codewithmukesh

LEVEL UP YOUR CAREER 🚀

.NET BACKEND DEVELOPER ROADMAP

2025 EDITION

mukesh murugan
@iammukeshm

codewithmukesh

The Developer Fundamentals

Build a strong foundation by mastering these essentials:

- Understand how the internet works, including HTTP/HTTPS and APIs.
- Learn common HTTP status codes (e.g., 200, 404, 500).
- Master object-oriented programming (OOP) principles.
- Enhance problem-solving skills and basic data structures & algorithms.
- Refine research skills and optimize ChatGPT (and other AI tools) usage. This is very important.
- Learn Git in-depth and explore GitHub through practical projects.

These basics will set you up for success in advanced topics.

mukesh murugan
@iammukeshm

for beginners only

codewithmukesh

Into the World of .NET

To get started with .NET, focus on these essentials:

- Explore the .NET Ecosystem: Understand its framework, runtime, libraries, and use cases.
- Learn Basic C# Syntax: Master language features, starting with C# 12 (the latest LTS version in .NET 8).
- Start Simple: Build a basic .NET console app to grasp variables, conditionals, loops, and fundamentals.
- Master the .NET CLI: Learn key commands for project creation, build, run, and publish.
- Use the Right Tools: Get comfortable with IDEs like Visual Studio or editors like VS Code.
- Understand NuGet: Learn to manage dependencies and packages effectively.
- Focus on Modern .NET: Prioritize working with .NET 6 / 8 or above (strictly). At the same time, be aware of features that shipped with .NET 9!

mukesh murugan
@iammukeshm

for beginners only

codewithmukesh

Advanced C#

Now that you've mastered the basics, take the next steps:

- Adopt Clean Code Practices: Focus on writing clear, maintainable, and efficient code.
- Explore C# Project Templates: Familiarize yourself with templates using the .NET CLI.
- Write Decoupled Code: Emphasize modularity and separation of concerns.
- Master Key Concepts: Learn interfaces, dependency injection, and the dependency inversion principle.
- Understand SOLID Principles: Apply these for better design and maintainability.
- Learn Design Patterns: Start with a few essential ones, like Singleton, Factory, and Observer.
- Engage in Feedback: Participate in peer reviews or contribute to open-source projects.
- Keep It Simple, Silly (KISS): Avoid overengineering; prioritize simplicity.
- Read Code: Explore open-source C# projects on GitHub for inspiration.
- Write Unit Tests: Ensure code reliability and catch bugs early.

These steps will elevate your .NET development skills significantly.

mukesh murugan
@iammukeshm

codewithmukesh

Database Fundamentals

A critical step in your learning journey is mastering databases:

- Understand Database Design: Learn normalization, relationships, and schema design.
- Learn SQL Syntax: Master CRUD operations, joins, and advanced queries.
- Explore Stored Procedures: While not always essential, they're valuable for specific use cases.
- Relational vs NoSQL Databases: Understand the differences and when to choose each.
- Optimize Databases: Learn indexing, query optimization, and performance tuning.
- Work with Popular Databases: Gain hands-on experience with relational databases like PostgreSQL and MSSQL, and NoSQL options like MongoDB, DynamoDB, or CosmosDB.
- **I prefer working with PostgreSQL for relational and AWS DynamoDB for NoSQL, as they offer a great balance of features and scalability.**

Mastering these concepts will provide a strong foundation for handling data in your applications.

mukesh murugan
@iammukeshm

codewithmukesh

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

To integrate your C# classes with a database, focus on these key techniques:

- Entity Framework Core (EF Core): Use it for seamless data access and integration.
- Code First & Database First: Learn both approaches to database schema management.
- Migrations & EF CLI: Master migrations and .NET EF CLI commands for database versioning.
- LINQ: Leverage LINQ for querying data in a concise and readable way.
- Data Loading Strategies: Understand eager, lazy, and explicit loading.
- DbContext & Query Filters: Use DbContext for managing entities and query filters for reusable queries.
- Interceptors: Customize and extend EF Core behaviors using interceptors.
- Dapper: For raw SQL queries and more control over performance.
- NoSQL Integration: Work with MongoDB, DynamoDB, or other NoSQL databases as needed.

