# Short Answer:

Answer the following questions with complete sentences in your own words. You are encouraged to conduct your own research online or through other methods before answering the questions. If you research online, please consult multiple sources before you write down your answers. You are expected to be able to explain your answers in detail (Provide examples for each question).

1. What are the HTTP request methods? Describe what each one does.

HTTP Request Method

HTTP defines a set of request methods to indicate the desired action to be

performed for a given resource.

● GET — The GET method requests a representation of the specified resource.

Requests using GET should only retrieve data.

● POST — The POST method is used to submit an entity to the specified resource,

often causing a change in state or side effects on the server.

● PUT — The PUT method replaces all current representations of the target resource

with the request payload.

● DELETE — The DELETE method deletes the specified resource.

● HEAD, CONNECT, OPTIONS, TRACE, PATCH

● HTTP Request

○ http method: get, post, put, update, delete

GET is used to request data from a specified resource.

POST is used to send data to a server to create/update a resource.

PUT is used to send data to a server to create/update a resource. PUT is idempotent, POST is not

DELETE is used to delete the specified resource.

PATCH is used to apply partial modifications to a resource.

HTTP

● HTTP stands for Hypertext Transfer Protocol and is used to structure requests

and responses over the internet.

● HTTP requires data to be transferred from one point to another over the

network

● <http://www.google.com/>

HTTPS

● Hypertext Transfer Protocol Secure, it’s a secure version of HTTP

○ It’s encrypted to increase the security of data transferring

● HTTPS uses TLS/SSL (Transport Layer Security/Secure Socket Layer) to encrypt

communication by using an asymmetric public key infrastructure

○ Private Key – Controlled by the owner of a website

○ Public Key – Available to everyone who wants to interact with the server

● <https://www.google.com/>

HTTP Request

HTTP Request is a packet of Information that one computer sends to another

computer to communicate something

To its core, HTTP Request is a packet of binary data sent by the Client to server.

An HTTP Request contains following parts

● Request Line

● Headers, 0 or more Headers in the request

● An optional Body of the Request

Request Line

A Request Line specifies the Method Token (GET, PUT … ) followed by the

Request URI and then the HTTP Protocol that is being used

Graphical user interface, text, application, email

Description automatically generated

There are several HTTP request methods that a client can use when making a request to a server. Here is a list of the most common HTTP request methods, along with a brief description of what each one does:

1. **GET**: This method requests a representation of the specified resource. Resources can be data or HTML documents, images, videos, etc.
2. **HEAD**: This method is similar to **GET**, but it only requests the headers for a resource, rather than the resource itself.
3. **POST**: This method submits an entity to the specified resource, often causing a change on the server (e.g., creating a new resource).
4. **PUT**: This method replaces the current representation of the target resource with the request payload.
5. **DELETE**: This method deletes the specified resource.
6. **CONNECT**: This method establishes a tunnel to the server identified by the target resource.
7. **OPTIONS**: This method describes the communication options for the target resource.
8. **TRACE**: This method performs a message loop-back test along the path to the target resource.
9. **PATCH**: This method applies partial modifications to a resource.

Here is an example of a **GET** request:

GET /index.html HTTP/1.1

Host: [www.example.com](http://www.example.com)

This **GET** request is asking the server to retrieve the resource at the specified URL (**/index.html**) and return it to the client.

And here is an example of a **POST** request:

POST /users HTTP/1.1

Content-Type: application/json

{

"name": "John",

"email": "john@example.com"

}

This **POST** request is creating a new user resource on the server with the specified name and email.

1. Explain the differences between POST, PUT, and PATCH.

POST VS. PUT

POST — The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line

● In other word, POST is used to create/add

PUT — The PUT method requests that the enclosed entity be stored under the supplied

RequestURI.

● If the Request-URI refers to an already existing resource, the enclosed entity SHOULD be

considered as a modified version of the one residing on the origin server.

● If the Request-URI does not point to an existing resource, and that URI is capable of being

defined as a new resource by the requesting user agent, the origin server can create the

resource with that URI.

