Exam 70-357: Developing Mobile Apps – Skills Measured

Audience Profile

Candidates for this exam are experienced developers who design and develop Windows 10 apps for deployment to either the Windows Store or Windows Store for Business.

Candidates typically have more than two years of experience developing Windows apps using C# and XAML, in addition to experience with WPF or Silverlight for Windows Phone. Candidates should also have experience with the Model-View-ViewModel (MVVM) design pattern and Entity Framework, along with authentication technologies. Visual Studio 2015 experience is highly recommended.

Skills Measured

NOTE: The bullets that appear below each of the skills measured are intended to illustrate how we are assessing that skill. This list is not definitive or exhaustive.

NOTE: In most cases, exams do NOT cover preview features, and some features will only be added to an exam when they are GA (General Availability).

Develop a XAML page layout for an adaptive UI (10–15%)

Construct a page layout

- configure a RelativePanel layout; select the appropriate XAML layout panel based on the UI requirement; configure a grid with appropriate column and row properties; configure alignment, margins, and padding

Implement responsive and adaptive UI behaviors

- differentiate between responsive and adaptive UI behaviors, create responsive and adaptive UIs by using VisualStateManager and AdaptiveTriggers, implement settings syntax for element properties and attached properties

Create and use custom controls within an adaptive UI

- evaluate when to create a custom control; create a custom control; implement styles, themes, and resource dictionaries; apply styles to custom controls by using Generic.xaml

Optimize a page layout

- reduce complexity for performance gains, reduce unnecessary nesting

Implement page navigation and lifecycle events (10–15%)

Choose the appropriate navigation structure for an app

- evaluate when to implement the Hub, Master/Details, Tabs and Pivot, and Nav Pane navigation patterns; evaluate when to implement a custom navigation pattern

Implement Nav Pane navigation

- load page content by using Frame.Navigate, implement page navigation by using the Nav Pane pattern; implement a SplitView control for use as a navigation pane; support accessibility requirements within navigation by implementing key based navigation, UI automation, and narrator; handle Back button behavior for different Windows 10 device families

Manage app activation

- launch an app, activate an app on Startup, implement activation from a deep link, implement activation based on Search integration, implement activation from a secondary tile

Manage app suspension and resuming

- prepare an app for suspension, resume from suspension or termination, extend execution and monitor suspension errors

Implement data access and data binding (20–25%)

Access data by using Entity Framework (EF)

- access data by using EFCore with SQLite, implement a local SQLite database

Implement the {Binding} extension

Implement the {x:Bind} extension

Implement MVVM classes and class interactions

- implement event binding by applying command patterns, implement a Dispatcher to update the UI thread with async return data

Implement app-to-app communications

- integrate a Share contract to share content with another app, integrate drag-and-drop, launch an app for results, implement app extensions, implement App Services

Implement REST Web Services

- implement JSON and data serialization, access cloud data and Web APIs by using HttpClient

Implement file system access

- manage storage by using StorageFile, StorageFolder, and StorageItem; access a file location by using FilePickers; implement data roaming and roaming folders

Implement feature detection for adaptive coding (10–15%)

Implement API detection within adaptive code

Implement Type detection within adaptive code

Implement supported capabilities

- implement support for a microphone, implement support for a webcam, implement support for location, implement support for enterprise authentication

Manage user input and custom user interactions (10–15%)

Implement command bars, flyouts, and dialogs

- implement command bars and AppBarButton buttons, implement context menus and menu flyouts, implement content dialogs, display a tooltip by using ToolTipService, display a pop-up menu, implement control over app settings

Implement support for traditional and touch input devices

- support touch input, support mouse input, support keyboard and virtual keyboard input

Implement speech and voice commands

- support speech synthesis, support speech recognition, support Cortana integration, support Personal Assistant Launch capability, support voice commands

Implement alternative forms of input

- implement inking, implement camera input, implement location services and GPS input

Manage authentication and identity management (10–15%)

Implement authentication using Web Authentication Broker

- implement web service authentication, implement OAuth, implement Azure Active Directory authentication

Manage credentials securely with Credential Locker

Implement two-factor authentication

- implement two-factor authentication using Microsoft Passport, implement two-factor authentication using Windows Hello

Implement notifications, background tasks, and reusable components (15–20%)

Create and consume class libraries and Windows Runtime components

- develop Windows Runtime components, develop class libraries, integrate class libraries and Windows Runtime components

Implement tile and toast notifications

- implement adaptive and interactive toast notifications, implement local tile notifications

Create and register a background task

- create a background task project and reference the background task within a project, implement background task event triggers and conditions

Implement and manage a background task

- monitor background task progress and completion, manage task lifecycle, share data and events between an app and its background tasks, call a background task directly

Create and consume a Universal Windows Platform (UWP) app service

- specify the AppService extension, implement app service as a background task, deploy the app service provider, call app services