**Microservice Architecture in Dotnet core Training Course Contents Day wise**

**Duration: 5 days (8 hrs per day)**

**Lab Set up:**

**Hardware Requirements:**

* CPU: 64-bit processor with virtualization support
* Memory: At least 12GB of RAM (16 GB recommended)
* Storage: At least 30 GB of free disk space
* Network: Good internet connectivity

**Software Requirement**

**Microservices, Docker and Azure Requirements.**

**Microservices Requirements:**

* Visual Studio 2022
* Visual Studio Code
* Programming languages and frameworks (.NET Core)
* API gateway
* Containerization platform (e.g. Docker, Kubernetes)
* Monitoring and logging tools
* Database system (e.g. SQL server, PostgreSQL, MongoDB)
* CI/CD tool (Jenkins, Git)

**Docker Requirements:**

* 64-bit operating system (Linux, Windows, or macOS)
* Docker engine (Docker Desktop for Mac or Windows, Docker CE for Linux)
* Sufficient CPU, memory, and storage resources to run containers
* Network connectivity for container communication and access to external resources
* Docker images and containers

**Azure Requirements:**

* Microsoft Azure account
* Subscription to Azure services (e.g. Azure Virtual Machines, Azure Container Instances, Azure Kubernetes Service)
* Sufficient CPU, memory, and storage resources to run services

**Network connectivity for service communication and access to external resources**

**Course Contents**

**Day 1**

**Module 1: Introduction to Microservices Architecture**

* What are Microservices
* Advantages of Microservices Architecture
* Principles of Microservices

Labs

* Kick start the e-shopping case study using microservice architecture.

**Module 2: Implementing Microservices Architecture using ASP.NET Core**

* Building RESTful APIs using ASP.NET Core – Only creation – no details – for swagger dependencies.
* Versioning API in Details as discussed
* Docker Concept with more than one service

Labs:

* Create Web Api and add versioning and host the same on docker

**Module 3: Documenting Microservices using Swagger**

* Overview of Swagger
* Adding Swagger to Microservices
* Testing APIs with swagger

Labs:

* Add Swagger in all microservices and test them through Swagger by hosting the same on docker.

**Day 2**

**Module 4: Decomposing Existing N-Tier application into Microservices (Concepts)**

* Identifying Monoliths and Decomposing Them
* Domain Driven Design Business driven design Transaction Driven
* Understanding Bounded Contexts

Labs: - Break down e-shopping Monolithic project into domain driven design / We can also use the above case-study.

**Module 5: Decomposing a centralized database into distributed databases (Basic concept)**

* Understanding the need for Distributed Databases
* Data Partitioning Techniques
* Implementing Distributed Transactions Saga Pattern

Labs: - Implement Saga pattern and distribute the databases

**Day 3**

**Module 6: Implementing Ocelot API Gateway**

* What is an API Gateway
* Overview of Ocelot
* Implementing an API Gateway using Ocelot features like Rate limiting and caching technoque
* Authentication and Authorization using Ocelot

Labs:-

* Implement API gateway using Ocelot and Add an Authentication layer . Implementing Rate limiting and caching either using own caching or Ocelot cache manager

**Module 7: Querying Microservices using CQRS pattern**

* Overview of CQRS pattern
* Implementing CQRS pattern using ASP.NET Core
* Implementing Query and Command services

**Labs: -**

* Implement CQRS pattern in project

**Day 4**

**Module 8: Implementing Asynchronous Communication - RabbitMQ**

* Overview of Asynchronous Communication
* Using RabbitMQ for Asynchronous Communication
* Handling Errors and Retries
* Logging

**Labs: -**

* Implement RabbitMQ in project

**Module 9: Implementing Observability Patterns: Logging, Tracing and Monitoring**

* Introduction to Observability Patterns
* Implementing Logging using Serilog
* Implementing Tracing using OpenTelemetry.
* Implementing Circuit Breaker and Retry patterns using Polly
* Implementing Health Checks
* Microservices Testing Strategies
* Microservices Security Best Practices

Labs: -

* Use Serliog package to add logging in project

**Module 10: Implementing Cross Cutting Concern Patterns: Service Discovery, Circuit Breaker and Retry patterns**

* Introduction to Cross Cutting Concern Patterns
* Implementing Service Discovery using Consul
* Best Practices for Microservices Architecture
* Anti-patterns to avoid

Labs: -

* Demo on patterns

**Day 5**

**Module 12: Microservices Deployment and Scaling**

* Deploying Microservices on Azure Kubernetes Service
* Scaling Microservices
* concept Blue/Green Deployment
* concept Implementing Canary Deployment

Labs: -

* Demo with Microservices

**Conclusion:**

* Recap of the course content
* Q&A Session
* Feedback and evaluation