- **What is role of client?**
    - A client is an individual, organization, or system that seeks services, resources, or expertise from another party, often called a service provider or server.
    - The client initiates requests and defines requirements, playing a pivotal role in various contexts such as computer networking, business services, software development, finance, web development, healthcare, and customer relations.
    - Effective communication and collaboration between clients and service providers are essential for successful outcomes.
- **What is role of server?**
    - A server is a computer or system that provides services, resources, or functionalities to clients.
    - Its primary role is to respond to client requests, fulfill their needs, and facilitate communication or data exchange.
    - Servers play a central role in various domains, including computer networking, web hosting, and database management.
    - They store, process, and deliver information, enabling clients to access and utilize services or data.
- **What is client server architecture?**
    - Client-server architecture is a computing model where tasks are divided between clients (requesters) and servers (providers).
    - Clients initiate requests for services, and servers fulfill those requests.

- Communication occurs over a network, with clients and servers having distinct roles.
- This architecture allows for scalability, centralized management of resources, and efficient distribution of processing tasks.
- Examples include web applications and database systems.

- **What is role of compiler?**
  - A compiler is a software tool that translates source code written in a high-level programming language into machine code or an intermediate code.
  - Its primary role is to convert human-readable code into a format that can be executed by a computer's hardware.
  - The compilation process involves several stages, including lexical analysis, syntax analysis, optimization, and code generation.
  - The compiler ensures that the program is free of syntax errors, translates it into an efficient form, and produces an executable file that can be run on a computer.

- **What is difference between Compiler and Interpreter?**
  - The main differences between a compiler and an interpreter are:

- Translation:

  ✓ Compiler: Translates the entire source code before execution, generating a standalone executable.

  ✓ Interpreter: Translates and executes the code line by line at runtime.

- Execution:

  - ✓ Compiler: Results in a separate executable file independent of the source code.

  - ✓ Interpreter: Requires the original source code during execution.

- Performance:

  - ✓ Compiler: Generally produces faster and more optimized code.

  - ✓ Interpreter: Tends to have slower execution due to real-time interpretation.

- Debugging:

  - ✓ Compiler: Involves a separate debugging process on compiled code.

  - ✓ Interpreter: Allows for easier debugging as errors can be corrected in the original source code during runtime.

- Memory Usage:

  - ✓ Compiler: Higher memory usage during compilation but can be more memory-efficient during execution.

  - ✓ Interpreter: Lower memory requirements during interpretation, but overall execution may be less memory-efficient.

- Portability:

  - ✓ Compiler: Generates platform-specific executables, may lack portability without recompilation.

  - ✓ Interpreter: More portable, as the source code can be executed on different systems with the appropriate interpreter.

Both have their advantages and trade-offs, and the choice depends on factors like performance, debugging needs, and portability requirements. Some languages use a hybrid approach with both compilation and interpretation.

- **What is MVC?**

    - MVC, which stands for Model-View-Controller, is a software architectural pattern commonly used in the design of interactive and user-friendly applications.

    - It separates an application into three interconnected components:

1. Model (Data):

- Represents the application's data structure and business logic.
- Manages the data, logic, and rules of the application.
- Notifies the View of any changes in the data.

2. View (User Interface):

- Represents the presentation layer or user interface of the application.
- Displays information to the user and captures user input.
- Receives data from the Model and presents it to the user.

3. Controller (Logic):

- Acts as an intermediary between the Model and the View.
- Handles user input, processes it, and updates the Model and View accordingly.
- Contains the application's business logic and orchestrates the flow of data between the Model and the View.

The key idea behind MVC is to separate the concerns of data (Model), presentation (View), and user interaction (Controller), making the codebase more modular, maintainable, and scalable. This separation

allows for easier development, testing, and modification of each component independently.

  - MVC is widely used in web development, desktop applications, and other software projects to create a clean and organized architecture.

  - It promotes the principle of "separation of concerns," enhancing the overall maintainability and flexibility of the software.

- **What is communication protocol and what is difference between HTTP and HTTPS?**

- **What is .net?**

  - .NET is a cross-platform framework developed by Microsoft for building diverse applications, including web, desktop, mobile, and cloud-based solutions.

  - It includes a Common Language Runtime (CLR) for execution, a Base Class Library (BCL) for core functionality, and supports multiple programming languages like C#, VB.NET, and F#.

  - Components such as ASP.NET, Xamarin, and Entity Framework enhance capabilities for web, mobile, and database development.

  - With .NET 5 and beyond, Microsoft has unified the platform, combining features from both .NET Core and the traditional .NET Framework.

- **What is CLR?**
- **What is difference between CLS and CTS?**
- **What do you mean by design pattern?**
- **What is difference between Asp.Net and MVC.Net?**