CREDO SYSTEMZ
Simplifying IT

START YOUR FULL STACK DEVELOPER CAREER TODAY!!

CREDO SYSTEMZ

AI POWERED FULL STACK DEVELOPER COURSE

India's 1st AI-Driven IT Training – Credo Systemz

Capstone Projects :

Real Time Full Stack Developer Training Projects. Build the frontend and backend, add smart features using AI, and solve practical problems with your solution.

Food Delivery App

Building an application to browse restaurants, order food, and track deliveries in real-time.

Twitter/Instagram Clone

Developing a social media platform to post content, follow others, with likes and comments.

Chat Application

Building a real-time chat app in which users can message each other in one-on-one or group chats.

Event Booking App

You can see events and book tickets online. It sends email confirmation after booking.

Learning Management System (LMS)

People can search and apply for jobs. Companies can post job openings.

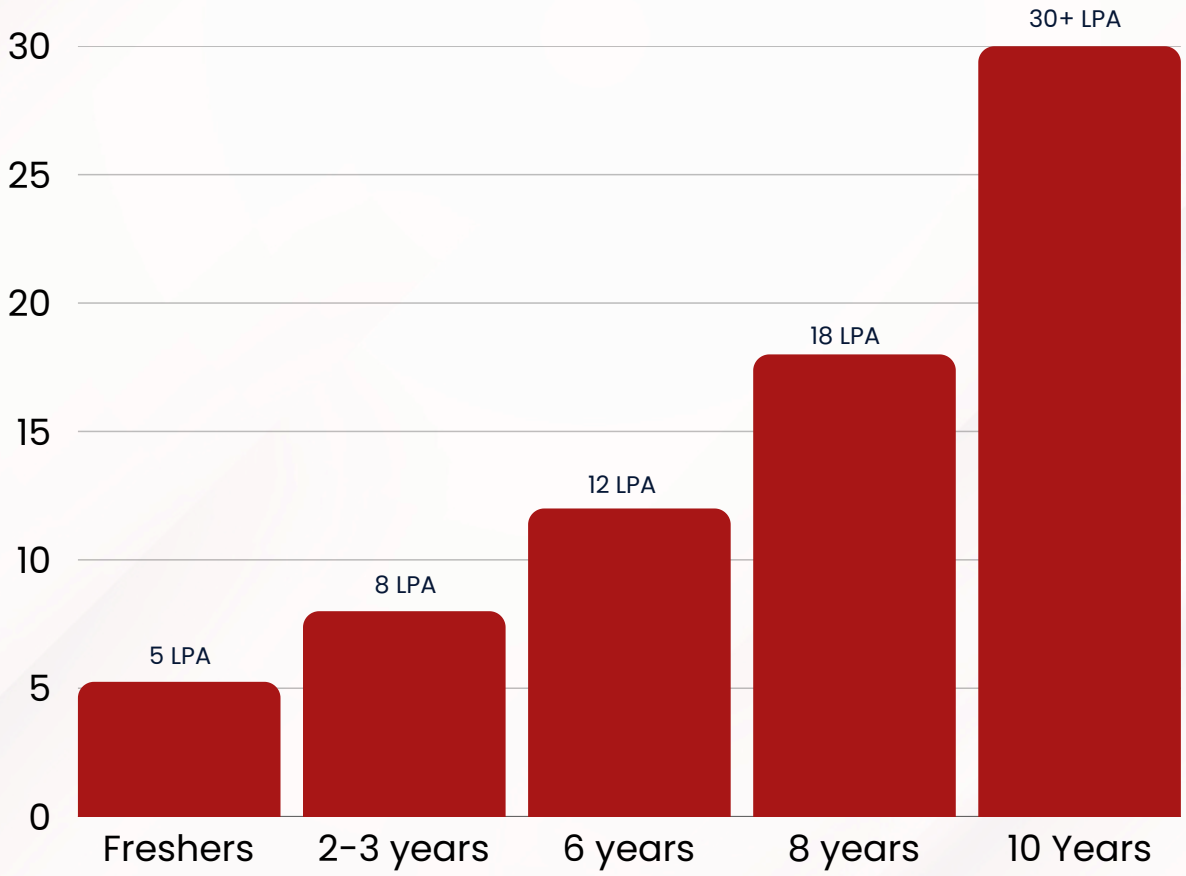
Learning App

Users can order food online from restaurants. It shows live order updates and payment options.

