**How HTTPS works behind the scene?**

User enters a url => browser checks if available in browser cache => DNS look up [root dns lookup => recursive dns => get ip ] => establish a tcp connection => send http req => get the static page from web server=> if authentication is required that happens [either read cookies from browser OR ask user to enter credentials] => somehow gets the dynamic elements as well [how ? , there is some lazy initialization here ?] => Then user performs some action[clicks a link or something] => check browser cache => if not avail [take the input parameters and embed in the url in some manner [may be encrypt some things if required] => hits a load balancer => directed to a application server [depending on how the LB selects a host] => application server cache is checked [Memcached or some kind of caching, not sure if this "normally" happens here or at some other level] => application server tries to understand the request [if its a service listening on some port, http port 80 it will get the URL and parse to perform some operations] => database is queried if required to => there might again be connection mgmt/caching/parallel queries etc here => database returns back the result to app server => app server creates a result payload and headers [http] => sends it to browser for rendering => browser cache is updated => user reacts to the response.

**What are different http methods available and what they exactly do.**

|  |  |
| --- | --- |
| **S.N.** | **Method and Description** |
| 1 | **GET**  The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data. |
| 2 | **HEAD**  Same as GET, but transfers the status line and header section only. |
| 3 | **POST**  A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms. |
| 4 | **PUT**  Replaces all current representations of the target resource with the uploaded content. |
| 5 | **DELETE**  Removes all current representations of the target resource given by a URI. |
| 6 | **CONNECT**  Establishes a tunnel to the server identified by a given URI. |
| 7 | **OPTIONS**  Describes the communication options for the target resource. |
| 8 | **TRACE**  Performs a message loop-back test along the path to the target resource. |

**Understand and explain the use of various http response codes.**

* 205 Reset Content
* 300 Multiple Choices
* 419 Authentication Timeout
* 450 Blocked by Windows Parental Controls
* 1xx – informational
* 2xx – successful
* 3xx – redirection
* 4xx – client error
* 5xx – server error
* 302 Found (and perhaps the other redirect codes) – Supporting this status code will let clients follow a redirect, which can be particularly useful when you need to reroute a request to another host.
* 304 Not Modified – Supporting this status code, along with conditional GET requests (ETag/If-None-Match, Date/If-Modified-Since), will let clients and proxies cache responses. This is particularly important for JavaScript running in the browser, but other clients support this too, including OkHttp on Android and AFNetworking on iOS.
* 429 Too Many Requests – Supporting this status code, along with the Retry-After header, will let clients perform automatic backoff in the case of rate-limiting.

**What are the different web communication protocols and their use cases?**

1. Transmission Control Protocol (TCP): TCP is a popular communication protocol which is used for communicating over a network. It divides any message into series of packets that are sent from source to destination and there it gets reassembled at the destination.
2. Internet Protocol (IP): IP is designed explicitly as addressing protocol. It is mostly used with TCP. The IP addresses in packets help in routing them through different nodes in a network until it reaches the destination system. TCP/IP is the most popular protocol connecting the networks.
3. User Datagram Protocol (UDP): UDP is a substitute communication protocol to Transmission Control Protocol implemented primarily for creating loss-tolerating and low-latency linking between different applications.
4. Post office Protocol (POP): POP3 is designed for receiving incoming E-mails.
5. Simple mail transport Protocol (SMTP): SMTP is designed to send and distribute outgoing E-Mail.
6. File Transfer Protocol (FTP): FTP allows users to transfer files from one machine to another. Types of files may include program files, multimedia files, text files, and documents, etc.
7. Hyper Text Transfer Protocol (HTTP): HTTP is designed for transferring a hypertext among two or more systems. HTML tags are used for creating links. These links may be in any form like text or images. HTTP is designed on Client-server principles which allow a client system for establishing a connection with the server machine for making a request. The server acknowledges the request initiated by the client and responds accordingly.
8. Hyper Text Transfer Protocol Secure (HTTPS): HTTPS is abbreviated as Hyper Text Transfer Protocol Secure is a standard protocol to secure the communication among two computers one using the browser and other fetching data from web server. HTTP is used for transferring data between the client browser (request) and the web server (response) in the hypertext format, same in case of HTTPS except that the transferring of data is done in an encrypted format. So it can be said that https thwart hackers from interpretation or modification of data throughout the transfer of packets.
9. Telnet: Telnet is a set of rules designed for connecting one system with another. The connecting process here is termed as remote login. The system which requests for connection is the local computer, and the system which accepts the connection is the remote computer.
10. Gopher: Gopher is a collection of rules implemented for searching, retrieving as well as displaying documents from isolated sites. Gopher also works on the client/server principle.

**Pros and cons of Single page and multi-page applications.**

Single Page Application Pros

Single-page applications are fast as most of the resources, including HTML, CSS, and Scripts, are loaded once, and only data is transmitted back and forth.

1. Quick Loading Time

A page taking over 200 milliseconds to load can significantly affect your online business and, eventually, sales. With the SPA approach, your full page loads quicker than traditional web applications, as it only has to load a page at the first request. On the other hand, traditional web apps have to load pages at every request, taking more time.

2. Seamless User Experience

SPAs deliver an experience like a desktop or mobile app. Users do not have to watch a new page load, as only the content changes and not the page, making the experience an enjoyable one.

3. Ease in Building Feature-rich Apps

SPA application makes it easy to add advanced features to a web application. For example, it is easier to build a content editing web app with real-time analysis using SPA development. Doing this with a traditional web app requires a total page reload to perform content analysis.

4. Uses Less Bandwidth

It is no surprise that SPAs consume less bandwidth since they only load web pages once. Besides that, they can also do well in areas with a slow internet connection. Hence, it is convenient for everyone to use, regardless of internet speed.

Single Page Application Cons

Single page application architecture is best for developing high-performing SAAS platforms and social networks. However, this approach has some disadvantages that make it unsuitable for developing highly secure, and SEO optimized websites.

1. Doesn’t Perform Well With SEO

One of the metrics that search engines use is the number of pages a website has. However, since SPAs only load a single page, it serves as a disadvantage when ranking on search engines

2. Uses a Lot of Browser Resources

SPAs require many resources from the web browser since the browser is doing most of the tasks for the SPAs. Creating SPAs often need users to use the latest browsers with support for some modern features.

3. Security Issues

As compared to multi-page apps, SPAs are more prone to cross-site scripting attacks. Using XSS, it becomes easy for hackers to introduce client-side scripts into a web app. Also, SPAs are more likely to expose sensitive data to all users.

Pros of the Multiple-Page Application:

1. It’s the perfect approach for users who need a visual map of where to go in the application. Solid, few level menu navigation is an essential part of traditional Multi-Page Application.
2. Very good and easy for proper SEO management. It gives better chances to rank for different keywords since an application can be optimized for one keyword per page.

Cons of the multiple-page application:

1. There is no option to use the same backend with mobile applications.
2. Frontend and backend development are tightly coupled.
3. The development becomes quite complex. The developer needs to use frameworks for either client and server side. This results in the longer time of application development.