.net core

1. Basics of .NET Core and C#

- **Objective**: Solidify your understanding of .NET Core and advanced C# features.
- **Topics**:
  - Overview of .NET Core vs .NET Framework
  - Setting up a .NET Core environment (CLI, Visual Studio, Visual Studio Code)
  - C# Basics (if needed):
    - Data types, control flow, and loops
    - Object-Oriented Programming (OOP) concepts (Encapsulation, Inheritance, Polymorphism, Abstraction)
    - Exception handling
  - Advanced C#:
    - LINQ (Language Integrated Query)
    - Async/Await and Task-based programming
    - Delegates, Events, and Lambdas
    - Dependency Injection (DI)

---

2. Building APIs with ASP.NET Core

- **Objective**: Learn to create robust and secure RESTful APIs.
- **Topics**:
  - Introduction to ASP.NET Core MVC and Web API
  - Creating Controllers and Actions
  - Routing (Attribute Routing vs Conventional Routing)

- Model Binding and Validation
- Returning JSON responses
- Middleware:
  - Custom middleware for logging and authentication
- Using Dependency Injection in Controllers
- Error handling and Logging (using Serilog or NLog)

---

3. Entity Framework Core

- **Objective**: Work with databases effectively using EF Core.
- **Topics**:
  - Database-first vs Code-first approaches
  - Setting up EF Core in a .NET Core project
  - CRUD operations using DbContext
  - Relationships (One-to-One, One-to-Many, Many-to-Many)
  - LINQ Queries with EF Core
  - Migrations:
    - Creating, applying, and managing migrations
  - Performance:
    - Query optimization (e.g., eager loading, lazy loading, and explicit loading)
    - Using `AsNoTracking` for read-only queries
  - Unit of Work and Repository patterns

---

4. Advanced API Development

- **Objective**: Create production-ready APIs.
- **Topics**:
  - Authentication and Authorization:

        - JWT (JSON Web Token)
        - Role-based access control
    - Filters:
        - Custom Action Filters for logging/validation
        - Exception Filters
    - API Versioning
    - Rate Limiting (using middleware or libraries like AspNetCoreRateLimit)
    - File Upload and Download APIs
    - Pagination and Sorting
    - Testing APIs:
        - Unit testing with `xUnit` or `NUnit`
        - Integration testing with `TestServer`

---

5. Frontend Integration

- **Objective**: Learn to integrate .NET Core APIs with front-end frameworks.
- **Topics**:
    - Basics of JSON data and how it is consumed
    - API integration using Postman
    - Working with Angular or React:
        - Connecting to APIs
        - Error handling on the client side
        - Displaying data from APIs

---

6. Middleware and Customizations

- **Objective**: Gain expertise in building and managing custom middleware.
- **Topics**:

- Writing custom middleware
- Understanding the request pipeline
- Built-in middleware (e.g., Authentication, Static Files, Routing)

---

7. Performance Optimization

- **Objective**: Learn techniques to make applications scalable and efficient.
- **Topics**:
  - Caching (In-Memory Cache, Distributed Cache, and Response Caching)
  - Asynchronous programming best practices
  - Minimizing API response times
  - Profiling and Debugging APIs
  - Optimizing database interactions (indexes, stored procedures)

---

8. Security Best Practices

- **Objective**: Secure APIs against common vulnerabilities.
- **Topics**:
  - Data validation to prevent SQL Injection
  - Securing sensitive information using configuration and secrets
  - HTTPS setup and enforcing SSL/TLS
  - Implementing authentication mechanisms (OAuth2, OpenID Connect)
  - Data encryption and hashing (e.g., BCrypt, AES)

---

9. Background Tasks and Real-Time Communication

- **Objective**: Learn to handle tasks running in the background and enable real-time features.
- **Topics**:
  - Background tasks using `IHostedService` and `Hangfire`

- SignalR for real-time communication (e.g., chat, live updates)

