Introductory

1. brief about your experience & worked in diff. Technologies

2. have u ever worked on any project which was longer than few months? And been longer till year?

3. what kind of role you played in prev. Projects? What were the responsibilities?

4. did u managed providing technical soln?

Project Soln. Architecture

1. **how do u propose an architecture to client for any project?**
2. **How did u managed giving solution architect – what initial steps you took?**
3. **Did the front-end team (angular) had to wait for back-end team (APIs) – OR how could they manage integration independently?**
4. **Did u used swagger hub? Or any other swagger – to give just API notes via swagger to front end team (before the actual APIs URLs are ready)**
5. **how to configure API URLs in API manager (API gateway)**

Misc.

Have you worked with any latest technologies - did u worked in angular / JavaScript / jQuery?

Have you ever worked on real estate projects? Developed any Google API? Worked on google analytic?

Role and Responsibilities in your current project

Have you worked with any latest technologies

Have you implemented Agile sprints in your project

SQL

1.What is Foreign Key

- foreign key is a field or a column that is used to establish a link between two tables.
- foreign key in one table used to point primary key in another table.
- ```
  CREATE TABLE Orders (
      OrderID int NOT NULL,
      OrderNumber int NOT NULL,
      PersonID int,
      PRIMARY KEY (OrderID),
      FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
  );
  ```

2. Difference between float and decimal

- DECIMAL and FLOAT both are used to store numerical values.
- Float stores an approximate value and decimal stores an exact value.
- Float is Approximate-number data type, which means that not all values in the data type range can be represented exactly, it rounds up the values.
- Decimal is Fixed-Precision data type, which means that all the values in the data type range can be represented exactly with precision and scale, it doesn't round up the values.
- In summary, exact values like money should use decimal, and approximate values like scientific measurements should use float.
- Difference Between Decimal and Float | by mayuri budake | Medium

Decimal:

- decimal [(p [,s])]
- p stands for Precision, the total number of digits in the value, i.e. on both sides of the decimal point
- s stands for Scale, number of digits after the decimal point

The default value of p is 18 and s is 0 and for both these values, the minimum is 1 and the maximum is 38.

