NestJS

****

# Prerequisites

**To get the most out of this training, participants should have sound knowledge of**

* **JavaScript / TypeScript Basics**
  + ES6+ syntax (let/const, arrow functions, async/await, modules)
  + Strong understanding of TypeScript types, interfaces, and generics
* **Node.js & NPM/Yarn Basics**
  + Running Node apps (node index.js)
  + Installing packages (npm install / yarn add)
* **Database Fundamentals**
  + Basic SQL (CRUD queries)
  + Understanding of schemas, tables, and relations
* **Web API Knowledge**
  + REST principles (GET, POST, PUT, DELETE)
  + JSON request/response structures



# Software Requirements

**All participants should set up the below prior to commencement of training Core Environment**

* **Node.js (LTS version ≥ 18.x)**
* **Package Manager** (Choose one)
  + **npm** (Comes with Node.js)
  + OR **yarn** (npm install -g yarn)
* **NestJS CLI**
  + For scaffolding projects
* **Nx CLI** (for monorepo support)
* **TypeScript (≥ 5.x)**

**Database**

* **PostgreSQL or MySQL**

**Testing & Validation**

* **Jest (Default in NestJS projects)**
* **Class-validator & Class-transformer** (Already part of training)
* **Typia** (Runtime Type Validation + SDK Generation)

**Development Tools**

* **VS Code (Latest)**
* **Postman**

**Security & Utilities**

* **Docker Desktop**
  + Needed for Containerization (On Day 5)
  + Install from https://[www.docker.com/products/docker-desktop/](http://www.docker.com/products/docker-desktop/)
* **Git**
  + Source control + repo sharing

# Day-wise Program Agenda

**Day 1 – Foundations & First API**

***Goal: Understand NestJS Architecture, set up a project, and build a basic REST API.***

* **Introduction & Setup**
  + What is NestJS & Why use it? (OOP + FP + FRP philosophy)
  + Understanding Architecture (Modules, Controllers, Providers, DI)
  + Installing Nest CLI & Nx CLI (Brief Nx Intro: Why Monorepos?)
  + Creating a NestJS app with Nx workspace
  + Project Structure & Conventions
* **Controllers & Routing**
  + Creating Controllers and Routes
  + Path Parameters, Query Params, and Request bodies
  + DTOs with class-validator and class-transformer
  + Quick Intro to Typia (typia.validate()) for runtime type safety

**Hands-on Lab: Create first REST API with DTO validation**

****

# Day 2 – Services, Dependency Injection & Testing

***Goal: Learn Service layer, DI patterns, and Testing basics.***

* **Services & Dependency Injection**
  + Creating and injecting services
  + Understanding providers and DI tokens
* **Unit Testing**
  + Jest & Vitest – Intro to Testing in NestJS
  + Writing Unit Tests for Controllers and Services
  + Test Setup with TestingModule
* **Advanced DI Patterns**
  + Custom Providers (useClass, useFactory, useValue)
  + Dynamic Modules (Similar to Spring’s BeanFactory style)

**Hands-on Lab: Service layer + Unit Tests with mocks**

****

# Day 3 – Data, Validation & Configuration

***Goal: Persist Data, Manage Configurations, and Set up Validation/Error Handling.***

* **Database Integration**
  + TypeORM with PostgreSQL
  + Entities, Repositories, and Migrations
  + Repository Injection in services
* **Configuration & Environment Management**
  + Using @nestjs/config for environment variables
  + Multiple environment files (dev/test/prod)
  + Validating env vars with Joi
  + Global Constants and Config patterns
* **Validation & Error Handling**
  + DTO Validation with class-validator and Typia schemas
  + Custom pipes for Transformation/Validation
  + Global Pipes and Filters for Consistent Error Handling
  + NestJS Logger Service & Structured Lgging (Winston Integration)

**Hands-on Lab: CRUD API with DB + Logging + Error Handling**

# GraphQL with NestJS

# GraphQL Basics

# What is GraphQL? Queries, Mutations, Subscriptions

# GraphQL vs REST in real-world scenarios

# NestJS GraphQL Module

# Code-First vs Schema-First approaches

# Using @nestjs/graphql, decorators, resolvers

# Defining ObjectTypes, InputTypes, Args, Scalars

# Database Integration

# TypeORM / Prisma setup with GraphQL

# Relations (One-to-Many, Many-to-Many) in GraphQL

# Subscriptions & Real-Time

# WebSockets, GraphQL Subscriptions in NestJS

# Lab: Extend BookService into GraphQL API with queries & mutations

# Advanced GraphQL in NestJS

# Authentication & Authorization

# JWT + Guards with GraphQL

# Role-based access control in resolvers

# Error Handling & Caching

# Apollo Error handling patterns

# Query caching with Redis

# Federation & Modular GraphQL

# Apollo Federation concepts

# Splitting schema across multiple modules

# Testing GraphQL APIs

# Unit testing resolvers & services

# Integration testing with GraphQL Playground / Apollo Server

# Lab: Secure GraphQL API with JWT auth and role-based access

****

# Day 4 – Advanced API Features & Security

***Goal: Build Secure, Documented, and Maintainable APIs.***

* **Middleware, Guards & Interceptors**
  + Middleware for request Logging & Auth preprocessing
  + Guards (CanActivate) for role-based access control
  + Interceptors for response shaping and caching
* **Authentication & Security**
  + Oauth2 with @nestjs/passport and passport-jwt
  + Refresh token strategy basics
  + Security best practices: Helmet, CORS, rate limiting (@nestjs/throttler), input sanitization
* **Documentation & File Handling**
  + Setting up Swagger (@nestjs/swagger) for interactive API docs
  + Annotating DTOs for docs
  + File uploads with Multer (@nestjs/platform-express)
  + Serving static files securely

**Hands-on Lab: Secure API with JWT + Generate Swagger docs**

**Micro-Frontend (MFE) Concepts**

* **Micro-Frontend Architecture**
  + **What are MFEs? Advantages & challenges**
  + **MFE vs Monolith Frontend**
  + **Module Federation (Webpack 5) basics**
* **Integrating NestJS Backend with MFEs**
  + **API Gateway patterns for GraphQL & REST**
  + **GraphQL Federation with multiple frontends**
* **Frontend Frameworks with MFE**
  + **Angular MFE with Nx / Webpack Module Federation**
  + **React MFE integration with Apollo Client**
* **Lab: Build a simple MFE app (React + Angular) consuming NestJS GraphQL backend**

# Day 5 – Nx, Nestia, Async & Microservices

***Goal: Learn Enterprise Monorepo patterns, Type-safe APIs, and Microservices.***

* **Nx Monorepo Structure**
  + Why Nx for NestJS?
  + Understanding project.json and Nx plugins
  + Sharing DTOs & interfaces across backend and frontend
  + Project boundaries & Dependency graph
* **Nestia & Typia**
  + What is Nestia and how it generates type-safe SDKs
  + Integrating Nestia in Nx for auto-generated API clients
  + Using Typia for Runtime validation + compile-time types
  + Auto-update frontend SDK when backend DTO changes
* **Async Features & Microservices**
  + Scheduling Tasks with @nestjs/schedule
  + EventEmitter for in-app async events
  + **Microservices in NestJS:** TCP/Redis/RabbitMQ Basics
  + **Deployment Tips:** Dockerizing NestJS

**Hands-on Lab: Microservice + Docker + frontend SDK auto-generation**

****