**References**

naveen.k.puvvada@centene.com--

[Prakyathi.Kuntumadugu@centene.com](mailto:Prakyathi.Kuntumadugu@centene.com)

Employer details:

Mohan.k

Devoir Software solutions

(O): +1 8722169394

Address: 700 Crown Industrial CT, Suite #E, Chesterfield, MO-63005

email: hr@devoiritsys.com

Phone:+1 8722169394

**Final Script**

I am a Java Full Stack Developer with over 7 years of experience in designing and developing web-based applications. I have expertise in Java/J2EE technologies, JavaScript frameworks, and UI development using HTML, CSS, and JavaScript. My skills include Spring framework, Hibernate ORM, RESTful web services, and database management with both relational and NoSQL databases. I am skilled in utilizing Angular features such as components, directives, and services to create interactive user interfaces. With my knowledge of Angular, I can effectively enhance the front-end functionality and user experience of web applications. I have experience working with cloud platforms like AWS and Google Cloud, as well as containerization technologies like Docker and Kubernetes. I am proficient in Agile methodologies and have a strong understanding of software development life cycle. I am passionate about creating efficient and scalable solutions and continuously learning new technologies to stay up to date. In my experience I got a chance to work with healthcare, Financial and Banking domains.

**Tech Stack:**

* Front end : React, Angular 8, HTML 5, CSS 3, jQuery, JavaScript, Bootstrap, Typescript.
* Backend : JAVA 8,11, core java, Spring MVC, Spring Boot, Node JS, Express JS
* Databases: SQL, Oracle, MySQL, Microsoft SQL Server, and PostgreSQL, PL/SQL

No SQL: MongoDB, HBase, Cassandra.

* Web/ Application Servers: Oracle Web Logic, Apache Camel (framework for MOM)
* Software Methodologies: Agile, Scrum, SDLC Waterfall
* Cloud- AWS, PCF, Azure
* Web Services: RESTful, SOAP
* Microservices - Spring Boot, Node JS
* Testing -Junit, Mockito, Jasmine
* Build Tools- Maven, Gradle, Apache Ant
* Bug Tracking tools: Jira, Bugzilla
* CI/CD Pipeline – Jenkins, Bamboo
* Messaging: JMS, IBMs MQ, Kafka, Anypoint MQ
* Version control – GIT, Concurrent Versions System (CVS), Apache Subversion (SVN)
* Deployment – AWS, Docker, Kubernetes
* Operating System: Windows, Linux, Ubuntu, Mac OS
* IDES- Visual studio, Eclipse, IntelliJ, Putty, IBM, Anypoint Studio, Net Beans

**What does Typical Java developer do in a day?**

* Checking emails and communicating with team members to discuss project progress, updates, and issues.
* Writing code to implement new features, fix bugs, and maintain existing code.
* Participating in daily stand-up meetings to provide updates on progress, discuss any issues, and plan the work for the day.
* Collaborating with designers to implement user interfaces using HTML, CSS, and JavaScript frameworks like React or Angular.
* Developing and maintaining RESTful APIs to integrate the frontend and backend parts of the application.
* Writing unit tests to ensure the quality and correctness of the code.
* Participating in code reviews to review and provide feedback on code written by other team members.
* Debugging and troubleshooting issues that arise in the application.
* Staying up to date with new and emerging technologies and frameworks relevant to the full stack development.
* Participating in agile development practices, such as sprint planning and retrospectives, to ensure timely delivery of high-quality software.
* Overall, a Full Stack Java Developer spends their day working on various tasks to ensure the timely delivery of high-quality software that meets the requirements and specifications of the stakeholders. This may involve working with a team of designers, developers, and project managers to implement new features, fix bugs, and maintain existing code.

Participate in daily/weekly stand ups, code reviews, and QA walkthroughs.

Adhere to platform architecture and best practices.

Work with scrum team in developing the sprint plan.

Complete assigned tasks within the sprint timeline and with high quality

Ability to provide 24x7 on-call support on a rotation assignment.

**Current Experience:**

**Client: Centene(Jan 2022- Present**)

* I am currently working as a Full Stack Java Developer for the client Centene for 1.5 years now.

Coming to my **Technology Stack** :

* I used the Spring Boot for building Microservices and Spring-MVC concepts for building REST APIs. I also used some of the Core Java concepts such as Object-Oriented Programming **OOP Concepts, Collections Framework, Exception Handling, Multi-Threading.**
* I played a significant role in designing and implementing RESTful Web Services using Mule ESB.
* For the front end, I used Angular JS to create the user views and Angular directives and scope of Angular to generate the views and validate the information.
* Implemented Angular Authentication using the HTTP interceptors and HTTP Client.
* I worked on integrating with Oracle and Cassandra databases, designing schemas, and implementing database operations using PL/SQL packages, triggers, and sequences.
* Besides that, I have deployed applications in higher environments in AWS, and I used storage services like EC2(Amazon Elastic Compute Cloud), Amazon Simple Storage Service (S3) buckets and EBS(Amazon Elastic Block Store), and deployed applications on cloud platforms like IASS, SASS, PASS (infrastructure as a service, software, and platform).
* I used Spring-boot and Spring-MVC concept for rest APIs and building the microservices.
* Implemented the automatic documentation for the API using the Swagger and OpenAPI specifications.
* We use GIT to keep a track of the state of the code and JIRA for project, text cases and bug management.
* Coming to the life cycle of the development of the project, we have a product owner that gathers the requirements for the project and we as a team sit together to analyse the requirements and create user stories and assign JIRA tickets. Once the JIRA tickets have been created and assigned to the team, we start working on the implementation. We perform unit testing from end- to end by writing test cases to test the code locally and pass on the code to the QA team.
* In my current experience, my Team follows the Agile Environment where we have daily Stand-Up calls and bi -weekly Sprint Plannings followed by retrospectives. And I am working in a team of 10 individuals with 3 developers on the backend, 2 on the frontend, 2 on devops with 1 tester, 1 scrum master and 1 lead.

**Daily Routine**

* Since we follow the agile methodology, we have daily stand-up calls (discuss what we did yesterday and what we are going to do today.)
* After the stand-up call, I usually have a habit of checking the health of my application and look if the application is throwing any error codes. If there are any error codes, I try to resolve them or inform my manger about it, in next day stand-up.
* Then After that I work on my Jira story.
* Periodically we have code reviews. I do comments on pull requests, and I give suggestions on code or accept any suggestions from my peers.
* Developing code and testing in lower environments. If everything goes well, we promote the code to higher environments using ci/cd pipelines.
* Actively participating during deployments and resolving any issues associated with cloud.
* I have sync meetings everyday with other teams( to avoid dependency and blockers).

**Current employment: Centene (August 2021- Present) ---MediPro**

Currently, I am working in Centene Corporations as Full Stack Java Developer. . I am a part of the team **Med Pro Project.**

In this project, I am mostly responsible for creating web services and API that provides access to patients for viewing their information like health report, check-up, update the upcoming appointment date and API for doctors to use it to keep track of all the patient information.

I have used Angular for creating components, defining routes, and handling user input and integrating the React front-end with the Java back-end. To get the customer data I used forms where the user enters their data. I have used angular components to build the form and I have used many validations on the forms using the angular framework. After validation, the data is sent to the server using the HTTP request.

Used Cassandra for storing internal statements and reports and worked extensively in Development which includes updating and retrieving data internally from the back-end.

Writing unit tests and performing code testing to ensure the quality and reliability of the application, as well as debugging and fixing issues reported by the QA team

We use Oracle to store and retrieve user data for the RESTful applications by following CI/CD strategies using Jenkins. Coming to the Cloud platform, we have been mostly using AWS.

Our team follows the agile methodology so, we have daily stand-up calls and bi-weekly sprint calls followed by the retrospectives. Coming to the life cycle of the development of the project, we have a product owner that gathers the requirements for the project and we as a team sit together to analyse the requirements and create user stories and assign JIRA tickets. Once the JIRA tickets have been created and assigned to the team, we start working on the implementation. We perform unit testing from end- to end by writing test cases to test the code locally and pass on the code to the QA team. I am working in a team of 10 individuals with 3 developers on the backend, 2 on the frontend, 2 on devops with 1 tester, 1 scrum master and 1 lead.

**Coming to the technology stack, we use**

Front end: Angular. JS, HTML, CSS, JavaScript, Jquery.

Backend: Java, J2EE, JDK

Database: MySQL

ORM: Hibernate

Web Services: SOAP, RESTful

Testing: JUnit

Build Tool: Maven

Version Control: Git

Continuous Integration/Deployment: Jenkins

Cloud Platform: Amazon Web Services (AWS)

**Client2: Comcast, India**

I worked with the client Comcast as a full stack java developer. I work with the team called Xfinity Broadband that deals with generating Invoices for the services provided by Comcast. I mostly work for the broadband services. We have a third-party vendor and generate bills for them on a pay per use basis whenever they utilize our broadband services. In order to charge the vendors, we work on generating automated invoices. But before sending the invoice, we perform number of validations and send them the invoice via email.

We have been developing micro services using the Spring and Spring Boot technologies. I have used the React JS, JSX,on the front end for the user views. And for the backend we used the core Java, Spring Boot, Node.js. As I mentioned earlier about validating the information before generating the invoices, we use React which follows a component-based architecture, where the UI is divided into reusable and independent components. Each component encapsulates its logic and rendering, making it easier to manage and maintain the codebase. We use Mongo DB to store and retrieve user data for the applications from the devices. We follow CI/CD strategies using Jenkins. Once we merge the code into master branch, it gets deployed into our dev environment for testing purpose. We use Maven for dependency management we use Docker and AWS for deploying our projects.

Our team follows the agile methodology so, we have daily stand-up calls and bi-weekly sprint calls followed by the retrospectives. Coming to the life cycle of the development of the project, we have a product owner that gathers the requirements for the project and we as a team sit together to analyse the requirements and create user stories and assign JIRA tickets. Once the JIRA tickets have been created and assigned to the team, we start working on the implementation. We perform unit testing from end- to end by writing test cases to test the code locally and pass on the code to the QA team. I am working in a team of 10 individuals with 3 developers on the backend, 2 on the frontend, 2 on devops with 1 tester, 1 scrum master and 1 lead.

Coming to the technology stack, we use

Front-end: AngularJS, JSP (Jakarta Server Pages), jQuery, ReactJS, JSX

Back-end: Java, Spring Boot, Node.js

Database: MongoDB, Oracle

Java Frameworks: Spring, Spring MVC, Spring IOC, Spring Boot

Data Processing tool: Apache Spark

Deployment: Docker, AWS

Messaging: JMS, Kafka

Web Services: REST

Build tool: Maven.

Bug Tracking: Jira

Testing: Junit, Jasmine,Cucumber

Version Control : Git IDE: IntelliJ

**About Comcast,Philadelphia,PA:**

Comcast is an American multinational telecommunications company which is located in Philadelphia, Pennsylvania.

We were into Services project, our team are simplifying the core network architecture and improving efficiency and flexibility while reducing Capex and Opex costs associated with keeping a network up and running.

(Capital expenditures (CAPEX) are a company's major, long-term expenses while operating expenses (OPEX)

Comcast provides a one of the kind entertainment platforms, High Speed Data platform, and Internet of Things services including Home Security services .

We create world-class experiences that people love and trust and drive innovation that builds value. We bring millions TV and entertainment, sports and news, communications and home management, theme parks, home security, Voice, and high-speed Internet access. Comcast brings Entertainment, Internet, Voice, and other services to the customers via the largest footprint of networks in the country.

**Technology Detailed Description**

**Backend:**

**NodeJS\_:**

1. Yes, I have been using **Node JS** for around 2+ years now. I have created a webserver which deals with numerous core modules of Node JS like the Operating system, File System and Path Module. Along with Node JS, I have also used the Express JS framework for creating the REST APIs as it helps manage servers and routes.
2. I have knowledge in using the **NodeJS** framework for creating scalable network applications and back-end APIs. It is mostly used for building traditional web sites and back-end API services.
3. Used the Node.js Express server, along with technologies like, JavaScript/ jQuery and JSON to make connections between the servers and databases like MongoDB and MySQL.
4. We build REST web services by building Node.js Server in the back end to handle requests sent from the front-end that is the jQuery Ajax calls.
5. I have experience in using the Grunt task runner to perform frequent tasks such as minification, compilation, unit testing, and linting and installed latest version of libraries with Bower based on running in the Node.js environment.
6. I Used Node JS as the JavaScript runtime environment, Webpack for packaging and NPM for building the web applications.
7. We also use NODE JS to communicate between the angular application and the packages.

**Spring\_:**

I have experience with the Spring and Spring boot technologies.

**Spring Boot\_:**

1. I Implemented the back-end services using **Spring Boot** for microservices and to batch process the input files, scanning and posting the batch job requests to MQ.
2. Involved in the development using Spring boot with Data access/integration layer consisting of ORM module (Hibernate).
3. Create RFP (Request for Proposal) microservice to provide RESTful API utilizing Spring Boot with Spring MVC
4. Extensively Implemented Spring boot, dependency Injection (DI) and aspect-oriented programming (AOP) features along with **hibernate.**
5. Involved in setting up the application Service layer using Spring IOC framework.
6. Maintained relationship between objects using Spring IOC.
7. Used Spring DAO Module along with Struts Framework to deal with Database.

**Spring Boot Auto Configuration\_:**

* Spring Boot automatically configures a spring application based on dependencies present or not present in the class path as a jar, beans, properties, etc.
* It makes development easier and faster as there is no need to define certain beans that are included in the auto-configuration classes.

**Spring MVC Dispatcher\_:**

* In Spring MVC all incoming requests go through a single servlet is called Dispatcher Servlet (front controller). The front controller is a design pattern in web application development. A single servlet receives all the request and transfers them to all other components of the application.

