70-357.exam.27q

Number: 70-357
Passing Score: 800
Time Limit: 120 min

70-357

Developing Mobile Apps

Exam A

QUESTION 1
Case Study

This is a case study. Case studies are not limited separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.
To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.
At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.
To start the case study
To display the first question on this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Background

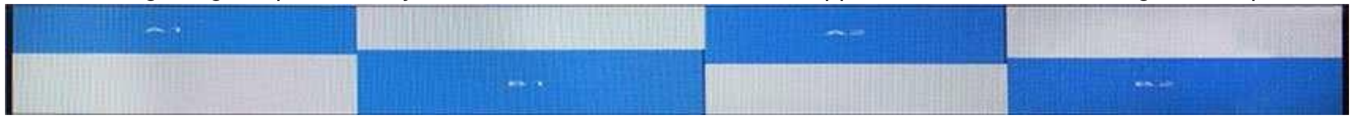
You are developing an application named Timeline that presents information on a timeline. The app allows users to create items and enter details about the item. The app displays item names on a timeline. When users select an item name on the timeline, the app displays additional content about the item.

Business requirements

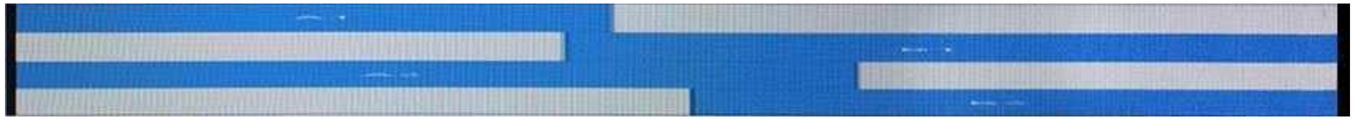
Timeline section

The timeline element of the app has the following layout requirements:
- The timeline must adapt to the screen size and orientation of the device.
- The timeline size must dynamically change if the window containing the content is resized by the user.
- The user must be able to scroll through the timeline horizontally when the device is in landscape mode.
- The user must be able to scroll through the timeline vertically when the device is in portrait mode.
- The timeline must begin scrolling as soon as a scroll is detected. Scrolling must continue for a short distance after the scroll input has stopped. ▪ Scroll bars or panning controls must always be visible.

The following image depicts the layout for the timeline section of the app when the device is using landscape orientation:



The following image depicts the layout for the timeline section of the app when the device is using portrait orientation:

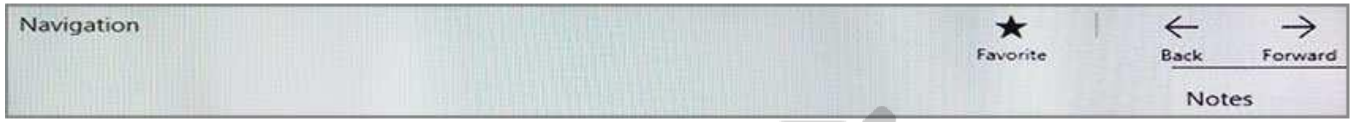


Content section

The content element of the app has the following layout requirements:
▪ When a user selects an item on the timeline, the details for that item must display beneath or to the right of the timeline.
▪ The content section must display one page of information. The element must be a child of the selected item in the timeline. ▪ Users must be able to return to a previously selected event by pressing the Back button.

User interface

The user must be able to navigate the application using the interface below:



▪ The Favorite button marks the current content to be displayed in a Favorites panel.
▪ The Back and Forward buttons navigate through the app selection history. Both buttons must be available on all devices. ▪ The Notes button allows the user to manage notes about the current content.
▪ The app must support touch, mouse, and stylus input.
▪ The app layout must automatically adapt to the screen size and orientation.

Technical requirements

Layout

You identify the following layout requirements:

General
▪ All user interface (UI) elements must continuously scale when a user resizes the window.
▪ UI controls must be smaller and spaced closer together if there is a mouse or stylus available.
▪ UI controls must be larger and spaced farther apart if the device supports touch and there is no mouse or pointer available.

Timeline

▪ The timeline must be displayed in a horizontal layout when the device is in a landscape orientation or when the horizontal width is greater than the vertical height.
▪ The timeline must be displayed in a vertical layout when the device is in a portrait orientation or when the vertical height is greater than the horizontal width.
▪ Each item in the past must be linked to the next item in the future.
▪ Users must be able to scroll from past events to future events or from future events to past events. ▪
The app must only allow one level of detail to be linked to each item in the timeline.

Optimization

You must optimize the app using the following guidelines:
▪ You must minimize the time it takes to display content when an item on the timeline is selected. ▪
The app must respect memory and resource constraints for all devices.

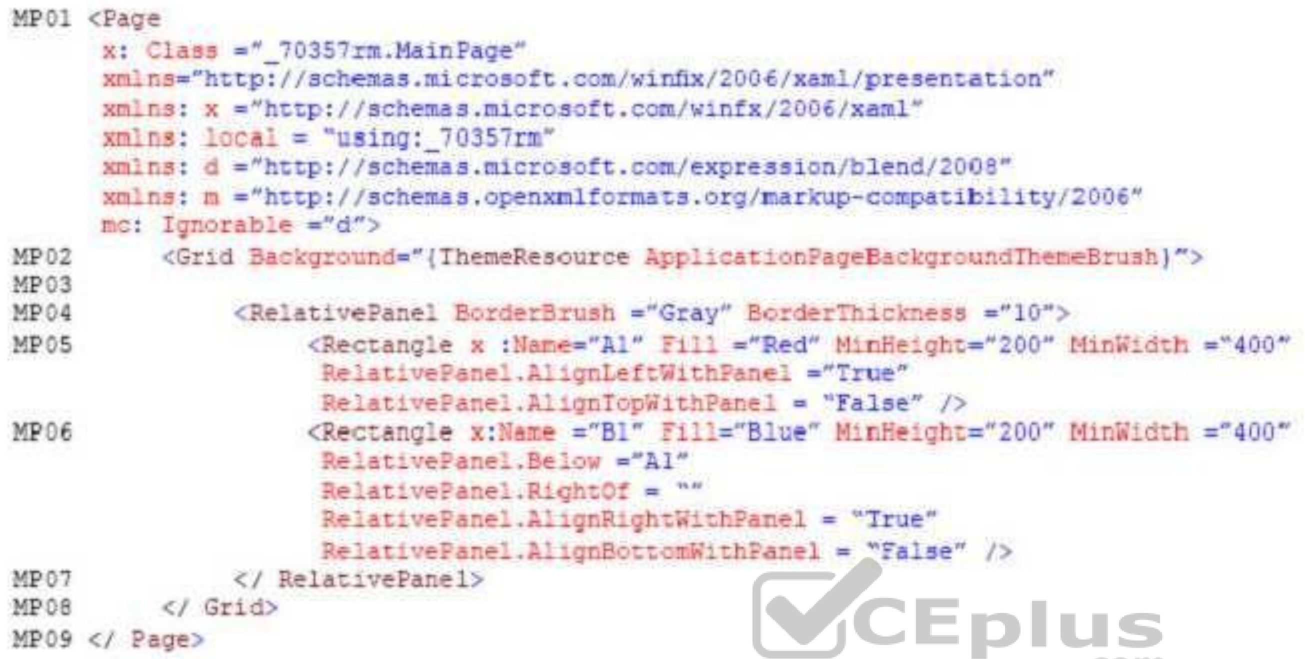
XAML coding style

All code and markup must conform to the following style guidelines:
▪ Use resource dictionaries for styles that are used more than once.
▪ Limit the use of nested panels.
▪ Use built-in properties of existing panels instead of using separate style objects.
▪ Use the navigation structure that best models the data without exceeding the requirements of the app. Application

structure

MainPage.xaml

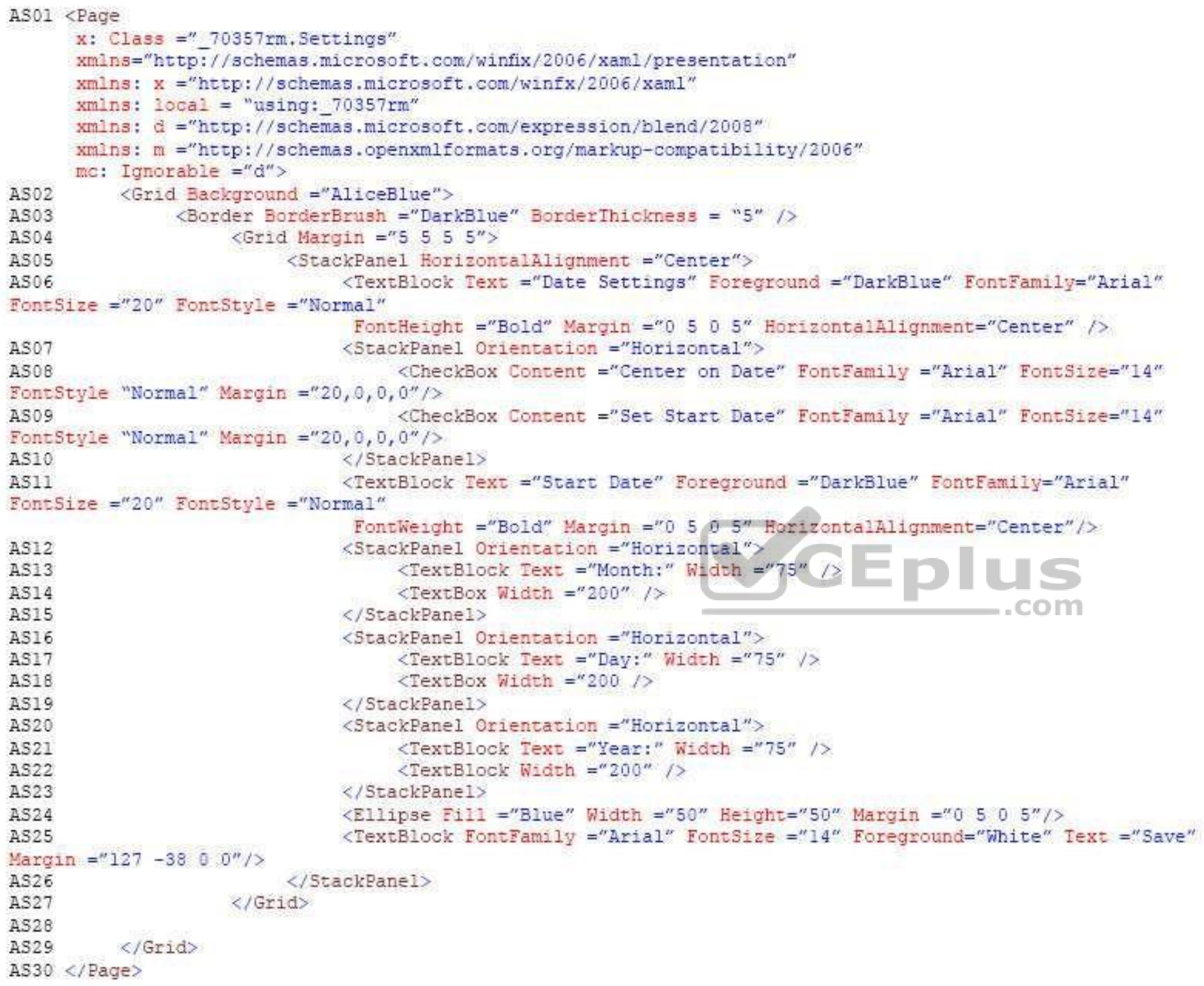
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



```
MP01 <Page
       x: Class ="_70357rm.MainPage"
       xmlns="http://schemas.microsoft.com/winfix/2006/xaml/presentation"
       xmlns: x ="http://schemas.microsoft.com/winfx/2006/xaml"
       xmlns: local = "using:_70357rm"
       xmlns: d ="http://schemas.microsoft.com/expression/blend/2008"
       xmlns: m ="http://schemas.openxmlformats.org/markup-compatibility/2006"
       mc: Ignorable ="d">
MP02       <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
MP03
MP04            <RelativePanel BorderBrush ="Gray" BorderThickness ="10">
MP05                 <Rectangle x :Name="A1" Fill ="Red" MinHeight="200" MinWidth ="400"
                      RelativePanel.AlignLeftWithPanel ="True"
                      RelativePanel.AlignTopWithPanel = "False" />
MP06                 <Rectangle x:Name ="B1" Fill="Blue" MinHeight="200" MinWidth ="400"
                      RelativePanel.Below ="A1"
                      RelativePanel.RightOf = ""
                      RelativePanel.AlignRightWithPanel = "True"
                      RelativePanel.AlignBottomWithPanel = "False" />
MP07            </ RelativePanel>
MP08       </ Grid>
MP09 </ Page>
```

Settings.xaml

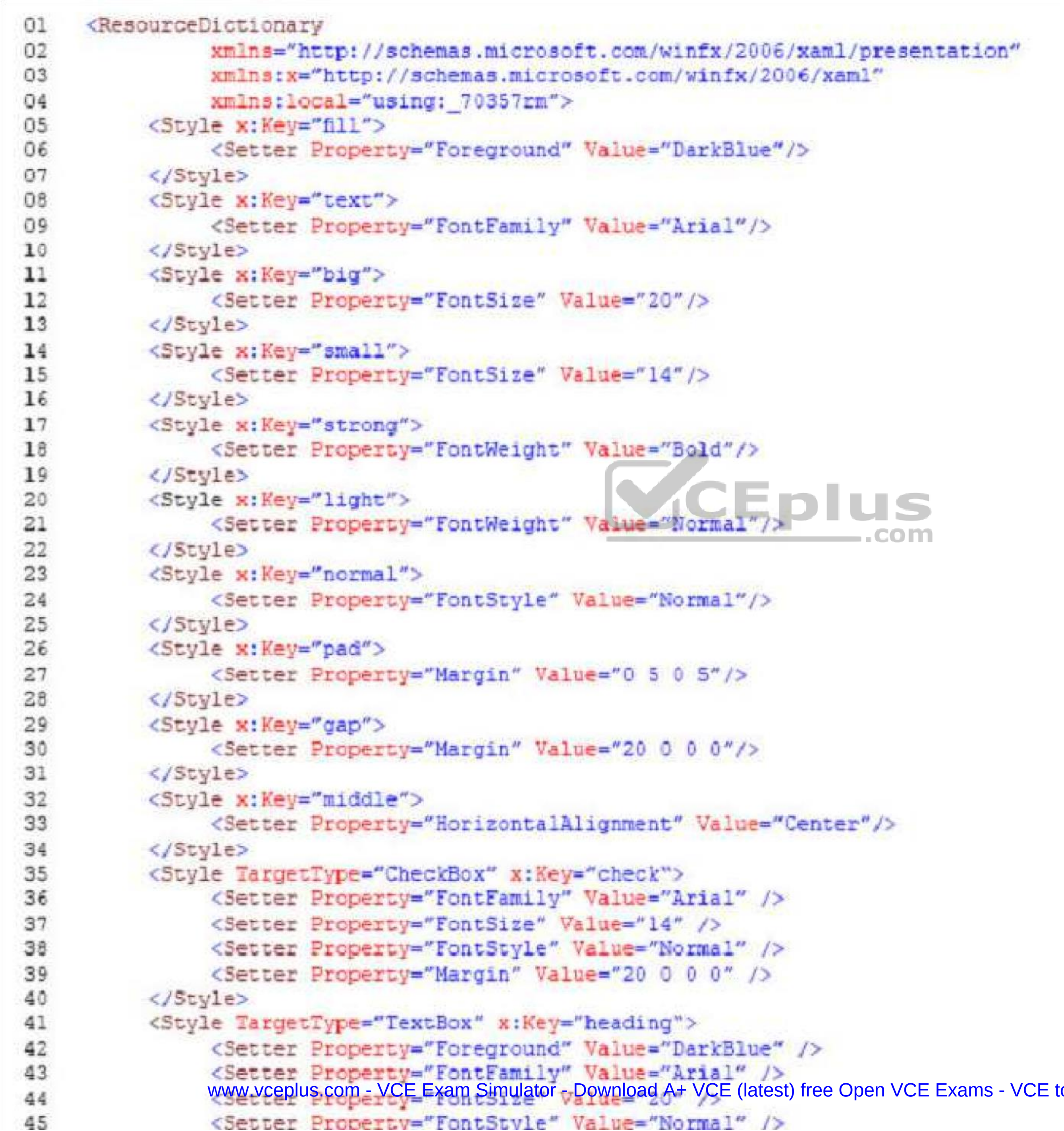
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



```
AS01 <Page
      x: Class ="_70357rm.Settings"
      xmlns="http://schemas.microsoft.com/winfix/2006/xaml/presentation"
      xmlns: x ="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns: local = "using:_70357rm"
      xmlns: d ="http://schemas.microsoft.com/expression/blend/2008"
      xmlns: m ="http://schemas.openxmlformats.org/markup-compatibility/2006"
      mc: Ignorable ="d">
AS02      <Grid Background ="AliceBlue">
AS03           <Border BorderBrush ="DarkBlue" BorderThickness = "5" />
AS04                <Grid Margin ="5 5 5 5">
AS05                     <StackPanel HorizontalAlignment ="Center">
AS06                          <TextBlock Text ="Date Settings" Foreground ="DarkBlue" FontFamily="Arial"
FontSize ="20" FontStyle ="Normal"
                               FontHeight ="Bold" Margin ="0 5 0 5" HorizontalAlignment="Center" />
AS07                          <StackPanel Orientation ="Horizontal">
AS08                               <CheckBox Content ="Center on Date" FontFamily ="Arial" FontSize="14"
FontStyle "Normal" Margin ="20,0,0,0"/>
AS09                               <CheckBox Content ="Set Start Date" FontFamily ="Arial" FontSize="14"
FontStyle "Normal" Margin ="20,0,0,0"/>
AS10                          </StackPanel>
AS11                          <TextBlock Text ="Start Date" Foreground ="DarkBlue" FontFamily="Arial"
FontSize ="20" FontStyle ="Normal"
                               FontWeight ="Bold" Margin ="0 5 0 5" HorizontalAlignment="Center"/>
AS12                          <StackPanel Orientation ="Horizontal">
AS13                               <TextBlock Text ="Month:" Width ="75" />
AS14                               <TextBox Width ="200" />
AS15                          </StackPanel>
AS16                          <StackPanel Orientation ="Horizontal">
AS17                               <TextBlock Text ="Day:" Width ="75" />
AS18                               <TextBox Width ="200 />
AS19                          </StackPanel>
AS20                          <StackPanel Orientation ="Horizontal">
AS21                               <TextBlock Text ="Year:" Width ="75" />
AS22                               <TextBlock Width ="200" />
AS23                          </StackPanel>
AS24                          <Ellipse Fill ="Blue" Width ="50" Height="50" Margin ="0 5 0 5"/>
AS25                          <TextBlock FontFamily ="Arial" FontSize ="14" Foreground="White" Text ="Save"
Margin ="127 -38 0 0"/>
AS26                     </StackPanel>
AS27                </Grid>
AS28
AS29      </Grid>
AS30 </Page>
```

ResourceDictionery.xaml

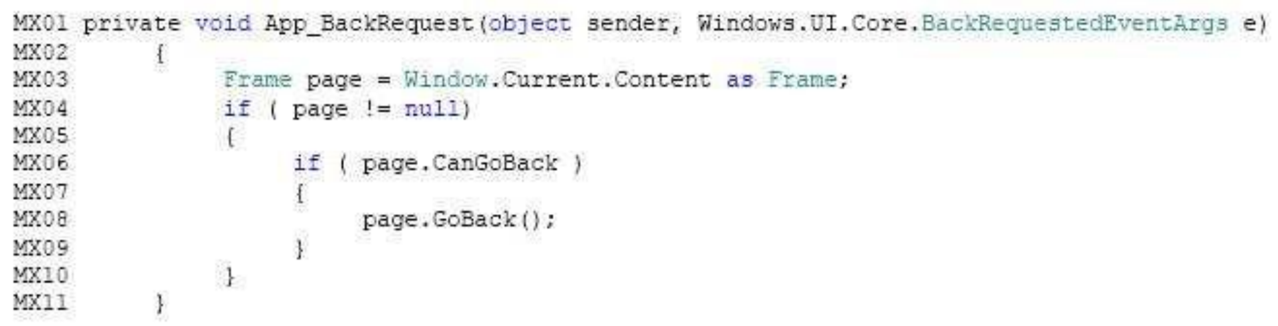
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only.)



```
<ResourceDictionary
          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:local="using:_70357rm">
     <Style x:Key="fill">
          <Setter Property="Foreground" Value="DarkBlue"/>
     </Style>
     <Style x:Key="text">
          <Setter Property="FontFamily" Value="Arial"/>
     </Style>
     <Style x:Key="big">
          <Setter Property="FontSize" Value="20"/>
     </Style>
     <Style x:Key="small">
          <Setter Property="FontSize" Value="14"/>
     </Style>
     <Style x:Key="strong">
          <Setter Property="FontWeight" Value="Bold"/>
     </Style>
     <Style x:Key="light">
          <Setter Property="FontWeight" Value="Normal"/>
     </Style>
     <Style x:Key="normal">
          <Setter Property="FontStyle" Value="Normal"/>
     </Style>
     <Style x:Key="pad">
          <Setter Property="Margin" Value="0 5 0 5"/>
     </Style>
     <Style x:Key="gap">
          <Setter Property="Margin" Value="20 0 0 0"/>
     </Style>
     <Style x:Key="middle">
          <Setter Property="HorizontalAlignment" Value="Center"/>
     </Style>
     <Style TargetType="CheckBox" x:Key="check">
          <Setter Property="FontFamily" Value="Arial" />
          <Setter Property="FontSize" Value="14" />
          <Setter Property="FontStyle" Value="Normal" />
          <Setter Property="Margin" Value="20 0 0 0" />
     </Style>
     <Style TargetType="TextBox" x:Key="heading">
          <Setter Property="Foreground" Value="DarkBlue" />
          <Setter Property="FontFamily" Value="Arial" />
          <Setter Property="FontSize" Value="20" />
          <Setter Property="FontStyle" Value="Normal" />
```

MainPage.xaml.cs

Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



```
MX01 private void App_BackRequest(object sender, Windows.UI.Core.BackRequestedEventArgs e)
MX02       {
MX03             Frame page = Window.Current.Content as Frame;
MX04             if ( page != null)
MX05             {
MX06                   if ( page.CanGoBack )
MX07                   {
MX08                         page.GoBack();
MX09                   }
MX10             }
MX11       }
```

You need to design the navigation for the timeline.

What navigation should you use?

A. hierarchy
B. peer
C. hub
D. master/details

Correct Answer: A
Section: (none)
Explanation

Explanation/Reference:
Explanation:
From scenario:
▪ Each item in the past must be linked to the next item in the future.
▪ Users must be able to scroll from past events to future events or from future events to past events. ▪
The app must only allow one level of detail to be linked to each item in the timeline.

Here we can use a hierarchy with each parent node having only one single child node.

Hierarchical structures are good for organizing complex content that spans lots of pages or when pages should be viewed in a particular order. The downside is that hierarchical pages introduce some navigation overhead: the deeper the structure, the more clicks it takes for users to get from page to page.
We recommend a hiearchical structure when:
You expect the user to traverse the pages in a specific order. Arrange the hierarchy to enforce that order.
There is a clear parent-child relationship between one of the pages and the other pages in the group.
There are more than 7 pages in the group.
When there are more than 7 pages in the group, it might be difficult for users to understand how the pages are unique or to understand their current location within the group. If you don't think that's an issue for your app, go ahead and make the pages peers

Reference: https://docs.microsoft.com/en-us/windows/uwp/layout/navigation-basics

QUESTION 2
Case Study
This is a case study. Case studies are not limited separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.
To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.
At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.
To start the case study
To display the first question on this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

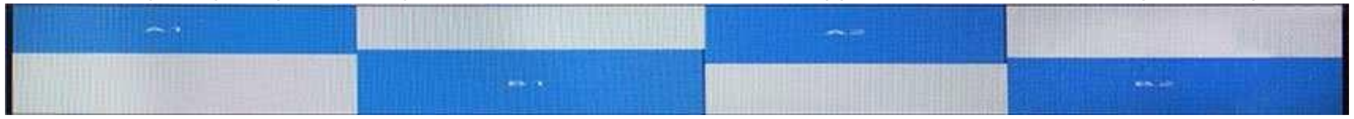
Background

You are developing an application named Timeline that presents information on a timeline. The app allows users to create items and enter details about the item. The app displays item names on a timeline. When users select an item name on the timeline, the app displays additional content about the item.

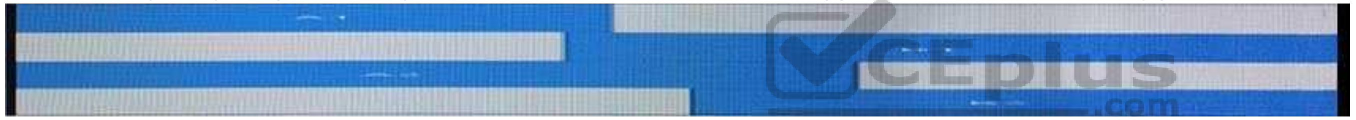
Business requirements
Timeline section

The timeline element of the app has the following layout requirements:
▪ The timeline must adapt to the screen size and orientation of the device.
▪ The timeline size must dynamically change if the window containing the content is resized by the user.
▪ The user must be able to scroll through the timeline horizontally when the device is in landscape mode.
▪ The user must be able to scroll through the timeline vertically when the device is in portrait mode.
▪ The timeline must begin scrolling as soon as a scroll is detected. Scrolling must continue for a short distance after the scroll input has stopped. ▪
Scroll bars or panning controls must always be visible.

The following image depicts the layout for the timeline section of the app when the device is using landscape orientation:



The following image depicts the layout for the timeline section of the app when the device is using portrait orientation:

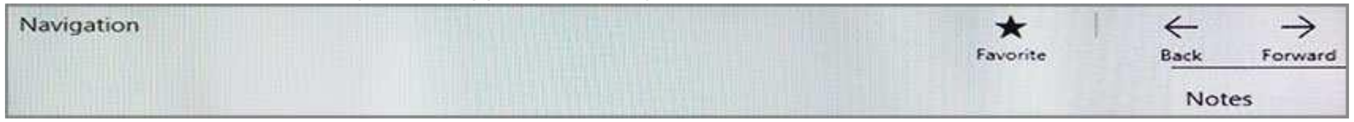


Content section

The content element of the app has the following layout requirements:
▪ When a user selects an item on the timeline, the details for that item must display beneath or to the right of the timeline.
▪ The content section must display one page of information. The element must be a child of the selected item in the timeline. ▪
Users must be able to return to a previously selected event by pressing the Back button.

User interface

The user must be able to navigate the application using the interface below:



▪ The Favorite button marks the current content to be displayed in a Favorites panel.
▪ The Back and Forward buttons navigate through the app selection history. Both buttons must be available on all devices.
▪ The Notes button allows the user to manage notes about the current content.

▪ The app must support touch, mouse, and stylus input.
▪ The app layout must automatically adapt to the screen size and orientation.

Technical requirements

Layout

You identify the following layout requirements:

General
▪ All user interface (UI) elements must continuously scale when a user resizes the window.
▪ UI controls must be smaller and spaced closer together if there is a mouse or stylus available.
▪ UI controls must be larger and spaced farther apart if the device supports touch and there is no mouse or pointer available.

Timeline

▪ The timeline must be displayed in a horizontal layout when the device is in a landscape orientation or when the horizontal width is greater than the vertical height. ▪ The timeline must be displayed in a vertical layout when the device is in a portrait orientation or when the vertical height is greater than the horizontal width. ▪ Each item in the past must be linked to the next item in the future.
▪ Users must be able to scroll from past events to future events or from future events to past events. ▪
The app must only allow one level of detail to be linked to each item in the timeline.

Optimization

You must optimize the app using the following guidelines:
▪ You must minimize the time it takes to display content when an item on the timeline is selected. ▪
The app must respect memory and resource constraints for all devices.

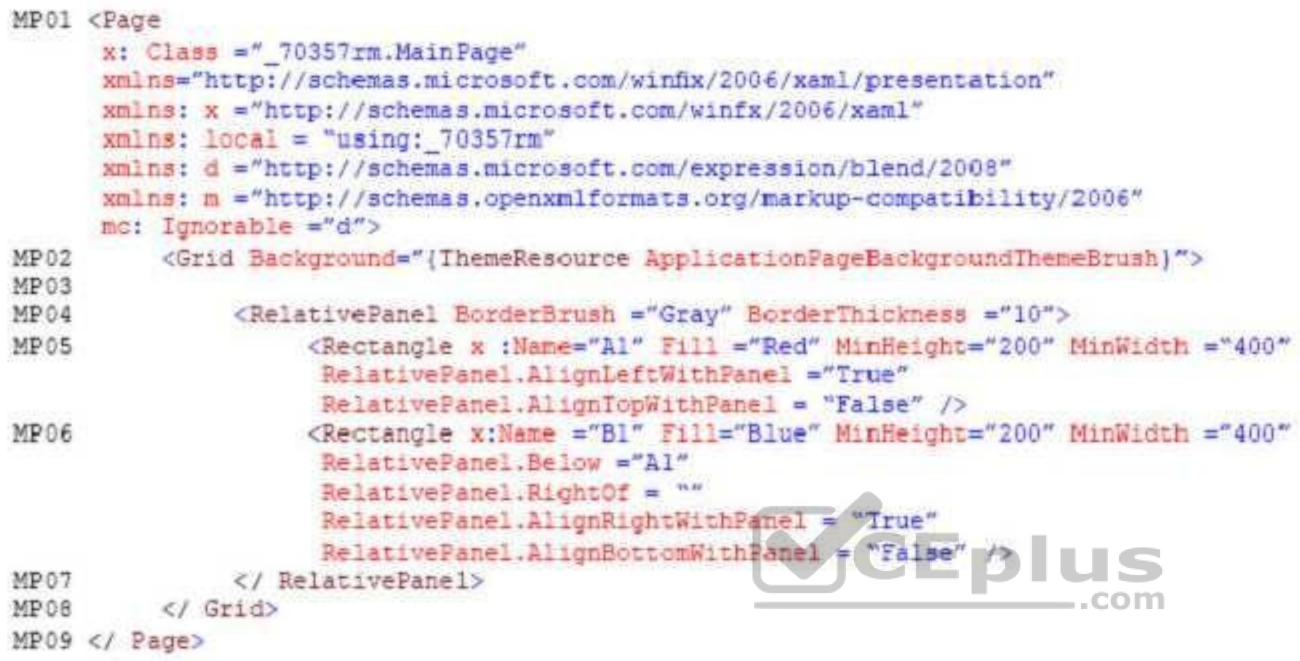
XAML coding style

All code and markup must conform to the following style guidelines:
▪ Use resource dictionaries for styles that are used more than once.
▪ Limit the use of nested panels.
▪ Use built-in properties of existing panels instead of using separate style objects.
▪ Use the navigation structure that best models the data without exceeding the requirements of the app. Application

structure

MainPage.xaml

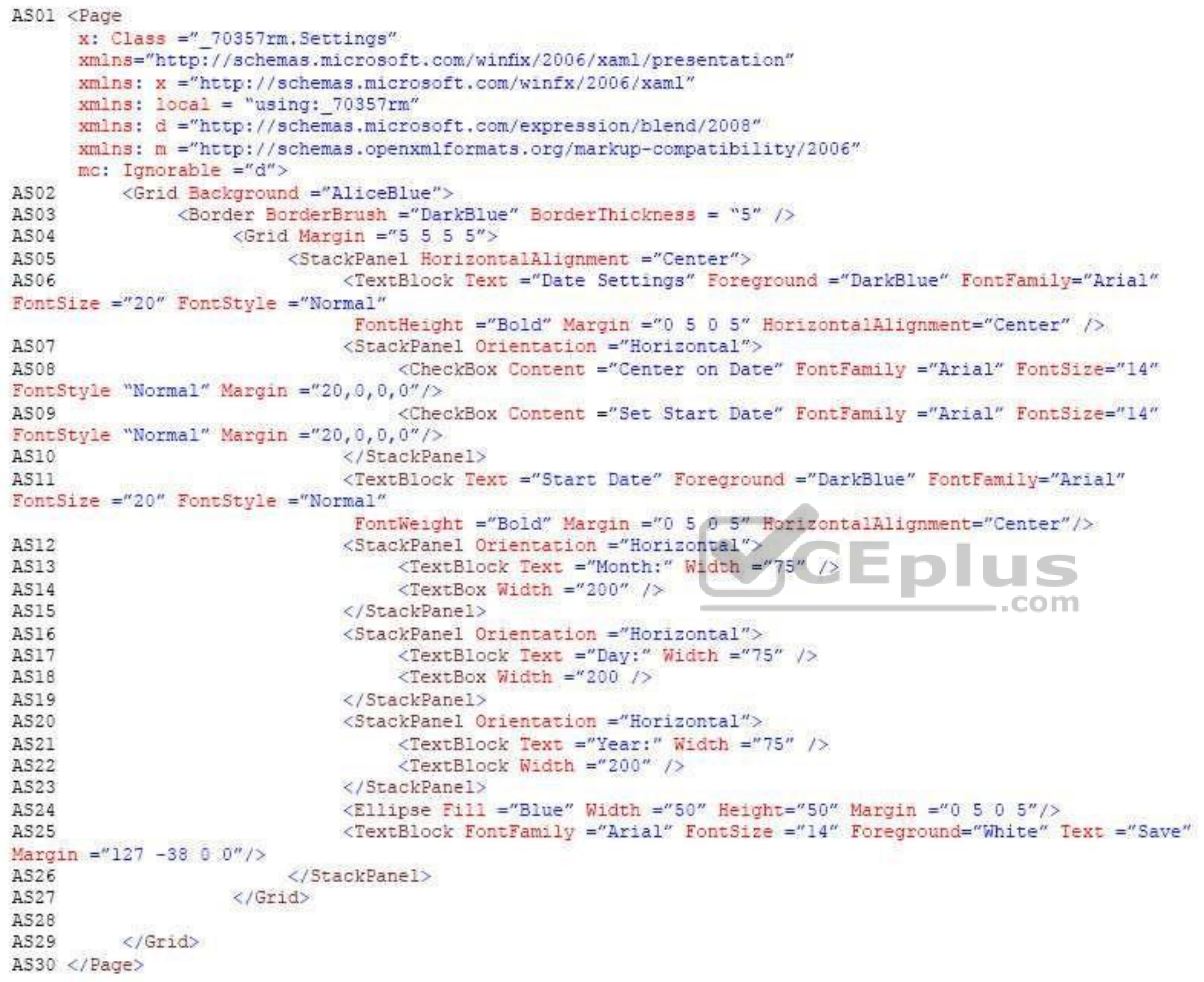
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



```
MP01 <Page
       x: Class ="_70357rm.MainPage"
       xmlns="http://schemas.microsoft.com/winfix/2006/xaml/presentation"
       xmlns: x ="http://schemas.microsoft.com/winfx/2006/xaml"
       xmlns: local = "using:_70357rm"
       xmlns: d ="http://schemas.microsoft.com/expression/blend/2008"
       xmlns: m ="http://schemas.openxmlformats.org/markup-compatibility/2006"
       mc: Ignorable ="d">
MP02      <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
MP03
MP04            <RelativePanel BorderBrush ="Gray" BorderThickness ="10">
MP05                 <Rectangle x :Name="A1" Fill ="Red" MinHeight="200" MinWidth ="400"
                      RelativePanel.AlignLeftWithPanel ="True"
                      RelativePanel.AlignTopWithPanel = "False" />
MP06                 <Rectangle x:Name ="B1" Fill="Blue" MinHeight="200" MinWidth ="400"
                      RelativePanel.Below ="A1"
                      RelativePanel.RightOf = ""
                      RelativePanel.AlignRightWithPanel = "True"
                      RelativePanel.AlignBottomWithPanel = "False" />
MP07            </ RelativePanel>
MP08      </ Grid>
MP09 </ Page>
```