Full Stack

Opportunities & Demand

- Lead Full Stack Engineer
- Full Stack Architect
- Technical Lead (Full Stack)
- Full Stack DevOps Engineer
- Cloud Full Stack Developer
- Full Stack AI Developer



FULL STACK WITH AI COURSE SYLLABUS

ANGULAR + JAVA SPRING

Duration : 100 hrs

Phase 1 – AI Introduction

Section 1 : Full Stack with AI

Introduction

- What is Artificial Intelligence(AI)?
- Introduction to Artificial Intelligence
- Evolution of AI in the software industry.
- Key differences between AI, ML, and automation.
- How AI understands prompts and generates outputs in development.

How Developers Use AI Tools Today

- Real-time code suggestions and auto-completion.
- Debugging and resolving errors.
- Auto-generating documentation and test cases.
- Enhancing productivity in frontend and backend development.
- Learning new technologies via AI-powered explanations.

Overview of AI Tools (Copilot, ChatGPT, Codeium, Tabnine)

- GitHub Copilot – Code completion and generation inside VS Code.
- ChatGPT – Conversational AI for explaining concepts, solving bugs, and writing logic.
- Codeium – Lightweight open-source AI assistant for developers.
- Tabnine – Predictive code suggestions for multiple languages and editors.
- Amazon CodeWhisperer – AI coding tool optimized for cloud and AWS-based applications.
- Feature comparison and use cases.

Installing GitHub Copilot in VS Code

- Prerequisites: GitHub account, VS Code setup.

- Installation steps: Search, install, and activate the Copilot extension.
- Verifying Copilot functionality within the editor.

Activating Copilot with GitHub Account

- Connecting VS Code with GitHub credentials.
- Subscription setup (Free/Trial/Pro plans).
- Enabling Copilot across projects and environments.

Phase 1 – Frontend Development

Section 2 : HTML 5

- Introduction - Web
- What are the different Types of Web Apps overview?
- Introduction to HTML
- Define HTML Structure
- Difference between Tag vs Element
- Difference between Semantic vs Non-semantic elements
- Difference between Block level elements vs Inline elements
- HTML Elements
- HTML Forms & its Attributes
- HTML Input Elements
- HTML Global Attributes
- HTML Element Specific Attributes

AI in Action:

- Use ChatGPT to explain HTML tags, forms, and semantic elements.
- Generate HTML templates using GitHub Copilot or Codeium from simple prompts like “Create a contact form.”
- Auto-suggest attributes and fix structure issues using AI linters or extensions in VS Code.

Section 3 : CSS & Bootstrap

- What is CSS?
- Understanding the CSS Syntax.
- CSS Selectors
- How To Add CSS in HTML
- CSS Colors & Backgrounds

- CSS Borders, Margins, Padding, Height and Width
- Responsive Web Design – Media queries
- What is Bootstrap?
- Get Start with Bootstrap
- Containers
- Grid System
- Structure of a Bootstrap Grid
- Bootstrap Colors
- Bootstrap Tables
- Bootstrap Jumbotron
- Bootstrap Alerts, Buttons

AI in Action:

- Use Copilot to autocomplete style rules and responsive media queries.
- Generate custom CSS frameworks using AI prompt tools like Tabnine.
- Ask ChatGPT to refactor or explain complex CSS rules and layout bugs.
- Build Bootstrap grids or buttons by typing "Create 3-column responsive layout using Bootstrap."

