Angular 21

Course code: - / Course Duration: 5 day; / 35 hours; Instructor-led/ remote online training
Time: 9.00am – 5.00pm
Break: 10.15am – 10.30am /3.15pm – 3.30pm
Lunch: 1.00pm – 2.00pm

OVERVIEW
Angular 21 is designed for building modern, high-performance, scalable web applications with a focus on fine-grained reactivity, signals, standalone APIs, and improved developer experience.

Participants will set up a modern Angular 21 development environment and learn to build applications using component-driven architecture, signals-based state management, and modern routing and HTTP patterns.

This course covers Angular 21 components, directives, services, forms, routing, API integration, and modern reactive programming techniques.

PREREQUISITES
Before attending this course, students should have general programming experience and knowledge of HTML, CSS, and JavaScript.

METHODOLOGY
This program will be conducted with interactive lectures, PowerPoint presentations, discussions and practical exercise

COURSE OUTLINES

DAY 1 - TypeScript and Angular Foundation

Module 1: Introduction to TypeScript
- Overview
- Environment Setup
- Basic Syntax
- Types, Variables and Operators
- Decision Making and Loops
- Functions and Arrow Functions
- Generics
- Enums
- Numbers and Strings
- Arrays
- Tuples
- Union and Intersection Types
- Interfaces, Classes and Objects
- Type Aliases
- Modules (ES Modules)
- Decorators
- Utility Types
- Summary
- Lab 1: TypeScript Fundamentals
  - Build a TypeScript-based inventory model:
  - Define interfaces and classes
  - Use generics for reusable utilities
  - Implement enum and union types
  - Compile and run using ts-node

Module 2: Introduction to Angular
- Why Angular 21
- Angular 21 Key Features
- Angular CLI Overview
- Installing and Using Angular 21
- Creating the First Angular Project
- Project Structure Overview
- Standalone Components Architecture
- Dependency Injection Overview
- Change Detection Overview
- What Is New in Angular 21
- Summary
- Lab 2: Create Your First Angular App
  - Install Angular CLI
  - Create a standalone Angular 21 project
  - Generate components
  - Run and explore dev tools

Module 3: Introduction to Single Page Applications
- What Is a Single Page Application (SPA)
- SPA Workflow
- Traditional Web Application Architecture
- SPA Advantages
- SPA Challenges
- Implementing SPAs Using Angular 21
- SPA Using Standalone Components
- Component-Based Navigation
- Displaying Components Dynamically
- Implementing SPA Using Angular Router
- Summary
- Lab 3: Build a Simple SPA Layout
  - Create layout components
  - Add navigation
  - Dynamically load views

Day 2 – Components, Binding and Forms

Module 4: Building with Components

- Introduction to Angular Components
- Standalone Components
- Component Metadata and Decorators
- Templates and Inline Templates
- Interpolation – Text, Objects and Arrays
- Component Styling
- View Encapsulation
- Integrating External CSS Frameworks
- Creating Header Component
- Creating Footer Component
- Creating Product Component
- Summary
- Lab 4: Build a Product Dashboard UI
  - Create:
  - Header
  - Sidebar
  - Product List
  - Product Card Component

Module 5: Data Binding and Event Handling

- Angular Binding Syntax
- One-Way Data Binding
- Property Binding
- Attribute Binding
- Event Binding
- Event Binding Examples
- Template Reference Variables
- Two-Way Binding Using ngModel
- Input and Output Properties
- Parent–Child Component Communication
- Summary
- Lab 5: Parent–Child Communication
  - Pass product data to child components
  - Emit events from child to parent
  - Implement two-way binding

Module 6: Attribute Directives

- What Are Directives
- Directive Types
- Built-in Attribute Directives
- Dynamic Class Binding
- Dynamic Style Binding
- Conditional Styling
- Dynamic Property Binding
- Controlling Element Visibility
- Dynamic Image and Hyperlink Binding
- Summary

Module 7: Modern Control Flow and Structural Rendering

- Introduction to Angular Control Flow
- Conditional Rendering Using @if
- Else and Nested Conditions
- Looping Using @for
- Tracking Items with track
- Accessing Index and Context Variables
- Conditional Switching Using @switch
- Rendering Templates Dynamically
- Performance Benefits of Modern Control Flow
- Summary

Module 8: Template Driven Forms

- Overview of Angular Forms
- Creating a Basic Angular Form
- Binding Input Fields
- Accessing Form State
- Handling Form Submission
- HTML5 Validation
- Angular Validation Overview
- Built-in Validators
- Validation States
- Displaying Validation Errors
- Disabling Submit for Invalid Forms
- Working with Checkboxes
- Select (Drop-Down) Fields
- Date and Number Inputs
- Radio Buttons
- Summary

Module 9: Reactive Forms

- Introduction to Reactive Forms
- Registering Reactive Forms Module
- Creating FormControl
- Registering Controls
- Managing Control Values
- Creating FormGroup
- Connecting Model and View
- Nested Form Groups
- Updating Form Values
- Using FormBuilder
- Built-in Validators
- Custom Validators
- Dynamic Forms Using FormArray
- Summary
- Lab 6: Advanced Reactive Forms
  - Create dynamic product form using FormArray
  - Add custom validator