**Spring Cloud\_:**

* I have used Spring Cloud to develop short lived microservices and run them locally.
* I have worked on Spring Cloud Stream framework for building highly scalable event-driven microservices connected with shared messages.
* I have experience in Spring Cloud Function; implemented business logic via functions and enable Spring Boot features like: Auto-configuration, dependency injection on serverless providers.
* I have in-depth knowledge in Spring Cloud Security for building secure applications. I have used feature likes: relay SSO tokens from front-end to back-end service or relay tokens between resources servers.
* I have worked on Spring Cloud for AWS which offer convenient way to interact with AWS provided services.

**Spring Batch\_:**

1. Spring Batch can be **used** in both simple **use** cases (such as reading a file into a database or running a stored procedure) as well as complex, high volume **use** cases (such as moving high volumes of data between databases, transforming it, and so on).) **for writing Logic.**
2. I worked on Conversion Application; migrate some financial transaction data from CSV to XML
3. I have experience in validation to ensure all input/output records are correct and consistent.
4. I have in-depth knowledge in using partitioning which allow large batch of application to run concurrently. I have partitioned both the input file and large database file.

**Spring IOC\_:**

1. Spring IoC to achieve loose coupling between Object’s dependencies. To achieve loose coupling and dynamic binding of the objects at runtime, objects dependencies are injected by other assembler objects.
2. Involved in setting up the application Service layer using Spring IOC framework and Maintained relationship between objects using Spring IOC.

**React\_:**

1. Used React JS to create views to hook up models to the DOM and synchronize data with server as a Single Page Application (SPA).
2. And also, to create Controllers to handle the events triggered by clients and send requests to server.
3. Experience in building isomorphic applications using React.js and Redux with GraphQL on server side. (Give everyone an easy and efficient way of accessing data (it uses less resource than the REST API, especially with mobile applications).
4. Proficient in using React JS for creating reusable components along with sliders, charts, grids and handling events and DOM objects. Extensively worked in writing minimal code in building re-usable forms with their validations using React with Redux Architecture.
5. Developed various web pages and components using React JS by implementing redux q components and libraries and predefined components from NPM (Node Package Manager).

**ANGULAR\_:**

**Angular: Angular JS (java script) and Angular 5 or 7 (typescript)**

* Worked on angular caching.
* Worked on material design from google within the angular
* Functionalities for navigating with in the pages.
* Functionalities for routing
* Have conducted unit testing with in the angular using karma and jasmine.
* efficiently worked with angular properties like event binding, property binding, data binding etc.,

1. I have experience in building the Single Page Applications (SPA) in front-end pages.
2. Developed Single Page Applications (SPA) using Angular, Typescript, HTML5, CSS3, and Bootstrap 3.
3. Created Services to consume REST API's and to communicate between components using Dependency Injection provided by Angular 6.
4. Built Web pages that are more user-interactive using Angular Framework.
5. Used various Angular 6 custom directives and developed reusable components and templates that can be re-used at various places in the application.
6. Implemented Angular Router to enable navigation from one view to the next as customer performs application tasks.
7. I have worked in http service library to work between front-end server and backend server of different framework or environment.
8. I have in-depth knowledge in route service for using query and route parameter in front-end page and also making route guards.
9. I have worked on building services, component, modules, and route guards.

**AWS\_:**

* I have good experience with AWS.
* I used to Plan, deploy, monitor, and maintain Amazon AWS cloud infrastructure consisting of multiple EC2 nodes and VMWare VMs as required in the environment.
* Worked in managing VMs in Amazon using AWS and EC2.
* Migrated applications from internal data center to AWS.
* Created Server instances on AWS.
* involved in maintaining the user accounts (IAM), RDS, Route S3 services in AWS Cloud.
* Selecting the appropriate AWS service based on data, compute, database, or security requirements.
* Deploy applications on AWS by using Elastic Beanstalk.
* Created monitors, alarms, and notifications for EC2 hosts using Cloud Watch.  Migrated applications to the AWS cloud.
* Integrated Amazon Cloud Watch with Amazon EC2 instances for monitoring and storing the log files on S3.
* Implemented and maintained the monitoring and alerting of production and corporate servers such as EC2 and storage such as S3 buckets using AWS Cloud Watch.

**AWS\_:**

1. Yes, I have used AWS for more than 2 years. I have used the concept like EC2 instance- created them and, managed them as per the requirement.
2. S3 bucket: which is used to store the file folders and images in it and later used it.
3. RDS: it is like a database for the aws.

**AWS LAMBDA/LAA:**

1. I have used the lambda to run the code in response to events and automatically manages the computer resources required for the code.
2. Suppose you want to create a thumbnail for each image file that is uploaded to a bucket. You can create a Lambda function (Create Thumbnail) that Amazon S3 can invoke when objects are created. Then, the Lambda function can read the image object from the source bucket and create a thumbnail image target bucket.

**Amazon SQS:**

* Amazon Simple Queue Service (Amazon SQS) is a pay-per-use web service for storing messages in transit between computers. Developers use SQS to build distributed applications with decoupled components without having to deal with the overhead of creating and maintaining message queues.

**Amazon SES:**

 With Amazon SES, I have sent transactional email, marketing messages, or any other type of high-quality content to your customers.

**What is Amazon EKS?**

* Amazon EKS (Elastic Container Service for Kubernetes) is a managed Kubernetes service that allows you to run Kubernetes on AWS without the hassle of managing the Kubernetes control plane.
* The Kubernetes control plane plays a crucial role in a Kubernetes deployment as it is responsible for how Kubernetes communicates with your cluster — starting and stopping new containers, scheduling containers, performing health checks, and many more management tasks.
* The big benefit of EKS, and other similar hosted Kubernetes services, is taking away the operational burden involved in running this control plane. You deploy cluster worker nodes using defined AMIs and with the help of CloudFormation, and EKS will provision, scale and manage the Kubernetes control plane for you to ensure high availability, security and scalability.

**Hibernate \_:**

* I have in-depth knowledge in using Hibernate as Object Relational Mapping (ORM) framework that implements the JPA java specifications in the Spring Boot, Spring MVC and Spring Framework
* I have experience in building DAO (Data Access Object) for required table in relational database and using DML (Data Manipulation Language) for CRUD operation.
* I have worked in Hibernate Validator; applying constraint on bean, method or creating custom constraints to make sure correct data in inserted in the database.
* Worked on Java framework, hibernate for the development of Java application to interact with the database. As in, it is an open source, lightweight, ORM (Object Relational Mapping) tool.

**Java\_:**

1. I have in depth knowledge of the Core Java, OOPS concepts and Collection Framework
2. I have experience in using proper Data Structure for the data storage and sorting algorithm.
3. I have used multi-threading to work with synchronized programming.
4. I am proficient in working with lambda and functional programming.
5. I Used Java 8 Lambda expressions for array operations, collections, and Enhanced Concurrency methods throughout the application for well-defined programming.
6. And Java 8 Stream APIs and functional style of programming.
7. I Used Core Java features of Util package for maintaining data structures, Threads and Exception’s handling.
8. I have experience in developing the RESTful APIs dynamically using Java, Spring MVC.
9. Used Java Message Service (JMS) for reliable and asynchronous exchange of important information that consumes the message from Java Message Queue
10. Java EE is built on top of Java SE, and it is used for developing web applications and large-scale enterprise applications.
11. Even Developed the RESTful APIs dynamically using Java, Spring MVC.

JPA (Java Persistence API)\_:

* The Java Persistence API (**JPA**) is one possible approach to ORM. Via **JPA** I have mapped, stored, updated, and retrieved data from relational databases to Java objects and vice versa.
* It is an ORM standard for storing, accessing, and managing Java object in a relational database.
* It is set of concepts that can be implemented by any tool or framework.

Why use JRE?

* JRE contains class libraries, JVM, and other supporting files. It does not include any tool for Java development like a debugger, compiler, etc.
* It uses important package classes like math, swing, util, lang, awt (abstract window toolkit), and runtime libraries.
* If you have to run Java applets, then JRE must be installed in your system.

Why JVM?

* JVM provides a platform-independent way of executing Java source code.
* It has numerous libraries, tools, and frameworks.
* Once you run a Java program, you can run on any platform and save lots of time.
* JVM comes with JIT (Just-in-Time) compiler that converts Java source code into low-level machine language. Hence, it runs faster than a regular application.

Why use JDK?

* JDK contains tools required to write Java programs and JRE to execute them.
* It includes a compiler, Java application launcher, Apple tv viewer, etc.
* Compiler converts code written in Java into byte code.
* Java application launcher opens a JRE, loads the necessary class, and executes its main method.

**API\_:**

API stands for Application Programming Interface, which specifies how one component should interact with the other. It consists of a set of routines, protocols, and tools for building the software applications.

1. REST/RESTFUL/REST API’s

Rest stands for Representation State Transfer.

It means Every Resource can be represented in multiple forms like JSON, XML,YAML,HTML etc.

Whenever there is a request from the client then the representation of current state of the resource is being transferred between client and server.

In my application I used postman and Rest Client for testing rest api’s. I used annotations like @Get , @Post, @Put, @Delete which are basically CRUD operations.

**Restful API**

* Good experience with REST
* Worked on exposing the APIs and consuming them.
* Uses HTTP requests to GET, PUT, POST, DELETE data.
* Preferred because less bandwidth consumption and designed to be suitable for internet usage.

**RESTful API & GraphQL API**

1. I have in-depth knowledge in CRUD (Create Read Update Delete) operation and building API in Spring Boot and Node JS Framework
2. Created Services to consume REST API's and to communicate between components using Dependency Injection provided by Angular 6.
3. I have experience in testing RESTful API using tools like: Postman to send request to server and check whether response is accurate.
4. I have worked in fetching the data from the GraphQL API using Node JS framework and installing required libraries for it.
5. Experience in building isomorphic applications using React.js and Redux with GraphQL on server side. (Give everyone an easy and efficient way of accessing data (it uses less resource than the REST API, especially with mobile applications).
6. Yes, I have created all the form that is put post delete and get of the rest api.

**Microservices\_:**

Microservices: Microservices is an approach to application development in which a large application is built as a suite of modular services. Each module supports a specific business goal and uses a simple, well-defined interface to communicate with other sets of services. Microservices make it easier to test, understand, and maintain application builds with the combination of independent components. It is an excellent solution for building large-scale products and improving workflows and productivity. Distributed teams often find it easier to create applications with microservice architecture. Faster Project development, flexibility, productivity, Scalability, and isolation

Microservices Advantages: Are easily deployed, require less production time, can Scale quickly, can be reused among different projects, work well with containers, such as Docker, Complement cloud activities.

Microservices Disadvantages: Potentially too granular, Latency during heavy use, testing can be complex.

Microservices vs. monolithic architecture: Microservices came about to help solve the frustrations developers were having with large applications that require change cycles to be tied together. In a monolithic application, any small change or update required building and deploying an entirely new application. This inevitably means any application development entails a certain amount of planning, preparation, time and, potentially, money.

Microservices architecture vs. SOA: SOA has been the standard development practice for nearly two decades. However, the resourcefulness of SOA is coming into question when working with cloud computing. With cloud, SOA lacks scalability and slows down with work request changes, limiting application development.

Some developers maintain that microservices are just a more granular approach to service-oriented architecture. Proponents of the SOA model consider microservices architecture as the natural evolution of SOA to accommodate cloud computing.

How microservices work: In a microservices architecture, each service runs a unique process and usually manages its own database. This not only provides development teams with a more decentralized approach to building software, it also allows each service to be deployed, rebuilt, redeployed and managed independently.

Microservices security: Microservices architecture can alleviate some security issues that arise with monolithic applications. Microservices simplifies security monitoring because the various parts of an app are isolated. A security breach could happen in one section without affecting other areas of the project. Microservices provide resistance against distributed denial-of-service attacks when used with containers by minimizing an infrastructure takeover with too many server requests.

**MVC pattern\_:**

* MVC is the architecture pattern which was introduced in the 1970s and was the first and foremost option to be used for web application designing. MVC tries to let the developer build the application in a manner where the layers in concern are separated. This helps, in turn, to take away all the efforts needed to extend, test and maintain the application. In the normal traditional software development method, we make use of user control or write relevant code to include the view in the definition class. This process increases the size of view class among UI, data binding logic and business operations. To sum up, the MVC architecture pattern works towards making a code good code by reducing its size. It works on to make the codes easily manageable and cleaner.

**MVP pattern**

* If you compare the MVP pattern with that of MVC, you will find a good number of similarities. Here in MVP, ‘P’ stands for Presenter. The page controls are managed and displayed by View. On behalf of the View, The Presenter is responsible for addressing all the UI events. Presenter collects all the input coming from the users and then moves it to the Model side to side, which then takes the result to the View. It is from the logic end that the Presenter does all its work for gestures like directing through the navigation or pushing a button. When it comes to implementation, MVP is a compound pattern, but if you manage to apply it as an amazingly well-designed solution, it will offer you great benefits. In most of the cases, MVP is selected for ASP.NET Web Forms and Windows Forms applications. It can be used in Java applications and so is the right choice for Android developers.

**Model View Model (MVVM) Pattern**

* It is from the MVC pattern that the MVVM has been defined. In the case of the MVVM pattern, you will find two-way data binding taking place between View and View-Model. Here inside the View-Model and to the View, it allows carrying out modifications automatically. In order to make changes in the View-Model, the View-Model makes use of the observer pattern. The MVVM pattern is mostly used by Silverlight, WPF, nRoute, Caliburn, etc.

**SOAP\_:**

* SOAP is an acronym for Simple Object Access Protocol. It is an XML-based messaging protocol for exchanging information among computers. It is an application of the XML specification. SOAP is a communication protocol designed to communicate via Internet.

