codewithmukesh

level up your career 🚀

.NET BACKEND DEVELOPER ROADMAP

2024 EDITION

mukesh murugan
@iammukeshm

codewith**mukesh**

Learning Developer Basics

Make sure that you are aware of the very basics as listed below.

- Understand how the internet works.
- HTTP / HTTPS Protocols.
- API Concepts.
- HTTP Status Codes.
- Learn **Object Oriented Programming** Concepts.
- Improve your Problem-Solving Skills.
- Learn Basic Data Structures & Algorithms.
- Be Fluent in looking up solutions on the Internet.
- Become Comfortable with **ChatGPT** Prompts
- Learn **GIT** in depth.
- Explore **GitHub** with some practical usage.

mukesh murugan
@iammukeshm

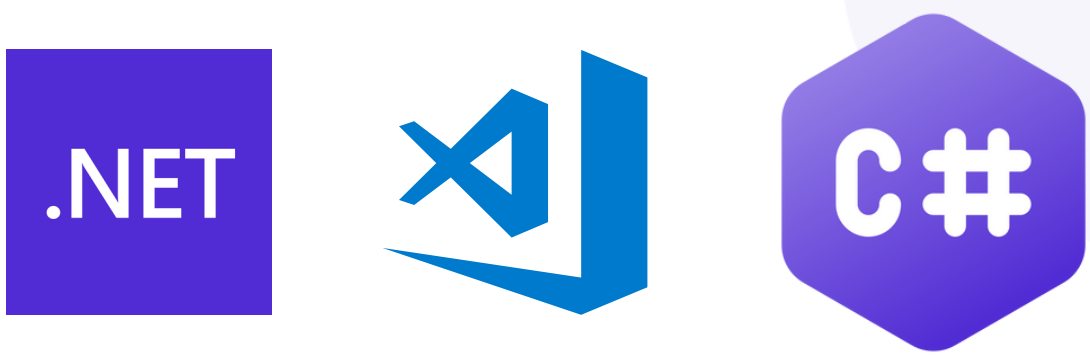
for beginners only

codewithmukesh

Into the World of .NET

To get started with .NET, learn the following basics.

- You should learn about the .NET Ecosystem.
- Learn **Basic C# Syntaxes and Language Features**. **C# 12** is the latest language version that shipped with .NET 8.
- Start with the ground basics, by building a simple .NET Console App. Learn about variables, if, else, while, and all the basic stuff.
- Become an expert in using .NET CLI commands.
- Master **IDEs like Visual Studio** or Code Editors like Visual Code.
- Understand NuGet Package Management.
- Focus only on .NET 6 or above. Ideally, .NET 8 is what you should be looking at.



mukesh murugan
@iammukeshm

for beginners only

codewith**mukesh**

Advanced C#

Now that you are familiar with the basics, let's get going.

- Learn the Basics of Clean Code Practices.
- Understand various C# Project Templates. Use .NET CLI!
- Practice Writing De-Coupled Code.
- Understand Interfaces, Dependency Injection, and Dependency Inversion.
- **Learn SOLID Principles.**
- Design Patterns are important. Be aware of at least a few.
- Engage in Peer Reviews / Open Source Contributions to get more feedback.
- Do Not Complicate Code, **Keep it Simple, Silly (KISS).**
- Read a lot of code! There are tons of open-source C# projects on GitHub!
- **Write Unit Tests!**

mukesh murugan
@iammukeshm

codewithmukesh

Database Fundamentals

A very crucial step in your learning path.

- Learn about Database design and related concepts.
- You need to know SQL Syntaxes.
- Add Stored Procedures too! (you might not use this regularly, but this is good to know).
- Understand Relational vs NoSQL Databases, and when to choose what.
- Learn about Database Optimizations.
- Work with RDMS like PostgreSQL, MSSQL, etc, and NoSQL Databases like MongoDB, AWS DynamoDB, Azure ComsosDB, etc. **I prefer PostgreSQL and AWS DynamoDB**.

mukesh murugan
@iammukeshm

codewith**mukesh**

ASP.NET Core - Powerhouse!

Focus on ASP.NET Core and Web Development! We would use this tech to build powerful APIs.

