UWP-021 - Implementing a Simple Hamburger Navigation Menu

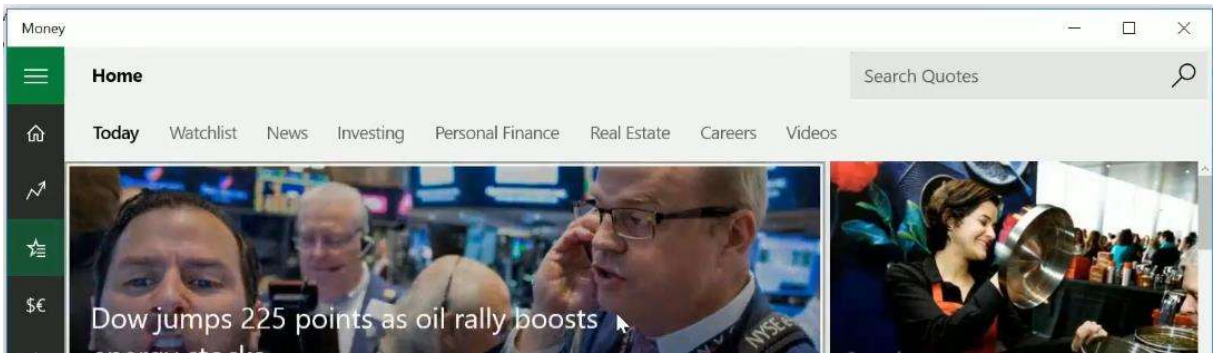
In this lesson we are going to combine several ideas that we covered in previous lessons to create what I call a "poor man's" hamburger navigation Layout. Why am I calling it a "poor man's" version? Because I'm trying to give you the simplest, cleanest, clearest path so that you can implement hamburger navigation right away in your Windows 10 apps.

There is a "rich man's" version of this detailed at Jerry Nixon's blog here at the shortcut on screen.

http://bit.do/hamburger-nav

Like I said, Jerry Nixon works for Microsoft. He has done an awesome job with this article: "Implementing an Awesome Hamburger Button with XAML's SplitView Control." The reason why I would call it a "rich man's" version is because it's more robust. It includes many additional features. But it's also a bit more complex if you are, indeed, an absolute beginner. Just keep that in mind. You might want to visit it and bookmark it for later, maybe after we go through several more lessons. You'll be able to use his version instead of mine.

What I want to do now is to recreate the hamburger layout that we saw in the Money app.



When I click on the hamburger little icon at the top, the SplitView opens up. When I click it again, it closes and all we can see is the icons.

Furthermore, whatever we currently select, you can see that one of the secondary colors, in this case, a darker green color, highlights the fact that we're no longer on the Home tab but rather we're on what I guess is the Watchlist.

I've created a new project named HamburgerExample. All I've done prior to creating this lesson is just open the App.xaml.cs and I removed the Frame counter.

In the MainPage.xaml I create a couple of RowDefinitions seting the Height="Auto" in the Row one and setting the Height="*" on Row two. Next, I'll create a RelativePanel.

```
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <RelativePanel>



        </RelativePanel>

    </Grid>
</Page>
```

Inside of the RelativePanel I'll add a Button. I'll come back to that in a moment, but that will be our hamburger button.

Beneath that I'll add a SplitView and place it into Grid.Row="1" -- that second Row that takes up the * size. This is where we're going to do the majority of our work.

```
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <RelativePanel>
            <Button />
        </RelativePanel>
        <SplitView Grid.Row="1">

        </SplitView>

    </Grid>
</Page>
```

Inside of this SplitView I'm going to use a ListBox control for all the reasons I talked about in that previous lesson. You'll see it all come together here in just a moment. The ListBox is going to require us to create a number of ListBoxItems. I know we're going to have several of those as well. I'll just copy and paste a couple.