**Why Maven?**

Maven is a Java Tool which helps developer to build java projects easily.

It has many samples project sceptical or layouts in its repository. Based on our application it quickly builds the basic structure of our web application including dependencies.

Maven does not have any xml file It maintains only one POM.xml file for all dependencies and configuration.

Nowadays mostly everyone is using maven to develop applications.

**Advantage:**

If we developed our project in maven and in future if we would like to use the updated version of tools then we just need to update the maven dependencies we don’t need to develop our whole application from scratch.

For creating a maven project, In eclipse IDE we have to click on file and then new Maven project. We have to give group id, artefact id, version , package, and any dependencies (database, jpa etc) if needed. Then maven automatically creates a project structure with dependencies in pom.xml file.

**Maven\_:**

1. I have created the application using the Maven dependencies through the POM.xml file, which adds these dependencies automatically in the application.
2. Maven is a project management and comprehension tool that provides developers a complete build lifecycle framework.
3. Maven can add all the dependencies required for the project automatically by reading pom file.
4. Maven provides developers ways to manage the following:

* Builds
* Documentation
* Reporting
* Dependencies S
* SCMs
* Releases
* Distribution
* Mailing list
* POM stands for Project Object Model.

## **Front-end JavaScript/Angular/React**

**JavaScript\_:**

1. Yes, I have used JavaScript for almost all of my projects, I have used all the ES5 and ES6 features of the JavaScript like async, await and promises for asynchronous code and worked with many of the JavaScript libraries and frameworks like NodeJS, ExpressJS, Angular 8 and react.
2. I have used in the AJAX in web development.
3. I have worked in call back function and event binding for user response.

**Angular\_:**

Angular JS (java script) and Angular 5 or 7 (cc)

* Angular is a framework used to create a single dynamic web page application.
* Angular divides our complex web application into number of components and modules . Each component has its own template and business logic.
* There are also pipes in Angular which transforms our output. Some of the built-in pipes are uppercase pipe which converts our output into all Capital letters. Date pipe, Which displays date. Symbol of pipe is |.
* We have some more inbuilt pipes like slice pipe, currency pipe etc which we can find in the api column of angular official website.

**ANGULAR\_:**

**Angular: Angular JS (java script) and Angular 5 or 7 (typescript)**

* Worked on angular caching.
* Worked on material design from google within the angular
* Functionalities for navigating with in the pages.
* Functionalities for routing
* Have conducted unit testing with in the angular using karma and jasmine.
* efficiently worked with angular properties like event binding, property binding, data binding etc.,

1. I have experience in building the Single Page Applications (SPA) in front-end pages.
2. Developed Single Page Applications (SPA) using Angular, Typescript, HTML5, CSS3, and Bootstrap 3.
3. Created Services to consume REST API's and to communicate between components using Dependency Injection provided by Angular 6.
4. Built Web pages that are more user-interactive using Angular Framework.
5. Used various Angular 6 custom directives and developed reusable components and templates that can be re-used at various places in the application.
6. Implemented Angular Router to enable navigation from one view to the next as customer performs application tasks.
7. I have worked in http service library to work between front-end server and backend server of different framework or environment.
8. I have in-depth knowledge in route service for using query and route parameter in front-end page and also making route guards.
9. I have worked on building services, component, modules and route guards.

**React\_:**

1. Used React JS to create views to hook up models to the DOM and synchronize data with server as a Single Page Application (SPA).
2. And also, to create Controllers to handle events triggered by clients and send request to server.
3. Experience in building isomorphic applications using React.js and Redux with GraphQL on server side. (Give everyone an easy and efficient way of accessing data (it uses less resource than the REST API, especially with mobile applications).
4. Proficient in using React JS for creating reusable components along with sliders, charts, grids and handling events and DOM objects. Extensively worked in writing minimal code in building re-usable forms with their validations using React with Redux Architecture.
5. Developed various web pages and components using React JS by implementing redux qcomponents and libraries and predefined components from NPM (Node Package Manager).

**Class Component\_:**

* Class component are the one used for the state management. Like suppose I have to change the state or make management of the state then we are using the Class component.
* It is also called as the Stateful components.
* In class-based component we used the concept called as the Life Cycle Hooks.

**Functional Component\_:**

* Functional components are not used for the state management and hence also called as the stateless component. It is also called as Presentational component.
* The Life cycle hooks does not works in the Functional Components, rather the React Hooks works in the functional components.

**Redux\_:**

* The redux is not tightly coupled with the react, we need to install it and then then used it in the react application. The redux is used for the state management.
* It works as per 3 concepts:

1. Action: It is only to connect the react app with the redux.
2. It has the action to be performed… give the ice cream example.
3. Reducer: It is the one that take the initial State and the action and returns the final State.
4. Store: store is the place where we can see the all the updated data.

**React Native\_**

Basic Concept:

1. View: It is most fundamental component for building a UI, View is a container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, <div>, android. View, etc. View is designed to be nested inside other views and can have 0 to many children of any type.
2. Text: A React component for displaying text. supports nesting, styling, and touch handling.
3. Image: A React component for displaying different types of images, including network images, static resources, temporary local images, and images from local disk, such as the camera roll.
4. TextInput: A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.
5. Scroll View: Component that wraps platform Scroll View while providing integration with touch locking "responder" system. Keep in mind that Scroll Views must have a bounded height in order to work.

**React\_:**

* Used React JS to create views to hook up models to the DOM and synchronize data with server as a Single Page Application (SPA).
* And also, to create Controllers to handle events triggered by clients and send request to server.
* Experience in building isomorphic applications using React.js and Redux with GraphQL on server side. (Give everyone an easy and efficient way of accessing data (it uses less resource than the REST API, especially with mobile applications).
* Proficient in using React JS for creating reusable components along with sliders, charts, grids and handling events and DOM objects. Extensively worked in writing minimal code in building re-usable forms with their validations using React with Redux Architecture.
* Developed various web pages and components using React JS by implementing redux components and libraries and predefined components from NPM (Node Package Manager).

Class Component\_:

• Class component are the one used for the state management. Like suppose I have to change the state or make management of the state then we are using the Class component.

• It is also called as the Stateful components.

• In class-based component we used the concept called as the Life Cycle Hooks.

**Functional Component\_:**

• Functional components are not used for the state management and hence also called as the stateless component. It is also called as Presentational component.

• The Life cycle hooks does not work in the Functional Components, rather the React Hooks works in the functional components

**Redux\_:**

The redux is not tightly coupled with the react, we need to install it and then then used it in the react application. The redux is used for the state management.

It works as per 3 concepts:

* Action: It is only to connect the react app with the redux.
* It has the action to be performed… give the ice cream example.
* Reducer: It is the one that take the initial State and the action and returns the final State.
* Store: store is the place where we can see the all the updated data.

**React Native (0.62)\_:**

Basic Concept:

    View : It is most fundamental component for building a UI, View is a container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, <div>, android.view, etc. View is designed to be nested inside other views and can have 0 to many children of any type.

Text : A React component for displaying text. supports nesting, styling, and touch handling.

Image : A React component for displaying different types of images, including network images, static resources, temporary local images, and images from local disk, such as the camera roll.

TextInput: A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.

Scroll View : Component that wraps platform Scroll View while providing integration with touch locking "responder" system. Keep in mind that Scroll Views must have a bounded height in order to work,

Stylesheet :  CSS

IOS  showActionSheetWithOptions()\_:

Action Sheet: An action sheet is a specific style of alert that presents two or more choices related to the current context. On smaller screens, an action sheet slides up from the bottom of the screen; on larger screens, an action sheet appears all at once as a popover.

       Use an action sheet to request confirmation before performing a potentially destructive operation. To give people a choice of items or actions that are related to a non-destructive task.

Advantage:

* React Native speeds up the development and saves money.
* React is the most popular cross-platform framework for mobile development.
* Allow third party plugins, wrote in JS which is familiar for most developer, Have pre-build component
* Developed an mobile application for both IOS and Android Platform as
* It was easier to learn since I know most of the React concept and had to learn few of the Native concept
* It is flexible to work with other platform-specific code like swift
* I have worked in redux for React Native to write application to run in different environment

Microsoft Power BI is a business intelligence platform that provides nontechnical business users with tools for aggregating, analysing, visualizing and sharing data. Power BI's user interface is fairly intuitive for users familiar with Excel and its deep integration with other Microsoft products makes it a very versatile self-service tool that requires little upfront training.

A free version of Power BI is intended for small to midsize business owners; a professional version called Power BI Plus is available for a monthly subscription fee. Users can download an application for Windows 10, called Power BI Desktop, and native mobile apps for Windows, Android and iOS devices. There is also Power BI Report Server for companies that must maintain their data and reports on premises. That version of Power BI requires a special version of the desktop app -- aptly called Power BI Desktop for Power BI Report Server.

Common uses of Power BI

Microsoft Power BI is used to find insights within an organization's data. Power BI can help connect disparate data sets, transform and clean the data into a data model and create charts or graphs to provide visuals of the data. All of this can be shared with other Power BI users within the organization.

The data models created from Power BI can be used in several ways for organizations, including telling stories through charts and data visualizations and examining "what if" scenarios within the data. Power BI reports can also answer questions in real time and help with forecasting to make sure departments meet business metrics.

Power BI can also provide executive dashboards for administrators or managers, giving management more insight into how departments are doing.

What is Tableau?

Tableau is a powerful and fastest growing data visualization tool used in the Business Intelligence Industry. It helps in simplifying raw data in a very easily understandable format. Tableau helps create the data that can be understood by professionals at any level in an organization. It also allows non-technical users to create customized dashboards.

Data analysis is very fast with Tableau tool and the visualizations created are in the form of dashboards and worksheets.

The best features of Tableau software are:

* Data Blending
* Real time analysis
* Collaboration of data

What is Tableau used for?

Usage of Tableau software are listed below:

* Tableau software is used to translate queries into visualization.
* It is also used for managing metadata.
* Tableau software imports data of all sizes and ranges.
* For a non-technical user, Tableau is a life saver as it offers the facility to create ‘no-code’ data queries.
* Ever since it was introduced, this data visualization tool is used for Business Intelligence industry.

Angular Developer\_:

* Created Services to consume REST API's and to communicate between components using Dependency Injection provided by Angular 8.
* Developed Single Page Applications (SPA) using Angular, Typescript, HTML5, CSS3, and Bootstrap 3.
* Built Web pages that are more user-interactive using Angular Framework.
* I have experience in building the Single Page Application (SPA) in front-end page
* I have worked in http service library to work between front-end server and backend server of different framework or environment
* I have in-depth knowledge in route service for using query and route parameter in front-end page and also making route guards
* I have worked on building services, component, modules and route guards
* Used various Angular 8 custom directives and developed reusable components and templates that can be re-used at various places in the application.
* Implemented Angular Router to enable navigation from one view to the next as customer performs application tasks.

What is Grafana & What Is It Used For?

* Grafana is an open source solution for running data analytics, pulling up metrics that make sense of the massive amount of data & to monitor our apps with the help of cool customizable dashboards.
* Grafana connects with every possible data source, commonly referred to as databases such as Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL etc.
* Grafana being an open source solution also enables us to write plugins from scratch for integration with several different data sources.
* The tool helps us study, analyse & monitor data over a period of time, technically called time series analytics.
* It helps us track the user behaviour, application behaviour, frequency of errors popping up in production or a pre-prod environment, type of errors popping up & the contextual scenarios by providing relative data.
* A big upside of the project is it can be deployed on-prem by organizations which do not want their data to be streamed over to a vendor cloud for security & other reasons.
* Over time this framework has gained a lot of popularity in the industry & is deployed by big guns such as PayPal, eBay, Intel & many more. I’ll talk about the industry use cases up ahead in the article.
* Besides the core open source solution there are other two services offered by the Grafana team for businesses known as the Grafana Cloud & the Enterprise.

New Relic One is an observability platform built to help engineers create more perfect software. From monoliths to serverless, you can instrument everything, then analyse, troubleshoot, and optimize your entire software stack. All from one place.

Kotlin : Functional Programming

* Main purpose is to make code and method more concise and smaller.
* Can also work with JS , Android and Native

**Node JS**: (It is a server-side framework)

* Experience in writing Node.js express server, and using technologies like , JavaScript/ jQuery, JSON to make connections between server and databases like MongoDB and MySQL.
* Build REST web service by building Node.js Server in the back end to handle requests sent from the front-end jQuery Ajax calls.
* Good knowledge with Grunt task runner and installed latest version of libraries with Bower based on running in the Node.js environment.
* Used Node JS as JavaScript runtime environment, Webpack for packaging and NPM for building the web application.
* Using NODE JS to communicate between the angular application and the packages.

**SOAP**

SOAP is an acronym for Simple Object Access Protocol. It is an XML-based messaging protocol for exchanging information among computers. It is an application of the XML specification. SOAP is a communication protocol designed to communicate via Internet.

