|  |
| --- |
|  |
| Understanding Technologies Used |
|  |

|  |
| --- |
| Jincy P Janardhanan |

CONTENTS

[Backend Technologies 2](file:///E:\Mini%20Project\Career-Information-and-Recruitment-Portal\References\Understanding%20Technologies%20Used.docx#_Toc45823619)

[1 Java 3](#_Toc45823620)

[2 Frameworks 5](#_Toc45823621)

[2.1 Spring Framework for Java 5](#_Toc45823622)

[2.1.1 Spring Web MVC Framework 6](#_Toc45823623)

[2.1.2 Spring Boot Framework 7](#_Toc45823624)

[2.1.2.1 Embedded Tomcat 7](#_Toc45823625)

[2.1.3 Spring Security 7](#_Toc45823626)

[3 Database Management Systems (DBMS) 8](#_Toc45823627)

[3.1 MongoDB 11](#_Toc45823628)

[Front-end Technologies 12](file:///E:\Mini%20Project\Career-Information-and-Recruitment-Portal\References\Understanding%20Technologies%20Used.docx#_Toc45823629)

[4 Components of a Webpage 13](#_Toc45823630)

[4.1 Introduction 13](#_Toc45823631)

[4.1.1 User Interface 13](#_Toc45823632)

[4.1.1.1 Markup Languages 13](#_Toc45823633)

[4.1.2 User Experience 14](#_Toc45823634)

[4.2 HTML5 14](#_Toc45823635)

[4.3 CSS 14](#_Toc45823636)

[4.4 JavaScript 14](#_Toc45823637)

[4.4.1 Frameworks for Javascript 15](#_Toc45823638)

[4.4.1.1 Angular 8 15](#_Toc45823639)

[5 REST and RESTful APIs 17](#_Toc45823640)

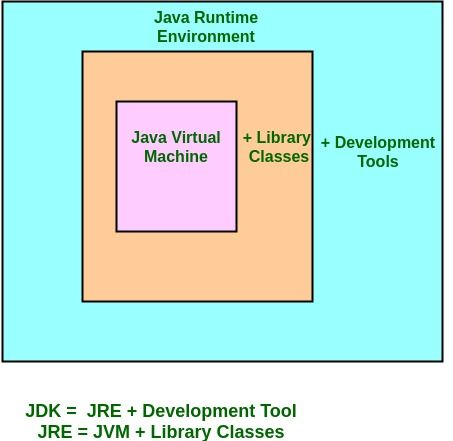
[6 Thymeleaf 18](#_Toc45823641)

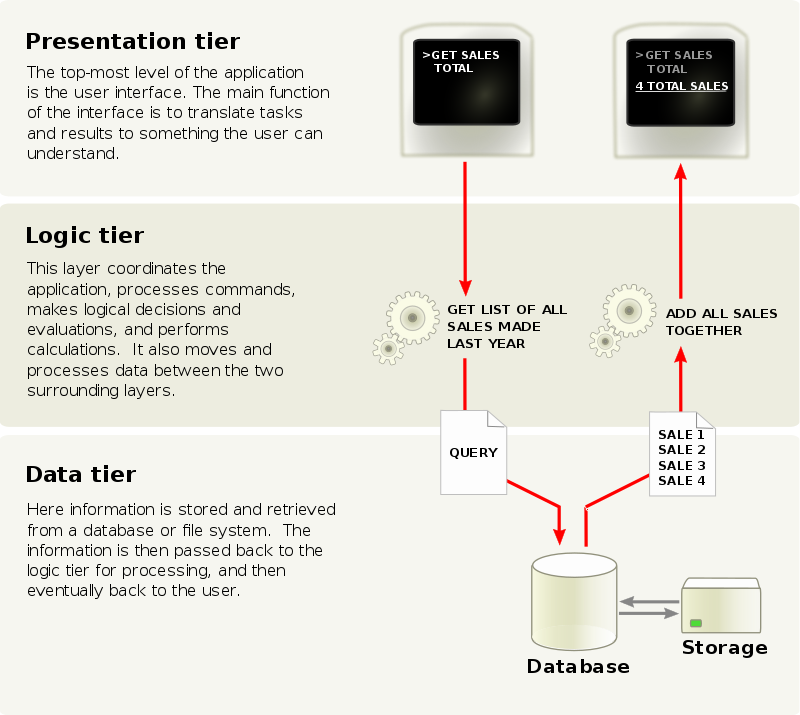
Backend Technologies

Backend of a web application consists of a server, an application, and a database.

