**Hyper Text Transfer Protocol:** The HTTP protocol is a request/response protocol based on the client/server based architecture where web browsers, robots and search engines, etc. act like HTTP clients, and the Web server acts as a server. Communication between clients and servers is done by requests and responses. The HTTP methods:

1. Post : Send data to the server to create or update data
2. GET: requesting for data.
3. PUT: Send data to the server in order to send or update data request. The only difference between PUT and Post is the fact that PUT can carry the same request multiple times which will produce the same output.
4. Head : Same as GET but only without any response body
5. Delete: deletes the specified request.
6. Patch: A PATCH request is considered a set of instructions on how to modify a resource. Contrast this with PUT; which is a complete representation of a resource.
7. TRACE: echoes the received request to check client usability.

**HTTPS**: Extension of HTTP but this is used for secured communication over CN. This is mostly used for internet.

**Representational state transfer (REST):** software architecture that constraints web services. The rules of rest.

1. Offer access through resources.
2. Represent resources by representations. In a REST system, addresses identify things, not formats, so all systems use the same address for the same thing.
3. Connect resources through links. Through the use of link any inside resource can be identified by the developer.

**Application program interface (API):** Acts as a language boundless service. Any kind of system such as web, mobile can call it and get a JSON format response.

**JavaScript Object Notation (JSON):** A syntax for data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.

**Client Server Model:** In this model the client (the computer that asks/requests for the data through web) asks for data to the server (who stores resources, hosts websites, store data). The serve runs 24/7 in search for a web request. This model is an architecture that splits computer into two parts namely the clients who send request and the computer that responds to that request (server). It follows **request-respond cycle** method HTTP messages.

Different kinds of servers: web servers (apache), database server (mysql), file server, Virtual server (google colab).

**MVC**:

**Model**: It includes all the data and its related logic.

**View**: Present data to the user or handles user interaction.

**Controller**: An interaction between the Model and the view.

In this architecture the user interacts with the view. The view alerts the controller about any event. The controller updates the model. Consequently, the view is alerted by the view about changes. Then the view changes accordingly to the model.

**Cookies**: Cookies are messages that web servers pass to the web browser when you visit web sites. The browser stores each message in a small file and uses it for sending to the server. This is stored on the client side.

**Sessions**: A session is a way to store information (in variables) to be used across multiple pages. This happened in server side.

**Routes:** Routing is a mapping between an URL and the functionality of a web server. It is like the mechanism to decide which functionality to call based on the provided URL and parameters.

Q : what is chlickjacking and how to prevent it?

**Clickjacking:** Clickjacking is an attack that tricks a user into clicking a webpage element which is invisible or disguised as another element.  It works as a bait for unwittingly download malware, get passwords, transfer money or buy online product.

# identification of clickjacking

<html>

<head>

<title>Clickjack test page</title>

</head>

<body>

<p>Website is vulnerable to clickjacking!</p>

<iframe src="http://www.yoursite.com/sensitive-page" width="500" height="500"></iframe>

</body>

</html>

If the text “Website is vulnerable to clickjacking” appears and below it you see the content of your sensitive page, the page is vulnerable to clickjacking.

**Prevention:**

1. Frame Killing: When a page loads the java script will check for the domain of that page for matching the domain of the browser window. If it is not found to be true then it will not load in the frame. This means no other frame can display in my frame.

<style>

/\* Hide page by default \*/

html { display : none; }

</style>

<script>

if (self == top) {

// Everything checks out, show the page.

document.documentElement.style.display = 'block';

} else {

// Break out of the frame.

top.location = self.location;

}

</script>

//java code

public void doGet(HttpServletRequest request, HttpServletResponse response)

{

response.addHeader("X-Frame-Options", "DENY");

response.addHeader("Content-Security-Policy", "frame-ancestors 'none'");

}

1. Using X Frame –Options : The X frame options HTTP Header can be used to indicate whether or not a browser should be allowed to render a page in a <frame>, <iframe> or <object> tag. It was designed specifically to help protect against clickjacking.

add\_header X-Frame-Options "sameorigin" always; //enabling x frame options

header always set X-Frame-Options "sameorigin" //enable X frame on apache

//enable on IIS

<system.webServer>

...

<httpProtocol>

<customHeaders>

<add name="X-Frame-Options" value="sameorigin" />

</customHeaders>

</httpProtocol>

...

</system.webServer>