

Microsoft

Exam 70-357

Developing Mobile Apps

Version: Demo

[Total Questions: 10]

Topic break down

Topic	No. of Questions
Topic 1: Case Study: 1	2
Topic 2: Mix Questions	3
Topic 3: Case Study 2	4
Topic 4: Fabrikam incBackground:	1

Topic 1, Case Study: 1

Background

Business requirements

In this section, you will see one or more sets of questions with the same scenario and problem. Each question presents a unique solution to the problem, and you must determine whether the solution meets the slated goals. Any of the solutions might solve the problem. It is also possible that none o' the solutions solve the problem.

Technical requirements

Application structure

Once you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appe.ir in the review screen.

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:



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



- The Favorite button mark 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 Note 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.

Layout Requirement:

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
- User 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

New Tab:

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.

XML 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.

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.)

Microsoft 70-357 : Practice Test

```

MP01 <Page
  x: Class = "_70357rm.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/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>

```

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/winfx/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"
AS07           FontSize = "20" FontStyle = "Normal"
AS08           FontWeight = "Bold" Margin = "0 5 0 5" HorizontalAlignment="Center" />
AS09         <StackPanel Orientation = "Horizontal">
AS10           <CheckBox Content = "Center on Date" FontFamily = "Arial" FontSize="14"
AS11           FontStyle "Normal" Margin = "20,0,0,0" />
AS12           <CheckBox Content = "Set Start Date" FontFamily = "Arial" FontSize="14"
AS13           FontStyle "Normal" Margin = "20,0,0,0" />
AS14         </StackPanel>
AS15         <TextBlock Text = "Start Date" Foreground = "DarkBlue" FontFamily="Arial"
AS16           FontSize = "20" FontStyle = "Normal"
AS17           FontWeight = "Bold" Margin = "0 5 0 5" HorizontalAlignment="Center" />
AS18         <StackPanel Orientation = "Horizontal">
AS19           <TextBlock Text = "Month:" Width = "75" />
AS20           <TextBox Width = "200" />
AS21         </StackPanel>
AS22         <StackPanel Orientation = "Horizontal">
AS23           <TextBlock Text = "Day:" Width = "75" />
AS24           <TextBox Width = "200" />
AS25         </StackPanel>
AS26         <StackPanel Orientation = "Horizontal">
AS27           <TextBlock Text = "Year:" Width = "75" />
AS28           <TextBox Width = "200" />
AS29         </StackPanel>
AS30         <Ellipse Fill = "Blue" Width = "50" Height="50" Margin = "0 5 0 5" />
AS31         <TextBlock FontFamily = "Arial" FontSize = "14" Foreground="White" Text = "Save"
AS32         Margin = "127 -38 0 0" />
AS33       </StackPanel>
AS34     </Grid>
AS35   </Grid>
AS36 </Page>

```

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.)

Microsoft 70-357 : Practice Test

```

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" />
46         <Setter Property="FontWeight" Value="Bold" />
47         <Setter Property="Margin" Value="0 5 0 5" />
48         <Setter Property="HorizontalAlignment" Value="Center" />
49     </Style>
50 </ResourceDictionary>

```

New Tab:

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     }

```

Question No : 1 - (Topic 1)

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

You need to implement the appropriate XAML layout (or the Timeline app).

Solution: You create an instance of a RelativePanel class.

Does this meet the goal?

- A. Yes
- B. No

Answer: A

Explanation: RelativePanel lets you layout UI elements by specifying where they go in relation to other elements and in relation to the panel. By default, an element is positioned in the upper left corner of the panel.

Question No : 2 - (Topic 1)

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?

Microsoft 70-357 : Practice Test

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

Answer: A

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 ResourceDictionary.xaml file.

You access members of the resource dictionary like any other dictionary.

Topic 2, Mix Questions

Question No : 3 - (Topic 2)

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 that same location.

Does this meet the goal?

- A. yes
- B. No

Answer: A

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.

Question No : 4 - (Topic 2)

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 `BindingExpression.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.

Answer: B

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.