In the back-end, we build and maintain the technology that powers those components which, together enables the front-end of an application.

# Java

1. What is Java?
   * Java is a fast, powerful, secure, reliable and platform-independent, free, open-source, object-oriented programming language and computing platform.
   * First released by Sun Microsystems in 1995.
   * Owned by Oracle.
2. What are the different editions or programming language platforms for Java?
   * There are four different editions or Java programming platforms.
     + **Java Platform, Standard Edition (Java SE)**
       - Java SE provides all the core functionalities of Java language.
       - It defines everything from the basic types and objects of the Java programming language to high-level classes that are used for networking, security, database access, graphical user interface (GUI) development, and XML parsing.
     + **Java Platform, Enterprise Edition (Java EE)**
       - Java EE is built on top of Java SE platform, intended for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications.
     + **Java Platform, Micro Edition (Java ME)**
       - Java ME is used for developing and running Java applications on small devices, like mobile phones.
       - It is a subset of Java SE which includes special class libraries useful for small device application development.
       - Java ME applications are often clients of Java EE platform services.
     + **JavaFX**
       - JavaFX is a platform for creating rich internet applications using a lightweight user-interface API.
       - JavaFX applications use hardware-accelerated graphics and media engines to take advantage of higher-performance clients and a modern look-and-feel as well as high-level APIs for connecting to networked data sources.
       - JavaFX applications may be clients of Java EE platform services.
3. What is a JDK, JRE, AND JVM?
   * **Java Development Kit (JDK)** 
     + The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets.
     + It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a *compiler (javac), an archiver (jar), a documentation generator (Javadoc)* and other tools needed in Java development.
   * **Java Runtime Environment (JRE)**
     + The Java Runtime Environment (JRE) provides the minimum requirements for executing a Java application; it consists of the Java Virtual Machine (JVM), core classes, and supporting files.
   * **Java Virtual Machine (JVM)**
     + Java Runtime Environment (JRE) is an installation package which provides environment to only run and not develop a java program (or application) on a machine.
4. What is a multi-tiered application, or what is multi-tiered architecture?
   * Any application developed and distributed among more than one layer is a multi-tiered application.
   * Also, any application that depends on or uses a middleware application is known as a multi-tier application.
   * Multi-tier applications logically separate the different application-specific and operational layers.
   * There can be ‘n’ number of layers or tiers.
   * **Three-tier** is the most commonly used architecture for multi-tiered applications. The three-tiered architecture is explained in the figure below.



# Frameworks

1. What is a framework for a programming language?
   * A framework is a collection of libraries and tools, which implements inversion of control, to support writing a particular class of applications. It is actual software written in a programming language.
   * A framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software.
   * Frameworks are standard ways to build and deploy applications.
2. What is Inversion of Control (IoC)?
   * Inversion of Control is a design principle used to invert different kinds of controls in object-oriented design (additional responsibilities other than the main objective of the class), so as to achieve loose coupling.
   * This include control over the flow of an application, and control over the flow of an object creation or dependent object creation and binding.