Settings.xaml

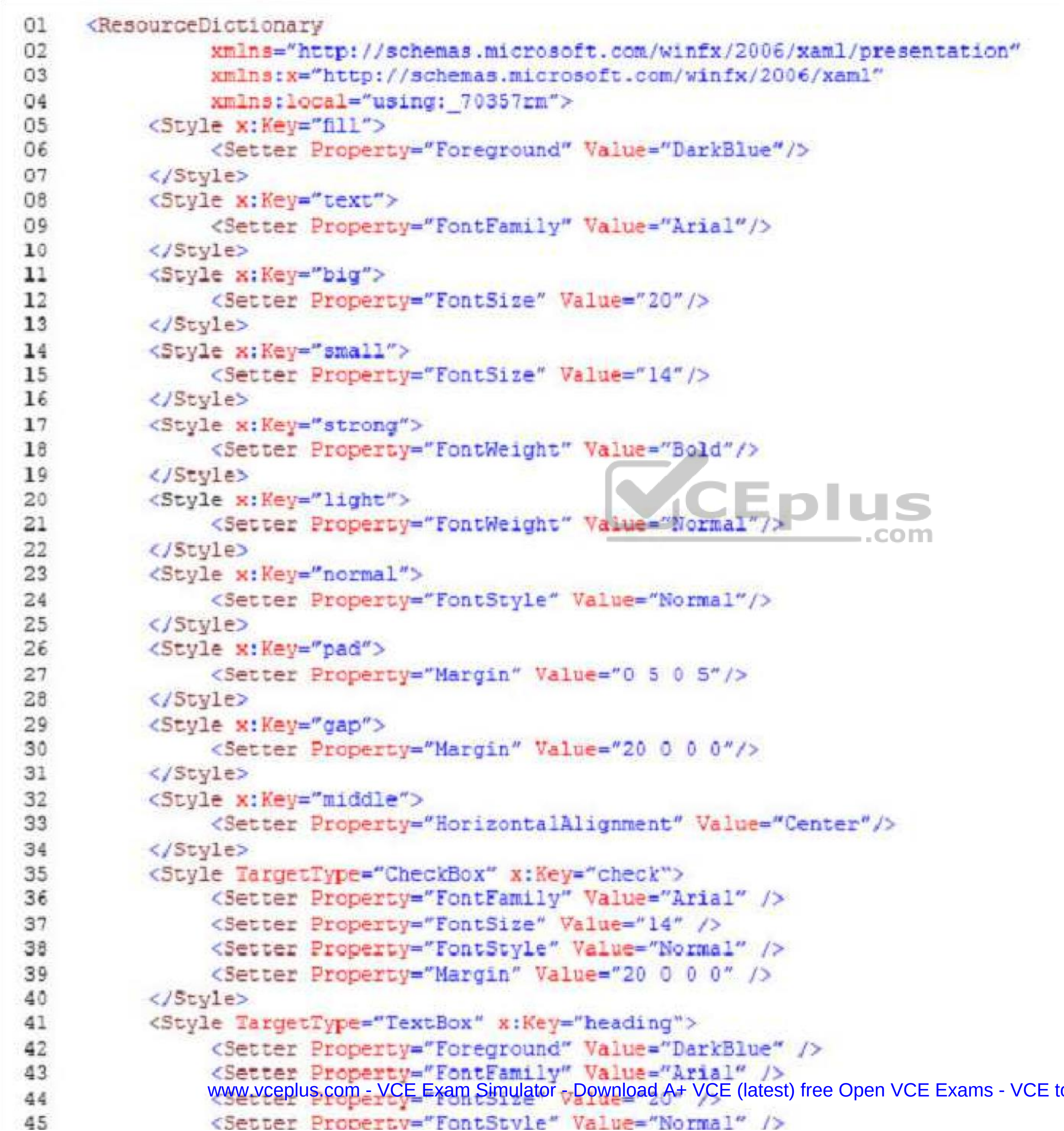
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



```
AS01 <Page
      x: Class ="_70357rm.Settings"
      xmlns="http://schemas.microsoft.com/winfix/2006/xaml/presentation"
      xmlns: x ="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns: local = "using:_70357rm"
      xmlns: d ="http://schemas.microsoft.com/expression/blend/2008"
      xmlns: m ="http://schemas.openxmlformats.org/markup-compatibility/2006"
      mc: Ignorable ="d">
AS02      <Grid Background ="AliceBlue">
AS03           <Border BorderBrush ="DarkBlue" BorderThickness = "5" />
AS04                <Grid Margin ="5 5 5 5">
AS05                     <StackPanel HorizontalAlignment ="Center">
AS06                          <TextBlock Text ="Date Settings" Foreground ="DarkBlue" FontFamily="Arial"
FontSize ="20" FontStyle ="Normal"
                               FontHeight ="Bold" Margin ="0 5 0 5" HorizontalAlignment="Center" />
AS07                          <StackPanel Orientation ="Horizontal">
AS08                               <CheckBox Content ="Center on Date" FontFamily ="Arial" FontSize="14"
FontStyle "Normal" Margin ="20,0,0,0"/>
AS09                               <CheckBox Content ="Set Start Date" FontFamily ="Arial" FontSize="14"
FontStyle "Normal" Margin ="20,0,0,0"/>
AS10                          </StackPanel>
AS11                          <TextBlock Text ="Start Date" Foreground ="DarkBlue" FontFamily="Arial"
FontSize ="20" FontStyle ="Normal"
                               FontWeight ="Bold" Margin ="0 5 0 5" HorizontalAlignment="Center"/>
AS12                          <StackPanel Orientation ="Horizontal">
AS13                               <TextBlock Text ="Month:" Width ="75" />
AS14                               <TextBox Width ="200" />
AS15                          </StackPanel>
AS16                          <StackPanel Orientation ="Horizontal">
AS17                               <TextBlock Text ="Day:" Width ="75" />
AS18                               <TextBox Width ="200 />
AS19                          </StackPanel>
AS20                          <StackPanel Orientation ="Horizontal">
AS21                               <TextBlock Text ="Year:" Width ="75" />
AS22                               <TextBlock Width ="200" />
AS23                          </StackPanel>
AS24                          <Ellipse Fill ="Blue" Width ="50" Height="50" Margin ="0 5 0 5"/>
AS25                          <TextBlock FontFamily ="Arial" FontSize ="14" Foreground="White" Text ="Save"
Margin ="127 -38 0 0"/>
AS26                     </StackPanel>
AS27                </Grid>
AS28
AS29      </Grid>
AS30 </Page>
```

ResourceDictionery.xaml

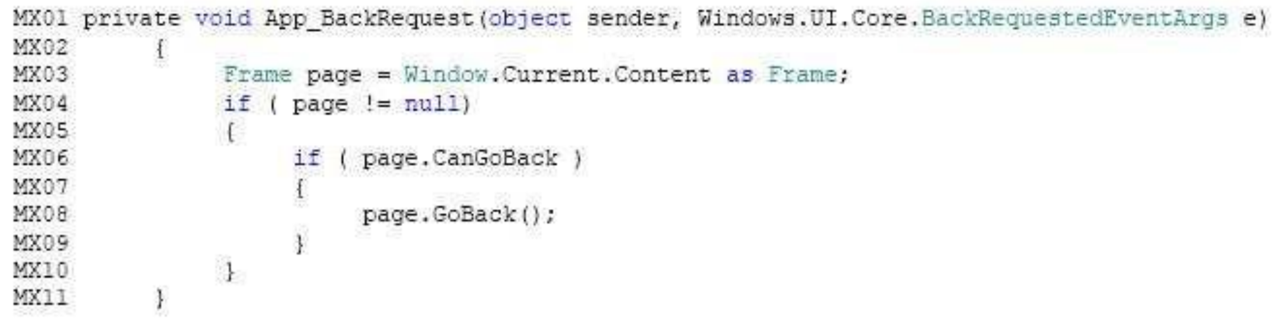
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only.)



```
<ResourceDictionary
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="using:_70357rm">
    <Style x:Key="fill">
        <Setter Property="Foreground" Value="DarkBlue"/>
    </Style>
    <Style x:Key="text">
        <Setter Property="FontFamily" Value="Arial"/>
    </Style>
    <Style x:Key="big">
        <Setter Property="FontSize" Value="20"/>
    </Style>
    <Style x:Key="small">
        <Setter Property="FontSize" Value="14"/>
    </Style>
    <Style x:Key="strong">
        <Setter Property="FontWeight" Value="Bold"/>
    </Style>
    <Style x:Key="light">
        <Setter Property="FontWeight" Value="Normal"/>
    </Style>
    <Style x:Key="normal">
        <Setter Property="FontStyle" Value="Normal"/>
    </Style>
    <Style x:Key="pad">
        <Setter Property="Margin" Value="0 5 0 5"/>
    </Style>
    <Style x:Key="gap">
        <Setter Property="Margin" Value="20 0 0 0"/>
    </Style>
    <Style x:Key="middle">
        <Setter Property="HorizontalAlignment" Value="Center"/>
    </Style>
    <Style TargetType="CheckBox" x:Key="check">
        <Setter Property="FontFamily" Value="Arial" />
        <Setter Property="FontSize" Value="14" />
        <Setter Property="FontStyle" Value="Normal" />
        <Setter Property="Margin" Value="20 0 0 0" />
    </Style>
    <Style TargetType="TextBox" x:Key="heading">
        <Setter Property="Foreground" Value="DarkBlue" />
        <Setter Property="FontFamily" Value="Arial" />
        <Setter Property="FontSize" Value="20" />
        <Setter Property="FontStyle" Value="Normal" />
```

MainPage.xaml.cs

Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)

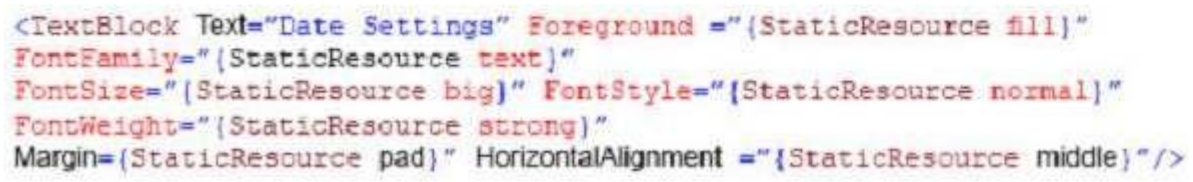


```
MX01 private void App_BackRequest(object sender, Windows.UI.Core.BackRequestedEventArgs e)
MX02      {
MX03           Frame page = Window.Current.Content as Frame;
MX04           if ( page != null)
MX05           {
MX06                if ( page.CanGoBack )
MX07                {
MX08                     page.GoBack();
MX09                }
MX10           }
MX11      }
```

You need to ensure that the Timeline app meets the XAML coding requirements.

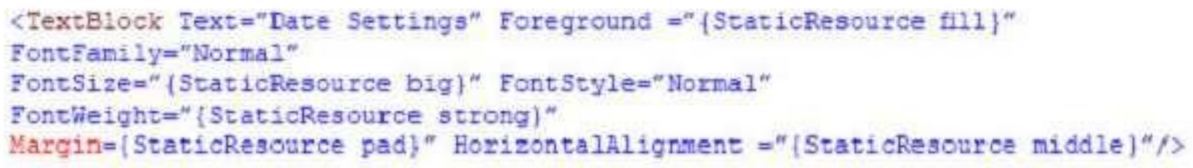
In Settings,xaml, which markup segment should you select to replace the markup segment at line AS06?

A.



```
<TextBlock Text="Date Settings" Foreground ="{StaticResource fill}"
FontFamily="{StaticResource text}"
FontSize="{StaticResource big}" FontStyle="{StaticResource normal}"
FontWeight="{StaticResource strong}"
Margin={StaticResource pad}" HorizontalAlignment ="{StaticResource middle}"/>
```

B.



```
<TextBlock Text="Date Settings" Foreground ="{StaticResource fill}"
FontFamily="Normal"
FontSize="{StaticResource big}" FontStyle="Normal"
FontWeight="{StaticResource strong}"
Margin={StaticResource pad}" HorizontalAlignment ="{StaticResource middle}"/>
```

C.

```
<TextBlock Text="Date Settings" Style="{ ThemeResource heading }" />
```

D.

```
<TextBlock Text="Date Settings" Style="{ StaticResource heading }" />
```

A. Option A
B. Option B
C. Option C
D. Option D

Correct Answer: A
Section: (none)
Explanation

Explanation/Reference:
Explanation:
From scenario: All code and markup must conform to the following style guidelines: ▪
Use resource dictionaries for styles that are used more than once.
▪ Use built-in properties of existing panels instead of using separate style objects.

XAML resources are objects that are referenced from markup more than once. Resources are defined in a ResourceDictionary, typically in a separate file or at the top of the markup page. In this scenario the ResourceDictionary is defined in the ResourceDictionery.xaml file.
You access members of the resource dictionary like any other dictionary.
Reference: https://docs.microsoft.com/en-us/windows/uwp/controls-and-patterns/resourcedictionary-and-xaml-resource-references

QUESTION 3 HOTSPOT

Case Study
This is a case study. Case studies are not limited separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.
At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.
To start the case study
To display the first question on this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

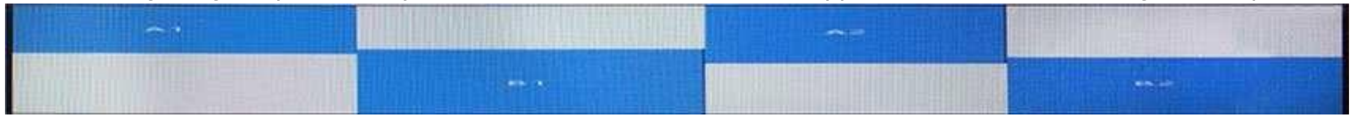
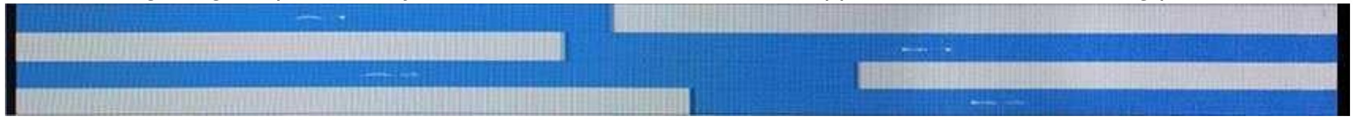
Background

You are developing an application named Timeline that presents information on a timeline. The app allows users to create items and enter details about the item. The app displays item names on a timeline. When users select an item name on the timeline, the app displays additional content about the item.

Business requirements

Timeline section

The timeline element of the app has the following layout requirements:
▪ The timeline must adapt to the screen size and orientation of the device.
▪ The timeline size must dynamically change if the window containing the content is resized by the user.
▪ The user must be able to scroll through the timeline horizontally when the device is in landscape mode.
▪ The user must be able to scroll through the timeline vertically when the device is in portrait mode.
▪ The timeline must begin scrolling as soon as a scroll is detected. Scrolling must continue for a short distance after the scroll input has stopped. ▪ Scroll bars or panning controls must always be visible.

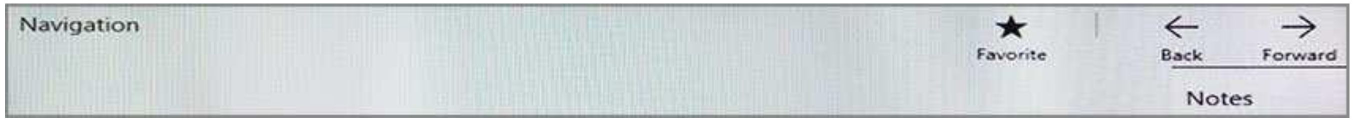
The following image depicts the layout for the timeline section of the app when the device is using landscape orientation:

The following image depicts the layout for the timeline section of the app when the device is using portrait orientation:


Content section

The content element of the app has the following layout requirements:
▪ When a user selects an item on the timeline, the details for that item must display beneath or to the right of the timeline.
▪ The content section must display one page of information. The element must be a child of the selected item in the timeline. ▪ Users must be able to return to a previously selected event by pressing the Back button.

User interface

The user must be able to navigate the application using the interface below:



▪ The Favorite button marks the current content to be displayed in a Favorites panel.
▪ The Back and Forward buttons navigate through the app selection history. Both buttons must be available on all devices.
▪ The Notes button allows the user to manage notes about the current content.
▪ The app must support touch, mouse, and stylus input.
▪ The app layout must automatically adapt to the screen size and orientation.

Technical requirements

Layout

You identify the following layout requirements:

General
▪ All user interface (UI) elements must continuously scale when a user resizes the window.
▪ UI controls must be smaller and spaced closer together if there is a mouse or stylus available.
▪ UI controls must be larger and spaced farther apart if the device supports touch and there is no mouse or pointer available.

Timeline
▪ The timeline must be displayed in a horizontal layout when the device is in a landscape orientation or when the horizontal width is greater than the vertical height. ▪ The timeline must be displayed in a vertical layout when the device is in a portrait orientation or when the vertical height is greater than the horizontal width. ▪ Each item in the past must be linked to the next item in the future.
▪ Users must be able to scroll from past events to future events or from future events to past events. ▪ The app must only allow one level of detail to be linked to each item in the timeline.

Optimization

You must optimize the app using the following guidelines:

▪ You must minimize the time it takes to display content when an item on the timeline is selected. ▪
The app must respect memory and resource constraints for all devices.

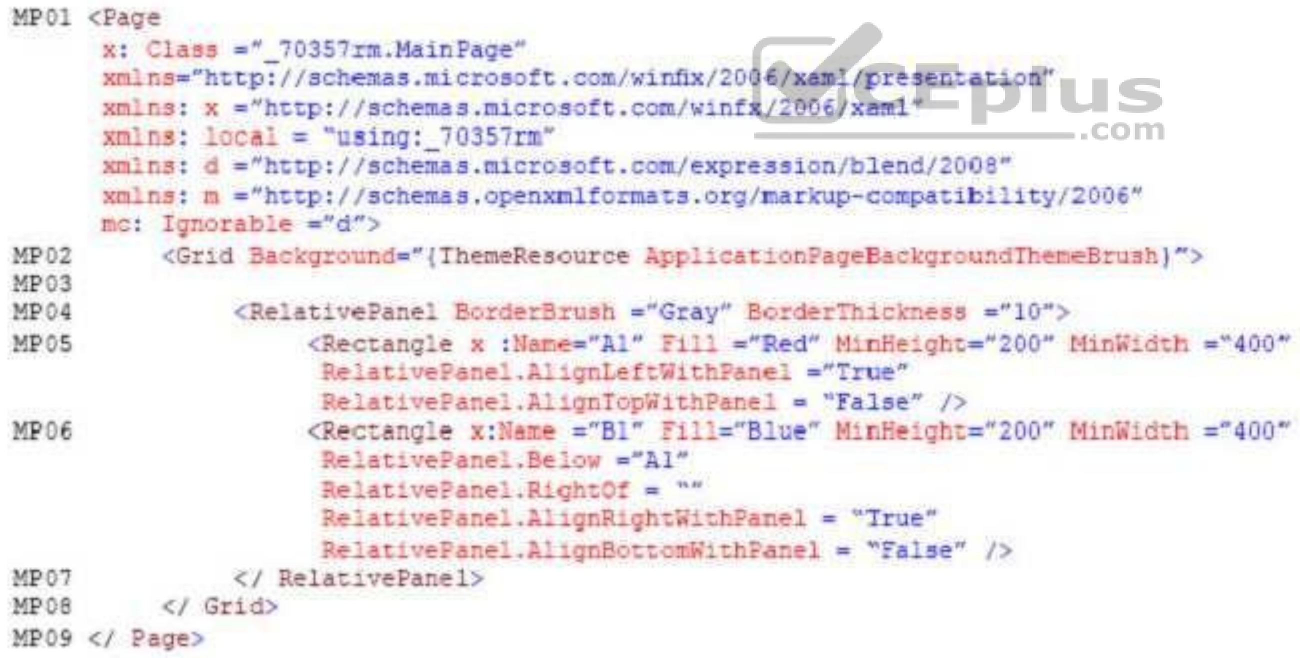
XAML coding style

All code and markup must conform to the following style guidelines:
▪ Use resource dictionaries for styles that are used more than once.
▪ Limit the use of nested panels.
▪ Use built-in properties of existing panels instead of using separate style objects.
▪ Use the navigation structure that best models the data without exceeding the requirements of the app. Application

structure

MainPage.xaml

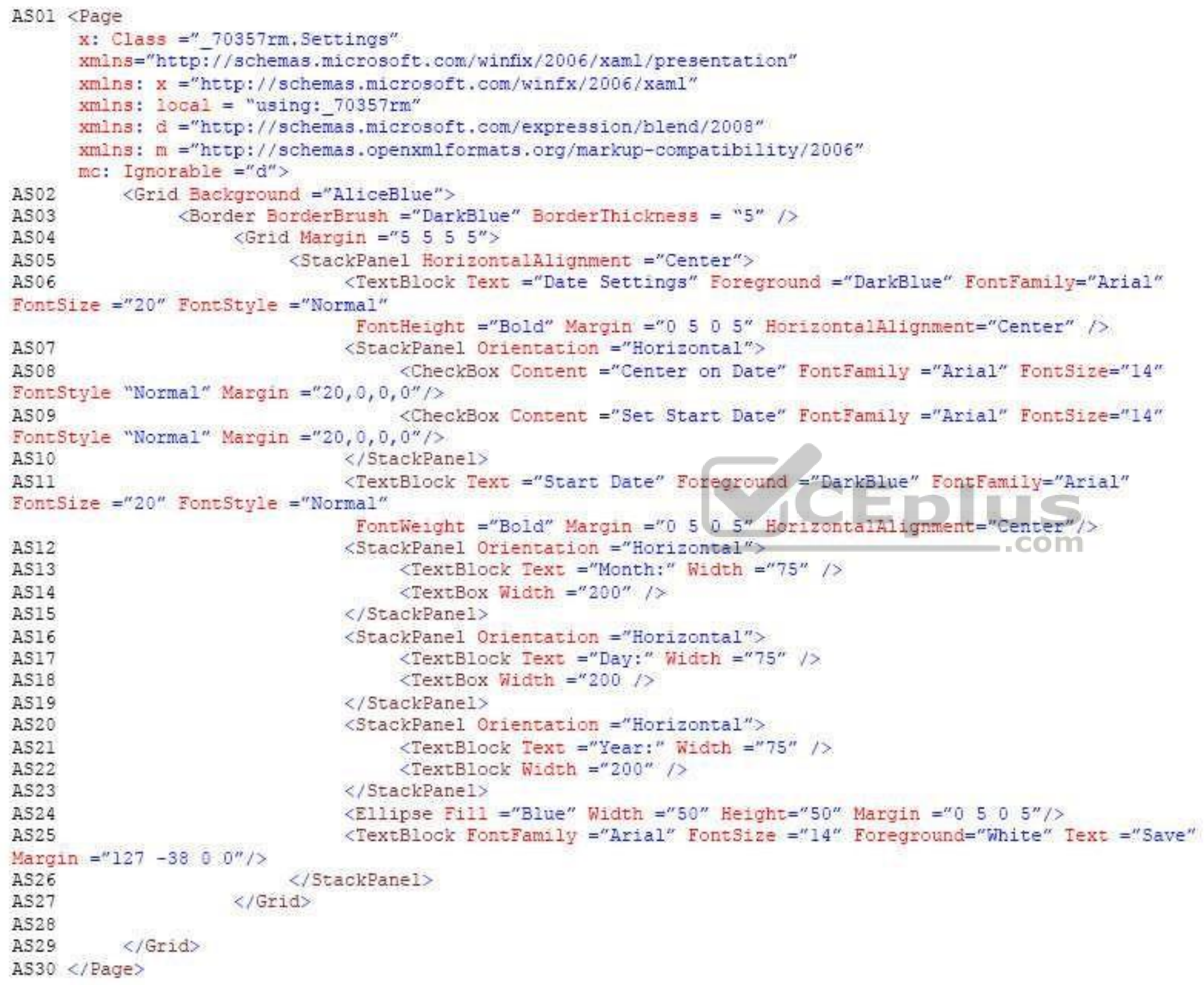
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



```
MP01 <Page
      x: Class ="_70357rm.MainPage"
      xmlns="http://schemas.microsoft.com/winfix/2006/xaml/presentation"
      xmlns: x ="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns: local = "using:_70357rm"
      xmlns: d ="http://schemas.microsoft.com/expression/blend/2008"
      xmlns: m ="http://schemas.openxmlformats.org/markup-compatibility/2006"
      mc: Ignorable ="d">
MP02      <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
MP03
MP04           <RelativePanel BorderBrush ="Gray" BorderThickness ="10">
MP05                <Rectangle x :Name="A1" Fill ="Red" MinHeight="200" MinWidth ="400"
                     RelativePanel.AlignLeftWithPanel ="True"
                     RelativePanel.AlignTopWithPanel = "False" />
MP06                <Rectangle x:Name ="B1" Fill="Blue" MinHeight="200" MinWidth ="400"
                     RelativePanel.Below ="A1"
                     RelativePanel.RightOf = ""
                     RelativePanel.AlignRightWithPanel = "True"
                     RelativePanel.AlignBottomWithPanel = "False" />
MP07           </ RelativePanel>
MP08      </ Grid>
MP09 </ Page>
```

Settings.xaml

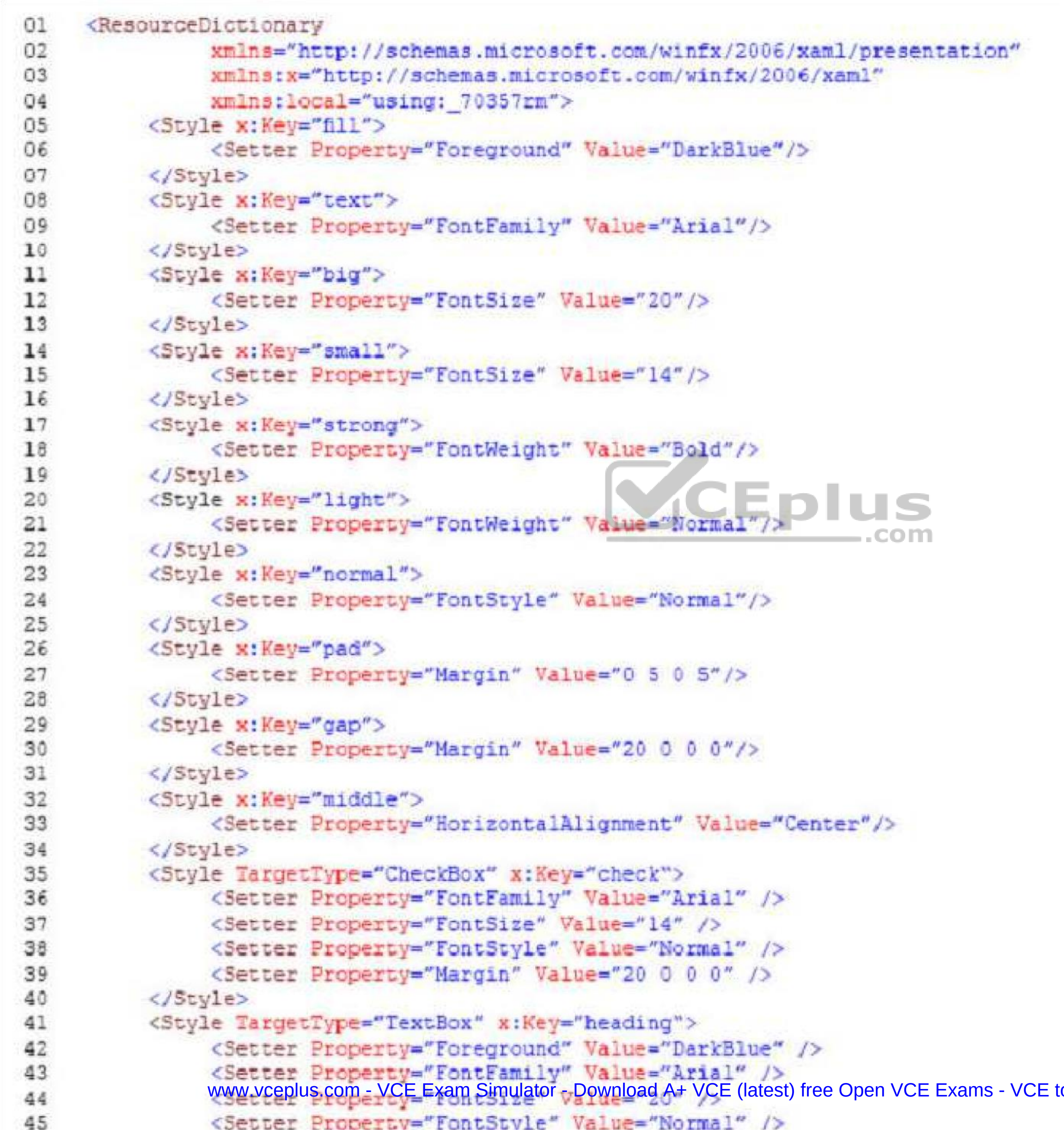
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



```
AS01 <Page
      x: Class ="_70357rm.Settings"
      xmlns="http://schemas.microsoft.com/winfix/2006/xaml/presentation"
      xmlns: x ="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns: local = "using:_70357rm"
      xmlns: d ="http://schemas.microsoft.com/expression/blend/2008"
      xmlns: m ="http://schemas.openxmlformats.org/markup-compatibility/2006"
      mc: Ignorable ="d">
AS02      <Grid Background ="AliceBlue">
AS03           <Border BorderBrush ="DarkBlue" BorderThickness = "5" />
AS04                <Grid Margin ="5 5 5 5">
AS05                     <StackPanel HorizontalAlignment ="Center">
AS06                          <TextBlock Text ="Date Settings" Foreground ="DarkBlue" FontFamily="Arial"
FontSize ="20" FontStyle ="Normal"
                               FontHeight ="Bold" Margin ="0 5 0 5" HorizontalAlignment="Center" />
AS07                          <StackPanel Orientation ="Horizontal">
AS08                               <CheckBox Content ="Center on Date" FontFamily ="Arial" FontSize="14"
FontStyle "Normal" Margin ="20,0,0,0"/>
AS09                               <CheckBox Content ="Set Start Date" FontFamily ="Arial" FontSize="14"
FontStyle "Normal" Margin ="20,0,0,0"/>
AS10                          </StackPanel>
AS11                          <TextBlock Text ="Start Date" Foreground ="DarkBlue" FontFamily="Arial"
FontSize ="20" FontStyle ="Normal"
                               FontWeight ="Bold" Margin ="0 5 0 5" HorizontalAlignment="Center"/>
AS12                          <StackPanel Orientation ="Horizontal">
AS13                               <TextBlock Text ="Month:" Width ="75" />
AS14                               <TextBox Width ="200" />
AS15                          </StackPanel>
AS16                          <StackPanel Orientation ="Horizontal">
AS17                               <TextBlock Text ="Day:" Width ="75" />
AS18                               <TextBox Width ="200 />
AS19                          </StackPanel>
AS20                          <StackPanel Orientation ="Horizontal">
AS21                               <TextBlock Text ="Year:" Width ="75" />
AS22                               <TextBlock Width ="200" />
AS23                          </StackPanel>
AS24                          <Ellipse Fill ="Blue" Width ="50" Height="50" Margin ="0 5 0 5"/>
AS25                          <TextBlock FontFamily ="Arial" FontSize ="14" Foreground="White" Text ="Save"
Margin ="127 -38 0 0"/>
AS26                     </StackPanel>
AS27                </Grid>
AS28
AS29      </Grid>
AS30 </Page>
```

ResourceDictionery.xaml

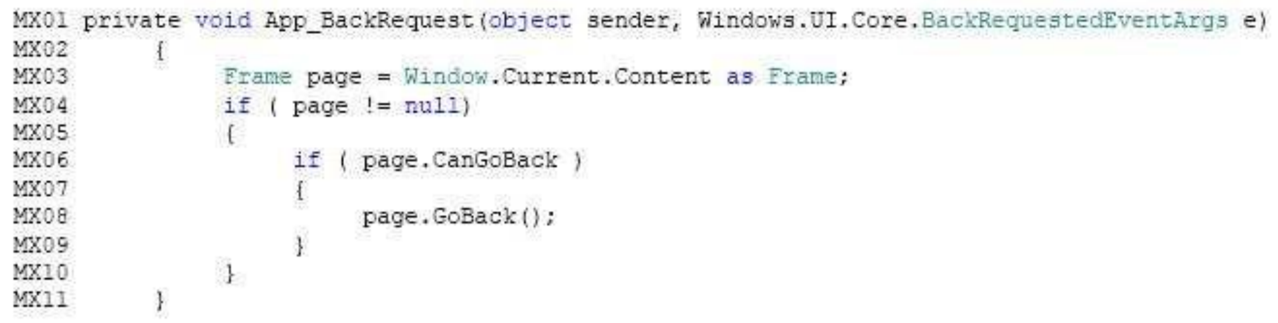
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only.)



```
<ResourceDictionary
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="using:_70357rm">
    <Style x:Key="fill">
        <Setter Property="Foreground" Value="DarkBlue"/>
    </Style>
    <Style x:Key="text">
        <Setter Property="FontFamily" Value="Arial"/>
    </Style>
    <Style x:Key="big">
        <Setter Property="FontSize" Value="20"/>
    </Style>
    <Style x:Key="small">
        <Setter Property="FontSize" Value="14"/>
    </Style>
    <Style x:Key="strong">
        <Setter Property="FontWeight" Value="Bold"/>
    </Style>
    <Style x:Key="light">
        <Setter Property="FontWeight" Value="Normal"/>
    </Style>
    <Style x:Key="normal">
        <Setter Property="FontStyle" Value="Normal"/>
    </Style>
    <Style x:Key="pad">
        <Setter Property="Margin" Value="0 5 0 5"/>
    </Style>
    <Style x:Key="gap">
        <Setter Property="Margin" Value="20 0 0 0"/>
    </Style>
    <Style x:Key="middle">
        <Setter Property="HorizontalAlignment" Value="Center"/>
    </Style>
    <Style TargetType="CheckBox" x:Key="check">
        <Setter Property="FontFamily" Value="Arial" />
        <Setter Property="FontSize" Value="14" />
        <Setter Property="FontStyle" Value="Normal" />
        <Setter Property="Margin" Value="20 0 0 0" />
    </Style>
    <Style TargetType="TextBox" x:Key="heading">
        <Setter Property="Foreground" Value="DarkBlue" />
        <Setter Property="FontFamily" Value="Arial" />
        <Setter Property="FontSize" Value="20" />
        <Setter Property="FontStyle" Value="Normal" />
```

MainPage.xaml.cs

Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



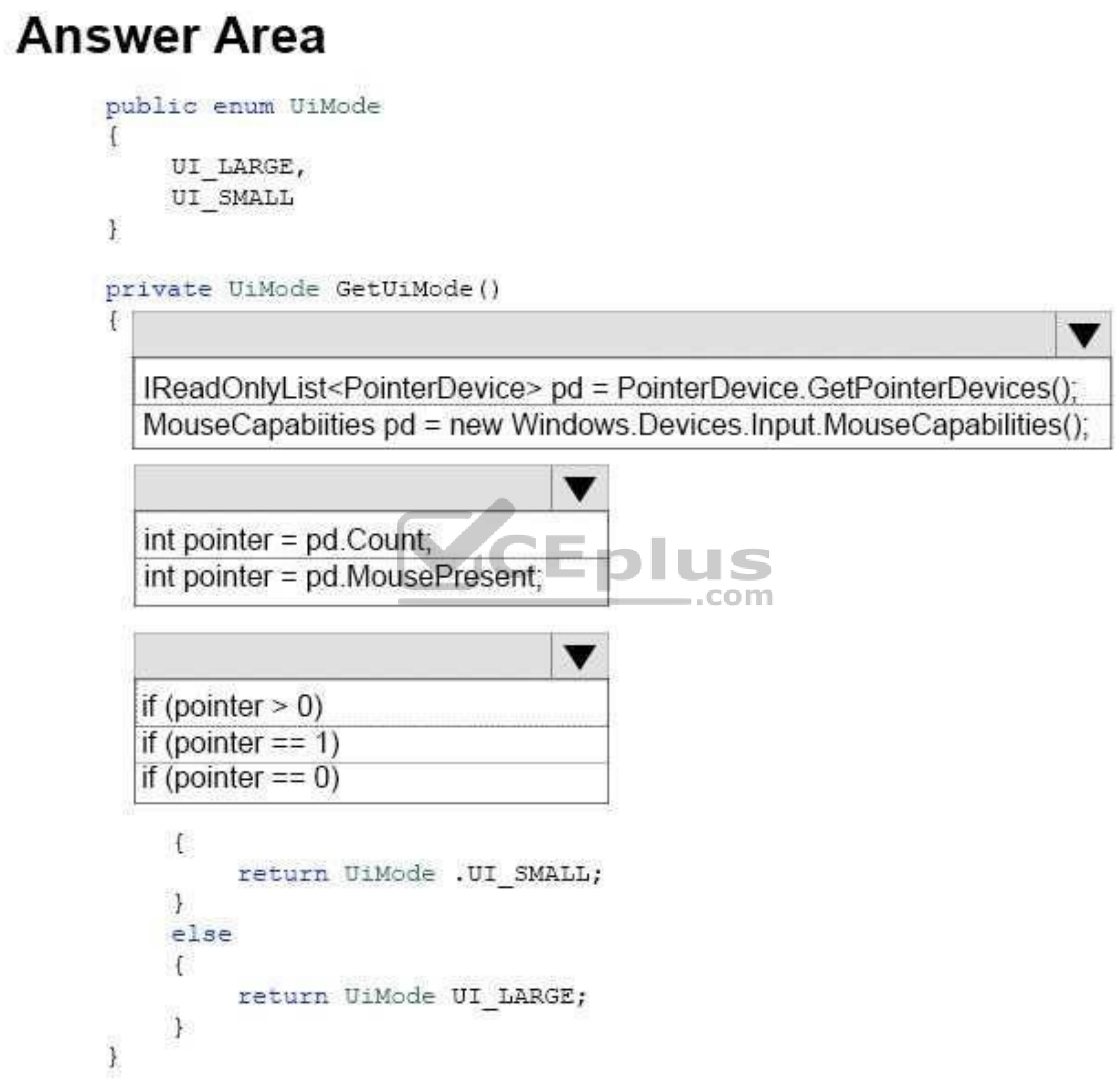
```
MX01 private void App_BackRequest(object sender, Windows.UI.Core.BackRequestedEventArgs e)
MX02       {
MX03             Frame page = Window.Current.Content as Frame;
MX04             if ( page != null)
MX05             {
MX06                   if ( page.CanGoBack )
MX07                   {
MX08                         page.GoBack();
MX09                   }
MX10             }
MX11       }
```

You need to properly handle the size of the user interface objects.

How should you complete the method? To answer, select the appropriate code segment from each list in the answer area.

NOTE: Each correct selection is worth one point.

