Sample Question from Chapter 2.

1.What is .net core? Benifits of .net core application?

Developers describe **ASP.NET Core** as *"A cross-platform .NET framework for building modern cloud-based web applications on Windows, Mac, or Linux".* A free and open-source web framework, and higher performance than ASP.NET, developed by Microsoft and the community. It is a modular framework that runs on both the full .NET Framework, on Windows, and the cross-platform .NET Core.

Major Benefits of ASP.Net Core:

- Asp.Net core is a much leaner and modular framework because of multiple of architecture
- Asp.net Core is an open-source framework.
- Easy to build cross-platform asp.net app on Windows, Mac, and Linux.
- The configuration is a cloud-ready environment.
- Ability to host on:

  A) Kestral

  B) IIS

  C) HTTP.sys

  D) Nginx

  E) Apache

  F) Docker

2. List out different Characteristics of .net core?

Open-source Framework: .NET Core is an open-source framework maintained by Microsoft and available on GitHub under MIT and Apache 2 licenses. It is a .NET Foundation project.

 Cross-platform: .NET Core runs on Windows, macOS, and Linux operating systems. There are different runtime for each operating system that executes the code and generates the same output.

 Consistent across Architectures: Execute the code with the same behavior in different instruction set architectures, including x64, x86, and ARM.

 Wide-range of Applications: Various types of applications can be developed and run on .NET Core platform such as mobile, desktop, web, cloud, IoT, machine learning, microservices, game, etc.

 Supports Multiple Languages: You can use C#, F#, and Visual Basic programming languages to develop .NET Core applications. You can use your favorite IDE, including Visual Studio 2017/2019, Visual Studio Code, Sublime Text, Vim, etc.

 Modular Architecture: supports modular architecture approach using NuGet packages for various features that can be added to the .NET Core project as needed. Even the .NET Core library is provided as a NuGet package. The NuGet package for the default .NET Core application model is

Microsoft.NETCore.App. It reduces the memory footprint, speeds up the performance, and easy to maintain.

 CLI Tools: .NET Core includes CLI tools (Command-line interface) for development and continuous-integration.

 Flexible Deployment: .NET Core application can be deployed user-wide or system-wide or with Docker Containers.

 Compatibility: Compatible with .NET Framework and Mono APIs by using .NET Standard specification

3. Difference between asp.net and asp.net core

ASP.NET	ASP.NET CORE
Asp.Net Build for Windows	Asp.Net Core Build for Windows, Mac and Linux
Asp.Net has a Good Performance	ASP.Net core has higher performances than ASP.Net 4x.
It runs on .Net Framework or commonly called as full .Net Framework	It runs on .Net Core and Full .Net Framework.
Asp.Net Supports WebForm, Asp.Net MVC and Asp.Net WebAPI.	Asp.Net Core does not support WebForm. It supports MVC, Web API and Asp.Net Web pages originally added in .Net Core 2.0.
Asp.Net used the only IIS with dependant on System.web.dll.	Asp.Net Core has not dependant System.web.dll and so the IIS.
Support C#, VB and many other languages and also support WCF, WPF and WF	Support only C#, F# language. VB support to added a short time and no support WCF, WPF and WF but support for WCF client libraries are available.
Asp.Net MVC application added Web.config, Global.asax, Application Start.	Core did not support Web.config and Global.asax files. It is supporting appsettings.json.
Container support not more than better as the ASP.Net Core application.	Container support best suited for deployments like Docker.
All major versions supported	Support Core from Visual Studio 2015 update 3 and current version VS 2017.
We Need to re-compile after the code change.	Core Browser refresh will compile and executed the code no need for re-compile.

4. What is MVC and explain its architecture?

MVC stands for Model, View, and Controller. MVC separates an application into three components- Model, View, and Controller.

 Model: represents the shape of the data. A class in C# is used to describe a model. Model objects store data retrieved from the database. Model representsthe data.

 View: View in MVC is a user interface. View display model data to the user and also enables them to modify them. View in ASP.NET MVC is HTML, CSS, and some special syntax (Razor syntax) that makes it easy to communicate with the model and the controller.