## Spring Framework for Java

1. What is Spring Framework for Java?
   * The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.
   * It enables you to build applications from “plain old Java objects” (POJOs) and to apply enterprise services non-invasively to POJOs.
   * It can be applied to Java SE, and full and partial Java EE.
2. What are the features or capabilities of Spring?
   * **Microservices**
     + Microservices are a form of service-oriented architecture style wherein applications are built as a collection of different smaller services rather than one whole app.
     + The small scale and relative isolation of microservices can lead to many additional benefits, such as more flexibility, resilience, easier maintenance, improved productivity, greater fault tolerance, better business alignment, etc.
   * **Reactive**
     + Reactive processing is a paradigm that enables developers build non-blocking, asynchronous applications that can handle back-pressure (flow control).
     + Reactive systems better utilize modern processors.
     + The inclusion of back-pressure in reactive programming ensures better resilience between decoupled components.
     + Ideal for low-latency, high-throughput workloads.
     + *Project Reactor* and the *Spring portfolio* work together to enable developers to build enterprise-grade reactive systems that are responsive, resilient, elastic, and message-driven.
   * **Cloud**
     + For developing distributed systems.
     + In cloud-native applications, complexity is moved from the application layer to the network layer and demands greater interaction between services. It deals with [12-factor](https://12factor.net/) issues such as external configuration, statelessness, logging, and connecting to backing services.
   * **Web Applications**
     + Spring removes much of the boilerplate code and configuration associated with web development, so that the developers get a modern web programming model that streamlines the development of server-side HTML applications, REST APIs, and bidirectional, event-based systems.
   * **Serverless**
     + Serverless applications take advantage of modern cloud computing capabilities and abstractions to let you focus on logic rather than on infrastructure.
     + For serverless applications, developers need to write only the business logic that is invoked by the platform, thus allowing for a more resilient requirement evolution as business needs change.
     + The Spring portfolio provides a robust collection of functionalities for use within serverless applications.
   * **Event-driven**
     + Event-driven programming is a programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs or threads.
     + It also supports streaming data, i.e. a constant flow of events.
   * **Batch**
     + Batch processing is the processing of a finite amount of data in a manner that does not require external interaction or interruption.
     + Batch processes are an extremely efficient way of processing large amounts of data.
     + Spring Batch’s implementation of industry-standard processing patterns lets you build robust batch jobs on the JVM.

### Spring Web MVC Framework

1. What is Spring Web MVC?
   * Spring Web MVC is a Java framework which is used to build web applications. It follows the Model-View-Controller design pattern.



* + - **Model** - A model contains the data of the application.
    - **Controller** - A controller contains the business logic of an application.
    - **View** - A view represents the provided information in a particular format.
    - **Front Controller** - In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.
  + Spring Boot web applications are built on top of Spring MVC.

### Spring Boot Framework

1. What is Spring Boot?
   * Spring Boot is an open source Java-based framework used to create microservices.
   * It is used to build stand-alone Spring applications.
   * It provides production-ready features such as metrics, health checks, and externalized configuration.
   * It provides opinionated 'starter' dependencies to simplify your build configuration.
   * It automatically configures Spring and 3rd party libraries whenever possible.
   * There is absolutely no code generation and no requirement for XML configuration.
   * With Spring boot, we can embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files).

#### Embedded Tomcat

The default web server for Spring Boot applications is embedded Tomcat.

