Sr. Software Engineer (SSE)

ABSTRACT
This is one of the subject from my personal notes series named "Coding-With-Arqam" that I am developing from the start of my professional development career.

Subject
Web Development

WEB DEVELOPMENT

--> Notes:

-> Uber is using Mongodb.

-> GraphQL developed by facebook.

-> Naming conventions help in automation and testing.

-> <link> elements to apply external stylesheets (CDK) and <style> elements to apply internal stylesheets to HTML.

-> <script> is used to add external JS.

-> Tutorial Point is in ".htm" not in ".html". It is Used for constructing web pages.

-> Date should be stored in UTC (Coordinated Universal Time) with a special UTC designator ("Z").

--> Hypertext:

-> A software system allowing extensive cross-referencing between related sections of text and associated graphic material.

-> A document presented on a computer in hypertext.

-> Hypertext is text which contains links to other texts.

--> Anchor and Links:

-> The links embedded in a document are known as hyperlinks.

--> Hypermedia:

-> HyperMedia is a term used for hypertext which is not constrained to be text: it can include graphics, video and sound.

--> World Wide Web:

-> Commonly known as the Web.

-> Information system where documents and other web resources are identified by Uniform Resource Locators (URLs), which may be interlinked by hypertext, and are accessible over the Internet.

-> Web servers and clients may be located at any part of the world and connected to each other by telecommunication links.

-> Web is in some sense a digital library with no single location.

--> Pages:

-> Static and Dynamic.

-> Dynamic Page:

-> A dynamic web page is a web page that displays different content each time it's viewed.

-> Like we create components within layouts in Angular.

-> Types:

-> CLIENT-SIDE SCRIPTING:

-> Web pages that change in response to an action within that web page, such as a mouse or a keyboard action, use client-side scripting.

-> Client-side scripts generate client-side content. Client-side content is content that's generated on the user's computer rather than the server.

-> Scripting languages such as JavaScript and Flash allow a web page to respond to client-side events.

-> SERVER-SIDE SCRIPTING:

-> Web pages that change when a web page is loaded or visited use server-side scripting.

-> Server-side content is content that's generated when a web page is loaded.

-> For example, login pages, forums, submission forms, and shopping carts, all use server-side scripting since those web pages change according to what is submitted to it.

-> Scripting languages such as PHP, ASP, ASP.NET, JSP, ColdFusion and Perl allow a web page to respond to submission events.

--> Network Protocols:

-> Standard way of regulating data transmission between computers.

-> Two prominent protocols are:

TCP (Transaction Control Protocol) and IP (Internet Protocol), together known as TCP/IP

-> Rules, procedures and formats that defines communication between two or more devices over a network.

-> Layers from Bottom to Top:

-> Physical -> Data Link(Ethernet) -> Network(IP) -> Transport(TCP) -> Session -> Presentation -> Application(HTTP)

--> Web Application (Webapp):

-> Unlike standalone application, runs over the Internet.

-> E.g: google, amazon, ebay, facebook and the UCT website.

-> Tiers: (mainly three tiers) -> client, server & database application run over the Internet.

-> Components: (five components):

-> HTTP Server:

-> E.g: Apache HTTP Server, Apache Tomcat Server, Microsoft Internet Information Server (IIS), nginx, Google Web Server (GWS),etc.

-> HTTP Client (or Web Browser):

-> E.g: Internet Explorer (MSIE), FireFox, Chrome, Safari, and others.

-> Database:

-> E.g: Open-source MySQL, MariaDB, Apache Derby, SQLite, PostgreSQL, OpenOffice's Base; Commercial Oracle, IBM DB2, SAP SyBase, MS SQL Server, MS Access; and others. You

-> Client-Side Programs:

-> could be written in HTML Form, JavaScript, VBScript, Flash, and others.

-> Server-Side Programs:

-> could be written in Java Servlet/JSP, ASP, PHP, Perl, Python, CGI, and others.

--> Uniform Resource Locator (URL):

-> http://machine_name:port/file_name.file_extension

-> machineName = IP, DNS

-> port = optional part of url.

-> path = relative file path from the server's document root

-> fileExtension = like .html

--> HyperText Markup Language (HTML):

-> A language provides the format for specifying simple logical structure and links in a hypertext document.

-> Versions: HTML 2.0 (Basic), HTML 4.01 (major), HTML 5(latest published in 2012).

--> HTML VS HTM:

-> Extension:

-> .html is being used as extension for "HTML page" or file => extension of "HTML file" or page.

-> Use:

-> Currently => Old

--> HTML vs XML:

-> Not case sensitive => Yes

-> Presentation language => XML is neither a presentation nor a programming language

-> Pre-defined tags and cannot be overriden => Users can define their tags as per use.

-> Notepad, Notepad++ are some of the common free tool => XML editors (generally used).

--> HyperText Transfer Protocol (HTTP):

-> Network protocol used to retrieve documents from a variety of machines in a minimum amount of time.

-> Port: 80 (commonly)

--> HyperText Transfer Protocol Secure (HTTPS):

-> Port: 443 (commonly) (SSL)

--> File transfer protocol (FTP):

-> Port: 21 (commonly)

--> File transfer protocol Secure (FTPS):

-> Port: 21/990 (commonly) (SSL/TLS)

--> MIME Types:

-> Multipurpose Internet Mail Extensions.

-> Indicates the nature and format of a document, file, or assortment of bytes.

-> E.g: .bin, .bmp, .torrent, .7z, etc.

--> Web Servers:

-> Types of servers: FTP servers, gopher servers, Web servers, etc.

--> IFrames:

-> Inline frame.

-> An HTML document embedded inside another HTML document on a website.

-> The IFrame HTML element is often used to insert content from another source, such as an advertisement, into a Web page.

-> Iframe is an inline frame and it is also used as <iframe> tag in HTML.

-> Iline frame means it is used to embed some other document within the current HTML document.

-> An iframe (short for inline frame).

--> Markup Language:

-> Language that uses tags to define elements within a document.

-> Markup = marks up the page with tags.

-> Markup files contain standard words, rather than typical programming syntax.

--> HTML:

-> HTML is used to specify both the structure and the formatting of Web documents.

-> Discovered after XML.

--> XML:

-> A subset of SGML.

-> Extensible Markup Language.

-> Markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

-> XML is a markup language much like HTML.

-> It contains XML declaration, DTDs, text, elements, processing instructions, comments, entity references.

--> SGML:

-> Standard Generalized Markup Language.

-> A standard for how to specify a document markup language or tag set.

-> A standard for defining generalized markup languages for documents.

--> DTD:

-> Document type definition.

-> A set of markup declarations that define a document type for a SGML-family markup language.

--> SOAP VS REST:

-> REST APIs uses multiple standards like HTTP, JSON, URL, and XML while SOAP APIs is largely based on HTTP and XML.