Hot Area:

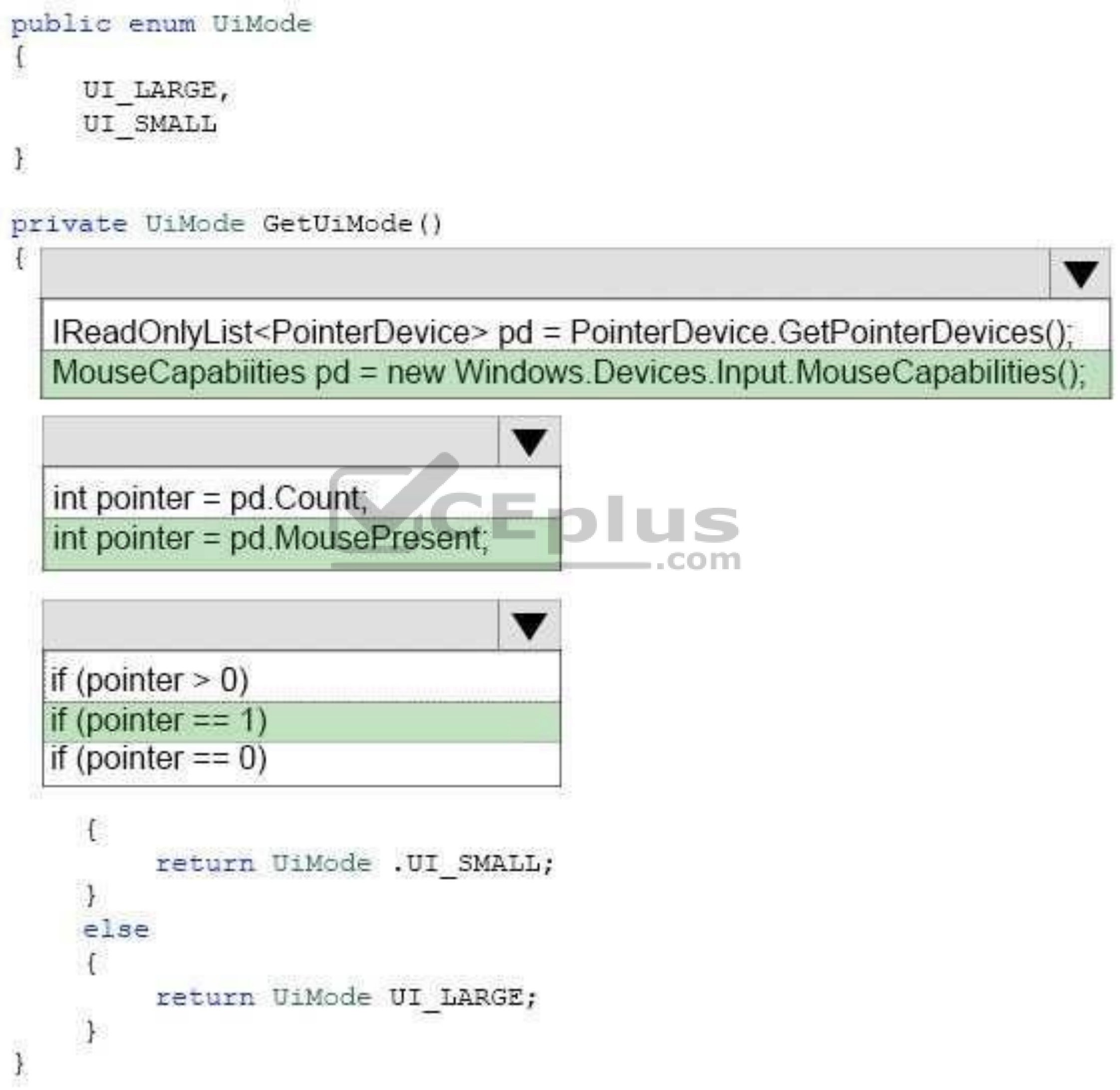


Answer Area

```
public enum UiMode
{
    UI_LARGE,
    UI_SMALL
}

private UiMode GetUiMode()
{
    [ ▼ ]
      IReadOnlyList<PointerDevice> pd = PointerDevice.GetPointerDevices();
      MouseCapabiities pd = new Windows.Devices.Input.MouseCapabilities();

    [ ▼ ]
      int pointer = pd.Count;
      int pointer = pd.MousePresent;

    [ ▼ ]
      if (pointer > 0)
      if (pointer == 1)
      if (pointer == 0)
    {
        return UiMode .UI_SMALL;
    }
    else
    {
        return UiMode UI_LARGE;
    }
}
```

Correct Answer:

Answer Area



```
public enum UiMode
{
    UI_LARGE,
    UI_SMALL
}

private UiMode GetUiMode()
{
    IReadOnlyList<PointerDevice> pd = PointerDevice.GetPointerDevices();
    [✓ MouseCapabiities pd = new Windows.Devices.Input.MouseCapabilities();]

    int pointer = pd.Count;
    [✓ int pointer = pd.MousePresent;]

    if (pointer > 0)
    [✓ if (pointer == 1)]
    if (pointer == 0)

    {
        return UiMode .UI_SMALL;
    }
    else
    {
        return UiMode UI_LARGE;
    }
}
```

Section: (none)
Explanation

Explanation/Reference:
Explanation:

From scenario:
UI controls must be smaller and spaced closer together if there is a mouse or stylus available.
UI controls must be larger and spaced farther apart if the device supports touch and there is no mouse or pointer available.

Box 1: MouseCapabilities pd = new Windows.Devices.Input.MouseCapabilities();
The Windows.Devices.Input namespace contains the MouseCapabilities class used to retrieve the properties exposed by one or more connected mice. Just create a new MouseCapabilities object and get the properties you're interested in.

Box 2: int pointer = pd.MousePresent;
Example:
MouseCapabilities mouseCapabilities = new Windows.Devices.Input.MouseCapabilities();
MousePresent.Text = mouseCapabilities.MousePresent != 0 ? "Yes" : "No";

Box 3: if (pointer == 1)
This is true if a mouse is present.

Reference: https://docs.microsoft.com/en-us/windows/uwp/input-and-devices/identify-input-devices

QUESTION 4
Case Study
This is a case study. Case studies are not limited separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.
To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.
At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.
To start the case study
To display the first question on this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

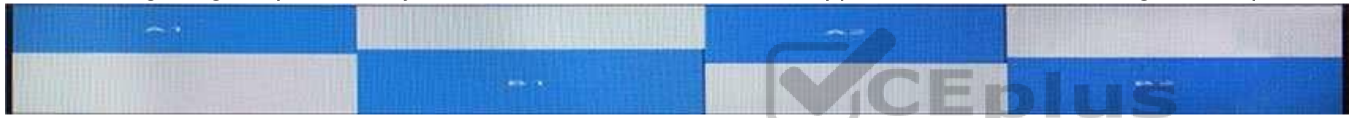
Background
You are developing an application named Timeline that presents information on a timeline. The app allows users to create items and enter details about the item. The app displays item names on a timeline. When users select an item name on the timeline, the app displays additional content about the item.

Business requirements

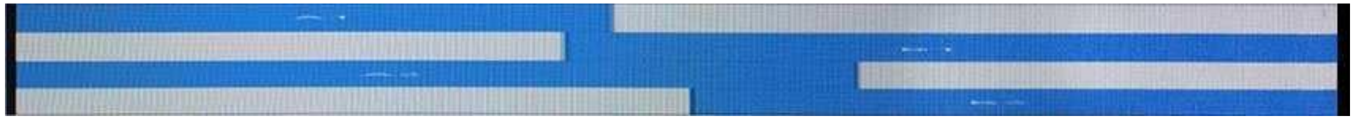
Timeline section

The timeline element of the app has the following layout requirements:
▪ The timeline must adapt to the screen size and orientation of the device.
▪ The timeline size must dynamically change if the window containing the content is resized by the user.
▪ The user must be able to scroll through the timeline horizontally when the device is in landscape mode.
▪ The user must be able to scroll through the timeline vertically when the device is in portrait mode.
▪ The timeline must begin scrolling as soon as a scroll is detected. Scrolling must continue for a short distance after the scroll input has stopped. ▪
Scroll bars or panning controls must always be visible.

The following image depicts the layout for the timeline section of the app when the device is using landscape orientation:



The following image depicts the layout for the timeline section of the app when the device is using portrait orientation:

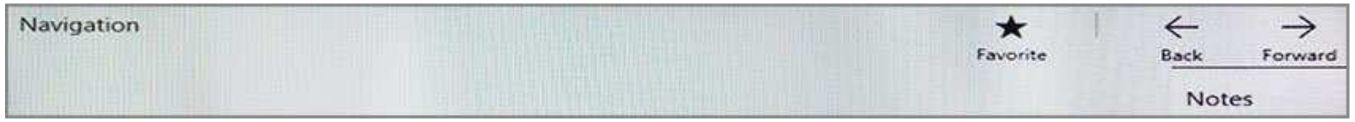


Content section

The content element of the app has the following layout requirements:
▪ When a user selects an item on the timeline, the details for that item must display beneath or to the right of the timeline.
▪ The content section must display one page of information. The element must be a child of the selected item in the timeline. ▪
Users must be able to return to a previously selected event by pressing the Back button.

User interface

The user must be able to navigate the application using the interface below:



▪ The Favorite button marks the current content to be displayed in a Favorites panel.
▪ The Back and Forward buttons navigate through the app selection history. Both buttons must be available on all devices. ▪
The Notes button allows the user to manage notes about the current content.
▪ The app must support touch, mouse, and stylus input.
▪ The app layout must automatically adapt to the screen size and orientation.

Technical requirements

Layout

You identify the following layout requirements:

General
▪ All user interface (UI) elements must continuously scale when a user resizes the window.
▪ UI controls must be smaller and spaced closer together if there is a mouse or stylus available.
▪ UI controls must be larger and spaced farther apart if the device supports touch and there is no mouse or pointer available.

Timeline

▪ The timeline must be displayed in a horizontal layout when the device is in a landscape orientation or when the horizontal width is greater than the vertical height. ▪ The timeline must be displayed in a vertical layout when the device is in a portrait orientation or when the vertical height is greater than the horizontal width. ▪ Each item in the past must be linked to the next item in the future.
▪ Users must be able to scroll from past events to future events or from future events to past events. ▪
The app must only allow one level of detail to be linked to each item in the timeline.

Optimization

You must optimize the app using the following guidelines:
▪ You must minimize the time it takes to display content when an item on the timeline is selected. ▪
The app must respect memory and resource constraints for all devices.

XAML coding style

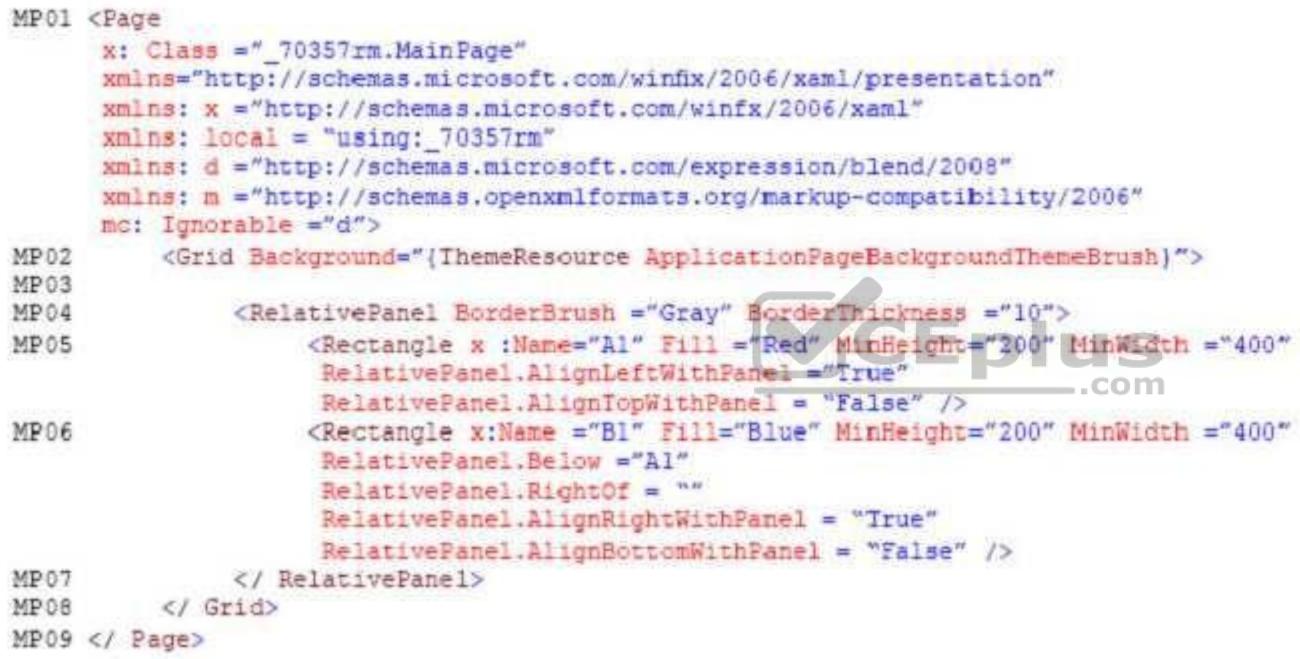
All code and markup must conform to the following style guidelines:
▪ Use resource dictionaries for styles that are used more than once.
▪ Limit the use of nested panels.

- Use built-in properties of existing panels instead of using separate style objects.
- Use the navigation structure that best models the data without exceeding the requirements of the app. Application

structure

MainPage.xaml

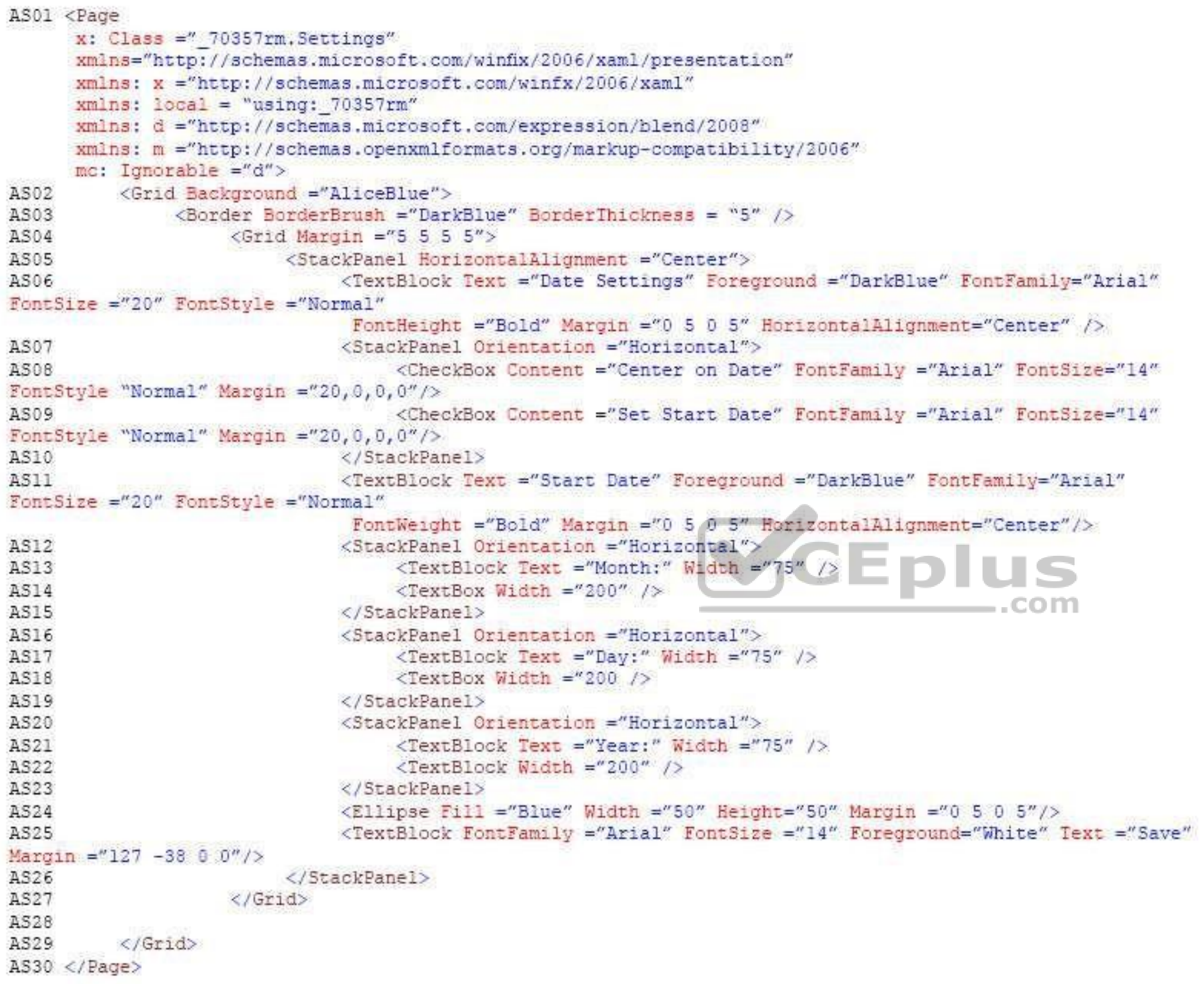
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



```
MP01 <Page
      x: Class ="_70357rm.MainPage"
      xmlns="http://schemas.microsoft.com/winfix/2006/xaml/presentation"
      xmlns: x ="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns: local = "using:_70357rm"
      xmlns: d ="http://schemas.microsoft.com/expression/blend/2008"
      xmlns: m ="http://schemas.openxmlformats.org/markup-compatibility/2006"
      mc: Ignorable ="d">
MP02     <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
MP03
MP04         <RelativePanel BorderBrush ="Gray" BorderThickness ="10">
MP05             <Rectangle x :Name="A1" Fill ="Red" MinHeight="200" MinWidth ="400"
                  RelativePanel.AlignLeftWithPanel ="True"
                  RelativePanel.AlignTopWithPanel = "False" />
MP06             <Rectangle x:Name ="B1" Fill="Blue" MinHeight="200" MinWidth ="400"
                  RelativePanel.Below ="A1"
                  RelativePanel.RightOf = ""
                  RelativePanel.AlignRightWithPanel = "True"
                  RelativePanel.AlignBottomWithPanel = "False" />
MP07         </ RelativePanel>
MP08     </ Grid>
MP09 </ Page>
```

Settings.xaml

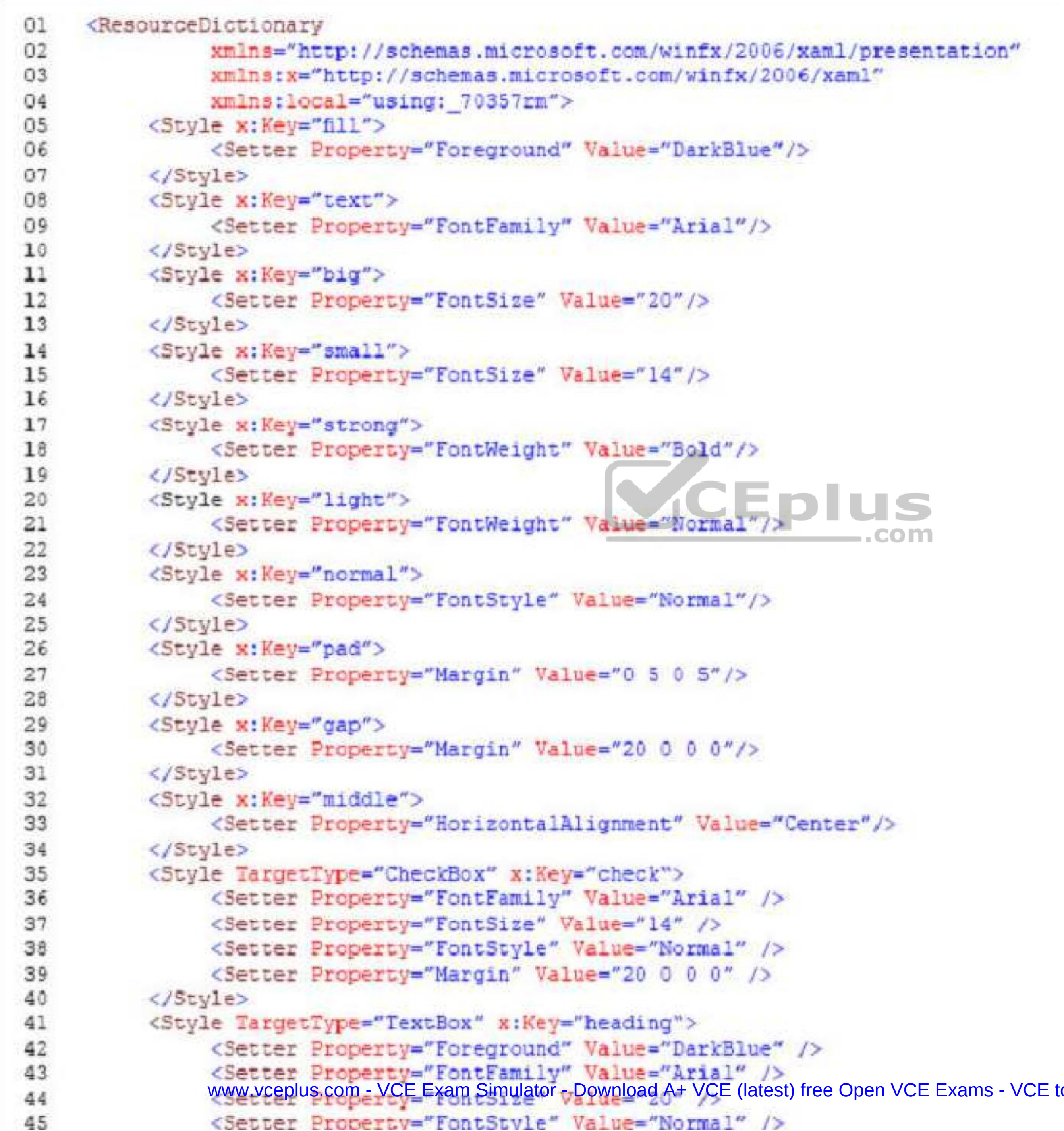
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



```
AS01 <Page
      x: Class ="_70357rm.Settings"
      xmlns="http://schemas.microsoft.com/winfix/2006/xaml/presentation"
      xmlns: x ="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns: local = "using:_70357rm"
      xmlns: d ="http://schemas.microsoft.com/expression/blend/2008"
      xmlns: m ="http://schemas.openxmlformats.org/markup-compatibility/2006"
      mc: Ignorable ="d">
AS02     <Grid Background ="AliceBlue">
AS03          <Border BorderBrush ="DarkBlue" BorderThickness = "5" />
AS04               <Grid Margin ="5 5 5 5">
AS05                    <StackPanel HorizontalAlignment ="Center">
AS06                         <TextBlock Text ="Date Settings" Foreground ="DarkBlue" FontFamily="Arial"
FontSize ="20" FontStyle ="Normal"
                              FontHeight ="Bold" Margin ="0 5 0 5" HorizontalAlignment="Center" />
AS07                         <StackPanel Orientation ="Horizontal">
AS08                              <CheckBox Content ="Center on Date" FontFamily ="Arial" FontSize="14"
FontStyle "Normal" Margin ="20,0,0,0"/>
AS09                              <CheckBox Content ="Set Start Date" FontFamily ="Arial" FontSize="14"
FontStyle "Normal" Margin ="20,0,0,0"/>
AS10                         </StackPanel>
AS11                         <TextBlock Text ="Start Date" Foreground ="DarkBlue" FontFamily="Arial"
FontSize ="20" FontStyle ="Normal"
                              FontWeight ="Bold" Margin ="0 5 0 5" HorizontalAlignment="Center"/>
AS12                         <StackPanel Orientation ="Horizontal">
AS13                              <TextBlock Text ="Month:" Width ="75" />
AS14                              <TextBox Width ="200" />
AS15                         </StackPanel>
AS16                         <StackPanel Orientation ="Horizontal">
AS17                              <TextBlock Text ="Day:" Width ="75" />
AS18                              <TextBox Width ="200 />
AS19                         </StackPanel>
AS20                         <StackPanel Orientation ="Horizontal">
AS21                              <TextBlock Text ="Year:" Width ="75" />
AS22                              <TextBlock Width ="200" />
AS23                         </StackPanel>
AS24                         <Ellipse Fill ="Blue" Width ="50" Height="50" Margin ="0 5 0 5"/>
AS25                         <TextBlock FontFamily ="Arial" FontSize ="14" Foreground="White" Text ="Save"
Margin ="127 -38 0 0"/>
AS26                    </StackPanel>
AS27               </Grid>
AS28
AS29     </Grid>
AS30 </Page>
```

ResourceDictionery.xaml

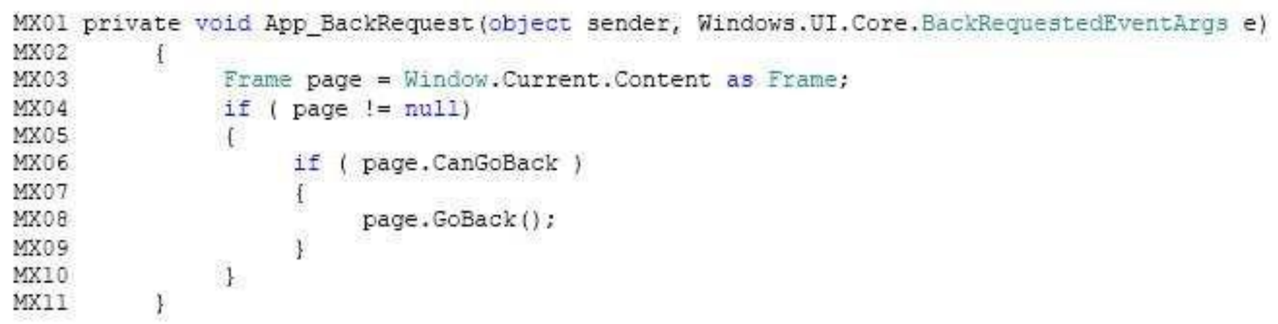
Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only.)



```
01  <ResourceDictionary
02            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
03            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
04            xmlns:local="using:_70357rm">
05       <Style x:Key="fill">
06            <Setter Property="Foreground" Value="DarkBlue"/>
07       </Style>
08       <Style x:Key="text">
09            <Setter Property="FontFamily" Value="Arial"/>
10       </Style>
11       <Style x:Key="big">
12            <Setter Property="FontSize" Value="20"/>
13       </Style>
14       <Style x:Key="small">
15            <Setter Property="FontSize" Value="14"/>
16       </Style>
17       <Style x:Key="strong">
18            <Setter Property="FontWeight" Value="Bold"/>
19       </Style>
20       <Style x:Key="light">
21            <Setter Property="FontWeight" Value="Normal"/>
22       </Style>
23       <Style x:Key="normal">
24            <Setter Property="FontStyle" Value="Normal"/>
25       </Style>
26       <Style x:Key="pad">
27            <Setter Property="Margin" Value="0 5 0 5"/>
28       </Style>
29       <Style x:Key="gap">
30            <Setter Property="Margin" Value="20 0 0 0"/>
31       </Style>
32       <Style x:Key="middle">
33            <Setter Property="HorizontalAlignment" Value="Center"/>
34       </Style>
35       <Style TargetType="CheckBox" x:Key="check">
36            <Setter Property="FontFamily" Value="Arial" />
37            <Setter Property="FontSize" Value="14" />
38            <Setter Property="FontStyle" Value="Normal" />
39            <Setter Property="Margin" Value="20 0 0 0" />
40       </Style>
41       <Style TargetType="TextBox" x:Key="heading">
42            <Setter Property="Foreground" Value="DarkBlue" />
43            <Setter Property="FontFamily" Value="Arial" />
44            <Setter Property="FontSize" Value="20" />
45            <Setter Property="FontStyle" Value="Normal" />
```

MainPage.xaml.cs

Relevant portions of the app files are shown below. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)



```
MX01 private void App_BackRequest(object sender, Windows.UI.Core.BackRequestedEventArgs e)
MX02        {
MX03             Frame page = Window.Current.Content as Frame;
MX04             if ( page != null)
MX05             {
MX06                  if ( page.CanGoBack )
MX07                  {
MX08                       page.GoBack();
MX09                  }
MX10             }
MX11        }
```

You need to configure the app to meet the load time requirements.

What should you do?

A. Set the value of the CacheSize to 0.
B. Set the value of the CacheMode property to BitmapCache.
C. Set the value of the NavigationCacheMode property to Enabled.
D. Set the value of the NavigationCacheMode property to Disabled.

Correct Answer: C
Section: (none)
Explanation

Explanation/Reference:
Explanation:

Scenario: You must optimize the app using the following guidelines:
▪ You must minimize the time it takes to display content when an item on the timeline is selected.
▪ The app must respect memory and resource constraints for all devices.
You use the NavigationCacheMode property to specify whether a new instance of the page is created for each visit to the page or if a previously constructed instance of the page that has been saved in the cache is used for each visit.

The default value for the NavigationCacheMode property is Disabled. Set the NavigationCacheMode property to Enabled or Required when a new instance of the page is not essential for each visit. By using a cached instance of the page, you can improve the performance of your application and reduce the load on your server.

Reference: https://msdn.microsoft.com/en-us/library/system.windows.controls.page.navigationcachemode(v=vs.95).aspx

QUESTION 5
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution. Determine whether the solution meets the stated goals.

You are developing a Universal Windows Platform (UWP) app.

Your app stores files on a user's device.

You need to be able to replace the existing files with new files generated by the user.

Solution: You run the StorageFile.OpenSequentialReadAsync method to replace the existing file.

Does this meet the goal?

A. Yes
B. No

Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:
Explanation:
The OpenSequentialReadAsync() method opens a sequential-access stream over the current file for reading file contents.

Reference: https://docs.microsoft.com/en-us/uwp/api/windows.storage.storagefile

QUESTION 6
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution. Determine whether the solution meets the stated goals.

You are developing a Universal Windows Platform (UWP) app.

Your app stores files on a user's device.
You need to be able to replace the existing files with new files generated by the user.

Solution: You run the StorageFile.GetParentAsync method to get a reference to the existing file. Then, you run the StorageFile.CreateStreamedFileAsync method to create the new file at the same location.

Does this meet the goal?

A. Yes
B. No

Correct Answer: A
Section: (none)
Explanation

Explanation/Reference:
Explanation:
The GetParentAsync() method gets the parent folder of the current file.
The CreateStreamedFileAsync method can be used to create a StorageFile that can be passed to other methods or passed to another app through app contracts.

Reference: https://docs.microsoft.com/en-us/uwp/api/windows.storage.storagefile

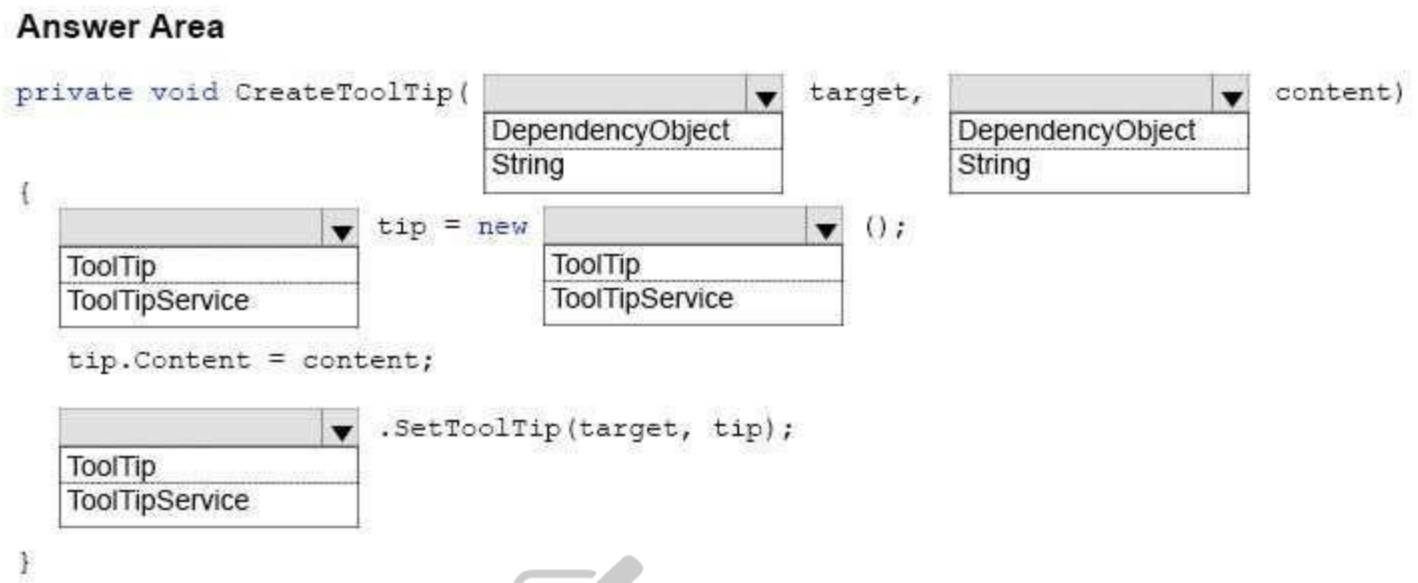
QUESTION 7 HOTSPOT

You are developing an app that displays photos.

You need to create a method that displays informational text when a user hovers the pointer over a photo.

How should you complete the method? To answer, select the appropriate code segment from each list in the answer area.

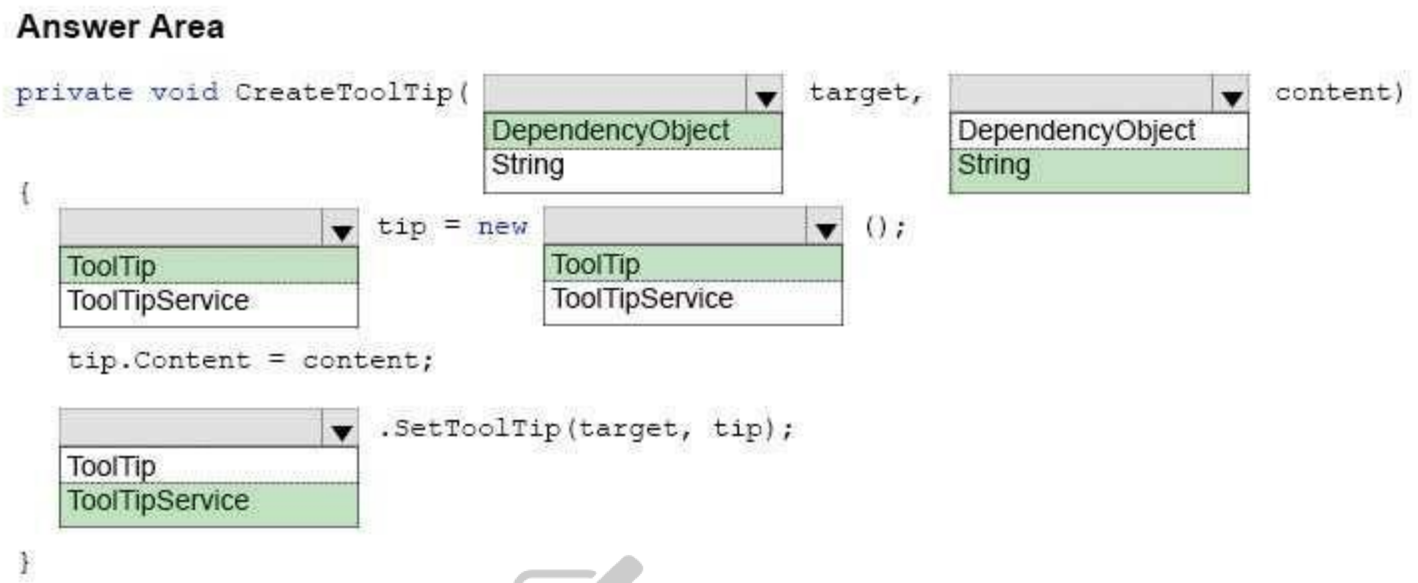
Hot Area:



Answer Area

```
private void CreateToolTip( [DependencyObject | String] target, [DependencyObject | String] content)
{
    [ToolTip | ToolTipService] tip = new [ToolTip | ToolTipService] ();
    tip.Content = content;
    [ToolTip | ToolTipService] .SetToolTip(target, tip);
}
```

Correct Answer:



Answer Area

```
private void CreateToolTip( [✓ DependencyObject | String] target, [DependencyObject | ✓ String] content)
{
    [✓ ToolTip | ToolTipService] tip = new [✓ ToolTip | ToolTipService] ();
    tip.Content = content;
    [ToolTip | ✓ ToolTipService] .SetToolTip(target, tip);
}
```

Section: (none)
Explanation

Explanation/Reference:
Explanation:

Box 1: DependencyObject
Box 2: String
Box 3: ToolTip
Box 4: ToolTip
Box 5: ToolTipService

A ToolTip must be assigned to another UI element that is its owner. In Extensible Application Markup Language (XAML), use the ToolTipService.Tooltip attached property to assign the ToolTip to an owner. In code, use the ToolTipService.SetToolTip method to assign the ToolTip to an owner.

The SetToolTip(DependencyObject, Object) method sets the value of the ToolTipService.ToolTip XAML attached property.

Reference: https://docs.microsoft.com/en-us/uwp/api/windows.ui.xaml.controls.tooltipservice#Windows_UI_Xaml_Controls_ToolTipService_ToolTipProperty
QUESTION 8

You are developing a Universal Windows Platform (UWP) app.

You need to provide a solution that moves the scroll bars of the ScrollViewer when a user rotates the mouse wheel.

Which two actions should you perform? Each correct answer presents part of the solution.

A. Evaluate the CurrentPoint.Properties.MouseWheelDelta property of the PointerEvenArgs object. Call the ChangeView() method of the ScrollViewer.
B. Update the XAML of the ScrollViewer to include the PointerWheelChanged event with a new event handler. Evaluate the Pointer.IsInRange property of the PointerRoutedEventArgs object within the event handler. Call the ChangeView() method of the ScrollViewer. C. Add an event handler to the PointerRoutedAway event for the current window.
D. Evaluate the CurrentPoint.Properties.IsHorizontalMouseWheel property of the PointerEventArgs object. Call the ChangeView() method of the ScrollViewer.
E. Add an event handler to the PointerWheelChanged event for the current window.

Correct Answer: CD
Section: (none)
Explanation

Explanation/Reference:
Explanation:
The PointerRoutedAway event occurs on the process receiving input when the pointer input is routed to another process.

Reference:
https://docs.microsoft.com/en-us/uwp/api/windows.ui.xaml.controls.scrollviewer https://docs.microsoft.com/en-us/uwp/api/windows.ui.core.corewindow#Windows_UI_Core_CoreWindow_PointerWheelChanged

QUESTION 9
You have two Universal Windows Platform (UWP) apps named Catalog and Research, respectively.