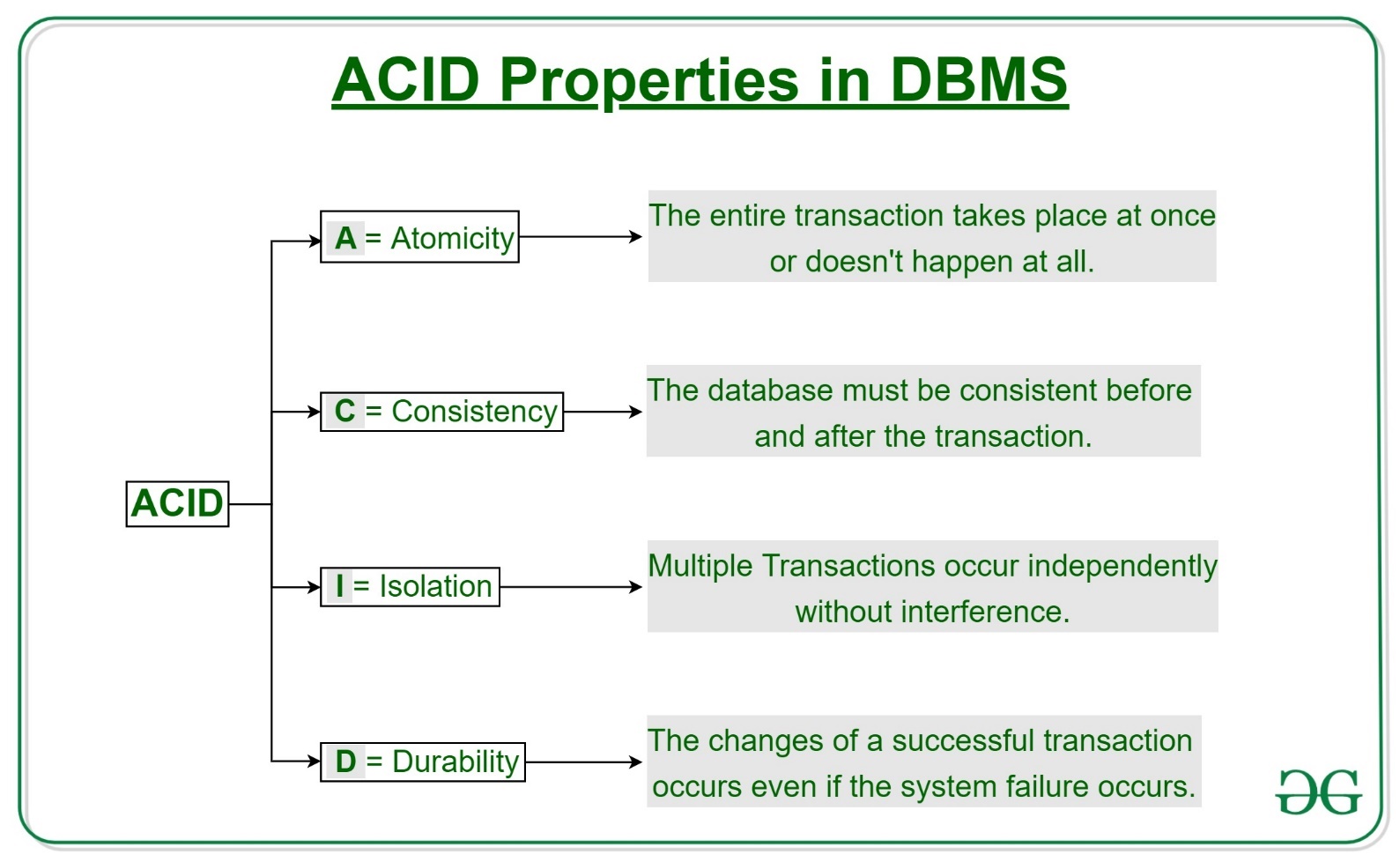
1. What is Tomcat (or Apache Tomcat)?
   * Apache Tomcat is an open-source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and WebSocket technologies.
   * It provides a "pure Java" HTTP web server environment in which Java code can run.
2. What is meant by embedded Tomcat?
   * Embedded Tomcat means packaging a full Tomcat server distribution as a compressed JAR, WAR or ZIP file into a single Java web application.

### Spring Security

1. What is Spring Security?
   * Spring Security is a powerful and highly customizable authentication and access-control framework for Java/Java EE.
   * It focuses on providing both authentication and authorization to Java applications.
   * It is the de-facto standard for securing Spring-based applications.
2. What are the major features of Spring Security?
   * Comprehensive and extensible support for both Authentication and Authorization.
   * Protection against attacks like session fixation, clickjacking, cross site request forgery, etc.
   * Servlet API integration.
   * Optional integration with Spring Web MVC.