- **Create a sample ASP.NET Core Web API** and learn the basics.
- Understand the lifecycle of a request within an ASP.NET Core application.
- Learn about Controllers and **Minimal APIs**.
- Understand Middlewares.
- Routing.
- Filters and Attributes.
- Various API Architectures like **REST API**, GraphQL, and gRPC.
- REST API Conventions.
- Understand **MVC** Pattern.
- How to load Configurations and **IOptions** pattern.
- Service Lifecycles (Scoped, Transient & Singleton), and Dependency Injection.
- **Authentication & Authorization** (We will have a separate page for this)
- Extension Methods.
- Exception Handling with IExceptionHandler.

mukesh murugan
@iammukeshm

codewithmukesh

ORM : Object Relational Mapping

A technique to create a bridge between your C# classes and an actual database. In other words, a reliable way to integrate your .NET applications with a database.

- **Entity Framework Core (Highly Recommended)**
  - **Code First**, Database First Approaches.
  - Migrations & .NET EF CLI Commands.
  - **LINQ** is your friend.
  - Various Data Loading Strategies.
  - DbContext, Query Filters, and Schemas.
  - Interceptors in **EF Core**.
- Dapper, if you are looking for a more RAW way to execute your SQL Commands and Queries.
- MongoDB, AWS DDB, or other NoSQL database driver and integration.

mukesh murugan
@iammukeshm

codewith**mukesh**

Authentication & Authorization

A very crucial step in your learning path.

- Understand **why you need** Authentication and authorization.
- Spin up your own simple Auth system in ASP.NET Core WebAPI by allowing a user to send in an **email/password**, try to encrypt the password, and store it in a database.
- Learn **ASP.NET Core Identity.**
- **Identity Endpoints in .NET 8.**
- **Users, Roles, and Permissions.**
- Learn about Cookies.
- JSON Web Tokens (JWT).
- Identity Server.
- OpenID Authentication.
- **OAuth 2.0** protocols and different auth flows.
- Learn about OAuth providers like **KeyCloak**, **Auth0**, AWS Cognito, or Azure AD. You will be probably using this in a future project.
- Claims and Permissions Management.
- Multitenancy with **Finbuckle.**

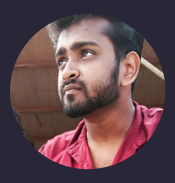
mukesh murugan
@iammukeshm

codewithmukesh

System & Solution Designs

Now that you have a basic understanding of building a .NET application, focus on learning system design concepts.

- Start building applications that are easy to maintain and readable.
- Develop good system design knowledge to break down systems into smaller pieces and solve problems.
- Learn about Monolith and Microservices, and understand when to use them.
- Event Driven Architecture.
- Practice Clean Architecture with separation of concerns.
- **Vertical Slice Architecture.**
- **Domain Driven Concepts** - Domain Events, Value Objects, Rich vs Anemic Domain Modelling, Bounded Contexts.



mukesh murugan
@iammukeshm

codewith**mukesh**

Microservices!

To build truly decoupled systems, you need to know about Microservices.

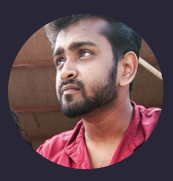
- When to use Microservices.
- The Tradeoffs and Perks.
- Communication Between Services using Message Brokers & Buses like RabbitMQ, or Kafka.
- **Masstransit**.
- Event Sourcing and Eventual Consistency.
- Observability.
- Containerization and Orchestration.
- Secrets.
- API Gateways like **YARP**, Ocelot.
- Load Balancing.
- And a lot more!

mukesh murugan
@iammukeshm

codewithmukesh

TIP!

Never start with Microservices! In most of the cases, a well-designed Monolith is what you would need.

