**.Net Core and Angular Training**

**Total Duration: 5Days**

**ASP.NET Training Objective**

**At the completion of this course, attendees will be able to;**

* Understand .NET Core architecture and Advantages
* Build and run cross-platform ASP.NET apps on Windows, Mac and Linux
* Understand ASP.NET Core middleware
* Configure ASP.NET Core MVC
* Use Dependency Injection in ASP.NET Core
* Work with Entity Framework Core
* Handle Errors in ASP.NET Core
* Create REST Service using Web API
* Understand and Implement Repository, Unit of Work and Dependency Injection Design Pattern
* Publish theirs Apps on IIS
* Angular UI

**Pre-requisites**

**Anyone who wants to learn ASP.NET Core course should have a basic knowledge of C# and HTML.**

**Tools/SDK/IDE**

**Visual Studio 2022 or higher, SQL Server 2017 or higher**

**Day 1**

* Introduction to .Net Core 6
* Setting up .Net Core 6 SDK
* .Net Core vs .Net Framework
* .netstadard2.x
* New templates of VS 2022
* What is .Net Core FX
* .Net Core 6 Architecture

NET CLI

* Introduction to DNX, DNVM, DNU
* Introduction to .NET CLI
* .NET CLI Commands

Introduction to ASP.NET Core

* Understanding ASP.NET Core
* ASP.NET Core Features
* ASP.NET Core 1.0 vs. ASP.NET MVC5 vs. ASP.NET WebForms
* Advantages of ASP.NET Core

Visual Studio Project Templates

* Understanding Visual Studio ASP.NET Core Templates
* Creating an ASP.NET Core project
* Understanding ASP.NET Core project folder structure
* Understanding configuration files
* Roysln compiler and RyuJIT and CoreCLR (how it works)
* async and await
* Dotnet CLI and Visual Studio Code
* ASP.Net Core Web application
* Middlewares and custom middleware
* Dependency injection with custom services and lifetime
* Working with Configuration
* Working with Environment
* Inprocess vs Output process
* Kestrel web server vs IISExpress
* Working with Razor Pages
* Working with Razor Library
* Overview of Blazor

**Day 2**

* What is Entity Framework Core
* EFCore API
* DB first and Code First
* Migrations in EF Core
* Custom Validators
* DTO
* CRUD operations with EF Core and SQL Server
* Transactions with EF Core
* ASP.Net Core MVC and EF Core Case study

Web API:

* What is RESTful services
* Why RESTful services
* What is WEB API?
* Filters in Web APi of .Net Core
* Web API methods (get/post/put/delete)
* Postman to test API methods
* Working with Swagger
* Angular and Web api (Full stack Case Study)
* CORS with API

Security

* Authentication (JWT Token) - SSO
* Azure Active directory
* Authorization
* CORS
* CSRF
* XSS

**Day 3**

1.1.1 Angular 12 Architecture

Demands of Modern Front-end Applications

Functional and Non-Functional Requirements

Quick review on other JavaScript Frameworks

Modular Design Requirements

Separation of Concerns

Identifying Cross-Cuts

Inversion of Control

Dependency Injection

Late-binding and Lazy Loading

Support for Architectural Styles

Evolution of ES and TypeScript

1.1.2 Evolution of Angular and New Features in Angular

AngularJS to Angular 12.0

Angular CLI

Ivy Rendering Engine

Module With Providers Support

Changes to Angular Forms

Lazy-loaded components

Latest TypeScript versions

Webpack 5 Support

Improved Logging and Reporting

Updated Language Service Preview

Support for Module Replacement

1.1.3 Review of Object-Oriented Programming with TypeScript

Programming Languages for Use with Angular

TypeScript Syntax

Types in TypeScript

Object Oriented Programming

Classes and Objects

Interfaces

Arrow Functions

1.1.4 Reviewing Installing and Configuring Angular with Boiler plate

Installation and Configuration

Anatomy of a Basic Application

Dependencies

Repository

Linting configurations

Web Server Configurations

1.1.5 Architecting an Angular Application

Translating Functional and Non-functional Requirements

Addressing UI Concerns

Mapping UI Requirements into Angular Components

Composing Hierarchical Components by Separating Concerns

Leveraging on Component Life-cycle hooks

OnInit and its relevance

Identifying candidates for Module packaging

Bootstrapping

1.1.6 Architectural Patterns and Practices

Model View Presenter (MVP)

Model View View Model(MVC)

Movel View Controller (MVVM)

1.1.7 Binding Components to Data

Binding Syntax

One-Way Output Binding

Binding Displayed Output Values

Setting Component Properties

Setting DOM Element Properties

**Day 4**

Directives, Pipes, Services, Routing and Security

2.2.1 Implementing Event-driven Architecture with Event Binding

Binding Events from UI to Component Properties

Event Emitter as an event aggregator

Firing Event from a Component

@Output() Example - Child Component

@Output() Example - Parent Component

Two-Way Binding of Input Fields

Use Two Way Binding

2.2.2 Angular Directives

Introduction to Directives

Structural and Attribute Directives

Syntactical differences between Structural & Attribute Directives.

