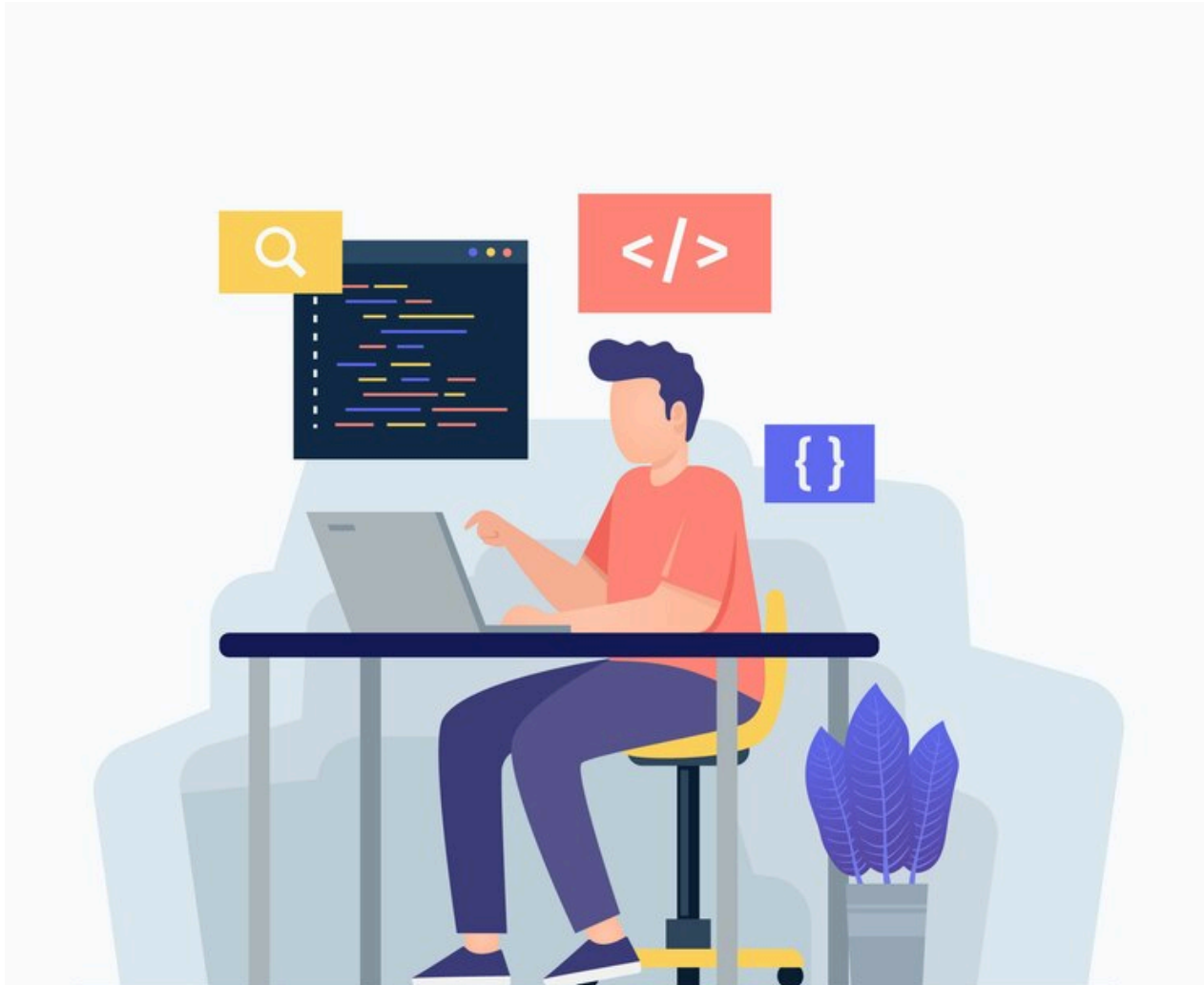


Java Full Stack Development

V.V.S.K Chaitanya

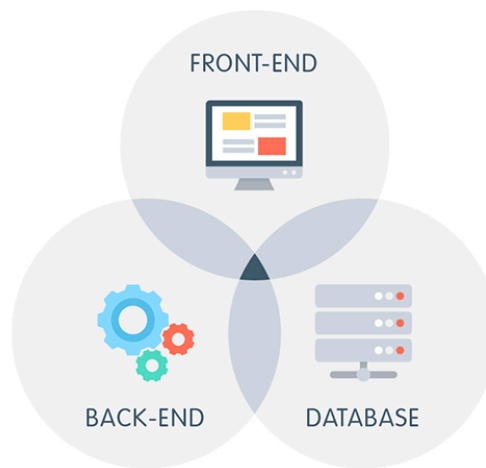


Overview

Full-stack web development is **Specialization** within Software Development.

It specifically focuses on building **Web Applications** that involve both front-end and back-end technologies to create functional and interactive web applications.

FULL-STACK DEVELOPMENT



“Full stack development is the process of designing, developing, building, testing, and deploying a complete web application from start to finish. It involves working with various technologies and tools, including front-end web development, back-end development, database and other integrations”

Breakdown

Full-stack developer works on all aspects of web development by possessing a wide range of skills, below is a brief overview of technologies, skills and responsibilities of full-stack web developer.

1. Front-End Development:

- **Languages:** HTML, CSS, JavaScript
- **Frameworks/Libraries:** React.js, Angular, Vue.js
- **Responsibilities:** Creating the visible parts of a website or web application that users interact with directly. This includes designing layouts, implementing responsive designs, and ensuring a seamless user experience.



The screenshot shows a code editor with two panels. The left panel is titled 'HTML' and contains the following code:

```
1 <div class="container">
2   <div class="tigre">
3     <div class="body">
4       <div class="cola"></div>
5       <div class="cola3"></div>
6       <div class="body7"></div>
7     <div class="body5">
8       <div class="ray"></div>
9     </div>
10    <div class="legs"></div>