-> SOAP API, has an official standard.

-> SOAP stands for Simple Object Access Protocol whereas REST stands for Representational State Transfer.

-> SOAP is a protocol whereas REST is an architectural pattern.

-> SOAP only works with XML formats whereas REST work with plain text, XML, HTML and JSON.

-> SOAP cannot make use of REST whereas REST can make use of SOAP.

-> Soap is Language, platform, and transport independent (REST requires use of HTTP) .

-> Data cannot be cahche in soap but can be in rest.

-> SOAP has built in error handling and ACID compliance where rest has not.

-> SOAP is just like an evelop (overhead) where rest is like a an open message while communication between server and app.

--> SGML -> XML -> HTML -> XHTML

--> Style Sheet (Style.css):

-> Collection of style rules that that tells a browser how the various styles are to be applied to the HTML tags to present the document.

-> Style Sheets can be external or internal to your Web site and/or external or internal to your Web pages.

-> Cascading Style Sheets (CSS).

-> Each property has a name and a value, separated by a colon (:). Each property declaration is separated by a semi-colon (;)

-> The style sheet standard supported by modern browsers is called cascading style sheets.

-> CSS.CSS files contain a set of rules for the formatting of HTML documents.

-> Selector {property: value}

--> CSS vs SASS vs LESS vs SCSS:

-> CSS:

-> Cascading style sheets (CSS) set the presentation of HTML pages.

-> CSS has its limits.

-> SASS:

-> Variables:

-> you can save information in variables

-> Mathematical functions:

*-> mathematical operations like +, -, *, /, or %.*

-> Functions, Loops, Case distinctions(checks), Mixins, Indentations, Nesting, Inheritances.

-> LESS:

->

->

->

-> SCSS:

-> SCSS contains all the features of CSS and contains more features that are not present in CSS which makes it a good choice for developers to use it.

-> SCSS offers variables, you can shorten your code by using variables.

-> Variable like $color: #f00 , then use this variable further in the file.

--> Front-end Classes:

-> Sometimes you may wish to give different formats to different paragraphs in the same HTML document.

--> Id VS Class:

-> ID can be used to identify one element.

-> A class can be used to identify more than one.

-> ID's and Classes are "hooks".

-> Default basic elements are : h1, h2, p, ul, li, etc.

-> ID's are unique.

-> Each element can have only one ID

-> Each page can have only one element with that ID

-> ID's have special browser functionality

-> ID's do have one very important trick up their sleeve. This is the "hash value" in the URL.

-> http://yourdomain.com#comments, the browser will attempt to locate the element with an ID of "comments" and will automatically scroll the page to show that element.

-> Elements can have BOTH. E.g: <li id="comment-27299" class="item">

--> Logical Address VS Physical Address:

-> Logical address is generated by CPU in perspective of a program.

-> physical address is a location that exists in the memory unit.

--> Design patterns:

-> Programming language independent strategies for solving a common problem.

-> A design pattern represents an idea, not a particular implementation.

-> By using the design patterns you can make your code more flexible, reusable and maintainable.

-> Types in JS:

-> Immediately Invoked Function Expressions (IIFE):

-> Allows you to define and call a function at the same time.

```
->    return function ()
              {
              number++
              return number
              }
```

-> Factory method pattern:

-> Acts as a tool you can implement to clean your code up a bit.

-> Allows you to centralize the logic of creating objects (meaning, which object to create and why) in a single place.

-> Singleton pattern:

-> Helps you keep track of how many instances of a class you're instantiating.

-> Observer pattern:

-> Chain of responsibility:

--> Why the Internet is Insecure.

-> Developers are often not very well versed with secure programming techniques.

--> Mock tables:

Mock tables are guides for programming final tables to present summaries of clinical study data.

They are most commonly created using a word processor, e.g., Microsoft Word, and are often tedious to create from scratch or modify afterwards when columns and rows are added, deleted, or reformatted.

--> Inheritance:

Protected Inheritance − When deriving from a protected base class, public and protected members of the base class become protected members of the derived class.

Private Inheritance − When deriving from a private base class, public and protected members of the base class become private members of the derived class.

--> Cookies:

Cookies generally do not contain any information that would identify a person.

Usually they contain a string of text or "unique identifier". This acts like a label.

When a website sees the string of text it set in a cookie, it knows the browser is one it has seen before.

A cookie (called an Internet or Web cookie) is the term given to describe a type of message that is given to a web browser by a web server. The main purpose of a cookie is to identify users and possibly prepare customized Web pages or to save site login information for you.

Info like targeting audience, etc for analysis purpose. It doesn't keeps priv info like pass, etc.

--> Cache:

-> The need for cache comes from the fact that generating results based on the database is costly.

-> An in-memory store that holds a subset of information typically stored in the database.

-> Caching comes with a cost. Only some subsets of information can be stored in memory.

-> The most common data pruning strategy is to evict items that are least recently used (LRU).

-> A lot of modern web applications, including Facebook, rely on a distributed caching system called Memcached.

-> Memcached:

-> Used to speed up dynamic database-driven websites by caching data and objects in RAM to reduce the number of times an external data source (such as a database or API) must be read.

-> Memcached runs on Unix-like operating systems (at least Linux and OS X) and on Microsoft Windows.

-> It depends on the libevent library.

-> Using in fb, yt, twitter, wikipedia, azure,amazon, etc.

-> The system uses a client–server architecture. The servers maintain a key–value associative array.

-> E.g: memcached_fetch(":xyz_seed:")

--> Hashing:

-> The idea behind hashing is fast access to data.

-> If the data is stored sequentially, the time to find the item is proportional to the size of the list.

-> Perfecting hashing is difficult to acheive but improves a lot of look-up time.

-> Example:

->

--> Apache VS Tomcat(Servelet):

In simple words, Apache is a web-server meant to serve static web-pages.

Apache Tomcat, on the other hand, is an application server meant to serve Java applications (Servlets, JSPs etc).

You can serve web-pages as well through Tomcat, but it is less efficient at that as compared to Apache.

IRCTC is one such website.

--> WWW vs WWW3:

WWW stands for WorldWideWeb and is the Standard Sub-Domain ob a Webpage (normaly works also without).

Often an other Sub-Domain (e.g. www3 etc.) is used to load e.g. images from an other Server -

so Domain is equal but Sub-Domain is different means that can be different server machines e.g. for loadbalancing etc.

--> RFP:

Request for proposal (RFP) is a document that solicits proposal, often made through a bidding process,

by an agency or company interested in procurement of a commodity, service, or valuable asset, to potential

suppliers to submit business proposals.

--> Redux:

-> This library is based on the Flux pattern.

-> React:

-> Redux is a reactive state management library developed by Facebook and used in the React library.