- For instance, **decimal (4,2)** indicates that the number will have 2 digits before the decimal point and 2 digits after the decimal point, something like this has to be the number value: **##.##**
  ```

3. Join types and difference between Left Join and Inner join

- Inner Join
- LEFT JOIN
- RIGHT Join
- FULL Join
- Self-Join
- Cross Join

⇨ INNER JOIN

- Inner join returns only the matching rows between both the tables, non-matching rows are eliminated.

⇨ LEFT JOIN or LEFT OUTER JOIN

- Left Join or Left Outer Join returns only the matching rows between both the tables, plus nonmatching rows from the left table.

4. Query Optimisation or SP Optimization

- ❖ Indexing
- ❖ Select specific columns
- ❖ Avoid using SELECT DISTINCT
- ❖ Inner joins vs WHERE clause (use Inner Join)
- ❖ LIMIT command
- ❖ Loops versus Bulk insert/update

- ❖ Use SET NOCOUNT ON
  In the first procedure, I have not used SET NOCOUNT ON statement, so it has printed a message for how many rows are affected. It means if we do not use this then it will print an extra message and it affects performance.
- ❖ Use schema name before objects. It helps SQL Server to find the object.
- ❖ Use EXISTS () instead of COUNT ()
- ❖ Don’t use functions in the WHERE clause
- ❖ Use NO LOCK
- ❖ Specify column names instead of using * in SELECT statement
- ❖ Avoid temp temporary table
- ❖ Create Proper Index
- ❖ Use Join query instead of sub-query and co-related subquery
- ❖ Do not create your procedure name prefix with sp_ always use different prefix i.e., usp_YourStoreProcName.

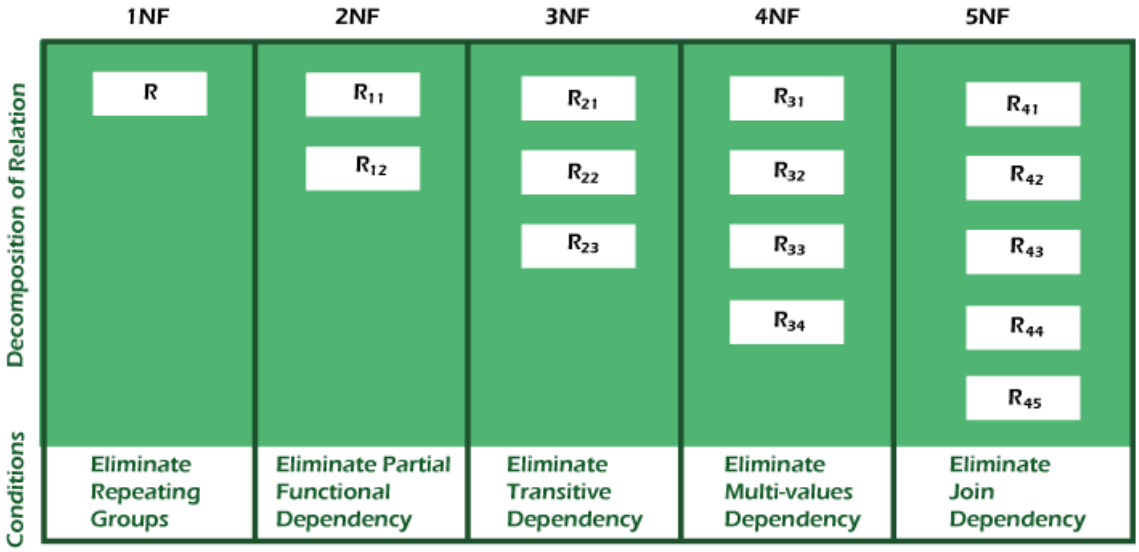
SP_StoreProcName is bad practice because when SQL server searches for procedure name then it will first search in the master database then your database. If you prefix with SP_ then it will start looking in system procedures so it will take more time.

- ❖ Keep your transaction as short as possible
- ❖ Use the Primary key & Index properly in the table
- ❖ Prevent the usage of DISTINCT and HAVING clauses.

- ➢ Prefer to use views and stored procedures in spite of writing long queries. It'll also help in minimizing network load.
- ➢ It's better to introduce constraints instead of triggers. They are more efficient than triggers and can increase performance.
- ➢ Make use of table-level variables instead of temporary tables.
- ➢ The UNION ALL clause responds faster than UNION. It doesn't look for duplicate rows whereas the UNION statement does that regardless of whether they exist or not.
- ➢ Prevent the usage of DISTINCT and HAVING clauses.
- ➢ Avoid excessive use of SQL cursors.
- ➢ Make use of SET NOCOUNT ON clause while building stored procedures. It represents the rows affected by a T-SQL statement. It would lead to reduced network traffic.
- ➢ It's a good practice to return the required column instead of all the columns of a table.
- ➢ Prefer not to use complex joins and avoid disproportionate use of triggers.
- ➢ Create indexes for tables and adhere to the standards.

5. What is normalization and What is IF, 2F, 3F

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.



What is Normalization in DBMS (SQL)? 1NF, 2NF, 3NF Example (guru99.com)

1NF (First Normal Form) Rules

- Each table cell should contain a single value.
- Each record needs to be unique.

2NF (Second Normal Form) Rules

- Rule 1- Be in 1NF
- Rule 2- Single Column Primary Key that does not functionally dependant on any subset of candidate key relation

3NF (Third Normal Form) Rules

- Rule 1- Be in 2NF
- Rule 2- Has no transitive functional dependencies

6. What is Indexes in SQL, Difference between them?

Difference between clustered & non-clustered indexes.

Indexes are used to retrieve data from the database more quickly

cluster and non-cluster index

https://www.sqlshack.com/what-is-the-difference-between-clustered-and-non-clustered-indexes-in-sql-server/

- **A clustered index** defines the order in which data is physically stored in a table. Table data can be sorted in only way, therefore, there can be only one clustered index per table. In SQL Server, the primary key constraint automatically creates a clustered index on that column.
- SQL Server allows us to add up to 16 columns to the clustered index key

- **A non-clustered index** doesn't sort the physical data inside the table. In fact, a non-clustered index is stored at one place and table data is stored in another place. This is similar to a textbook where the book content is located in one place and the index is located in another. This allows for more than one non-clustered index per table.
- When a query is issued against a column on which the index is created, the database will first go to the index and look for the address of the corresponding row in the table. It will then go to that row address and fetch other column values. It is due to this additional step that non-clustered indexes are slower than clustered indexes.

6. Difference between Normalization & De-normalization. When to use normalization & when to use De-normalization?

- Normalization is the technique of dividing the data into multiple tables to reduce data redundancy and inconsistency and to achieve data integrity. On the other hand, Denormalization is the technique of combining the data into a single table to make data retrieval faster.
- Normalization is used in **OLTP (Online transaction processing)** system, which emphasizes on making the insert, delete and update anomalies faster. As against, Denormalization is used in **OLAP (Online Analytical Processing)** system, which emphasizes on making the search and analysis faster.
- Data integrity is maintained in normalization process while in denormalization data integrity harder to retain.
- Redundant data is eliminated when normalization is performed whereas denormalization increases the redundant data.
- Normalization increases the number of tables and joins. In contrast, denormalization reduces the number of tables and join.
- Disk space is wasted in denormalization because same data is stored in different places. On the contrary, disk space is optimized in a normalized table.

C#

---

1. **Difference between inheritance and composition and give example of composition.**

   Inheritance enables us to create a new class that reuses and extends behaviour from another class called a base class or superclass and the newly created class is called the derived class.

   When we inherit a class from any other class, the derived class implicitly gains all the members of the base class except the constructors and destructor. So, the derived class can reuse a base class method and property without re-implementing it. In other words, we can say, using inheritance, a derived class extends the base class functionality. The main advantage of inheritance is the reusability of the code. The other main advantages of class inheritance are it makes it easy to modify the current implementation of the base class by just overriding an operation within the derived class. The limitation with inheritance is, we cannot change the implementation of the base class at run time.

   composition allows a class to contain an object instance of another class. Composition can be denoted as being an "as a part" or a "has a" relationship between classes.