# Database Management Systems (DBMS)

1. What is a database?
   * A database is an organized collection of structured information, or data, typically stored electronically in a computer system.
   * A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just ‘database’.
2. What is a DBMS?
   * A database management system (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database.
   * A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure.
   * It also defines rules to validate and manipulate this data.
   * A DBMS relieves users of framing programs for data maintenance.
3. What are ACID properties of a database?



1. What are the 4 basic types of databases (or database models)?
   * **Relational Database**
     + A relational database is a type of database that stores and provides access to data points that are related to one another.
     + In a relational database, each row in the table is a record with a unique ID called the key.
     + The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.
     + Relational databases are based on the **relational model** - an intuitive, straightforward way of representing data in tables.
     + E.g. for RDBMS - MySQL, PostgreSQL, MariaDB
   * **Object Oriented Database**
     + Object Oriented Database is a type of database in which data is represented in the form of objects and classes.
     + They follow the fundamental principles of object-oriented programming (OOP).
     + E.g. for OODBMS - InterSystems Caché, Versant Object Database, ODABA
   * **Hierarchical Database**
     + In a hierarchical database, data are organized into a tree-like structure.
     + Data are stored as records which are connected to one another through links.
     + E.g. for Hierarchical DBMS - IBM Information Management System (IMS), Windows registry
   * **Network Database**
     + In a network database, multiple member records or files can be linked to multiple owner files and vice versa.
     + E.g. for Network DBMS - Integrated Database Management System (IDMS)
2. What are multi-model databases?
   * Any combination of different database models, is a multi-model database.
   * E.g. for Multi-model Database - MarkLogic, ArangoDB, Azure Cosmos DB, FoundationDB
3. SQL vs. NoSQL databases

|  |  |  |
| --- | --- | --- |
|  | SQL Databases | NoSQL Databases |
| Data Storage Model | Tables with fixed rows and columns | **Document:** JSON documents  **Key-value:** key-value pairs  **Wide-column:** tables with rows and dynamic columns **Graph:** nodes and edges |
| Development History | Developed in the 1970s with a focus on reducing data duplication | Developed in the late 2000s with a focus on scaling and allowing for rapid application change driven by agile and DevOps practices. |
| Examples | Oracle, MySQL, Microsoft SQL Server, and PostgreSQL | **Document:** MongoDB and CouchDB  **Key-value:** Redis and DynamoDB  **Wide-column:** Cassandra and HBase  **Graph:** Neo4j and Amazon Neptune |
| Primary Purpose | General purpose | **Document:** general purpose  **Key-value:** large amounts of data with simple lookup queries  **Wide-column:** large amounts of data with predictable query patterns  **Graph:** analysing and traversing relationships between connected data |
| Schemas | Rigid | Flexible |
| Scaling | Vertical (scale-up with a larger server) | Horizontal (scale-out across commodity servers) |
| Multi-Record ACID Transactions | Supported | Most do not support multi-record ACID transactions. However, some—like MongoDB—do. |
| Joins | Typically required | Typically, not required |
| Data to Object Mapping | Requires ORM (object-relational mapping) | Many do not require ORMs. MongoDB documents map directly to data structures in most popular programming languages. |