-> Angular:

-> To use Redux in the Angular framework, we can use the NgRx library.

-> This is a reactive state management library. With NgRx, we can get all events (data) from the Angular app and put them all in the same place (Store).

-> With NgRx, we can get all events (data) from the Angular app and put them all in the same place (Store).

--> Redux VS Flux:

-> The main difference between Flux and Redux is how they handle actions.

-> In the case of Flux, we usually have multiple stores and a dispatcher, whereas Redux has a single store, which means a dispatcher is not needed.

--> Standardized web technologies:

-> HTML, CSS, JavaScript

--> Navigation:

-> Linear:

-> Forward/Back.

-> Web apps use it.

-> Non-Linear:

-> Seperate navigation stacks for each tab.

-> Mobile apps use it.

--> Mobile Stores:

-> App Store (IOS)

-> Play Store (Android)

--> Serverless Programming:

-> A cloud computing execution model in which the cloud provider runs the server, and dynamically manages the allocation of machine resources.

--> Stateful VS Stateless:

-> Both Stateless and stateful protocols are the network protocols for the web servers and web browsers.

-> In Stateless, server is not needed to keep the server information or session details to itself.

-> In stateful, a server is required to maintain the current state and session information.

-> Stateful means the computer or program keeps track of the state of interaction, usually by setting values in a storage field designated for that purpose

-> Stateless means there is no record of previous interactions and each interaction request has to be handled based entirely on information that comes with it.

-> For example, an operation to sort a list/array in a stateful language will probably do it destructively, while in a stateless language it will return a fresh list/array.

-> HTTP is stateless protocol because each command is request is executed independently, without any knowledge of the requests that were executed before it.

-> A stateless protocol does not require the server to retain information or status about each user for the duration of multiple requests.

-> Stateful Protocol is a network protocol in which if client send a request to the server then it expects some kind of response, in case of no response then it resend the request. FTP (File Transfer Protocol), Telnet.

--> Rest VS GraphQL:

-> With REST, the server determines the shape and size of the resource whereas, in GraphQL, the server simply declares the available resources and the client can ask for exactly what it needs.

-> REST automatically puts caching into effect whereas GraphQL has no automatic caching system.

-> GraphQL is the better REST.

-> Over the past decade, REST has become the standard (yet a fuzzy one) for designing web APIs.

-> It offers some great ideas, such as stateless servers and structured access to resources.

-> However, REST APIs have shown to be too inflexible to keep up with the rapidly changing requirements of the clients that access them.

-> GraphQL was developed to cope with the need for more flexibility and efficiency.

-> It solves many of the shortcomings and inefficiencies that developers experience when interacting with REST APIs.

-> Major differences between REST and GraphQL when it comes to fetching data from an API.

-> Example:

-> In a blogging application, an app needs to display the titles of the posts of a specific user.

The same screen also displays the names of the last 3 followers of that user. (NOTE: Follow the link for details)

-> Example:

-> friends of friends in fb. means nesting

-> If we want to get all users name, in rest we call /users which also take other user info line dob, etc with name.

-> Actually graphQL sends a query which makes the things faster and efficient.

-> GraphQL Query Example: // Getting only the required data even from nested object.

```
query { Student( id:1 ){
        firstName
        lastName
        classes {
        className
                }
        }
}
```

--> CORS:

-> Means cross-domain requests.

-> Sometimes you might want to let other sites call your web API.

-> Stands For: Cross-origin resource sharing.

-> Cross Origin Resource Sharing (CORS) is a W3C standard that allows a server to relax the same-origin policy.

-> Using CORS, a server can explicitly allow some cross-origin requests while rejecting others.

-> Enabling CORS in express.js:

-> var cors = require('cors');

-> Enable globally: app.use(cors());

-> Enable for specific route: app.get('/products/:id', cors(), function (req, res, next) {// code here});

--> Compiled Languages VS Scripting Languages:

-> Compiled:

-> non-interpreted language.

-> Faster than scripting because they are first converted native machine code.

-> Scripting = interpreted language.

--> DRY Principal:

-> Don't Repeat Yourself.

-> Aimed at reducing repetition of information.

-> "Every piece of knowledge or logic must have a single, unambiguous representation within a system."

--> Web protocols:

-> Web communication protocols are technology used to transfer information across the internet.

-> Example:

-> a web browser uses these protocols to request information from a web server, which is then displayed on the browser screen in the form of text and images.

-> Types:

-> DHCP: Dynamic Host Configuration Protocol

-> DNS: Domain Name Service

-> FTP: File Transfer Protocol

-> HTTP: Hypertext Transfer Protocol

-> IP: Internet Protocol

-> POP3: Post Office Protocol version 3

-> SMTP: Simple Mail Transfer Protocol

-> SSL: Secure Sockets Layer

-> SSH: Secure Shell

-> TCP: Transmission Control Protocol

-> UPD: User Datagram Protocol

--> Cron Job:

-> A Linux utility which schedules a command or script on your server to run automatically at a specified time and date

-> A cron job is the scheduled task itself.

-> Cron jobs can be very useful to automate repetitive tasks.

-> starts from "Every Minute" to "Every Year".

--> Cloud Computing Services:

-> Hosting

-> Remote network of servers

-> Advantages:

-> Scalability, Data Security, Accessibility, Pay-Per-Use.

-> Clouding Environment:

-> Private: Cloud services are delivered by a single organization, data center or infrastructure to internal users.

-> Public : Third-party providers deliver services hosted in the cloud over the internet. Pay-per-use.

-> Hybrid : Capitalizes on the advantages of both public and private environments.

-> Categories:

-> SaaS:

-> Software as a service.

-> Software that's available via a third-party over the internet.

-> A software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted.

-> It is sometimes referred to as "on-demand software".

-> LaaS:

-> Cloud-based services, pay-as-you-go for services such as storage, networking, and virtualization.

-> Examples include Amazon Web Services used by Netflix to house their database of movie and television offerings.

-> PaaS:

-> Hardware and software tools available over the internet.

-> On-premise:

-> Software that's installed in the same building as your business.

--> Single Page Application:

-> SPA

-> Instead of clicking a link and loading a new page, all new data is loaded into the page via AJAX calls.

-> when the user "visits" a page, client-side JavaScript builds the page and content.

-> The role of the backend is to deliver just one page–the single page–to the user.

--> Boilerplate code / Boilerplate:

-> Sections of code that have to be included in many places with little or no alteration.

--> Verbose:

-> Verbose mode is an option available in many computer operating systems and programming languages that provides additional details as to what the computer is doing and what drivers and software it is loading during startup or in programming it would produce detailed output for diagnostic purposes

--> Tech stack:

-> The set of technologies an organization uses to build a web or mobile application.