```
public class Address
{
    public string GetAddress() => "Address";
}
```

Now, the `BrickHouse` class can have `Address` as its component:

```
public class BrickHouse : House
{
    private readonly Address _address;

    public BrickHouse()
    {
        _address = new Address();
    }

    public string GetAddress() => _address.GetAddress();
}
```

2. **Difference between Interface and Abstract class and where you can implement it.**

   Abstraction is detail hiding (implementation hiding). - using abstract classes

   and interfaces

An abstract class is a way to achieve the abstraction in C#. To declare abstract class, we use abstract keyword. An Abstract class is never intended to be instantiated directly. This class must contain at least one abstract method, which is marked by the keyword or modifier **abstract** in the class definition. The Abstract classes are typically used to define a base class in the class hierarchy.

Interface can have methods, properties, events, and indexers as its members. But interfaces will contain only the declaration of the members. The implementation of interface's members will be given by the class who implements the interface implicitly or explicitly.

3. **Difference between Interface and Abstract class and where you can implement**

- A class can implement any number of interfaces, but a subclass can at most use only one abstract class.
- An abstract class can have non-abstract Methods (concrete methods) while in case of Interface all the methods have to be abstract.
- An abstract class can declare or use any variables while an interface is not allowed to do so.
- An abstract class can have constructor declaration while an interface cannot do so.
- An abstract Class is allowed to have all access modifiers for all of its member declaration while in interface we cannot declare any access modifier (including public) as all the members of interface are implicitly public.

4. **What is Static Class and Sealed Class, Difference between static class and sealed class, where you can implement**

Static And Sealed Class In C# (c-sharpcorner.com)

Static Class:

- It can only have static members.
- It cannot have instance members as static class instance cannot be created.
- It is a sealed class.
- As static class is sealed, so no class can inherit from a static class.
- We cannot create instance of static class that's the reason we cannot have instance members in static class, as static means shared so one copy of the class is shared to all.
- Static class also cannot inherit from other classes.

Sealed class:

- Sealed class can be instantiated.
- It can inherit from other classes.
- It cannot be inherited.

5. **What is Generics and give one example of it.**

Generics allow you to write a class or method that can work with any data type.