1. Extended set of databases (can be multi-model databases)
   * **Distributed Database**
     + A distributed database is a database that consists of two or more files located in different sites either on the same network or on entirely different networks.
     + Portions of the database are stored in multiple physical locations and processing is distributed among multiple database nodes.
     + E.g. for Distributed DBMS - Apache Cassandra, Apache CouchDB, Google Cloud Bigtable
   * **Data Warehouses**
     + A data warehouse is a type of database specifically designed for fast query and analysis, mostly on historical data derived from transactional sources and is used in data analytics to empower data-driven decisions for business intelligence and also for data mining purposes.
     + It is a central repository of all data in an organization.
     + E.g. for Data Warehouses - Teradata, Oracle Autonomous Data Warehouse, Amazon Web Services (AWS) Data Warehouse
   * **Graph Database**
     + A graph database stores data in terms of entities and the relationships between entities.
     + E.g. for Graph Database - Neo4j, Giraph, JanusGraph
   * **Document/JSON Database**
     + Designed for storing, retrieving, and managing document-oriented information.
     + Document databases are a modern way to store data in JSON format rather than rows and columns.
     + E.g. for Document/JSON Database - MongoDB, ArangoDB, MarkLogic
   * **Cloud Databases**
     + A cloud database is a collection of data, either structured or unstructured, that resides on a private, public, or hybrid cloud computing platform.
     + There are two types of cloud database models: traditional and database as a service (DBaaS). With DBaaS, administrative tasks and maintenance are performed by a service provider.
     + E.g. for Cloud Databases - Google Cloud Bigtable, Amazon DynamoDB, Azure CosmosDB
   * **Self-driving Databases**
     + Self-driving databases (also known as autonomous databases) are cloud-based and use machine learning to automate database tuning, security, backups, updates, and other routine management tasks traditionally performed by database administrators.
     + Oracle Autonomous Data Warehouse, Peloton
2. OLAP vs. OLTP
   * **On-Line Analytical Processing (OLAP)**
     + It is used for analysis of database information from multiple database systems at one time such as sales analysis and forecasting, market research, budgeting and etc.
     + OLAP databases are not normalized, and it manages all insert, update and delete transactions.
     + E.g. Data warehouse.
   * **On-Line Transactional processing (OLTP)**
     + It is used for maintaining the online transaction and record integrity in multiple access environments.
     + It manages very large number of short online transactions.
     + OLTP database tables are normalized and are mainly used for data reading.
     + E.g. ATM systems

## MongoDB

1. What is MongoDB?
   * MongoDB is a cross-platform, open-source, distributed, document-oriented (NoSQL) database program.
   * It is highly scalable and flexible.
2. What are the features of MongoDB?
   * **Ad Hoc Queries** - You can search by field, range query and it also supports regular expression searches.
   * **Indexing** - You can index any field in a document.
   * **Replication** 
     + Replication is the process creating a replica of the data. It need not be an exact copy of the data.
     + It is used for synchronizing data between a server and its client.
     + MongoDB supports **master-slave replication**: A master can perform reads and writes and a slave copies data (replication) from the master and it can only be used for reads or back up (not writes).
   * **Duplication**
     + Duplication is the process of creating an exact copy of the data.
     + MongoDB can run on multiple servers and data is duplicated across all servers.
     + It helps for synchronizing data among multiple servers.
     + Data redundancy is achieved via duplication, i.e. the system up will not fail in case of hardware failure.
   * **Load balancing**
     + MongoDB has an automatic load balancing configuration because of data placed in shards (horizontal partitions of a database stored in different database servers).
   * Supports **map reduce and aggregation tools**.
   * Uses **JavaScript instead of procedures**.
   * It is a **schema-less database** written in C++.
   * Provides **high performance** **and scalability**.

Front-end Technologies

Backend of a web application consists of the user interface of the application.

# Components of a Webpage

A webpage is mainly composed of three elements: HTML, CSS and JavaScript.

## Introduction

### User Interface

1. What is UI?
   * A User Interface (UI) is a way of communicating and interacting with a user.
2. What are the different types of UI?
   * The different types of User Interfaces (UI) are:
     + **Command Line Interface (CLI)** - It is a text-based user interface in which the user interacts with the computer via commands from a predefined set of commands.
     + **Graphical User Interface (GUI)** - It is a form of user interface in which the user can interact with the computer via graphical icons, images, buttons, etc.
       - **Menu Driven Interface** - It is a type of GUI in which the user navigates through a series of screens by choosing options from lists, i.e. menus. E.g. interfaces in ATMs, washing machines, etc.
       - **Form Based Interface** - It is a type of GUI which uses text boxes, drop-down menus, checkboxes, radio buttons, etc. to collect information from (interact) with the user.
     + **Natural Language Interface (NLI)** - It is a type of UI in which the user communicates to the computer by talking to it. Also known as *conversational interface.*
     + **Gesture Based Interface** - It is a type of UI in which user interacts to the computer via gestures like tap, scroll, pinch, pan, etc. E.g. UI for smartphones.
     + **Gesture Driven Interface** - It is a type of UI in which the user can communicate to the computer via gestures like hand movements, facial expressions, eye movements, etc.