11    <div class="ray3"></div>
12    <div class="ray5"></div>
13    <div class="legs3"></div>
14  </div>
```

The right panel is titled 'CSS' and contains the following code:

```
1 body{
2   background:black;
3 }
4 .container{
5   position:relative;
6   width:800px;
7   height:600px;
8   background:#214760;
9   margin:0px auto;
10 }
11 }
12 .tigre{
13   position:absolute;
14   margin:271px 350px
```



HTML5



CSS3



Javascript



Angular



React



Vue JS

2. Back-End Development:

- **Languages:** Java, Node.js, Python, Ruby, PHP.
- **Frameworks:** Spring (Java), Express.js (Node.js), Django (Python), Ruby on Rails (Ruby) , Laravel (PHP), etc.
- **Databases:** SQL (MySQL, PostgreSQL), NoSQL (MongoDB, CouchDB)
- **Responsibilities:** Handling server-side logic, databases, and ensuring the application's functionality, performance, and security. This involves server management, database integration, and API development.



Java



Node JS



.NET
Framework



Spring
Framework



Spring Boot

3. Database Management

- **SQL Databases:** Relational databases like MySQL, PostgreSQL
- **NoSQL Databases:** Document-based (MongoDB), Key-value stores (Redis), etc.
- **Responsibilities:** Storing, managing, and retrieving data efficiently and securely. Understanding different database types and choosing the appropriate one based on project requirements.



MongoDb



Postgre Sql



Elastic Search



Oracle



Mysql



Microsoft Sql
Server



Redis

Furthermore

4. Version Control / SCM:

- **Tools:** Git, GitHub, GitLab, Bitbucket
- **Responsibilities:** Tracking changes in code, collaborating with teams, and managing different versions of the project.