Question No : 5 DRAG DROP - (Topic 2)

You are creating a Universal Windows Platform (UWP) app that takes pictures.

You want to use the camera's built-in interface for taking the pictures.

You need to capture an image from the device's built-in camera.

How should you complete the method? 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.

Code segments

- CameraCaptureUI
- CameraCaptureUIPhotoFormat
- MediaCapture
- ImageEncodingProperties
- PhotoSettings



Answer Area

```
public async void GetImage()  
{  
    [Code segment] capture = new [Code segment] ();  
    capture.PhotoSettings.Format = CameraCaptureUIPhotoFormat.Jpeg;  
    StorageFile photo = await capture.CaptureFileAsync(CameraCaptureUIMode.Photo);  
}
```

Answer:

Microsoft 70-357 : Practice Test

Code segments

```
CameraCaptureUI  
CameraCaptureUIPhotoFormat  
MediaCapture  
ImageEncodingProperties  
PhotoSettings
```

••••

Answer Area

```
public async void GetImage()  
{  
    CameraCaptureUI capture = new CameraCaptureUI ();  
    capture.PhotoSettings.Format = CameraCaptureUIPhotoFormat.Jpeg;  
    StorageFile photo = await capture.CaptureFileAsync(CameraCaptureUIMode.Photo);  
}
```

Explanation:

Answer Area

```
public async void GetImage()  
{  
    CameraCaptureUI capture = new CameraCaptureUI ();  
    capture.PhotoSettings.Format = CameraCaptureUIPhotoFormat.Jpeg;  
    StorageFile photo = await capture.CaptureFileAsync(CameraCaptureUIMode.Photo);  
}
```

Box 1: CameraCaptureUI

Box 2: CameraCaptureUI

Example: Using Windows.Media.Capture.CameraCaptureUI API to capture a photo

CameraCaptureUI dialog = new CameraCaptureUI();

Size aspectRatio = new Size(16, 9);

dialog.PhotoSettings.CroppedAspectRatio = aspectRatio;

StorageFile file = await dialog.CaptureFileAsync(CameraCaptureUIMode.Photo);

Topic 3, Case Study 2

Background

You are developing a Universal Windows Platform (UWP) app for LitWare, Inc. that will assist video artists. The app allows artists to create videos, share videos through other mobile apps, and upload the videos through LitWare, Inc's web services. What helps set

LitWare Inc's app apart from competitors is their focus on speed and performance.

Business requirements

Support many devices

. Users may have phones, tablets, or laptops. The app must support all devices with a fluid layout that grows off-screen and adapts to each device.

Record video

- ✍ Users must be able to record videos and view them in their videos library.
- ✍ The app must display information about the recorded video.
- ✍ Users must be able to edit the upload queue in the app.

Branding

The app must allow deferral and scheduling of video uploads.

Users must be able to view the status of video uploads.

Any videos created with this app or shared with this app must be uploaded without user interaction.

Download video

The app must have the option to automatically download videos.

Users must be able to initiate downloading of videos.

Share video

The app must allow be allowed to receive videos from other apps.

- ✍ Technical requirements
- ✍ Application structure

Technical requirements

Support multiple devices

The app must support the following:

- ✍ Use horizontal layout for larger screens.
- ✍ Use vertical layout for smaller screens.
- ✍ Use one layout control per view.

The app must be compatible with current and future XBOX app that use C++.

Uploading and downloading

The app must use a background operations to upload and download videos.

Code reuse

- ✍ The app must use a common pool of XAML resources and custom controls. All custom controls must use a consistent theme throughout the app.
- ✍ You must create code that can be reused in C++, C#, JavaScript, whenever possible.
- ✍ The app must call the background service to avoid duplication of code.

Security

- ✍ End users must be authenticated using OAuth.
- ✍ Web services must be authenticated.
- ✍ Users must have the option to use single sign-on.

Recording

The app must use the microphone and webcam to support audio and video recording. In addition, the app must support the use of the back camera buttons, if present.