#### Markup Languages

1. What are markup languages?
   * Mark-up languages are ways of annotating an electronic document. They are used to specify how something should be displayed or what something means.
2. What are the most popular of markup languages?
   * **Hypertext Markup Language (HTML)** - It is the standard markup language for documents displayed in a web browser.
   * **eXtensible Markup Language (XML)** - It is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
   * **eXtensible Hypertext Markup Language (XHTML)** - It is a stricter, more XML-based version of HTML.
   * **Keyhole Markup Language (KML)** - It is an XML notation for expressing geographic annotation and visualization within two-dimensional maps and three-dimensional Earth browsers.
   * **Mathematical Markup Language (MathML)** - It is an application of XML to describe mathematical notations and capture its structure and content. It is used for integrating mathematical formulae into World Wide Web pages and other documents.
   * **Scalable Vector Graphics (SVG)** - It is an XML-based markup language for describing two-dimensional based vector graphics.
   * **Standard Generalized Markup Language (SGML)** - It is a standard for specifying a document markup language or tag set.

### User Experience

1. What is meant by UX?
   * User Experience (UX) is what a user of a particular product experiences when using that product.
   * UX is determined by how easy or difficult it is to interact with the user interface elements.
   * UX designers are in charge of determining how the user interface operates.

## HTML5

1. What is HTML5?
   * HTML5 is the latest specification (or standard) of the HTML language.
2. What are the features of HTML5?
   * It encourages semantic (meaningful) markup.
   * Separating design from content.
   * It promotes accessibility and design responsiveness.
   * It reduces the overlap between HTML, CSS, and JavaScript.
   * It supporting rich media experiences while eliminating the need for plugins such as Flash or Java.

## CSS

1. What is CSS?
   * Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML).
   * CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

## JavaScript

1. What is JS?
   * JavaScript (JS) is a programming language used most often for dynamic client-side scripts on webpages, but it is also often used on the server-side, using a runtime such as Node.js.

*(Node.js Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside a web browser)*

* + JavaScript is mostly used in the browser, enabling developers to manipulate webpage content through the DOM, manipulate data with AJAX and IndexedDB, draw graphics with canvas, interact with the device running the browser through various APIs, and more.

1. What is an API?
   * An **Application Programming Interface (API)** is a set of features and rules that exist inside a software program (the application) enabling interaction with it through software - as opposed to a human user interface.
   * An API specifies how different software components should interact.

*(It can be thought of as an interface between different software.)*

1. What is DOM?
   * **Document Object Model (DOM)** is an API that represents and interacts with any HTML or XML document.
   * The DOM is a document model loaded in the browser and representing the document as a node tree, where each node represents part of the document (e.g. an element, text string, or comment).
   * Event listeners can be added to nodes and triggered on occurrence of a given event.
2. What is AJAX?
   * **Asynchronous JavaScript And XML (AJAX)**, is a programming practice of building complex, dynamic webpages using a technology known as XMLHttpRequest.
   * **XMLHttpRequest (XHR)** is a JavaScript API to create AJAX requests. Its methods provide the ability to send network requests between the browser and a server.
   * Ajax allows you to update parts of the DOM of an HTML page instead without the need for a full-page refresh.
   * Ajax also lets you work asynchronously, meaning your code continues to run while the targeted part of your web page is trying to reload.
   * With interactive websites and modern web standards, Ajax is gradually being replaced by functions within JavaScript frameworks and the official **Fetch API** Standard.
3. What is Fetch API?
   * The **Fetch API** is a simple interface for fetching resources.
   * Fetch makes it easier to make web requests and handle responses than with the older XMLHttpRequest, which often requires additional logic (for example, for handling redirects).
   * Fetch supports the **Cross Origin Resource Sharing (CORS)**, i.e., a web application running at one origin can access selected resources from another origin.
4. What is IndexedDB?
   * **IndexedDB** is a Web API for storing large data structures within browsers and indexing them for high-performance searching.
   * Like an SQL-based RDBMS, IndexedDB is a transactional database system. However, it uses JavaScript objects rather than fixed columns tables to store data.
5. What is Canvas?
   * The **Canvas** element is part of HTML5 and allows for dynamic, scriptable rendering of 2D and 3D shapes and bitmap images.
   * It provides an empty graphic zone on which specific JavaScript APIs (e.g. Canvas 2D or WebGL) can draw.