## **Databases Oracle/PostgreSQL/SQL/mongo dB/MySQL**

**I worked on databases like Oracle (Knowledge about the oracle database for storing and retrieving the database, as it stores the large amount of database in the computer system and we can also create a multiple table and the relation between them by using the SQL query language, MongoDB (I have used Mongo DB for the storing and retrieving of the data in the flexible way, as the data store is in JASON format rather than tables format which makes the access of the data faster)**

* **PostgreSQL\_ is an advanced, enterprise class open-source relational database that supports both SQL (relational) and JSON (non-relational) querying. It is a highly stable database management system, backed by more than 20 years of community development which has contributed to its high levels of resilience, integrity, and correctness. PostgreSQL is used as the primary data store or data warehouse for many web, mobile, geospatial, and analytics applications. The latest major version is PostgreSQL 12.**

* **SQL \_ is used for storing and retrieving the data between the databases. It is a query language that stores and retrieve, update the data from the database as per the requirement. Even created the database and the new tables using the sql.**
* **MySQL\_ I am proficient in working with MySQL as database for the web application. I used phpMyAdmin as the software tools for web interface, apache tomcat.**

1. **Recursive Theory**
2. **Anchor Theory**

**Oracle\_:**

1. **I have hands-on experience with the Oracle database for storing and retrieving the database, as it stores a large amount of database in the computer system**
2. **I have worked in large web application project with Spring Boot, Spring MVC as backend service. Client required faster performance and needed to be saved large amount of data/record in both Structure (MySQL) and Non-Structural (NoSQL) database. We used Oracle as database. I have hands-on experience working with oracle and it’s other service like oracle autonomous database.**
3. **Oracle DB(11g - 12c) / My Sql (before 5.7- now 8.0)**

* **I used Oracle Db in my application for storing user data. I have configured my driver class name, url, user name and password in my configuration file.**
* **It is a relational database.Database which organizes its data in form of tables.**
  + **Each table has rows and column. Each row is called a record and each column is called as field or attribute. We use SQL queries to interact with database. Examples of relational database is oracle, Mysql, IBM DB2 etc.**

**ORACLE:**

**used joins stored procedures, views, unions in Oracle database to fetch the data.**

**Developed stored procedures, functions, SQL, NOSQL queries and backend programs using Oracle database and tools such as TOAD and SQL developer.**

**Implemented MVC architecture using JSF framework using JSPs as GUI development and DB Modelling using Oracle 10g.**

**CTE: Common Table Expression**

**Written recursive CTE. 3 kinds of implementations:**

1. **Separated**

**MongoDB\_:**

1. **I have used Mongo DB for the storing and retrieving of the data in a flexible way, as the data store is in JASON format rather than tables format which makes the access of the data faster.**
2. **I have experience in building web application with Node JS as back end and Angular as Front-end service. I have used MongoDB Atlas as cloud database and used mongoose ORM and its API to perform CRUD operation in database.**

**A relational database is structured, meaning the data is organized in tables. Many times, the data within these tables have relationships with one another, or dependencies. A non-relational database is document-oriented, meaning, all information gets stored in more of a laundry list order. Within a single construct, or document, you will have all of your data listed out.**

**SQL Databases (Relational)**

**SQL is short for Structured Query Language, basically meaning a very firm way of sorting through data in the form of tables, columns, and rows. How is data structured in an SQL database? The table itself would be made up really of one variable or object that we would be looking through. The column would represent the data point itself that needs to be stored and the row is a record of the data points per column.**

**Relational Databases are very organized**

**For example, if you are looking to sort data regarding what the weather is at a certain time of the day during a certain day, it would be structured as the following:**

**Table: Weather**

**Columns: Days of the Week**

**Rows: Time of Day**

**Data Points: Degrees Fahrenheit**

**In this structure, all queries would be related to this table and the structure of the table would allow for easy sorting, filtering, computations, etc. If we ever need to establish a connection between tables, say, you want to know what the weather was at a certain time and relate that to a baseball game’s predicted score, then what we do is create what is called a key. This key allows for connections to be made between two or more tables to solidify associations between the two.**

**Some popular SQL database systems include:**

* **Oracle**
* **Microsoft SQL Server**
* **PostgreSQL**
* **MySQL**
* **MariaDB**

**Pros of a Relational Database**

* **Data is easily structured into categories.**
* **Your data is consistent in input, meaning, and easy to navigate.**
* **Relationships can be easily defined between data points.**

**NoSQL Databases (Non-Relational)**

**In contrast to a relational database, a NoSQL database is one that is less structured/confined in format, and thus, allows for more flexibility and adaptability. If you are going to be dealing with a dataset that isn’t clearly defined, meaning not organized or structured, you likely won’t have the luxury of establishing defined tables and relationships amongst the datasets.**

**Non-relational databases are less structured**

**For example, Facebook Messenger uses a NoSQL database, because the information that is being gathered isn’t structured enough to be segmented into tables and define relationships between each other. With tons of unstructured information, it needs to be held in a non-relational database. Think of the information as being stored on one large word document. Everything is there. As more information gets entered, the document gets longer. If you want to find and pull data, you have to in essence ‘control/command + F’ and search for the data itself.**

**Some popular NoSQL databases include:**

* **MongoDB**
* **Cassandra**
* **Redis**
* **Apache HBase**
* **Amazon DynamoDB**

**Pros of a Non-Relational Database**

* **Data is not confined to a structured group.**
* **You can perform functions that allow for greater flexibility.**
* **Your data and analysis can be more dynamic and allow for more variant inputs.**

## **Devops (CI/CD), Jenkins, docker, Kubernetes**

**CI/CD\_:**

1. We were having the DevOps engineer in our teams, so I have not created the CI-CD pipelines but have worked effectively with the team and has a deep understanding of it.
2. I have used the CI/CD pipeline to automate the software delivery process. Then this builds the code, runs the test, and safely deploy a new version of application.
3. I have used CI (Continuous integration) to build, test and merge the application code changes.
4. I have used the CD (continuous delivery) to deploy the code in the environment by the manual trigger.

(IF they ask have u created it: say no... I have consumed.)

**Docker\_:**

1. I used docker to create, deploy, and run applications.
2. I’m using Docker to provide our restful API to enable a remote API on the docker container to build and modify the docker image in the harbour. Docker uses containers on the host's operating system to run applications.
3. Docker Containers • Docker containers are the lightweight alternatives of the virtual machine. • It allows developers to package up the application with all its libraries and dependencies and ship it as a single package.

**Kubernetes\_: v:1.12**

1. I prefer Kubernetes because it provides a framework to run distributed systems resiliently, and it is open source.
2. I used Kubernetes to bundle and run my application in containers.
3. Containers are important because they run the application and ensure that there is no downtime. For example, if one container goes down, another container needs to start. So, Kubernetes is helping us to maintain this process.

**Jenkins\_:**

1. Jenkins is an open-source automation tool written in Java programming language that allows continuous integration.
2. WITH Jenkins I builds and tests our software projects which continuously make it easier for developers to integrate changes to the project and making it easier for users to obtain a fresh build.
3. It also allows us to continuously deliver our software by integrating with many testing and deployment technologies.

**Kubernetes vs Dockers**

1. Kubernetes allows us to use our existing docker containers and workloads but allow us to tackle the complexity issue run into.
2. I prefer Kubernetes because it provides a framework to run distributed systems resiliently, and it is open source.
3. I used Kubernetes to bundle and run my application in containers.
4. Containers are important because they run the application and ensure that there is no downtime. For example, if one container goes down, another container needs to start. So, Kubernetes is helping us to maintain this process.
5. Ubuntu OS when working with docker and install the docker daemon when using it. After scaling out the simple deployment is easy but when user increase then we need to scale out for better user experience.
6. When scaling we can use new hardware, but dev team created a new microservices and big advantage using microservice is to scale out individual component individually. This is where Kubernetes as orchestra tools, let us use docker zed application and make effective use of server and space.
7. In server stack, Kubernetes known it as working node and has main node called master node. As all worker node relates to the master node and decide where to host the application, piece them together and orchestra them.

**Main advantages of Kubernetes are:**

* deployment,
* making development easier
* providing monitoring tools.

**Scenario:**

1. When we want to deploy frontend application 8 times and have policy of space it takes. Kubernetes deployment is not one-time thing but grows with the application.
2. If any application crash, then it will redeploy then Kubernetes will automatically restart it. If frontend need to access data point with microservices then It need to connect with one of them.
3. Kubernetes deploy load balancers and takes advantage of service registry and discovery abilities and make application talks to each other using something called Kubernetes service.
4. Kubernetes need to create separate service for the separate application. It makes development easier. It has built-in technology to investigate log, CPU load and open source has developed number of tools.

I worked on databases like Oracle (Knowledge about the oracle database for storing and retrieving the database, as it stores the large amount of database in the computer system and we can also create a multiple table and the relation between them by using the SQL query language, MongoDB (I have used Mongo DB for the storing and retrieving of the data in the flexible way, as the data store is in JASON format rather than tables format which makes the access of the data faster)

**PostgreSQL** is an advanced, enterprise class open-source relational database that supports both SQL (relational) and JSON (non-relational) querying. It is a highly stable database management system, backed by more than 20 years of community development which has contributed to its high levels of resilience, integrity, and correctness. PostgreSQL is used as the primary data store or data warehouse for many web, mobile, geospatial, and analytics applications. The latest major version is PostgreSQL 12.

**SQL** is used for storing and retrieving the data between the databases. It is a query language that stores and retrieve, update the data from the database as per the requirement. Even created the database and the new tables using the SQL.

**MySQL** I am proficient in working with MySQL as database for the web application. I used phpMyAdmin as the software tools for web interface, Apache tomcat.

**Oracle\_:**

1. I have hands-on experience with the Oracle database for storing and retrieving the database, as it stores a large amount of database in the computer system.
2. I have worked in large web application project with Spring Boot, Spring MVC as backend service. Client required faster performance and needed to be saved large amount of data/record in both Structure (MySQL) and Non-Structural (NoSQL) database. We used Oracle as database. I have hands-on experience working with oracle and it’s other service like oracle autonomous database.

**MongoDB\_:**

1. Generally, we use MongoDB as the main data store for the operational requirements with live needs. Generally, MongoDB is suitable for 80% of the applications which we develop today. MongoDB is simple to operate and extent in ways that are tough if they are not possible with the relational databases.
2. MongoDB stands out in various use cases where the relational databases are not suitable, like applications with semi-structured, structured, along with the big scalability needs or the multi-datacentre deployments.
3. MongoDB cannot be suitable for some applications. For instance, applications that need complex transactions and scan-based applications that access huge subsets of the data largely cannot be suitable for MongoDB.
4. Some general uses of MongoDB comprise product catalogues, mobile apps, content management, real-time personalization, and applications providing individual views throughout several systems.
5. I have used Mongo DB for the storing and retrieving of the data in a flexible way, as the data store is in JASON format rather than tables format which makes the access of the data faster.
6. I have experience in building web application with Node JS as back end and Angular as Front-end service. I have used MongoDB Atlas as cloud database and used mongoose ORM and its API to perform CRUD operation in database. Building internet and business applications that need to evolve quickly and scale elegantly. MongoDB is popular with developers of all kinds who are building scalable applications using agile methodologies.  
   MongoDB is a great choice if one needs to:
7. Support a rapid iterative development.
8. Scale to high levels of read and write traffic - MongoDB supports horizontal scaling through Sharding, distributing data across several machines, and facilitating high throughput operations with large sets of data.
9. Scale your data repository to a massive size.
10. Evolve the type of deployment as the business changes.
11. Store, manage and search data with text, geospatial, or time-series dimensions.

A **relational database** is structured, meaning the data is organized in tables. Many times, the data within these tables have relationships with one another, or dependencies.

A **non-relational database** is document-oriented, meaning, all information gets stored in more of a laundry list order. Within a single construct, or document, you will have all your data listed out.

**SQL Databases (Relational)**

SQL is short for Structured Query Language, basically meaning a very firm way of sorting through data in the form of tables, columns, and rows. How is data structured in an SQL database? The table itself would be made up really of one variable or object that we would be looking through. The column would represent the data point itself that needs to be stored and the row is a record of the data points per column.

Relational Databases are very organized.

For example, if you are looking to sort data regarding what the weather is at a certain time of the day during a certain day, it would be structured as the following:

Table: Weather

Columns: Days of the Week

Rows: Time of Day

Data Points: Degrees Fahrenheit

In this structure, all queries would be related to this table and the structure of the table would allow for easy sorting, filtering, computations, etc. If we ever need to establish a connection between tables, say, you want to know what the weather was at a certain time and relate that to a baseball game’s predicted score, then what we do is create what is called a key. This key allows for connections to be made between two or more tables to solidify associations between the two.

Some popular SQL database systems include:

* Oracle
* Microsoft SQL Server
* PostgreSQL
* MySQL
* MariaDB

Pros of a Relational Database

* Data is easily structured into categories.
* Your data is consistent in input, meaning, and easy to navigate.
* Relationships can be easily defined between data points.

**NoSQL Databases (Non-Relational)**

In contrast to a relational database, a NoSQL database is one that is less structured/confined in format, and thus, allows for more flexibility and adaptability. If you are going to be dealing with a dataset that isn’t clearly defined, meaning not organized or structured, you likely won’t have the luxury of establishing defined tables and relationships amongst the datasets.

**Non-relational databases are less structured.**

For example, Facebook Messenger uses a NoSQL database, because the information that is being gathered isn’t structured enough to be segmented into tables and define relationships between each other. With tons of unstructured information, it needs to be held in a non-relational database. Think of the information as being stored on one large word document. Everything is there. As more information gets entered, the document gets longer. If you want to find and pull data, you have to in essence ‘control/command + F’ and search for the data itself.

Some popular NoSQL databases include:

* MongoDB
* Cassandra
* Redis
* Apache HBase
* Amazon DynamoDB

Pros of a Non-Relational Database

* Data is not confined to a structured group.
* You can perform functions that allow for greater flexibility.
* Your data and analysis can be more dynamic and allow for more variant inputs.

## **Cloud AWS, PCF**

**AWS\_:**

* I have good experience with AWS.
* I used to Plan, deploy, monitor, and maintain Amazon AWS cloud infrastructure consisting of multiple EC2 nodes and VMWare VMs as required in the environment.
* Worked in managing VMs in Amazon using AWS and EC2.
* Migrated applications from internal data center to AWS.
* Created Server instances on AWS.
* involved in maintaining the user accounts (IAM), RDS, Route S3 services in AWS Cloud.
* Selecting the appropriate AWS service based on data, compute, database, or security requirements.
* Deploy applications on AWS by using Elastic Beanstalk.
* Created monitors, alarms, and notifications for EC2 hosts using Cloud Watch.  Migrated applications to the AWS cloud.
* Integrated Amazon Cloud Watch with Amazon EC2 instances for monitoring and storing the log files on S3.
* Implemented and maintained the monitoring and alerting of production and corporate servers such as EC2 and storage such as S3 buckets using AWS Cloud Watch.