You need to create a service in the Catalog app that can be queried by the Research app.

Which three tasks should you perform? Each correct answer presents part of the solution.

A. Enter the package family name of the Catalog app in the Catalog app.
B. Add a Windows Runtime component to the Catalog app.
C. Enter the package family name of the Catalog app in the Research app.
D. Add an app service extension to package.appmanifest file in the Research app.
E. Add a Windows Runtime component to the Research app.
F. Add an app service extension to package.appmanifest file in the Catalog app.

Correct Answer: BCF
Section: (none)
Explanation

Explanation/Reference:
Explanation:
F: Example: Add an app service extension to package.appxmanifest
In the AppServiceProvider project's Package.appxmanifest file, add the following AppService extension to the <Application> element. This example advertises the com.Microsoft.Inventory service and is what identifies this app as an app service provider. The actual service will be implemented as a background task. The app service app exposes the service to other apps

B: Create the app service
An app service is implemented as a background task. This enables a foreground application to invoke an app service in another application to perform tasks behind the scenes. Add a new Windows Runtime Component project to the solution.

C: Deploy the service app and get the package family name
The app service provider app must be deployed before you can call it from a client. You will also need the package family name of the app service app in order to call it.

Reference: https://docs.microsoft.com/en-us/windows/uwp/launch-resume/how-to-create-and-consume-an-app-service

QUESTION 10
HOTSPOT

You are developing a Universal Windows Platform (UWP) app by using XAML and C#. A team member has written a XAML page that includes a button with an event handler method named ButtonSendNotification_Click() registered to the Click event.

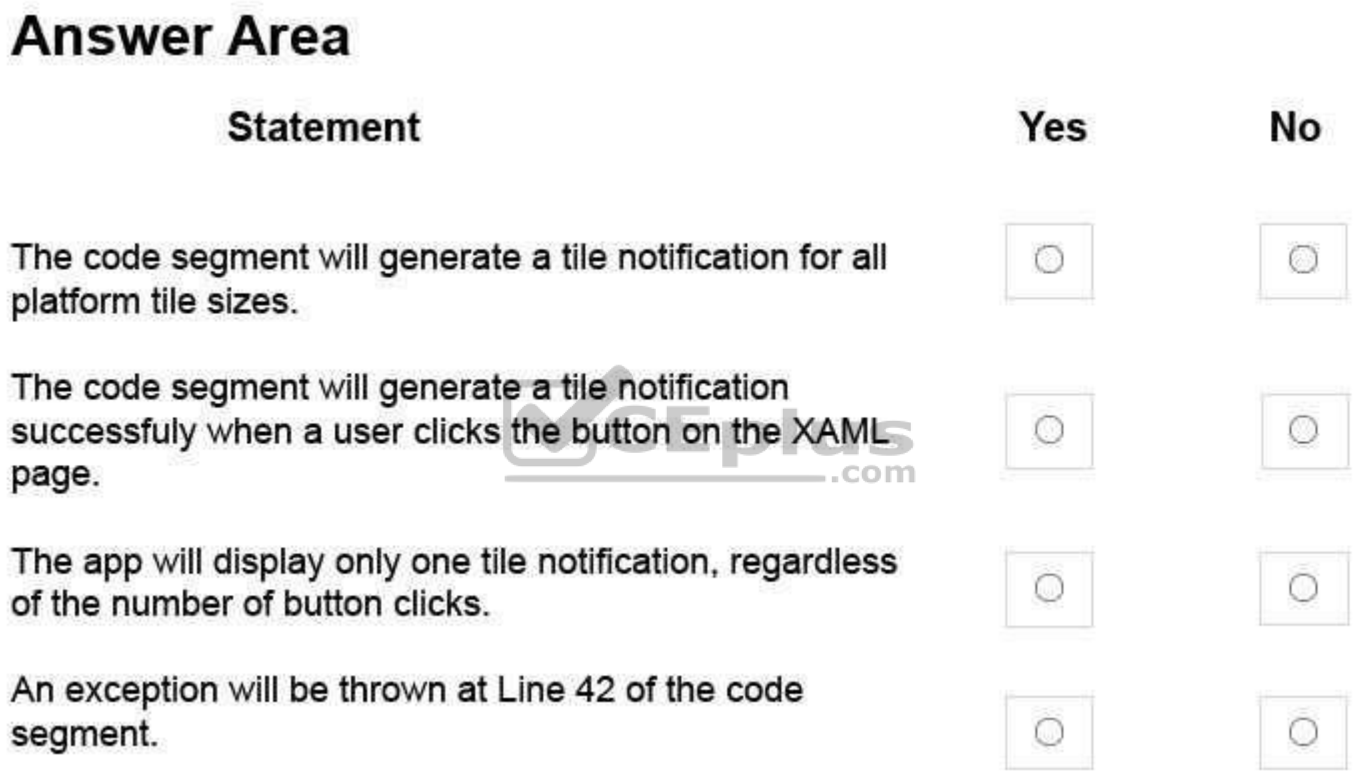
You are reviewing the following code segment written by the team member (line numbers are added for reference only):

```
public sealed partial class MainPage : Page
{
    public MainPage()
    {
        InitializeComponent();
        TileUpdateManager.CreateTileUpdaterForApplication().EnableNotificationQueue( true);
    }
    private void ButtonSendNotification_Click( object sender, RoutedEventArgs e)
    {
        SendTileNotification();
    }
    private static string GetNewsTitle()
    {
        ...
    }
    private void SendTileNotification()
    {
        TileNotification tileNotification = GenerateTileNotification();
        tileNotification.Tag = "newsItem" + GetNewsTitle();
        TileUpdateManager .CreateTileUpdaterForApplication().Update(tileNotification);
    }
    private TileNotification GenerateTileNotification()
    {
        string xml = $@"
            <tile version='3'>
                <visual branding='name'>
                    <binding template='TileMedium'>
                        <text hint-wrap='true'>This just in...</text>
                        <text hint-wrap='true' hint-style='captionSubtle'/>
                    </binding>
                    <binding template='TileWide'>
                        <text hint-wrap='true'>This just in...</text>
                        <text hint-wrap='true' hint-style='captionSubtle'/>
                    </binding>
                    <binding template='TileLarge'>
                        <text hint-wrap='true'>This just in...</text>
                        <text hint-wrap='true' hint-style='captionSubtle'/>
                    </binding>
                </visual>
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No.

NOTE: Each selection is worth one point.

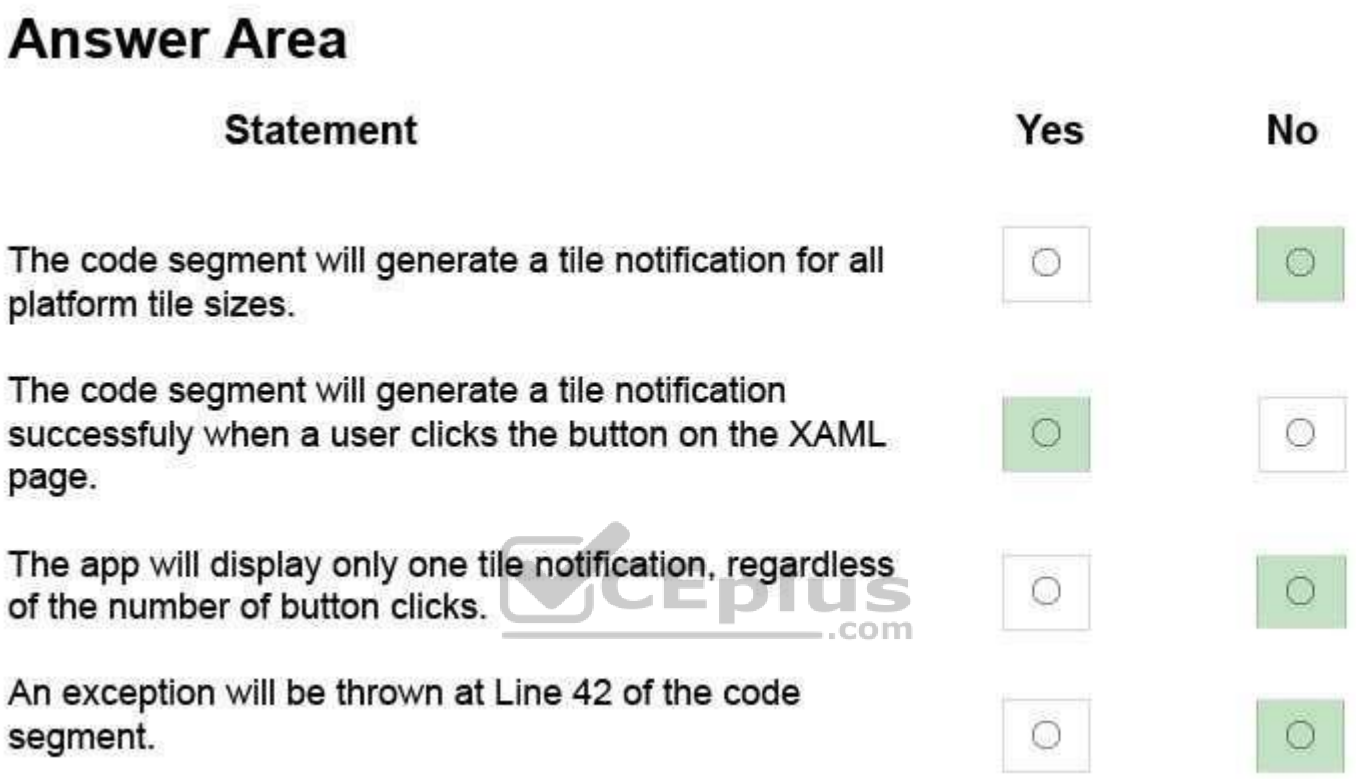
Hot Area:



Answer Area

Statement	Yes	No
The code segment will generate a tile notification for all platform tile sizes.	○	○
The code segment will generate a tile notification successfuly when a user clicks the button on the XAML page.	○	○
The app will display only one tile notification, regardless of the number of button clicks.	○	○
An exception will be thrown at Line 42 of the code segment.	○	○

Correct Answer:



Answer Area

Statement	Yes	No
The code segment will generate a tile notification for all platform tile sizes.	○	[✓ ○]
The code segment will generate a tile notification successfuly when a user clicks the button on the XAML page.	[✓ ○]	○
The app will display only one tile notification, regardless of the number of button clicks.	○	[✓ ○]
An exception will be thrown at Line 42 of the code segment.	○	[✓ ○]

Section: (none)
Explanation

Explanation/Reference:
Explanation:

Box 1: No
There are four tile sizes: small, medium, wide, large. Only three are reference in the TileNofitication definition.

Box 2: Yes

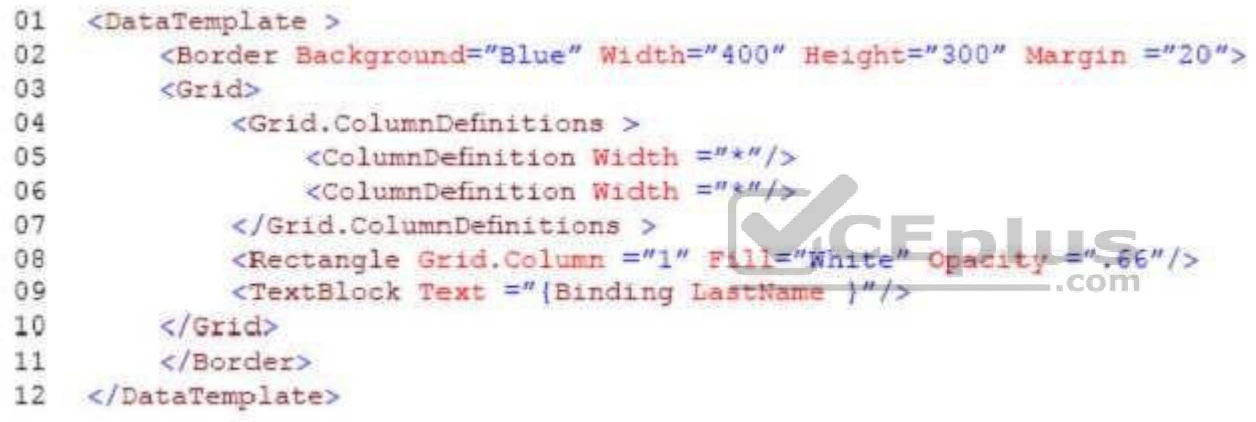
Box 3: No

Box 4: No
Line 42 is doc.LoadXml(xml);
The LoadXml method loads an XML document from a string. Returns TRUE on success or FALSE on failure.
If called statically, returns a DOMDocument or FALSE on failure.

If an empty string is passed as the source, a warning will be generated. This warning is not generated by libxml and cannot be handled using libxml's error handling functions.

Reference: https://docs.microsoft.com/en-us/windows/uwp/controls-and-patterns/tiles-and-notifications-app-assets

QUESTION 11
You have a Universal Windows Platform (UWP) app. The app has a page that includes the following XAML markup. Line numbers are included for reference only.



```
01  <DataTemplate >
02      <Border Background="Blue" Width="400" Height="300" Margin ="20">
03      <Grid>
04          <Grid.ColumnDefinitions >
05              <ColumnDefinition Width ="*"/>
06              <ColumnDefinition Width ="*"/>
07          </Grid.ColumnDefinitions >
08          <Rectangle Grid.Column ="1" Fill="White" Opacity =".66"/>
09          <TextBlock Text ="{Binding LastName }"/>
10      </Grid>
11      </Border>
12  </DataTemplate>
```

Users report that the page takes a long time to refresh.

You need to improve the load time for the page while maintaining the same layout and functionality.

What should you do?

A. Move the attributes from the BORDER element at line 02 to the GRID element at line 03. Then, remove the BORDER elements at line 02 and line 11.
B. Replace the TEXTBLOCK element at line 09 with a TEXTBOX element.
C. Swap the markup at line 02 with the markup at line 03. Swap the markup at line 10 with the markup at line 11.
D. Move the Fill and Opacity attributes and value from the RECTANGLE element at line 08 to the GRID element at line 03. Then, Remove the RECTANGLE element.

Correct Answer: D

Section: (none)
Explanation

Explanation/Reference:
Explanation:
Use single-cell grids for overlapping UI
A common UI requirement is to have a layout where elements overlap each other. Typically padding, margins, alignments, and transforms are used to position the elements this way. The XAML Grid control is optimized to improve layout performance for elements that overlap.

Reference: https://docs.microsoft.com/en-us/windows/uwp/debug-test-perf/optimize-your-xaml-layout

QUESTION 12
You must create a control that meets the following requirements: ▪ allows you to extend the behavior of a combo box ▪ allows the arrow image that is located at the right edge of a standard control to be replaced with a new image ▪ has a property that sets and returns the image
▪ maintains all of the properties of a standard combo box control ▪
has a visual interface of the control that is defined by using XAML ▪
defines the properties for the control in code

You need to create the control.

Which object should you use?

A. ContentDialog
B. StaticResource
C. ThemeResource
D. UserControl

Correct Answer: A
Section: (none)
Explanation

Explanation/Reference:
Explanation:
ContentDialog represents a dialog box that can be customized to contain checkboxes, hyperlinks, buttons and any other XAML content.

Reference: https://docs.microsoft.com/en-us/uwp/api/windows.ui.xaml.controls.contentdialog
QUESTION 13
HOTSPOT

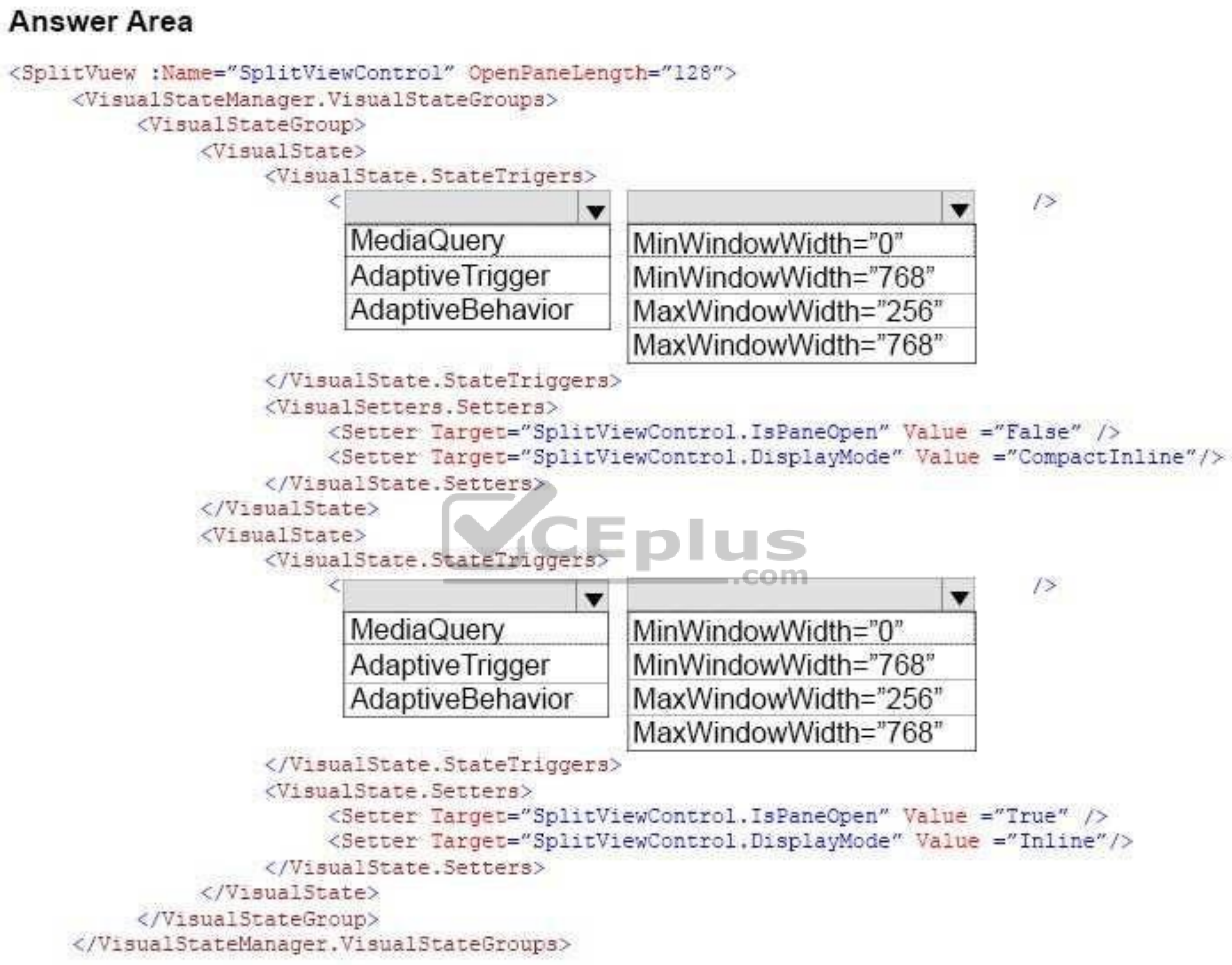
You are developing a Universal Windows Platform (UWP) app.

The app does not display content properly on mobile devices.

You need to support smaller window sizes.

How should you complete the relevant XAML markup? To answer, select the appropriate markup segment from each list in the answer area.

Hot Area:



Answer Area

```
<SplitVuew :Name="SplitViewControl" OpenPaneLength="128">
     <VisualStateManager.VisualStateGroups>
          <VisualStateGroup>
               <VisualState>
                    <VisualState.StateTrigers>
                         <  [▼]  [▼]  />
                    </VisualState.StateTriggers>
                    <VisualSetters.Setters>
                         <Setter Target="SplitViewControl.IsPaneOpen" Value ="False" />
                         <Setter Target="SplitViewControl.DisplayMode" Value ="CompactInline"/>
                    </VisualState.Setters>
               </VisualState>
               <VisualState>
                    <VisualState.StateTriggers>
                         <  [▼]  [▼]  />
                    </VisualState.StateTriggers>
                    <VisualState.Setters>
                         <Setter Target="SplitViewControl.IsPaneOpen" Value ="True" />
                         <Setter Target="SplitViewControl.DisplayMode" Value ="Inline"/>
                    </VisualState.Setters>
               </VisualState>
          </VisualStateGroup>
     </VisualStateManager.VisualStateGroups>
```

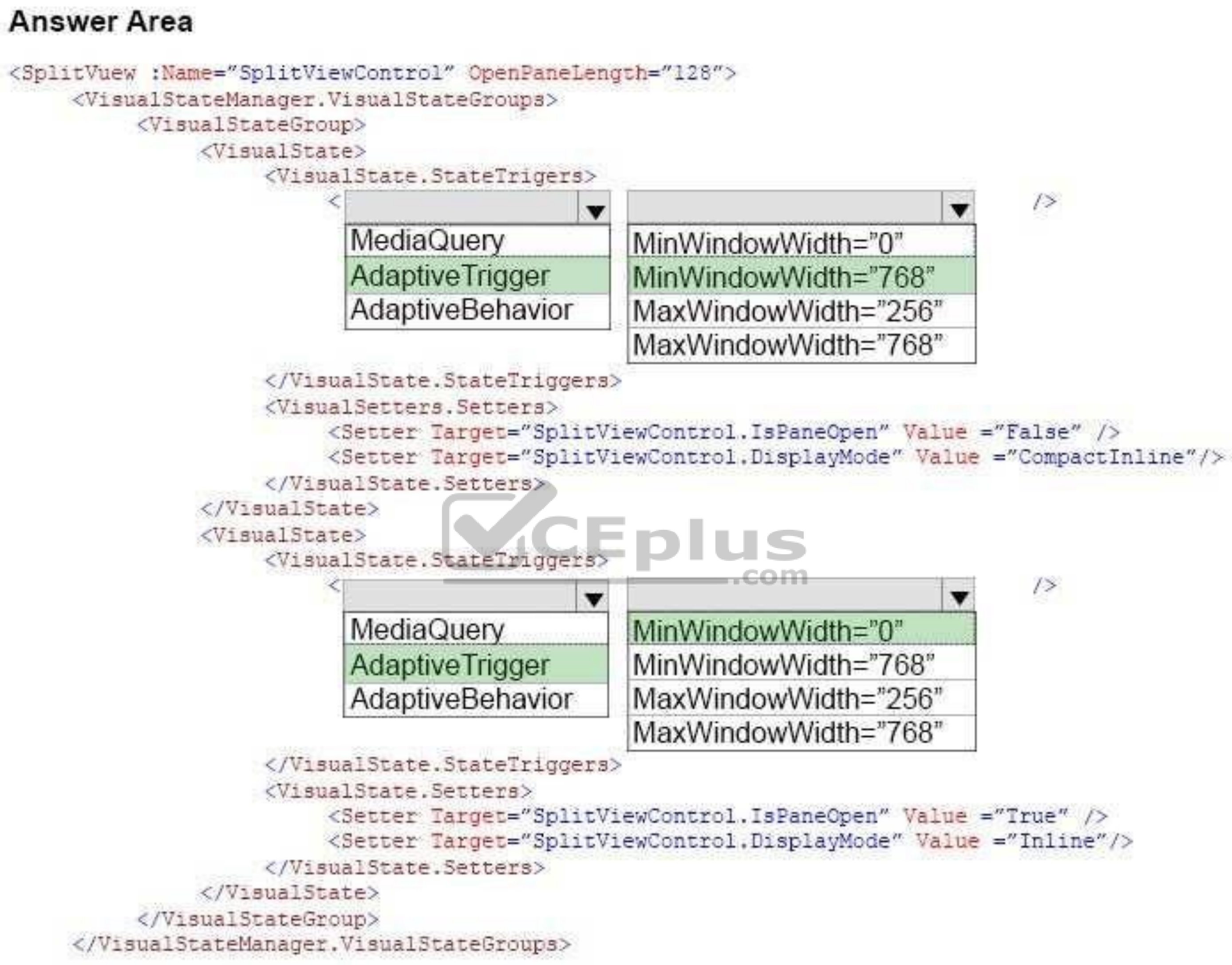
First trigger, first dropdown:
- MediaQuery
- AdaptiveTrigger
- AdaptiveBehavior

First trigger, second dropdown:
- MinWindowWidth="0"
- MinWindowWidth="768"
- MaxWindowWidth="256"
- MaxWindowWidth="768"

Second trigger, first dropdown:
- MediaQuery
- AdaptiveTrigger
- AdaptiveBehavior

Second trigger, second dropdown:
- MinWindowWidth="0"
- MinWindowWidth="768"
- MaxWindowWidth="256"
- MaxWindowWidth="768"

Correct Answer:



Answer Area

```
<SplitVuew :Name="SplitViewControl" OpenPaneLength="128">
     <VisualStateManager.VisualStateGroups>
          <VisualStateGroup>
               <VisualState>
                    <VisualState.StateTrigers>
                         <                                                  />
```

MediaQuery	MinWindowWidth="0"
[✓ AdaptiveTrigger]	[✓ MinWindowWidth="768"]
AdaptiveBehavior	MaxWindowWidth="256"
	MaxWindowWidth="768"

```
                    </VisualState.StateTriggers>
                    <VisualSetters.Setters>
                         <Setter Target="SplitViewControl.IsPaneOpen" Value ="False" />
                         <Setter Target="SplitViewControl.DisplayMode" Value ="CompactInline"/>
                    </VisualState.Setters>
               </VisualState>
               <VisualState>
                    <VisualState.StateTriggers>
                         <                                                  />
```

MediaQuery	[✓ MinWindowWidth="0"]
[✓ AdaptiveTrigger]	MinWindowWidth="768"
AdaptiveBehavior	MaxWindowWidth="256"
	MaxWindowWidth="768"

```
                    </VisualState.StateTriggers>
                    <VisualState.Setters>
                         <Setter Target="SplitViewControl.IsPaneOpen" Value ="True" />
                         <Setter Target="SplitViewControl.DisplayMode" Value ="Inline"/>
                    </VisualState.Setters>
               </VisualState>
          </VisualStateGroup>
     </VisualStateManager.VisualStateGroups>
```

Section: (none)
Explanation

Explanation/Reference:
Explanation:

One of the tools that Microsoft gives us for building adaptive UIs in UWP apps is state triggers. The version of Windows 10 released at BUILD 2015 features one state trigger: a class named AdaptiveTrigger. AdaptiveTrigger has two important properties: MinWindowWidth and MinWindowHeight. You use AdaptiveTrigger in conjunction with Visual State Manager to adapt the UI to screens and windows of various sizes.

* Inline
The pane is always visible and doesn't overlay the content area. The pane and content areas divide the available screen real estate.

* CompactInline
A narrow portion of the pane is always visible in this mode, which is just wide enough to show icons. The default closed pane width is 48px, which can be modified with CompactPaneLength. If the pane is opened, it will reduce the space available for content, pushing the content out of its way.

Reference: http://www.wintellect.com/devcenter/jprosise/using-adaptivetrigger-to-build-adaptive-uis-in-windows-10

QUESTION 14
You have to connect your app to an online identity provider that uses OAuth authentication protocol.

The app must securely use the WebAuthenticationBroker object for authentication.

You need to ensure that the app registers with the provider.

Which two actions should you perform? Each correct answer presents part of the solution.

A. Construct a HTTP request URI.
B. Call the GetCurrentApplicationCallbackUri method.
C. Call the AuthenticateAsync method.
D. Construct a HTTPS request URI.

Correct Answer: CD
Section: (none)
Explanation

Explanation/Reference:
Explanation:
The current application callback URI is used as an implicit value of the callbackUri parameter of the AuthenticateAsync method. However, applications need the URI value to add it to the request URI as required by the online provider.

The requestUri parameter must be a HTTPS address: an exception will be thrown if an HTTP address is used, even for local testing scenarios.

Reference: https://docs.microsoft.com/en-us/uwp/api/windows.security.authentication.web.webauthenticationbroker

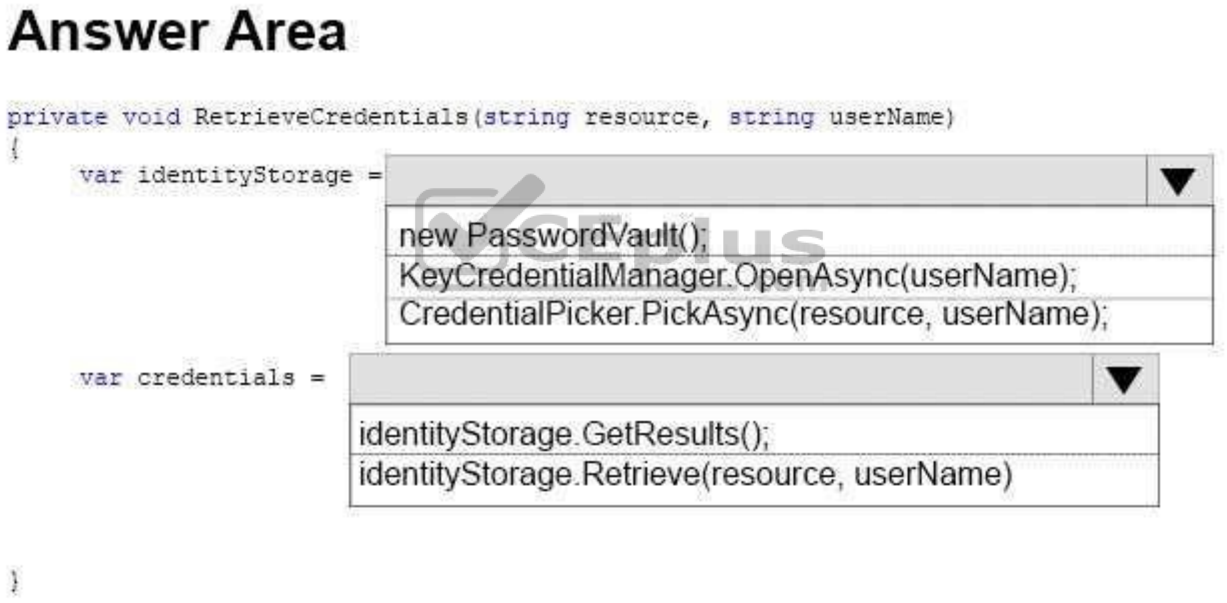
QUESTION 15
HOTSPOT

You are developing a Universal Windows Platform (UWP) app that stores credentials by using the Credential Locker service.

You need to securely retrieve credentials for the current user.

How should you complete the method? To answer, select the appropriate code segment from each list in the answer area.

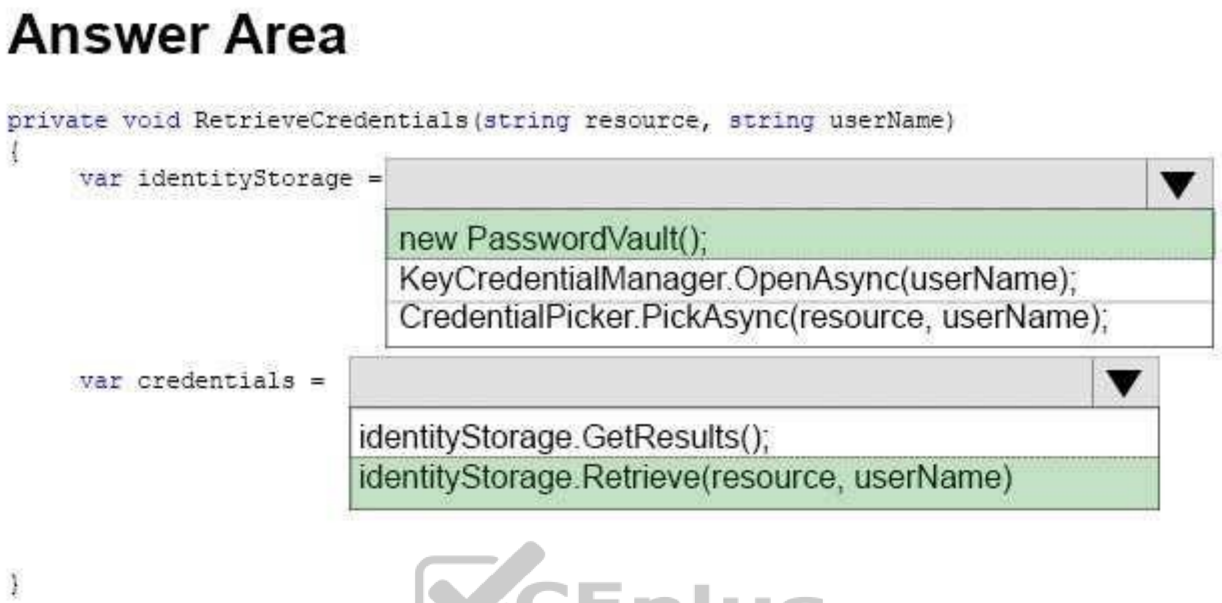
Hot Area:



Answer Area

```
private void RetrieveCredentials(string resource, string userName)
{
    var identityStorage = [▼]
                          new PasswordVault();
                          KeyCredentialManager.OpenAsync(userName);
                          CredentialPicker.PickAsync(resource, userName);

    var credentials = [▼]
                      identityStorage.GetResults();
                      identityStorage.Retrieve(resource, userName)

}
```

Correct Answer:



Answer Area

```
private void RetrieveCredentials(string resource, string userName)
{
    var identityStorage =
```

[✓ new PasswordVault();]
KeyCredentialManager.OpenAsync(userName);
CredentialPicker.PickAsync(resource, userName);

```
    var credentials =
```

identityStorage.GetResults();
[✓ identityStorage.Retrieve(resource, userName)]

```
}
```

Section: (none)
Explanation

Explanation/Reference:
Explanation:

Box 1:
Example:
var vault = new Windows.Security.Credentials.PasswordVault();

Box 2:
Example continued:
// When there are multiple usernames,
// retrieve the default username. If one doesn't
// exist, then display UI to have the user select
// a default username.

defaultUserName = GetDefaultUserNameUI();
credential = vault.Retrieve(resourceName, defaultUserName);

Reference: https://docs.microsoft.com/en-us/windows/uwp/security/credential-locker

QUESTION 16
You are designing a roadside assistance mobile app. The app displays a persistent list of links to pages. The pages provide a quick way to move between different views of the app.

You need to recommend a user interface pattern that meets the following requirements:
▪ Allow users to navigate to frequently accessed, distinct content categories.
▪ Provide two or more content panes that have corresponding category headers.
▪ Display the navigation controls on the top of the screen. ▪
Highlight the currently selected navigation control.

Which pattern should you recommend?

A. hub
B. tabs and pivots
C. active canvas
D. master/details

Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:
Explanation:
The Pivot control and related tabs pattern are used for navigating frequently accessed, distinct content categories. Pivots allow for navigation between two or more content panes and relies on text headers to articulate the different sections of content.
Tabs are a visual variant of Pivot that use a combination of icons and text or just icons to articulate section content. Tabs are built using the Pivot control.

Reference: https://docs.microsoft.com/en-us/windows/uwp/controls-and-patterns/tabs-pivot

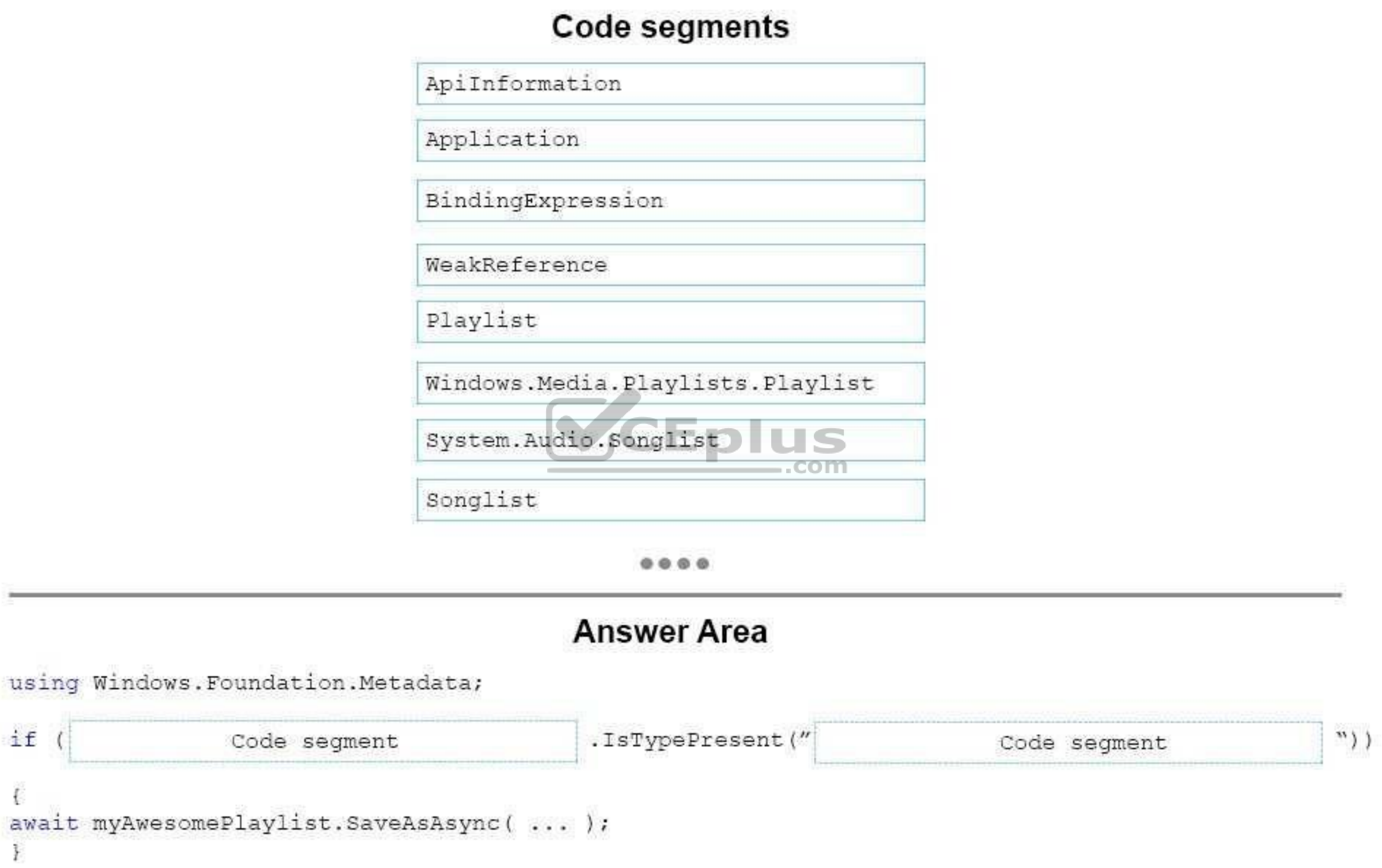
QUESTION 17
DRAG DROP

You are developing a Universal Windows Platform (UWP) app. The app runs on multiple device families, including desktop, Windows Phone, and Xbox.

The app must be able to access a user's media playlists if the device supports this feature. If the device does not support this feature, the app must continue to function.
You need to detect whether a device supports accessing user playlists.

How should you complete the relevant code? To answers, drag the appropriate code segment to the correct location or locations. Each code segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

Select and Place:



Code segments

- `ApiInformation`
- `Application`
- `BindingExpression`
- `WeakReference`
- `Playlist`
- `Windows.Media.Playlists.Playlist`
- `System.Audio.Songlist`
- `Songlist`

Answer Area