-> It is a combination of programming languages, frameworks, libraries, patterns, servers, UI/UX solutions, software, and tools used by its developers.

-> FaceBook (IP Country = United States):

-> RxJS

-> Facebook

-> PHP

-> HTTP/2

-> React

-> Mautic

-> Google Analytics

-> Google (IP Country = United States):

-> Material Design Lite

-> GSAP

-> Google Font API

-> YouTube

-> Google Analytics

-> Angular JS

-> Modernizr

-> reCAPTCHA

-> Google Tag Manager

-> Google Web Server

-> YouTube (IP Country = United States):

-> Google Font API

-> Polymer

-> Google AdSense

-> Youtube

-> Google Analytics

-> HTTP/2

-> Java

-> Python

-> Reddit

-> OpenGSE

-> Most of the architecture was built from scratch (or atleast with something raw) which includes streaming, video conversions pipeline, digital fingerprint, recommendation algorithm etc.

-> W3 Schools:

-> Microsoft ASP.NET

-> Google Analytics

-> Google Tag Manager

-> Google Font API

-> Amazon ECS

-> Docker

-> Azure CDN

-> Amazon Web Services

-> DoubleClick for Publishers (DFP)

-> Prebid

-> Code Mirror

-> Google Plus

-> IIS

-> Windows Server

-> Uber's Mobile App:

-> Google's Location APIs

-> Twilio for Push Notices.

-> Google Cloud Messaging (GCM)

-> PayPal's

-> Presentation Tier:

-> client_frameworks: backbone angular

-> css_frameworks: less

-> Logic Tier:

-> languages: javascript python node java ObjectiveC ruby coffeescript

-> queues: kafka

-> search: elasticsearch

-> web_frameworks: rubyonrails

-> Data Tier:

-> databases: mysql postgres

-> kvstores: mongodb redis

-> bigdata: hadoop spark

-> Infrastructure Tier:

-> web_servers: nginx

-> os: android ios

-> Service Stack:

-> Collaboration: asana

-> Saas: sendgrid

-> Technology: Google Analytics

-> Slack:

-> Web client is written in a mix of JavaScript and ES6, with React. We use Electron to ship it as a desktop app.

-> Android client is written in a mix of Java and Kotlin.

-> IOS client is written in a mix of Objective C and Swift.

-> Medium:

-> Node.js with nginx.

-> Also use Java, one might guess for heavy-computational stuff.

-> Redis is used as well, probably for recommendation/collection feeds.

-> Medium relies heavily on AWS services, as they use EC2, EBS, S3, Route53, EMR, DynamoDB, SQS, CloudFront and SES. All their assets are being served via CloudFront.

-> On the front-end, I'm guessing they built their own JS framework.

--> Throughput:

-> Measure of how many units of information a system can process in a given amount of time.

--> Angular VS React:

-> Developed by: Google => Facebook

-> Laguange: JavaScript, HTML => JSX

-> Type: Open Source MVC Framework => Open Source JS Framework

-> Rendering: Client-Side => Server-Side

-> Packaging: Weak => Strong

-> Data-Binding: Bi-directional => Uni-directional

-> DOM: Regular DOM => Virtual DOM

-> App Architecture: MVC => Flux

-> Dependencies: It manages dependencies automatically. => It requires additional tools to manage dependencies.

-> Routing: It requires a template or controller to its router configuration, which has to be managed manually. => It doesn't handle routing but has a lot of modules for routing, eg., react-router.

-> Performance: Slow => Fast, due to virtual DOM.

-> Best For: It is best for single page applications that update a single view at a time. => It is best for single page applications that update multiple views at a time.

-> Debugging: Difficult because of printing long stack traces (because it is event driven) => Easy

-> Templates: Enhanceed HTML templates and forces and force to learn angular syntax => JSX is an optional processor for HTML.

-> Framework/Library: Framework => Library and can be paired with any type of language

-> Learning Curve: Angular has a deep learning curve as it has huge documentation and complex => Few lifecycle methods

-> Native Apps: Native script for native applications and iconic framework for Hybrid applications => React-native for native applications and react-native-render for cross-platform applications.

--> ReactJS VS React Native:

-> It uses a JavaScript library and CSS for animations. => It comes with built-in animation libraries.

-> It uses React-router for navigating web pages. => It has built-in Navigator library for navigating mobile applications.

-> It uses HTML tags. => It does not use HTML tags.

-> It provides high security. => It provides low security in comparison to ReactJS.

-> In this, the Virtual DOM renders the browser code. => In this, Native uses its API to render code for mobile applications.

--> React VS Vue:

-> Preferred Language: JavaScript/JavaScript XML => javascript/HTML

-> Size: 100 kilobytes (approx) => 60 kilobytes (approx)

-> Performance: Its performance is slow as compared to Vue.

-> Flexibility: Great => Limited

-> Data Binding: one-way data binding => both one-way and two-way data binding.

-> Long Term Support: It is suitable for long term supports. => It is not suitable for long term support.

--> Loopback VS Express:

-> If you're working for a small brochure application which needs some minimal APIs and content management you should be using Express. js with some npm package for SQL (Database). But if you're working on an Enterprise application where you need to work on some complex data models you should definitely go with Loopback.

-> Loopback 4 Control Cycle: Model -> Repository -> DataSource -> Connector -> Database

--> How HTML works:

-> The M stands for Mark-up; in other words, an HTML file contains both content and commands.

-> To separate content and commands, HTML reserves three characters: <, >, and &.

-> HTML tags are wrapped with the < and > characters, which means you can't use them for anything else.

-> So what if you want to write < or > in your content? You need to use HTML entities.

-> These are commands that embed special characters. Specifically these four entities:

-> Symbol: "<" (Less-than) => Entity Code: "&lt;"

-> Symbol: "&" (Ampersand) => Entity Code: "&amp;"

--> Web Security:

-> OWASP: Open Web Application Security Project

-> An online community that produces articles, methodologies, documentation, tools, and technologies in the field of web application security.

-> Top Attacks / vulnerabilities:

-> Injection:

-> A code injection happens when an attacker sends invalid data to the web application with the intention to make it do something that the application was not designed/programmed to do.

-> Most common example around this security vulnerability is the SQL query consuming untrusted data.

-> Example:

*-> Query: String query = "SELECT * FROM accounts WHERE custID = '" + request.getParameter("id") + "'";*

-> URL: http://example.com/app/accountView?id=' or '1'='1

-> Prevention:

-> Use save API.

-> Use Object Relational Mapping Tools (ORMs).

-> Note: Even when parameterized, stored procedures can still introduce SQL injection if PL/SQL or T-SQL concatenates queries and data.

-> Use LIMIT and other SQL controls within queries to prevent mass disclosure of records in case of SQL injection.