**AWS\_:**

1. Yes, I have used AWS for more than 2 years. I have used the concept like EC2 instance- created them and, managed them as per the requirement.
2. S3 bucket: which is used to store the file folders and images in it and later used it.
3. RDS: it is like a database for the aws.

**AWS LAMBDA/LAA:**

1. I have used the lambda to run the code in response to events and automatically manages the computer resources required for the code.
2. Suppose you want to create a thumbnail for each image file that is uploaded to a bucket. You can create a Lambda function (Create Thumbnail) that Amazon S3 can invoke when objects are created. Then, the Lambda function can read the image object from the source bucket and create a thumbnail image target bucket.

**Amazon SQS:**

* Amazon Simple Queue Service (Amazon SQS) is a pay-per-use web service for storing messages in transit between computers. Developers use SQS to build distributed applications with decoupled components without having to deal with the overhead of creating and maintaining message queues.

**Amazon SES:**

 With Amazon SES, I have sent transactional email, marketing messages, or any other type of high-quality content to your customers.

**What is Amazon EKS?**

* Amazon EKS (Elastic Container Service for Kubernetes) is a managed Kubernetes service that allows you to run Kubernetes on AWS without the hassle of managing the Kubernetes control plane.
* The Kubernetes control plane plays a crucial role in a Kubernetes deployment as it is responsible for how Kubernetes communicates with your cluster — starting and stopping new containers, scheduling containers, performing health checks, and many more management tasks.
* The big benefit of EKS, and other similar hosted Kubernetes services, is taking away the operational burden involved in running this control plane. You deploy cluster worker nodes using defined AMIs and with the help of CloudFormation, and EKS will provision, scale and manage the Kubernetes control plane for you to ensure high availability, security and scalability.

**STEPS**

You will need to make sure you have the following components installed and set up before you start with Amazon EKS:

* AWS CLI – while you can use the AWS Console to create a cluster in EKS, the AWS CLI is easier. You will need version 1.16.73 at least. For further instructions, click here.
* Kubectl – used for communicating with the cluster API server. For further instructions on installing, click here. This endpoint is public by default but is secured by proper configuration of a VPC (see below).
* AWS-IAM-Authenticator – to allow IAM authentication with the Kubernetes cluster. Check out the repo on GitHub for instructions on setting this up.

Step 1: Creating an EKS role.

Step 2: Creating a VPC for EKS.

Step 3: Creating the EKS cluster.

Step 4: Launching Kubernetes worker nodes.

Step 5: Installing a demo app on Kubernetes.

**PCF (Pivotal Cloud Foundry) \_:**

By using the Pivotal I deployed our application with help of Cloud Foundry inside AWS Cloud. I have used PCF to provide the underlying infrastructure that organizations need to facilitate continuous delivery of software updates, manage the application life cycle, and streamline the development, deployment, and scaling of web-based applications.

1. It is platforms that enable us to focus on cloud-native application development and put focus on developer experience. In every step like developing, testing it, moving it all the way into production and automating all those flows.
2. It is interoperable which means although new technology is changing over the years with growth of docker and Kubernetes. CF keeps the same developer experience to the user and have support for the Kubernetes underneath.
3. polyglot environment which means we can take an application written in a number of different languages running in different platform and CF will have built pack that will take them and automate all the building and deploying process.
4. It is open source and open governance model. There are third party services and listed in marketplace to make advantage of open service broker API.
5. VM – Virtual Machine is good but with container people are starting to make use of the Kubernetes to manage and orchestra.
6. As said, it focuses on the development, developer can truly focus on what matters development and developer assume that the infrastructure works. Serverless technology where developer focus only on writing function.

### **GCP\_:**

In GCP, I was pretty much dealing with Google compute engine for delivering the VM running on the Google’s innovative data center and this Google compute engine VM would be coming up with high performance persistence to deliver consistent performance.

For building the web applications I have used Google App Engine as a because it provides us with built in services and API’s such as NOSQL, data stores, user authentication API which is common to most of the applications.

It will scale automatically in response to the amount of traffic it receives.

I have also used Google Kubernetes engine for cluster management i.e. for running docker containers which basically use containers into the clusters, keep them healthy and manages them automatically based on the requirements we define such as CPU and memory.

I have created a private docker repository in GCP which is Google cloud container registry that works with popular continuous delivery systems. It basically runs on GCP to provide consistent uptime on an infrastructure.

**TDD/BDD**

TDD- Test Driven Development; I have hand-on experience in TDD testing methodology in the Agile Methodology in SDLC (Software Development Life Cycle). I had to write test cases for the code and make sure that its functionality works properly. This methodology helps to explore bugs or error very quickly and enhance the productivity.

BDD- Behavioral Driven Development; I also have good experience with the BDD testing methodology while working on the systems behaviors. It uses Given (User Credential)-When (Click Bottons)-Then (Showing Validation message ) approach to write the test cases. It is cost-effective technique, focus on how code or system should behave from customer and developer perspective’s, In SDLC, This approach help debug the errors in the latter stage of SDLC which could be expensive if not detected. It provides several ways to illustrate real-world scenarios and enable client – developer to collaborate for understanding the requirement.

## Testing Junit /Mockito/Cucumber/Selenium/Jest/Mocha

**Junit\_:**

1. Performed unit testing using the Junit (junit for writing and running the code and to check the flow of the execution of the code with different annotations.) and Mockito frameworks (I have used the Mockito for the effective unit testing of the java applications)
2. I am proficient in working with JUnit Test Framework features like: Fixtures, Test Suites, Test runners and Junit Classes.
3. I have experience with Junit testing using POJO class, Business Logic class and a test class
4. I have hand-on experience in integrating Junit Framework with tools such as: Eclipse, Ant and Maven
5. I have worked with Mockito as for the mocking Framework in unit testing of Java application

**Mockito\_:**

1. Mockito is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications.
2. Mockito is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing.

**Cucumber\_:**

1. Used the cucumber to write the acceptance test for the web applications as it allows the automation of functional validation in easily readable and understandable format to business analyst, Developers, Testers, etc.
2. Cucumber is an automation testing framework that acts as a bridge between the business and the technical language.
3. I have experience using cucumber tools for the BDD (Behavioral Driven Development) testing methodology using building tools like maven and gradle.
4. I have experience in writing production code from scenarios provided from the client to meet the requirement. I have worked on multiple threads to provide parallel executions using Junit and maven test execution plugin.

**Selenium\_:**

1. I have used the selenium to automate the testing across the various web browsers. By using the selenium Web driver, we can easily automate the browser testing.
2. It supports various browsers like Chrome, Mozilla, Safari, IE, etc.
3. As Selenium is framework for testing web application, I have experience in writing test scripts in many types of testing like: Acceptance, Performance, Functional, and BDD.
4. I have worked on some of selenium tools like: Selenium RC to run test against different browser on different OS and test web application with complex AJAX- based scenarios. Selenium Grid; to run Selenium RC scripts in multiple browsers and operating systems simultaneously.

**Jest vs Mocha: What are the differences?**

* Jest is an open-source testing framework developed by Facebook. Built into the popular create-react-app package, is faster and smoother to write idiomatic JavaScript tests. It has built.
* in mocking and assertion abilities while running tests in parallel, providing a smoother, faster test-run. One unique feature of Jest is it provides Snapshot testing for complete control over UI.
* Mocha provides developers with a base test framework, with options such as assertion, mocking, and spy libraries. It is one of the most flexible JavaScript testing libraries. The slight downside to Mocha is the additional setup and configuration required. Chai, the most popular open-source assertion library is used with Mocha.

**What is the Unit Test?**

* Unit Tests are conducted by developers and test the unit of code (aka module, component) he or she developed. It is a testing method by which individual units of source code are tested to determine if they are ready to use. It helps to reduce the cost of bug fixes since the bugs are identified during the early phases of the development lifecycle.

**What is Integration Test?**

* Integration testing is executed by testers and tests integration between software modules. It is a software testing technique where individual units of a program are combined and tested as a group. Test stubs and test drivers are used to assist in Integration Testing. Integration test is performed in two ways, they are a bottom-up method and the top-down method.

**Key Difference**

* Unit testing is a testing method by which individual units of source code are tested to determine if they are ready to use, whereas Integration testing checks integration between software modules.
* Unit Testing test each part of the program and shows that the individual parts are correct, whereas Integration Testing combines different modules in the application and test as a group to see they are working fine.
* Unit Testing starts with the module specification, while Integration Testing starts with interface specification.
* Unit Testing can be performed at any time, on the other hand, Integration Testing is performed after unit testing and before system testing.
* Unit Testing is executed by the developer, whereas Integration Testing is performed by the testing team.
* Unit Testing errors can be found easily, whereas Integration Testing it is difficult to find errors.
* Unit Testing is a kind of white box testing, whereas Integration Testing is a kind of black-box testing.

**JIRA\_:**

Jira Software is part of a family of products designed to help teams of all types manage work. Originally, Jira was designed as a bug and issue tracker. But today, Jira has evolved into a powerful work management tool for all kinds of use cases, from requirements and test case management to agile software development.

* I used JIRA for project management, text cases management and bug management.

**GIT\_:**

Git is an open-source distributed version control system.

• It is designed to handle minor to major projects with high speed and efficiency.

• It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

• GitHub is a Git repository hosting service. GitHub is a distributed version control tool that can manage a programmer's source code history.

GitHub is a cloud-based tool developed around the Git tool. A developer installs Git tool locally. GitHub is an online service to store code and push from the computer running the Git tool. Git focused on version control and code sharing. GitHub focused on centralized source code hosting. It is a command-line tool. It is administered through the web. It facilitates with a desktop interface called Git Gui. It also facilitates with a desktop interface called GitHub Gui. Git does not provide any user management feature. GitHub has a built-in user management feature. It has minimal tool configuration feature. It has a marketplace for tool configuration. Git SVN It's a distributed version control system. It's a Centralized version control system Git is an SCM (source code management). SVN is revision control. Git has a cloned repository. SVN does not have a cloned repository. The Git branches are familiar to work. The Git system helps in merging the files quickly and also assist in finding the unmerged ones. The SVN branches are a folder which exists in the repository. Some special commands are required for merging the branches. Git does not have a Global revision number. SVN has a Global revision number. Git has cryptographically hashed contents that protect the contents from repository corruption taking place due to network issues or disk failures. SVN does not have any cryptographically hashed contents. Git stored content as metadata. SVN stores content as files. Git has more content protection than SVN. SVN's content is less secure than Git. Linus Torvalds developed git for Linux kernel. CollabNet, Inc developed SVN. Git is distributed under GNU (General public license). SVN is distributed under the open-source license.

**Thymeleaf\_:**

Thymeleaf is a java Based library used to create web application.

* To deal with UI we need Thymeleaf because whenever output is generated by java code, we need someone, who displays this output on UI.
* We can use Thymeleaf templates to create a web application in Spring Boot. We have to follow some project structure.
* To use Thymeleaf library we have to add the dependency in the pom.xml file and update our project. I am using < Spring-boot-starter-thymeleaf > dependencies in my project.
* We have to write all the html files in the Templates folder and all other non-HTML files like CSS, JS and Image file in the Static folder.
* We no need to write any code to read this file in Spring-Boot because by default Spring Boot will look for this file in particular folder.

**Gradle \_:**

Gradle makes it easy to build common types of projects. I have used it in automate building, testing, deployment.

* In Java if we want to make any application then we use some libraries and frameworks.
* Dependency management automatically downloads these artifacts from a repository and makes them available to your application.
* Gradle provides its own implementation. It is compatible with existing dependency management infrastructures (such as Maven and Ivy).
* Gradle's ability to manage dependencies isn't limited to external libraries. As your project grows in size and complexity, you'll want to organize the code into modules with clearly defined responsibilities.
* Gradle provides powerful support for defining and organizing multi project builds, as well as modelling dependencies between projects.

**Ant\_:**

* A build tool is used to automate repetitive tasks during this process.
* This can be, for example, compiling source code, running software tests and creating files and documentation for the software deployment. Popular build tools in the Java space are Maven, Gradle and Apache Ant.

**Struts\_:**

Struts Framework Struts is an open-source framework that extends the Java Servlet API and employs a Model, View, Controller (MVC) architecture. It enables you to create maintainable, extensible, and flexible web applications based on standard technologies, such as JSP pages, JavaBeans, resource bundles, and XML.

**Agile Methodology\_:**

In software application development, agile software development (ASD) is a methodology for the creative process that anticipates the need for flexibility and applies a level of pragmatism into the delivery of the finished product. Agile software development focuses on keeping code simple, testing often, and delivering functional bits of the application as soon as they're ready. The goal of ASD is to build upon small client-approved parts as the project progresses, as opposed to delivering one large application at the end of the project.

**Thymeleaf:**

Thymeleaf is a java Based library used to create web application.

To deal with UI we need Thymeleaf.

We can use Thymeleaf templates to create a web application in Spring Boot. We have to follow some project structure.

To use Thymeleaf library we have to add the dependency in pom.xml file and update our project. I am using < Spring-boot-starter-thymeleaf > dependencies in my project.

We have to write all the html files in the Templates folder and all other non-HTML files like CSS, JS and Image file in Static folder.

We no need to write any code to read this files in Spring-Boot because by default Spring-Boot will look for this file in particular folder.

**THYMELEAF**

I have written html, CSS and nice bootstrap and converted those in to thyme leaf template to use its features.