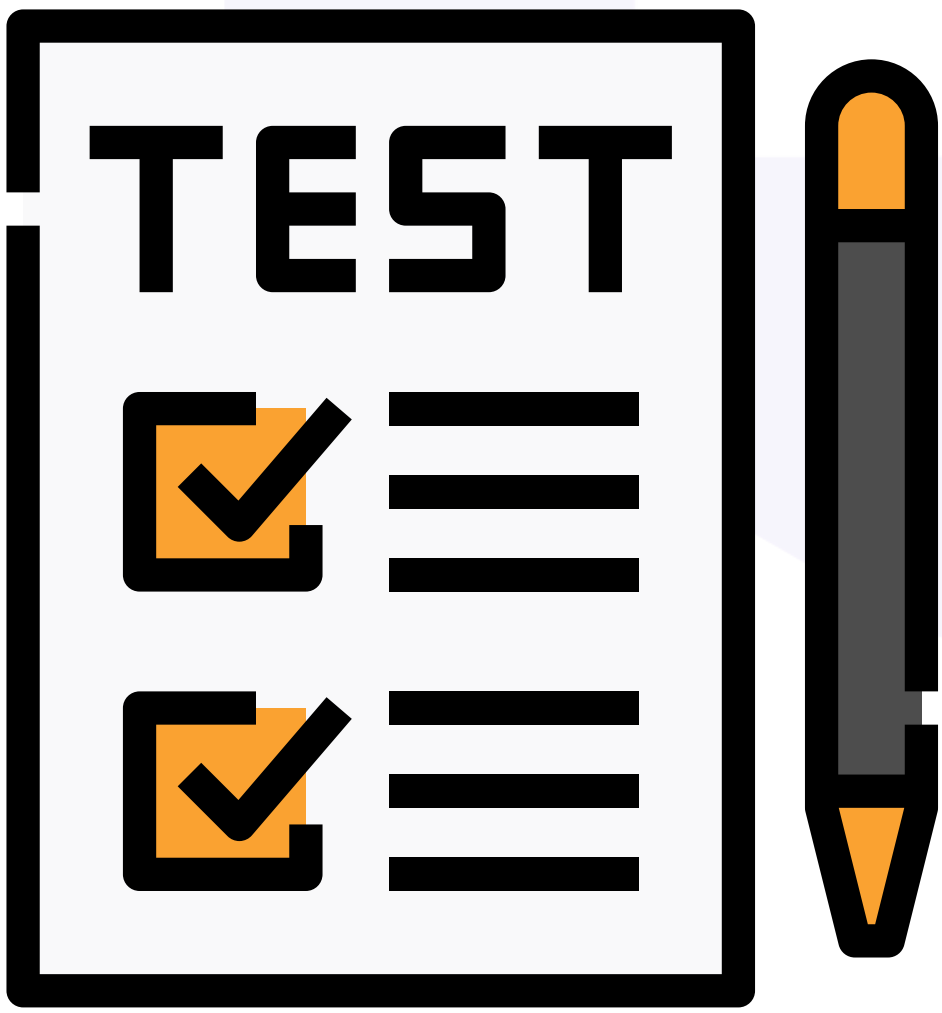


mukesh murugan
@iammukeshm

Testing

This is something we often tend to skip or put for later. But make sure that you write your test cases.

- Unit Testing.
- Integration Testing.
- xUnit / NUnit NuGet Packages.
- **Fluent Assertions (Recommended).**
- Bogus for dummy data generation.
- K6 for load testing.
- Test Driven Programming. Make your code testable and ensure you have good code coverage.



mukesh murugan
@iammukeshm

codewithmukesh

Caching

A crucial feature to improve your Web Application’s performance.

- Learn various concepts about caching.
- Cache Invalidation Techniques.
- Cache Expirations - Sliding Window, Absolute Expiration.
- In Memory / In Process Caching.
- Distributed Caching with Redis.
- Application Level Caching (Response Caching, EF Caching)