These techniques will help you efficiently work with databases in .NET applications.

mukesh murugan
@iammukeshm

codewithmukesh

Authentication & Authorization

Authentication and authorization are crucial for securing your applications. Here's how to get started:

- Understand Authentication & Authorization: Learn why they are essential for securing APIs.
- Implement a simple ASP.NET Core Web API for email/password authentication, encrypting passwords and storing them securely.
- Dive into built-in Identity functionality for user management.
- Identity Endpoints (ASP.NET Core 8):
- Users, Roles, and Permissions: Learn how to manage users, roles, and access control.
- Cookies & JWT: Understand session management with cookies and token-based auth using JWT.
- Identity Server: Explore IdentityServer for handling more complex authentication scenarios.
- OpenID Connect & OAuth 2.0: Learn these protocols and different authentication flows.
- OAuth Providers: Work with services like **Keycloak**, Auth0, AWS Cognito, or Azure AD for authentication in real-world projects.
- Claims & Permissions: Manage user claims and permissions for fine-grained access control.
- Multitenancy with Finbuckle: Understand how to implement multi-tenant architectures in your applications.

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, Microservices, and Modular Monoliths, 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

codewithmukesh

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

Modular Monoliths!

The application remains a single deployable unit in a modular monolith, but the code is organized into distinct modules.

Each module has its domain, clear boundaries, and internal implementation details hidden from others. This approach provides the clarity and maintainability benefits of microservices while retaining the simplicity of a monolithic deployment.

I prefer to use this architecture before inching towards Microservices!

The modular monolith shines because it addresses real-world development challenges pragmatically.

mukesh murugan
@iammukeshm

codewithmukesh

TIP!