5. Deployment/DevOps:

- **Platforms:** AWS, Azure, GCP, etc.
- **Tools:** Jenkins, Docker, Kubernetes, etc.
- **Responsibilities:** Deploying and maintaining applications, automating processes, ensuring scalability, and monitoring performance.

6. Additional Skills:

- **Design Principles:** OOPS Concepts, DesignPatterns, High Level Design, SOLID principles, Twelve-Factor App methodology.
- **Testing:** Unit testing, integration testing, end-to-end testing (e.g., Jest, Mocha, Selenium)
- **Security:** Knowledge of web security practices, HTTPS, encryption, Authentication and Authorization and handling vulnerabilities
- **Responsive Design:** Ensuring the website works well across different devices and screen sizes
- **Soft Skills:** Problem-solving, communication, teamwork, and adaptability

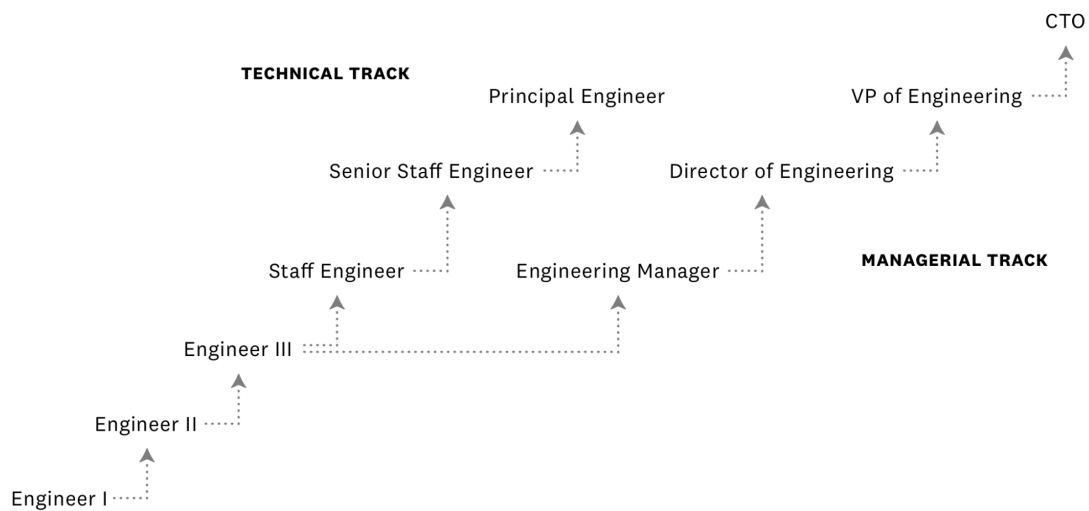
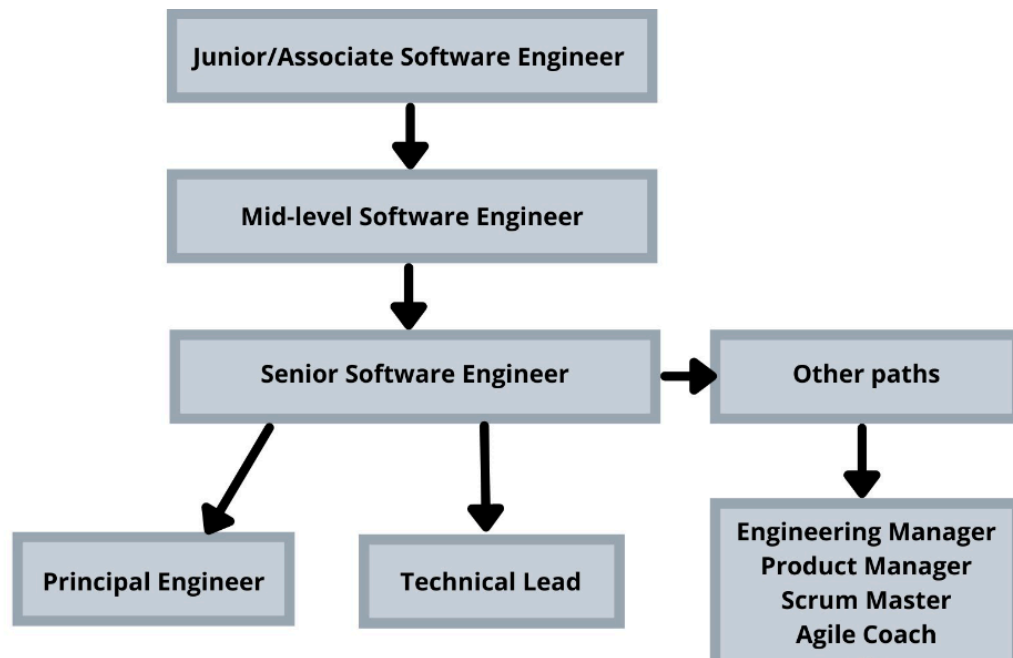
Advantages of Full Stack