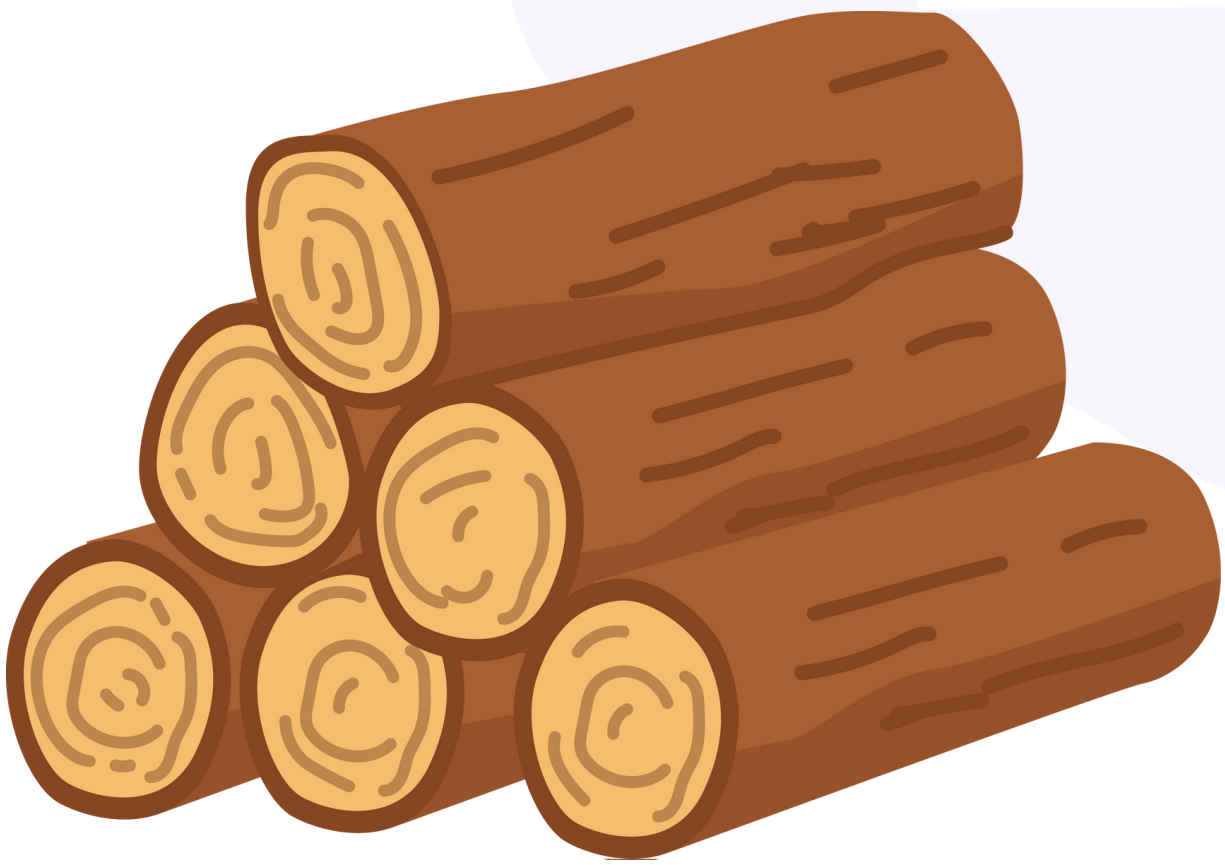
mukesh murugan
@iammukeshm

codewithmukesh

Logging

Enhance Your Application's Observability and Bug Tracking.

- ILogger Interface from Microsoft
- **Serilog**: This popular logging package is often the only one you'll ever need and is my **go-to tool** for every new C# project I work on!
  - Learn about various Sinks.
  - Structured Logging.
  - Serilog SEQ Logging for Development purposes.
  - Serilog Configurations and CoRelationId
- NLog: a good alternative.
- Learn about Kibana and the ELK stack too!