-> From these recommendations you can abstract two things:

-> Separation of data from the web application logic

-> Implement settings and/or restrictions to limit data exposure in case of successful injection attacks.

-> Broken Authentication:

-> Brute-forcing of user credentials attacks.

-> Brute force is the act of trying many possible combinations.

-> It allow an attacker to use manual and/or automatic methods to try to gain control over any account they want in a system.

-> Usually refers to logic issues that occur on the application authentication's mechanism, like bad session management prone to username enumeration.

-> CAPTCHA usage is also another common mechanism used against brute-forcing.

-> There are two ways an attacker will brute force user credentials:

-> They'll attack a single user and try permutations of passwords until they hit a match.

-> They'll attack a multitude of users and use a list of the most commonly used passwords to shallowly brute force until they find a hit.

-> Types:

-> Weak passwords.

-> Weak Forgot password process.

-> Exposes session IDs in the URL (e.g., URL rewriting).

-> Sensitive Data Exposure:

-> There are two types of data:

-> Stored data – data at rest.

-> Transmitted data – data that is transmitted internally between servers, or to web browsers.

-> Protecting Data in Transit:

-> Through SSL (Secure Sockets Layer) certificate.

-> It is the standard security technology for establishing an encrypted link between a web server and a browser.

-> SSL certificates help protect the integrity of the data in transit between the host (web server or firewall) and the client (web browser).

-> Save sesitive data like password, credit card no, etc in encrypted form.

-> Strong hashing.

-> XML External Entities (XXE):

-> Attack against an application that parses XML input.

-> This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser.

-> Broken Access Control:

-> "Access Control" means putting a limit on what sections or pages visitors can reach, depending on their needs.

-> Example:

-> Access to a hosting control / administrative panel

-> Access to a server via FTP / SFTP / SSH

-> Access to a website's administrative panel

-> Access to other applications on your server

-> Access to a database

-> Prevention:

-> With the exception of public resources, deny by default.

-> Implement access control mechanisms once and reuse them throughout the application, including minimizing CORS usage.

-> Security Misconfigurations:

-> At its core, brute force is the act of trying many possible combinations, but there are many variants of this attack to increase its success rate.

-> Misconfiguration can happen at any level of an application stack, including storage, web servers, etc.

-> Example:

-> Unpatched flaws

-> Default configurations

-> Unused pages

-> Unprotected files and directories

-> Unnecessary services

-> XSS (Cross Site Scripting):

-> XSS attacks consist of injecting malicious client-side scripts into a website and using the website as a propagation method.

-> It allows an attacker to inject content into a website and modify how it is displayed, forcing a victim's browser to execute the code provided by the attacker while loading the page.

-> XSS is present in about two-thirds of all applications.

-> Types: Reflected XSS, Stored XSS, DOM XSS.

-> Protection:

-> Using frameworks that automatically escape XSS by design

-> A naive way to protect ourselves against XSS and code injection is to sanitize user input by doing the something like _.escape(userInput)

-> Input sanitization is a great first layer of defense, but isn't enough by itself.

-> Some more sturdy ways we can protect our user's sensitive data is to use an ORM / ODM, which enforces parameterized queries.

-> Or we could use Stored Procedures if we're using SQL, which defines the query procedure at the database level as opposed to the code level.

-> Insecure Deserialization:

-> The process of serialization is converting objects to byte strings.

-> The process of deserialization is converting byte strings to objects.

-> Using Components with known vulnerabilities:

-> These days, even simple websites such as personal blogs have a lot of dependencies.

-> Insufficient Logging and Monitoring:

-> Monitor your website regularly.

-> While 100% security is not a realistic goal, there are ways to keep your website monitored on a regular basis .

-> An attacker getting a hold of a user's JWT / cookie (Compromised JWT):

-> This is the worst possible attack we can come across because it's the hardest to resolve.

-> Since we're never storing the JWT in LocalStorage and only ever in the cookie, malicious attackers will not be able to steal our user's JWT using XSS.

-> If an attacker somehow manages to steal a user's JWT, then there's unfortunately not much that can really be done.

-> To minimize damages, you should design your application to require reauthentication before performing any high profile transaction such as a purchase or the changing of a password.

-> And your JWTs should also have an expiration date.

-> An attacker getting a copy of or read credentials to our database (Compromised Database):

-> If we have a password, we don't want to store it as plain-text in our database.

-> Instead, we can salt and hash the password and store the salt and hash instead of our plain-text password.

-> A salt is a random string of characters that we append to that password which we then pass through a hashing algorithm.

-> password + salt => hash // salt and hash are stored in user table.

-> Bcrypt:

-> Bcrypt is a password hashing algorithm that's been around since 1999.

-> It standardizes the salting and hashing of passwords.

-> 'Hashing algorithm' + 'Hash cost' + 'Salt value' + '.' + 'Hashed password'.

-> The higher the hash cost, the more computational power it takes to authenticate.

-> Extracting of data from token:

-> Local strategy:

-> It username and password from req.body and verifies the user by verifying it against the User table.

-> JWT strategy:

-> It extracts the JWT from the cookie, and uses the application's secret to verify its signature.

--> Cisco Certification VS Microsoft Certification:

-> Networking => Development

--> Operating system:

-> Manages computer hardware, software resources, and provides common services for computer programs.

--> 32-Bit VS 64-Bit Operating System:

-> Architecture:

-> General computing => The registers are divided into different groups like integer, floating, control.

-> Hardware:

-> Stores 2^32 values => Stores 2^64 values.

-> Can Address up to 4GB of RAM => Can address around 16 exabytes of memory.

-> Software:

-> Compatible with 64-bit => Not vice versa.

-> Calculations per second:

-> Dual-core and quad-core versions available => Dual-core, quad-core, six core, and eight core versions.

--> Agile VS Scrum VS Waterfall:

-> Agile is strategy while scrum is a framework.

-> Definition:

-> Incremental and interactive model development => Fixed length iteration.

-> Man power:

-> Developers and business people must work together => Begins with the product vision.Project owner and teamwork.

-> Planning:

-> Three levels (Release, iteration & Daily Planning) => Sprint (Release planning, Sprint planning, daily scrum, sprint review meeting)

-> Flexibility:

-> More => less (clearly defined rules) => Very less

-> Mode of communication:

-> Face to face within a team => Daily/Weekly based on printed schedule.

-> Ease of change:

-> Focus on customer satisfaction => Suitable for the project.

-> Process flow:

-> Exploration, Planning, Production, Maintenance, Delivery => Pre-Game, Development, Post-Game.

-> Performance Role:

-> Leadership role => No project manager. Has scrim master. He gives report to product owner.

