**WEEK 3:**

1. **HTPP METHODS:**

HTTP (Hypertext Transfer Protocol) is used for communication on the World Wide Web. HTTP methods are the actions or verbs that indicate what a client wants to do with a resource.

Here are some common HTTP methods:

**GET**: Retrieve data from a specified resource. It should have no side effects on the server.

**POST**: Submit data to be processed to a specified resource. It may cause side effects, such as the creation or update of a resource.

**PUT:** Update a resource or create a new resource if it does not exist at the specified URL.

**DELETE:** Delete the specified resource.

**PATCH:** Apply partial modifications to a resource.

**HEAD:** Retrieve the headers of a resource without the actual data.

**OPTIONS**: Get information about the communication options available for the target resource.

**TRACE**: Perform a message loop-back test along the path to the target resource.

1. **HTPP MECHANISMS**

Hashing is a process of converting input data (or 'message') into a fixed-size string of characters, which is typically a hexadecimal number. Hash functions are commonly used in computer science and cryptography for various purposes.

Here are some key aspects related to hashing mechanisms:

**Hash Function:**

A hash function takes input data (or message) and produces a fixed-size string of characters, which is usually a hash value or hash code. It should be deterministic, meaning that the same input will always produce the same hash output. It should be fast to compute the hash value for any given data. Generating the same hash value from different inputs (collision resistance) should be infeasible.

**Common Hash Functions:**

**MD5 (Message Digest Algorithm 5):** This hash function produces a 128-bit hash value, typically rendered as a 32-character hexadecimal number. However, MD5 is considered weak for cryptographic purposes due to vulnerabilities.

**SHA-1 (Secure Hash Algorithm 1):** SHA-1 produces a 160-bit hash value. Like MD5, SHA-1 is no longer considered secure for cryptographic purposes due to vulnerabilities.

**SHA-256, SHA-384, SHA-512:**

These are part of the SHA-2 family and produce hash values of 256, 384, and 512 bits, respectively. They are currently considered secure and widely used for various cryptographic applications.

**Applications of Hashing:**

**Data Integrity:** Hash functions are used to verify the integrity of data. If the hash value of the original data matches the hash value of the received or stored data, it indicates that the data has not been altered.

**Password Storage:** Hash functions are used to store password hashes securely. Instead of storing plain-text passwords, systems store the hash values, making it more difficult for attackers to retrieve the original passwords.

**Digital Signatures:** Hash functions are used in digital signatures to ensure the integrity of the signed data.

**Cryptographic Hash Functions:** Cryptographic hash functions have additional properties, such as being resistant to preimage attacks (given a hash value, it should be computationally infeasible to find the original input) and collision attacks (it should be infeasible to find two different inputs that produce the same hash value).

1. **RUBBER DUCK DEBUGGING**

"Rubber duck debugging" is a concept used in programming and debugging where a programmer explains their code or problem to an inanimate object, often a rubber duck. The idea is that articulating the problem out loud can help the programmer understand it better and potentially find a solution.

**Get a Rubber Duck:** The programmer acquires a rubber duck or any other inanimate object

t. **Explain the Code:** The programmer goes through the code line by line, explaining each part in detail to the rubber duck. This includes the purpose of the code, expected behavior, and any issues encountered.

**Articulate the Problem:** While explaining the code, the programmer may realize where the problem or bug lies. The act of verbalizing the code often helps in identifying logical errors, misconceptions, or overlooked details.

**Problem Solving:** The programmer may continue the conversation with the rubber duck, discussing potential solutions and debugging strategies. This can lead to insights and a better understanding of the code.

1. **ROBOTS.txt**

The robots.txt file is a text file that webmasters create to instruct web robots (often called web crawlers or spiders) how to interact with the pages of their website. The file is typically placed at the root of a website's domain, and its primary purpose is to communicate with search engines and other web robots about which parts of the site should not be crawled or indexed.

**Format:** The robots.txt file is a simple text file that follows a specific format. It typically contains a series of directives, each specifying a set of rules for web robots.

**User-Agent:** The User-Agent field is used to specify the web robot or crawler to which the rules apply. Different web robots may have different behaviors, so site owners can tailor the instructions accordingly.

**Disallow:** The Disallow directive is used to specify the URLs or directories that the web robot should not crawl. For example, if a line in the robots.txt file says Disallow: /private/, it means that the web robot should not crawl any URL starting with "/private/".

**Allow:** The Allow directive can be used to override Disallow directives, indicating specific areas that are allowed to be crawled even if a broader restriction is in place.

1. **SITEMAP.xml**

A sitemap.xml file is an XML file that contains a list of URLs from a website along with additional information about each URL. This file is used to help search engines understand the structure and content of a website, making it easier for them to crawl and index the site's pages. The sitemap.xml file is one of the tools that website owners and developers use to improve the search engine optimization (SEO) of their sites.

**Purpose:** The primary purpose of a sitemap.xml file is to provide search engines with information about the URLs on a website, including metadata about each URL, such as when it was last modified and how frequently it changes.

**Structure**: The sitemap.xml file is structured in XML format, which is a markup language similar to HTML but designed to be easily readable by machines. It typically includes a set of <url> elements, each containing information about a specific URL on the website.

**URL Information:** Each <url> element may include sub-elements such as <loc> (URL of the page), <lastmod> (date of last modification), <changefreq> (how often the page changes), and <priority> (priority of the URL relative to other URLs on the site).

**Location:** The sitemap.xml file is usually placed in the root directory of the website, and its location is specified in the website's robots.txt file to inform search engines where to find it.