Used spring specific features of thyme leaf to create User Interface

Used Spring security with Thymeleaf to secure some content from displaying.

**Servlet:**

A servlet is a Java programming language class. To create Dynamic Web Project, we need Servlet.

Each Servlet name should be unique. We have to create one Web.xml file in WEB-INF folder.

We are using 5 methods in Servlet like Inite, service, destroy (life cycle method).

HTTP Servlet is a grandchild of servlet class so by using HTTP Servlet we can achieve all the features of servlet class too.

The HTTP Servlet class provides methods, such as doGet and doPost, for handling HTTP-specific services.

Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers.

The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets.

**RABBIT MQ\_:**

Worked with RabbitMQ and AMQP Java API for queues to build application as a SOA (Service oriented architecture)

To convert entire event flow as SOA based architecture, I used RabbitMQ as it deals with Queue

Implemented function to send And Receive AMQP messages on RabbitMQ asynchronously.

* open-source message broker software
* accepts messages from producers and delivers them to consumers.
* reduce loads and delivery times taken by web application servers.
* Steps for producer to send messages:
  + Create a connection to Queue.
  + Specify name of Queue to which we want to send our message and publish message to queue
  + Close channel and connection
* Steps for consumer to receive messages:
  + Create a connection to Queue.
  + Specify name of Queue to which we want to consume our message.
  + Since queue will send us messages asynchronously, we provide a call back in the form of an object that will buffer the messages until we’re ready to use them.
* Different exchange types are direct, topic, and fanout.
* Relationship between exchange and a queue is called a binding and uses binding key to define it.

<http://blogs.quovantis.com/rabbit-mq-working/>

**Web services\_:**

Web Services allows different applications to talk to each other and share data and services among themselves.

Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.

REST: Representational State Transfer. It is an architectural style.

SOAP: Simple Object Access Protocol

**Web Services:**

* In Layman terms, Services offered over the web can be called as a Web Service.
* Web Service is responsible for communication between two micro service application and making sure that the communication is smooth and successful.
* Web Services provides a standard protocol for communication.
* If two microservice applications are developed in two different languages then by using web service, those 2 applications can communicate with each other.
* Most widely used Web Services are REST and SOAP. Out of these two RESTS is more Flexible.

**Web services description:**

Web Services allows different applications to talk to each other and share data and services among themselves.

Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.

**REST\_ : Representational State Transfer. is an architectural style.**

**SOAP\_ : Simple Object Access Protocol**

* 1. SOAP is a protocol whereas REST is an architectural style.

SOAP server and client applications are tightly coupled and bind with the WSDL contract whereas there is no contract in REST web services and client.

Learning curve is easy for REST when compared to SOAP web services.

REST web services request and response types can be XML, JSON, text etc. whereas SOAP works with XML only.

JAX-RS is the Java API for REST web services whereas JAX-WS is the Java API for SOAP web services.

* SOAP is a standard protocol for creating web services.
* REST is an architectural style to create web services.
* SOAP uses WSDL to expose supported methods and technical details.
* REST exposes methods through URIs, there are no technical details.
* SOAP web services and client programs are bind with WSDL contract.
* REST doesn’t have any contract defined between server and client.
* SOAP web services and client are tightly coupled with contract.
* REST web services are loosely coupled.
* SOAP supports XML data format only.
* REST supports any data type such as XML, JSON, image etc.
* SOAP web services are hard to maintain, any change in WSDL contract requires us to create client stubs again and then make changes to client code.
* REST web services are easy to maintain when compared to SOAP, a new method can be added without any change at client side for existing resources.
* SOAP web services can be tested through programs or software such as Soap UI.
* SOAP is acronym for Simple Object Access Protocol.
* REST is acronym for representational State Transfer.

## **Testing Junit /Mockito/Cucumber/Selenium/Jest/Mocha**

**Junit\_:**

* Performed unit testing using the Junit (junit for writing and running the code and to check the flow of the execution of the code with different annotations.) and Mockito frameworks (I have used the Mockito for the effective unit testing of the java applications)
* I am proficient in working with JUnit Test Framework features like: Fixtures, Test Suites, Test runners and Junit Classes.
* I have experience with Junit testing using POJO class, Business Logic class and a test class
* I have hand-on experience in integrating Junit Framework with tools such as: Eclipse, Ant and Maven
* I have worked with Mockito as for the mocking Framework in unit testing of Java application

**Mockito\_:**

* Mockito is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications.
* Mockito is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing.

**Cucumber\_:**

* Used the cucumber to write the acceptance test for the web applications as it allows the automation of functional validation in easily readable and understandable format to business analyst, Developers, Testers, etc.
* Cucumber is an automation testing framework that acts as a bridge between the business and the technical language.
* I have experience using cucumber tools for the BDD (Behavioral Driven Development) testing methodology using building tools like maven and Gradle.
* I have experience in writing production code from scenarios provided from the client to meet the requirement. I have worked on multiple threads to provide parallel executions using Junit and maven test execution plugin.

**Selenium\_:**

* I have used the selenium to automate the testing across the various web browsers. By using the selenium Web driver, we can easily automate the browser testing.
* It supports various browsers like Chrome, Mozilla, Safari, IE, etc.
* As Selenium is framework for testing web application, I have experience in writing test scripts in many types of testing like: Acceptance, Performance, Functional, and BDD.
* I have worked on some of selenium tools like: Selenium RC to run test against different browser on different OS and test web application with complex AJAX- based scenarios. Selenium Grid; to run Selenium RC scripts in multiple browsers and operating systems simultaneously.

**Jest vs Mocha: What are the differences?**

* Jest is an open-source testing framework developed by Facebook. Built into the popular create-react-app package, is faster and smoother to write idiomatic JavaScript tests.
* It has built in mocking and assertion abilities while running tests in parallel, providing a smoother, faster test-run. One unique feature of Jest is it provides Snapshot testing for complete control over UI.
* Mocha provides developers with a base test framework, with options such as assertion, mocking, and spy libraries. It is one of the most flexible JavaScript testing libraries. The slight downside to Mocha is the additional setup and configuration required. Chai, the most popular open-source assertion library is used with Mocha.

**What is the Unit Test?**

* Unit Tests are conducted by developers and test the unit of code (aka module, component) he or she developed. It is a testing method by which individual units of source code are tested to determine if they are ready to use. It helps to reduce the cost of bug fixes since the bugs are identified during the early phases of the development lifecycle.

**What is Integration Test?**

* Integration testing is executed by testers and tests integration between software modules. It is a software testing technique where individual units of a program are combined and tested as a group. Test stubs and test drivers are used to assist in Integration Testing. Integration test is performed in two ways, they are a bottom-up method and the top-down method.

**Key Difference**

* Unit testing is a testing method by which individual units of source code are tested to determine if they are ready to use, whereas Integration testing checks integration between software modules.
* Unit Testing test each part of the program and shows that the individual parts are correct, whereas Integration Testing combines different modules in the application and test as a group to see they are working fine.
* Unit Testing starts with the module specification, while Integration Testing starts with interface specification.
* Unit Testing can be performed at any time, on the other hand, Integration Testing is performed after unit testing and before system testing.
* Unit Testing is executed by the developer, whereas Integration Testing is performed by the testing team.
* Unit Testing errors can be found easily, whereas Integration Testing it is difficult to find errors.
* Unit Testing is a kind of white box testing, whereas Integration Testing is a kind of black-box testing.

**JIRA:**

Jira Software is part of a family of products designed to help teams of all types manage work. Originally, Jira was designed as a bug and issue tracker. But today, Jira has evolved into a powerful work management tool for all kinds of use cases, from requirements and test case management to agile software development.

* I used JIRA for project management, text cases management and bug management.

**GIT:**

Git is an open source distributed version control system.

• It is designed to handle minor to major projects with high speed and efficiency.

• It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

• GitHub is a Git repository hosting service. GitHub is a distributed version control tool that can manage a programmer's source code history.

GitHub is a cloud-based tool developed around the Git tool. A developer installs Git tool locally. **GitHub is an online service to store code and push from the computer running the Git tool.** Git focused on version control and code sharing. GitHub focused on centralized source code hosting. It is a command-line tool. It is administered through the web. It facilitates with a desktop interface called Git Gui. It also facilitates with a desktop interface called GitHub Gui. Git does not provide any user management feature. GitHub has a built-in user management feature. It has minimal tool configuration feature. It has a marketplace for tool configuration. Git SVN It's a distributed version control system. It's a Centralized version control system Git is an SCM (source code management). SVN is revision control. Git has a cloned repository. SVN does not have a cloned repository. The Git branches are familiar to work. The Git system helps in merging the files quickly and also assist in finding the unmerged ones. The SVN branches are a folder which exists in the repository. Some special commands are required for merging the branches. Git does not have a Global revision number. SVN has a Global revision number. Git has cryptographically hashed contents that protect the contents from repository corruption taking place due to network issues or disk failures. SVN does not have any cryptographically hashed contents. Git stored content as metadata. SVN stores content as files. Git has more content protection than SVN. SVN's content is less secure than Git. Linus Torvalds developed git for Linux kernel. CollabNet, Inc developed SVN. Git is distributed under GNU (General public license). SVN is distributed under the open-source license.

**Advantages of SVN over GIT**

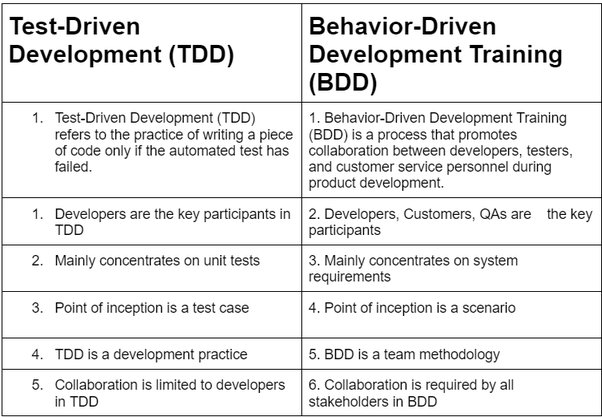
* Git repositories can't handle large binary files.
* SVN repositories can handle large binary files, in addition to code.
* Storing large binary files in SVN would take up less space than in Git.

**Kafka\_:**

1. Worked in event streaming for real-time data capturing data from database and software applications.

Kafka is primarily used **to build real-time streaming data pipelines and applications that adapt to the data streams**. It combines messaging, storage, and stream processing to allow storage and analysis of both historical and real-time data.

* Worked with **Kafka** and created topics and stored the data into MongoDB for storing Big Data using **Microservices**.
* Experience in handling messaging services with **Apache Kafka**
* Implemented Spring boot microservices to process the messages into the **Kafka** cluster setup.
* Improved the performance of the **Kafka** cluster by fine tuning the Kafka Configurations at producer, consumer, and broker level.
* Deployed **Kafka** manager for getting better insights into our Kafka clusters.
* Fixed **Kafka** related Production issues across multiple clusters.
* Integrated General Liability and Umbrella liability applications through **Apache** **Kafka** broker JMS.
* Managing Messaging topics using **Apache** **Kafka** messaging broker
* Designed and developed custom message adapter components using **Apache Kafka** that allowed message to travel through **Microservices** to base on **RESTful API** utilizing **Spring Boot** with **Spring MVC**.
* Used **Kafka** to load the data on Hadoop File system and move the same data to **MongoDB** database.

**TDD vs BDD\_:**

**Thymeleaf \_:**

Thymeleaf is a java Based library used to create web application.

* To deal with UI we need Thymeleaf because whenever output is generated by java code, we need someone, who displays this output on UI.
* We can use Thymeleaf templates to create a web application in Spring Boot. We have to follow some project structure.
* To use Thymeleaf library we have to add the dependency in the pom.xml file and update our project. I am using < Spring-boot-starter-thymeleaf > dependencies in my project.
* We have to write all the html files in the Templates folder and all other non-HTML files like CSS, JS and Image file in the Static folder.
* We no need to write any code to read this file in Spring-Boot because by default SpringBoot will look for this file in particular folder.

Gradle\_:

Gradle makes it easy to build common types of projects. I have used it in automate building, testing, deployment.

* In Java if we want to make any application then we use some libraries and frameworks.
* Dependency management automatically downloads these artifacts from a repository and makes them available to your application.
* Gradle provides its own implementation. It is compatible with existing dependency management infrastructures (such as Maven and Ivy).
* Gradle's ability to manage dependencies isn't limited to external libraries. As your project grows in size and complexity, you'll want to organize the code into modules with clearly defined responsibilities.
* Gradle provides powerful support for defining and organizing multi project builds, as well as modelling dependencies between projects.

Ant:

* A build tool is used to automate repetitive tasks during this process.
* This can be, for example, compiling source code, running software tests and creating files and documentation for the software deployment. Popular build tools in the Java space are Maven, Gradle and Apache Ant.

Struts:

Struts Framework Struts is an open-source framework that extends the Java Servlet API and employs a Model, View, Controller (MVC) architecture. It enables you to create maintainable, extensible, and flexible web applications based on standard technologies, such as JSP pages, JavaBeans, resource bundles, and XML.

Agile Methodology\_:

In software application development, agile software development (ASD) is a methodology for the creative process that anticipates the need for flexibility and applies a level of pragmatism into the delivery of the finished product. Agile software development focuses on keeping code simple, testing often, and delivering functional bits of the application as soon as they're ready. The goal of ASD is to build upon small client-approved parts as the project progresses, as opposed to delivering one large application at the end of the project.

**Spring:**

The Spring Framework is an open-source application framework that provides infrastructure support for developing Java applications.

Spring enables you to build applications from “plain old Java objects” (POJOs) and to apply enterprise services non-invasively to POJOs.

The Spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc.