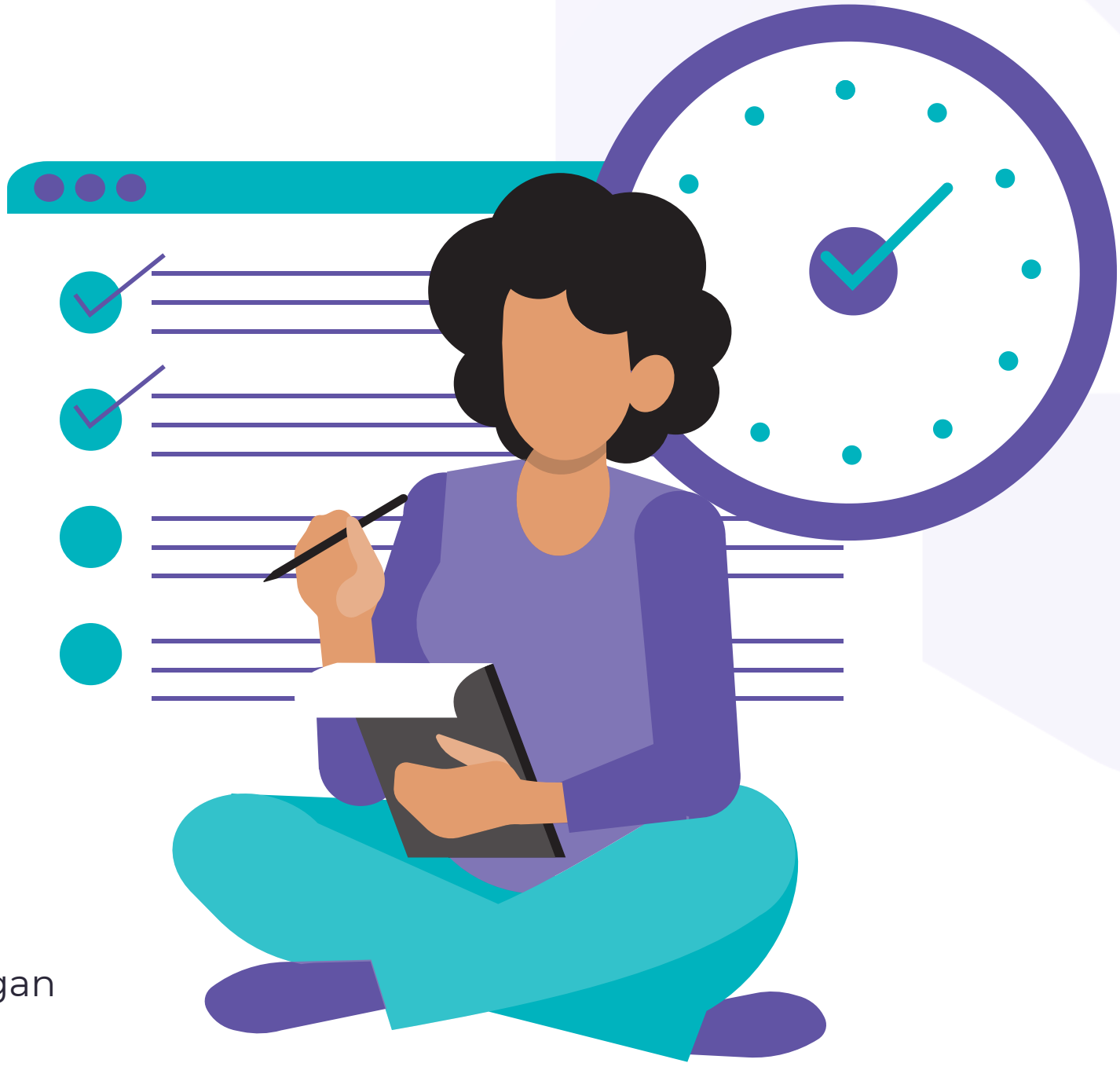
mukesh murugan
@iammukeshm

codewithmukesh

Background Tasks

Every application would require background tasks at some point in time.

- Native **IHostedService** and BackgroundService interface for simple task scheduling.
- CRON concepts.
- **Hangfire**.
- Quartz.



mukesh murugan
@iammukeshm

codewithmukesh

Good to Know Libraries.

- Use Refit for HTTP calls.
- **FluentValidation** to validate incoming requests.
- ProblemDetails for organized Error Throws.
- Use Polly for the Retry Mechanism.
- SignalR for real-time communication.
- API Versioning.
- **Scrutor** for Automated Dependency Injection.
- **Carter** for better Minimal API routing.
- AutoMapper/Mapster/Mapperly for Object Mapping.
- Sonar Analyzers.
- OpenIddict.
- YARP Reverse Proxy.
- **Learn CQRS Pattern with the MediatR library.**
- Open API / **Swashbuckle** for API Documentation.
- Benchmark.NET for evaluating your application performance.

mukesh murugan
@iammukeshm

codewithmukesh

Beyond C#

It takes more than just a single technology to excel as a developer in 2024 and beyond!

mukesh murugan
@iammukeshm

codewithmukesh

Containers & Orchestration

Very important for backend developers to make your deployments so much easier.

