Course 5: NestJS Mastery

■ Overview

NestJS is an opinionated Node.js framework built with TypeScript that brings powerful architectural patterns, scalability, and maintainability to backend development.

■ Prerequisites

- Strong TypeScript fundamentals
- Prior experience with Express.js basics

■ Outcomes

By completing this course, learners will be able to confidently build scalable backend applications with NestJS, leveraging modules, dependency injection, pipes, guards, interceptors, database integrations, authentication, microservices, and end-to-end testing.

■ Benefits of NestJS Mastery

This course equips learners with the ability to build enterprise-grade Node.js backends using NestJS's modular architecture and TypeScript-first approach. Students will master dependency injection, custom providers, and advanced request lifecycle features like pipes, guards, and interceptors. They will learn how to integrate databases with TypeORM and Mongoose, implement authentication with Passport (JWT & sessions), and add microservices communication with Redis or NATS. Beyond feature development, the course emphasizes caching, queues, and deployment with Docker, ensuring production readiness. Learners will also gain strong testing skills with @nestjs/testing for robust integration and e2e test suites. By the end, participants will have the expertise to deliver scalable, maintainable, and future-proof applications that stand out in professional environments.

■ Training Key Features

- Modular Architecture – Learn feature-driven and vertical slicing approaches for clean project design.
- Dependency Injection Mastery – Work with providers, scopes, dynamic modules, and custom DI strategies.
- Advanced Request Lifecycle – Implement middleware, DTO validation with class-validator, pipes, guards, and interceptors.
- Database Integrations – Build APIs with both relational (TypeORM) and NoSQL (Mongoose) databases, including transactions.
- Authentication & Authorization – Secure apps with Passport strategies, JWT tokens, sessions, and refresh token flows.
- Microservices Ready – Implement TCP, Redis, and NATS for distributed systems and message-driven architectures.
- Caching & Queues – Optimize performance with CacheModule and implement background jobs with BullMQ.
- Testing Confidence – Write unit, integration, and end-to-end tests with @nestjs/testing.
- Production Deployment – Dockerize NestJS apps and prepare for cloud-native environments.

- Industry-Ready Skills – Gain the expertise companies look for in enterprise-scale backend engineers.

■ Module Breakdown

- Module 1 – Architecture & Modules ::: Feature vs vertical slicing && Modular architecture best practices
- Module 2 – Providers & Dependency Injection ::: Provider scopes && Custom providers && Dynamic modules
- Module 3 – Controllers & Routing ::: REST endpoints && Parameter decorators && Request/response handling
- Module 4 – Middleware, Pipes & Validation ::: Middleware flow && DTO validation with class-validator && Custom pipes
- Module 5 – Guards & Interceptors ::: Role-Based Access Control (RBAC) && Logging && Caching strategies
- Module 6 – Database Integrations ::: TypeORM module && Mongoose module && Transactions & lifecycle management
- Module 7 – Authentication & Authorization ::: Passport strategies && JWT & sessions && Refresh token flow
- Module 8 – Microservices Module ::: TCP transport && Redis transport && NATS basics
- Module 9 – Caching & Queues ::: CacheModule && BullMQ for background jobs && Performance optimization
- Module 10 – Testing & Deployment ::: e2e tests with @nestjs/testing && Unit/integration testing && Docker deployment