 Controller: handles the user request. Typically, the user uses the view and raises an HTTP request. Controller processes request and returns the appropriate view as a response. Controller is the request handler

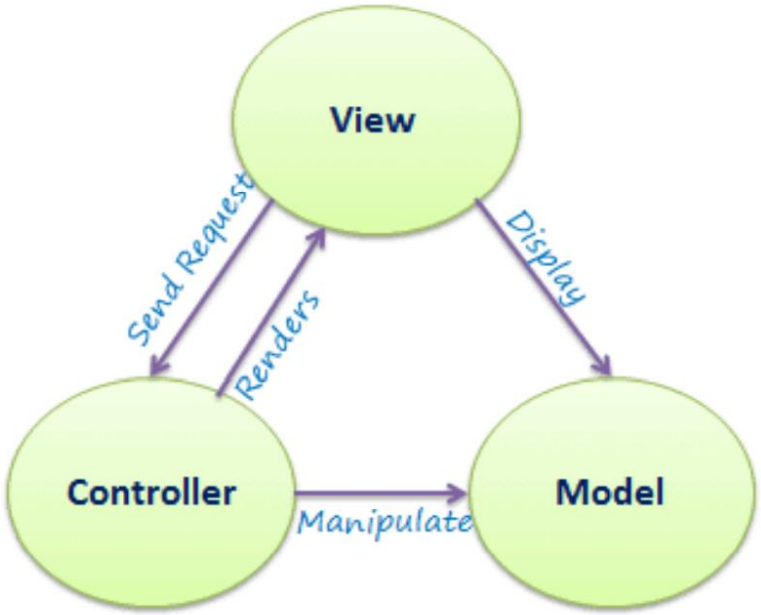
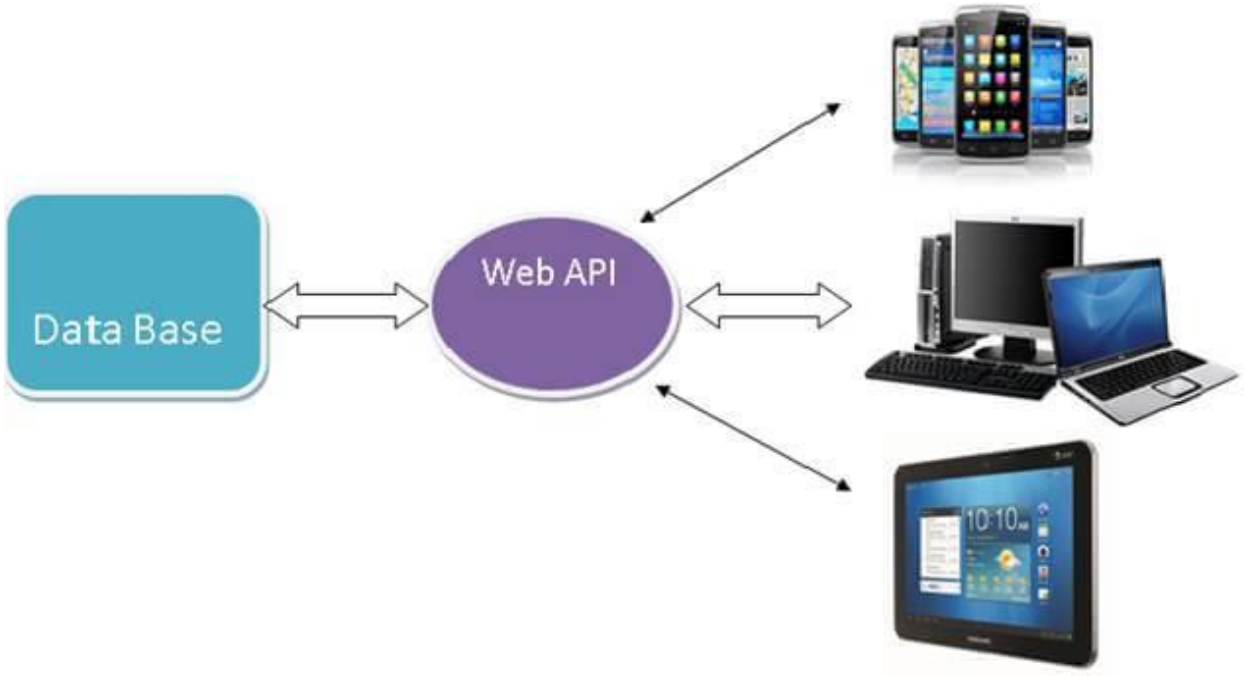