- Learn about the benefits of Docker and containerization.
- Learn Docker & Docker Compose.
- Dockerfile.
- Built-in containerization feature from .NET 7 and above (you no longer need a Dockerfile).
- Multi-Stage Docker Builds.
- Docker Hub & CLI commands.
- Docker Networking.
- Try to build a docker image of your .NET application, push it to an image repository like DockerHub, and Run the container locally or on Cloud Platforms like AWS or Digital Ocean.
- Have an understanding of Kubernetes and kubectl commands.

mukesh murugan
@iammukeshm

codewithmukesh

Observability

Building large systems requires you to have an efficient way to track your system state, logs, and metrics.

- Monitoring
  - ELK Stack
  - Serilog + SEQ
  - Prometheus
  - Graffana Dashboards
  - Open Telemetry in .NET
- Cloud Logging Providers like Cloudwatch.
- Metrics Collection and Cloud Insights.

mukesh murugan
@iammukeshm

codewithmukesh

Cloud

Cloud is unavoidable in 2024 and beyond. Almost all B2B and B2C SaaS products are hosted on a Cloud Provider like AWS, Azure, or GCP.

- At a senior level, it is mandatory to have a good grasp of cloud technologies.
- AWS has a better market share.
  - Learn AWS Lambda, S3, EC2. ECS, Kinesis, IAM, SQS, SNS, DynamoDB, and other core services.
  - Serverless Knowledge is crucial.
  - Get yourself cloud-certified.
  - Have good experience in designing cloud-based systems and infrastructure.
- Once you are familiar with Manual deployments into the cloud, learn to automate it.
- Learn Automated Cloud Infrastructure Deployments or IAC Tools like Terraform, AWS CDK, or Pulumi.

mukesh murugan
@iammukeshm

codewithmukesh

CI/CD

CI/CD (Continuous Integration/Continuous Deployment) is a software development practice that involves automating the process of building, testing, and deploying code changes.

- GitHub Actions (Easy to get started, and it's FREE).
- AWS Code Pipeline
- Azure Pipeline
- Jenkins
- Learn Scripting / Python for automation tasks.
- Learn to Create Deployment / Build Pipelines that run tests on the cloud and deploy them onto your infrastructure/containers / virtual machines.

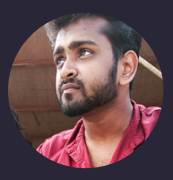
mukesh murugan
@iammukeshm

codewithmukesh

Improve your Soft Skills!

Improving soft skills is essential for developers, as it not only enhances collaboration and communication but also contributes to overall professional success.

- Work on expressing thoughts and ideas concisely and understandably.
- Spend time reading documentation.
- Develop a system to prioritize and manage tasks efficiently.
- Build an online presence, and improve your network.
- Learn from the Top Content Creators in your space.
- Build a portfolio website for yourself. Start writing blogs, and make sure to document your progress. This might be helpful to someone.
- Build a Solid Resume.
- Stay open to new technologies, methodologies, and ways of working.
- Enhance your analytical skills to approach problems logically and systematically.
- Volunteer for leadership roles or lead small projects to build leadership skills.
- Keep up with industry trends and advancements to remain relevant and valuable.



mukesh murugan
@iammukeshm

codewithmukesh

Congrats!

You are just one interview away!

mukesh murugan

@iammukeshm

codewithmukesh

LEARNING NEVER REALLY STOPS!

Keep adding more to your roadmaps and stay up to date

mukesh murugan
@iammukeshm

codewithmukesh

BONUS

You will learn only by building applications! Make sure that you implement concepts as soon as you learn them. Build your Solution Templates to ease your development time. Refer to the following projects on GitHub to learn code implementations with clean architecture.

- **Full Stack Hero - .NET 8 Web API Starter Kit (Search on Google)**
- eShopOnWeb & eShopOnContainers.

mukesh murugan
@iammukeshm

codewithmukesh

Subscribe to my .NET Newsletter!

I will be starting a **.NET 8 Zero to Hero** Series soon via my Newsletter, for **FREE**.

Join the waitlist by Subscribing.

Link in the Description!

mukesh murugan
@iammukeshm

codewithmukesh

WAS THIS HELPFUL?

Share with a friend who needs it!

mukesh murugan

@iammukeshm

Follow me for more!