Integration:

Other apps must be able to share videos with this app through a Universal Windows Platform (UWP)

Architecture and coding style

- ✍ The app must follow the Model-View-ViewModel (MVVM) pattern

Microsoft 70-357 : Practice Test

- ✍ The app's user interface (UI) must be optimized for performance.
- ✍ The app must use compile time coding techniques over runtime.

Package appxmanifest

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.

```

PM01 <Package>
PM02     <Applications>
PM03     <Application id="App" Executable=" $targetnametoken$.exe" EntryPoint="Litware.-
MultiMediaApp.App">
PM04         <Extensions>
PM05             <Extension Category = "windows.appService"
                EntryPoint="Litware.MultiMediaApp.VideoUploadService.VideoUpload">
PM06
PM07             </Extension>
PM08         <Extensions>
PM09             <uap:VisualElements DisplayName="Litware.MultiMediaApp"
Square150x150Logo=" Assets\Square150x150Logo.png"
                Square44x44Logo = "Assets\Square44x44Logo.png" Description="
Litware.MultiMediaApp"
                BackgroundColor = "transparent" >
PM10             <uap:DefaultTile Wide310x150Logo="Assets\Wide310x150Logo.png">
PM11             </uap:DefaultTile>
PM12             <uap:SplashScreen Image = "Assets\SplashScreen.png" />
PM13             </uap:VisualElements>
PM14         </Application>
PM15     </Applications>
PM16 </Package>

```

Background Task

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.

```

BA01 private void RegisterBackgroundTask(object sender, RoutedEventArgs e)
BA02 {
BA03     if (! this.IsAlreadyRegistered(this .videoDownloadBackgroundTaskName))
BA04     {
BA05         var backgroundTaskRegistration = this.RegisterBackgroundTask(
BA06         this .videoDownloadBackgroundTaskName,
BA07         this .videoDownloadBackgroundTaskEntryPoint,
BA08
BA09         null);
BA10         backgroundTaskRegistration.Progress += OnProgress;
BA11         backgroundTaskRegistration.Completed += OnCompleted;
BA12     }
BA13 }

```

Question No : 6 - (Topic 3)

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 must create a project for shared code.

Solution: You implement the shared code in a Windows Runtime component.

Does this meet the goal?

- A. Yes
- B. No

Answer: A

Question No : 7 - (Topic 3)

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 must create a project for shared code.

Solution: You implement the shared code in a Shared Project.

Does this meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

The .NET Framework Portable Class Library project type in Visual Studio helps you build cross-platform apps and libraries for Microsoft platforms quickly and easily.

Portable class libraries can help you reduce the time and costs of developing and testing code. Use this project type to write and build portable .NET Framework assemblies, and then reference those assemblies from apps that target multiple platforms such as Windows and Windows Phone.

Even after you create a Portable Class Library project in Visual Studio and start developing it, you can change the target platforms. Visual Studio will compile your library with the new assemblies, which helps you identify the changes you need to make in your code.

From scenario:

The app must be compatible with current and future XBOX apps that use C++.

Question No : 8 - (Topic 3)

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 must create a project for shared code.

Solution: You implement the shared code in a Class Library (Universal Windows).

Does this meet the goal?

- A. Yes
- B. No

Answer: A

Explanation:

The .NET Framework Portable Class Library project type in Visual Studio helps you build cross-platform apps and libraries for Microsoft platforms quickly and easily.

Portable class libraries can help you reduce the time and costs of developing and testing code. Use this project type to write and build portable .NET Framework assemblies, and then reference those assemblies from apps that target multiple platforms such as Windows and Windows Phone.

Even after you create a Portable Class Library project in Visual Studio and start developing it, you can change the target platforms. Visual Studio will compile your library with the new assemblies, which helps you identify the changes you need to make in your code.

From scenario:

The app must be compatible with current and future XBOX apps that use C++.

Question No : 9 - (Topic 3)

You are a developer for LitWare,Inc.'Universal Windows Platform (UWP) app.

Access to the hardware within the app is not functioning correctly.

You need to add the capabilities to the package.appmanifest file.

Which markup segment should you insert at line PM16?