● In other word, PUT is used to create/add and update

● When to use

○ Do you name your URL objects you create explicitly, or let the server

decide? If you name them then use PUT. If you let the server decide then

use POST.

○ PUT is idempotent, so if you PUT an object twice, it has no effect.

○ You can update or create a resource with PUT with the same object URL

○ POST twice with the same data means that create two identical users with

different ids

Pragmatic Advice

● To save an existing user, or one where the client generates the id, and it's been

verified that the id is unique

○ PUT /user/12345 HTTP/1.1 <-- create the user providing the id 12345

○ GET /user/12345 HTTP/1.1 <-- return that user

● Otherwise, use POST to initially create the object, and PUT to update the object:

○ POST /user HTTP/1.1 <--- create the user, server returns 12345

○ PUT /user/12345 HTTP/1.1 <--- update the user

When it comes to system design, we only need to choose one either POST or PUT and

support it well in the application.

POST, PUT, and PATCH are HTTP request methods that are used to create, update, and modify resources on the server. Here is a brief summary of the main differences between these request methods:

1. POST: The POST method is used to create a new resource on the server. When making a POST request, the client sends data to the server, which is then used to create a new resource. The POST method is often used to submit form data to a server.
2. PUT: The PUT method is used to update an existing resource on the server. When making a PUT request, the client sends data to the server, which is used to update the resource at the specified URL. The PUT method replaces the current representation of the target resource with the request payload.
3. PATCH: The PATCH method is used to apply partial modifications to a resource. When making a PATCH request, the client sends data to the server, which is used to update specific fields of the resource at the specified URL. The PATCH method only updates the specified fields and leaves the rest of the resource unchanged.

In summary, the POST method is used to create new resources, the PUT method is used to update existing resources, and the PATCH method is used to apply partial updates to existing resources.

1. What does it mean for HTTP to be stateless?

When an HTTP protocol is said to be "stateless", it means that the server does not store any information about the client's session and does not retain any information from previous requests.

In a stateless protocol, each request from the client is treated as an independent request, and the server does not need to maintain any information about the client's session. This means that the server does not need to remember anything about the client from one request to the next, and it does not need to maintain any session state.

One advantage of stateless protocols is that they are simple and easy to implement. They can also be more scalable, since the server does not need to maintain any state information for each client.

However, stateless protocols can also be less efficient in some cases, since the server cannot use information from previous requests to optimize the handling of subsequent requests. To overcome this limitation, many web applications use techniques such as cookies or session IDs to maintain state across multiple requests.

1. What is URL and What is URI?

A URL (Uniform Resource Locator) is a specific type of URI (Uniform Resource Identifier) that specifies the location of a resource on the internet.

Every URL consists of three main parts:

1. The protocol (e.g., "http" or "https")
2. The domain name (e.g., "example.com")
3. The path (e.g., "/path/to/resource")

For example, the URL "<https://www.example.com/path/to/resource>" specifies a resource that can be accessed using the HTTPS protocol at [www.example.com](http://www.example.com/), at the location "/path/to/resource".

A URI, on the other hand, is a more general term that refers to any identifier that identifies a name or a resource. This can include URLs, but it can also include other types of identifiers, such as URNs (Uniform Resource Names).

In summary, all URLs are URIs, but not all URIs are URLs. URLs are a specific type of URI that specify the location of a resource, while URIs are a more general term that can refer to any identifier for a resource.

URL VS. URI

URL — uniform resource locator

URI — uniform resource identifier

Look at an example

● 1 — Protocol name (HTTP)

● 2 — Host name

● 3 — Port number(Optional)

● 4 — Path

A picture containing shape

Description automatically generated

URL VS. URI

● URI can be a name, locator, or both for an

online resource where a URL is just the

locator

● URLs are a subset of URIs. That means all

URLs are URIs.

● Your name could be a URI because it

identifies you, but it couldn't be a URL

because it doesn't help anyone find your

location

● On the other hand, your address is both a

URI and a URL because it both identifies

you and it provides a location for you

Diagram, venn diagram

Description automatically generated

Request Header

In the request section, whatever follows Request Line till before Request Body

everything is a Header

Headers are used to pass additional information about the request to the server

Graphical user interface, text, application, email

Description automatically generated

Request Body

Request Body is the part of the HTTP Request where additional content can be sent

to the server.