- Learn **Java** (Back-end), **Angular** (Front end) and Develop E2E Application from scratch.
- Build **REST API, Microservices** using Spring Boot Framework.
- Build **Web UI, Single Page Applications** using Angular latest version.
- Full stack helps to understand the overall picture of web applications.
- Get Hands on **real time development** projects
- Enhance your skill set with **trending development frameworks**(Ex: Spring framework, Microservices, SPA, Testing Framework, Build and CI / CD Tools)
- Update Resume with multiple skills
- **High Requirement** of App development and App support.
- **Demand of Fullstack Developers** in current market
- Most Companies are looking out for **multiple tech stack** resources.

What you will learn from this course

- ★ Java Application Development
- ★ Building REST API and Microservices using Spring Boot
- ★ Dynamic Web Page Development using Front End Technologies
- ★ Development of Angular UI / Single Page Applications
- ★ Unit Testing / Integration testing / Automation Testing
- ★ Using Tools like Git, Maven, Gradle, Jenkins
- ★ Realtime Industry Software Development Experience
- ★ 2 - 3 Hands on Development Projects

Typical Software Engineer Career Path in most of Industry



Java Fullstack Course Syllabus

Core Java

- Introduction
 - ◆ Overview of Java and Setup of Java / JDK / IDE
- Java Basics
 - ◆ Java Structure, Syntax and Reserved Keywords
 - ◆ Variables and Data Types
 - ◆ Operators and Expressions
 - ◆ Array and String
 - ◆ Control Flow Statements
- OOPS Concepts
 - ◆ Class / Objects / Fields / Methods
 - ◆ Classification, Encapsulation, Abstraction, Inheritance, Polymorphism
 - ◆ Access Modifiers, Method Overloading, Overriding, Static Methods
- Exception Handling
 - ◆ Control flow of exceptions, Exception Hierarchy
 - ◆ Custom Exceptions
- File Handling
 - ◆ Read and write files using File API
- Multithreading
 - ◆ Multithreading in Java and its advantages
 - ◆ Thread Lifecycle, Creating Threads, Synchronization and best practices
- Java Collection Framework
 - ◆ List - ArrayList, LinkedList, Vector
 - ◆ Map - HashMap, TreeMap, LinkedHashMap
 - ◆ Set - HashSet, TreeSet, LinkedHashSet
 - ◆ Iterators and Comparators
 - ◆ Collections - Stream API

Database - SQL

- SQL Introduction
- Java Application Using JDBC
- JDBC Sessions / Query / Transaction / ResultSet
- CRUD Application using MySQL and JDBC

Backend - Spring Boot API Development

→ Spring Boot Intro

- ◆ Spring Framework
- ◆ Spring Boot vs Spring Framework
- ◆ Build Tools - Maven and Gradle
- ◆ Spring Boot Starter / Create Spring Boot Application

→ Spring Web

- ◆ Spring Web Module
- ◆ Spring Beans and Scope
- ◆ Web Annotations
- ◆ Creating First API
- ◆ Spring Boot Project Structure