```
using Windows.Foundation.Metadata;

if ( [ Code segment ] .IsTypePresent(" [ Code segment ] "))

{
await myAwesomePlaylist.SaveAsAsync( ... );
}
```

Correct Answer:

Code segments

```
Application
```

```
BindingExpression
```

```
WeakReference
```

```
Playlist
```

```
System.Audio.Songlist
```

```
Songlist
```

Answer Area

```
using Windows.Foundation.Metadata;

if ( ApiInformation .IsTypePresent(" Windows.Media.Playlists.Playlist "))

{
await myAwesomePlaylist.SaveAsAsync( ... );
}
```

Section: (none)
Explanation

Explanation/Reference:
Explanation:

Box 1: ApiInformation

Box 2:
Windows.Media.Playlists.Playlist
example: using
Windows.Foundation.Metadata;

```
if(ApiInformation.IsTypePresent("Windows.Media.Playlists.Playlist"))
{
await myAwesomePlaylist.SaveAsAsync( ... );
}
```

This code makes a runtime check for the presence of the Playlist class, then statically references and calls the SaveAsAsync method on the class.

Reference: https://blogs.windows.com/buildingapps/2015/09/15/dynamically-detecting-features-with-api-contracts-10-by-10/

QUESTION 18
You are developing a Universal Windows Platform (UWP) app that allows users to take photos and record videos.

The photos and videos must be stored in the user's Photos library and Videos library, respectively. The app must not display a user interface for saving files.

You need to configure the app.

Which set of capabilities should you declare in the app manifest?

A. Internet (client), microphone, location and proximity
B. webcam, microphone, Pictures library and Video library
C. Internet (client), Documents library, Videos library, and proximity
D. webcam, location, proximity and Pictures library

Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:
Explanation:
You must specify the webcam or microphone capabilities in your app manifest file if you are using MediaCapture to capture audio, photos, or video programmatically.

Reference: https://docs.microsoft.com/en-us/windows/uwp/audio-video-camera/capture-photos-and-video-with-cameracaptureui

QUESTION 19
You are developing a Universal Windows Platform (UWP) app.

The app must be available on Windows Phone, Windows tablet devices, and Xbox.

When the app is running on a device, you need to determine which members of a specific class you can use.

Which of the following methods should you use?

A. ApiInformation.IsPropertyPresent
B. UserInformation.NameAccessAllowed
C. Selector.GetIsSelectionActive
D. AppExtensionCatalog.FindAllAsync

Correct Answer: D
Section: (none)
Explanation

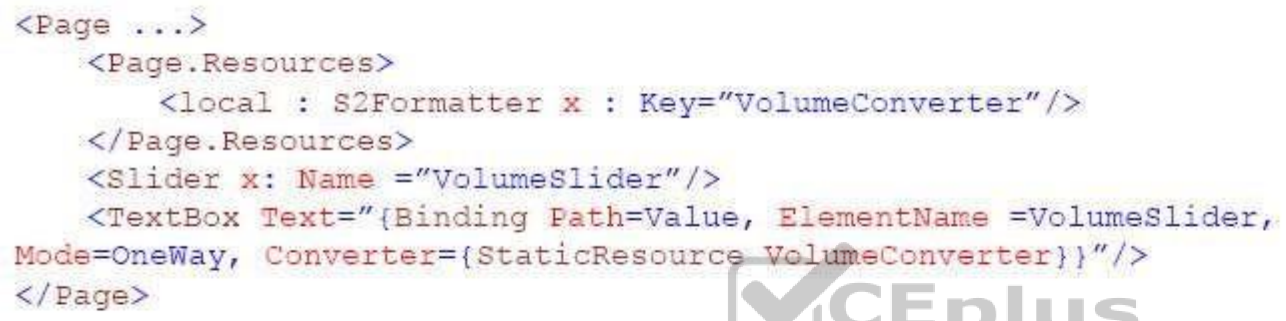
Explanation/Reference:
Explanation:
The AppExtensionCatalog class represents a device. This class allows access to well-known device properties as well as additional properties specified during device enumeration.
A Successful completion of FindAllAsync results in a DeviceInformationCollection containing DeviceInformation objects.

Reference: https://docs.microsoft.com/en-us/uwp/api/windows.devices.enumeration.deviceinformation

QUESTION 20
HOTSPOT
You are developing a Universal Windows Platform (UWP) app that plays audio recordings.

You are creating a page where the user can set a volume level for the app using a slider control. You need to display the volume level in a TextBox right below the slider. You have C# class named VolumeConverter that converts slider values to a number.

You have a page that includes the following markup:

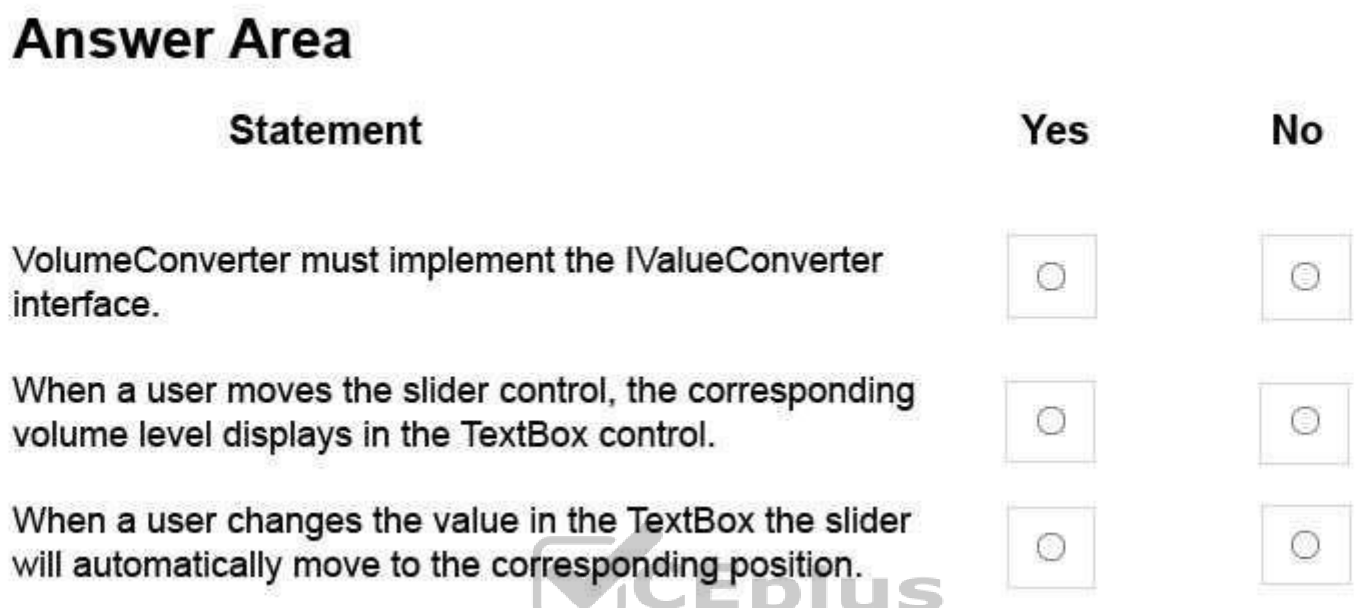


```
<Page ...>
    <Page.Resources>
        <local : S2Formatter x : Key="VolumeConverter"/>
    </Page.Resources>
    <Slider x: Name ="VolumeSlider"/>
    <TextBox Text="{Binding Path=Value, ElementName =VolumeSlider,
Mode=OneWay, Converter={StaticResource VolumeConverter}}"/>
</Page>
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No.

NOTE: Each correct selection is worth one point.

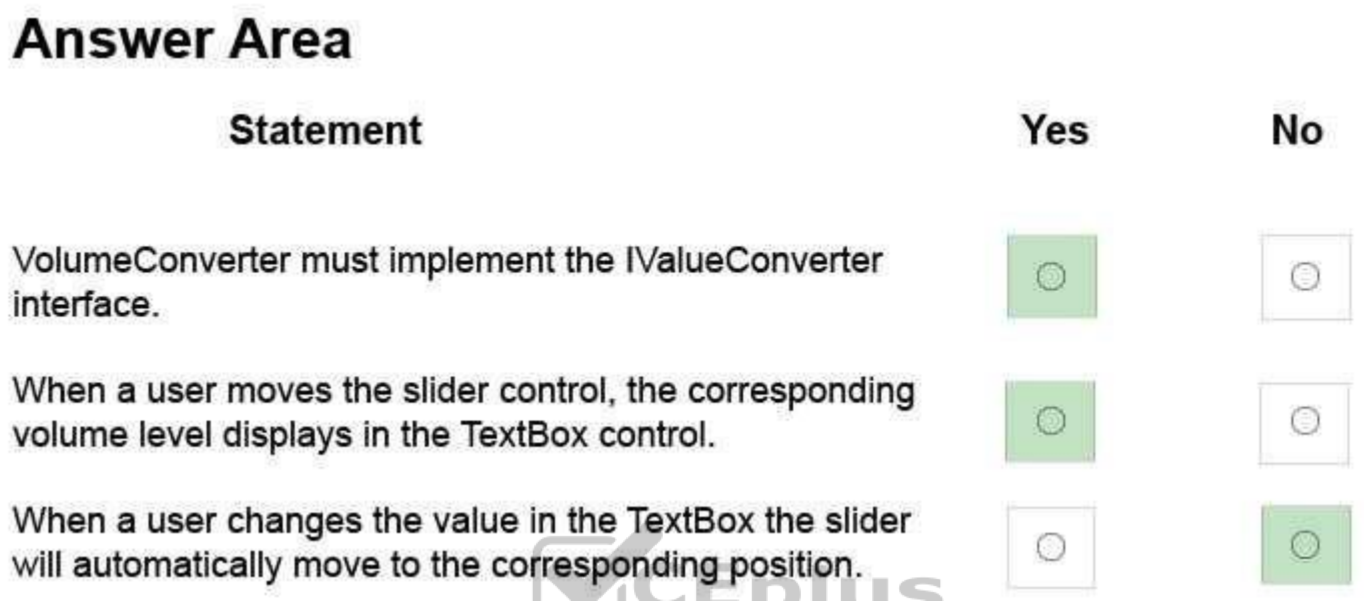
Hot Area:



Answer Area

Statement	Yes	No
VolumeConverter must implement the IValueConverter interface.	○	○
When a user moves the slider control, the corresponding volume level displays in the TextBox control.	○	○
When a user changes the value in the TextBox the slider will automatically move to the corresponding position.	○	○

Correct Answer:



Answer Area

Statement	Yes	No
VolumeConverter must implement the IValueConverter interface.	🟩 ✓	○
When a user moves the slider control, the corresponding volume level displays in the TextBox control.	🟩 ✓	○
When a user changes the value in the TextBox the slider will automatically move to the corresponding position.	○	🟩 ✓

Section: (none)
Explanation

Explanation/Reference:
Explanation:

Box 1: Yes
Box 2: Yes
Box 3: No

Reference: https://docs.microsoft.com/en-us/uwp/api/windows.ui.xaml.controls.slider

QUESTION 21
You are developing a Universal Windows Platform (UWP) app that uses XAML and C#. The app must use the Model-View-ViewModel (MVVM) pattern.

The user interface (UI) triggers an event.

You need to bind the event to a view model method.

What should you do?

A. Create a custom behavior and attach the behavior to the UI element. Bind the behavior's event trigger to the command declared in the view model.
B. Create an attached property of type ICommand. Bind the UI element's event to the attached property.
C. Assign the value of the DataContext property to the view model. Use the BindingExression.UpdateSource() method to update the data source.
D. Add a strongly-typed view model property to the view. In the code behind file for the view, invoke the view model method.

Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:
Explanation:
Commands are an implementation of the ICommand interface that is part of the .NET Framework. This interface is used a lot in MVVM applications.

Reference: https://msdn.microsoft.com/en-us/magazine/dn237302.aspx

QUESTION 22
You are developing a Universal Windows Platform (UWP) app. The app must allow the user to select only one file at a time.

You need to ensure that the app displays the appropriate dialog window.

Which method should you use?

A. FileOpenPicker.PickSingleFileAsync()
B. FileOpenPicker.PickMultipleFilesAsync()
C. StorageItem.OpenSequentialReadAsync()
D. StorageItem.GetFileFromPathAsync()
E. StorageItem.OpenReadAsync()

Correct Answer: A
Section: (none)
Explanation

Explanation/Reference:
Explanation:
To pick a single file.
Windows.Storage.StorageFile file = await picker.PickSingleFileAsync();
if (file != null)

```
{
// Application now has read/write access to the picked file
this.textBlock.Text = "Picked photo: " + file.Name;
}
else
{
this.textBlock.Text = "Operation cancelled.";
}
```

Reference: https://docs.microsoft.com/en-us/windows/uwp/files/quickstart-using-file-and-folder-pickers#pick-a-single-file-complete-code-listing

QUESTION 23

Case Study

This is a case study. Case studies are not limited separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.

To start the case study

To display the first question on this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Background

Fabrikam is a commercial bank. The primary customers are individuals and employers with up to 10,000 employees. Fabrikam provides Internet banking services to customers.

You are developing a Universal Windows Platform (UWP) app for Fabrikam that extends the Internet banking to a UWP app.

Business Requirements

Core functionality

Users must be able to access accounts, view balances, view recent transactions, and deposit checks by using the UWP app.

Usability

The app must use industry proven design patterns across the app. All navigational elements must be visible at all times.

Security

The app must provide secure transactions to protect customer privacy.

Technical Requirements

Data

The app must use a file based database. You must use a code first entity framework approach.

User interface

- You must use a Model-View-ViewModel (MVVM) pattern.
- Users must be able to access all content through the top-level navigation after they sign in.
- The app must allow the user to upload up to 50 images (front and back) of checks to deposit.
- During the upload process, the app must be responsive to any other user actions.
- The app must only upload images when no other pending inputs are in the queue.

You must implement the following pages:

Page	Description
Sign-In	This page displays when the app is launched. It prompts users to enter credentials.
Transactions	This page allows users to view transactions for the last 30 days.
Balances	This page allows users to view current balance amount for all accounts.
Deposit	This page allows users to deposit checks by uploading images of checks.
Statements	This page lists the available bank statements for all accounts.

Network and web service

The app must meet the following requirements related to networking and web services:
- Connect to a web service over a secure HTTP connection to upload images.
- Connect to Fabrikam's core web service to retrieve account information.
- Use networking technology already available in the .Net Framework.
- Consume the JSON that the Fabrikam core web service provides.

User data and alerts

The app must meet the following requirements related to user data and alerts:
▪ Download new monthly bank statements when possible. The download process must not affect the performance of the app. ▪ Report to the user when the statements are downloaded to the device.
▪ Write a log entry when statement downloads are not successful.
▪ Periodically check for user activity and automatically log the user out when there is no activity for more than 15 minutes.

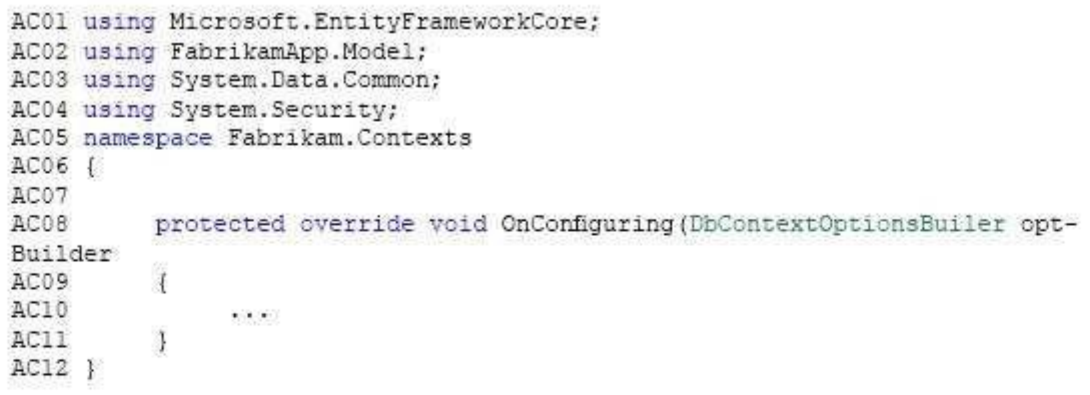
Security

The app must meet the following requirements related to security:
▪ Use a multi-factor authentication (MFA) by using email and a verification code to identify the user.
▪ Securely store credentials and retrieve credentials.
▪ Automatically sign in the user irrespective of the device that is used to sign in to the app.
▪ Store the resource name within the app itself.
▪ Connect to an authentication app by using the URI schema fabrikam-security://oauth/.

Application Structure

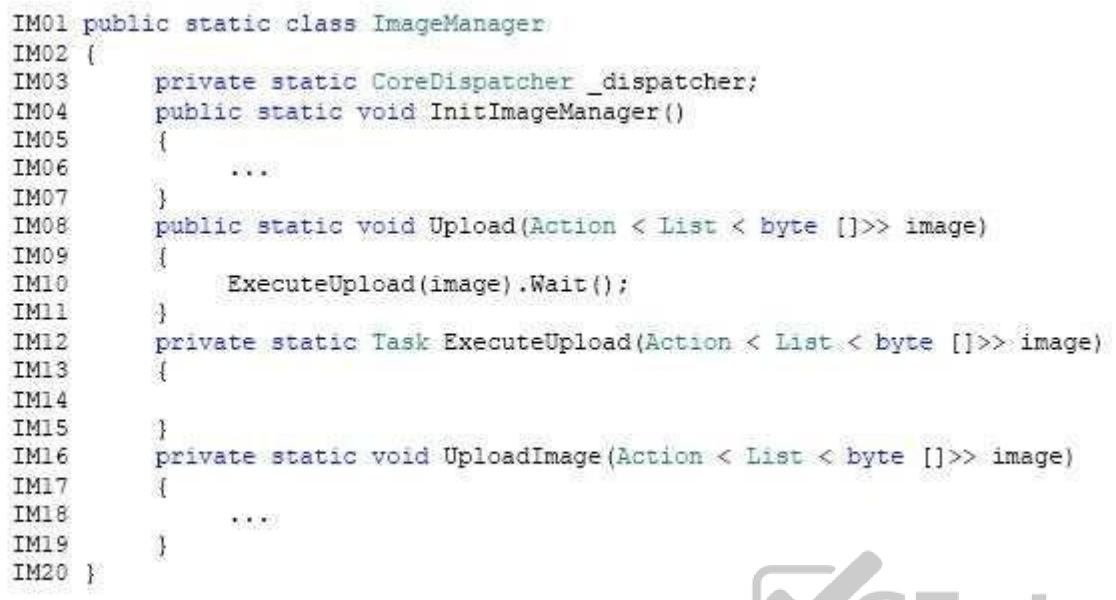
AccountContext.cs

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
AC01 using Microsoft.EntityFrameworkCore;
AC02 using FabrikamApp.Model;
AC03 using System.Data.Common;
AC04 using System.Security;
AC05 namespace Fabrikam.Contexts
AC06 {
AC07
AC08      protected override void OnConfiguring(DbContextOptionsBuiler opt-
Builder
AC09      {
AC10           ...
AC11      }
AC12 }
```

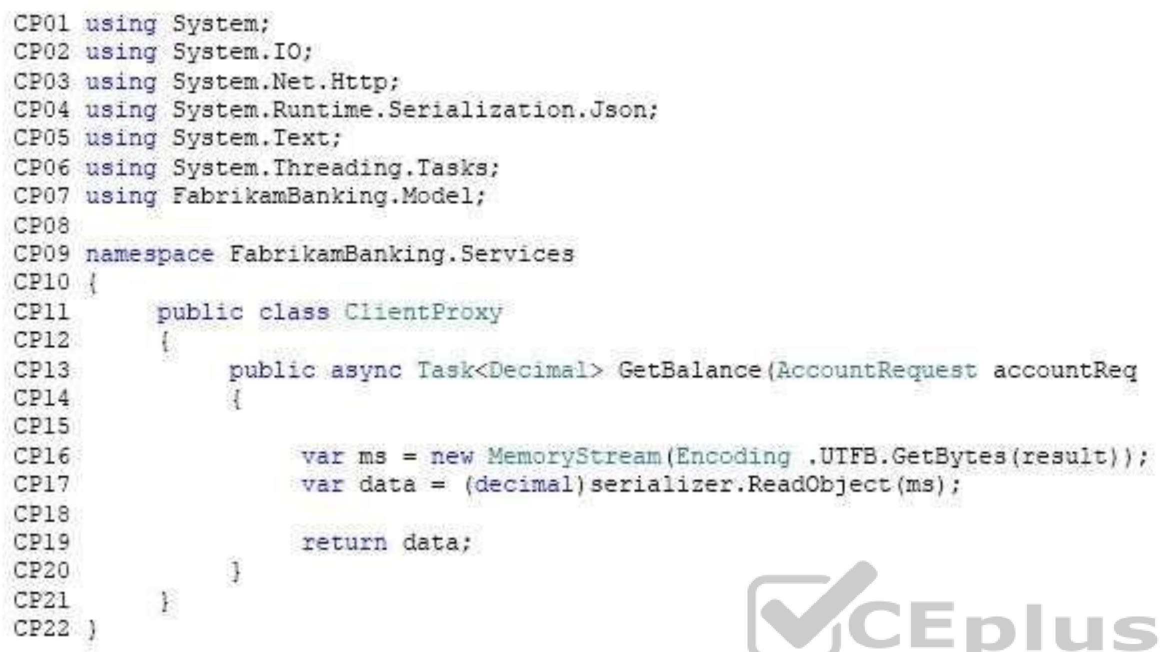
ImageManager.cs
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
IM01 public static class ImageManager
IM02 {
IM03      private static CoreDispatcher _dispatcher;
IM04      public static void InitImageManager()
IM05      {
IM06           ...
IM07      }
IM08      public static void Upload(Action < List < byte []>> image)
IM09      {
IM10           ExecuteUpload(image).Wait();
IM11      }
IM12      private static Task ExecuteUpload(Action < List < byte []>> image)
IM13      {
IM14
IM15      }
IM16      private static void UploadImage(Action < List < byte []>> image)
IM17      {
IM18           ...
IM19      }
IM20 }
```

ClientProxy.cs

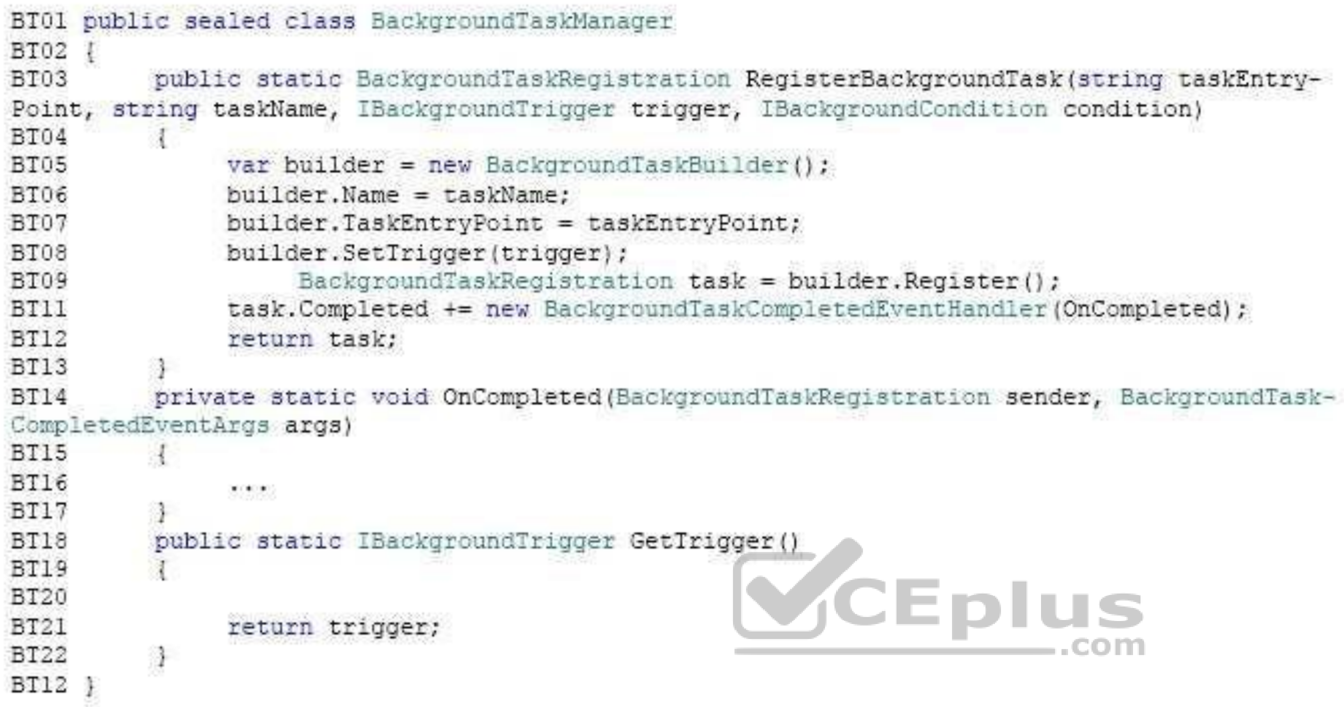
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
CP01 using System;
CP02 using System.IO;
CP03 using System.Net.Http;
CP04 using System.Runtime.Serialization.Json;
CP05 using System.Text;
CP06 using System.Threading.Tasks;
CP07 using FabrikamBanking.Model;
CP08
CP09 namespace FabrikamBanking.Services
CP10 {
CP11      public class ClientProxy
CP12      {
CP13           public async Task<Decimal> GetBalance(AccountRequest accountReq
CP14           {
CP15
CP16                var ms = new MemoryStream(Encoding .UTF8.GetBytes(result));
CP17                var data = (decimal)serializer.ReadObject(ms);
CP18
CP19                return data;
CP20           }
CP21      }
CP22 }
```

BkgTaskMgr.cs

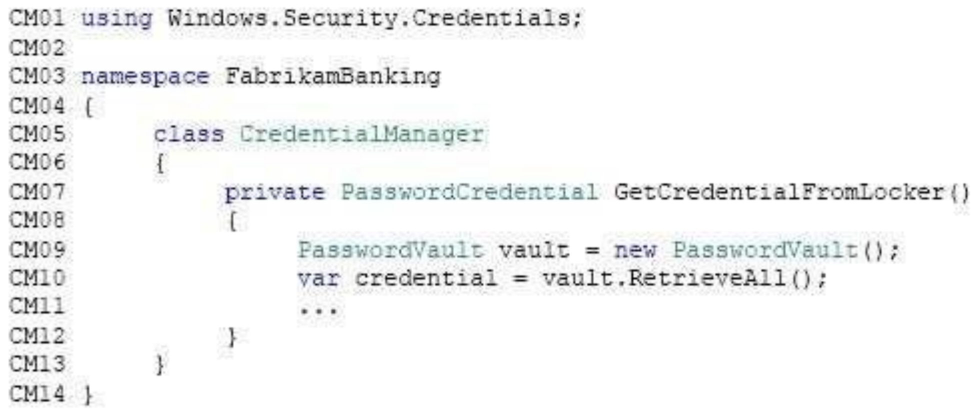
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
BT01 public sealed class BackgroundTaskManager
BT02 {
BT03      public static BackgroundTaskRegistration RegisterBackgroundTask(string taskEntry-
Point, string taskName, IBackgroundTrigger trigger, IBackgroundCondition condition)
BT04      {
BT05           var builder = new BackgroundTaskBuilder();
BT06           builder.Name = taskName;
BT07           builder.TaskEntryPoint = taskEntryPoint;
BT08           builder.SetTrigger(trigger);
BT09                BackgroundTaskRegistration task = builder.Register();
BT11           task.Completed += new BackgroundTaskCompletedEventHandler(OnCompleted);
BT12           return task;
BT13      }
BT14      private static void OnCompleted(BackgroundTaskRegistration sender, BackgroundTask-
CompletedEventArgs args)
BT15      {
BT16           ...
BT17      }
BT18      public static IBackgroundTrigger GetTrigger()
BT19      {
BT20
BT21           return trigger;
BT22      }
BT12 }
```

CredentialManager.cs

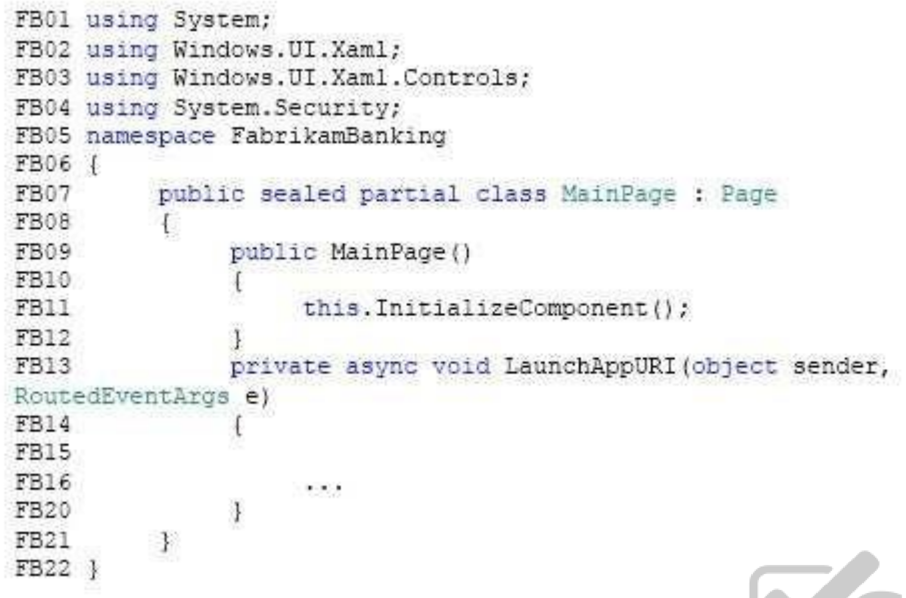
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
CM01 using Windows.Security.Credentials;
CM02
CM03 namespace FabrikamBanking
CM04 {
CM05       class CredentialManager
CM06       {
CM07             private PasswordCredential GetCredentialFromLocker()
CM08             {
CM09                   PasswordVault vault = new PasswordVault();
CM10                   var credential = vault.RetrieveAll();
CM11                   ...
CM12             }
CM13       }
CM14 }
```

MainPage.cs

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
using System;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using System.Security;
namespace FabrikamBanking
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }
        private async void LaunchAppURI(object sender, RoutedEventArgs e)
        {

            ...
        }
    }
}
```

You need to configure authentication for the app.

Which two technologies should you use? Each correct answer presents part of the solution.

A. Windows Hello
B. Windows Kerberos
C. Azure Active Directory
D. Microsoft Passport

Correct Answer: AD
Section: (none)
Explanation

Explanation/Reference:
Explanation:
Microsoft Hello

Microsoft Hello provides simple multi-factor authentication using facial recognition (or iris, or fingerprints) that is used to access the Microsoft Passport private key stored in the secure TPM chip. For the first time, Microsoft has included the biometric software (middleware) in Windows 10 to support biometrics for authentication.
In previous versions of Windows, the OEM (HP, Dell, Lenovo, etc) needed to add its own biometric middleware to support biometric authentication.

From scenario: The app must meet the following requirements related to security:
- Use a multi-factor authentication (MFA) by using email and a verification code to identify the user.
- Securely store credentials and retrieve credentials.
- Automatically sign in the user irrespective of the device that is used to sign in to the app.
- Store the resource name within the app itself.
- Connect to an authentication app by using the URI schema fabrikam-security://oauth/.

Note: Microsoft Passport
Microsoft has resurrected the Passport moniker for a new PKI credential system that requires multi-factor authentication. Most interesting about Microsoft Passport is that it fully supports the Fast IDentity Online (FIDO) Alliance standards which means it will work with many web/cloud services without modification. The plan is that users of cloud services supporting FIDO is that there will no longer be passwords associated with the user's account.
Microsoft Passport involves a user logging onto the Windows 10 computer with multi-factor (PIN, face, iris, fingerprint, etc) and either creating a new account or associating an existing account with an IDentity Provider (IDP). Windows generates a public/private key pair with the private key stored securely outside of the Windows 10 OS. The public key is associated with the account so that a challenge can be sent that can only correctly respond to the IDP. Another key point to the Microsoft Passport credential system is that the user needs to enroll every device used to access the service (IDP). Reference:
https://adsecurity.org/?p=1535

QUESTION 24
DRAG DROP

Case Study
This is a case study. Case studies are not limited separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.
To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.
At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.
To start the case study
To display the first question on this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Background

Fabrikam is a commercial bank. The primary customers are individuals and employers with up to 10,000 employees. Fabrikam provides Internet banking services to customers.
You are developing a Universal Windows Platform (UWP) app for Fabrikam that extends the Internet banking to a UWP app.

Business Requirements

Core functionality

Users must be able to access accounts, view balances, view recent transactions, and deposit checks by using the UWP app.

Usability

The app must use industry proven design patterns across the app. All navigational elements must be visible at all times.

Security

The app must provide secure transactions to protect customer privacy.

Technical Requirements

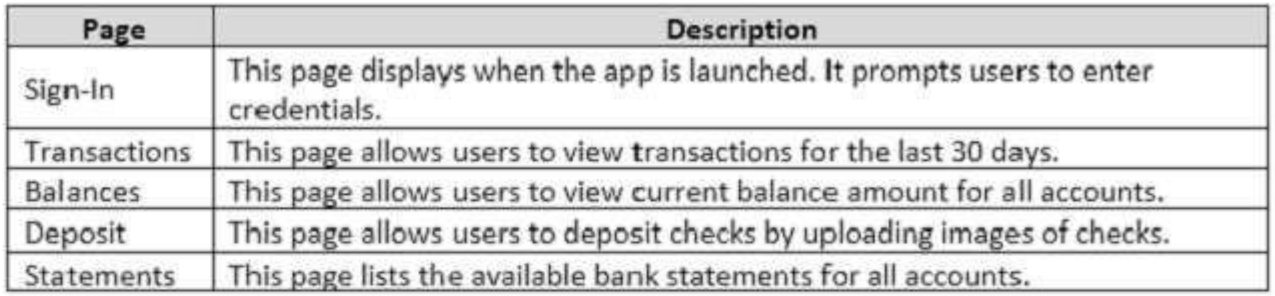
Data

The app must use a file based database. You must use a code first entity framework approach.

User interface

- You must use a Model-View-ViewModel (MVVM) pattern.
- Users must be able to access all content through the top-level navigation after they sign in.
- The app must allow the user to upload up to 50 images (front and back) of checks to deposit.
- During the upload process, the app must be responsive to any other user actions.
- The app must only upload images when no other pending inputs are in the queue.

You must implement the following pages:



Page	Description
Sign-In	This page displays when the app is launched. It prompts users to enter credentials.
Transactions	This page allows users to view transactions for the last 30 days.
Balances	This page allows users to view current balance amount for all accounts.
Deposit	This page allows users to deposit checks by uploading images of checks.
Statements	This page lists the available bank statements for all accounts.

Network and web service

The app must meet the following requirements related to networking and web services:
▪ Connect to a web service over a secure HTTP connection to upload images.
▪ Connect to Fabrikam's core web service to retrieve account information.
▪ Use networking technology already available in the .Net Framework. ▪
Consume the JSON that the Fabrikam core web service provides.

User data and alerts

The app must meet the following requirements related to user data and alerts:
▪ Download new monthly bank statements when possible. The download process must not affect the performance of the app.
▪ Report to the user when the statements are downloaded to the device.
▪ Write a log entry when statement downloads are not successful.
▪ Periodically check for user activity and automatically log the user out when there is no activity for more than 15 minutes.