### Frameworks for Javascript

#### Angular 8

1. What is Angular 8?
   * **Angular 8** is an open-source, client-side TypeScript based JavaScript framework. It is written in TypeScript and complied into JavaScript.
   * Angular 8 is a platform used to create dynamic web applications. It is very similar to its previous versions of Angular except from having some extensive features.
2. What is TypeScript?
   * **Typescript** is a strict syntactical **superset of JavaScript** and adds optional static typing to the language.
   * TypeScript is designed for development of large applications.
3. Which is the latest version of Angular?
   * **Angular 9**, released on Feb 6, 2020.
4. **Securing web applications with Spring security, MongoDB RESTful Authentication and Angular 8**
   * The Angular 8 used as the frontend is using the `HttpClient`, `HttpInterceptor`, and `RouteGuard` modules.
   * The secure page is guarded by Angular 8 Route Guard and the secure RESTful API is guarded by Spring Security REST.

# REST and RESTful APIs

1. What is REST?
   * **REpresentational State Transfer (REST)**, is an architectural style for distributed hypermedia systems, providing standards between computer systems on the web, making it easier for systems to communicate with each other.

*(“Hypermedia” refers to any content that contains links to other forms of media such as images, movies, and text.)*

* + The REST architecture defines a set of constraints to be used for creating web services.

*(A web service is a unit of managed code that can be remotely invoked using HTTP. That is, it can be activated using HTTP requests.)*

* + **REST-compliant systems or RESTful systems** (systems using REST architecture), are characterized by how they are stateless *(server does not need to know anything about what state the client is in and vice versa)* and separate the concerns of client and server.

1. What is a REST API?
   * A **RESTful API (or REST API)** is an application program interface (API) that uses HTTP requests to retrieve or manipulate data from a database.
   * A REST API service usually comes as a medium/interface for the **frontend to consume data served from backend.** Sometimes, even the backend uses REST API service to get data from another backend service.
2. What are HTTP requests (methods) that can be made from a REST API?
   * **HTTP GET** - To retrieve resource representation/information only.
   * **HTTP POST** - To create new subordinate resources.
   * **HTTP PUT** - To update existing resource.
   * **HTTP DELETE** - To delete resources.
   * **HTTP PATCH** - To make partial update on a resource.

# Thymeleaf

1. What is Thymeleaf?
   * Thymeleaf is a modern server-side Java XML/XHTML/HTML5 template engine that can work both in web and non-web environments.

*(A template engine is a software designed to combine templates with a data model to produce result documents.)*

* + Thymeleaf is used to **render the frontend of an application.**
  + It is better suited for serving XHTML/HTML5 at the view layer of MVC-based web applications, but it can process any XML file even in offline environments
  + Thymeleaf's brings elegant natural templates to your development workflow — HTML that can be correctly displayed in browsers and also work as static prototypes, allowing for stronger collaboration in development teams.
  + Thymeleaf is ideal for modern-day HTML5 JVM web development.
  + HTML templates written in Thymeleaf still look and work like HTML, letting the actual templates that are run in your application keep working as useful design artifacts.