-> Stand up meeting:

-> Sprint planning => Daily. helps a lot

-> Roles:

-> Programmer, Customer, Tester, Tracker => Dev team, Product owner, Scrim master

-> Task & Time:

-> Task is variable and Time is variable => Task is variable and Time is variable => Task is fixed and time is variable

--> Alpha Testing vs Beta Testing:

-> Tested before final release => After release

-> Performed by internal employees => By end users or client

-> Involves both white and black box testing techniques => only black box

-> Not in detail relevant to security etc => measures security etc in depth

-> Identify the bugs => quality testing

-> Lab environment => Real environment

--> Array vs ArrayList:

-> Flexibility:

-> Static data structure => Dynamic

-> Primitives:

-> Can store both primitives and object type of elements => can only store objects of same type

-> Type Safety:

-> Store elements of similar type otherwise throw "ArrayStoreException" => Type safe as it is generic.

-> Iterating the values:

-> Uses "for" or "for each" loop => Uses "ierator()"

-> Adding elements:

-> can insert elemenst by assigment to some index => Use "add()" method.

-> Dimention:

-> Multi => Single.

-> Example:

-> Array:

```
int a[]=new int[2];
a[0]=20; a[1]=30;
for(){}
```

-> ArrayList:

```
ArrayList a=new ArrayList();
a.add("20");
Iterator itr = a.iterator();
while(itr.hasNext()){print(itr.next());}
```

--> ASP.NET vs C#:

-> Design:

-> Framework to bew developed using C# => Programming Language

-> Applications:

-> Develop web applications using any CLS copliance lang such as C#, VB.NET, etc => Language used along with ASP.NET

-> Use:

-> Library od code can be used with C# => It is a CLS(common lang specification) language

-> Designed By:

-> Microsoft => Microsoft but later standardized by ECMA.

-> Operating System:

-> Supports Windows, Linux, MacOS => Supports mosly all major OS.

--> Bootstrap VS Jquery:

-> Definition:

-> Open-source front-end framework => Open-source JS library for client-side scripting

-> Programmed:

-> Mainly written in HTML, CSS, Less, Saas and Javascript => Mainly written in JavaScript

-> Performance:

-> Fast => Slow

-> Applications:

-> Mainly used to develop desktop apps => Mobile apps

-> Websites:

-> Getbootstrap.com => Jquery.com

--> C++ reference VS Pointer:

-> Variables:

-> Variable that represent the alias to an existing variable => Unlike other variable like int, char, double, it stores memory address to make the programming easy.

-> Usage:

-> If we pass a reference paramater to a function, then the function will work on the

original variable despite the copy of that variable.

-> Declaration:

*-> type &newName=existingName; OR type& newName=existingName; OR type & newName=existingName => type *pntr; OR type* pntr; type * pntr;*

-> Reassignment:

-> Cannot => Can be(handy).

-> Example:

*-> int x=5; int y=6; int &r=x;   => int x=5; int y=6; int *p; p=&x; p=&y*

-> Memory Address:

-> Shares the same address => Own memory stores in the stack

-> NULL Value:

-> Cannot have => Can have

-> When to use:

-> Function formal parameters to support pass-by-reference. Changes inside the function are reflected outside the function.

--> Coding vs Programming:

-> Definition:

-> Process of writing codes from one lang to another => Process of creating and developing an executable machine program which executes set of instructions

-> Skills:

-> Initial step of introsucing programming and so coders have less expertise.

-> Advanced Features:

-> Part of programming approach involves translating requirements, writing lines, etc => Bigger pictures involves debugging, compiling, etc.

--> Controller Person VS Comptroller Person:

-> The person who manages all the financial accounts of an organization => highest-level financial officer that takes care of the overall cost and who is specialized in checking ledgers.

-> Private org => Gov org

--> cPanel vs Plesk:

-> Definition:

-> Web-based hosting control panel and Linux based => Web-based hosting control panel for windows-based hosting and servers.

-> Loading:

-> Fastest => Slow

-> Reliability:

-> less time while creating account => more time

--> Docker vs VMs:

-> Docker containers bring along with them numerous amounts of tags where it majorly aims to promote the

cloud portability feature by running the same application in different virtual environments.

-> Docker is a mechanism which is used to isolate the dependencies of each application by packaging them into a container.

-> Containers are safer and scalable to use and deploy comparatively.

-> Boot Time:

-> Can bot in seconds => It takes minutes

-> Execution:

-> Uses execution engine => Uses hypervisor

-> Memory:

-> More Efficient => Less because OS needs to be loaded before starting the service.

-> Ease of deployment:

-> Extremely easy => Comparatively difficult

-> Usage:

-> Complex usage => Easier

--> Programming Languages vs Scripting Languages:

-> Definition:

-> Formal lang that specifies a set of instructions => Supports scripts and it is interpreted rahter than compile.

-> Type:

-> Compiler based => Interpreter based.

-> Usage:

-> Developing anything needful from scratch => To combile existing components.

-> Interpretation:

-> Compiled into a more compact form => Interpreted within another program(like Js is put within HTML) and interpreted by the browsers.

-> Running:

-> Independent => Run inside another program

-> Designed:

-> Designed to give full usage of a lang => Design to make coding fast and simple.

-> Conversion:

-> One shot conversion => Line by line conversion

-> Creation:

-> It creates ".exe" file => does not create.

-> Compilation:

-> Program compilation is necessary => Not necessary

-> Coding:

-> Full code of program => Scripts are piece of code.

-> Complexity:

-> Complex => Easy

-> Host:

-> Does not require a host. It is self executable (.exe) => Mandatoy

-> Example:

-> C, C++, C#, Java, VB, etc => JS, VB Script, Python, ROR, PHP, etc.

--> FTP vs SFTP:

-> File Transfer Protocol => Secure File Transfer Protocol

-> FTP is an internet service that allows you to connect to a particular server or computer.

-> Uploads and Downloads its data without security => Authentication SSH protocol

-> Anyone can access => By only server owner as port 22 ia not open in case of sharing.

-> Usually don't encrypt while sending info to other server.

-> Client-Server architecture is used => SSH aarchitecture is used.

-> Commonly used => Rare

-> Direct method for transfering the files => Tunneling method

-> Port 21 => Port 22

--> Functional Programming vs OOP:

-> Definition:

-> Functions evaluations => Objects evaluations

-> Data:

-> Uses immutable (do not change) data => Mutable

-> Execution:

-> Statements can be exucuted in any order => Particular order

-> Iterations:

-> Recursion is used for it => Loops

-> Usage:

-> When there are few things with more operations => Vice versa

--> HashMap VS TreeMap:

-> Synyax:

-> public class HashMap extends AbstractMap implements Map, Cloneable, Serializable

=> public class TreeMap extends AbstractMap implementsNavigableMap, Cloneable, Serializable

-> Ordering:

-> Does not provide any order for elements => Elements are ordered in a natural or customized order.