Section 4: JavaScript

- Introduction – Javascript
- JavaScript Events
- JavaScript Functions
- Inbuilt methods in JS
- Arrays in JavaScript
- Objects in JavaScript
- Conditional and loops in JavaScript
- HTML DOM Manipulation

AI in Action:

- Copilot can complete functions or suggest better loop structures.
- Use ChatGPT to explain callback vs promise vs async/await in your own words.
- Auto-generate ES6 classes, arrow functions,
- Destructuring examples using natural language.
- Debug JavaScript errors via ChatGpt Prompts
- Debug Logic mistakes via ChatGPT prompts.

Section 5: TypeScript

- TypeScript Introduction
- TypeScript Environment Setups
- Variables in TS
- Datatypes in TS
- OOPS in TypeScript
- Features in TS

AI in Action:

- Get type suggestions using Copilot, auto-generate interfaces/classes, and refactor TS code with AI tools.

Section 6: Angular Introduction

- Angular- Definition
- Difference between Framework & Library?
- History of Angular and its versions.
- Why Angular?
- What are the Features of Angular
- Define Single Page Application
- What is the Difference between SPA & Traditional Application?
- Define MVC
- How MVC works in Client & Server sides?

Section 7: Angular Environment setups

- What is Angular CLI?
- What is the Purpose of the CLI?
- Angular CLI installation.
- CLI vs Without CLI Overview.
- Create an Angular App by using CLI.
- Compiling the Angular App & Open it in a browser.
- Angular app Bootstrapping process
- About Angular libraries
- Brief explanation about the structure of the Angular App.

Section 8: Main Building Blocks of Angular

- An Overview of the below Main Building blocks of Angular

- Modules
- Components
- Decorator
- Metadata
- Templates
- Data binding - Directives
- Services
- Dependency Injection.

Section 9: Angular Modules

- Angular Module Overview.
- Define the Importance of the Module?
- Why Modules?
- Root Module, Core Module, Feature Module and Shared Module - Overview.
- How to create Angular Modules?
- @NgModule Decorator & its Meta data properties - Overview.

Section 10: Components

- Angular Component - Overview.
- @Component decorator & its Meta data properties.
- Component's Structure overview.
- What are the ways to render a Component in the view?
- Component Lifecycle Hooks.
- Nested or Parent & Child Component - Overview

Section 11: Data Binding, Property Binding, Event Binding & 2-way Data Binding

- Data Binding Introduction
- Define String Interpolation.
- Property Binding - Overview.
- what is Custom Property Binding?
- Overview of Event Binding
- String Interpolation VS Property Binding
- Two-way Data Binding
- Implementing the 2-way Data Binding

Section 12: Services

- Introduction - Service.
- Importance of Service.
- How to create Services in Angular?
- What are the ways to Provide Services in Angular?
- Dependency Injection - Overview.
- How to use Dependency Injection?
- What is @Injectable()?

Section 13: Directives

- Directives - Introduction.
- Component VS Directives
- What are the Different kind of Directives available in Angular?
- Difference between Structural & Attribute Directives.
- Overview of All Structural & Attribute Directives.
- @Input decorator and its methods.
- What is ElementRef and its purpose?

Section 14: Components Communication

- Overview of Components Interaction.
- Component Interaction from Parent to Child.
- Component Interaction from Child to Parent.
- @ViewChild decorator overview
- @Input & @Output decorator overview

Section 15: Reactive Form in Angular

- What is Reactive Form?
- Difference between Template Drive & Reactive Form
- What is form group & form control?
- How to sync view & Reactive form TS?
- What is Patch Values & Set Values?
- How to get Reactive from Values?

Section 16: Service and Dependency Injection in Angular

- Service in Angular
- Create & configure Service in Angular.
- How to do Dependency Injection in Angular?
- Define Singleton Object

Section 17: Routing in Angular

- What is Routing?
- How Routing makes our App into SPA?
- How to configure Routing in an Application?
- Load our components dynamically based on url path.
- How to create Child Route?
- Navigating to other links programmatically.
- Passing Parameter to the Routes.
- Client-side authorization using Route Guard

Section 18: HTTP & Observable in Angular

- HTTP Client in Angular.
- REST API - Overview
- How to establish HTTP request to Server side.
- How HTTP Mechanism works?
- What are the methods available in HTTP?
- Define Observable & Observer
- What are the call back methods available in Observable?
- Creation of a Custom Observable
- Define next(), error() and complete()
- How to send Query Params & Custom Headers?
- How to connect any backend & APIs?