---

10. Testing and Debugging

- **Objective**: Ensure code reliability through testing and debugging.
- **Topics**:
  - Writing Unit Tests for Controllers, Services, and Repositories
  - Mocking dependencies using `Moq` or `NSubstitute`
  - Debugging techniques in Visual Studio and VS Code
  - Setting up CI/CD pipelines (optional for testing deployment automation)

---

11. Working with Cloud and Deployment

- **Objective**: Understand cloud concepts and how to deploy .NET Core apps.
- **Topics**:
  - Hosting ASP.NET Core apps on IIS and Kestrel
  - Publishing to Azure or AWS
  - Using Docker to containerize .NET Core apps
  - CI/CD pipelines with GitHub Actions or Azure DevOps

---

12. Advanced Topics (Optional)

- **Objective**: Broaden your skills for larger projects.
- **Topics**:
  - CQRS (Command Query Responsibility Segregation)
  - Event-driven architecture
  - Microservices with .NET Core
  - Working with gRPC for high-performance communication
  - MediatR library for managing requests

Practice Projects

- **CRUD API with Authentication**: Build an API for managing employee records with JWT-based authentication.
- **E-Commerce Backend**: Create APIs for product listing, shopping cart, and orders.
- **Blog Platform**: Implement features like creating, reading, and managing blogs with file upload and authentication.
- **Real-Time Chat App**: Use SignalR for chat functionality.
- **Background Tasks**: Implement a scheduled email sender using Hangfire.

Mastery Roadmap

1. Deep Dive into .NET Core Fundamentals

- **Goal**: Gain a deeper understanding of the internals of .NET Core and advanced C#.
- **Topics**:
  - CLR (Common Language Runtime):
    - Garbage Collection (GC) internals
    - Memory management (stack vs heap, finalizers, IDisposable)
    - Assembly loading and runtime behavior
  - Advanced C# Features:
    - Reflection and Expression Trees
    - Span<T> and Memory<T> for high-performance applications
    - Threading and Parallel Programming (Tasks, Threads, and `System.Threading.Channels`)
    - Value types vs Reference types, Boxing/Unboxing
  - Build and Debug Tools:
    - Custom build tools with MSBuild

        - Profiling applications with tools like PerfView or dotTrace

---

2. Advanced API Development

- **Goal**: Create APIs ready for high traffic and enterprise scenarios.
- **Topics**:
    - OData (Open Data Protocol) for queryable APIs
    - gRPC in .NET Core for high-performance communication
    - Implementing CQRS and Event Sourcing
    - API Gateway concepts (e.g., Ocelot)
    - Resiliency patterns:
        - Circuit Breaker, Retry, Fallback (using libraries like Polly)
        - Load Balancing strategies
    - Advanced middleware for custom scenarios
    - Real-world API testing:
        - Contract testing using tools like Pact
        - Performance testing with tools like k6 or JMeter

---

3. Enterprise Architecture and Design Patterns

- **Goal**: Learn scalable architecture principles and patterns.
- **Topics**:
    - DDD (Domain-Driven Design):
        - Aggregates, Entities, Value Objects
        - Bounded Contexts and Ubiquitous Language
    - SOLID Principles applied in .NET Core
    - Clean Architecture (Uncle Bob) and Onion Architecture
    - Microservices Architecture:

        - Service communication (REST vs Messaging)
        - Distributed transactions with Sagas
        - Service discovery (e.g., Consul, Eureka)
    - Mediator Pattern with `MediatR`
    - Modular Monoliths for large applications

---

4. Mastering Entity Framework Core

- **Goal**: Optimize and expand your knowledge of EF Core.
- **Topics**:
    - Advanced Query Optimization:
        - Compiled Queries for repetitive queries
        - Query plans and execution analysis
    - Bulk operations with third-party libraries (e.g., EFCore.BulkExtensions)
    - Handling concurrency with EF Core:
        - Optimistic and pessimistic concurrency
    - Temporal tables for tracking data changes
    - Multi-tenancy in EF Core
    - Writing complex stored procedures and mapping results in EF Core

