MTA: Software Development Fundamentals – 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 98-361: Software Development Fundamentals

Understanding core programming (15-20%)

Understand computer storage and data types

- how a computer stores programs and the instructions in computer memory, memory stacks and heaps, memory size requirements for the various data storage types, numeric data and textual data

Understand computer decision structures

- various decision structures used in all computer programming languages; If decision structures; multiple decision structures, such as If…Else and switch/Select Case; reading flowcharts; decision tables; evaluating expressions

Identify the appropriate method for handling repetition

- For loops, While loops, Do...While loops, and recursion

Understand error handling

- structured exception handling

Understanding object-oriented programming (20-25%)

Understand the fundamentals of classes

- properties, methods, events, and constructors; how to create a class; how to use classes in code

Understand inheritance

- inheriting the functionality of a base class into a derived class

Understand polymorphism

- extending the functionality in a class after inheriting from a base class, overriding methods in the derived class

Understand encapsulation

- creating classes that hide their implementation details while still allowing access to the required functionality through the interface, access modifiers

Understanding general software development (15-20%)

Understand application life cycle management

- phases of application life cycle management, software testing

Interpret application specifications

- reading application specifications and translating them into prototypes, code, select appropriate application type, and components

Understand algorithms and data structures

- arrays, stacks, queues, linked lists, and sorting algorithms; performance implications of various data structures; choosing the right data structure

Understanding web applications (15-20%)

Understand web page development

- HTML, Cascading Style Sheets (CSS), JavaScript

Understand Microsoft ASP.NET web application development

- page life cycle, event model, state management, client-side versus server-side programming

Understand web hosting

- creating virtual directories and websites, deploying web applications, understanding the role of Internet Information Services

Understand web services

- web services that will be consumed by client applications, accessing web services from a client application, SOAP and Web Service Definition Language (WSDL)

Understanding desktop applications (15-20%)

Understand Windows apps

- UI design guideline categories, characteristics and capabilities of Store Apps, identify gestures

Understand console-based applications

- characteristics and capabilities of console-based applications

Understand Windows Services

- characteristics and capabilities of Windows Services

Understanding databases (15-20%)

Understand relational database management systems

- characteristics and capabilities of database products, database design, Entity Relationship Diagrams (ERDs), normalization concepts

Understand database query methods

- Structured query language (SQL), creating and accessing stored procedures, updating data and selecting data

Understand database connection methods

- connecting to various types of data stores, such as flat file; XML file; in-memory object; resource optimization