Fig: MVC Architecture

5.What is WEB API? List Advantages of WEB API.

Web API is an application programming interface (API) that is used to enable communication or interaction with software components with each other. ASP.NET Web API is a framework that makes it easy to build HTTP Service that reaches a broad range of clients, including browsers and mobile devices. Using ASP.NET, web API can enable communicating by different devices from the same database.



Uses of Web API

- It is used to access service data in web applications as well as many mobile apps and other external devices.
- It is used to create RESTful web services. REST stands for Representational State Transfer, which is an architectural style for networked hypermedia applications.
- It is primarily used to build Web Services that are lightweight, maintainable, and scalable, and support limited bandwidth.
- It is used to create a simple HTTP Web Service. It supports XML, JSON, and other data formats.

Benefits of Web API:

- Centralization of business logic will lessen your efforts, making your work worthwhile.
- These RESTful Web APIs are accessible by a variety of HTTP clients like Windows 7, Windows 8, Android, iPhone, iPad, WPF Client, Windows Phone 8 etc.
- It is an open source.
- It uses low bandwidth. (XML or JSON data). We can also pass html content if required.
- Web API Controller pattern is much similar to MVC Controller. Thus, it becomes easy to maintain and understand.

- Development using Web API itself makes a separation from UI Development.
- It is a lightweight architecture and is good for devices, which have limited bandwidth like smart cell phones.
- Its work is based on HTTP and is easy to define, expose and occupy in a RESTful way.
- It uses URIs for identifying information paths (resources)
- It also has message headers that are very meaningful and descriptive — headers that suggest us that the content type of the message's body, headers that explain how to secure it, how to cache information etc.
- Secure API endpoints with built-in support for industry standard JSON Web Tokens (JWT). Policy-based authorization gives you the flexibility to define powerful access control rules—all in code.

6. Explain Design principle or SOLID Principle.

Net Design Principle

Developers start building applications with good and tidy designs using their knowledge and experience. But over time, applications might develop bugs. The application design must be altered for every change request or new feature request. After some time we might need to put in a lot of effort, even for simple tasks and it might require full working knowledge of the entire system. But we can't blame change or new feature requests. They are part of software development. We can't stop them or refuse them either. So who is the culprit here? Obviously it is the design of the application.

The following are the design flaws that cause damage in software, mostly.

1. Putting more stress on classes by assigning more responsibilities to them. (A lot of functionality not related to a class.)
2. Forcing the classes to depend on each other. If classes are dependent on each other (in other words tightly coupled), then a change in one will affect the other.
3. Spreading duplicate code in the system/application.

Solution

1. Choosing the correct architecture (in other words MVC, 3-tier, Layered, MVP, MVVP and so on).
2. Following Design Principles.
3. Choosing correct Design Patterns to build the software based on its specifications.

SOLID principles are the design principles that enable us to manage most of the software design problems. Robert C. Martin compiled these principles in the 1990s. These principles provide us with ways to move

from tightly coupled code and little encapsulation to the desired results of loosely coupled and encapsulated real needs of a business properly. SOLID is an acronym of the following.

- S: Single Responsibility Principle (SRP)
- O: Open closed Principle (OSP)
- L: Liskov substitution Principle (LSP)
- I: Interface Segregation Principle (ISP)
- D: Dependency Inversion Principle (DIP)

S: Single Responsibility Principle (SRP)
SRP says "Every software module should have only one reason to change".