**Spring Boot**

* The Spring boot is an open-source Java based framework used to create microservices.
* It is developed by pivotal team and is used to build standalone and production grade spring application that you can just run.
* You can get started with minimum configurations without the need for an entire Spring configuration setup.

**Difference between Struct and Spring**

|  |  |
| --- | --- |
| Spring | Struct |
| It is light weight framework | It is heavy weight framework |
| It does not support tag library | It supports tag library directives |
| It has loosely coupled modules | It has tightly coupled module |
| It is integrated with ORM technologies using which lesser code is required after and before the main logic | It supports manual coding |

**Why do we use MVN clean install?**

* It is used to remove all the target folder and all the intermediate results.
* When you build with Maven, the goal is to create some artifact(s) for deployment, this can be a jar for a library, a war for a web application or something else altogether.

**Hibernate**

* Hibernate is a java framework that simplifies the development of java application to interact with the database.
* It is open-source light weight object relational mapping tool.
* Hibernates primary feature is mapping from Java classes to database tables, and mapping from Java data types to SQL data types.
* Hibernate also provides data query and retrieval facilities. It generates SQL calls and relieves the developer from the manual handling and object conversion of the result set.

**Angular**

* **Angular is a JavaScript binding framework which binds the HTML UI and JavaScript Model. This helps you to reduce your effort on writing those lengthy lines of code for binding.**
* **It helps to implement MVW (Model view whatever) architecture in your web application.**
* **Also adding more to it, it also helps in building single page application by using the concept of routing, it has lot of other features such as routing, HTTP, DI, input output which you can create your own controls and so on.**
* The Angular CLI (command-line interface tool) that you use to initialize, develop, scaffold, and maintain Angular applications directly from a command shell.
* (Updated Angular is much more efficient compared to the older version of Angular, especially, the core functionality was moved to different modules. That’s why it becomes so much fast and smooth compared to the older one.

**Angular 7**

* Released in October 2018
* This is a major release and expanding to the entire platform including-  
  — Core framework,  
  — Angular Material,  
  — CLI
* CLI Prompts: The CLI will now prompt users as when running common commands likeng new or ng add [@angular/material](http://twitter.com/angular/material) with the intend of getting aid for building a new project using SCSS.
* Added a new interface — UrlSegment[] to CanLoad interface
* Added a new interface — DoBootstrap interface
* Angular 7 added a new compiler — Compatibility Compiler (ngcc)
* Introduce a new Pipe called — KeyValuePipe
* Angular 7 now supporting to TypeScript 2.9.

**Angular 8**

* Releasing March/April 2019
* Being smaller, faster and easier to use and it will be making Angular developers life easier.
* Added Support for TypeScript 3.2
* Added a Navigation Type Available during Navigation in the Router
* Added pathParamsOrQueryParamsChange mode for runGuardsAndResolvers in the Router
* Allow passing state to routerLink Directives in the Router
* Allow passing state to NavigationExtras in the Router
* Restore the whole object when navigating back to a page managed by Angular Router
* Added support for SASS

**AngularJS**

* AngularJS is JavaScript-based
* AngularJS is a JavaScript open-source front-end framework that is mainly used to develop single-page web applications (SPAs).
* It is a continuously growing and expanding framework which provides better ways for developing web applications. It can be freely used and changed by anyone.
* It extends HTML attributes with Directives, and data is bound with HTML.

**Angular Vs AngularJS**

|  |  |  |
| --- | --- | --- |
|  | **Angular JS (1.x)** | **Angular (2,4, 5, 6, 7, 8 and 9)** |
| **Language** | **JavaScript** | **Typescript** |
| **Architecture** | **Controller based architecture** | **Component based architecture** |
| **Mobile Compliant** | **Not mobile compliant** | **It is mobile compliant** |
| **CLI (command line argument)** | **No** | **Yes** |
| **Lazy loading** | **No** | **Yes** |
| **SEO (search engine optimization)** | **No** | **Yes** |
| **Server side** | **No** | **Yes** |

**Oracle DB**

An Oracle database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management

| Oracle 11g | Oracle 12c |
| --- | --- |
| It was released in 2008 and has no pluggable database | It is High performance RDbMS which is released in 2014. It is pluggable database. |
| We can't set primary key to raise automatically | We can set primary key to rise automatically. |
| We can't store Json directly to the column as Oracle 11g doesn't have JSON column type. | Oracle has introduced JSON column type. |
| Oracle 11g doesn't have cloud services | It provides Oracle cloud services. |
| No in-memory capabilities | It has in-memory capabilities. |

**What is difference between oracle and SQL**

Oracle supports many “Schemas” with the instance whereas SQL server offers “Schemas” within each user database. Oracle allows database, full, file-level, incremental & differential backups on the other hand, SQL server allows full, partial, and incremental backups.

**Microservices**

* Microservices are a set of software applications written in the Java programming language, designed for limited scope that work with each other to form a bigger solution.
* Each microservice, as the name implies, has minimal capabilities for the sake of creating a very modularized overall architecture.
* A microservices architecture is analogous to a manufacturing assembly line, where each microservice is like a station in the assembly line. Just as each station is responsible for one specific task, the same holds true for microservices.
* Each station and microservice are “experts” in their given responsibilities, thus promoting efficiency, consistency, and quality in the workflow and the outputs.

**Why Microservices?**

Microservices are increasingly used in the development world as developers work to create larger, more complex applications that are better developed and managed as a combination of smaller services that work cohesively together for larger, application-wide functionality.

Using microservices worked with **Kafka** and created topics and stored the data into MongoDB for storing Big Data.

**Web Services**

* A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. (XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language—Java can talk with Perl; Windows applications can talk with Unix applications.)
* Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web based.
* Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
* Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.

**REST**

* Restful Web Services is a stateless client-server architecture where web services are resources and can be identified by their URIs.
* REST Client applications can use HTTP GET/POST methods to invoke Restful web services. REST doesn’t specify any specific protocol to use, but in almost all cases it’s used over HTTP/HTTPS.
* When compared to SOAP web services, these are lightweight and doesn’t follow any standard.
* We can use XML, JSON, text, or any other type of data for request and response.

**JUnit**

JUnit is a unit testing open-source framework for the Java programming language. Java Developers use this framework **to write and execute automated tests**. In Java, there are test cases that must be re-executed every time a new code is added. This is done to make sure that nothing in the code is broken.

* Experience in writing unit test cases using **JUnit Framework.**
* Using **JUnit** test in Eclipse, developed Unit test cases and test suite**.**
* Performed Unit testing of the java modules using **JUnit**

**JSF**

JavaServer Faces (JSF) is a new standard Java framework for **building Web applications**. It simplifies development by providing a component-centric approach to developing Java Web user interfaces.

**MVC**

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used **to separate application's concerns**.

Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.

**View** - View represents the visualization of the data that model contains.

Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

ORM

An object-relational mapper provides an object-oriented layer between relational databases and object-oriented programming languages without having to write SQL queries.

**In** short, we are trying to interact with our database using our language of choice instead of SQL.

**Advantage of ORM is** You get to write in the language you are already using anyway. If we’re being honest, we probably aren’t the greatest at writing SQL statements. SQL is a ridiculously powerful language, but most of us don’t write in it often. We do, however, tend to be much more fluent in one language or another and being able to leverage that fluency is awesome!

**Thymeleaf**

* Thymeleaf is a java Based library used to create web application.
* To deal with UI we need Thymeleaf because whenever output is generated by java code, we need someone, who displays this output on UI.
* We can use Thymeleaf templates to create a web application in Spring Boot. We have to follow some project structure.
* To use Thymeleaf library we must add the dependency in the pom.xml file and update our project. I am using < Spring-boot-starter-thymeleaf > dependencies in my project.
* We must write all the html files in the Templates folder and all other non-HTML files like CSS, JS and Image file in the Static folder.
* We no need to write any code to read this file in Spring-Boot because by default Spring-Boot will look for this file in particular folder.

**Servlet**

* A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model.
* Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.
* The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. All servlets must implement the Servlet interface, which defines life-cycle methods.
* When implementing a generic service, you can use or extend the GenericServlet class provided with the Java Servlet API.
* The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services.

**Jenkins**

* Jenkins is an open-source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in the Java programming language. It is used to implement CI/CD workflows, called pipelines.
* I’m using Docker to provide our restful API to enable a remote API on the docker container to build and modify the docker image in the harbor.
* I’ve integrated maven with Jenkins. So, I can build and deploy the docker images without Jenkins and docker plugins.
* So once my docker image is ready then I can push that image to the docker hub using docker maven plugin. This way I have been working on docker maven plugin but avoiding docker Jenkins plugin.

**Gradle**

* In Java if we want to make any application then we use some libraries and frameworks.
* Dependency management automatically downloads these artifacts from a repository and makes them available to your application.
* Gradle provides its own implementation. It is compatible with existing dependency management infrastructures (such as Maven and Ivy).
* Gradle's ability to manage dependencies isn't limited to external libraries. As your project grows and complexity, you'll want to organize the code into modules with clearly defined responsibilities.
* Gradle provides powerful support for defining and organizing multi project builds, as well as modelling dependencies between projects.

**Swaggers**

**Swaggers is an open source set of rules, specifications and tools for developing and describing RESTful APIs. The swaggers framework allows developers to create interactive,machined and human-readable API documentation.**

**GCP**

**Google cloud platform is a suite of public cloud computing services offered by google. The platform includes a range of hosted services for compute, storage and the application development that runs on a google hardware, Google cloud services can be accessed by software developers, cloud administrators and other enterprises it professionals over the public internet or the dedicated network connection.**

**React**

* React is an open-source user interface library that is used for building user interfaces specifically for single-page applications.
* It's used for handling the view layer for web and mobile apps.
* React also allows us to create reusable UI components

**Junit**

* It is a widely used testing framework along with Java Programming Language.
* I used this automation framework for both unit testing and UI testing.
* It helps us define the flow of execution of our code with different Annotations.
* It also gives test runners for running tests effectively

**Web Logic**

* WebLogic servers are called middleware or application servers. They act as a middle layer between the front-end user interface and the backend database.
* WebLogic servers host the applications, or the business logic required to perform any action.
* For example, when you click a button on webpage, the request goes from your browser to the WebLogic server, the application deployed decides what action is to be taken for the request, queries database and gives the response on your screen.
* WebLogic servers are owned by Oracle. Its rivals are JBoss servers by RedHat and WebSphere by IBM.

**Ant**

* A build tool is used to automate repetitive tasks during this process.
* This can be, for example, compiling source code, running software tests and creating files and documentation for the software deployment. Popular build tools in the Java space are Maven, Gradle and Apache Ant.

**Mockito**

* Mockito is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications.
* Mockito is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing.

**SOAP**

* It is a protocol of how web services talk to each other or talk to client applications that invoke them.
* SOAP was developed as an intermediate language so that applications built on various programming languages could talk easily to each other and avoid the extreme development effort

**SOAP vs REST**

SOAP is designed to break traditional monolithic applications down into a multi-component, distributed form without losing security and control. In contrast, REST is a model of distributed computing interaction based on the HTTP protocol and the way that web servers support clients. REST over HTTP is almost always the basis for modern microservices development and communications. RESTful APIs uses HTTP requests to GET, PUT, POST and DELETE data.

REST/HTTP is simple, flexible, lightweight, and offers little beyond a way of exchanging information. SOAP can ride on HTTP as well, but it connects the elements of a complex set of distributed computing tools C as well as application components, and this forms a part of a total service-oriented framework.

**JSON (Java script object notation)**

* By default, you are returning an object, you will get object in JSON format.
* When we want to interact with other application, like .net, python or any other language we need one common language communication, so we use JSON format or XML format for this communication.
* Give JSON response to .NET guy so he will convert JSON to his student object, and he will be giving to the business logic. He will give request back in JSON format so my java code will be taking JSON and covert into my request related object.

**AWS**

* The AWS SDK for Java simpliﬁes use of AWS Services by providing a set of libraries that are consistent and familiar for Java developers.
* It provides support for API lifecycle consideration such as credential management, retries, data marshalling, and serialization.

**Difference between in inheritance and polymorphism**

|  |  |
| --- | --- |
| **Inheritance** | **Polymorphism** |
| Inheritance is one in which a new class is created (derived class) that inherits the features from the already existing class (Base class). | Whereas polymorphism is that which can be defined in multiple forms. |
| It is basically applied to classes. | Whereas it is basically applied to functions or methods. |
| Inheritance supports the concept of reusability and reduces code length in object-oriented programming. | Polymorphism allows the object to decide which form of the function to implement at compile-time (overloading) as well as run-time (overriding). |
| Inheritance can be single, hybrid, multiple, hierarchical, and multilevel inheritance. | Whereas it can be compiled-time polymorphism (overload) as well as run-time polymorphism (overriding). |
| It is used in pattern designing. | While it is also used in pattern designing. |

**Difference between Array List and LinkedList**

|  |  |
| --- | --- |
| Array list | LinkedList |
| ArrayList internally uses a dynamic array to store the elements. | LinkedList internally uses a doubly linked list to store the elements. |
| Manipulation with ArrayList is slow because it internally uses an array. If any element is removed from the array, all the other elements are shifted in memory. | Manipulation with LinkedList is faster than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory. |
| An ArrayList class can act as a list only because it implements List only. | LinkedList class can act as a list and queue both because it implements List and Deque interfaces. |
| ArrayList is better for storing and accessing data. | LinkedList is better for manipulating data. |
| The memory location for the elements of an ArrayList is contiguous. | The location for the elements of a linked list is not contagious. |
| Generally, when an ArrayList is initialized, a default capacity of 10 is assigned to the ArrayList. | There is no case of default capacity in a LinkedList. In LinkedList, an empty list is created when a LinkedList is initialized. |
| To be precise, an ArrayList is a resizable array. | LinkedList implements the doubly linked list of the list interface. |

Design Patterns used in Spring Framework

The four most common design patterns used in the Spring Framework:

1. Singleton pattern: **The singleton pattern is a mechanism that ensures only one instance of an object exists per application.** This pattern can be useful when managing shared resources or providing cross-cutting services, such as logging.
2. Factory Method pattern: **The factory method pattern entails a factory class with an abstract method for creating the desired object.**
3. Proxy pattern: Proxies are a handy tool in our digital world, and we use them very often outside of software. In code, **the proxy pattern is a technique that allows one object the proxy to control access to another object the subject or service.**
4. Template pattern: In many frameworks, a significant portion of the code is boilerplate code. For example, when executing a query on a database, the same series of steps must be completed: Establish a connection, Execute query, Perform cleanup, Close the connection. These steps are an ideal scenario for the template method pattern

Difference between Waterfall and Agile

* Waterfall is a Liner Sequential Life Cycle Model whereas Agile is a continuous iteration of development and testing in the software development process.
* Agile methodology is known for its flexibility whereas Waterfall is a structured software development methodology.
* Agile follows an incremental approach whereas the Waterfall is a sequential design process.
* Agile performs testing concurrently with software development whereas in Waterfall methodology testing comes after the “Build” phase.
* Agile allows changes in project development requirement whereas Waterfall has no scope of changing the requirements once the project development starts.

