MCSA: Universal Windows Platform – 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).

Exam 70-483: Programming in C#

Manage Program Flow (25-30%)

Implement multithreading and asynchronous processing

- use the Task Parallel library, including theParallel.For method, PLINQ, Tasks;create continuation tasks; spawn threads by using ThreadPool; unblock the UI; use async and await keywords; manage data by using concurrent collections

Manage multithreading

- synchronize resources; implement locking; cancel a long-running task; implement thread-safe methods to handle race conditions

Implement program flow

- iterate across collection and array items; program decisions by using switch statements, if/then, and operators; evaluate expressions

Create and implement events and callbacks

- create event handlers; subscribe to and unsubscribe from events; use built-in delegate types to create events; create delegates; lambda expressions; anonymous methods

Implement exception handling

- handle exception types, including SQL exceptions, network exceptions, communication exceptions, network timeout exceptions; use catch statements; use base class of an exception; implement try-catchfinally blocks; throw exceptions; rethrow an exception; create custom exceptions; handle inner exceptions; handle aggregate exception

Create and Use Types (25-30%)

Create types

- create value types, including structs and enum; create reference types, generic types, constructors, static variables, methods, classes, extension methods; create optional and named parameters; create indexed properties; create overloaded and overriden methods

Consume types

- box or unbox to convert between value types; cast types; convert types; handle dynamic types; ensure interoperability with code that accesses COM APIs

Enforce encapsulation

- enforce encapsulation by using properties; enforce encapsulation by using accessors, including public, private, protected, and internal; enforce encapsulation by using explicit interface implementation

Create and implement a class hierarchy

- design and implement an interface; inherit from a base class; create and implement classes based on the IComparable, IEnumerable, IDisposable, and IUnknown interfaces

Find, execute, and create types at runtime by using reflection

- create and apply attributes; read attributes; generate code at runtime by using CodeDom and Lambda expressions; use types from the System.Reflection namespace, including Assembly, PropertyInfo, MethodInfo, Type

Manage the object life cycle

- manage unmanaged resources; implement IDisposable, including interaction with finalization; manage IDisposable by using the Using statement; manage finalization and garbage collection

Manipulate strings

- manipulate strings by using the StringBuilder, StringWriter, and StringReader classes; search strings; enumerate string methods; format strings; use string interpolation

Debug Applications and Implement Security (25-30%)

Validate application input

- validate JSON data; choose the appropriate data collection type; manage data integrity; evaluate a regular expression to validate the input format; use built-in functions to validate data type and content

Perform symmetric and asymmetric encryption

- choose an appropriate encryption algorithm; manage and create certificates; implement key management; implement the System.Security namespace; hash data; encrypt streams

Manage assemblies

- version assemblies; sign assemblies using strong names; implement side-by-side hosting; put an assembly in the global assembly cache; create a WinMD assembly

Debug an application

- create and manage preprocessor directives; choose an appropriate build type; manage program database files (debug symbols)

Implement diagnostics in an application

- implement logging and tracing; profiling applications; create and monitor performance counters; write to the event log

Implement Data Access (25-30%)

Perform I/O operations

- read and write files and streams; read and write from the network by using classes in the System.Net namespace; implement asynchronous I/O operations

Consume data

- retrieve data from a database; update data in a database; consume JSON and XML data; retrieve data by using web services

Query and manipulate data and objects by using LINQ

- query data by using operators, including projection, join, group, take, skip, aggregate; create methodbased LINQ queries; query data by using query comprehension syntax; select data by using anonymous types; force execution of a query; read, filter, create, and modify data structures by using LINQ to XML

Serialize and deserialize data

- serialize and deserialize data by using binary serialization, custom serialization, XML Serializer, JSON Serializer, and Data Contract Serializer

Store data in and retrieve data from collections

- store and retrieve data by using dictionaries, arrays, lists, sets, and queues; choose a collection type; initialize a collection; add and remove items from a collection; use typed vs. non-typed collections; implement custom collections; implement collection interfaces

Exam 70-357: Developing Mobile Apps

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