Overview of Structural & Attribute Directives. ngIf & hidden directives.

Custom Attribute Directive

Custom Structural Directive

2.2.3 The importance of using Pipes as Utilities

Rationale for Angular Pipes

Difference between Pipes & Directives.

In-built Pipes

Pure and Impure Pipes

Creating Custom Pipes with multiple pipes

Chaining of Pipes

2.2.4 Injectables and Singletons in Angular Services

Introduction to Angular Services

Service Consumption

Creating Services in Angular

Singleton Object - Overview

Dependency Injection of Services

2.2.5 Service Oriented Achitecture and Web PI

SOA and RESTFUL Services

Connecting to API

CORS

Working with a NodeJS Web API

Configuring Mongo DB

Asynchronous Communications

2.2.6 Reactive Programming with RxJS and Observables in Angular

Overview of the RxJS Library

Subjects, Observers, Observables, and Operators

Movement of data from Observable to Observer

Transformations and Filters

map(), filter() & reduce()

Making and Retrying HTTP Requests

Utilities – tap()

Event subscribe/unsubcribe

Multicasting

Error Handling of API Responses

Inter-component Communication

3.3 Routing, Security, Guards & Interceptors

3.3.1 Routing and Navigation in Angular

Introduction to Routing

Intercepting URLs and Route Registry

Configuring Routes

Loading components dynamically based on url path.

Child Routes

Passing Parameter to the Routes.

Static Route vs Dynamic Route.

Fetching Route Params

3.3.2 Securing Angular Applications

Authentication

OAuth

Authorization

CORS

XSRF

Using LocalStorage

3.3.3 Implementing OAuth2, OIDC

angular-oauth2-oidc

JwksValidationHandler

Tokens

Implementing a Login Functionality with JwtToken

3.3.4 Angular Guards

Guards as Injectable

Developing a Route Guard

Injecting Route Guard to Modules

3.3.5 Authorization with Interceptors

Role & Membership based Authorizations Server side Authorization by using JWT Token.

HTTP interceptors

Injecting Interceptors

**Day 5**

4.4 Application State, Reactive Forms Elements, Web Workers, PWAs, Unit Testing and BDD

4.4.1 Application State Management with NgRx

Challenges in maintaining Application State

Flux Pattern

Store

Action

Reducer

Selector

Effect

NgRx and Tools

Schematics and Scaffolding

4.4.2 Template Driven Form in Angular

Purpose of Form.

Setting up an Angular Form

Accessing Form Values

What are the form states available in Angular?

Form Validations.

HTML5 Validation vs Angular form Validation.

How to set default values in form fields? Setup One-way data binding in Angular.

4.4.3 Reactive Form in Angular

Reactive Forms in Angular

Difference between Template Driven & Reactive Form

Syncing View

Reactive Form Controls

Form Validation

4.4.4 Favouring Inheritance over Composition with Angular Mixins for reusability

Types of composing Angular Component

Class Inheritance, Partial Classes, Component Composition

Pain Points of Class Inheritance

Typescript Mixins as Partial classes

Developing Granular Mixins

Composing Mixins into A Component

Draw backs of Mixins

Dependecies, Name clashes and growth problems

of 12

Best Practices of adopting Mixins

4.4.5 Angular Server-Side Rendering

Scenarios and Benefits of Server-Side Rendering

Universal Web Servers

Universal Template Engine

Routing

Bootstrapping modules

Filtering request urls

Blocking direct data requests

5.5 Angular Elements, PWAs, Web Workers, Unit Testing and BDD

5.5.1 Angular Elements

The need Reusable Web Component

Exposing granular Angular components as Custom Elements

Angular Elements Package

Using ngDoBootstrap() The customElement API

Consuming the Custom Element in UI

5.5.2 SPAs and PWAs

Addressing Accessibility and User Experience requirements

Service Workers

Web App Manifest

Using PWA manifest generator.

HTTPS communications

Adding Service Workers

Using HTTP Server

5.5.3 Improving Response Times with Web Workers

Introduction to Web Workers

Implementing Web Workers

Parameters for Measuring Performance with Web Workers

Diagnostics with Lighthouse

Web Worker Limitations and Pitfalls

5.5.4 Leveraging on Angular Ivy

Angular Compiler Options

Reducing Bundle Size

5.5.5 Test Driven Development of Angular Applications with Jasmine and Karma

Unit Testing Framework

Configuring Karma

Test Bed

Unit Testing Components and Services with Jasmine

Unit Testing Pipes

Monitoring Test Results

Using Fakes

Execution of Tests

Test Reporting

5.5.6 Behaviour Driven Development and Cucumber

Introduction to BDD

Cucumber and Gherkin

Gherking Syntax in feature files

Installing and Configuring Gherkin

Installing and Configuring Cucumber

protractor-cucumber-framework

Writing Spec files

Executing Tests

5.5.7 Deploying Angular Applications

Building Angular Application from Source

Deploying to Web Server

Dockerizing Angular Apps