Section 19: Authorization in Angular & JWT

- Client side Authorization vs Server side Authorization.
- Server side Authorization by using JWT Token.
- Set JWT Key Expiry time.
- HTTP interceptors - Overview.
- How to configure HTTP interceptors?

Section 20: Angular – With AI Integration

Use Copilot :

- Create components, services, and modules quickly
- Write routing setup and navigation logic
- Build forms with validation rules
- Add Dependency Injection in services
- Auto-complete commonly used Angular code

ChatGPT:

- Explore Angular topics like data binding and component communication
- Solve errors in templates or logic
- Lifecycle hooks in easy terms
- Generate examples for forms, routing, and API calls
- Answer “how-to” questions while learning

Use Codeium or Tabnine:

- Suggest HTML and binding code in templates
- Auto-fill *ngIf, *ngFor, and form inputs
- Style/class bindings
- Generate reusable code blocks
- Speed up writing component and service logic

Phase 3 – Backend Development

Section 21: Introduction - Java

- Introduction about Programming Language Paradigms
- Why Java?
- Flavors of Java.
- Java Designing Goal.
- Difference between JDK,JRE and JVM
- JVM –The heart of Java .
- Java Architecture
- Java’s Magic Bytecode.

Section 22: Programming Fundamentals

- Data types

- Difference between Primitive & Non Primitive Datatypes
- Variable Declaration & Initialization
- Type Casting
- Operators and its types

Section 23: Control Structure and Looping Statements

- IF conditions
- IF-ELSE conditions
- ELSE-IF Ladder conditions
- SWITCH-CASE statements
- "FOR" Loop
- "WHILE" Loop
- "DO-WHILE" Loop
- ENHANCED "FOR" Loop

Section 24: OOPS INTRODUCTION AND BASICS

- What is OOPS?
- Necessity and Advantage of OOPS
- OOPS Designs with real-time examples.
- What is mean by class and object?
- Relation between a Class and an Object
- How to create class and object

Section 25: OOPS BASICS (CLASSES & OBJECTS)

- Components of a Class
- Types of Variables and its uses.
- Constructor advantages and its types
- Ways to initialize the Object
- "this" keyword
- Static Block & Instance Block

Section 26: OOPS CONCEPTS - INHERITANCE

- Inheritance and its advantages
- The "extends" keyword
- Types of Inheritance

- Single Inheritance
- Use of “super” keyword and forms of “super” keyword

Section 27: OOPS CONCEPTS – ENCAPSULATION ALONG WITH PACKAGES

- What is package and its advantages
- Types of packages
- Static Import
- Access Modifiers
- Encapsulation

Section 28: ARRAYS AND STRINGS

- What is Array and its advantages
- Types of Array
- Jagged Array
- String, String Buffer, String Builder

Section 29: EXCEPTION HANDLING

- What is Exception and its types
- How to handle exception?
- Multiple catch blocks
- Finally block
- Difference between throw and throws
- Custom Exception

Phase 4 – Database Development

Section 30: DATABASE

- What is Database?
- What is MySQL?
- Parts of MySQL
- Operators and Clauses in MYSQL.
- Functions and procedures in MYSQL.

Section 31: JDBC

- What is JDBC?

- Types of Drivers
- Loading the drivers
- Connection, Statement, PreparedStatement, CallableStatement, ResultSet Interfaces
- Batch Updates.