● eg. a file type of JSON or XML.

It is optional

Graphical user interface, text, application, email

Description automatically generated

HTTP Response

HTTP Response is the packet of information sent by Server to the Client in response

to an earlier Request made by Client

HTTP Response contains the information requested by the Client

Just like HTTP Request, HTTP Response also has the same structure:

● Status Line

● Headers, 0 or more Headers in the request

● An optional Body of the Request

Status Line

A Status Line consists of three parts:

● HTTP Protocol Version

● Status Code

● Reason Phrase

Graphical user interface, text, application, email

Description automatically generated

1. What are the HTTP Response Status Codes?

HTTP Status Code

There are five classes of status code

● 1xx Informational – the request was received, continuing process

● 2xx Successful – the request was successfully received, understood and accepted

● 3xx Redirection – further action needs to be taken to complete the request

● 4xx Client Error – the request contains bad syntax or cannot be fulfilled

● 5xx Server Error – the server failed to fulfill an apparently valid request

Full status code reference: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

**200 OK**

## 401 Unauthorized Request

## 404 Page Not Found

## 500 Internal Server Error

## 503 Service Unavailalve

HTTP response status codes are three-digit codes that indicate the status of a response to a client's request. They are typically included in the response header sent by the server to the client.

Here are some common HTTP response status codes:

1. **200 OK**: This status code indicates that the request was successful and that the requested resource was returned in the response body.
2. **301 Moved Permanently**: This status code indicates that the requested resource has been permanently moved to a new URL, which is specified in the "Location" header field of the response.
3. **302 Found**: This status code indicates that the requested resource has temporarily moved to a new URL, which is specified in the "Location" header field of the response.
4. **401 Unauthorized**: This status code indicates that the request requires authentication. The response should include a "WWW-Authenticate" header field containing a challenge applicable to the requested resource.
5. **403 Forbidden**: This status code indicates that the server understands the request, but it refuses to authorize it.
6. **404 Not Found**: This status code indicates that the requested resource could not be found on the server.
7. **500 Internal Server Error**: This status code indicates that an unexpected condition was encountered by the server, and it could not complete the request.

There are many other HTTP response status codes, but these are some of the most common ones.

Response Header

Just like a Request Header, Response Header also contains zero or more Header

lines. However, it is very uncommon to have zero Headers in the response.

Lines just after the Status Line and before the Response Body are all Response

Headers lines

Graphical user interface, text, application, email

Description automatically generated

1. What is DNS and what role does it play?

DNS (Domain Name System) is a system that maps human-readable domain names to IP addresses. It plays a crucial role in the functioning of the internet by allowing users to access websites using easily memorable names, rather than having to remember the numerical IP addresses of servers.

When you enter a domain name into your web browser's address bar (e.g., [www.example.com](http://www.example.com/)), your computer sends a request to a DNS server to resolve the domain name to an IP address. The DNS server looks up the IP address for the domain name and sends it back to your computer. Your computer can then use the IP address to establish a connection with the server hosting the website and request the contents of the webpage.

DNS is often compared to a phonebook, as it allows you to look up the address (IP address) of a person (website) by their name (domain name).

In summary, DNS plays a crucial role in the functioning of the internet by mapping domain names to IP addresses and allowing users to access websites using easily memorable names.

Web Server

A program that uses HTTP for serving files that create web pages for users in

response to their requests that are sent by the HTTP clients of their computer is

called as a web server.

This server is always connected to the internet.

Every Web server that is connected to the Internet is given a unique address made

up of a series of four numbers between 0 and 256 separated by periods.

● eg. 68.178.157.132

Web Server

Computers hosting websites are web servers

The diagram below represents the basic elements of a web server

Graphical user interface, text, application

Description automatically generated

Web Server

When you register a web address, also known as a domain name, such as google.com you have to

specify the IP address of the Web server that will host the site

Then you can load up with Dedicated Servers that can support your web-based operations

There are four leading web servers

● Apache HTTP Server (like Tomcat) — widely used

● Internet Information Services (IIS) — hosting for .NET

● Nginx Web Server — high performance, stability, simple configuration and low resource usage.