```
var d365Response = await
        _baseService.GetAsync<ODataLocationResponse>(requestUri);
```

```
public async Task<T> GetAsync<T>(string requestUri)
{
try
{
   var response = await _httpClient.GetAsync(requestUri);
   if (response.IsSuccessStatusCode)
   {
      var responseString = await
                                    response.Content.ReadAsStringAsync();
      var odata = JsonConvert.DeserializeObject<T>(responseString);
      return odata;
   }
}

catch (Exception ex)
{
   throw ex;
}
}
```

6. **What is Ref and Out in a method**

- It is necessary the parameters should initialize before it pass to ref.
- It is not necessary to initialize parameters before it pass to out.

- It is not compulsory to initialize a parameter value before using it in a calling method.
- A parameter value must be initialized within the calling methos before its use.

- Passing a parameter value by Ref is useful when the called method is also needed to modify the pass parameter.
- Declaring a parameter to an out method is useful when multiple values need to be returned from a function or method.

- When we use REF, data can be passed bi-directionally.
- When we use OUT data is passed only in a unidirectional way (from the called method to the caller method).

```
        class GFG {
        static public void Main(d)
        {
                // Declaring variable without assigning value
                int G;

                // Pass variable G to the method using out keyword
                Sum(out G);

                // Display the value G
                Console.WriteLine("The sum of" +  " the value is: {0}", G);
        }

                // Method in which out parameter is passed and this method returns the
value of the passed parameter
        public static void Sum(out int G)
        {
                G = 80;
                G += G;
        }
}
```

O/p: The sum of the value is: 160

```
class GFG {
   public static void Main()
   {
      // Assign string value
      string str = "Geek";

      // Pass as a reference parameter
      SetValue(ref str);

      // Display the given string
      Console.WriteLine(str);
   }

   static void SetValue(ref string str1)
   {
      // Check parameter value
      if (str1 == "Geek") {
         Console.WriteLine("Hello!!Geek");
      }
```