-> Speed:

-> Fast => Slow

-> Memory consumptiom:

-> More => Less

-> Functionality:

-> Provides only basic features => Richer features.

-> Performance:

-> O (1) => O(log n)

-> Data structure:

-> Hash table => Red-Black tree.

-> Use Cases:

-> When we do not require key-value pairs in sorted order => Required to be sorted

--> IPv4 vs IPv6:

-> 32 bit IP address => 128 bit

-> Less secure => More

--> JavaScript (Client-Side) VS JQuery:

-> Type:

-> Programming language => It is an API (application programming interface)

-> Language:

-> Written in C => Uses resources given by JavaScript to make things easier

-> Compatibility:

-> Need to make for multiple browsers => It is a multi-browser library

-> Import"

-> In "script" tag within "html" tag. (Both but JQuery needs to be imported the library)

-> Syntax:

-> document.getElementById("demo").val = "hello world" => $("demo").val("hello world")

--> Jira vs Github:

-> Definition:

-> Bug tracking or defect management or project management software => Source code repository or version control system to host the source code of any project

-> Developed:

-> In Java lang => In ROR lang.

--> JSON vs CSV:

-> Definition:

-> Used as storing and exchanging the data => Delimiting text that uses the comma to separate the file

-> Full Form:

-> JavaScript Object Notation => Comma Separated Values.

-> Compact:

-> Less => More

--> Kali Linux VS Ubuntu:

->

--> Linux VS Windows:

-> Access:

-> Has access to the source code of kernel and alter the code according to his need => Every user won't have access to the source code

-> Security:

-> More => Less

-> Size:

-> Small => Large

--> Throw VS Throws:

->

--> TypeScript VS JavaScript:

-> Strongly type object-oriented compile language => Lightweight, interpreted programming language

-> Heavy weighted => Light weighted

-> Client-Side => Both

-> File Extension:

-> ts, tsx => js, jsx

-> A better choice for large Coding Projects

--> WebSocket vs Socket.io:

-> It is the protocol which is established over the TCP connection => It is the library to work with WebSocket

-> It doesn't support broadcasting => It supports

--> Black Box Testing VS White Box Testing:

-> Internals of the software is never exposed.

-> This can be carried out even by a non-technical person => Carried out by a software test engineer and also by software developers.

-> Also be referred to as ‘External Software testing’ and ‘Closed testing’ => ‘Internal Software testing’ and ‘open testing’.

-> Comsimes less time. => More

-> This is not suitable for algorithm testing.

--> Hackers vs Crackers:

-> Constantlt looking for flaws in comp and internal security to rectify and imporove these flaws => Breaks the security of the computer for malicius use.

--> WordPress vs Shopify:

-> Open source content management system based on PHP and MySQL => Shopify is solely an E-Commerce platform

-> Multiple setups => signup Shopify.com

-> Can be used online only => Can be used both online and offline

--> Ionic VS React Native:

-> The user interface is written in different languages. => One language.

-> Single shared library of reusable UI components wrapped by Cordova and PhoneGap => Not

-> More stable

-> Less investment -> Medium

-> Ionic is built on top of Angular. => React Native is built on top of JavaScript.

-> Plugins are not needed thats why it is inexpensive => Plugins have to be downloaded and used.

-> Ionic is slow due to its web apps. => Faster

--> Use Case VS Test Case:

-> Llist of actions or steps that defines interactions between a role (can be a user/external system) and a system to achieve a goal.

=> Group of conditions under which a tester will determine whether the developed system is working as per the design.

-> Apply for business purposes as well as to the developers to give a system overview => Testers

-> Derived from System Requirement Specification => Derived from the use case.

-> The result of the use cases is not verified => Always gets verified with the expected output.

-> Designed by business analysts by collecting the requirement => Designed by either software analyst, QA team, or by the test engineer.

-> Use cases aren’t dependent on test cases => Dependent

--> Stack VS Heap Memory:

-> Has a linear data type structure => Has a Hierarchical Data structure.

-> Has high-speed Access => Slower

-> Space is efficiently managed by OS => Not

-> Operating system dependent stack size limits => No specific memory size limit

-> Memory is assigned in a contiguous block manner => Allocated in a random manner

-> The cost of stack is less => More

-> Stack access time is very faster as compared to heap => Slower

--> Full Stack Developer:

-> Front-end technology

-> Backend technology (development language, database, and cache)

-> API

-> VCS

-> Server

--> Popular stacks:

-> Django stack: JavaScript – Python – Django – MySQL

-> Ruby on Rails: Javascript – Ruby – SQLite – PHP

-> MEAN stack: JavaScript – MongoDB- Express – Angular JS – Node.js

-> LEMP stack: JavaScript – Linux- Nginx – MySQL – PHP

-> LAMP stack: JavaScript – Linux – Apache – MySQL – PHP

-> MEVN stack: JavaScript – MongoDB- Express – Vue – Node.js

-> Serverless Stack: (AWS Lambda was one of the first serverless platforms. Google cloud is another)

--> Webinars:

-> Web seminar.

-> Live online conference or presentation.

--> DevOps:

-> Development + Operations

-> It is a culture to promote the development and operation process collectively.

-> Analyzing the requirements, designing, developing, and testing of the software components or frameworks.

-> Dev = Build, Code, Test, Plan

-> Ops = Monitor, Deploy, Operate, Release

-> Features:

-> Automation

-> Collaboration

-> Intergration

-> Configuration Management

-> Lifecycle:

-> Development => Integration => Testing => Monitoring => Feedback => Deployment => Operations

-> Tools:

-> Git, SVN, Puppet, Docker, Selenium, etc.

-> Azure DevOps:

-> Also known as Microsoft visual studio team services (VSTS).

-> AWS DevOps:

-> AWS is the best cloud service provider, and DevOps is the implementation of the software development lifecycle.

--> Landing Pages:

-> Standalone web pages that are designed to convert visitors into leads.

-> A landing page is specially designed for marketers to get more traffic from a marketing campaign.

-> It can be anything such as a home page of our website, a blog post, a lead capture page, and more.

-> Types:

-> Lead Generation Landing Pages:

->

-> Click-through Landing Pages:

->

-> Viral Landing Page:

--> Block Chain:

-> It can be used for the secure transfer of money, property, contracts, etc. without requiring a third-party intermediary such as bank or government.

-> Blockchain is a software protocol, but it could not be run without the Internet (like SMTP is for email).

-> Bitcoin:

-> Bitcoin is a cryptocurrency(virtual currency), or a digital currency.

-> It is commonly called decentralized digital currency.

-> Version:

-> 1.0 (Currency) => 2.0 (Smart Contracts) => 3.0 (DApps: Decentralized application)

-> Bitcoin Mining:

-> Process of adding transaction records to Bitcoin's public ledger (financial record) of past transactions.