● Google Web Server (GWS)

Except for those, there are several other web servers in market like Bea's Web Logic and IBM's

WebSphere

Web Server

SMTP Server — Simple Mail Transfer Protocol Server. This server takes care of delivering emails from one

server to another server

FTP — File Transfer Protocol. a standard network protocol used for the transfer of computer files between

a client and server on a computer network

ISP — Internet Service Provider. They are the companies who provide you service in terms of internet

connection to connect to the internet.

● eg. You will buy space on a Web Server from any Internet Service Provider. This space will be used to host your Website.

DNS — Domain Name System.

● When someone types in your domain name, www.example.com, your browser will ask the Domain Name System to find the IP that hosts your site. When you register your domain name, your IP address should be put in a DNS along with your domain name

Web Application

● Until now, we have already covered all basics in web application.

● So what happens when I type google.com in the browser? Or how does the web

application works?

How It Works

● A user enters a URL into a browser (for example, Google.com. This request is passed to a domain name server.

● The domain name server returns an IP address for the server that hosts the Website (for example, 68.178.157.132).

● The browser requests the page from the Web server using the IP address specified by the domain name server. (URL)

● The Web server returns the page to the IP address specified by the browser requesting the page. The page may also contain links to other files on the same server, such as images, which the browser will also request.

● The browser collects all the information and displays to your computer in the form of Web page.

Diagram

Description automatically generated

1. What is HTML? What is CSS and What is JavaScript?

Client

A client in web application usually refers to a server that presents the user interface

which a user actually interacts with, like browsers.

Languages supported by web browsers:

● HTML

● CSS

● JavaScript

HTML

HTML stands for Hyper Text Markup Language

HTML describes the structure of a Web page

HTML consists of a series of elements

HTML elements tell the browser how to display the content

HTML elements are represented by tags

HTML tags label pieces of content such as "heading", "paragraph", "table", and so on

Browsers do not display the HTML tags, but use them to render the content of the

Page

CSS

CSS stands for Cascading Style Sheets

CSS describes how HTML elements are to be displayed on screen, paper, or in other

media

CSS saves a lot of work. It can control the layout of multiple web pages all at once

External stylesheets are stored in CSS files

JavaScript

JavaScript is the programming language of HTML and the Web.

● HTML to define the content of web pages

● CSS to specify the layout of web pages

● JavaScript to program the behavior of web pages

HTML and CSS will only produce static web pages. JavaScript will make it dynamic

by different events to respond to different user interactions

HTML (Hypertext Markup Language) is a markup language used to structure content on the web. It is the standard markup language for creating web pages, and it is used to describe the structure of text, images, and other content on a webpage. HTML consists of a series of elements, which are represented by tags, and it uses these tags to define the structure and layout of content on a webpage.

CSS (Cascading Style Sheets) is a stylesheet language used for describing the look and formatting of a document written in HTML. CSS is used to define the styles for font, color, layout, and other aspects of the presentation of a webpage. It allows developers to separate the content of a webpage (written in HTML) from its presentation (defined by CSS). This makes it easier to maintain and update the appearance of a webpage over time.

JavaScript is a programming language that is commonly used to add interactivity to websites. It is a client-side scripting language, which means that it is executed by the user's web browser rather than on the server. JavaScript can be used to add features such as drop-down menus, form validation, and interactive elements to a webpage. It is often used in conjunction with HTML and CSS to build modern, dynamic websites.

1. What is .NET?

.NET is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It provides a platform for building, deploying, and running applications and services that use .NET technologies, such as C# and VB.NET.

The .NET framework consists of a common language runtime (CLR) and a set of class libraries. The CLR is the runtime environment that executes .NET applications and provides services such as memory management, security, and exception handling. The class libraries provide a set of reusable code that can be called by applications to perform various tasks, such as reading and writing files, accessing databases, and communicating over the network.