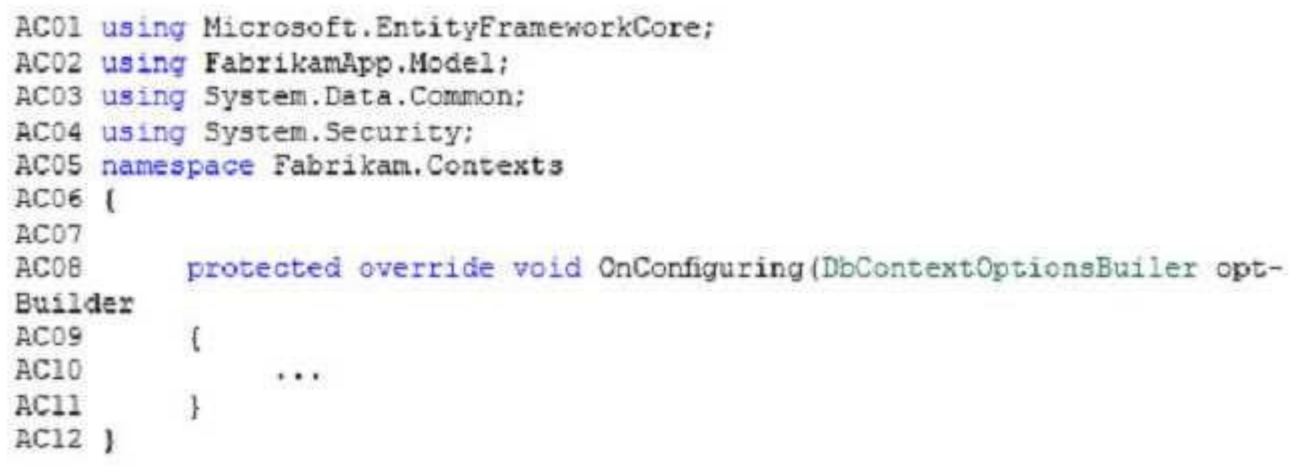
Security

The app must meet the following requirements related to security:
▪ Use a multi-factor authentication (MFA) by using email and a verification code to identify the user. ▪
Securely store credentials and retrieve credentials.
▪ Automatically sign in the user irrespective of the device that is used to sign in to the app.
▪ Store the resource name within the app itself.
▪ Connect to an authentication app by using the URI schema fabrikam-security://oauth/.

Application Structure

AccountContext.cs

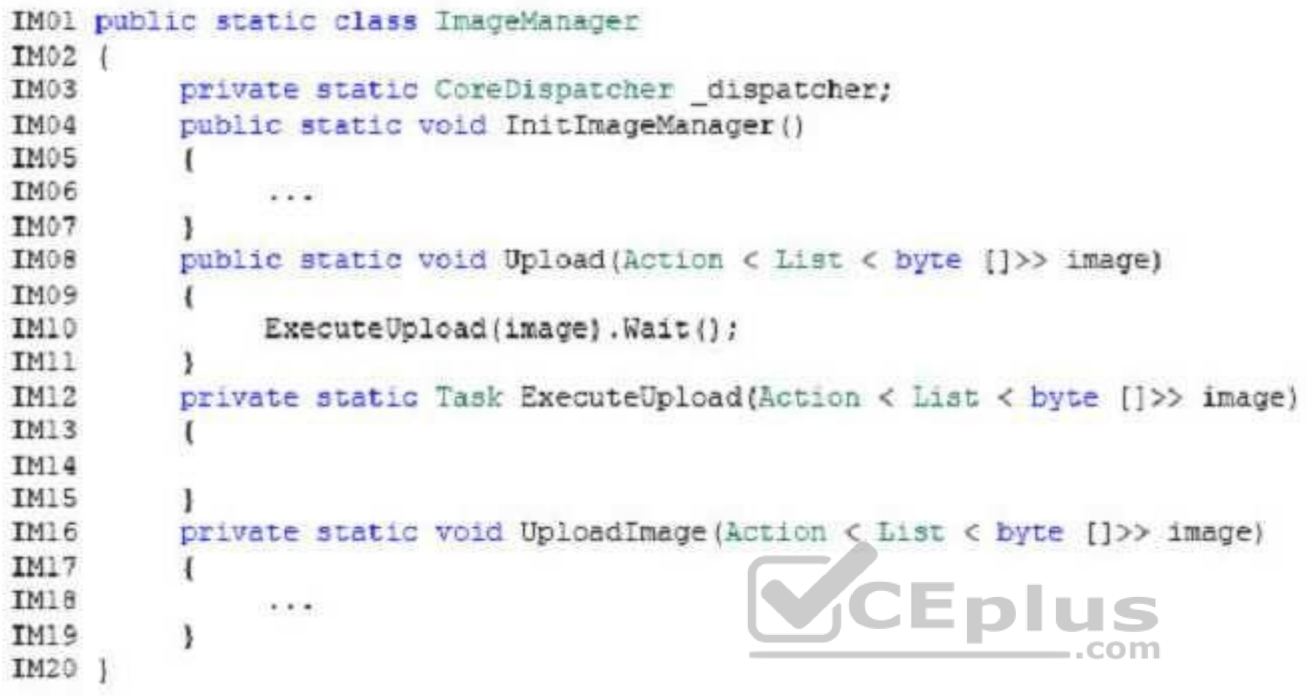
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
AC01 using Microsoft.EntityFrameworkCore;
AC02 using FabrikamApp.Model;
AC03 using System.Data.Common;
AC04 using System.Security;
AC05 namespace Fabrikam.Contexts
AC06 {
AC07
AC08        protected override void OnConfiguring(DbContextOptionsBuiler opt-
Builder
AC09        {
AC10                ...
AC11        }
AC12 }
```

ImageManager.cs

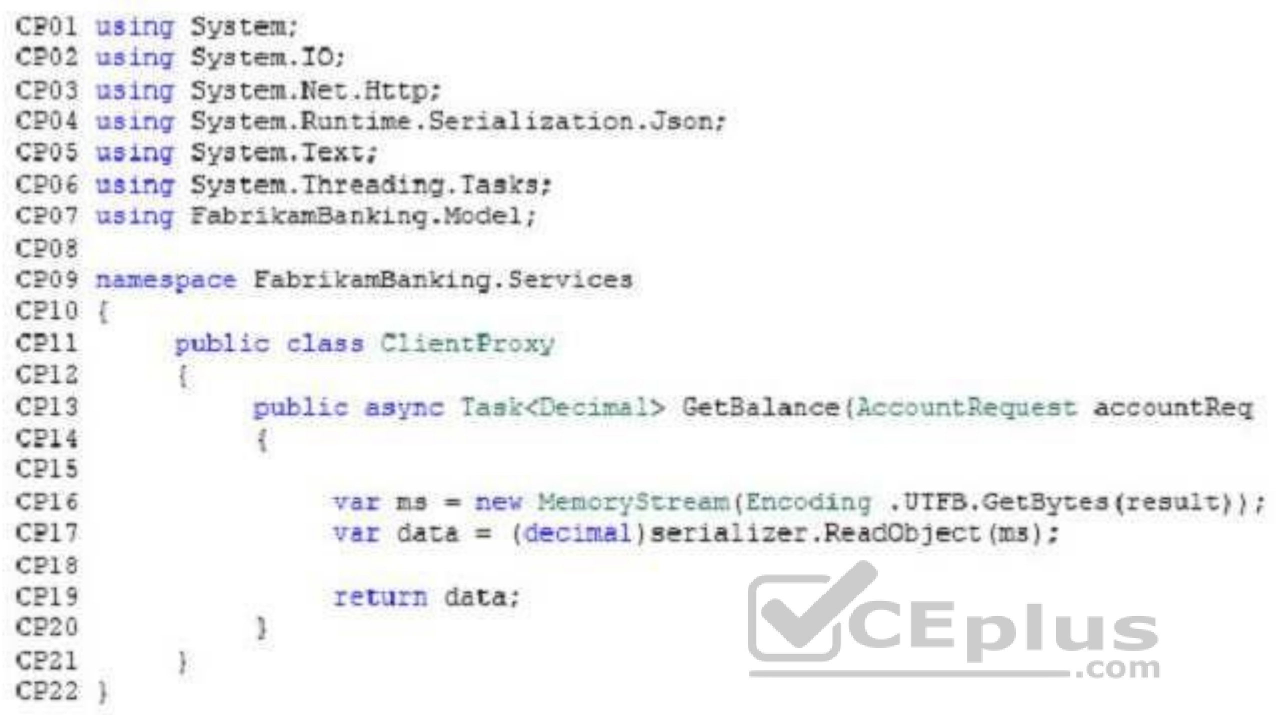
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
IM01 public static class ImageManager
IM02 {
IM03        private static CoreDispatcher _dispatcher;
IM04        public static void InitImageManager()
IM05        {
IM06             ...
IM07        }
IM08        public static void Upload(Action < List < byte []>> image)
IM09        {
IM10             ExecuteUpload(image).Wait();
IM11        }
IM12        private static Task ExecuteUpload(Action < List < byte []>> image)
IM13        {
IM14
IM15        }
IM16        private static void UploadImage(Action < List < byte []>> image)
IM17        {
IM18             ...
IM19        }
IM20 }
```

ClientProxy.cs

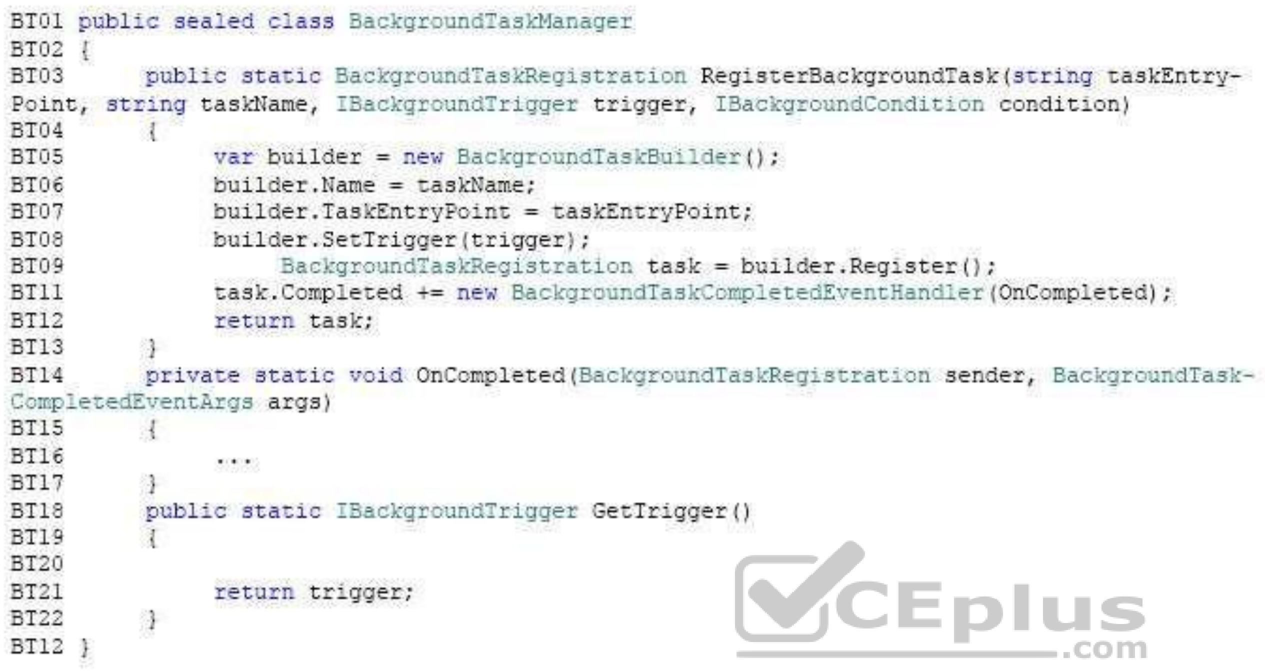
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
CP01 using System;
CP02 using System.IO;
CP03 using System.Net.Http;
CP04 using System.Runtime.Serialization.Json;
CP05 using System.Text;
CP06 using System.Threading.Tasks;
CP07 using FabrikamBanking.Model;
CP08
CP09 namespace FabrikamBanking.Services
CP10 {
CP11      public class ClientProxy
CP12      {
CP13           public async Task<Decimal> GetBalance(AccountRequest accountReq
CP14           {
CP15
CP16                var ms = new MemoryStream(Encoding .UTF8.GetBytes(result));
CP17                var data = (decimal)serializer.ReadObject(ms);
CP18
CP19                return data;
CP20           }
CP21      }
CP22 }
```

BkgTaskMgr.cs

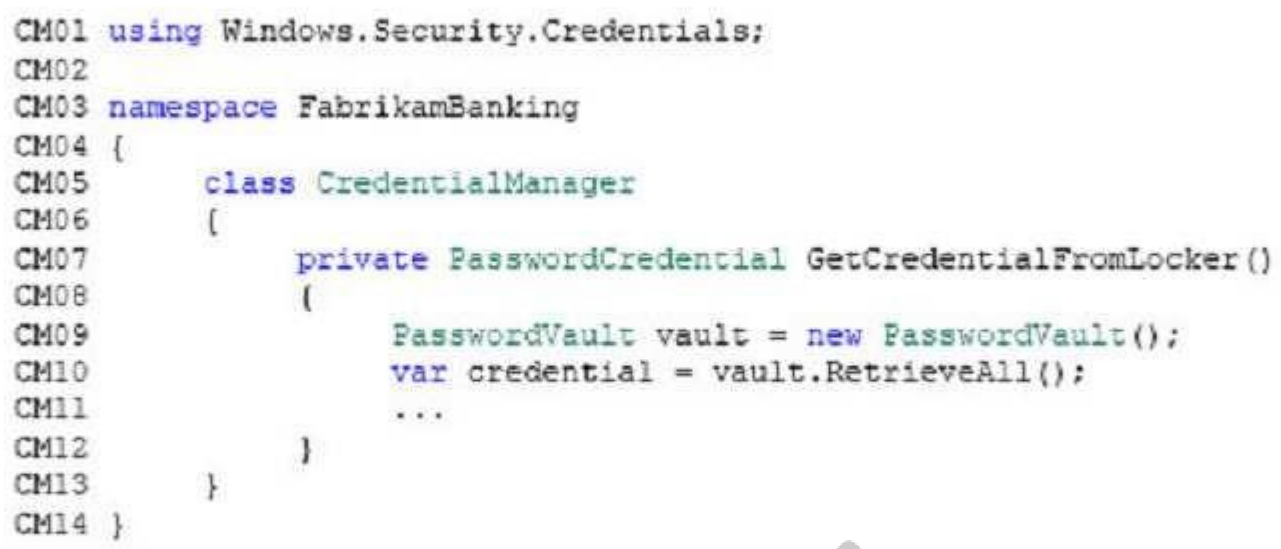
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
BT01 public sealed class BackgroundTaskManager
BT02 {
BT03      public static BackgroundTaskRegistration RegisterBackgroundTask(string taskEntry-
Point, string taskName, IBackgroundTrigger trigger, IBackgroundCondition condition)
BT04      {
BT05           var builder = new BackgroundTaskBuilder();
BT06           builder.Name = taskName;
BT07           builder.TaskEntryPoint = taskEntryPoint;
BT08           builder.SetTrigger(trigger);
BT09                BackgroundTaskRegistration task = builder.Register();
BT11           task.Completed += new BackgroundTaskCompletedEventHandler(OnCompleted);
BT12           return task;
BT13      }
BT14      private static void OnCompleted(BackgroundTaskRegistration sender, BackgroundTask-
CompletedEventArgs args)
BT15      {
BT16           ...
BT17      }
BT18      public static IBackgroundTrigger GetTrigger()
BT19      {
BT20
BT21           return trigger;
BT22      }
BT12 }
```

CredentialManager.cs

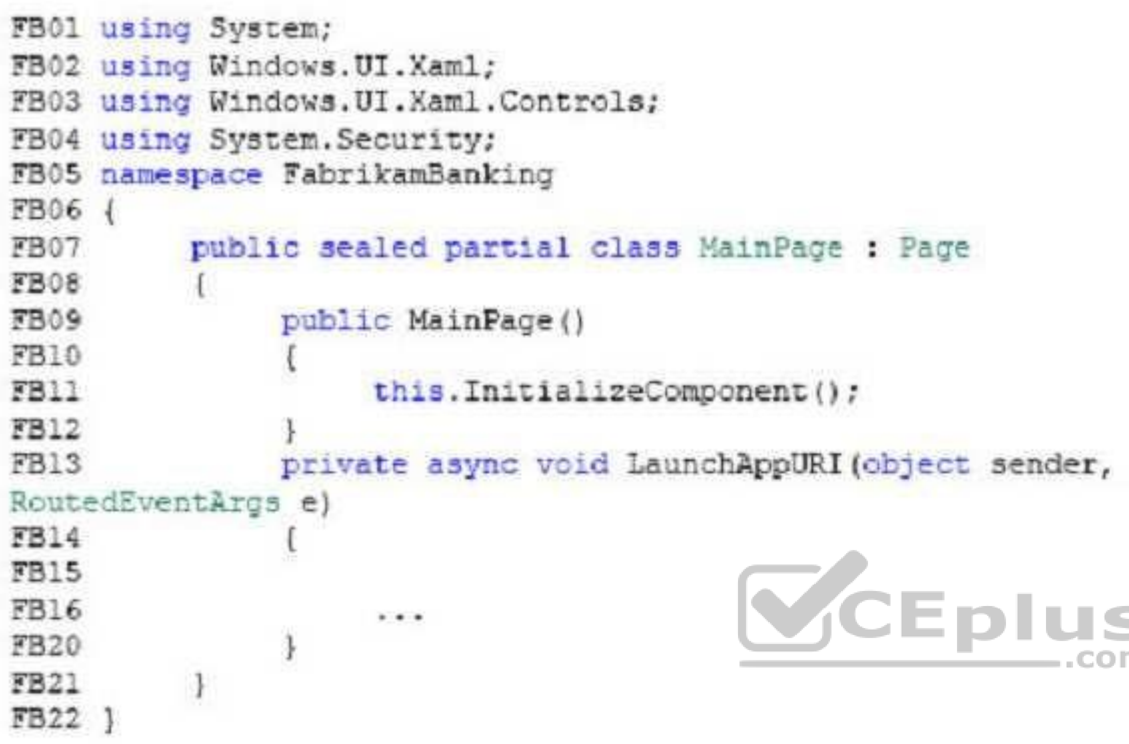
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
CM01 using Windows.Security.Credentials;
CM02
CM03 namespace FabrikamBanking
CM04 {
CM05        class CredentialManager
CM06        {
CM07               private PasswordCredential GetCredentialFromLocker()
CM08               {
CM09                      PasswordVault vault = new PasswordVault();
CM10                      var credential = vault.RetrieveAll();
CM11                      ...
CM12               }
CM13        }
CM14 }
```

MainPage.cs

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

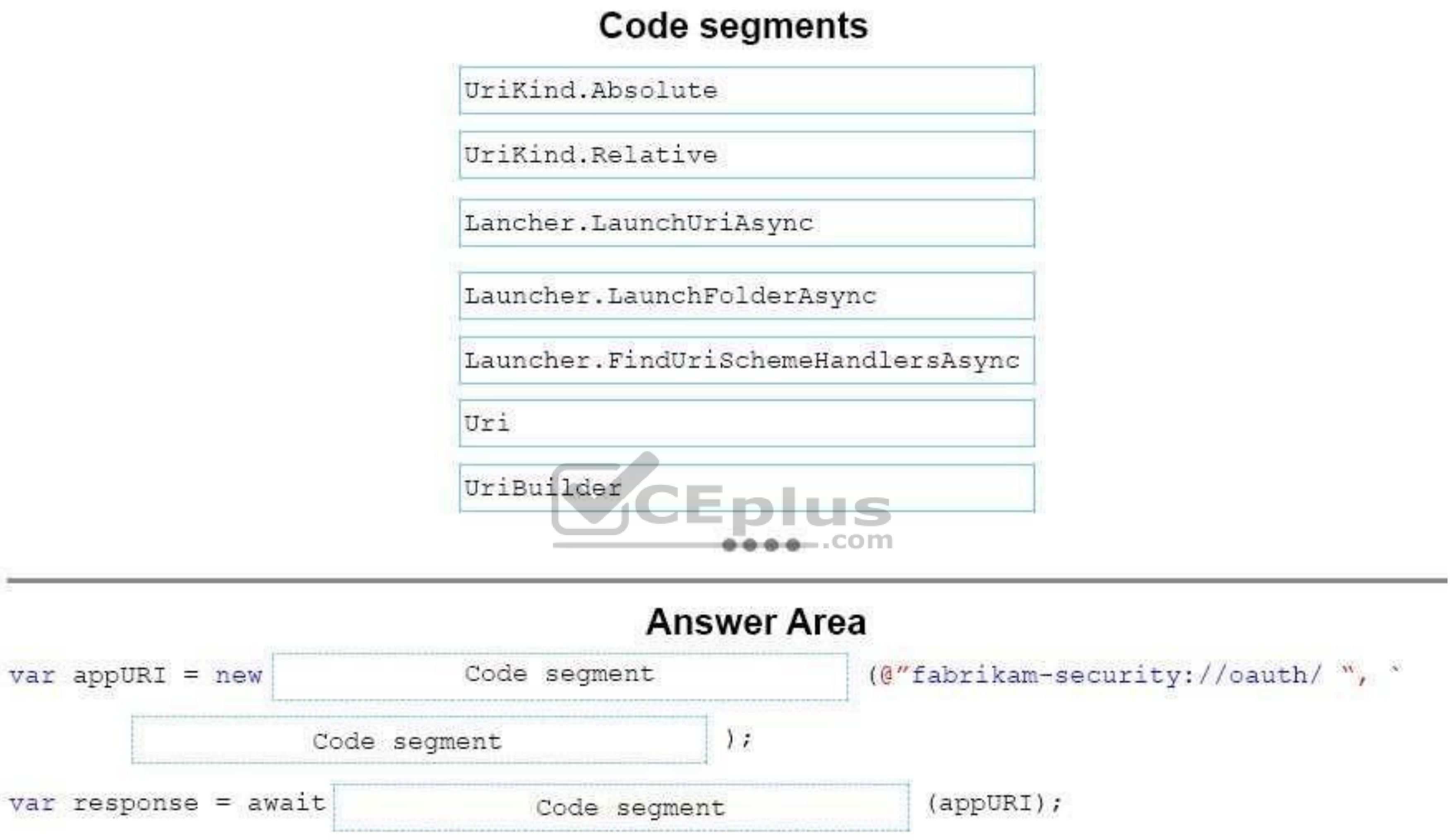


```
FB01 using System;
FB02 using Windows.UI.Xaml;
FB03 using Windows.UI.Xaml.Controls;
FB04 using System.Security;
FB05 namespace FabrikamBanking
FB06 {
FB07        public sealed partial class MainPage : Page
FB08        {
FB09              public MainPage()
FB10              {
FB11                    this.InitializeComponent();
FB12              }
FB13              private async void LaunchAppURI(object sender,
RoutedEventArgs e)
FB14              {
FB15
FB16                    ...
FB20              }
FB21        }
FB22 }
```

You need to launch the authentication app.

How should you complete the relevant code? To answer, drag the appropriate code segments to the correct location or locations. Each code segments may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

Select and Place:



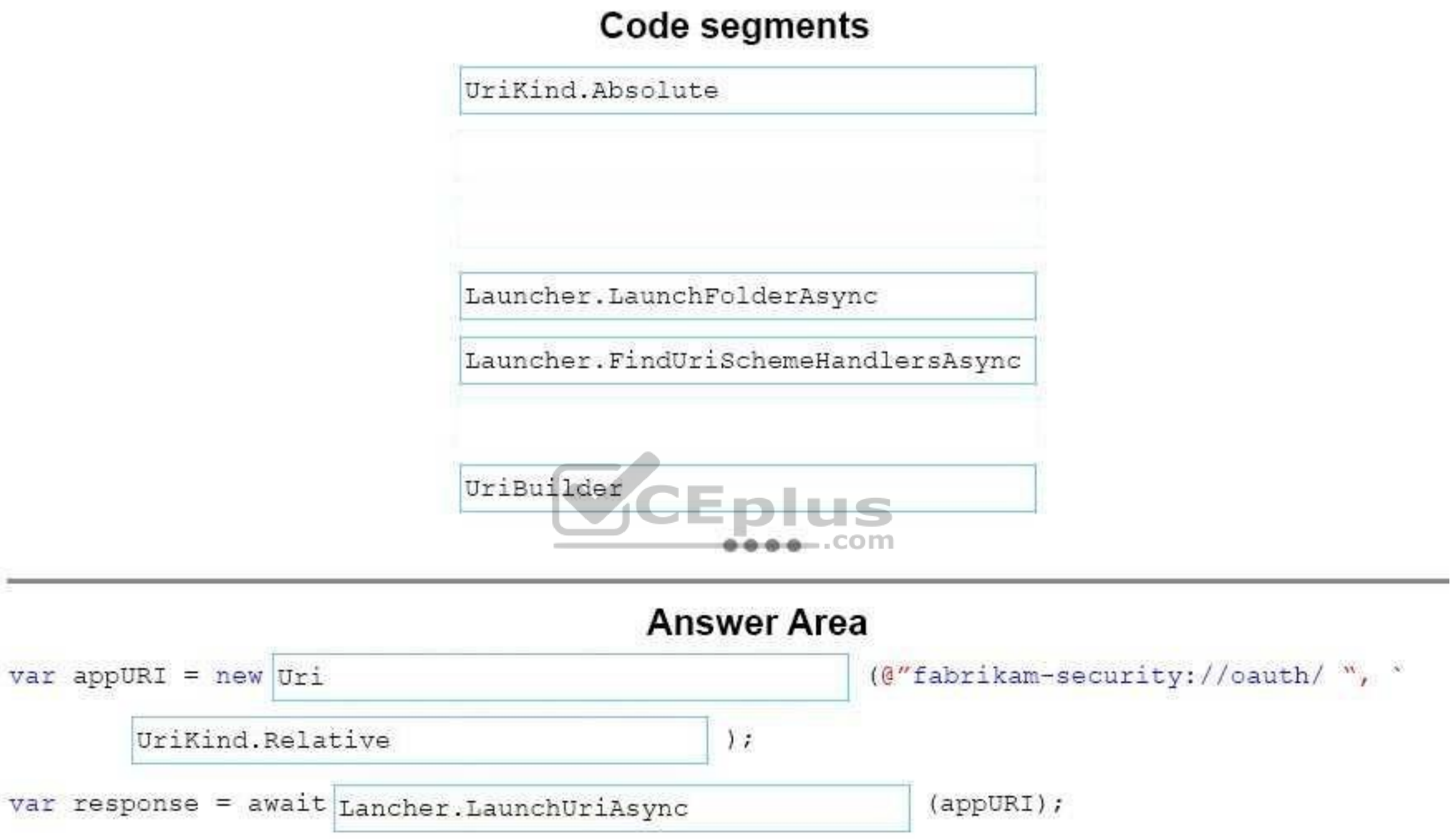
Code segments

```
UriKind.Absolute
UriKind.Relative
Lancher.LaunchUriAsync
Launcher.LaunchFolderAsync
Launcher.FindUriSchemeHandlersAsync
Uri
UriBuilder
```

Answer Area

```
var appURI = new [Code segment] (@"fabrikam-security://oauth/ ", `
        [Code segment] );
var response = await [Code segment] (appURI);
```

Correct Answer:



Code segments

```
UriKind.Absolute
```

```
Launcher.LaunchFolderAsync
```

```
Launcher.FindUriSchemeHandlersAsync
```

```
UriBuilder
```

Answer Area

```
var appURI = new [Uri] (@"fabrikam-security://oauth/ ", "
        [UriKind.Relative] );
var response = await [Lancher.LaunchUriAsync] (appURI);
```

Section: (none)
Explanation

Explanation/Reference:
Explanation:

Box 1: Uri
Box 2: UriKind.Relative
Box 3: Launcher.LaunchUriAsync

The Uri Constructor (Uri, Uri) initializes a new instance of the Uri class based on the combination of a specified base Uri instance and a relative Uri instance.

Syntax: public Uri(
Uri baseUri,
Uri relativeUri
)

Example to launch to URI: var success = await
Windows.System.Launcher.LaunchUriAsync(uri);

Reference:
https://msdn.microsoft.com/en-us/library/ceyeze4f(v=vs.110).aspx

QUESTION 25
DRAG DROP

Case Study
This is a case study. Case studies are not limited separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.
To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.
At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.
To start the case study
To display the first question on this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Background

Fabrikam is a commercial bank. The primary customers are individuals and employers with up to 10,000 employees. Fabrikam provides Internet banking services to customers.

You are developing a Universal Windows Platform (UWP) app for Fabrikam that extends the Internet banking to a UWP app.
Business Requirements

Core functionality

Users must be able to access accounts, view balances, view recent transactions, and deposit checks by using the UWP app.

Usability

The app must use industry proven design patterns across the app. All navigational elements must be visible at all times.

Security

The app must provide secure transactions to protect customer privacy.

Technical Requirements

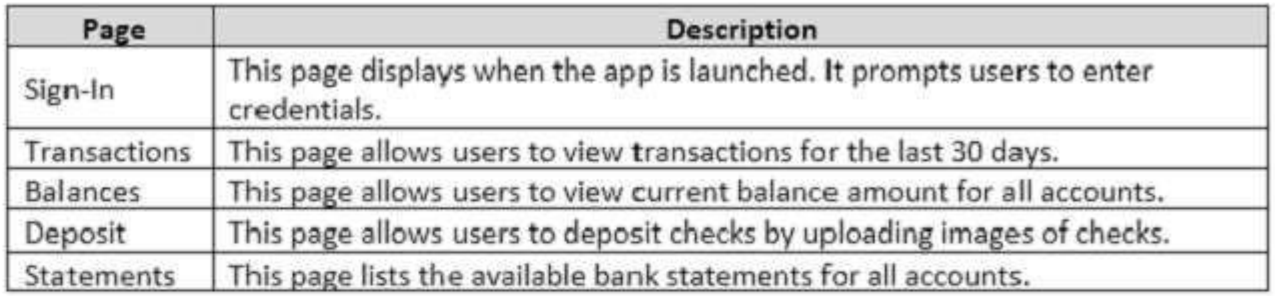
Data

The app must use a file based database. You must use a code first entity framework approach.

User interface

- You must use a Model-View-ViewModel (MVVM) pattern.
- Users must be able to access all content through the top-level navigation after they sign in.
- The app must allow the user to upload up to 50 images (front and back) of checks to deposit.
- During the upload process, the app must be responsive to any other user actions.
- The app must only upload images when no other pending inputs are in the queue.

You must implement the following pages:



Page	Description
Sign-In	This page displays when the app is launched. It prompts users to enter credentials.
Transactions	This page allows users to view transactions for the last 30 days.
Balances	This page allows users to view current balance amount for all accounts.
Deposit	This page allows users to deposit checks by uploading images of checks.
Statements	This page lists the available bank statements for all accounts.

Network and web service

The app must meet the following requirements related to networking and web services:
▪ Connect to a web service over a secure HTTP connection to upload images.
▪ Connect to Fabrikam's core web service to retrieve account information.
▪ Use networking technology already available in the .Net Framework. ▪
Consume the JSON that the Fabrikam core web service provides.

User data and alerts

The app must meet the following requirements related to user data and alerts:
▪ Download new monthly bank statements when possible. The download process must not affect the performance of the app. ▪
Report to the user when the statements are downloaded to the device.
▪ Write a log entry when statement downloads are not successful.
▪ Periodically check for user activity and automatically log the user out when there is no activity for more than 15 minutes.

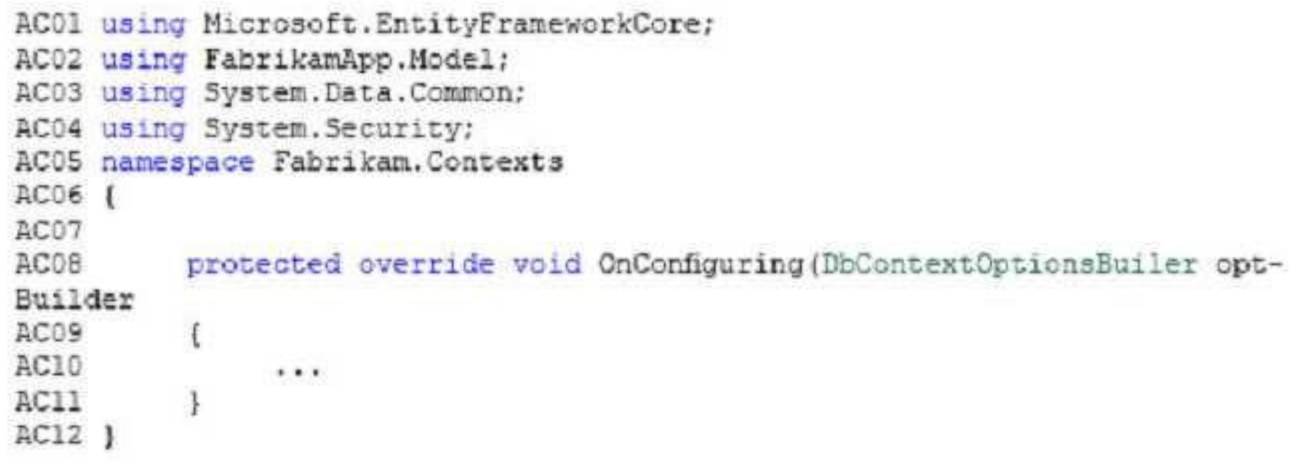
Security

The app must meet the following requirements related to security:
▪ Use a multi-factor authentication (MFA) by using email and a verification code to identify the user.
▪ Securely store credentials and retrieve credentials.
▪ Automatically sign in the user irrespective of the device that is used to sign in to the app.
▪ Store the resource name within the app itself.
▪ Connect to an authentication app by using the URI schema fabrikam-security://oauth/.

Application Structure

AccountContext.cs

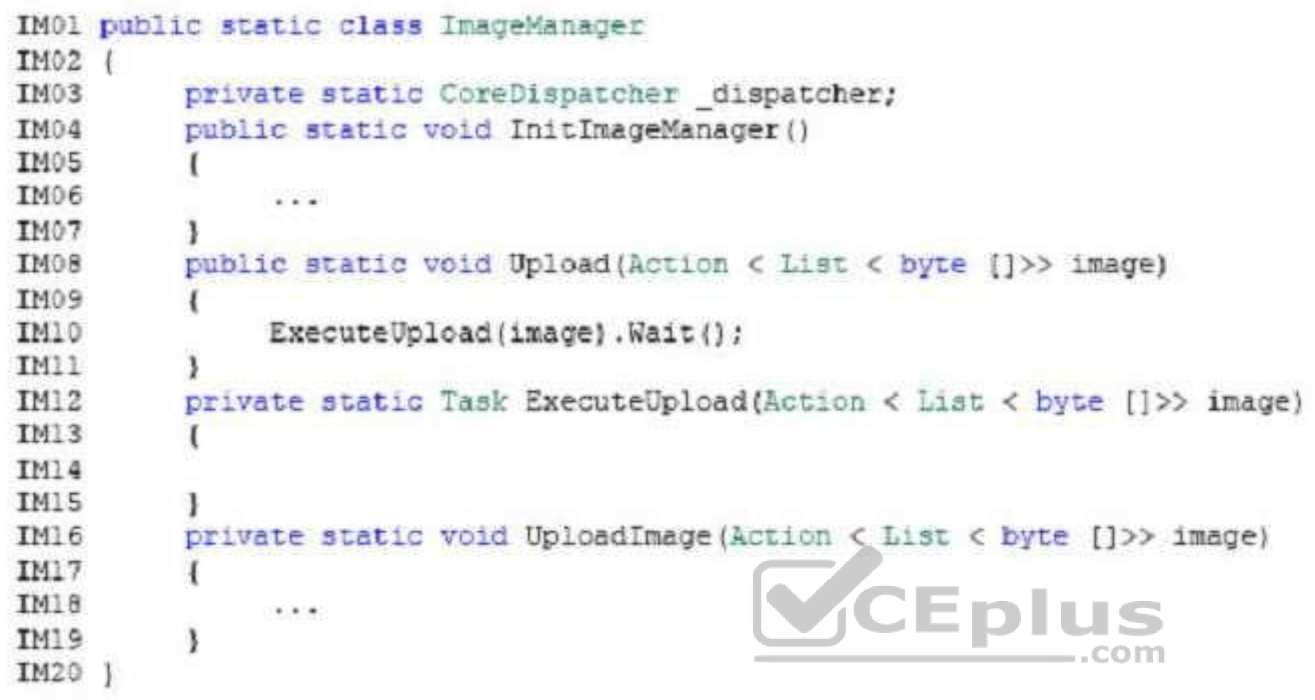
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
AC01 using Microsoft.EntityFrameworkCore;
AC02 using FabrikamApp.Model;
AC03 using System.Data.Common;
AC04 using System.Security;
AC05 namespace Fabrikam.Contexts
AC06 {
AC07
AC08         protected override void OnConfiguring(DbContextOptionsBuiler opt-
Builder
AC09         {
AC10               ...
AC11         }
AC12 }
```