-> Bitcoin mining is used to secure and verify transactions to the rest of the network.

-> Role of Bitcoin Miners:

-> They do this by actually solving math problems and resolving cryptographic issues.

-> These mathematical problems ensure that nobody is tampering with that data.

-> The bitcoin is created by rewarding these minors for their work in solving the mathematical and cryptographical problems.

-> Blockchain Hash Function:

-> A hash function takes an input string (numbers, alphabets, media files) of any length and transforms it into a fixed length.

-> The fixed bit length can vary (like 32-bit or 64-bit or 128-bit or 256-bit) depending on the hash function which is being used.

-> The fixed-length output is called a hash / Cryptographic byproduct of a hash algorithm.

-> Components:

-> Software

-> Cryptography

-> Hardware

-> Miners(Gaming Theory):

-> Bitcoin software creates a challenge. Now, there is a game begins, and there is a race that goes off. The race involves all these miners competing against each other to solve the challenges.

-> This task or challenge will take approximately 10 minutes to be completed.

-> Every single miner starts trying to find the solution to that one Nonce that will satisfy the hash for the block.

-> At some specific point, one of those miners in the global community with higher speed and great hardware specs will solve the cryptography challenge and be the winner of that race.

-> Now, the rest of the community will start verifying that block which is mined by the winner. This makes Bitcoin so strong, because in one stage of this cycle, the miners are competing against each other, and in the next stage of the cycle, the rest of the community rallies together to ensure that that solution is correct.

-> If the Nonce is correct, it will end up with the new block which will be added to the blockchain.

-> For this task or challenge, the winner will earn a reward. That reward is currently 12.5 bitcoins.

--> Program VS Software:

-> Software is more than programs.

-> Any program is a subset of software, and it becomes software only if documentation & operating procedures manuals are prepared.

-> Components of the software:

-> Program, Documentation, Operating Procedures

--> Software Development Life Cycle (SDLC):

-> Also termed process model.

-> Stage 1: Planning and requirement analysis

-> Stage 2: Defining Requirements

-> Stage 3: Designing the Software

-> Stage 4: Developing the project

-> Stage 5: Testing

-> Stage 6: Deployment

-> Stage 7: Maintenance

-- SDLC Models:

-> Waterfall:

-> The waterfall is a universally accepted SDLC model.

-> RAD:

-> Rapid Application Development

-> It targets developing software in a short period.

-> Spiral:

-> A risk-driven process model.

-> V-Model:

-> Also referred to as the Verification and Validation Model.

-> In this, each phase of SDLC must complete before the next phase starts.

-> It follows a sequential design process same as the waterfall model.

-> Testing of the device is planned in parallel with a corresponding stage of development.

-> Cycle:

-> Bussiness req => System req => High lvl design -> Low lvl design -> Coding -> unit testing -> component testing -> system integration testing -> Acceptance testing

-> Incremental:

->

->

-> Agile:

-> Cycles:

-> Requirements gathering -> Design the requirements -> Construction/ iteration -> Testing/ Quality assurance -> Deployment -> Feedback

-> Iterative:

-> Cycle:

-> Requirement gathering & analysis -> Design -> Implementation -> Testing -> Deployment -> Review -> Maintenance

-> Big Bang:

-> This model is required when this project is small like an academic project or a practical project..

-> This method is also used when the size of the developer team is small and when requirements are not defined,

-> Prototype:

-> Cycle:

-> Requirement Gathering and Analyst -> Quick Decision -> Build a Prototype -> Assessment or User Evaluation -> Prototype Refinement -> Engineer Product

--> Software Metrics:

-> Measure of software characteristics which are measurable or countable.

--> Risk Management:

-> Project risks

-> Technical risks

-> Business risks

--> Nginx:

-> Nginx is an open source, fast, lightweight and high-performance web server that can be used to serve static files.

-> NGINX is considered as the popular web server behind Apache web server and Microsoft's IIS.

-> NGINX is pronounced as "engine-ex"

-> In short, we can say that Nginx is just a kind of software that is used in web servers to serve concurrent requests.

--> Apache VS NGINX:

-> Apache is "Apache HTTP Server".

-> Apache remains the first choice among the server administrators because of its flexibility, architectural simplicity, power compatibility, and multi-platform support.

-> It can run on almost all operating systems such as Windows, UNIX, OSX, NetWare, etc. But it is commonly used in combination with Linux.

-> Apache became the backbone of the WWW (World Wide Web)

-> Apache was really on the top of the game, but when NGINX comes in the business, the server administrator's choice was changed.

-> Apache is designed to be a web server. => Nginx is both a web server and a proxy server.

-> Apache uses a multi-threaded approach to process client requests. => event-driven approach

-> A single thread can only process one connection. => A single thread can handle multiple connections.

--> URI vs URL:

->

--> Proxy servers:

->

--> RESTful services:

-> Problem with the SOAP was that with each request, Metadata is attached with data to be transferred.

-> Web API may or may not be RESTful services, but they are always HTTP based services.

-> REST stands for Representational State Transfer.

-> Principles of REST API:

-> Stateless:

-> When the request from the client is sent to the server, it contains all the required information to make the server process it.

-> A request may be part of QueryString or URL.

-> Client-Server:

-> Separating the functionality helps to increase user interface portability across multiple platforms as well as extended the scalability of the server components.

-> Uniform Interface:

-> To obtain the uniformity throughout the application, REST has defined four interface constraints for which are:

-> Resource Identification

-> Resource Manipulation using representations

-> Self-descriptive massages

-> Hypermedia as the engine of the web application

-> Cacheable:

-> In order to provide a better performance, applications are made cacheable.

-> Layered System:

-> The layered system allows an application to be most stable by limiting component behavior.

-> Also helps to enhance security

-> Code on demand

--> WCF:

-> Windows Communication Foundation.

-> Framework for building service-oriented applications.

-> Using WCF, you can send data as asynchronous messages from one service endpoint to another.

--> Web API Security:

-> Authentication

-> Authorization

--> Hashing vs Encryption vs Encoding:

->

Reference Links

- *https://www.howtographql.com/basics/1-graphql-is-the-better-rest/ (GraphQL VS Rest)*
- *https://expressjs.com/en/resources/middleware/cors.html (CORS)*
- *https://readwrite.com/2008/07/22/top_10_concepts_that_every_software_engineer_should_know/*
- *https://www.bigcommerce.com/ecommerce-answers/what-is-cloud-computing/ (Cloud Computing)*
- *https://www.javatpoint.com/*
- *https://www.upwork.com/articles/development*
- *https://www.educba.com/javascript-vs-node-dot-js/ (BEST link for all programming languages)*
- *https://www.javatpoint.com/nginx-tutorial (Nginx)*
- *https://www.javatpoint.com/web-api#RESTful*