This means that every class, or similar structure, in your code should have only one job to do. Everything in that class should be related to a single purpose. Our class should not be like a Swiss knife wherein if one of them needs to be changed then the entire tool needs to be altered. It does not mean that your classes should only contain one method or property. There may be many members as long as they relate to single responsibility.

O: Open/Closed Principle

Here "Open for extension" means, we need to design our module/class in such a way that the new functionality can be added only when new requirements are generated. "Closed for modification" means we have already developed a class and it has gone through unit testing. We should then not alter it until we find bugs.

L: Liskov Substitution Principle

The Liskov Substitution Principle (LSP) states that "you should be able to use any derived class instead of a parent class and have it behave in the same manner without modification". It ensures that a derived class does not affect the behavior of the parent class, in other words,, that a derived class must be substitutable for its base class.

This principle is just an extension of the Open Closed Principle and it means that we must ensure that new derived classes extend the base classes without changing their behavior. For example*: A father is a doctor whereas his son wants to become a cricketer. So here the son can't replace his father even though they both belong to the same family hierarchy.*

I: Interface Segregation Principle (ISP)

The Interface Segregation Principle states "that clients should not be forced to implement interfaces they don't use. Instead of one fat interface, many small interfaces are preferred based on groups of methods, each one serving one submodule.".

Suppose we need to build a system for an IT firm that contains roles like TeamLead and Programmer where TeamLead divides a huge task into smaller tasks and assigns them to his/her programmers or can directly work on them.

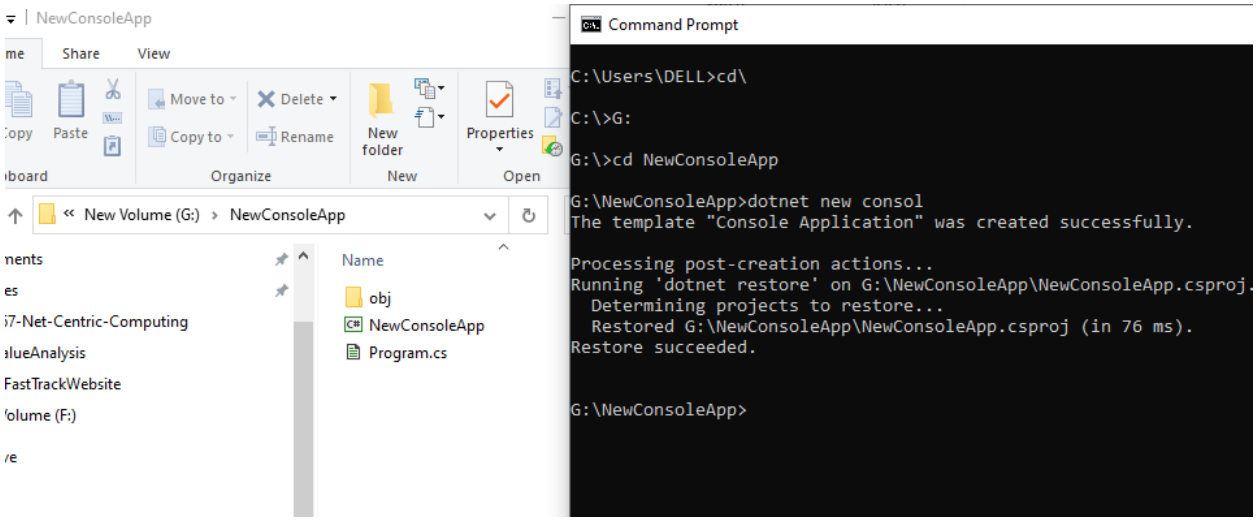
D: Dependency Inversion Principle

The Dependency Inversion Principle (DIP) states that high-level modules/classes should not depend on low-level modules/classes. Both should depend upon abstractions. Secondly, abstractions should not depend upon details. Details should depend upon abstractions.

7. Write steps to build, run, test and deploy .NET Core Applications

Deploy a NET Core console application using command line

1. Goto run type **cmd**
2. Specify location where we want to generate project.
3. Type `dotnet new console`



4. It will generate the following shown in picture.