ImageManager.cs

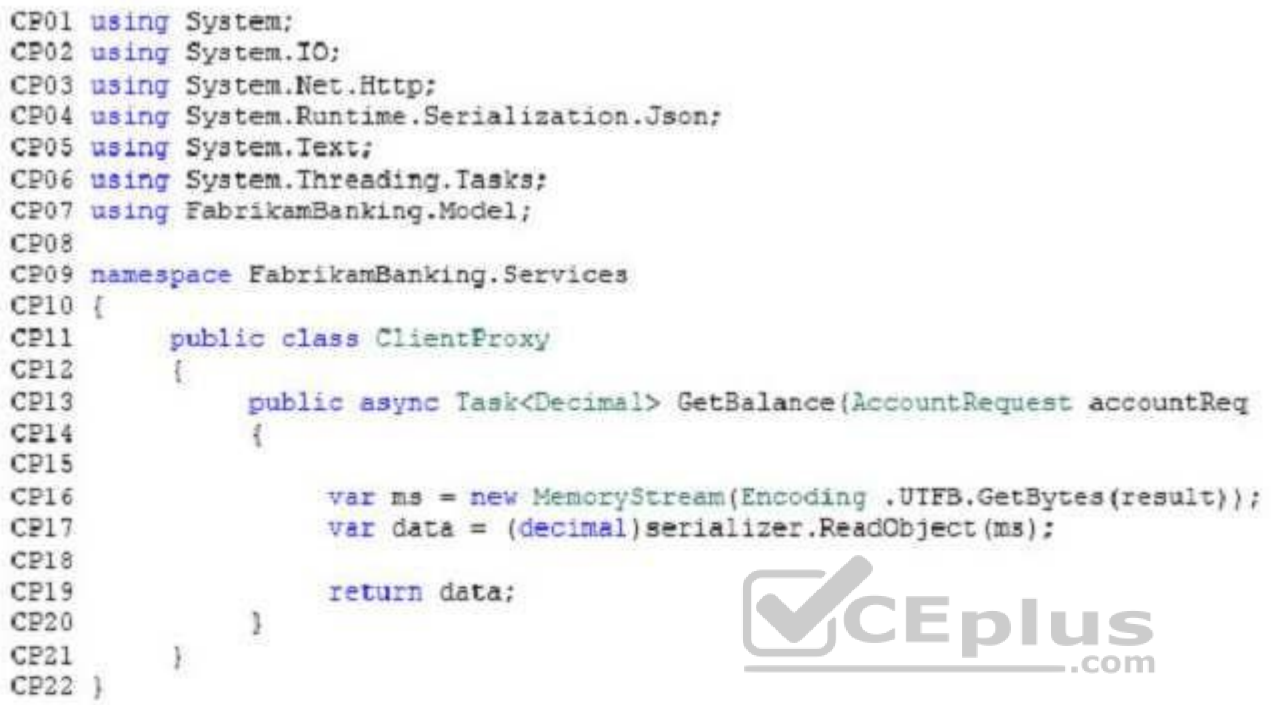
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
IM01 public static class ImageManager
IM02 {
IM03        private static CoreDispatcher _dispatcher;
IM04        public static void InitImageManager()
IM05        {
IM06            ...
IM07        }
IM08        public static void Upload(Action < List < byte []>> image)
IM09        {
IM10            ExecuteUpload(image).Wait();
IM11        }
IM12        private static Task ExecuteUpload(Action < List < byte []>> image)
IM13        {
IM14
IM15        }
IM16        private static void UploadImage(Action < List < byte []>> image)
IM17        {
IM18            ...
IM19        }
IM20 }
```

ClientProxy.cs

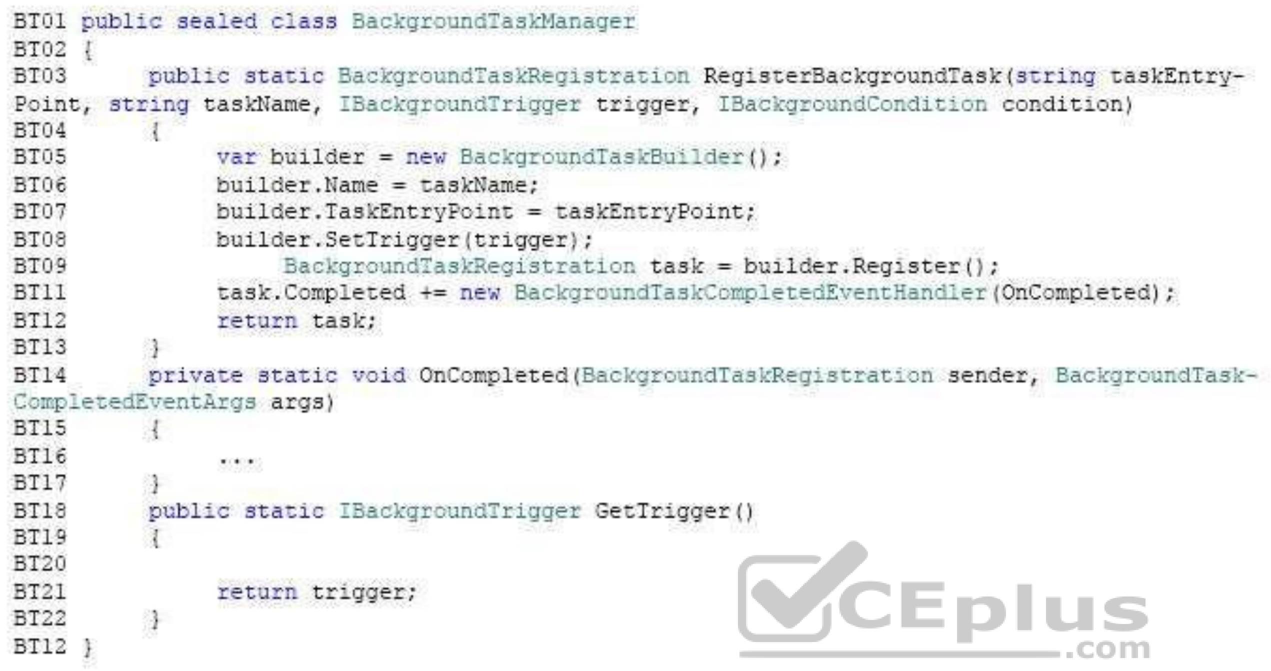
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
CP01 using System;
CP02 using System.IO;
CP03 using System.Net.Http;
CP04 using System.Runtime.Serialization.Json;
CP05 using System.Text;
CP06 using System.Threading.Tasks;
CP07 using FabrikamBanking.Model;
CP08
CP09 namespace FabrikamBanking.Services
CP10 {
CP11      public class ClientProxy
CP12      {
CP13           public async Task<Decimal> GetBalance(AccountRequest accountReq
CP14           {
CP15
CP16                var ms = new MemoryStream(Encoding .UTF8.GetBytes(result));
CP17                var data = (decimal)serializer.ReadObject(ms);
CP18
CP19                return data;
CP20           }
CP21      }
CP22 }
```

BkgTaskMgr.cs

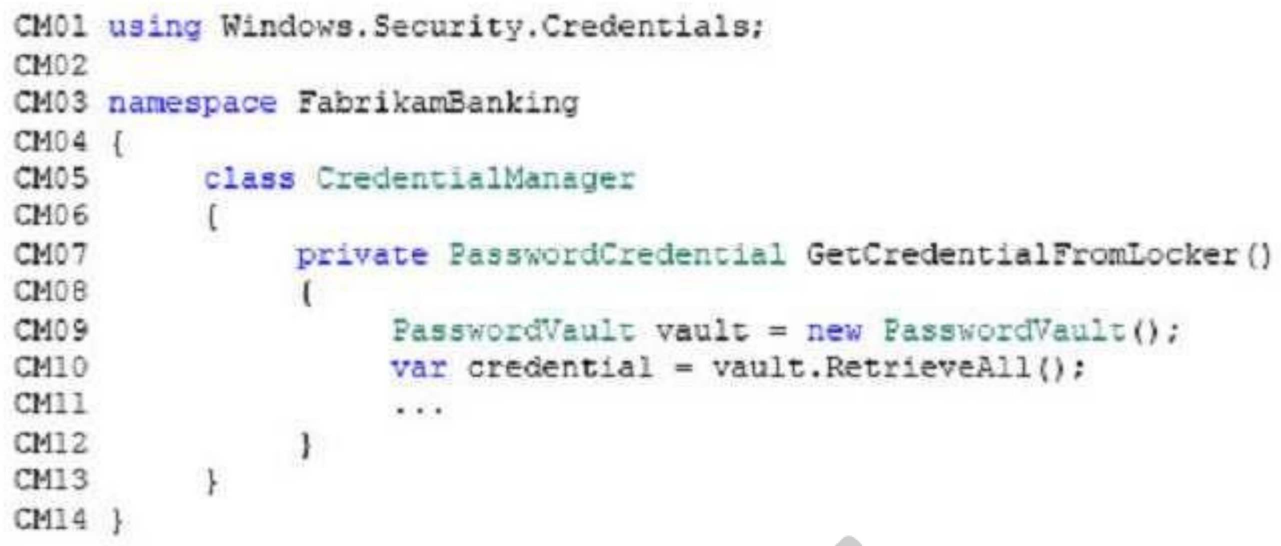
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
BT01 public sealed class BackgroundTaskManager
BT02 {
BT03      public static BackgroundTaskRegistration RegisterBackgroundTask(string taskEntry-
Point, string taskName, IBackgroundTrigger trigger, IBackgroundCondition condition)
BT04      {
BT05           var builder = new BackgroundTaskBuilder();
BT06           builder.Name = taskName;
BT07           builder.TaskEntryPoint = taskEntryPoint;
BT08           builder.SetTrigger(trigger);
BT09                BackgroundTaskRegistration task = builder.Register();
BT11           task.Completed += new BackgroundTaskCompletedEventHandler(OnCompleted);
BT12           return task;
BT13      }
BT14      private static void OnCompleted(BackgroundTaskRegistration sender, BackgroundTask-
CompletedEventArgs args)
BT15      {
BT16           ...
BT17      }
BT18      public static IBackgroundTrigger GetTrigger()
BT19      {
BT20
BT21           return trigger;
BT22      }
BT12 }
```

CredentialManager.cs

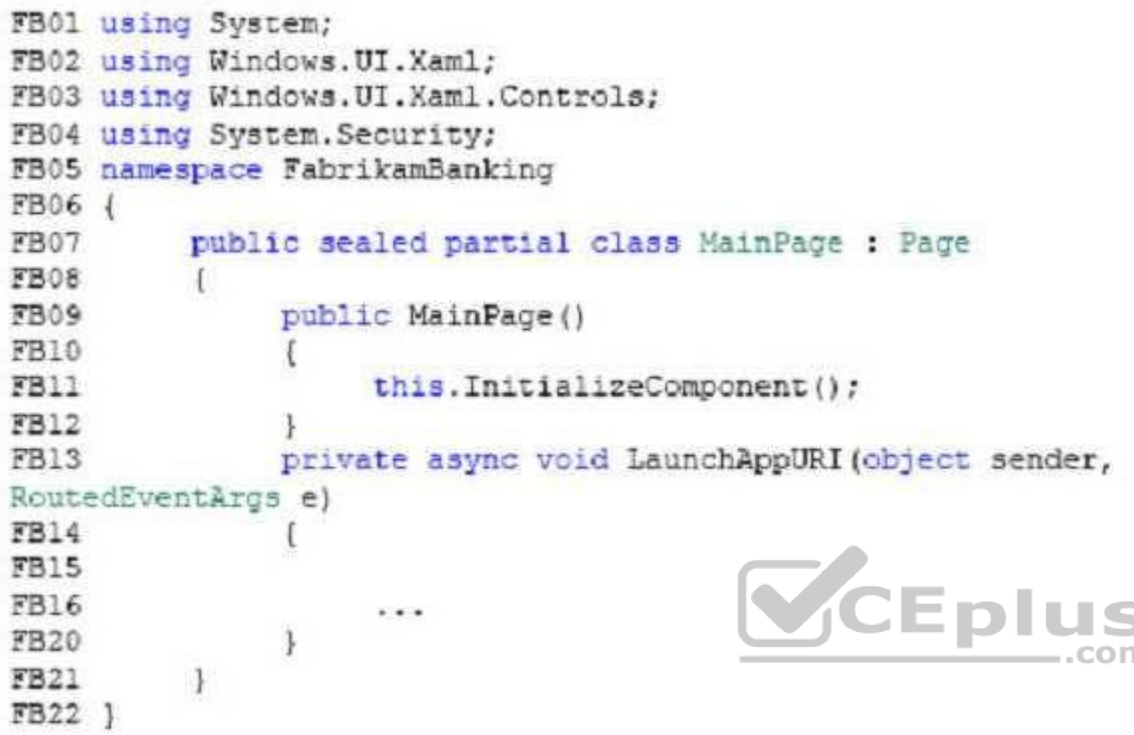
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
CM01 using Windows.Security.Credentials;
CM02
CM03 namespace FabrikamBanking
CM04 {
CM05       class CredentialManager
CM06       {
CM07             private PasswordCredential GetCredentialFromLocker()
CM08             {
CM09                   PasswordVault vault = new PasswordVault();
CM10                   var credential = vault.RetrieveAll();
CM11                   ...
CM12             }
CM13       }
CM14 }
```

MainPage.cs

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

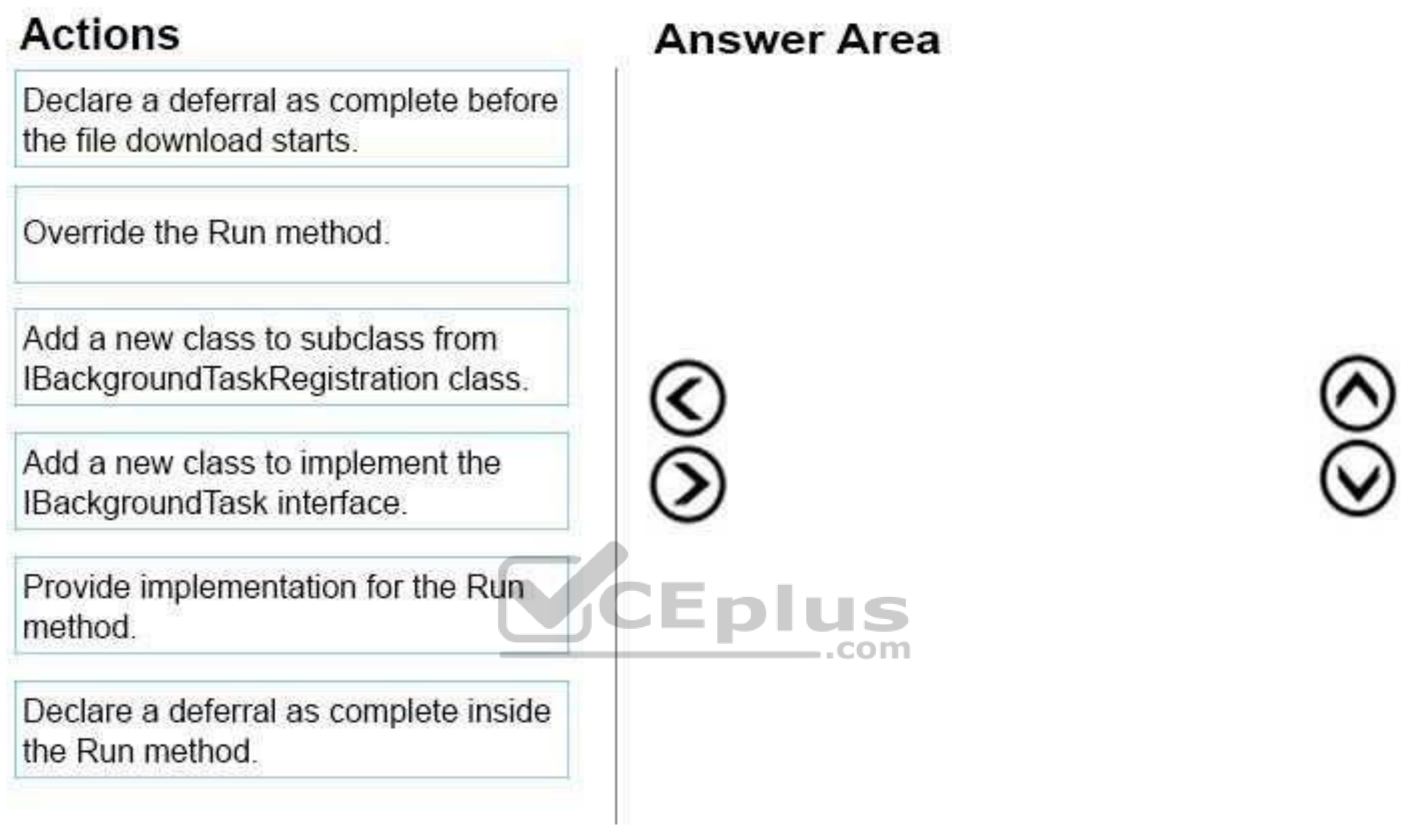


```
FB01 using System;
FB02 using Windows.UI.Xaml;
FB03 using Windows.UI.Xaml.Controls;
FB04 using System.Security;
FB05 namespace FabrikamBanking
FB06 {
FB07        public sealed partial class MainPage : Page
FB08        {
FB09               public MainPage()
FB10               {
FB11                      this.InitializeComponent();
FB12               }
FB13               private async void LaunchAppURI(object sender,
RoutedEventArgs e)
FB14               {
FB15
FB16                      ...
FB20               }
FB21        }
FB22 }
```

You need to download the bank statements and alert the user when the download is complete.

Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.

Select and Place:



Actions

Declare a deferral as complete before the file download starts.

Override the Run method.

Add a new class to subclass from IBackgroundTaskRegistration class.

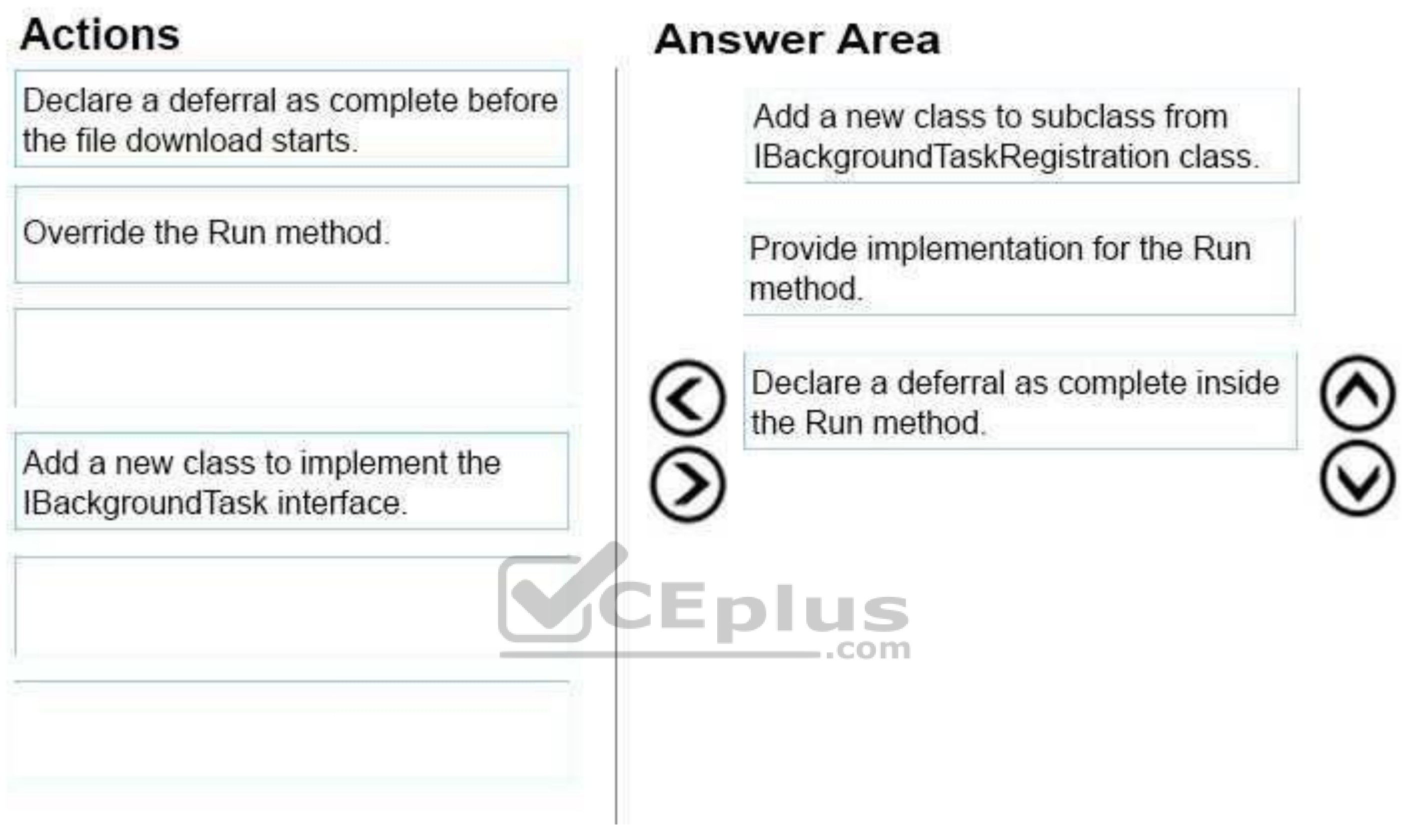
Add a new class to implement the IBackgroundTask interface.

Provide implementation for the Run method.

Declare a deferral as complete inside the Run method.

Answer Area

Correct Answer:



Actions

Declare a deferral as complete before the file download starts.

Override the Run method.

Add a new class to implement the IBackgroundTask interface.

Answer Area

Add a new class to subclass from IBackgroundTaskRegistration class.

Provide implementation for the Run method.

Declare a deferral as complete inside the Run method.

Section: (none)
Explanation

Explanation/Reference:
Explanation:

From Scenario:
The app must meet the following requirements related to user data and alerts:

- Download new monthly bank statements when possible. The download process must not affect the performance of the app.
- Report to the user when the statements are downloaded to the device.
- Write a log entry when statement downloads are not successful.

You can run code in the background by writing classes that implement the IBackgroundTask interface.

If you run any asynchronous code in your background task, then your background task needs to use a deferral. If you don't use a deferral, then the background task process can terminate unexpectedly if the Run method completes before your asynchronous method call has completed.
Request the deferral in the Run method before calling the asynchronous method. Save the deferral to a global variable so it can be accessed from the asynchronous method. Declare the deferral complete after the asynchronous code completes.

Reference: https://docs.microsoft.com/en-us/windows/uwp/launch-resume/create-and-register-a-background-task

QUESTION 26
DRAG DROP

Case Study
This is a case study. Case studies are not limited separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.
To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.
At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.
To start the case study
To display the first question on this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Background

Fabrikam is a commercial bank. The primary customers are individuals and employers with up to 10,000 employees. Fabrikam provides Internet banking services to customers.

You are developing a Universal Windows Platform (UWP) app for Fabrikam that extends the Internet banking to a UWP app.

Business Requirements

Core functionality

Users must be able to access accounts, view balances, view recent transactions, and deposit checks by using the UWP app.

Usability
The app must use industry proven design patterns across the app. All navigational elements must be visible at all times.

Security

The app must provide secure transactions to protect customer privacy.

Technical Requirements

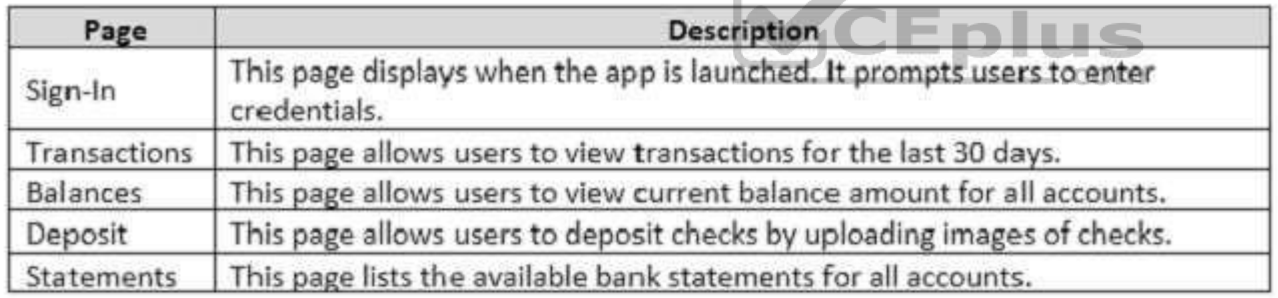
Data

The app must use a file based database. You must use a code first entity framework approach.

User interface

- You must use a Model-View-ViewModel (MVVM) pattern.
- Users must be able to access all content through the top-level navigation after they sign in.
- The app must allow the user to upload up to 50 images (front and back) of checks to deposit.
- During the upload process, the app must be responsive to any other user actions.
- The app must only upload images when no other pending inputs are in the queue.

You must implement the following pages:



Page	Description
Sign-In	This page displays when the app is launched. It prompts users to enter credentials.
Transactions	This page allows users to view transactions for the last 30 days.
Balances	This page allows users to view current balance amount for all accounts.
Deposit	This page allows users to deposit checks by uploading images of checks.
Statements	This page lists the available bank statements for all accounts.

Network and web service

The app must meet the following requirements related to networking and web services:
- Connect to a web service over a secure HTTP connection to upload images.
- Connect to Fabrikam's core web service to retrieve account information.
- Use networking technology already available in the .Net Framework.
- Consume the JSON that the Fabrikam core web service provides.

User data and alerts

The app must meet the following requirements related to user data and alerts:

▪ Download new monthly bank statements when possible. The download process must not affect the performance of the app. ▪ Report to the user when the statements are downloaded to the device.
▪ Write a log entry when statement downloads are not successful.
▪ Periodically check for user activity and automatically log the user out when there is no activity for more than 15 minutes.

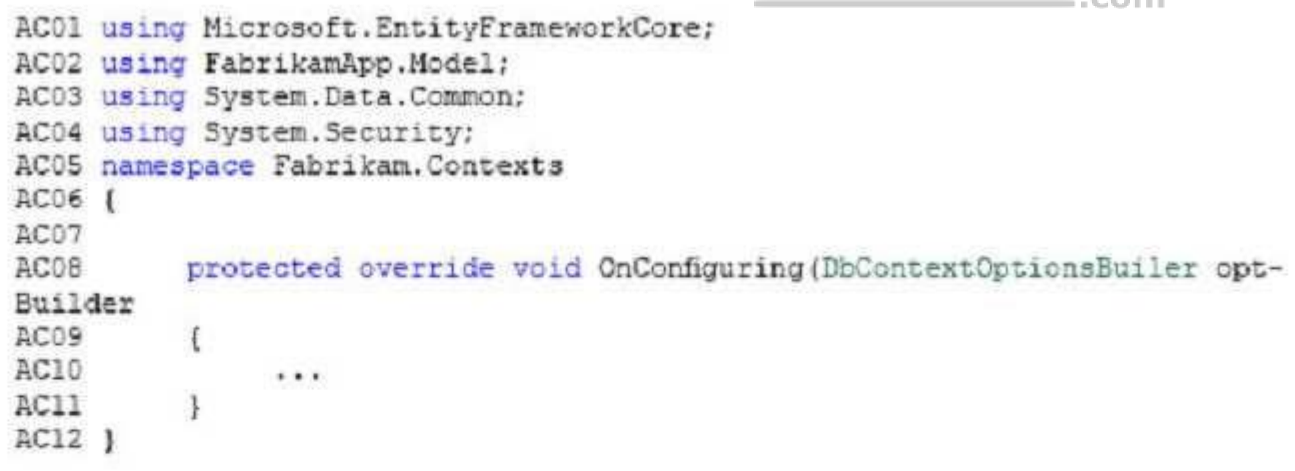
Security

The app must meet the following requirements related to security:
▪ Use a multi-factor authentication (MFA) by using email and a verification code to identify the user.
▪ Securely store credentials and retrieve credentials.
▪ Automatically sign in the user irrespective of the device that is used to sign in to the app.
▪ Store the resource name within the app itself.
▪ Connect to an authentication app by using the URI schema fabrikam-security://oauth/.

Application Structure

AccountContext.cs

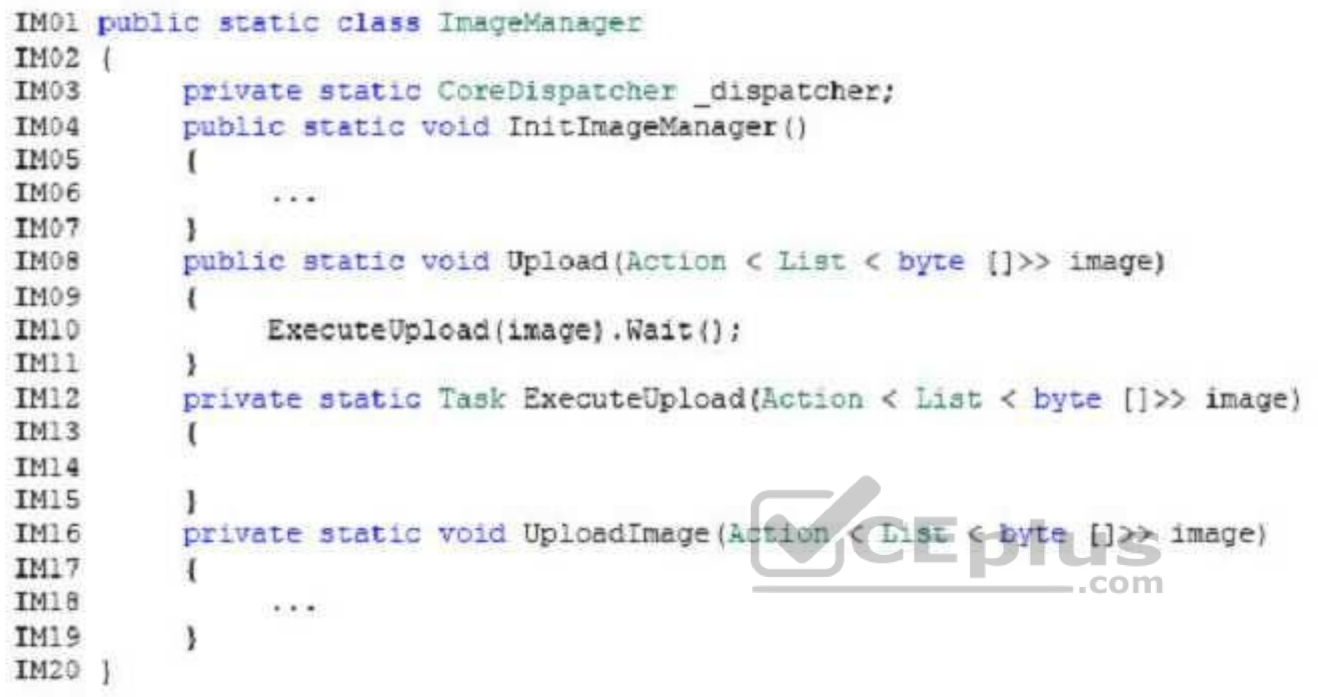
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
AC01 using Microsoft.EntityFrameworkCore;
AC02 using FabrikamApp.Model;
AC03 using System.Data.Common;
AC04 using System.Security;
AC05 namespace Fabrikam.Contexts
AC06 {
AC07
AC08         protected override void OnConfiguring(DbContextOptionsBuiler opt-
Builder
AC09         {
AC10              ...
AC11         }
AC12 }
```

ImageManager.cs

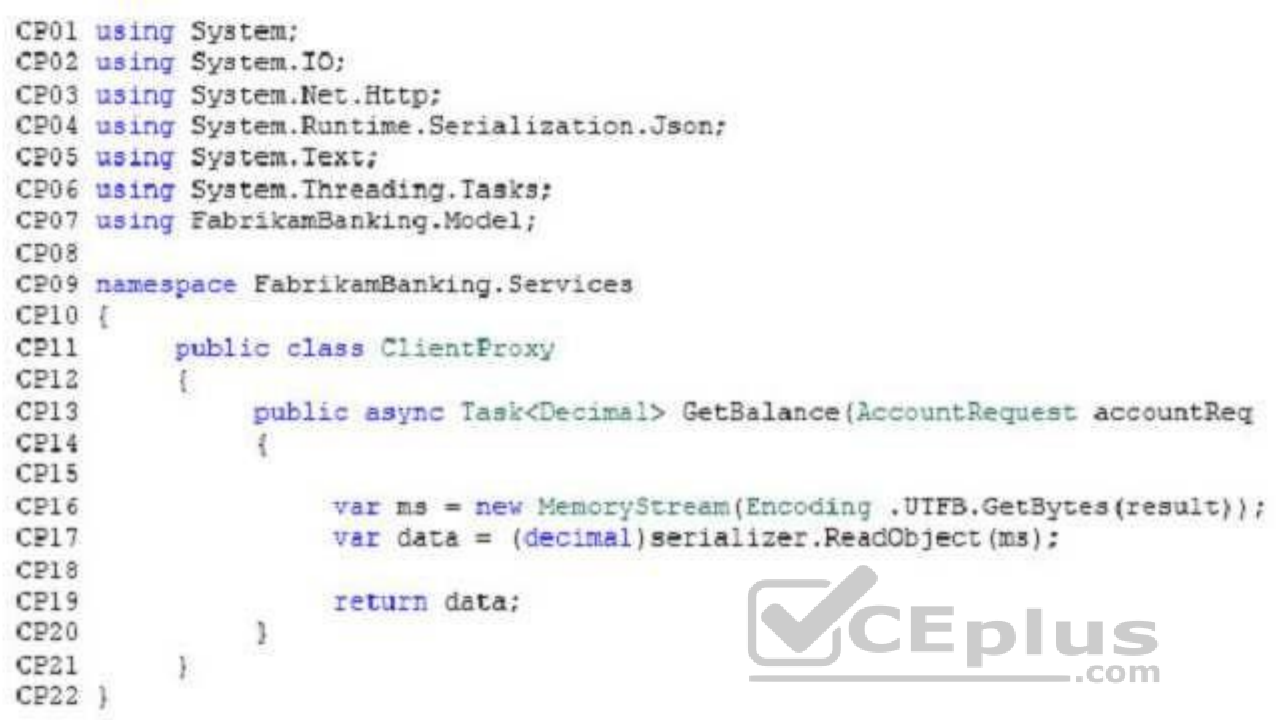
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
IM01 public static class ImageManager
IM02 {
IM03       private static CoreDispatcher _dispatcher;
IM04       public static void InitImageManager()
IM05       {
IM06            ...
IM07       }
IM08       public static void Upload(Action < List < byte []>> image)
IM09       {
IM10            ExecuteUpload(image).Wait();
IM11       }
IM12       private static Task ExecuteUpload(Action < List < byte []>> image)
IM13       {
IM14
IM15       }
IM16       private static void UploadImage(Action < List < byte []>> image)
IM17       {
IM18            ...
IM19       }
IM20 }
```

ClientProxy.cs

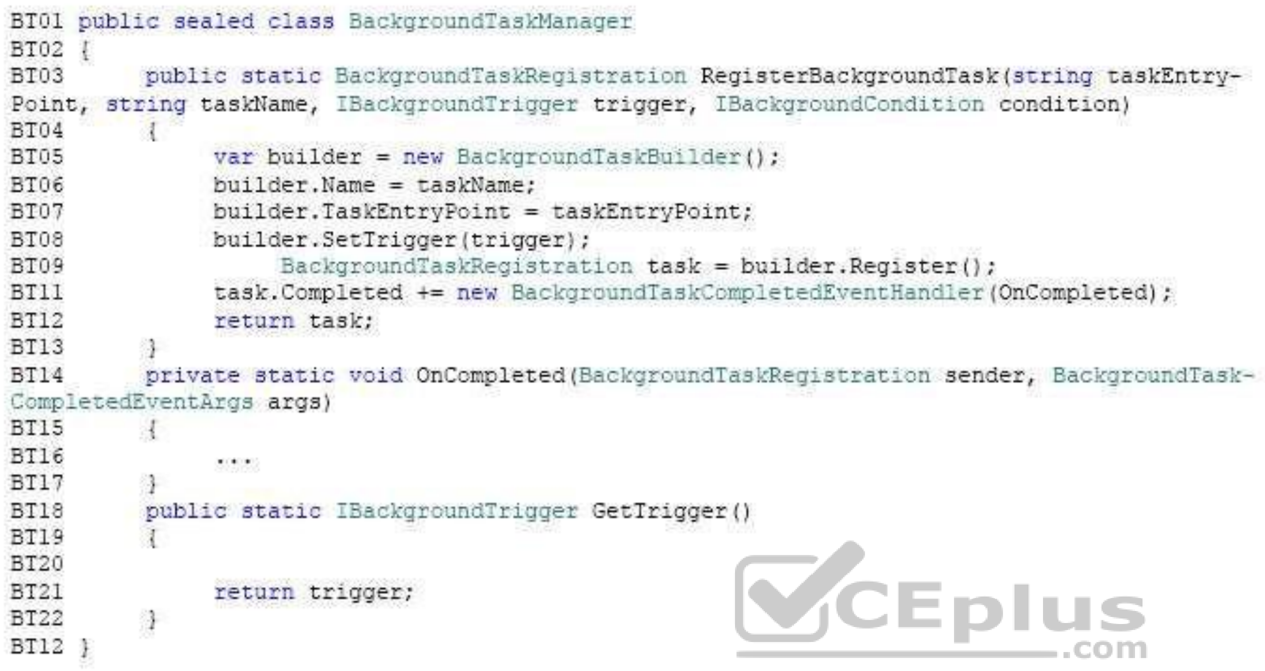
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
CP01 using System;
CP02 using System.IO;
CP03 using System.Net.Http;
CP04 using System.Runtime.Serialization.Json;
CP05 using System.Text;
CP06 using System.Threading.Tasks;
CP07 using FabrikamBanking.Model;
CP08
CP09 namespace FabrikamBanking.Services
CP10 {
CP11         public class ClientProxy
CP12         {
CP13                 public async Task<Decimal> GetBalance(AccountRequest accountReq
CP14                 {
CP15
CP16                         var ms = new MemoryStream(Encoding .UTF8.GetBytes(result));
CP17                         var data = (decimal)serializer.ReadObject(ms);
CP18
CP19                         return data;
CP20                 }
CP21         }
CP22 }
```

BkgTaskMgr.cs

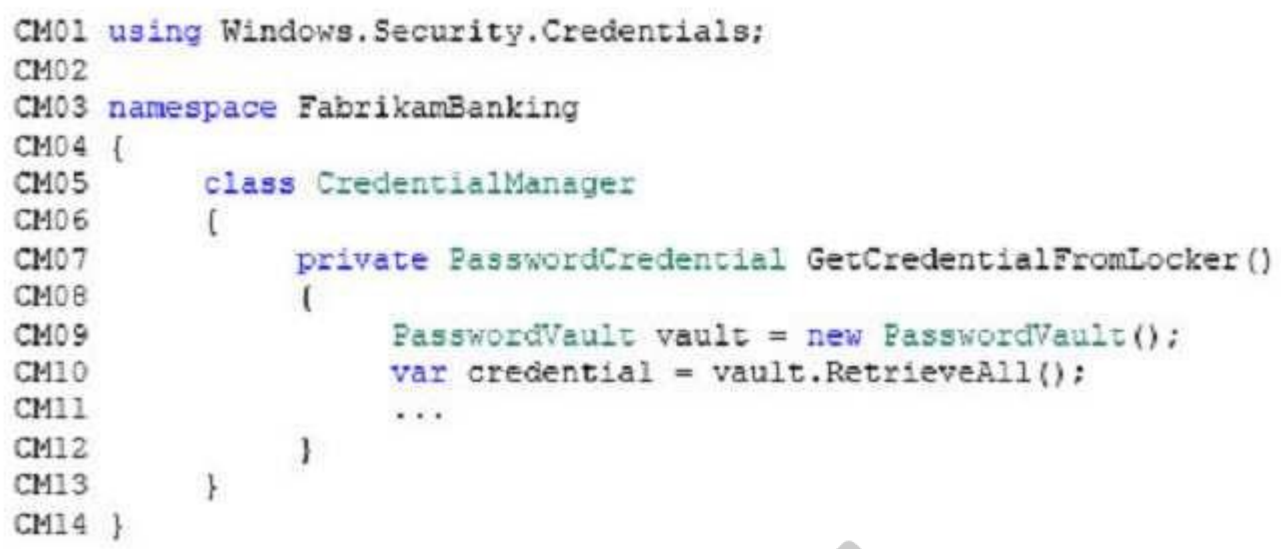
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
BT01 public sealed class BackgroundTaskManager
BT02 {
BT03      public static BackgroundTaskRegistration RegisterBackgroundTask(string taskEntry-
Point, string taskName, IBackgroundTrigger trigger, IBackgroundCondition condition)
BT04      {
BT05           var builder = new BackgroundTaskBuilder();
BT06           builder.Name = taskName;
BT07           builder.TaskEntryPoint = taskEntryPoint;
BT08           builder.SetTrigger(trigger);
BT09                BackgroundTaskRegistration task = builder.Register();
BT11           task.Completed += new BackgroundTaskCompletedEventHandler(OnCompleted);
BT12           return task;
BT13      }
BT14      private static void OnCompleted(BackgroundTaskRegistration sender, BackgroundTask-
CompletedEventArgs args)
BT15      {
BT16           ...
BT17      }
BT18      public static IBackgroundTrigger GetTrigger()
BT19      {
BT20
BT21           return trigger;
BT22      }
BT12 }
```