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

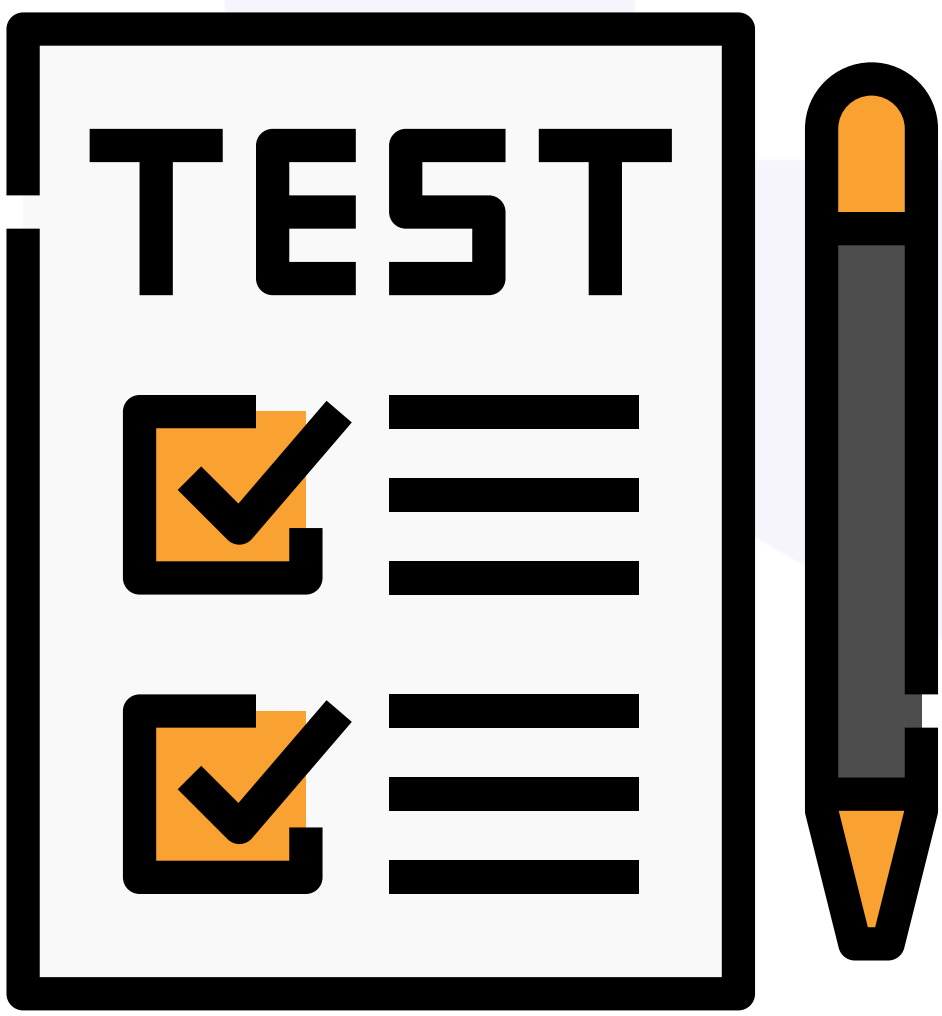
mukesh murugan
@iammukeshm

codewithmukesh

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.
- Learn about TestContainers to test your code / infrastructure with docker containers!



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.
- Hybrid Caching (.NET 9 onwards). This is the future!
- Application Level Caching (Response Caching, EF Caching)
- Various Cache Providers like Redis, AWS ElastiCache.