- Show real-time validation errors
- Submit and log structured JSON

Day 3 – Services, Signals and HTTP

Module 10: Services and Dependency Injection

- What Is a Service
- Creating Services
- Dependency Injection Fundamentals
- Injecting Services
- Service Scopes
- Using Services in Components
- Shared Services
- Optional Dependencies
- Host Dependencies
- Summary
- Lab 7: Shared Product Service
  - Create centralized state service
  - Inject into multiple components
  - Refactor local state into service

Module 11: Signals and Reactive Programming

- Introduction to Angular Signals
- Creating Signals
- Reading and Updating Signal Values
- Computed Signals
- Effect Functions
- Signal-Based State Management
- Signals vs Observables
- Using Signals with Components
- Signal-based component inputs
- Signal performance patterns
- Avoiding unnecessary effects
- Summary
- Lab 8: Signal-Based Cart System
  - Create cart signal
  - Compute total price
  - Auto-update UI using computed signals

Module 12: RxJS and Observables

- Introduction to RxJS
- Observables Overview
- Creating Observables
- Subscribing and Unsubscribing
- Common RxJS Operators
- Subject and BehaviorSubject
- Async Pipe
- Interoperability Between Signals and Observables
- switchMap, mergeMap, concatMap
- Error handling with catchError
- takeUntil and memory management
- Combining streams
- Summary

Module 13: HTTP Client and API Integration

- Angular HTTP Client Overview
- Setting Up HTTP Client
- Creating API Services
- Making HTTP GET Requests
- HTTP Headers and Options
- POST, PUT and DELETE Requests
- Handling HTTP Responses
- Error Handling Strategies
- Observables in HTTP
- Consuming APIs in Components
- HTTP Interceptors
- Authentication headers
- Global error handling
- Loading indicators
- Environment configuration
- Summary
- Lab 9: REST API Integration
  - Connect to mock API
  - Implement CRUD operations
  - Add interceptor for logging
  - Handle loading and error states

Day 4 – Routing, Architecture, and Advanced Angular

Module 14: Pipes and Data Formatting

- What Are Pipes
- Built-in Pipes
- Using Pipes in Templates
- Chaining Pipes
- Using Pipes in TypeScript
- Decimal Pipe
- Currency Pipe
- Date Pipe
- Creating Custom Pipes
- Pure and Impure Pipes
- Summary

Module 15: Angular Routing and Navigation

- Routing Overview
- Angular Router Architecture
- Route Configuration Using Standalone APIs
- Router Outlet
- Navigation Links

- Programmatic Navigation
- Route Parameters
- Query Parameters
- Retrieving Route Data
- Lazy Loading Routes
- Handling Invalid Routes
- Route Guards (CanActivate, CanDeactivate)
- Resolver
- Preloading strategies
- Standalone route configuration
- Nested routes
- Summary
- Lab 10: Secure Admin Route
  - Implement login mock
  - Protect route using guard
  - Load admin module lazily

Module 16: Component Lifecycle and Change Detection

- Lifecycle Hooks Overview
- ngOnInit, ngOnDestroy
- afterRender hooks (modern APIs)
- OnPush Change Detection
- Manual Change Detection
- Performance implications

Module 17: State Management Patterns

- Local component state
- Service-based state
- Signal-based global state
- When to use RxJS vs Signals
- Introduction to NgRx (Conceptual Overview)

Module 18: Content Projection and Dynamic Components

- ng-content
- Multiple slots
- ViewContainerRef
- Dynamic component loading
- Portals concept

Day 5 – Testing, Performance, Security, and Deployment

Module 19: Testing in Angular

- Testing Strategy Overview
- Unit Testing Components
- Testing Services
- TestBed
- Mocking Dependencies
- Testing Reactive Forms
- Testing HTTP Calls
- Introduction to E2E Testing
- Lab 12: Write Unit Tests
  - Test product service
  - Test form validation
  - Mock HTTP calls

Module 20: Performance Optimization

- Lazy Loading Best Practices
- Code Splitting
- OnPush Strategy
- trackBy Optimization
- Signals Performance Tuning
- Avoiding Memory Leaks
- Bundle Analysis

Module 21: Security Best Practices

- XSS Prevention
- Angular Sanitization
- Route Protection
- Token Storage Strategies
- CORS Overview
- Secure HTTP Communication

Module 22: Environment Configuration and Build

- Angular Environments
- Production Build
- AOT Compilation
- Build Optimization
- Source Maps
- Debugging Production Issues

Module 23: Final Lab (Capstone Project)

- Build a complete Product Management System
- Features:
- Authentication mock
- Protected admin routes
- Product CRUD with API
- Reactive form validation
- Signal-based cart
- Animations
- Route guards
- HTTP interceptor
- Unit tests
- Production build