



Pro Designing and Developing Windows Applications Using Microsoft .NET Framework 4 Exam 70-518

About this Exam

This exam is designed to test the candidate's knowledge and skills on making the appropriate job role decisions around Windows-based applications on the desktop using Windows Forms and WPF.

Questions that contain code will be presented in either VB or C#. Candidates can select one of these languages when they start the exam.

Audience Profile

- Candidates for this exam work on a team in a development environment that uses Microsoft Visual Studio .NET 2010 and the Microsoft .NET Framework 4 to develop desktop applications.
- Candidates should have a minimum of three years of experience developing applications, including one to two years of experience developing Windows-based applications.
- Candidates should have a thorough understanding of Windows Presentation Foundation (WPF) and Windows Forms technologies in the .NET Framework 3.5 and 4.
- Additionally, candidates should be able to demonstrate the following by using the .NET Framework 4:
 - Experience designing Windows client applications that access data and services
 - Experience designing data access layers and service layers for a Windows client application
 - Experience planning and designing user interaction solutions
 - Experience with the full development life cycle of Windows client applications
 - Experience developing and deploying to multi-tier environments



How long is this course?

2KO International does not currently present this course as a scheduled course but we can offer this training as a block booking or in-house over five days

Where does the course take place?

At our training centre in Greenpoint, Cape Town, South Africa.

Or we can run it in-house at your business, anywhere in the world (requires individualised quotation).



We aim to offer training at the best prices – if you get a cheaper quote than us, send it to us (official quotes only) and we will beat it. Simple!

Skills Being Measured

This exam measures your ability to accomplish the technical tasks listed below. The percentages indicate the relative weight of each major topic area on the exam.

Designing the Layers of a Solution (22%)

- Design a loosely coupled layered architecture.
This objective may include but is not limited to: separation of concerns including presentation, business logic, and data; minimizing logical dependencies; deciding how layers connect (e.g., content-based vs. context-based filtered routing)
- Design service interaction.
This objective may include but is not limited to: service granularity (cohesiveness); interface granularity (responsibilities of an operation), versioning, data and service contracts (using a message contract rather than a data contract); hosting and protocol; managing data integrity (re-validating data across trust boundaries); evaluating use of RESTful interface (URI/resource structure); choosing a message exchange pattern; choosing synchronous vs. asynchronous operation invocation; deciding whether to use custom SOAP headers
This objective does not include: interacting with existing/external systems
- Design the security implementation.
This objective may include but is not limited to: protecting data during transmission (encryption/hashing algorithm), authentication (client/proxy credential) and authorization (groups, built-in or custom role provider, claims, federated security), designing for least privilege (impersonation and/or delegation), planning for User Access Control (UAC) environments; auditing
- Design for interoperability with external systems.
This objective may include but is not limited to: choosing an appropriate strategy for communicating with COM components, mainframe services, and Web services
- Design for optimal processing.
This objective may include but is not limited to: parallel processing; asynchronous processing; service bus; gateway processes; scalability (scale out vs. scale up); designing tiers to minimize latency (batch retrieval, multiple small calls)
- Design for globalization and localization.
This objective may include but is not limited to: multi-locale services; designing for time zone, sorting, UI considerations; database design considerations

Designing the Presentation Layer (21%)



- Choose the appropriate Windows Client technology.
This objective may include but is not limited to: choosing between Windows Forms, WPF, or a combination; choosing an appropriate presentation pattern (Model View Presenter [MVP], Model View/View Model [MV-VM]); identifying areas for possible migration/interoperability from Windows Forms to WPF
- Design the UI layout and structure.
This objective may include but is not limited to: evaluating the conceptual design, deciding how the UI will be composed (e.g., static vs. dynamic screen); designing for the inheritance and re-use of visual elements (e.g., styles, resources); accessibility considerations; deciding when custom controls are needed
- Design application workflow.
This objective may include but is not limited to: user navigation, designing wizards, modal vs. non-modal; dependencies among UI elements; designing for input types based on environment and audience (kiosk, very large display, small display, indoors and outdoors)
- Design data presentation and input.
This objective may include but is not limited to: designing data validation; designing a data-binding strategy; designing a reporting strategy; choosing media services (audio, video, images, animation); managing data shared between forms
- Design presentation behavior.
This objective may include but is not limited to: determining which behaviors will be implemented and how; drag and drop functionality
- Design for UI responsiveness.
This objective may include but is not limited to: offloading operations from UI thread and reporting of progress, avoiding unnecessary screen refresh; media buffering; client vs. server side sorting and filtering of data; addressing UI memory issues

Designing the Data Access Layer (21%)

- Choose the appropriate data access strategy.
This objective may include but is not limited to: choosing the appropriate data access technology (Entity Framework, LINQ to SQL, Microsoft ADO.NET); supporting data sources such as XML data, flat files, and relational databases
- Design the data object model.
This objective may include but is not limited to: mapping to persistent storage (mapping to tables, XML files), abstracting from the service layer (encapsulating underlying schema details); designing a schema change management strategy
- Design data caching.
This objective may include but is not limited to: managing data cache (lifetime, targets, size, scope), managing data state (change notification, cache invalidation/synchronization)
- Design offline storage and data synchronization.
This objective may include but is not limited to: managing offline data, mapping a data store to local cache, designing synchronization; analyzing target data



environment (e.g., Microsoft SQL Server, SQL Express, workstation capabilities, OS, bandwidth, reliability)

- Design for a concurrent multi-user environment.
This objective may include but is not limited to: planning for concurrency and collision avoidance, optimistic vs. pessimistic locking, cross-tier distributed transactions
- Analyze data services for optimization.
This objective may include but is not limited to: object relational mapping (ORM) performance, optimizing roundtrips, lazy vs. eager loading, caching of frequently used data

Planning a Solution Deployment (17%)

- Define a client deployment strategy.
This objective may include but is not limited to: recommending an installation method (Xcopy, ClickOnce, MSI, third party); identifying prerequisites (target framework and bootstrap requirements), deploying COM components
- Plan a database deployment.
This objective may include but is not limited to: existing or shared instance; remote server; embedded database; deploying new objects (such as tables, stored procedures, and views) to a new or existing database; recognizing database security concerns (such as shared instances and access); remote vs. local database
This objective does not include: DLL deployment; shared GAC deployment
- Design a solution update strategy.
This objective may include but is not limited to: preserving shared components, data integrity, and user customizations; designing an update delivery method (e.g., automated update detection from the client), version mismatch (both local binaries and service interfaces)
- Plan for n-tier deployment.
This objective may include but is not limited to: mapping the solution to the topology (required hardware such as servers, routers, and RAM and required software such as OS); determining component installation order; reviewing security requirements

Designing for Stability and Maintenance (19%)

- Design for error handling.
This objective may include but is not limited to: collecting user feedback when errors occur, handling exceptions across tiers
This objective does not include: try/catch blocks
- Evaluate and recommend a test strategy.
This objective may include but is not limited to: recommending functional testing, recommending reliability testing (performance testing, stress testing, scalability testing, duration testing)
This objective does not include: unit testing



Course Outline 2011/2012

- Design a diagnostics and monitoring strategy.
- This objective may include but is not limited to: profiling, tracing, performance counters, audit trails (events and information); usage reporting; deciding where to log events (local vs. centralized reporting)