Section 32: UTIL PACKAGE – COLLECTIONS FRAMEWORK

- The “Collection” Interface
- List, Set and Map Interfaces
- Cursors in java and its differences
- Difference between the implementation classes of List, Set and Map

Section 33: GENERICS IN COLLECTIONS FRAMEWORK

- Generics in Java
- Types of Generics
- Uses and Limitations of Generics
- Custom objects with Generic collections
- Sorting in Collections

Section 34: SPRING BOOT ARCHITECTURE

- Spring Boot Framework definition
- Spring Boot Framework design principals
- Spring Boot interfaces
- Bean Factory
- Application Context
- Factory methods

Section 35: INVERSION OF CONTROL

- Dependency Injection advantages
- Setter Injection
- Constructor Injection
- Bean Life Cycle
- Inner Beans
- References to other beans
- Scopes in Spring

Section 36: AUTOWIRING & SPRING BOOT MISCELLANEOUS

- Autowiring advantages and modes
- Spring Boot with Collections
- Reading values from Properties file
- Expression Language in Spring Boot
- Events Handling in Spring Boot

Section 37: AOP

- Spring Boot AOP Advantages
- AOP Configuration
- AOP Terms
- AOP Types of Advices

Section 38: SPRING BOOT JDBC

- JDBC Templates
- Row Mapper
- BeanProperty Row Mapper
- Bean Row Mapper
- Resultset Extractor

Section 39: ARRAYS AND STRINGS

- Database access layer for object, relational databases
- Spring Boot Framework Benefits with hibernate
- Spring Boot with Hibernate configuration files
- SpringData JPA
- Crud Repository and JPA Repository

Section 40: SPRING WEB MVC AND SPRING BOOT

- MVC Architecture in Spring
- About Dispatcher Servlet
- Types of Handler Mappings
- Types of Controller
- Types of View Resolver
- Spring MVC Annotations

- Spring MVC with CRUD operations
- Spring with Web Services
- Spring Boot Application
- Difference between Spring and Spring Boot
- Spring Boot Annotations

Section 41: SPRING BOOT - OVERVIEW

- What is Spring Boot?
- Developing a Project with Spring Boot Initializer
- Developing a REST API Controller with Spring Boot
- Examining the Spring Boot Project Structure - Part 1
- Examining the Spring Boot Project Structure - Part 2
- Spring Boot Starters - A Curated List of Dependencies
- Parents for Spring Boot Starters

Section 42: SPRING BOOT DEV TOOLS AND SPRING BOOT ACTUATOR

- Spring Boot Dev Tools - Overview
- Spring Boot Actuator - Overview
- Spring Boot Actuator - Accessing Actuator Endpoints
- Spring Boot Actuator - Applying Security to Actuator Endpoints

Section 43: SPRING BOOT - RUNNING SPRING BOOT APPS FROM THE COMMAND LINE

- Running Spring Boot Apps from the Command Line - Overview
- Running Spring Boot Apps from the Command Line - Coding

Section 44: SPRING BOOT - APPLICATION PROPERTIES

- Injecting Custom Application Properties
- Configuring the Spring Boot Server

Section 45: Real Time Projects

- 5+ Real-time projects (Full stack coverage)
- Setup Git for local repository

- Create a GitHub account
- Sync the code base with GitHub repo
- Firebase Hosting Overview
- Setting up Firebase in local
- Host your project in Firebase

Section 46: Java Spring – With AI Integration

Use Copilot:

- Create controllers, services, and repositories faster
- Add annotations like @RestController, @Service, @Autowired
- Write REST API methods
- Set up application settings quickly
- Build CRUD operations with less typing

ChatGPT:

- Dependency injection and beans
- fix common errors and bugs
- Guide you in creating REST APIs
- Show examples of request and response handling

Use Codeium or Tabnine:

- Suggest service methods and repository code
- Complete JPA query method names
- Fill in config files like application.properties
- Help write test cases
- Speed up writing helper and error-handling code

Section 47: Placements

- Build your Professional Resume
- Update LinkedIn Profile
- Interview based Training – GD, Tech round, HR panel
- Minimum 5 Mock interviews before Real interview
- 100% Placement Guaranteed
- Join your Dream Job

