Harsh Sharma, Intern Web Apps, SFIN1350

1. How HTTPS works behind the scenes?

HTTPS stands for Hypertext Transfer Protocol Secure, which is a secure version of the HTTP protocol. It uses SSL/TLS (Secure Socket Layer/Transport Layer Security) protocol to provide secure communication over the internet. The SSL/TLS protocol provides encryption and decryption of data to ensure that the data transferred between the client and the server is secure and cannot be intercepted by a third party.

When a user tries to access an HTTPS website, the browser sends a request to the server to establish a secure connection. The server responds by sending a digital certificate to the browser to verify its identity. The browser verifies the certificate and if it is valid, it generates a unique symmetric key that will be used to encrypt and decrypt the data. The browser then sends a request to the server to start the encrypted session. The server responds with an acknowledgement message and the encrypted session begins.

During the encrypted session, the data is encrypted at the sender's end and decrypted at the receiver's end using the symmetric key. This ensures that any third party attempting to intercept the data will not be able to read or modify it.

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

There are several HTTP methods available, but the most used ones are:

- GET: Used to retrieve data from a server.
- POST: Used to submit data to a server.
- PUT: Used to update data on a server.
- DELETE: Used to delete data from a server.
- HEAD: Like GET, but only retrieves the headers of a response.
- OPTIONS: Used to determine the HTTP methods supported by a server.

2.1. Difference between PUT and Patch

PUT and PATCH are two HTTP methods used in web development for updating resources on a server. They are part of the HTTP protocol and are commonly used in REST API's.

- **PUT**:- The PUT method is used to update an entire resource or create a new resource at a specific URL.
- When you send a PUT request to a specific URL, it replaces the entire existing resource with the new representation provided in the request.

- If the resource does not exist at the specified URL, a new resource will be created with the given data.
- It means that if you want to update a single attribute of a resource, you need to send the complete representation of the resource with the updated attribute(s).
- PUT requests are idempotent, which means multiple identical requests have the same effect as a single request (i.e., making the same PUT request multiple times will not have any additional impact).

For Example: -

PUT/api/users/123
{ “name: “harsh”,
"age": 26,
“email” : “”
}

PATCH:-

- The PATCH method is used to make partial updates to a resource.
- Unlike PUT, which replaces the entire resource, PATCH applies only the changes specified in the request to the existing resource, leaving other attributes unchanged.
- It allows you to update specific fields of a resource without having to send the entire representation of the resource.
- **For Example**: -
  PATCH /api/users/123
  {
  "age": 31
  }

Partial update

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

HTTP response codes indicate the status of a request made by a client to a server. Some of the most used HTTP response codes are:

- 200 OK: Indicates that the request was successful.
- 201 Created: Indicates that a new resource has been successfully created on the server.
- 400 Bad Request: Indicates that the server cannot understand the request due to a client error.
- 401 Unauthorized: Indicates that the client does not have the necessary credentials to access the requested resource.
- 403 Forbidden: Indicates that the client does not have access to the requested resource.
- 404 Not Found: Indicates that the requested resource could not be found on the server.
- 500 Internal Server Error: Indicates that the server encountered an error while processing the request.

3.1. HTTP Code Ranges

HTTP response codes are three-digit numbers sent by the server in response to a client's request. These codes indicate the status of the requested operation and help the client to understand the outcome of the request. The response codes are grouped into several ranges, each with its significance. Here are some of the most common HTTP response code ranges and their meanings:

1. **Informational Response(1xx): 100-199**
    - These are informational responses indicating that the request was received, and the server is continuing to process it.
    - Examples: 100 Continue, 101 Switching Protocols.
2. **Success Response (2xx):200-299**,
    - These codes indicate that the request was successfully received, understood, and accepted.
    - Examples: 200 OK, 201 Created, 204 No Content.
3. **Redirection Response (3xx): 300-399**
    - These codes indicate that further action is required to complete the request, often involving redirects to a different URL.
    - Examples: 301 Moved Permanently, 302 Found, 307 Temporary Redirect.

4. **Client Errors (4xx):400-499**
    - These codes are returned when the client's request is invalid or cannot be fulfilled.
    - Examples: 400 Bad Request, 401 Unauthorized, 404 Not Found, 403 Forbidden.
5. **Server Errors (5xx):**
    - These codes are returned when the server encounters an error while processing the request.
    - Examples: 500 Internal Server Error, 502 Bad Gateway, 503 Service Unavailable.

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

There are several web communication protocols available, but some of the most used ones are:

- HTTP: Hypertext Transfer Protocol is used for transferring data over the internet.
- HTTPS: Hypertext Transfer Protocol Secure is a secure version of HTTP that uses SSL/TLS protocol to provide secure communication.
- FTP: File Transfer Protocol is used for transferring files over the internet.
- SMTP: Simple Mail Transfer Protocol is used for sending email over the internet.
- POP3: Post Office Protocol 3 is used for retrieving email from a server.

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

Single-page applications (SPAs) and multi-page applications (MPAs) are two common approaches for building web applications. Here are some pros and cons of each:

Pros of Single-page applications:

1. Faster and more responsive: SPAs load the entire application into the browser at once, so subsequent requests for data can be handled without reloading the page. This makes SPAs faster and more responsive, as users don't have to wait for a new page to load every time they want to interact with the application.

2. More engaging user experience: SPAs often provide a more seamless and engaging user experience, as users can interact with the application without being interrupted by page reloads. This can be particularly useful for applications that require a lot of user input or navigation.
3. Easier to build complex user interfaces: Because SPAs use client-side rendering, developers can build complex user interfaces using frameworks like React or Angular without having to worry about server-side rendering and page refreshes.

Cons of Single-page applications:

1. Slower initial load time: Because SPAs load the entire application at once, the initial load time can be slower than with MPAs. This can be particularly noticeable on slow connections or older devices.
2. SEO challenges: Because SPAs often rely on JavaScript to render content, search engine crawlers may have difficulty indexing the site, which can hurt SEO.
3. Increased complexity: SPAs can be more complex to build and maintain than MPAs, particularly as the application grows and complexity.

Pros of Multi-page applications:

1. Better for SEO: MPAs can be better for SEO because each page can have its own meta tags and URL, making it easier for search engine crawlers to index the site.
2. Faster initial load time: MPAs only load the content that is needed for each page, which can make the initial load time faster than with SPAs.
3. Simpler architecture: MPAs typically have a simpler architecture than SPAs, which can make them easier to build and maintain.

Cons of Multi-page applications:

1. Slower navigation: Because MPAs require a new page load for each request, navigation can be slower and less responsive than with SPAs.
2. More server requests: MPAs require more server requests than SPAs, which can put a greater strain on the server and slow down the application.
3. Less engaging user experience: Because MPAs require page reloads for each request, the user experience can be less seamless and engaging than with SPAs.