---

5. Performance Optimization

- **Goal**: Build highly performant applications.
- **Topics**:
    - Memory profiling and reducing allocations
    - Optimizing database access:
        - Using Dapper for lightweight ORM scenarios
    - Async/Await for scaling I/O-bound tasks

    - Response compression and caching strategies
        - Using `ResponseCaching` middleware and Redis
    - Load testing and performance tuning:
        - Benchmarking with BenchmarkDotNet
        - Profiling .NET Core with Visual Studio Diagnostics or PerfView

---

6. Security and Compliance

- **Goal**: Build secure, compliant applications.
- **Topics**:
    - Advanced Authentication:
        - Implementing OAuth2 and OpenID Connect
        - IdentityServer integration for Single Sign-On (SSO)
    - Data encryption at rest and in transit:
        - Using Azure Key Vault or AWS Secrets Manager
    - Securing APIs against OWASP Top 10 vulnerabilities
    - GDPR and HIPAA compliance in .NET applications
    - Token refresh strategies for long-lived user sessions

---

7. Cloud-Native Development

- **Goal**: Master deployment and scalability for cloud environments.
- **Topics**:
    - Dockerizing .NET Core applications:
        - Multi-stage Docker builds
        - Using Kubernetes for orchestration
    - Deploying to cloud platforms:
        - Azure App Services, Azure Functions

        - AWS Elastic Beanstalk or Lambda
        - Google Cloud Run or Kubernetes Engine
    - Serverless architecture with Azure Functions or AWS Lambda
    - Monitoring and logging:
        - Application Insights, Prometheus + Grafana
        - Distributed tracing with Jaeger or OpenTelemetry

---

8. Real-Time Applications

- **Goal**: Build dynamic, real-time features.
- **Topics**:
    - SignalR Advanced:
        - Scaling SignalR with Redis backplane
        - Using SignalR in distributed systems
    - WebSockets with ASP.NET Core
    - Real-time notifications and data streaming
    - Event streaming using Kafka or Azure Event Hubs

---

9. Advanced Testing and CI/CD

- **Goal**: Create robust, maintainable applications with automated pipelines.
- **Topics**:
    - Unit testing:
        - Mocking with Moq and AutoFixture
        - Writing testable code (DI, mocking dependencies)
    - Integration testing:
        - Using TestServer in ASP.NET Core
        - Testing APIs in microservices architecture

- Automated deployment:
  - Setting up CI/CD pipelines (Azure DevOps, GitHub Actions, Jenkins)
- Load testing APIs with Apache JMeter or k6
- Writing BDD tests with SpecFlow

---

10. Open Source and Community Contributions

- **Goal**: Give back to the community and learn from real-world scenarios.
- **Topics**:
  - Contributing to open-source projects (e.g., OrchardCore, IdentityServer)
  - Building your own libraries or NuGet packages
  - Writing technical blogs or presenting on .NET topics
  - Engaging in community events like .NET Conf, local meetups, or hackathons

---

11. Mastery in Tooling

- **Goal**: Fully leverage tools to enhance productivity.
- **Topics**:
  - Visual Studio:
    - Productivity shortcuts
    - Refactoring tools and Live Unit Testing
  - dotnet CLI for advanced tasks
  - Profiling and debugging tools:
    - Visual Studio Profiler
    - Rider's performance analysis
  - IDE plugins (Resharper, CodeRush)

---

12. Advanced Projects

To solidify your mastery, work on **enterprise-level projects**:

1. **E-Commerce Platform**:
    - Microservices architecture
    - Payment gateway integration
    - Real-time inventory updates with SignalR
2. **Social Media App**:
    - Real-time chat and notifications
    - Event streaming for user activity
    - Implementing GraphQL APIs
3. **Job Scheduling System**:
    - Background tasks with Hangfire
    - Multi-tenant architecture
    - Advanced reporting and analytics