1. **Difference between Restful Api verbs Post and put**

Ans

**POST: -**

* Always use Post for CREATE Operation
* POST is NOT idempotent. So, if you retry the request N times, you will end up having N resources with N different URIs created on server.
* Use Post when you want to add child resource under resource collection.
* Responses to this method are not cacheable, unless the response includes appropriate Cache-Control or Expires header fields.

**PUT: -**

* Generally, in practice, always use PUT for UPDATE operations.
* PUT method is idempotent. So, if you send retry a request multiple times, that should be equivalent to single request modification.
* Use Put When you want to modify a singular resource which is part of resources collection
* Though PUT is idempotent, we shall not cache its response.

**POST vs PUT: An Example**

Let’s say we are designing a network application. Let’s list down few URIs and their purpose to get better understanding when to use POST and when to use PUT operations.

* POST - /device-management/devices - Create a new device
* PUT - /device-management/devices/{id} - Update the device information identified by “id”

1. **Difference between abstract class and Interface. Which one should be used in which scenario and share the example if possible?**

Ans

**Abstract**

* Abstract is a class and it cannot be instantiated.
* It contains concrete and non-concrete methods as well.
* It accepts all access modifier.
* In future: if you add method, you don’t have to find all of the implementation and implement new method.
* Class Just inherit from one abstract

**Interface**

* Interface is not a class but it is entity by interface keyword.
* We can implement it, just method definition without any body
* It accepts only public.
* In Future: if you add method, you have to find all of implementation and implement new method.
* Class implement from more than one interface.

**Abstract Class Example** - When we have the requirement of a class that contains some common properties or methods with some common properties whose implementation is different for different classes, in that situation, it's better to use Abstract Class then Interface.

Abstract Methods : - price(),getTotalSeat(),colors();

Non-Abstract Methods:- Wheel(),CheckAC(),CallFaciltiy()

Abstract Method();

Non-Abstract Methods();



Hyundai

 Maruti

 Toyota

Abstract Class

Car

So As per the requirement Abstract class provide the better design solution because abstract methods override and the implement as per car model and non-abstract methods are common for all type of car model.

**In future if we have some unique feature in any one of the Car let suppose Hyundai introduce the new feature of GPS than if we are going to design another class for that and inherit that from Hyundai then it is not possible because c# does not allow to inherit multiple class. So, in that case need to implement the Interface to add for unique feature methods.**

**Interface Example: -** A new feature for the Hyundai car is Introduced called GPS which is not supported in Toyota cars. Then, we can go for the Interface.

Class Interface

Common Methods Unique Method

Inherits

Class

Common Methods

Unique Methods

SO, As per the above diagram what ever class of Car Model required to use Unique feature those are able to inherit the Interface others are not. And we can able to inherit multiple inheritance so we can add another new introduced features as per the model required.

1. **Find the occurrences of word from the book Share the pseudo code.**

**Ans**

In the Main Method

Start

Set Pattern as word(which need to find from Book Shared)

Set sentence/statement/string as Book Shared

Initialize and Set wordcount to zero

Foreach Match in Regex.Matches(sentence,Pattern)

Increase one to wordcount

Print the Pattern found the number of wordcount.

End