.NET also includes a set of tools and technologies for building applications, such as Visual Studio, a development environment for creating .NET applications.

.NET supports several programming languages, including C#, VB.NET, F#, and others. It is used to build a wide range of applications, including desktop applications, web applications, mobile applications, and cloud services.

.NET is an open-source developer platform, created by Microsoft, for building many

different types of applications.

It’s free and cross-platform, you can write .NET apps in C#, F# or Visual Basic.

Diagram, text

Description automatically generated A screenshot of a video game

Description automatically generated

Chart

Description automatically generated

1. What is the difference between .NET Framework and .NET core?

.NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It provides a platform for building, deploying, and running applications and services that use .NET technologies, such as C# and VB.NET. .NET Framework has been around for a long time and is used to build a wide range of applications, including desktop applications, web applications, and services.

.NET Core is a cross-platform, open-source version of .NET that was developed by Microsoft and the .NET community. It is a modular and lightweight version of .NET that can be used to build a wide range of applications, including console applications, web applications, and cloud services. .NET Core is designed to be portable, meaning that it can run on multiple platforms, including Windows, macOS, and Linux.

There are a few key differences between .NET Framework and .NET Core:

1. Platform support: .NET Framework is primarily designed to run on Windows, while .NET Core is cross-platform and can run on Windows, macOS, and Linux.
2. Open-source: .NET Core is open-source and developed in collaboration with the .NET community, while .NET Framework is proprietary and developed by Microsoft.
3. Modularity: .NET Core is designed to be modular, meaning that it can be customized to include only the components that are needed for a specific application. .NET Framework is a monolithic framework that includes a large number of libraries and components.
4. Cloud-optimized: .NET Core is designed to be optimized for cloud scenarios, such as microservices and containerized environments.

In summary, .NET Core is a cross-platform, open-source version of .NET that is optimized for cloud scenarios, while .NET Framework is a proprietary, Windows-based framework that is used to build a wide range of applications.

.NET Framework

.NET Framework is the original implementation of .NET. It supports running

websites, services, desktop apps, and more on Windows.

Diagram

Description automatically generated Text

Description automatically generated with medium confidence

.NET Core

.NET Core is a free, open-source development platform maintained by Microsoft. It’s

a cross-platform that runs on Windows, macOS, and Linux operating systems.

Diagram

Description automatically generated

Diagram

Description automatically generated Graphical user interface, table

Description automatically generated Diagram

Description automatically generated Graphical user interface, text, application

Description automatically generated

ASP.NET Core Project Type

● MVC

● API

● Razor Pages

● Blazor Server

● Blazor WebAssembly/Client

Text

Description automatically generated

NET Core:

·  Open Source

·  Cross platform: windows, Mac OS, Linux, etc.

·  Web applications, windows mobile, and windows store

Installation : A collection of NuGet packages

Deployment : When a new version is installed, it is updated on one computer at a time, resulting in new directories being created in an existing program without affecting it. Solid and adaptable.

Support : Support microservice and user has to create a REST API for its implementation.

Lightweight CLI -tools, high performance and scalability due to architecture and adaptability

.NET Framework:

·  Windows operating system

·  Desktop and web applications

·  Does not support microservices

Installation : All the libraries of .Net framework are packages and shipped together.

Deployment : When the updated version is released, it is first deployed on the internet information service only.

Support : Does not support microservices, but allows REST API services, a good choice for WCF.

Less effective in comparison, heavy and does not support mobile development

# Coding Questions:

Write code in c# to solve the following problems. Please write your own answers. You are highly encouraged to present more than one way to answer the questions. Please follow best practices when you write the code so that it is easily readable, maintainable, and efficient. Clearly state your assumptions if you have any. You may discuss with others on the questions, but please write your own code.

Develop a portfolio web pages which introduce yourself by HTML, CSS, JavaScript/TypeScript Content should include but not limit to the following:

1. Brief Introduction of yourself
2. Education history
3. Professional experience
   1. Project
   2. Work experience (Optional)
   3. Link to demo (Optional)
4. Hobby
5. Avatar/Photo