→ REST API

- ◆ JSON Structure
- ◆ Building REST API
- ◆ Testing from Postman

→ Spring Data

- ◆ Spring Data Module
- ◆ JPA Annotations
- ◆ CRUD API
- ◆ Embedded H2 Database

→ Testing

- ◆ Unit Testing
- ◆ Junit5 & Mockito
- ◆ Integration Testing

→ Advanced Modules

- ◆ Logging
- ◆ Configuration
- ◆ Spring Security
- ◆ Spring Actuator
- ◆ Spring Kafka

Frontend - UI Foundations

→ HTML

- ◆ HTML Introduction
- ◆ HTML Elements (Headings / Paragraph / Lists / Tables / Href / Div)
- ◆ HTML Attributes (Id / name / class)
- ◆ HTML Form Elements (Input / Button / Select)
- ◆ Advanced Tags of HTML5
- ◆ Static Plain HTML Web Page

→ CSS

- ◆ CSS Intro
- ◆ Alignment Properties (Margin / Padding / Position / Height / Width)
- ◆ Visual Properties (Color / Font / Background / Border)
- ◆ Display (block / inline-block / flex)
- ◆ Inline / Internal / External Styling
- ◆ CSS Selectors
- ◆ Advanced Properties (Transform / Transition / Animate)
- ◆ Static Plain HTML & CSS Web Page

→ Javascript

- ◆ JS Intro
- ◆ JS Syntax
- ◆ Datatypes / Objects / Arrays / Array Methods
- ◆ JSON Representation of Data
- ◆ Conditional and Loop Statements
- ◆ JS Functions
- ◆ HTML DOM - Window / Document
- ◆ Chrome Dev Tools and JS Console
- ◆ DOM Manipulation
- ◆ Event Handling
- ◆ Dynamic Web page (HTML / CSS / JS)

Frontend - UI Advanced

→ Bootstrap

- ◆ BS5 Intro
- ◆ Containers and Grid Basic
- ◆ Typography and Colors
- ◆ Navigation / Tables / Cards / Carousel
- ◆ Form Elements - Input / Button / Dropdown
- ◆ Modals and Tooltips
- ◆ Progress Spinner / Progress Bars
- ◆ Bootstrap Web Pages

→ W3 CSS

- ◆ W3 CSS Intro
- ◆ Containers and Panels
- ◆ Layout and Colors
- ◆ Navigation / Tables / Cards / Slideshow
- ◆ Google Fonts
- ◆ Google Material Icons

→ Responsive Design

- ◆ Responsive CSS Query
- ◆ Responsive Bootstrap classes
- ◆ Mobile Friendly Web Page

→ JQUERY

- ◆ JQuery Intro
- ◆ Syntax and Selectors
- ◆ DOM Manipulation using jQuery
- ◆ jQuery Event Handling
- ◆ jQuery AJAX calls
- ◆ Dynamic Web Pages using AJAX

Frontend - Angular Development

→ Introduction to Angular

- ◆ Client Side Scripting and Templating
- ◆ Introduction to Single Page Applications
- ◆ Angular Architecture
- ◆ Angular Local Setup

→ Components and Templates

- ◆ Components and Templates
- ◆ Two way Data Binding and Interpolation

→ Angular Directives and Pipes

- ◆ Angular directives for manipulating the DOM
- ◆ Use of pipes for data transformation

→ Angular Routing

- ◆ Routing Module for SPA
- ◆ Routes, params and child routes
- ◆ Route Guards

→ Angular Forms

- ◆ Template Drive Forms
- ◆ Form Data Binding
- ◆ Data Validation
- ◆ Reactive Forms

→ Services and Dependency Injection

- ◆ Services
- ◆ Dependency Injection
- ◆ Data Sharing
- ◆ Http Client for calling API

→ Building Complete Working Applications using Angular, JAVA, Spring Boot