A

```
<Capabilities>
  <Capability Name = "internetClient" />
  <uap:Capability Name = "videosLibrary" />
  <DeviceCapability Name = "microphone" />
  <DeviceCapability Name ="location" />
</Capabilities>
```

B

```
<Capabilities>
  <uap:Capability Name = "videosLibrary" />
  <uap:Capability Name = "removableStorage" />
  <DeviceCapability Name = "microphone" />
  <DeviceCapability Name ="webcam" />
</Capabilities>
```

C

```
<Capabilities>
  <Capability Name = "internetClient" />
  <uap:Capability Name = "picturesLibrary" />
  <DeviceCapability Name = "microphone" />
  <DeviceCapability Name ="webcam" />
</Capabilities>
```

D

```
<Capabilities>
  <Capability Name = "internetClient" />
  <uap:Capability Name = "vodeosLibrary" />
  <DeviceCapability Name = "microphone" />
  <DeviceCapability Name ="webcam" />
</Capabilities>
```

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

Answer: D

Explanation: From scenario:

✍ The app must use the microphone and webcam to support audio and video

recording. In addition, the app must support the use of the back and camera buttons, if present.

 Users must be able to record videos and view them in their videos library.

Topic 4, Fabrikam incBackground:

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.

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 Fabrikame'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 statements 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.

Security 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/.

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 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(DbContextOptionsBuilder opt-
Builder
AC09     {
AC10         ...
AC11     }
AC12 }
```

ImageMeanger.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 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 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

Microsoft 70-357 : Practice Test

```

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();
BT10         task.Completed += new BackgroundTaskCompletedEventHandler(OnCompleted);
BT11         return task;
BT12     }
BT13     private static void OnCompleted(BackgroundTaskRegistration sender, BackgroundTask-
CompletedEventArgs args)
BT14     {
BT15         ...
BT16     }
BT17     public static IBackgroundTrigger GetTrigger()
BT18     {
BT19         return trigger;
BT20     }
BT21 }
BT22 }
BT23 }

```

CredentialManagers.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 prefix that denotes the specific file to which they belong.

```


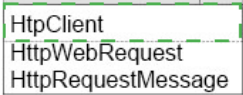

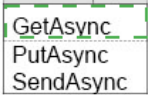
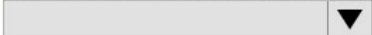

CM01 using Windows.Security.Credentials;
CM02
CM03 namespace FabrikamBanking
CM04 {
CM05     class CredentialManager
CM06     {
CM07         private PasswordCredential GetCredentialFromLocker()
CM08         {
CM09             PasswordVault vault = new PasswordVault();
CM10             var credential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

```

MainPage.cs


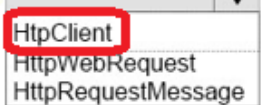

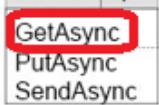

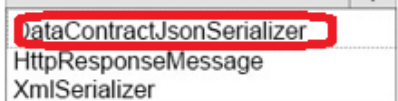
Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that prefix that denotes the specific file to which they belong.

Answer Area

```
var objRequest = new  ();  
      
var response = await objRequest.  
      
( "http://core.fabrikam.com/accounts/balance?number=" +  
accountReq.AccountInfo.Number + @&token=" + accountReq.Token);  
var result = await response.Content. ReadAsStringAsync ();  
var serializer = new  (typeof (decimal));  
    
```

Explanation:

Answer Area

```
var objRequest = new  ();  
      
var response = await objRequest.  
      
( "http://core.fabrikam.com/accounts/balance?number=" +  
accountReq.AccountInfo.Number + @&token=" + accountReq.Token);  
var result = await response.Content. ReadAsStringAsync ();  
var serializer = new  (typeof (decimal));  
    
```

From scenario:

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

Box 1: HttpClient

Box 2: GetAsync

Box 3: DataContractJsonSerializer

From scenario: Consume the JSON that the Fabrikam core web service provides.

Thank You For Trying 70-357 PDF Demo

To try our 70-357 Premium Files visit link below:

<https://examsland.com/latest-exam-questions/70-357/>

Start Your 70-357 Preparation

Use Coupon **EL25 for extra 25% discount on the purchase of Practice Test Software.**