CredentialManager.cs

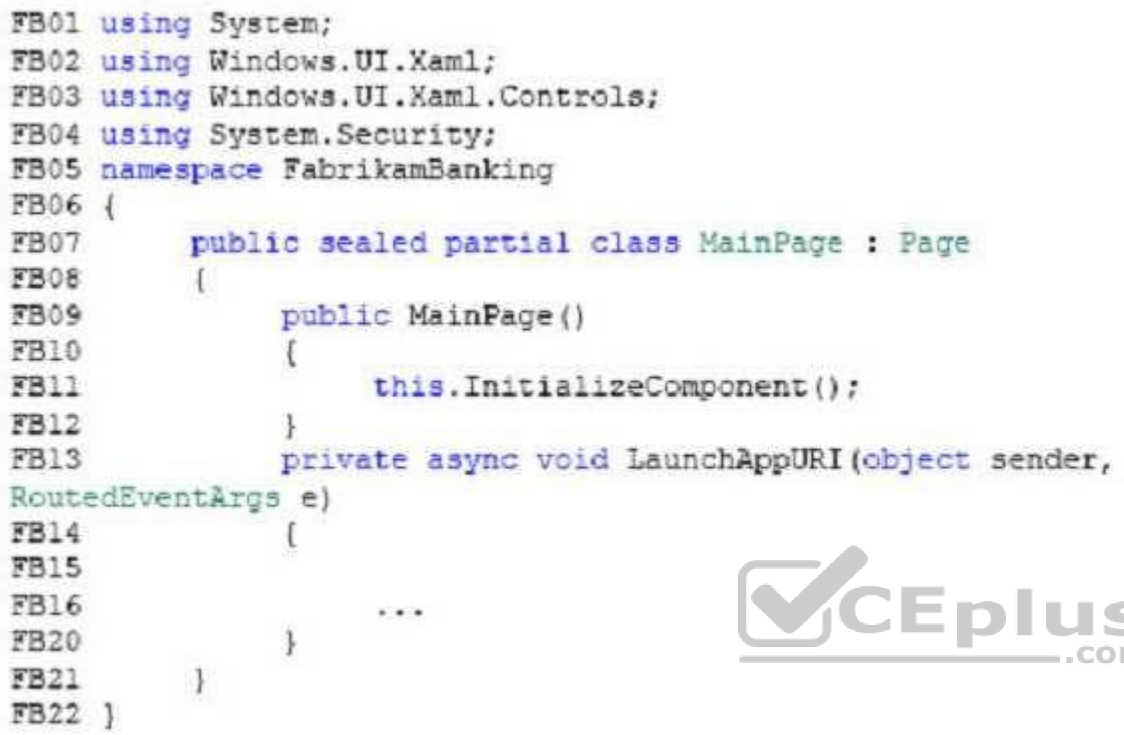
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
CM01 using Windows.Security.Credentials;
CM02
CM03 namespace FabrikamBanking
CM04 {
CM05       class CredentialManager
CM06       {
CM07             private PasswordCredential GetCredentialFromLocker()
CM08             {
CM09                   PasswordVault vault = new PasswordVault();
CM10                   var credential = vault.RetrieveAll();
CM11                   ...
CM12             }
CM13       }
CM14 }
```

MainPage.cs

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

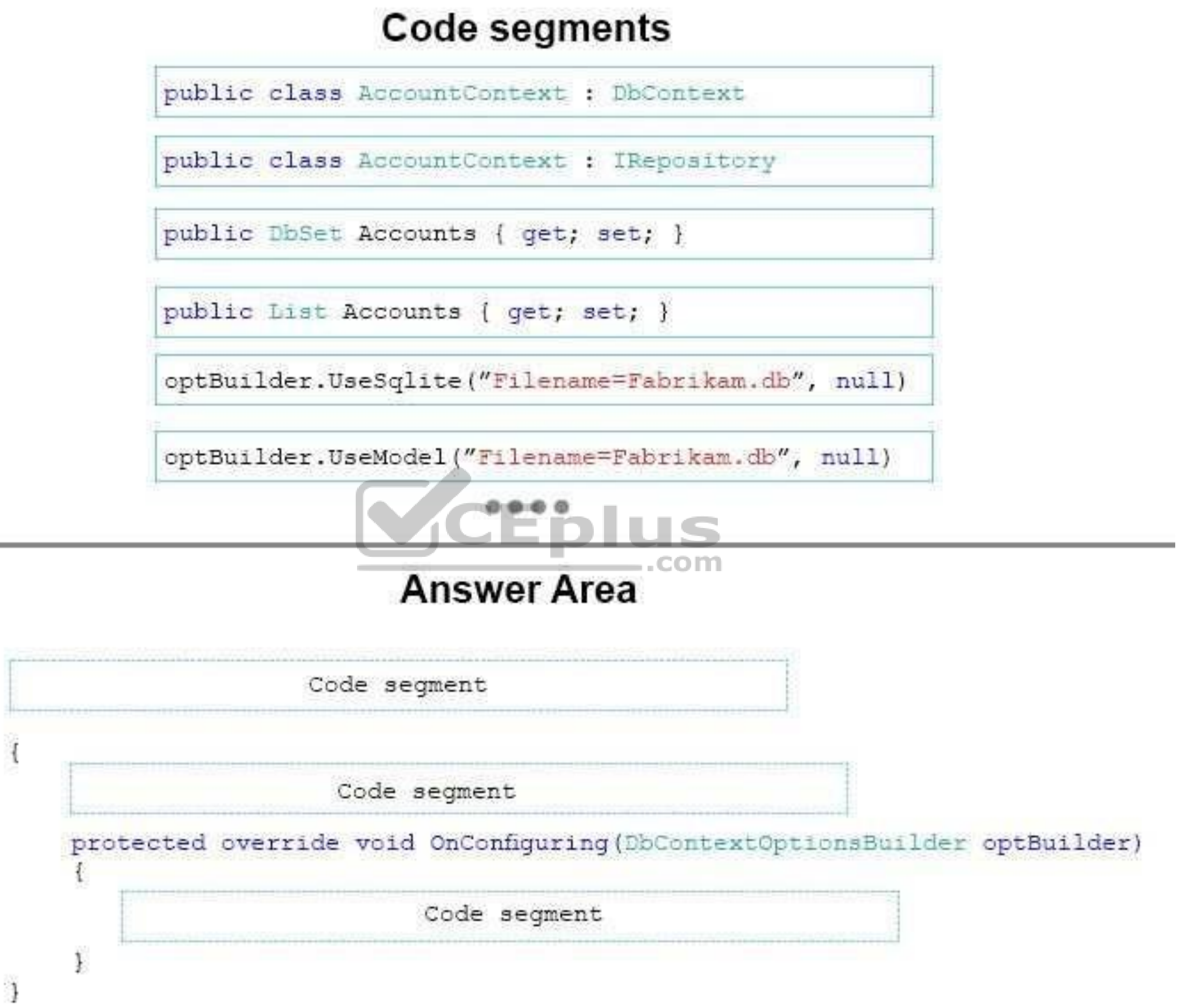


```
FB01 using System;
FB02 using Windows.UI.Xaml;
FB03 using Windows.UI.Xaml.Controls;
FB04 using System.Security;
FB05 namespace FabrikamBanking
FB06 {
FB07        public sealed partial class MainPage : Page
FB08        {
FB09               public MainPage()
FB10               {
FB11                      this.InitializeComponent();
FB12               }
FB13               private async void LaunchAppURI(object sender,
RoutedEventArgs e)
FB14               {
FB15
FB16                      ...
FB20               }
FB21        }
FB22 }
```

You need to insert code at line AC07 to create the database entities.

How should you complete the relevant code? To answer, drag the appropriate code segments to the correct location. Each code segments may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

Select and Place:



Code segments

```
public class AccountContext : DbContext
```

```
public class AccountContext : IRepository
```

```
public DbSet Accounts { get; set; }
```

```
public List Accounts { get; set; }
```

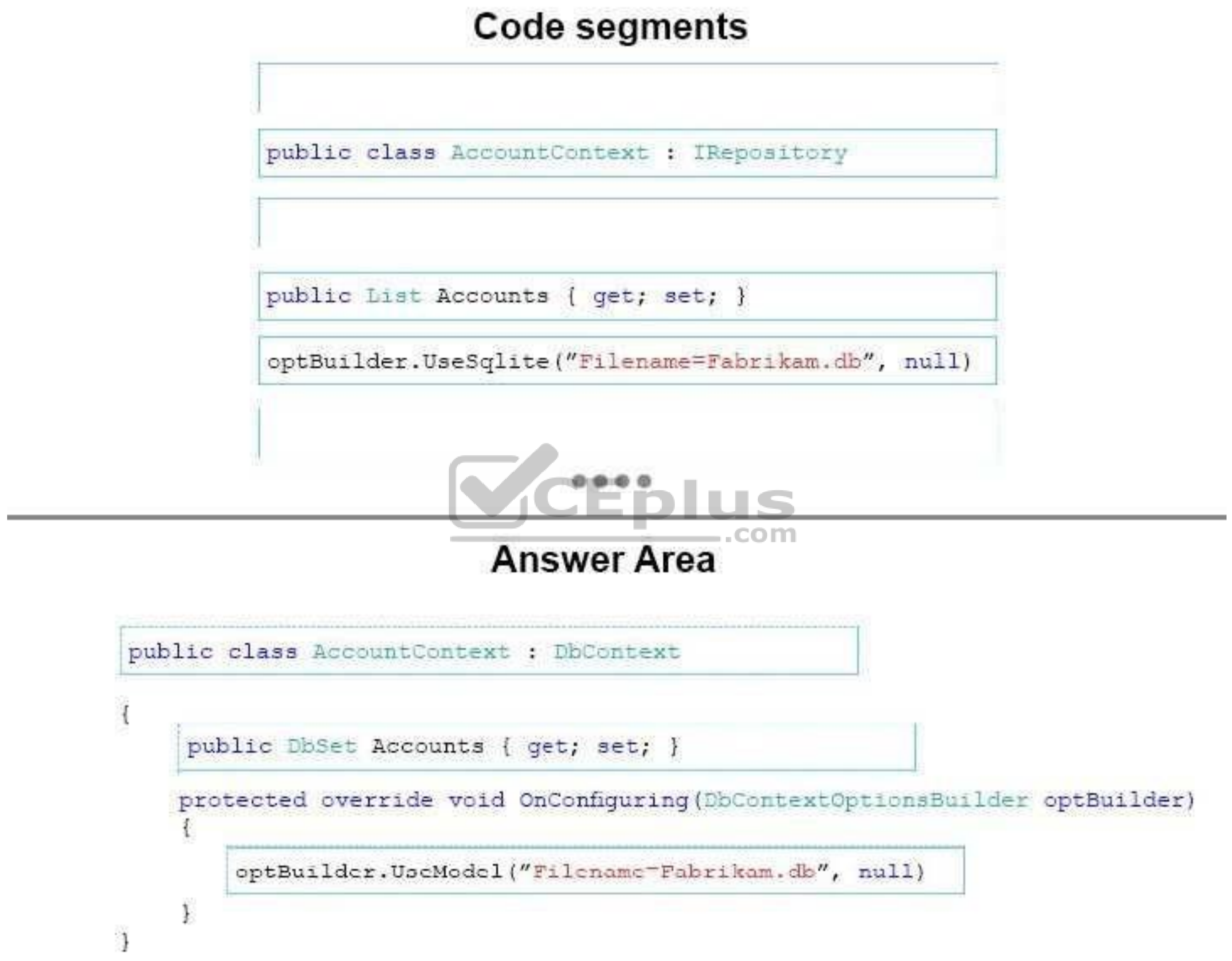
```
optBuilder.UseSqlite("Filename=Fabrikam.db", null)
```

```
optBuilder.UseModel("Filename=Fabrikam.db", null)
```

Answer Area

```
            Code segment
{
                Code segment
    protected override void OnConfiguring(DbContextOptionsBuilder optBuilder)
    {
                    Code segment
    }
}
```

Correct Answer:



Code segments

```
public class AccountContext : IRepository
```

```
public List Accounts { get; set; }
```

```
optBuilder.UseSqlite("Filename=Fabrikam.db", null)
```

Answer Area

```
public class AccountContext : DbContext
{
    public DbSet Accounts { get; set; }
    protected override void OnConfiguring(DbContextOptionsBuilder optBuilder)
    {
        optBuilder.UseModel("Filename=Fabrikam.db", null)
    }
}
```

Section: (none)
Explanation

Explanation/Reference:
Explanation:

From scenario:
The app must use a file based database. You must use a code first entity framework approach.

The DbContextOptionsBuilder Class provides a simple API surface for configuring DbContextOptions. Databases (and other extensions) typically define extension methods on this object that allow you to configure the database connection (and other options) to be used for a context.

A DbSet<TEntity> can be used to query and save instances of TEntity. LINQ queries against a DbSet<TEntity> will be translated into queries against the database.

UseModel(IModel) sets the model to be used for the context. If the model is set, then OnModelCreating(ModelBuilder) will not be run.

Reference: https://docs.microsoft.com/en-us/ef/core/api/microsoft.entityframeworkcore.dbcontextoptionsbuilder

QUESTION 27
Case Study
This is a case study. Case studies are not limited separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.
To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.
At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.
To start the case study
To display the first question on this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Background

Fabrikam is a commercial bank. The primary customers are individuals and employers with up to 10,000 employees. Fabrikam provides Internet banking services to customers.

You are developing a Universal Windows Platform (UWP) app for Fabrikam that extends the Internet banking to a UWP app.
Business Requirements

Core functionality

Users must be able to access accounts, view balances, view recent transactions, and deposit checks by using the UWP app.

Usability

The app must use industry proven design patterns across the app. All navigational elements must be visible at all times.

Security

The app must provide secure transactions to protect customer privacy.

Technical Requirements

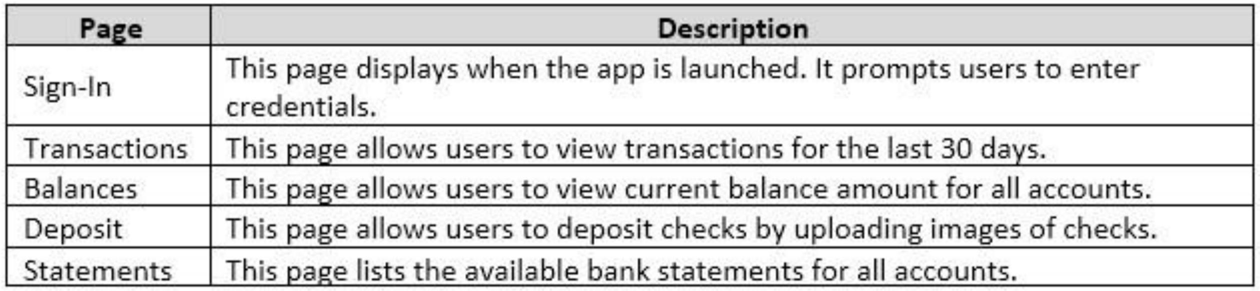
Data

The app must use a file based database. You must use a code first entity framework approach.

User interface

- You must use a Model-View-ViewModel (MVVM) pattern.
- Users must be able to access all content through the top-level navigation after they sign in.
- The app must allow the user to upload up to 50 images (front and back) of checks to deposit.
- During the upload process, the app must be responsive to any other user actions.
- The app must only upload images when no other pending inputs are in the queue.

You must implement the following pages:



Page	Description
Sign-In	This page displays when the app is launched. It prompts users to enter credentials.
Transactions	This page allows users to view transactions for the last 30 days.
Balances	This page allows users to view current balance amount for all accounts.
Deposit	This page allows users to deposit checks by uploading images of checks.
Statements	This page lists the available bank statements for all accounts.

Network and web service
The app must meet the following requirements related to networking and web services:
- Connect to a web service over a secure HTTP connection to upload images.
- Connect to Fabrikam's core web service to retrieve account information.

▪ Use networking technology already available in the .Net Framework. ▪
Consume the JSON that the Fabrikam core web service provides.

User data and alerts

The app must meet the following requirements related to user data and alerts:
▪ Download new monthly bank statements when possible. The download process must not affect the performance of the app. ▪
Report to the user when the statements are downloaded to the device.
▪ Write a log entry when statement downloads are not successful.
▪ Periodically check for user activity and automatically log the user out when there is no activity for more than 15 minutes.

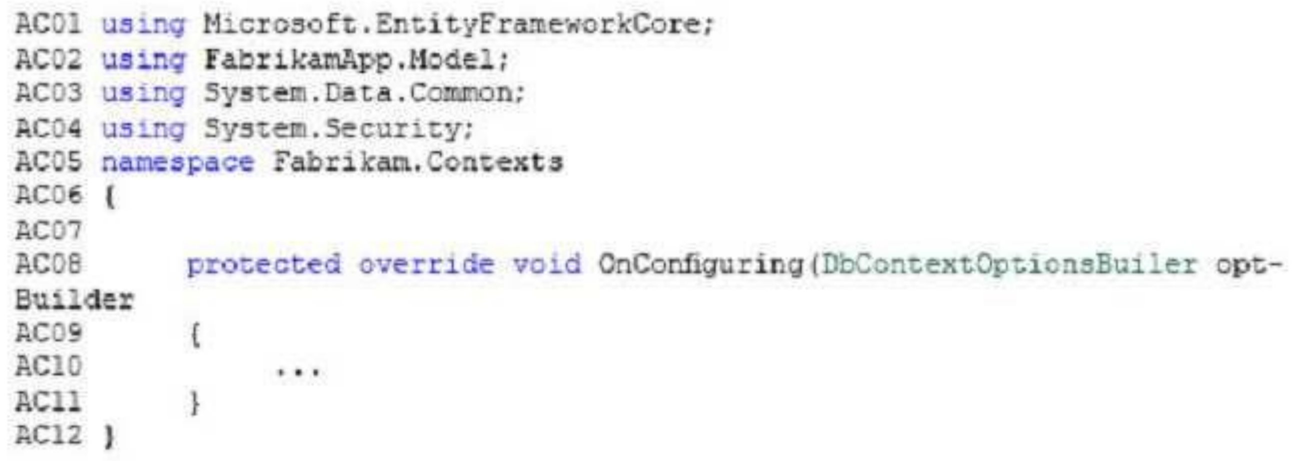
Security

The app must meet the following requirements related to security:
▪ Use a multi-factor authentication (MFA) by using email and a verification code to identify the user.
▪ Securely store credentials and retrieve credentials.
▪ Automatically sign in the user irrespective of the device that is used to sign in to the app.
▪ Store the resource name within the app itself.
▪ Connect to an authentication app by using the URI schema fabrikam-security://oauth/.

Application Structure

AccountContext.cs

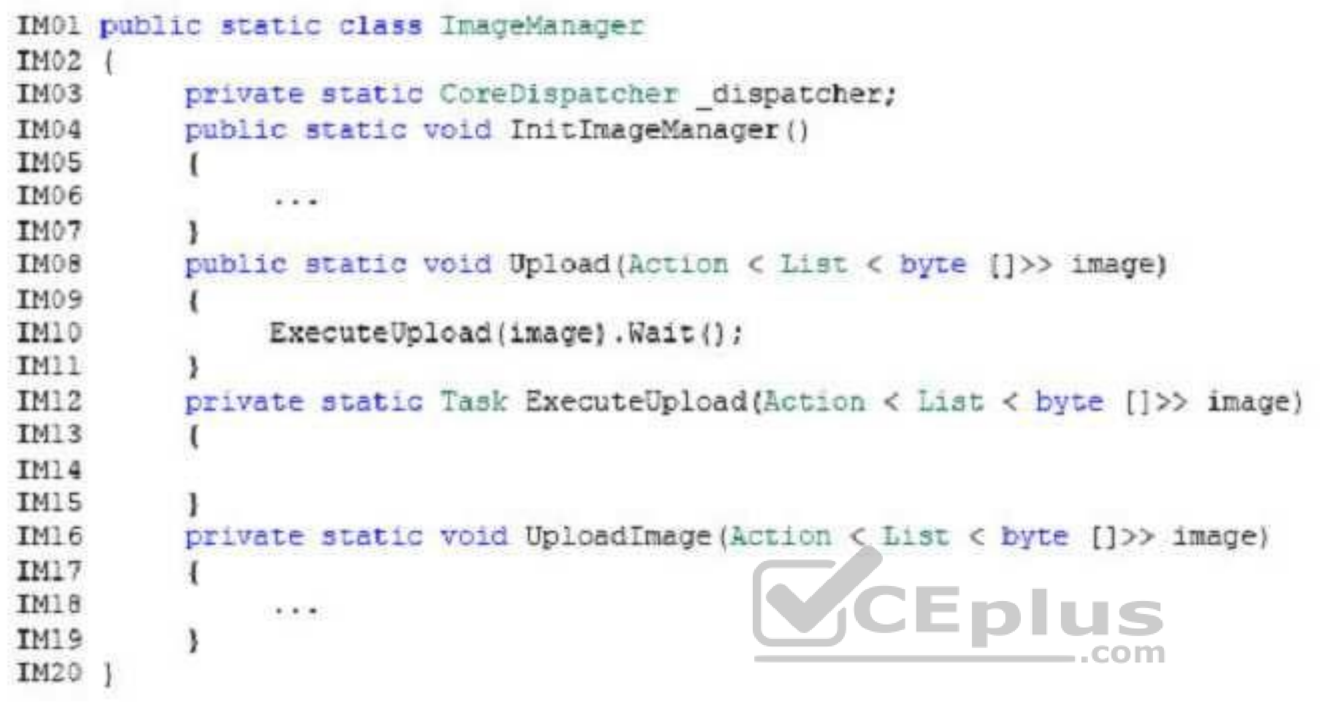
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
AC01 using Microsoft.EntityFrameworkCore;
AC02 using FabrikamApp.Model;
AC03 using System.Data.Common;
AC04 using System.Security;
AC05 namespace Fabrikam.Contexts
AC06 {
AC07
AC08        protected override void OnConfiguring(DbContextOptionsBuiler opt-
Builder
AC09        {
AC10             ...
AC11        }
AC12 }
```

ImageManager.cs

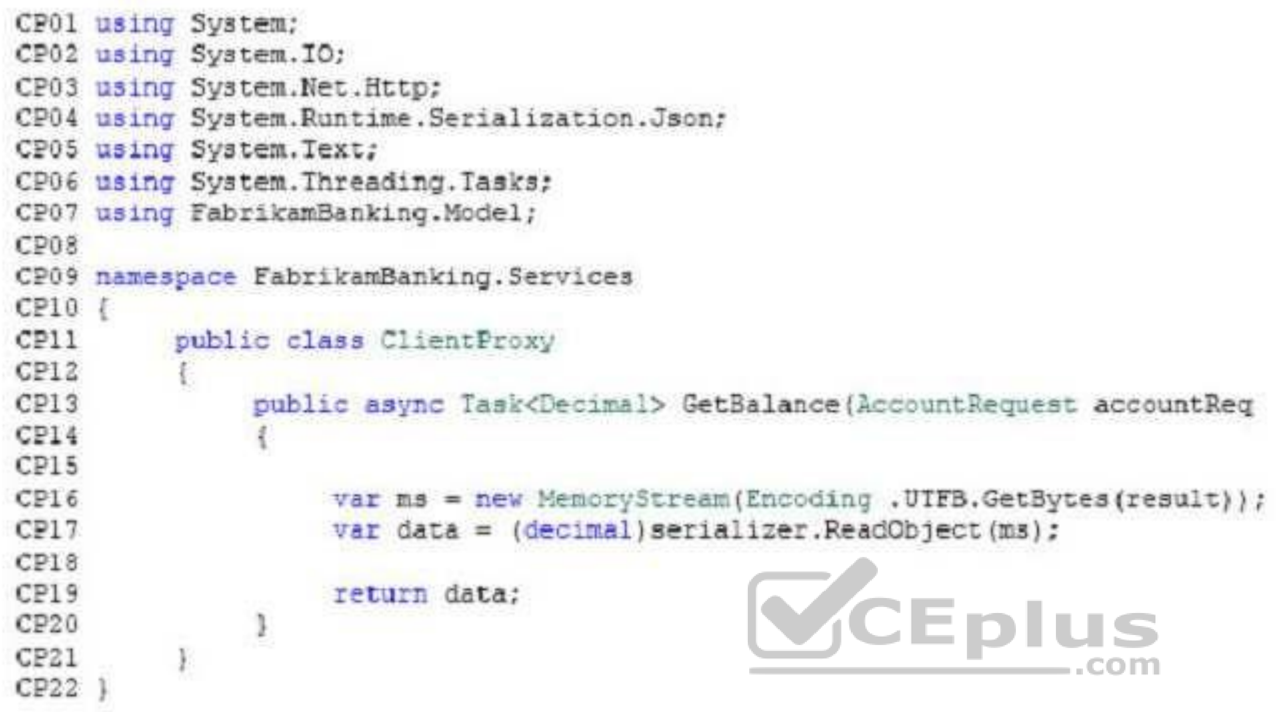
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
IM01 public static class ImageManager
IM02 {
IM03         private static CoreDispatcher _dispatcher;
IM04         public static void InitImageManager()
IM05         {
IM06             ...
IM07         }
IM08         public static void Upload(Action < List < byte []>> image)
IM09         {
IM10             ExecuteUpload(image).Wait();
IM11         }
IM12         private static Task ExecuteUpload(Action < List < byte []>> image)
IM13         {
IM14 
IM15         }
IM16         private static void UploadImage(Action < List < byte []>> image)
IM17         {
IM18             ...
IM19         }
IM20 }
```

ClientProxy.cs

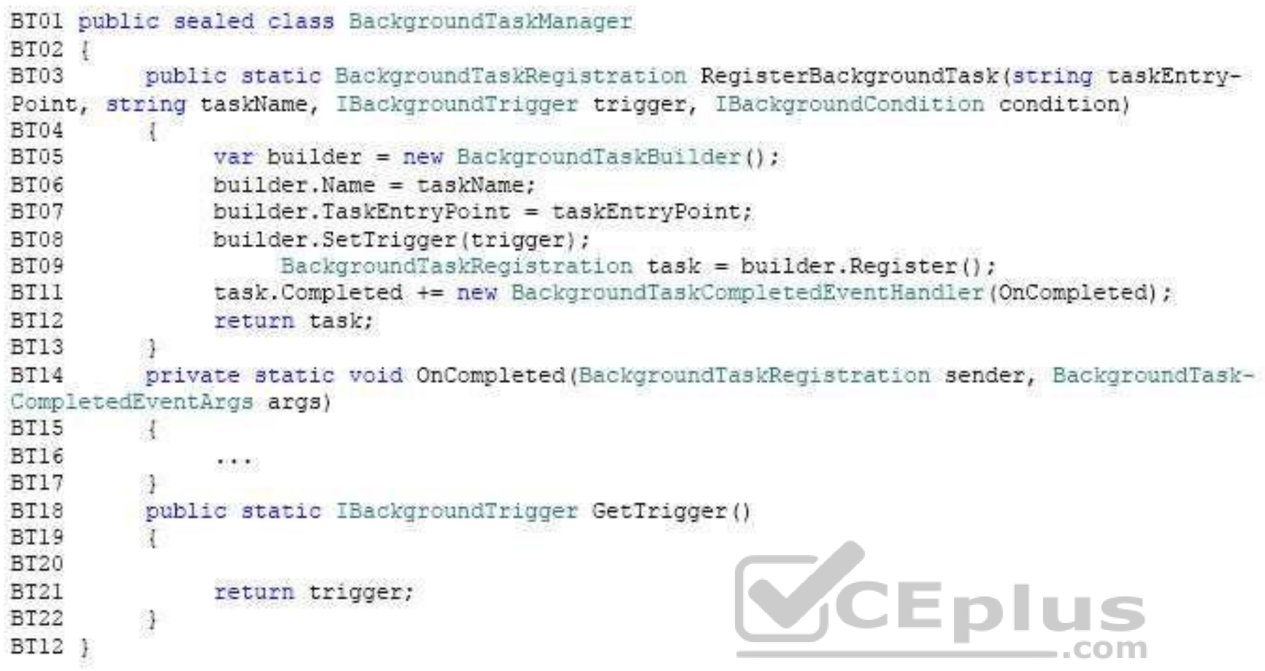
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
CP01 using System;
CP02 using System.IO;
CP03 using System.Net.Http;
CP04 using System.Runtime.Serialization.Json;
CP05 using System.Text;
CP06 using System.Threading.Tasks;
CP07 using FabrikamBanking.Model;
CP08
CP09 namespace FabrikamBanking.Services
CP10 {
CP11      public class ClientProxy
CP12      {
CP13           public async Task<Decimal> GetBalance(AccountRequest accountReq
CP14           {
CP15
CP16                var ms = new MemoryStream(Encoding .UTF8.GetBytes(result));
CP17                var data = (decimal)serializer.ReadObject(ms);
CP18
CP19                return data;
CP20           }
CP21      }
CP22 }
```

BkgTaskMgr.cs

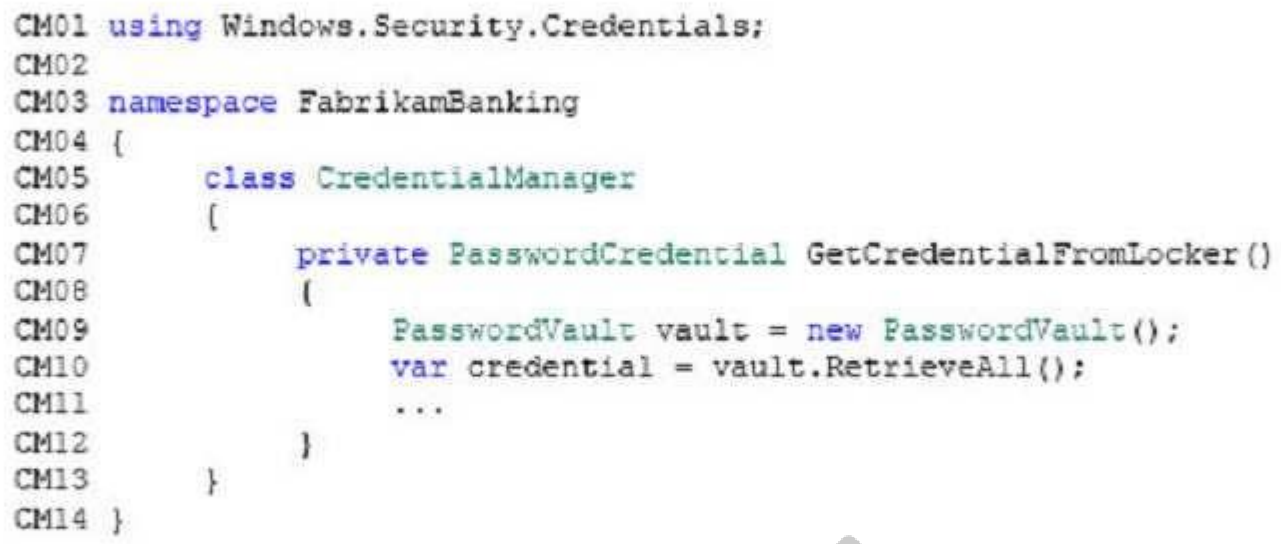
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
BT01 public sealed class BackgroundTaskManager
BT02 {
BT03      public static BackgroundTaskRegistration RegisterBackgroundTask(string taskEntry-
Point, string taskName, IBackgroundTrigger trigger, IBackgroundCondition condition)
BT04      {
BT05           var builder = new BackgroundTaskBuilder();
BT06           builder.Name = taskName;
BT07           builder.TaskEntryPoint = taskEntryPoint;
BT08           builder.SetTrigger(trigger);
BT09                BackgroundTaskRegistration task = builder.Register();
BT11           task.Completed += new BackgroundTaskCompletedEventHandler(OnCompleted);
BT12           return task;
BT13      }
BT14      private static void OnCompleted(BackgroundTaskRegistration sender, BackgroundTask-
CompletedEventArgs args)
BT15      {
BT16           ...
BT17      }
BT18      public static IBackgroundTrigger GetTrigger()
BT19      {
BT20
BT21           return trigger;
BT22      }
BT12 }
```

CredentialManager.cs

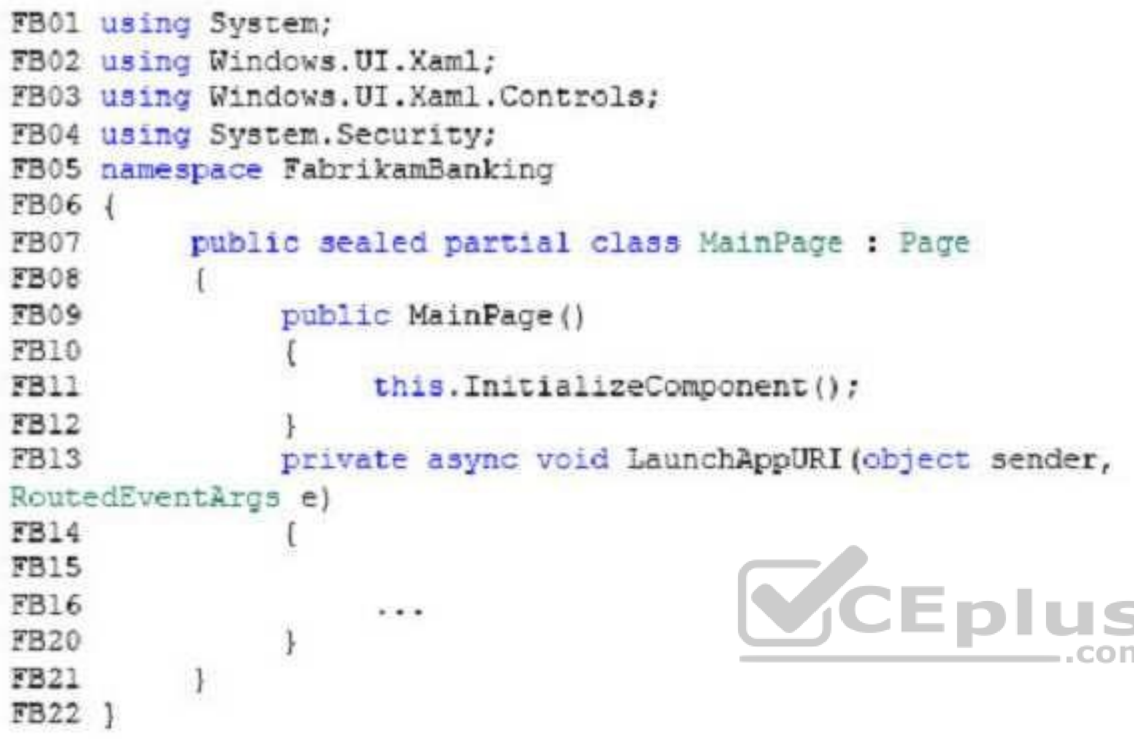
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
CM01 using Windows.Security.Credentials;
CM02
CM03 namespace FabrikamBanking
CM04 {
CM05         class CredentialManager
CM06         {
CM07                 private PasswordCredential GetCredentialFromLocker()
CM08                 {
CM09                         PasswordVault vault = new PasswordVault();
CM10                         var credential = vault.RetrieveAll();
CM11                         ...
CM12                 }
CM13         }
CM14 }
```

MainPage.cs

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.



```
using System;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using System.Security;
namespace FabrikamBanking
{
        public sealed partial class MainPage : Page
        {
                public MainPage()
                {
                        this.InitializeComponent();
                }
                private async void LaunchAppURI(object sender, RoutedEventArgs e)
                {

                        ...
                }
        }
}
```

You need to configure networking.

Which two networking technologies should you use? Each correct answer presents a complete solution.

A. Background Transfer API
B. StreamWebSocket class
C. HttpClient class
D. Custom WebSocket class
E. MessageWebSocket class

Correct Answer: AC
Section: (none)
Explanation

Explanation/Reference:

Explanation: