Microservices using ASP.NET Core Web API

New Batch: 2nd May 2025, 8.30 PM to 10.00 PM IST.
Registration Link: https://forms.gle/7yiT3tZHrhMbWDTE9
Telegram Group: https://telegram.me/microservicesusingaspnetcore
Contact us through Our WhatsApp Number: +91 7021801173
Call us for More Details: +91 7021801173
Duration: 3 Months
Course Fees: 10000 INR or 125 USD
Demo Sessions: 2nd, 3rd, and 4th May, 2025, from 8.30 PM to 10.00 IST

Detailed Course Syllabus

Module 1: Foundations of Microservices (Beginner Level)

Goal: Get absolute clarity on "What, Why, and How" through simple examples and first hands-on projects.

Chapter 1: Application Architectures Overview

Students will understand different application architectures, compare Monolithic and Microservices systems, learn real-world motivations for migrating architectures, and recognize where Microservices fit.

- What is an application?
- Monolithic vs Microservices: Detailed Comparison
- Real-World Examples (Netflix, Amazon, Uber)
- Pros and Cons of Monolithic and Microservices
- Migration Drivers (When and Why to Migrate)

Example: Understand the architecture diagrams for Monolithic vs Microservices Architecture (example: E-Commerce app split into multiple services).

Chapter 2: ASP.NET Core Web API Fundamentals

Students will learn the basics of ASP.NET Core Web API development — setting up the environment, creating controllers, routing requests, connecting to a database with EF Core, and documenting APIs.

- .NET Core vs .NET Framework
- Setting up Development Environment (Visual Studio, VS Code, SDKs)
- First Web API Project Setup (Controllers, Routing)
- Models, DTOs, and Dependency Injection
- CRUD Operations with Entity Framework Core (DbContext, Migrations, Seeding)
- API Testing (Postman) and Documentation (Swagger)

Example to Develop: Build a Student Management CRUD API using ASP.NET Core Web API and Entity Framework Core with database seeding.

Chapter 3: Core Microservices Design Principles

Students will understand the principles that make Microservices successful, such as loose coupling, bounded context, and why Domain-Driven Design matters.

- Characteristics of Microservices
- Loose Coupling, High Cohesion
- Single Responsibility Principle
- Bounded Context (with Examples)
- Introduction to Domain-Driven Design (DDD)

Example to Develop: Identify different Bounded Contexts for an online shopping platform (e.g., Product Context, Payment Context, Order Context).

Chapter 4: Structuring Microservices Projects

Students will learn how to organize microservice solutions, build layered architecture, and understand API versioning strategies for backward compatibility.

- **Clean Solution/Project Structure**
- **Folder Layout, Layered Organization**
- **Repository and Unit of Work Pattern**
- **API Versioning Strategies (Path, Header)**

Example to Develop: Refactor the Student Management API into a clean structure: separate folders for Controllers, Services, Repositories, DTOs.

Module 2: Building Real Microservices (Intermediate Level)

Goal: Learn real-world practices for microservices communication, security, caching, and hosting.

Chapter 5: Building Your First Microservices

Students will develop two independent microservices (Product and Order), apply the repository pattern, seed initial data, and document the services using Swagger.

- **Creating Independent Product and Order Services**
- **Connecting to Separate Databases (Database per Service)**
- **Using Repository Pattern for Data Access Layer**
- **Documenting APIs with Swagger**
- **Unit Testing individual microservices with for controllers, repositories, and services.**

Example to Develop: Build two independent services:

- **Product Service (managing products)**
- **Order Service (managing customer orders)**

Write unit tests for controllers and services.

Chapter 6: Inter-Service Communication Basics

Students will learn how microservices communicate over HTTP REST, handle HTTP status codes, and get introduced to asynchronous messaging concepts.

- **RESTful Communication Best Practices**
- **HTTP Status Codes, Error Handling**
- **Introduction to Asynchronous Communication**

Example to Develop: Connect Product Service and Order Service via HTTP Calls using HttpClient.

Chapter 7: Data Management Strategies

Students will learn about distributed data management patterns including Database per Service, Eventual Consistency, and overview of distributed transaction management.

- **Database per Service Pattern**
- **Eventual Consistency (Real-World Examples)**
- **Introduction to Distributed Transactions (Saga Pattern and Two-Phase Commit).**
- **How to implement Saga patterns for managing distributed workflows.**

Example to Develop: Simulate an Order Placement flow that spans two microservices using Saga Pattern steps outlined manually.

Chapter 8: API Gateway Fundamentals

Students will learn what API Gateway is, why it is needed, and how to configure basic routing and security using Ocelot.

- **What is an API Gateway? (Simple Analogy)**
- **Why API Gateway is Important in Microservices**
- **Setting up API Gateway with Ocelot**
- **Routing, Aggregation, and Basic Security at Gateway**

Example to Develop: Implement an Ocelot API Gateway that routes traffic to Product and Order services.

Chapter 9: Basic Caching with Redis

Students will learn caching fundamentals, integrate Redis for distributed caching, and understand eviction policies for optimizing cache usage.

- **Caching Fundamentals**
- **Introduction to Redis**
- **Distributed Caching**
- **Caching API Responses**
- **Cache Invalidation Strategies and Eviction Policies (TTL, LFU, LRU).**

Example to Develop: Add Redis caching to Product Service for caching product list queries with cache expiry.

Chapter 10: Authentication and Authorization Basics

Students will learn to secure microservices using JWT tokens, role-based authorization, and policies in ASP.NET Core.

- **Introduction to Authentication vs Authorization**
- **Implementing JWT Authentication in ASP.NET Core**
- **Role-based Authorization**
- **Securing APIs with Authorization Policies**

Example to Develop: Secure the Product Service API using JWT Bearer Tokens with user roles like Admin, Customer.

Chapter 11: Messaging Fundamentals

Students will explore message-based asynchronous communication using RabbitMQ and distinguish between queues and streams.

- **Synchronous vs Asynchronous Messaging**
- **Introduction to RabbitMQ**
- **Setting up RabbitMQ in Local Development**
- **Sending and Receiving Messages (Simple Order Notification)**
- **Message Queues vs. Event Streams**

Example to Develop: Implement Order Placed Notification using RabbitMQ to notify a separate Notification Service when an order is placed.

Chapter 12: Introduction to Docker

Students will understand containerization concepts, write Dockerfiles, and run microservices in containers.

- **Containers vs Virtual Machines**
- **Installing and Setting up Docker**
- **Writing a Simple Dockerfile for .NET Core Applications**
- **Running Microservices in Docker Containers**

Example to Develop: Containerize both Product and Order services individually using Docker.

Chapter 13: Docker Compose Basics

Students will learn to orchestrate multiple containers (services + database) using Docker Compose.

- **Introduction to Docker Compose**
- **Running Multi-Container Setup (Services + Database)**
- **Networking between Containers**
- **Managing Environment Variables in Docker Compose**

Example to Develop: Use docker-compose.yml to run Product Service, Order Service, and a SQL Server database together locally.

Module 3: Advanced Microservices Concepts (Advanced Level)

Goal: Master architecture patterns, resilience, messaging, deployment, and real-time communication.

Chapter 14: Advanced Service Communication – gRPC

Students will learn about gRPC protocol, understand its advantages over REST, and implement basic bidirectional streaming communication.

- **REST vs gRPC: Differences and Use Cases**
- **Building Simple gRPC Communication Between Microservices**
- **gRPC Bidirectional Streaming.**

Example to Develop: Create a gRPC Service for sending real-time order status updates between services.

Chapter 15: Event-Driven Architecture

Students will learn about the importance of event-driven systems, using Event Bus patterns, and implementing event sourcing basics.

- **Events vs Commands vs Queries**
- **Introduction to Event Bus Pattern**
- **Event Sourcing**
- **Building Simple Event-Driven Microservices**
- **How Event-driven design helps decouple microservices, and why it's a preferred choice for modern systems.**

Example to Develop: Implement a Product Added Event that multiple services can listen to using a basic event bus.

Chapter 16: Using MassTransit for Advanced Messaging

Students will learn advanced messaging using MassTransit, including retries, fault handling, and routing.

- **What is MassTransit?**
- **Setting up MassTransit with RabbitMQ**
- **Implementing Retry, Timeout, and Message Patterns**
- **What is Message Routing?**
- **How MassTransit can be used for reliable messaging, especially in production systems?**

Example to Develop: Replace basic RabbitMQ code with MassTransit integration in Order and Notification Services.

Chapter 17: CQRS and Event Sourcing Basics

Students will learn how CQRS separates reads and writes, and how Event Sourcing can store all events instead of the final state.

- **Introduction to CQRS (Simple CRUD vs CQRS)**
- **Implementing Command Query Responsibility Segregation**
- **Basics of Event Sourcing (Store Events Instead of Data)**

Example to Develop: Build a CQRS pattern inside Order Service using separate Command and Query handlers.

Chapter 18: Microservices Security Best Practices

Students will learn deep authentication concepts, use IdentityServer for authentication, and apply OAuth2 and OpenID Connect.

- **OAuth2 and OpenID Connect Overview**
- **Implementing IdentityServer for Authentication**

- **Securing APIs with OAuth2, OpenID Connect, and JWT**

Example to Develop: Create a simple Identity Server for managing user login and access tokens for microservices.

Chapter 19: Caching Advanced Techniques

Students will explore advanced caching like cache-aside, write-through, and Pub/Sub models in Redis.

- **Cache-Aside Strategy**
- **Write-Through, Write-Behind Patterns**
- **Advanced Redis Usage (Pub/Sub Basics)**

Example to Develop: Implement Redis Pub/Sub for Cache Invalidation on Product Update.

Chapter 20: Resilience and Failure Handling

Students will build resilience into microservices using Polly for retries, circuit breaker patterns, and learn about chaos engineering.

- **Why Resilience Matters**
- **Retry, Timeout, Circuit Breaker Patterns (Polly Implementation)**
- **Introduction to Chaos Engineering Principles**
- **Exponential Backoff and Circuit Breaker Patterns using Polly and how it prevents cascading failures in distributed systems.**

Example to Develop: Use Polly to wrap HttpClient calls between services with Retry and Circuit Breaker policies.

Chapter 21: Observability and Monitoring

Students will learn centralized logging, tracing, and metrics collection techniques to make distributed systems observable.

- **What is Observability?**
- **Centralized Logging with Serilog + ELK Stack**
- **Distributed Tracing Basics: OpenTelemetry, Jaeger**
- **Metrics and Dashboards: Prometheus, Grafana**
- **Alerting Strategies for distributed systems.**

Example to Develop: Integrate Serilog + Seq or ELK stack for structured logging in Product and Order services.

Chapter 22: Kubernetes and Container Orchestration

Students will learn Kubernetes basics, set up a cluster, deploy microservices, and use Helm for deployment automation.

- **What is Kubernetes?**
- **Kubernetes Concepts: Pods, Services, Deployments**
- **Setting Up Kubernetes Cluster (Minikube / Azure AKS)**
- **Deploying Microservices to Kubernetes**
- **Helm Basics (Packaging Kubernetes Apps)**

Example to Develop: Deploy Product and Order services to a Kubernetes Cluster using YAML manifests.

Chapter 23: Real-Time Communication in Microservices

Students will implement real-time features (notifications, chat) using WebSocket protocols like SignalR.

- **Introduction to WebSockets and SignalR**
- **Implementing Real-Time Features (Notifications/Chat) using SignalR in ASP.NET Core.**
- **Using WebSocket with Microservices Architecture**

Example to Develop: Add SignalR notification when a new order is placed to connected clients.

