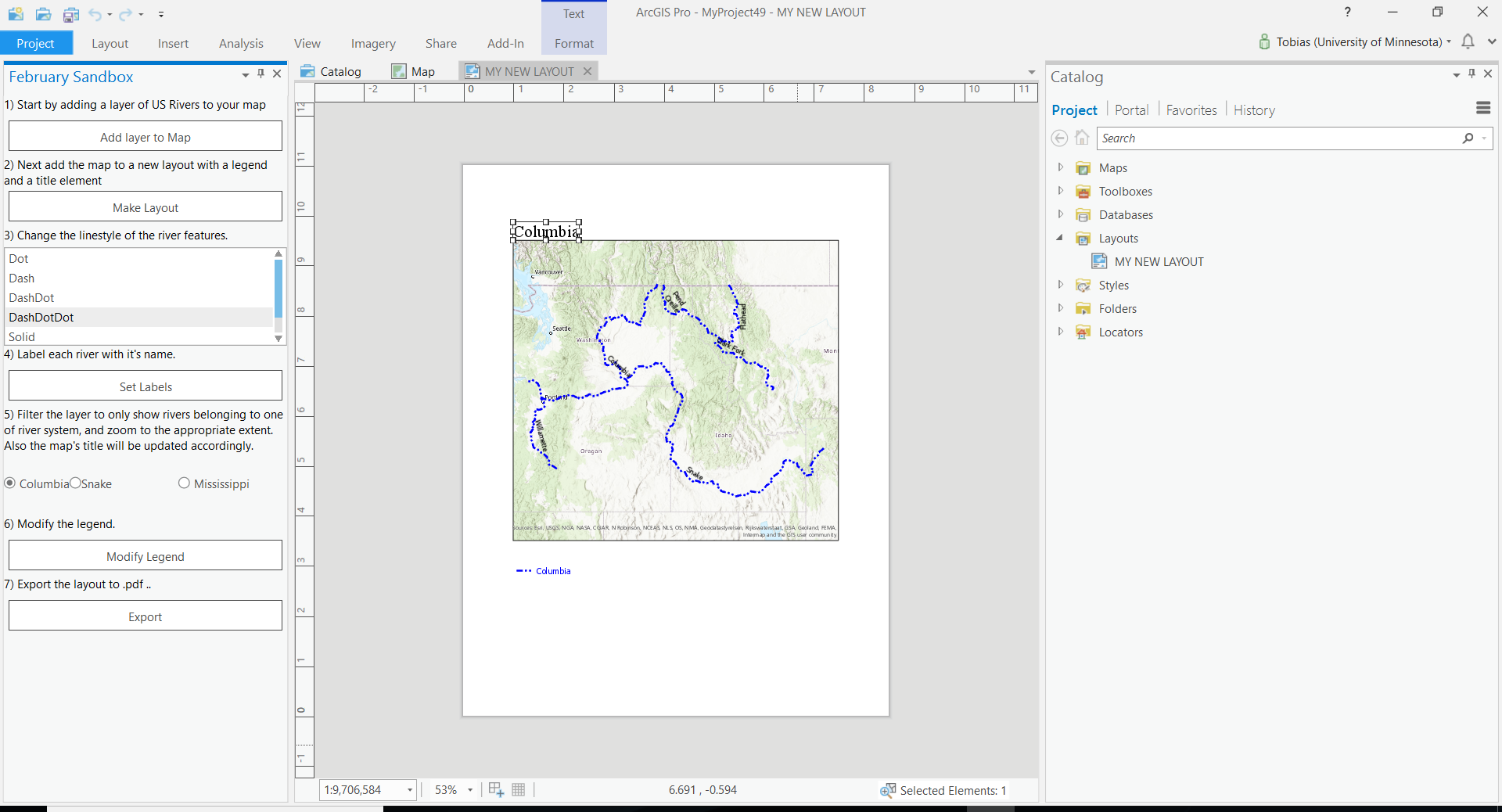
element.SetDefinition(CIMLegend);

}

}

});

}



1. Clicking the export button exports the layout to a .pdf file and displays a message box.

public async void btnExportLayout(object sender, EventArgs e)

{

LayoutProjectItem layoutItem = Project.Current.GetItems<LayoutProjectItem>().FirstOrDefault(item => item.Name.Equals("MY NEW LAYOUT"));

if (layoutItem != null)

{

await QueuedTask.Run(() =>

{

Layout layout = layoutItem.GetLayout();

if (layout == null)

return;

PDFFormat PDF = new PDFFormat()

{

Resolution = 300,

OutputFileName = @"C:\Users\fimpe\OneDrive\MGIS\GEOG 8990 Spring 2018\Layout.pdf"

};

if (PDF.ValidateOutputFilePath())

{

layout.Export(PDF);

MessageBox.Show("The layout was exported to a pdf.", "Well done!", MessageBoxButton.OK);

}

});

}

}