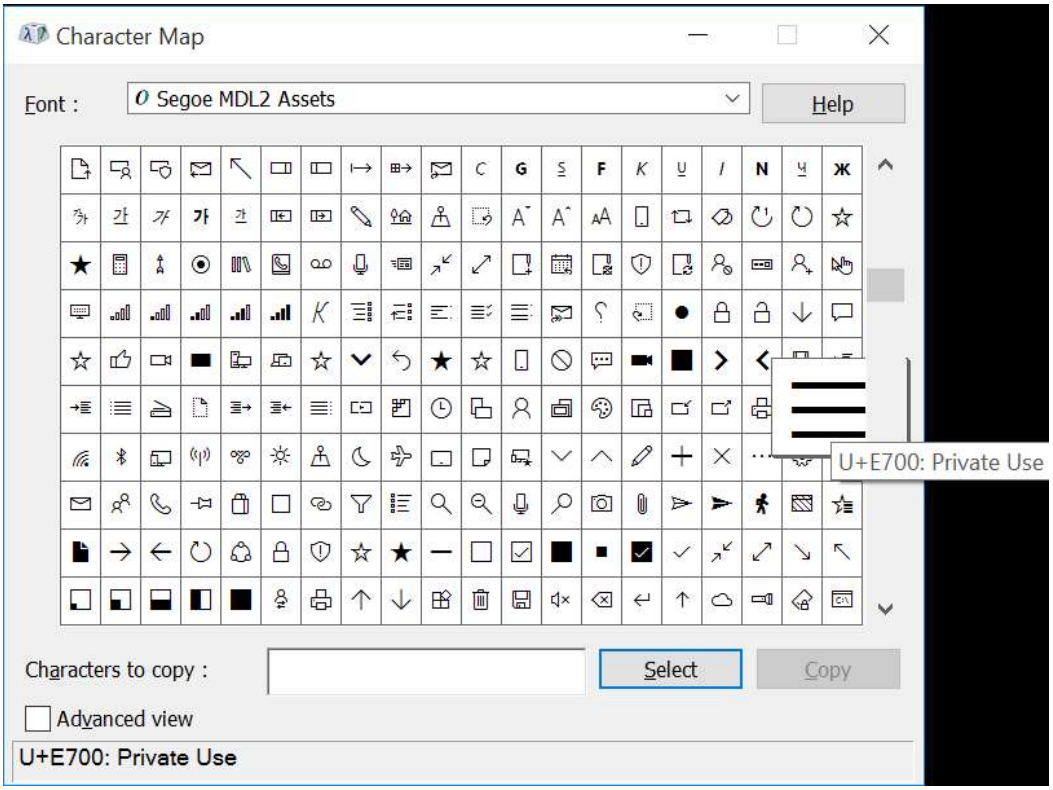
```
        <RelativePanel>
            <Button />
        </RelativePanel>
        <SplitView Grid.Row="1">
            <ListBox>
                <ListBoxItem />
                <ListBoxItem />
                <ListBoxItem />
            </ListBox>
        </SplitView>

```

That's our general layout. Now we just have to fill in the details.

The ListBox will sit inside of the Pane of the SplitView. Each ListBox item will have an icon on the left and then Text on the right. As you select an item from the ListBox, we will handle the SelectionChanged event and if this were a real app, we would build our functionality or display our cards in the main Content area.

I'll start with the Button control and name it HamburgerButton. Then I'll choose an icon that represents the hamburger. You'll recall that it has three vertical lines stacked above each other. To do that we're going to find that there are a specific FontFamily available on Windows 10 devices called Segoe MDL2 Assets.

I open the Character Map application the Cortana search bar or and searching for "Character Map". Once it's open I'll select the font: Segoe MDL2 Assets. In the lower area I will search for the hamburger symbol. You see a capital U+E700 and it's marked for private use. The only thing that's important to us is the "E700" part.



Back in the MainPage.xaml, I set the FontFamily to Segoe MDL2 Assets the Content="E700". I'll add the prefix:

&#x

And a semi-colon as a suffix. So the entire content is:

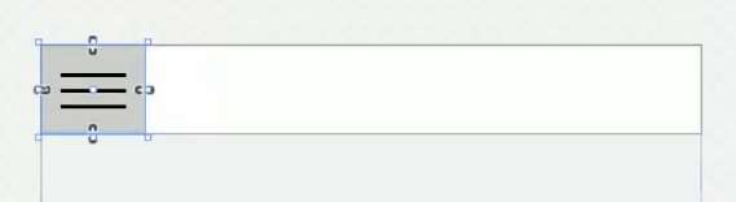
Content=""

I'm also going to set the FontSize to 36. That might be a bit large for a real application. Finally I'll add a Click event handler.

```
16          <RelativePanel>
17              <Button Name="HamburgerButton"
18                      FontFamily="Segoe MDL2 Assets"
19                      Content="&#xE700;"
20                      FontSize="36"
21                      Click="HamburgerButton_Click" />
22          </RelativePanel>
```

I think we're going to add the Click EventHandler for it to open and close it.

In the designer we can see we've got our button with the hamburger icon on it.



We could style up that button to be one of the primary colors for our apps, however I'm not going to do that in this particular example.

Next, I'll focus on the SplitView. I give it the name "MySplitView". I set DisplayMode="CompactOverlay" because I want to show the icons beneath it.

When the Pane is open, I'll set the width to 200. When it's closed set the CompactPanelLength="56". Also, I set the HorizontalAlignment="Left".

```
24          <SplitView Name="MySplitView"
25                     Grid.Row="1"
26                     DisplayMode="CompactOverlay"
27                     OpenPaneLength="200"
28                     CompactPaneLength="56"
29                     HorizontalAlignment="Left">
30
```

Next, I'll focus on the ListBox where I will display the ListBoxItems. I'll set the SelectionMode="Single" since I only want one item selected at a time. I'll set the Name="IconsListBox". Then I'll create a SelectionChanged event. I'll come back to that later.

Next, for each ListBoxItem I'll add a StackPanel and two TextBlocks inside of the StackPanel. The StackPanel will be oriented horizontally. The first TextBlock will contain the icon. The second TextBlock will contain the text.

Where will we get the icon from? Once again, we're going to use the FontFamily="Segoe MDL2 Assets" then set the FontSize="36". Finally, we're going to set the Text equal to and new icon from the Character Map. I'll choose E72D. So, I'll set the Content=""

On the second TextBlock I'll set the Text="Share" (just random text … don't read too much into that). I'll also give the ListBoxItem the name "ShareListBoxItem". I'll set the FontSize="24" and add some margin to the left.

I'll do the same thing for the other ListBoxItem. I'll copy and paste the first ListBoxItem, and change out the Text property to "" and "Favorites", respectively. I'll name this ListBoxItem "FavoritesListBoxItem".

```
31          <SplitView.Pane>
32              <ListBox SelectionMode="Single"
33                       Name="IconsListBox"
34                       SelectionChanged="IconsListBox_SelectionChanged">
35                  <ListBoxItem Name="ShareListBoxItem">
36                      <StackPanel Orientation="Horizontal">
37                          <TextBlock FontFamily="Segoe MDL2 Assets"
38                                     FontSize="36"
39                                     Text="&#xE72D;" />
40                          <TextBlock Text="Share"
41                                     FontSize="24"
42                                     Margin="20,0,0,0" />
43                      </StackPanel>
44                  </ListBoxItem>
45
46                  <ListBoxItem Name="FavoritesListBoxItem">
47                      <StackPanel Orientation="Horizontal">
48                          <TextBlock FontFamily="Segoe MDL2 Assets"
49                                     FontSize="36"
50                                     Text="&#xE734;" />
51                          <TextBlock Text="Favorites"
52                                     FontSize="24"
53                                     Margin="20,0,0,0" />
54                      </StackPanel>
55                  </ListBoxItem>
56
57              </ListBox>
58          </SplitView.Pane>
```

Now I'll back up to the HamburgerButton for a moment. Whenever someone clicks the button I will show or hide the SplitView's Pane. Using F12 on my keyboard I create an event handler method stub in the MainPage.xaml.cs code behind.

I write the following code:

```
30      private void HamburgerButton_Click(object sender, RoutedEventArgs e)
31      {
32        MySplitView.IsPaneOpen = !MySplitView.IsPaneOpen;
33      }
```

Nearing the end, I'll set the SplitView's Content. For our purposes I'm just going to put a TextBlock in the Content section, name it "ResultTextBlock".

To change the content of the ResultTextBlock I'll implement the IconsListBox_SelectionChanged event. I'll put my mouse cursor on the name and press F12. In the code behind, I'll add the following:

```
35      private void IconsListBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
36      {
37        if (ShareListBoxItem.IsSelected) { ResultTextBlock.Text = "Share"; }
38        else if (FavoritesListBoxItem.IsSelected) { ResultTextBlock.Text = "Favorites"; }
39      }
40    }
```

Obviously this is the simplest case possible. If this were a real app would probably want to implement navigation and load in another Page into a Frame and that Frame would sit in the middle where we currently have the Content area.

UWP-022 - Cheat Sheet Review: Windows 10 Layout Hamburger Navigation and Controls

UWP-017 - XAML Layout with the RelativePanel

It basically defines an area within which you can position and align child objects

-- in relation to each other
-- or in relation to the parent panel.

Controls use attached properties to position themselves.

1. Panel alignment relationships (AlignTopWithPanel, AlignLeftWithPanel, …) are applied first.
2. Sibling alignment relationships (AlignTopWith, AlignLeftWith, …) are applied second.
3. Sibling positional relationships (Above, Below, RightOf, LeftOf) are applied last.

```
<RelativePanel MinHeight="300" Grid.Row="1">
   <Rectangle Name="RedRectangle" RelativePanel.AlignRightWithPanel="True" />
   <Rectangle RelativePanel.LeftOf="RedRectangle" />
</RelativePanel>
```

UWP-018 - XAML Layout with the SplitPanel

The split view allows us to create a panel that can be displayed or hidden.

We would use the SplitView to implement hamburger navigation.

There are two parts to a SplitView:

1. The part that is hidden by default (Pane)
2. The part that is shown by default (Content)

You define other controls inside of the SplitView.Pane and SplitView.Content.

```
<SplitView Name="MySplitView"
   CompactPaneLength="50"
   IsPaneOpen="False"
   DisplayMode="CompactInline"
   OpenPaneLength="200" >
   <SplitView.Pane>
   </SplitView.Pane>
   <SplitView.Content>
   </SplitView.Content>
</SplitView>
```

DisplayMode Options:

Inline – Panel completely covers content. When expanded, panel pushes content.
CompactInline – Pane covers most of the Content. When expanded, panel pushes content.
Overlay – Panel completely covers content. When expanded, panel covers content.
CompactOverlay – Panel covers most of the content. When expanded, panel covers content.

Open / Close Pane in C#:

```
MySplitView.IsPaneOpen = !MySplitView.IsPaneOpen;
```

UWP_019 - Working with Navigation

Hierearchy:

App > Window > Frame > MainPage

You can load pages into a child frame or into the root frame:

```
Frame.Navigate(typeof(Page2), additionalParameter);
```

You can retrieve additionalParameter on the page you navigated to:

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
  value = (string)e.Parameter;
}
```

Traverse back stack (history):

```
if (Frame.CanGoBack) {
  Frame.GoBack();
}

if (Frame.CanGoForward) {
  Frame.GoForward();
}
```

Create a global variable by declaring a static internal field in the App class definition.

UWP-020 - Common XAML Controls - Part 1

```
<CheckBox Name="MyCheckBox" Content="Agree?" Tapped="MyCheckBox_Tapped" />

CheckBoxResultTextBlock.Text = MyCheckBox.IsChecked.ToString();
```

```
<RadioButton Name="YesRadioButton" Content="Yes" GroupName="MyGroup"
checked="RadioButton_Checked" />
<RadioButton Name="NoRadioButton" Content="No" GroupName="MyGroup"
Checked="RadioButton_Checked" />

RadioButtonTextBlock.Text = (bool)YesRadioButton.IsChecked ? "Yes" : "No";


<ComboBox SelectionChanged="ComboBox_SelectionChanged" >
   <ComboBoxItem Content="Fourth" />
   <ComboBoxItem Content="Fifth" />
   <ComboBoxItem Content="Sixth" IsSelected="True" />
</ComboBox>

if (ComboBoxResultTextBlock == null) return;
var combo = (ComboBox)sender;
var item = (ComboBoxItem)combo.SelectedItem;
ComboBoxResultTextBlock.Text = item.Content.ToString();


<ListBox Name="MyListBox" SelectionMode="Multiple"
SelectionChanged="ListBox_SelectionChanged">
   <ListBoxItem Content="First" />
   <ListBoxItem Content="Second" />
   <ListBoxItem Content="Third" />
</ListBox>

var selectedItems = MyListBox.Items.Cast<ListBoxItem>()
                .Where(p => p.IsSelected)
                 .Select(t => t.Content.ToString())
                  .ToArray();

ListBoxResultTextBlock.Text = string.Join(", ", selectedItems);


<Image Source="Assets/logo.png" Stretch="UniformToFill" />


<ToggleButton Name="MyToggleButton" Content="Premium Option" IsThreeState="True"
Click="MyToggleButton_Click" />

ToggleButtonResultTextBlock.Text = MyToggleButton.IsChecked.ToString();
```

```
<ToggleSwitch>
   <ToggleSwitch.OffContent>
      <TextBlock Text="I'm off right now." />
   </ToggleSwitch.OffContent>
   <ToggleSwitch.OnContent>
      <TextBlock Text="I'm on!" />
   </ToggleSwitch.OnContent>
</ToggleSwitch>
```

UWP-021 - Implementing a Simple Hamburger Navigation Menu

Jerry Nixon's Example: http://bit.do/hamburger-nav

Use Character Map to find the code to display icons using Segoe MDL2 Assets.

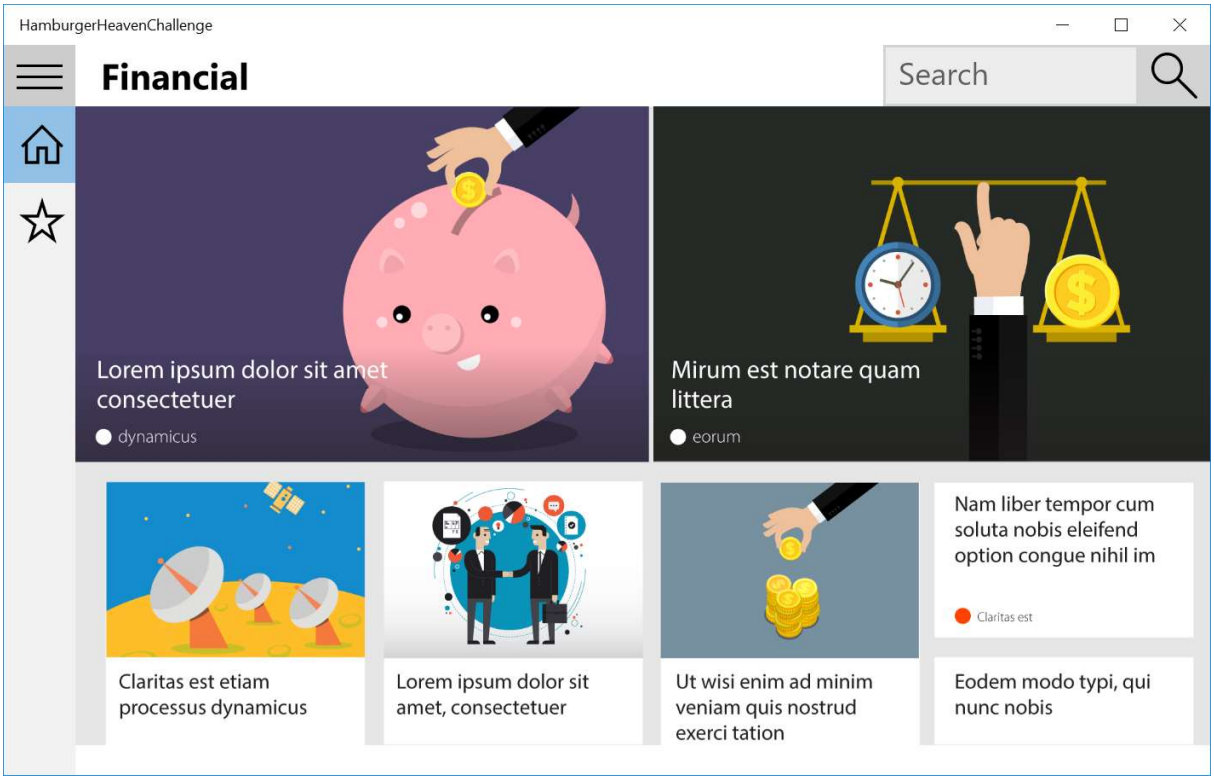
Hamburger: 

Use ListBox and ListBoxItems for the navigation links inside of a SplitView.

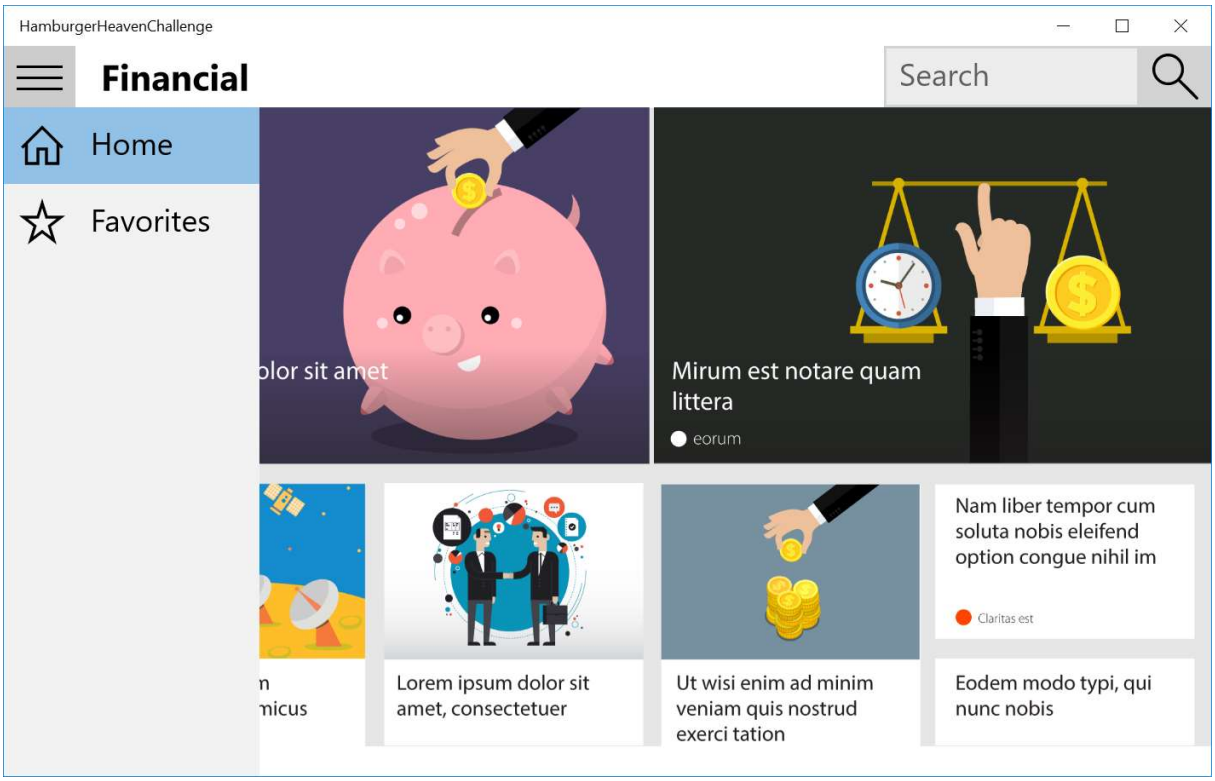
UWP-023 - Hamburger Heaven Challenge

The next challenge is called the “Hamburger Heaven Challenge”. This challenge will require you create a hamburger-style navigation for a fictitious Windows 10 application.

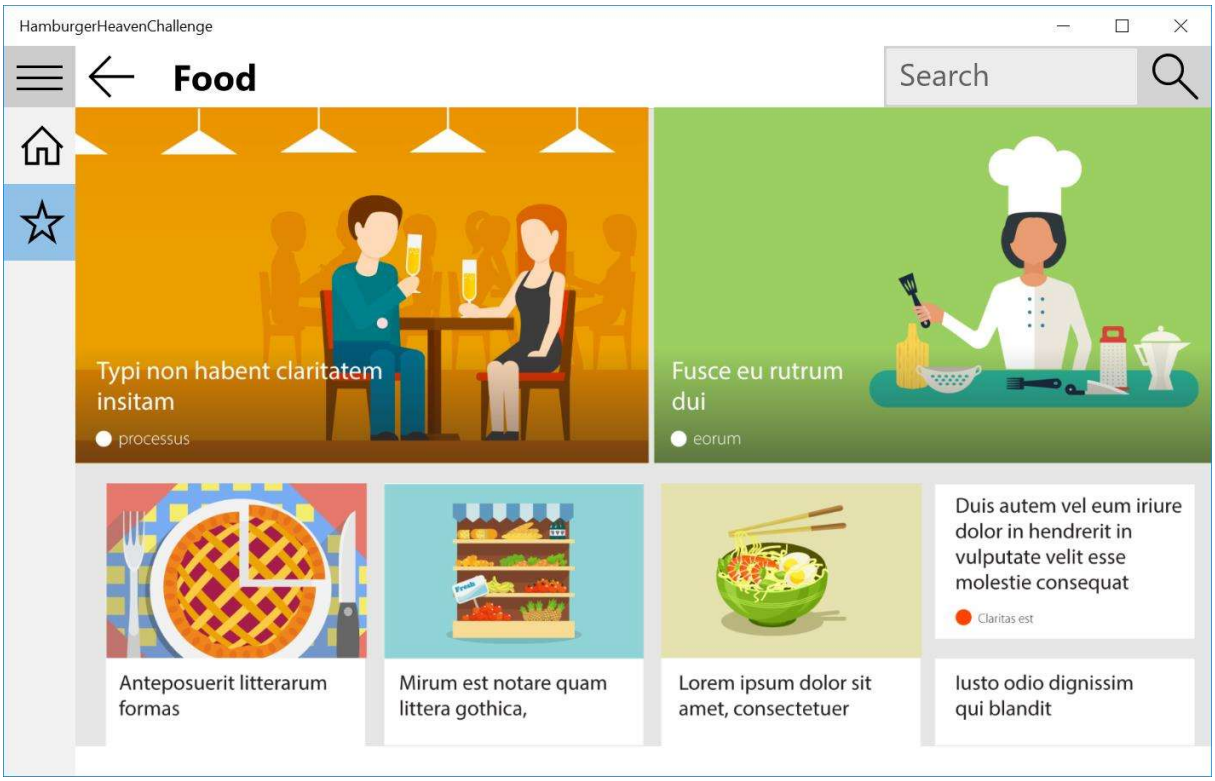
You will create two main navigation areas across the top. To the right, search area. To the left, hamburger button.



When I click it, the hamburger button SplitView’s Panel will be fully displayed



Furthermore I can select one of two primary sections of my application. When I go to the Financial page, you see that that's the Homepage for the application. When I change over to the Food page, notice that not only does the title change, but also I get this little icon that allows me to go back to the Financial page, the Homepage.



The colorful Content areas are a single image. All you'll need to do is display either the Financial.png or Food.png file that is available in the zipped folder associated with this lesson. You will host the images on Financial.xaml, and a Food.xaml, respectively. You'll use Pages in order to utilize the Frame's BackStack and the Navigation feature.

Also, in the zipped folder you will find the list of requirements in the text file named Instructions.txt. I'll discuss each of the items in that folder in the remainder of this lesson.

First, you'll create a folder named Assets and add the Food.png and the Financial.png to it.

Next, you'll add two Pages to the project named a Food.xaml and a Financial.xaml. And each of these pages will host a single Image control set to Food.png or Financial.png, respectively.

You'll add a Frame inside of this outermost MainPage.xaml

You'll implement the hamburger style navigation using a SplitView.

You'll display a Search bar in the upper right-hand corner and a hamburger button in the upper left-hand corner, and a back button next to that, and the title of the Page.

The Search area is for display only (it requires no C# code).

When the user navigates from Financial to Food the back arrow button should appear in the proper position on the Page. It should disappear once you return to the Financial page (which will be considered 'home').

In the Panel of the SplitView, choose any icon you want. Use the Character Map application to find an icon that you want to use.

The MainPage.xaml will host a Frame. When the application starts, the Financial.xaml page will be loaded into the Frame.

The search button should utilize the search icon. The back button must use a back arrow icon.

Try to figure out how to set the placeholder text “Search” in the TextBox. Note: I did not discuss this in the lesson material, however there is a special property that allows you to set that word "Search" in the TextBox but as soon as the user beings typing, that word will go away.

Font sizes that I used include 24, 28, and 36.

The colors that I use will include LightGrays, alright, and maybe a MediumGray.

Use these images as your guide for margins, font sizes and colors.

UWP-024 - Hamburger Heaven Challenge: Solution

Step 1: Create a new Project using the “Blank App, Universal Windows” template named “HamburgerHeavenChallenge”

Step 2: Add an Assets folder to the project, drag and drop the Food.png and Financial.png to the Assets folder.

Step 3: Add two Pages to your app, Food.xaml and Financial.xaml. And each of these will host a single Image control. We will set the source to Food.png and Financial.png.

Go to the Project menu, Add New Item, make sure that I add a Blank Page. Name the first one Financial.xaml and the second one Food.xaml.

Add an Image control, set the Source="Assets/Food.png" and the VerticalAlignment="Top" just to make sure that it butts itself up against the top.

Do the same for Financial.xaml setting the Source="Assets/Financial.png"

Step 4: In the App.xaml.cs remove the Frame counter.

Step 5: In MainPage.xaml create two Rows. The first one height should be Auto, the second height should be "*".

Step 6: Add a RelativePanel. Inside the RelativePanel:

First, add a Button named HamburgerButton.

Second, add a Button named BackButton.

Third, add a TextBlock named TitleTextBlock.

Fourth, add a Button named SearchButton.

Fifth, add a TextBox named SearchTextBox.

Sixth, set the HamburgerButton’s RelativePanel.AlignLeftWithPanel="True". Set the FontFamily=”Segoe MDL2 Assets”, the FontSize="36", Content="". Finally, add an event handler for Click named HamburgerButton_Click.

Seventh, set the BackButton’s RelativePanel.RightOf="HamburgerButton". Set the FontFamily=”Segoe MDL2 Assets”. Set the FontSize="36". Set the Content="". Add a click event handler.

Eighth, set the TitleTextBlock’s RelativePanel.RightOf="BackButton". Set its FontSize="28". Set the FontWeight="Bold".

Ninth, set the SearchButton’s to RelativePanel.AlignRightWithPanel="True". Set the FontFamily="Segoe MDL2 Assets". Set the FontSize="36". Set the Content=""

Tenth, set the SearchTextBox’s RelativePanel.LeftOf=”SearchButton”. Set its Height="48". Set the Width=”100”. Set the FontSize="24". Set the PlaceholderText="Search".

```
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>

        <RelativePanel>
            <Button Name="HamburgerButton"
                    RelativePanel.AlignLeftWithPanel="True"
                    FontFamily="Segoe MDL2 Assets"
                    FontSize="36"
                    Content="&#xE700;"
                    Click="HamburgerButton_Click" />

            <Button Name="BackButton"
                    RelativePanel.RightOf="HamburgerButton"
                    FontFamily="Segoe MDl2 Assets"
                    FontSize="36"
                    Content="&#xE0C4;"
                    Click="BackButton_Click"
                     />
```

```
32              <TextBlock Name="TitleTextBlock"
33                         RelativePanel.RightOf="BackButton"
34                         FontSize="28"
35                         FontWeight="Bold"
36                         Margin="20,5,0,0"/>
37
38              <Button Name="SearchButton"
39                      RelativePanel.AlignRightWithPanel="True"
40                      FontFamily="Segoe MDL2 Assets"
41                      FontSize="36"
42                      Content="&#xE1A3;" />
43
44              <TextBox Name="SearchTextBox"
45                       RelativePanel.LeftOf="SearchButton"
46                       Height="48"
47                       Width="200"
48                       FontSize="24"
49                       PlaceholderText="Search" />
50          </RelativePanel>
51
52      </Grid>
53  </Page>
```

Step 7: Add a SplitView control and name it “MySplitView “. Set the SplitView’s Grid.Row="1". Inside of the SplitView, create a SplitView.Pane, and then a SplitView.Content.

Set DisplayMode=”CompactOverlay”. Set OpenPanelLength="200". Set the CompactPaneLength="56"

Inside of SplitView.Content add a Frame named “MyFrame”.