Chapter 24: Advanced CI/CD Pipelines

Students will set up fully automated pipelines to build, test, and deploy microservices to cloud environments.

- **CI/CD Fundamentals**
- **Setting up Pipelines with Azure DevOps or GitHub Actions**
- **Automating Build, Test, Deploy for Microservices**
- **Continuous Deployment to Azure App Services / Kubernetes**

Example to Develop: Set up Azure DevOps pipeline for building, testing, and deploying Product and Order services.

Chapter 25: Clean Architecture and Advanced DDD

Students will apply Clean Architecture principles for separation of concerns and learn advanced Domain-Driven Design concepts.

- **Clean Architecture Principles (Separation of Concerns)**
- **Building Layers: Presentation, Application, Domain, Infrastructure**
- **Advanced Domain Driven Design Concepts**
- **Domain Events, Aggregates, Value Objects in Microservices**

Example to Develop: Refactor existing services into Clean Architecture Layers (Presentation, Application, Domain, Infrastructure).

Chapter 26: Steeltoe Framework for Microservices

Students will learn Steeltoe libraries for .NET Core to enable service discovery, dynamic configuration, and health checks.

- **Introduction to Steeltoe for .NET Core**
- **Centralized Configuration with Spring Cloud Config Server**
- **Service Discovery with Steeltoe**
- **How Steeltoe integrates with Spring Cloud for .NET Core applications in real-world microservices architectures**
- **Health Checks and Metrics provided by Steeltoe for resilient microservices.**

Example to Develop: Integrate Steeltoe Config Server and Discovery Client into Product and Order services.

Chapter 27: Service Mesh and Serverless Architectures

Students will understand service meshes for secure service-to-service communication and explore serverless architectures.

- **Introduction to Service Mesh (Istio, Linkerd Basics)**
- **Serverless Microservices (Azure Functions, AWS Lambda)**
- **Hybrid Architectures: Combining Containers and Serverless**

Example to Develop: Deploy an order notification as a Serverless Function on Azure using Azure Functions.

Chapter 28: GraphQL in Microservices

Students will learn how GraphQL APIs differ from REST APIs, build a GraphQL API in ASP.NET Core, and design schemas.

- **Why GraphQL over REST**
- **Introduction to GraphQL in ASP.NET Core**
- **Designing GraphQL Schema, Queries, and Mutations**
- **Introduce GraphQL vs REST:**

Example to Develop: Build a simple Product Query API using GraphQL instead of REST.

Chapter 29: Chaos Engineering and Fault Injection

Students will learn how to simulate failures to test system resilience using Chaos Engineering principles.

- **Chaos Testing Basics**
- **Simulating Failures to Build Resilient Systems**
- **Tools for Chaos Engineering (Gremlin, Chaos Monkey)**

Example to Develop: Simulate service failures using manual network disruptions and analyze the system's resilience behavior.

Chapter 30: Performance Tuning and Advanced Observability

Students will learn load testing, profiling, custom metrics collection, and advanced observability best practices.

- **Load Testing with JMeter, k6**
- **Profiling ASP.NET Core Applications (dotnet-trace, PerfView)**
- **Distributed Tracing with OpenTelemetry**
- **Custom Metrics and Advanced Alerts with Grafana**

Example to Develop: Perform Load Testing on Product API using k6 and create a basic dashboard in Grafana.

Chapter 31: Modern Deployment Strategies

Students will learn modern deployment patterns (Blue-Green, Canary, GitOps) for safe and efficient releases.

- **Blue-Green Deployment**
- **Canary Releases**
- **GitOps with FluxCD and ArgoCD**
- **Kubernetes Operators for Complex Deployment Workflows**

Example to Develop: Implement Canary Deployment strategy for updating Product Service in Kubernetes.

Note: If we missed any topics, if any new features are introduced, or if you want to learn any concepts not available in this Microservices using ASP.NET Core Web API Syllabus, please let us know, and that will also be included in this course. If you have any questions, please contact us.