ANT vs Maven vs Gradle

|  |  |  |
| --- | --- | --- |
| Ant | Maven | Gradel |
| Released in 2000 | Released in 2004 | Released in 2012 |
| Easy to learn and implement | Added new features on top of Ant and continued with XML | Uses DSL in place of XML, the resulted shorter and clearer script |

|  |  |  |
| --- | --- | --- |
| **Ant** | | **Maven** |
| Ant **doesn't has formal conventions**, so we need to provide information of the project structure in build.xml file. | | Maven **has a convention** to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file. |
| Ant is **procedural**, you need to provide information about what to do and when to do through code. You need to provide order. | | Maven is **declarative**, everything you define in the pom.xml file. |
| There is **no life cycle** in Ant. | | There is **life cycle** in Maven. |
| It is **a tool** box. | | It is **a framework**. |
| It is **mainly a build tool**. | | It is **mainly a project management tool**. |
| The ant scripts are **not reusable**. | | The maven plugins are **reusable**. |
| It is **less preferred** than Maven. | | It is **more preferred** than Ant. |
| Basis | Gradle | Maven |
| Based on | Gradle is based on developing domain-specific language projects. | Maven is based on developing pure Java language-based software. |
| Configuration | It uses a Groovy-based Domain-specific language (DSL) for creating project structure. | It uses Extensible Markup Language (XML) for creating project structure. |
| Focuses on | Developing applications by adding new features to them. | Developing applications in each time limit. |
| Performance | It performs better than maven as it optimized for tracking only current running task. | It does not create local temporary files during software creation and is hence – slower. |
| Java Compilation | It avoids compilation. | It is necessary to compile. |
| Usability | It is a new tool, which requires users to spend a lot of time to get used to it. | This tool is a known tool for many users and is easily available. |
| Customization | This tool is highly customizable as it supports a variety of IDE’s. | This tool serves a limited number of developers and is not that customizable. |
| Languages supported | It supports software development in Java, C, C++, and Groovy. | It supports software development in Java, Scala, C#, and Ruby. |

Stringbuffer vs string builder

|  |  |  |
| --- | --- | --- |
| **No.** | **StringBuffer** | **StringBuilder** |
| 1) | StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. | StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously. |
| 2) | StringBuffer is *less efficient* than StringBuilder. | StringBuilder is *more efficient* than StringBuffer. |
| 3) | StringBuffer was introduced in Java 1.0 | StringBuilder was introduced in Java 1.5 |

SQL vs PL/SQL

| **BASIS FOR COMPARISON** | **SQL** | **PL/SQL** |
| --- | --- | --- |
| Basic | In SQL you can execute a single query or a command at a time. | In PL/SQL you can execute a block of code at a time. |
| Full form | Structured Query Language | Procedural Language, extension of SQL. |
| Purpose | It is like a source of data that is to be displayed. | It is language that creates an application that display's the data acquired by SQL. |
| Writes | In SQL you can write queries and command using DDL, DML statements. | In PL/SQL you can write block of code that has procedures, functions, packages or variables, etc. |
| Use | Using SQL, you can retrieve, modify, add, delete, or manipulate the data in the database. | Using PL/SQL, you can create applications or server pages that display's the information obtained from SQL in a proper format. |
| Embed | You can embed SQL statement in PL/SQL. | You cannot embed PL/SQL in SQL |

**SQL vs NoSQL**

* SQL databases are relational, NoSQL databases are non-relational.
* SQL databases use structured query language and have a predefined schema. NoSQL databases have dynamic schemas for unstructured data.
* SQL databases are vertically scalable, while NoSQL databases are horizontally scalable.
* SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores.
* SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data like documents or JSON.

**Access Modifiers**

The four access modifiers are public, private, protected and default. We use them to determine who has access to information or keywords within a class."

Dependency injection (DI)

* It is the concept in which objects get other required objects from outside.
* DI can be implemented in any programming language.
* The general concept behind dependency injection is called Inversion of Control.
* A Java class has a dependency on another class if it uses an instance of this class.

Final vs finally vs finalizes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. no.** | **Key** | **Final** | **finally** | **finalize** |
| 1. | Definition | final is the keyword and access modifier which is used to apply restrictions on a class, method, or variable. | finally is the block in Java Exception Handling to execute the important code whether the exception occurs or not. | finalize is the method in Java which is used to perform clean up processing just before object is garbage collected. |
| 2. | Applicable to | Final keyword is used with the classes, methods, and variables. | Finally block is always related to the try and catch block in exception handling. | finalize() method is used with the objects. |
| 3. | Functionality | (1) Once declared, final variable becomes constant and cannot be modified. (2) final method cannot be overridden by sub class. (3) final class cannot be inherited. | (1) finally block runs the important code even if exception occurs or not. (2) finally block cleans up all the resources used in try block | finalize method performs the cleaning activities with respect to the object before its destruction. |
| 4. | Execution | Final method is executed only when we call it. | Finally, block is executed as soon as the try-catch block is executed.  Its execution is not dependent on the exception. | finalize method is executed just before the object is destroyed. |

Overloading and Overriding

|  |  |
| --- | --- |
| **Method Overloading** | **Method Overriding** |
| 1) | Method overloading is used *to increase the readability* of the program. | Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class. |
| 2) | Method overloading is performed *within class*. | Method overriding occurs *in two classes* that have IS-A (inheritance) relationship. |
| 3) | In case of method overloading, *parameter must be different*. | In case of method overriding, *parameter must be same*. |
| 4) | Method overloading is the example of *compile time polymorphism*. | Method overriding is the example of *run time polymorphism*. |
| 5) | In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter. | *Return type must be same or covariant* in method overriding. |

Collection

* The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.
* Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.
* Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

Advantages of NodeJS

* High-performance for Real-time Applications
* Easy Scalability for Modern Applications
* Cost-effective with Full stack JS
* Community Support to Simplify Development
* Easy to Learn and Quick to Adapt
* Helps in building Cross-functional Teams
* Improves App Response Time and Boosts Performance
* Reduces Time-to-Market of your applications
* Extensibility to Meet Customized Requirements
* Reduces Loading Time by Quick Caching
* Helps in Building Cross-Platform Applications

**Spring IOC**

* Spring IoC Container is the core of Spring Framework.
* It **creates the objects, configures, and assembles their dependencies, manages their entire life cycle**.
* The Container uses Dependency Injection (DI) to manage the components that make up the application.

**Spring AOP**

Aspect-Oriented Programming is the full form for AOP, as its name suggests we use aspects in programming language one of the major use of aspects is **that it allows us to break our code into different modules and this process known as modularization.**

In AOP aspects enable us to implement crosscutting concerns like logging, transaction, etc.

AOP provides the way to dynamically add the cross-cutting concern before, after or around the actual logic using simple pluggable configurations.

It makes easy to maintain code in the present and future as well. You can add/remove concerns without recompiling complete source code simply by changing configuration files (if you are applying aspects suing XML configuration).

**Hibernate**

Hibernate **reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects**.

It relieves programmer from manual handling of persistent data, hence reducing the development time and maintenance cost.

**Lazy loading**

we can achieve **lazy loading**using hibernate. consider an example where there is a list of users in the user table. the identity proof documents uploaded by the users are stored in the identity\_proof table. the user has a ‘one to many’ relationships with the identity\_proof. in this case, the user is the parent class and identity\_proof is the child class. if you fetch the parent class, i.e the user, all the documents associated with the user will also be fetched. imagine the size of each document. as the number of documents increases, the size of data to be processed also increases, and hence it will slow up the application.

with hibernate, you can specify the fetch type for data as lazy. if you do so when you fetch a user, documents will not be fetched. you can fetch the documents where you want using hibernate’s initialize() method.

### Can you describe the difference between JDK and JVM?

|  |  |  |
| --- | --- | --- |
| **JDK** | **JRE** | **JVM** |
| The full form of JDK is Java Development Kit. | The full form of JRE is Java Runtime Environment. | The full form of JVM is Java Virtual Machine. |
| JDK is a software development kit to develop applications in Java. | It is a software bundle which provides Java class libraries with necessary components to run Java code. | JVM executes Java byte code and provides an environment for executing it. |
| JDK is platform dependent. | JRE is also platform dependent. | JVM is highly platform dependent. |
| It contains tools for developing, debugging, and monitoring java code. | It contains class libraries and other supporting files that JVM requires to execute the program. | Software development tools are not included in JVM. |
| It is the superset of JRE | It is the subset of JDK. | JVM is a subset of JRE. |
| The JDK enables developers to create Java programs that can be executed and run by the JRE and JVM. | The JRE is the part of Java that creates the JVM. | It is the Java platform component that executes source code. |
| JDK comes with the installer. | JRE only contain environment to execute source code. | JVM bundled in both software JDK and JRE. |

### Can you tell explain the difference between path and class path variables?

Class path is specific to Java executables, and it's used to locate class files, whereas path exists in an operating system and is used to locate executables."

### What is a final class in Java?

"This is a designation that says a class cannot be extended or overridden by a subclass. Declaring a class final helps prevent inheritance and keeps information secure."

### What is your favorite aspect of Java?

"There are so many things to love about Java, but I think my favorite thing is the community. People use Java all over the world, and many of them are doing innovative things with it. I just really like knowing this programming language because it allows me to interact with a lot of people and get insight and inspiration for different ways to use it."

**JIRA**

Jira **assigns a task and customizes the issues and subtasks in a whole team, also manages the workflow and track the progress**. It also helps you to change the permission for a particular task within your team. You can change or also edit the permission at any time during the project.

**What do you know about cloud computing platforms?**

Through cloud computing, we can store and process data on remote servers instead of local machines and servers. Using the cloud to manage software, databases, networking, servers, etc., makes accessibility faster, flexible, and more scalable. The resources that are stored on the ‘cloud’ are available on-demand and are delivered over the internet. In cloud computing, a resource is a service — Infrastructure as a Service, Platform as a Service, and Software as a Service.

FRONT END LANGUAGES --- **HTML, CSS, JAVA SCRIPT**.

Front End Frameworks and Libraries – **ANGULAR JS, REACT JS, J QUERY, SAAS, FLUTTER.**

Back End Languages -- **PHP, C++, JAVA, PYTHON, NODE.JS.**

Back End Frameworks – **EXPRESS, DJANGO, SPRING.**

### Can you tell explain the difference between path and class path variables?

Class path is specific to Java executables, and it's used to locate class files, whereas path exists in an operating system and is used to locate executables."

### What is a final class in Java?

"This is a designation that says a class cannot be extended or overridden by a subclass. Declaring a class final helps prevent inheritance and keeps information secure."

### What is your favorite aspect of Java?

"There are so many things to love about Java, but I think my favorite thing is the community. People use Java all over the world, and many of them are doing innovative things with it. I just really like knowing this programming language because it allows me to interact with a lot of people and get insight and inspiration for different ways to use it."

**How will you manage the user stories or tasks when you run into some production issues**?

I will keep my team posted about the said issues and in case something happens I will complete the task in the next sprint.

**Vendor questions?**

How many developers do you have working with my technology?

Can you estimate the length of my project?

What technologies do you use?

**Sprint Size ?**

**14Days**

**We used Spring Boot in our last project. Earlier days we were using Struts, later on traditional Spring MVC framework to build web applications. Since these two frameworks build applications as war files, we were in need of external web servers. Which is causing both application and server software maintenance. Furthermore we were having huge xml content for configuration.**

**In spring boot, it comes with embedded web servers. So we no need to install them separately. In terms of configuration also, we no need to have traditional XML configuration. We can use starter packages which are very easy like, plug and play.**

**In addition to these features, we can create distributed applications with the help of Spring Boot Cloud. There are many features like, configuration server, Eureka server and more.**

**2.**

**Our recent project includes building a physical cache for one of our datasource.**

**As part of the system, we used to communicate with a data source to fetch the data. Since this is billable, we build a cache with the help of Mongo DB.**

**In this Mongo DB, we used to store the data as a pair of request and response. So in a particular period of time when we are encountering a request with similar data, we get it from DB instead of interacting with data source. This cache avoided a decent amount of billing for duplicate requests.**

**3.**

**We were having around 6 developers in our team. My responsibilities includes interacting with our architects and client stakeholders as well. We do connect with our stakeholders to provide technical solutions for their business requirements.**

**I do involve in technical meetings with our architects who are working on various modules of our applications.**

**We discuss about the current architecture and areas of improvement. Most of the time, our discussions revolves around trimming down the APIs into multiple reusable components. Adding caches, maintaining distributed services etc.**