```
52          <SplitView Grid.Row="1"
53                     Name="MySplitView"
54                     DisplayMode="CompactOverlay"
55                     OpenPaneLength="200"
56                     CompactPaneLength="56">
57              <SplitView.Pane>
58
59              </SplitView.Pane>
60              <SplitView.Content>
61                  <Frame Name="MyFrame"></Frame>
62              </SplitView.Content>
63
64          </SplitView>
```

Step 8: Inside of the SplitView's Pane add a ListBox with two ListBoxItems. Set the ListBox's SelectionMode="Single".

Step 9: In the first ListBoxItem, add a StackPanel and set the Orientation ="Horizontal". This will allow us to add two TextBlocks next to each other … the first for the icon, the second for the text.

In the first TextBlock, set the FontFamily="Segoe MDL2 Assets", and set the FontSize="36". I set the Content"="

In the second TextBlock set the Text="Financial". Set the FontSize="24".

I copy the entire ListBoxItem and paste it below the first ListBoxItem. I'll change the Content properties for the two TextBlocks. In the first TextBlock set the Content="" and in the second TextBlock set the Content="Food".

```
57          <SplitView.Pane>
58              <ListBox SelectionMode="Single"
59                       SelectionChanged="ListBox_SelectionChanged">
60                  <ListBoxItem Name="Financial">
61                      <StackPanel Orientation="Horizontal">
62                          <TextBlock FontFamily="Segoe MDL2 Assets"
63                                     FontSize="36"
64                                     Text="&#xE80F;" />
65                          <TextBlock FontSize="24"
66                                     Margin="20,0,0,0">Financial</TextBlock>
67                      </StackPanel>
68                  </ListBoxItem>
69                  <ListBoxItem Name="Food">
70                      <StackPanel Orientation="Horizontal">
71                          <TextBlock FontFamily="Segoe MDL2 Assets"
72                                     FontSize="36"
73                                     Text="&#xE1CE;" />
74                          <TextBlock FontSize="24"
75                                     Margin="20,0,0,0">Food</TextBlock>
76                      </StackPanel>
77                  </ListBoxItem>
78              </ListBox>
79          </SplitView.Pane>
```

Step 9: Next we'll implement the click event of the HamburgerButton to open and close the MySplitView's Pane. I put my mouse cursor inside of the event handler name and press hit F12 on my keyboard. Add the following code to the method stub:

```
33      private void HamburgerButton_Click(object sender, RoutedEventArgs e)
34      {
35        MySplitView.IsPaneOpen = !MySplitView.IsPaneOpen;
36      }
```

Step 10: Next we'll navigate directly to the Financial page when the app starts. We'll also set the TitleTextBlock to indicate the page we're currently on.

```
25      public MainPage()
26      {
27        this.InitializeComponent();
28
29        MyFrame.Navigate(typeof(Financial));
30        TitleTextBlock.Text = "Financial";
31       }
```

Step 11: Next we'll implement the ListBox_SelectionChanged event to allow the user to select one of the two items in our SplitView's Pane:

```
47      private void ListBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
48      {
49        if (Financial.IsSelected)
50        {
51        BackButton.Visibility = Visibility.Collapsed;
52          MyFrame.Navigate(typeof(Financial));
53          TitleTextBlock.Text = "Financial";
54        }
55        else if (Food.IsSelected)
56        {
57          BackButton.Visibility = Visibility.Visible;
58          MyFrame.Navigate(typeof(Food));
59          TitleTextBlock.Text = "Food";
60        }
61      }
```

Notice in the code that we're not only navigating to the proper page, but also changing the TitleTextBlock and the BackButton's visibility depending on the selection.

Step 12: Next I'll implement the BackButton's functionality. We'll check whether the MyFrame can go back and if it can, then not only should it navigate, but also select the Financial ListBoxItem:

```
38      private void BackButton_Click(object sender, RoutedEventArgs e)
39      {
40        if (MyFrame.CanGoBack)
41        {
42          MyFrame.GoBack();
43          Financial.IsSelected = true;
44        }
45      }
```

Step 13: Finally, I'll revisit MainPage() to ensure that the BackButton's visibility is collapsed when the app first opens. Furthermore, since we're navigating to the Financial page, we want to make sure the ListBoxItem is selected.

```
25      public MainPage()
26      {
27          this.InitializeComponent();
28
29          MyFrame.Navigate(typeof(Financial));
30          TitleTextBlock.Text = "Financial";
31
32          BackButton.Visibility = Visibility.Collapsed;
33          Financial.IsSelected = true;
34      }
35
```