SKILLS AND TOOLS

Skills Covered

HTML

CSS

Java

Mongo DB

My SQL

Node JS

Oracle

Javascript

Spring Boot

Bootstrap

API

Database

Angular

Ajax

TypeScript

Tools Covered

VS Code

Spring Tool Suite

Server

Postman

Intelj

Git

Eclipse

Command Line

PLACEMENT SUCCESS STORIES

	Designation	Company	Package
Rajiv	Full Stack Developer	SIMSON Softwares	7.8 LPA
YogaLakshmi	Front End Developer	ECS Cloud Infotech	8.5 LPA
Madhumathi	Software Developer	COLAN INFOTECH	7.8 LPA
Suja	Cloud Engineer	Beelogical The Magic of Logic	8.6 LPA
Sanakarapandiyan	Full Stack Developer	Shellinfo Global	8.0 LPA
Asai Thambi	Full Stack Developer	SUNSMART	11.0 LPA

OUR HIRING PARTNERS

TATA CONSULTANCY SERVICES

amazon

Capgemini

Tech Mahindra

cognizant

SKILL LYNC

IRIS

SUNSMART

Ascent24

ECS Cloud Infotech

LIDER
MINDFUL THOUGHTS DRIVEN BY PASSION

just Integrate IT
Helping Everyone Succeed

LATESHIPMENT.COM

SIMSON Softwares

Shellinfo Global

impress

VAYUZ

COLAN INFOTECH

Beelogical
The Magic of Logic

COGMED

ALTOS Technologies

Earn your Full Stack Course Completion Certificate

Credo Systemz's certificate is highly recognized by 30K Global companies around the world.

CREDO SYSTEMZ
Simplifying IT

CERTIFICATE
OF EXCELLENCE

THIS IS TO CERTIFY THAT

GUHAN. R

CANDIDATE ID : [illegible]
has successfully completed
FULLSTACK DEVELOPER
Course Conducted During March 2025 to May 2025

Date: 05-06-2025

Authorised Signatory

#30, 3rd Main, Rajalakshmi Nagar, Velachery, Chennai - 42. Ph : +91 98844 12301/ 96001 12302, Web : www.credosystemz.com

WHAT OUR TRAINEE SAYS?

Libin Charan

4.7 ★★★★½

Credo Systemz's Full Stack course is very practical and job-oriented. The trainers explained every topic clearly with real-time examples. Placement support was strong. I got placed after training. Totally worth the investment!

Mukesh Babu

4.2 ★★★★½

The Full Stack training at Credo Systemz covered front-end and back-end thoroughly. Live projects helped a lot. Mentors were always helpful. Thanks to their placement team, I'm now working in an IT company.

Padmesh

5.0 ★★★★★

I joined Credo Systemz for Full Stack training and it was the best decision. Great trainer, hands-on projects, and mock interviews helped me switch to IT. Strongly recommend for career changers!

Vasmitha

4.9 ★★★★½

Credo Systemz Full Stack program gave me confidence to attend interviews. I learned MERN stack with real-time examples. The support from admin and placement team was great. Got placed right after course completion.

Jason Isreal

4.0 ★★★★

As a fresher, Credo Systemz's Full Stack course gave me all the knowledge I needed. The trainer's guidance and regular assessments helped me grow. I got placed within weeks of completing the course!

Subritha

4.5 ★★★★½

I had no coding background but the Full Stack course at Credo Systemz was easy to follow. Trainers were supportive and explained clearly. Placement team constantly followed up until I got placed. Thanks!

CHENNAI

VELACHERY

New # 30, Old # 16A, Third Main Road, Rajalakshmi Nagar, Velachery, (Opp. to Murugan Kalyana Mandapam), Chennai – 600 042.

+91 98844 12301

OMR

Plot No.8, Vinayaga Avenue, Rajiv Gandhi Salai, (OMR), Okkiampettai, (Behind Okkiyampet Bus Stop) Chennai – 600 097.

+91 96001 12302

OVERSEAS

USA

Houchin Drive, Franklin, TN -37064. Tennessee

UAE

Sima Electronic Building, LLH Opposite, Electra Street – Abu Dhabi

India's 1st AI-Driven IT Training

Credo Systemz