```
        // Assign the new value of the parameter
        str1 = "GeeksforGeeks";
    }
}
```

O/p:
Hello!!Geek
GeeksforGeeks

https://www.geeksforgeeks.org/difference-between-ref-and-out-keywords-in-c-sharp/

7. **How to ensure the code is thread safe code.**

   like for list /collection, we should use Concurrent Bag or Concurrent Dictionary instead of Generic list or normal Dictionary.

   we can use lock mechanism whenever it is needed. (**Lock**: C# Lock keyword ensures that one thread is executing a piece of code at one time. The lock keyword ensures that one thread does not enter a critical section of code while another thread is in that critical section)

   use local variable within thread instead of common variable or object. depends on scenario.

   I am aware about this concept and can use whenever it is required.

   I didn't have more practical exp. but I am aware.

8. What is the difference between process and thread.

Comparison Basis	Process	Thread
Definition	A process is a program under execution i.e. an active program.	A thread is a lightweight process that can be managed independently by a scheduler
Context switching time	Processes require more time for context switching as they are heavier.	Threads require less time for context switching as they are lighter than processes.
Memory Sharing	Processes are totally independent and don't share memory.	A thread may share some memory with its peer threads.
Communication	Communication between processes requires more time than between threads.	Communication between threads requires less time than between processes.
Blocked	If a process gets blocked, remaining processes can continue execution.	If a user level thread gets blocked, all of its peer threads also get blocked.
Resource Consumption	Processes require more resources than threads.	Threads generally need less resources than processes.
Dependency	Individual processes are independent of each other.	Threads are parts of a process and so are dependent.
Data and Code sharing	Processes have independent data and code segments.	A thread shares the data segment, code segment, files etc. with its peer threads.
Treatment by OS	All the different processes are treated separately by the operating system.	All user level peer threads are treated as a single task by the operating system.
Time for creation	Processes require more time for creation.	Threads require less time for creation.
Time for termination	Processes require more time for termination.	Threads require less time for termination.

Difference between Process and Thread:

S.NO	Process	Thread
1.	Process means any program is in execution.	Thread means a segment of a process.
2.	The process takes more time to terminate.	The thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	The process is less efficient in terms of communication.	Thread is more efficient in terms of communication.
6.	Multiprogramming holds the concepts of multi-process.	We don't need multi programs in action for multiple threads because a single process consists of multiple threads.
7.	The process is isolated.	Threads share memory.
8.	The process is called the heavyweight process.	A Thread is lightweight as each thread in a process shares code, data, and resources.
9.	Process switching uses an interface in an operating system.	Thread switching does not require calling an operating system and causes an interrupt to the kernel.
10.	If one process is blocked then it will not affect the execution of other processes	If a user-level thread is blocked, then all other user-level threads are blocked.
11.	The process has its own Process Control Block, Stack, and Address Space.	Thread has Parents' PCB, its own Thread Control Block, and Stack and common Address space.
12.	Changes to the parent process do not affect child processes.	Since all threads of the same process share address space and other resources so any changes to the main thread may affect the behavior of the other threads of the process.
13.	A system call is involved in it.	No system call is involved, it is created using APIs.
14.	The process does not share data with each other.	Threads share data with each other.

9. What is difference between managed code / unmanaged code.

Code that executes under CLR execution environment is called as managed code. Unmanaged code executes outside CLR boundary. Unmanaged code is nothing, but code written in C++, VB6, VC++ etc. Unmanaged codes have their own environment in which the code runs and it's completely outside the control of CLR.

What is a garbage collector?

Garbage collector is a feature of CLR which cleans unused managed (it does not clean unmanaged objects) objects and reclaims memory. It's a background thread which runs continuously checks if there are any unused objects whose memory can be claimed.

clean unmanaged code:

.NET framework introduced some good ways to handle unmanaged object's memory deallocation such as:

1) using block

2) Dispose Method

3) Finalize

10. What is CLR?

As part of the Microsoft .NET Framework, the **Common Language Runtime (CLR)** is the programming (Virtual Machine component) that manages the execution of programs written in any language that uses the .NET Framework, for example C#, VB.Net, F# and so on.

Programmers write code in any language, including VB.Net, C# and F# then they compile their programs into an intermediate form of code called Common Intermediate Language (CIL) in a portable execution file (PE) that can be managed and used by the CLR and then the CLR converts it into machine code to be will executed by the processor.

The information about the environment, programming language, its version and what class libraries will be used for this code are stored in the form of metadata with the compiler that tells the CLR how to handle this code.

The CLR allows an instance of a class written in one language to call a method of the class written in another language.

11. What is MVC?

MVC (full form Model View Controller) is a software architecture or application design model containing 3 interconnected verticals or portions. These 3 portions are the model (data associated with the application), the view (which is the user interface of an MVC application), and the controller (the processes that are responsible for handling the input).

The MVC model is normally used to develop modern applications with user interfaces. It provides the central pieces for designing a desktop or mobile application, as well as modern web applications.

Model

A model can be defined as the data that will be used by the program. Commonly used examples of models in MVC are the database, a simple object holding data (such as any multimedia file or the character of a game), a file, etc.

View

A view is a way of displaying objects (user interfaces) within an application. This is the particular vertical through which end users will communicate.

Controller

A controller is the third vertical which is responsible for updating both models and views. It accepts input from users as well as performs the equivalent update. In other words, it is the controller which is responsible for responding to user actions.

12. What is use of using statement in a method.

The using declaration calls the Dispose method on the object in the correct way when it goes out of scope. **The using statement causes the object itself to go out of scope as soon as Dispose is called**. Within the using block, the object is read-only and can't be modified or reassigned.

The **using** statement ensures that **Dispose()** is called even if an exception occurs when you are creating objects and calling methods, properties and so on. Dispose() is a method that is present in the IDisposable interface that helps to implement custom Garbage Collection.

Testing

---

1. **How you are doing unit test in project**

2. **If new functionality implemented in your project, how can you test the new one without disturbing old one.**

3. **String x=”x y is my name” is a string print this like y=”name my is y x”**

```
string x = "x y is my name";
string reversestring = string.Empty;

string[] stringsArray = x.Split(separator: ' ');

for (int i = stringsArray.Length - 1; i > -1; i--)
{
        if (i == stringsArray.Length - 1)
        {
                reversestring += stringsArray[i];
        }
        else
        {
                 reversestring += ' ' + stringsArray[i];
        }
}
```