<u>TODAYS session includes:</u>

Cookies, Client-server communication – methods (GET, POST, HEAD, PUT, DELETE), safety and idempotence, URL syntax, Response codes, URL encoding and decoding, Server-side: Environment,
****

Cookies:

- Cookies are meant for personalization of webpages. Information about users is stored during or between sessions.
- They are ***cookie-value*** pairs stores in **document.cookie** as a local file.
- **document.cookie** stores all cookies
- cookies have expiry date and set using **expires** property (if such property is not set then the cookie expires as soon as the page is closed)
- cookies that disapper when the browser is closed are called "session cookies"
- alert(document.cookie);//cookie1=value1; cookie2=value2;...

setting a cookie (expires after 5 minutes):

```
let date = new Date(Date.now() + 5*60*1000);
exdate = date.toUTCString();
document.cookie = "user=alpha; expires=" + exdate;
```

Date Object:

toUTCString() : Converts a Date object to a string, according to universal time

now() : Returns the number of milliseconds since midnight Jan 1, 1970

Update a cookie

```
document.cookie="user=beta";
```
****

Web servers: high end PCs with complex hardware and software

Web Application: is a collection of related files and folders in a hierarchy. .html, .xml, .txt, .php, .jpg ……..

Server has knowledge of the types of files and their locations.

Client – browser and Server – web server

HTTP – hypertext transfer protocol

HTTPS – secure ………..

RESOURCE

HTTP request methods: (GET, POST, HEAD, PUT, DELETE)

<form action="www.myserver.com/check.php" method="GET">

Request methods are considered "**safe**" if their defined semantics are essentially read-only; i.e., the client does not request, and does not expect, any state change on the origin server as a result of applying a safe method to a target resource.

Safe methods - GET, HEAD

WWW.myserver.com/check.php?search=computer+science

A request method is considered "**idempotent**" if the intended effect on the server of multiple identical requests with that method is the same as the effect for a single such request.

Idempotent Methods – GET, HEAD, PUT, DELETE

Methods for Caching:

GET, HEAD:

Method Definitions:

GET: The GET method requests transfer of a current selected representation for the target resource. Associated response code: 200 OK.

HEAD: The HEAD method is identical to GET except that the server MUST NOT send content in the response. HEAD is used to obtain metadata about the selected representation without transferring its representation data, often for the sake of testing hypertext links or finding recent modifications.

POST: The POST method requests that the target resource process the representation enclosed in the request according to the resource's own specific semantics. NOT CACHED

PUT: The PUT method replaces all current representations of the target resource with the request payload.

DELETE: The DELETE method requests that the origin server remove the association between the target resource and its current functionality.

URI: UNIFORM RESOURCE IDENTIFIER:

Scheme or protocol://domain/path

URL: UNIFORM RESOURCE LOCATOR

Protocol or scheme://host:port/path?query

Ex: http://www.google.com/abc.php?querydata
192.128.160.32/
Root / host/ domain
http://127.0.0.1:8080   is same as   http://localhost:8080

URL encoding and decoding:

Symbols used {=,&, +, %xx}
= (to separate name and its corresponding value as NAME=VALUE)
&(to separate name value pairs as name1=value1&name2=value2)
+ (to encode blank spaces)
%xx(to encode other characters with a % followed by hexadecimal characters)

LETS move on to XAMPP

X- platform independent
A – Apache
M- MariaDB
P-PERL
P-PHP

ADP EXERCISES (31/05/2023)

1. Implement cookie functionality through a function, using document.cookie property.
2. Understand XAMPP server environment and take observations on the configuration files of apache (apache/conf/httpd.conf) and php(php/php.ini)

3. Implement simple client-server communication using GET and POST methods and take observations.

- Create a html page on the server and through GET request from a <form> display the page
- Create a .php page on the server and print the contents of the page through GET request.
- Create a .php page on the server that receives form data through GET/POST and sends all the form data back to the client in a HTML table.
  (Use echo, $_GET[" "], $_POST[" "], to print / retrieve the variables)
- Observe, embedding php inside html, and html output by php echo.

/sample code snippet: and both the .html and .php should reside on the server.

```
//simpleform.html
<html>
<body>
<form action="getdetails.php" method="get">
Name: <input type="text" name="name"><br>
Branch: <input type="text" name="branch"><br>
<input type="submit">
</form></body></html>

//getdetails.php
<?php
?>
<html>
<body>
Welcome <?php echo $_GET["name"]; ?><br>
Your branch is: <?php echo $_GET["branch"]; ?>
</body></html>
```

//refer to php cheatsheets for more

4. Start MySQL service in the XAMPP control panel . Access phpmyadmin through the link http://localhost/phpmyadmin. Spend time to understand the dash board (note the configuration of database server and web server shown on the right panel)

/