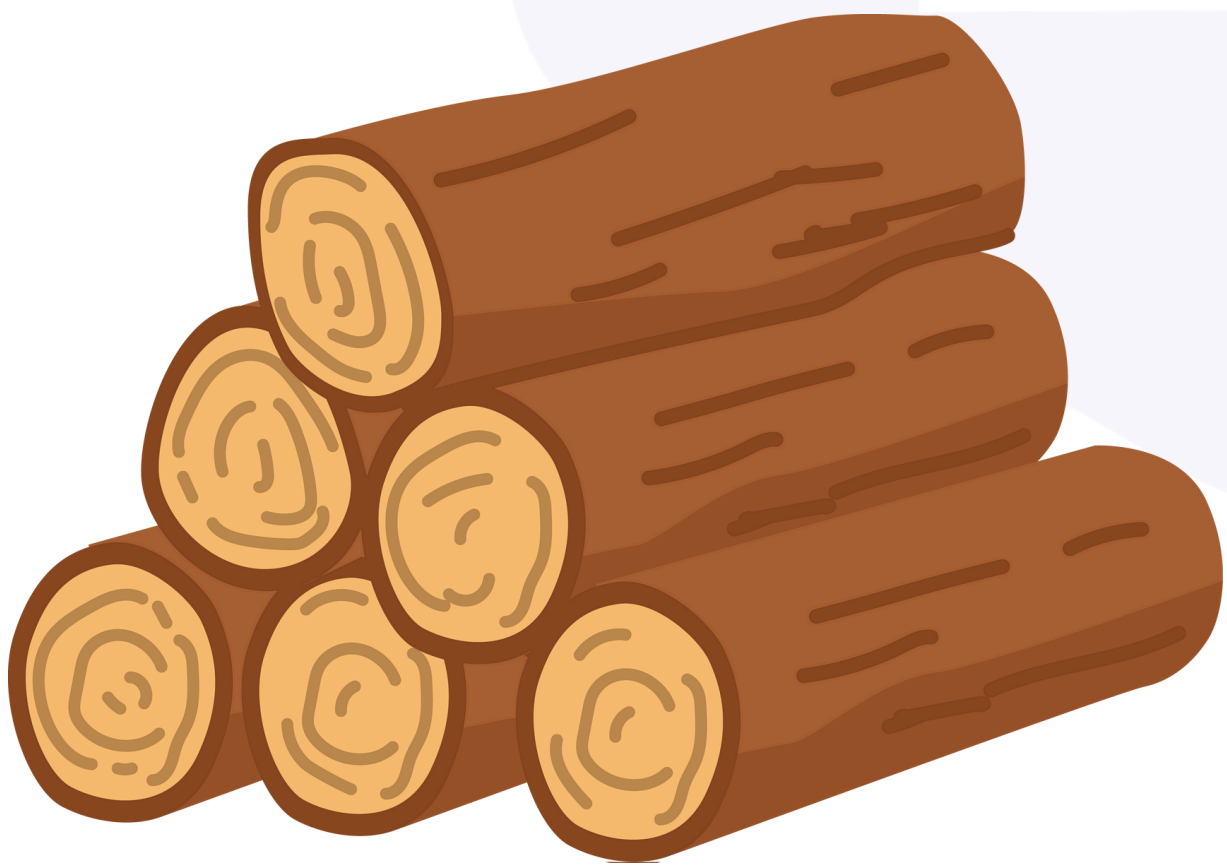
mukesh murugan
@iammukeshm

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 / HttpClientFactory 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.
- Mapster/Mapperly for Object Mapping.
- Sonar Analyzers.
- OpenIddict.
- YARP Reverse Proxy.
- **Learn CQRS Pattern with the MediatR library.**
- Open API / **Scalar** for API Documentation. (Swagger is no longer included as a default package .NET 9 onwards)
- 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 2025 and beyond!

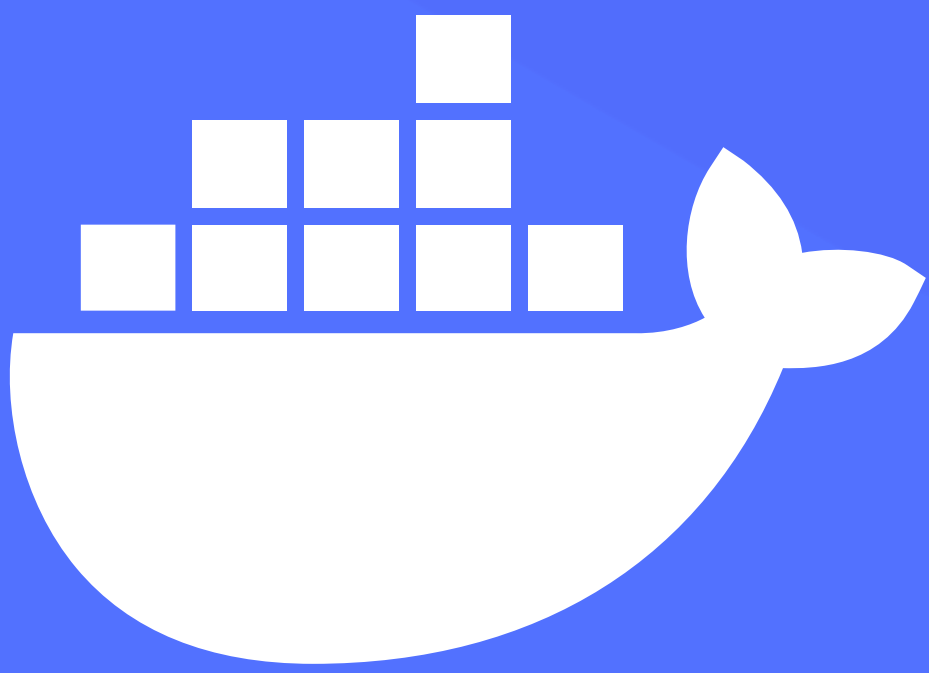
mukesh murugan
@iammukeshm

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. (it's good to know although you might move to Aspire)
- 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 (optional).



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.
- OpenTelemetry Collectors.

mukesh murugan
@iammukeshm

codewithmukesh

Cloud

Cloud is unavoidable in 2025 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

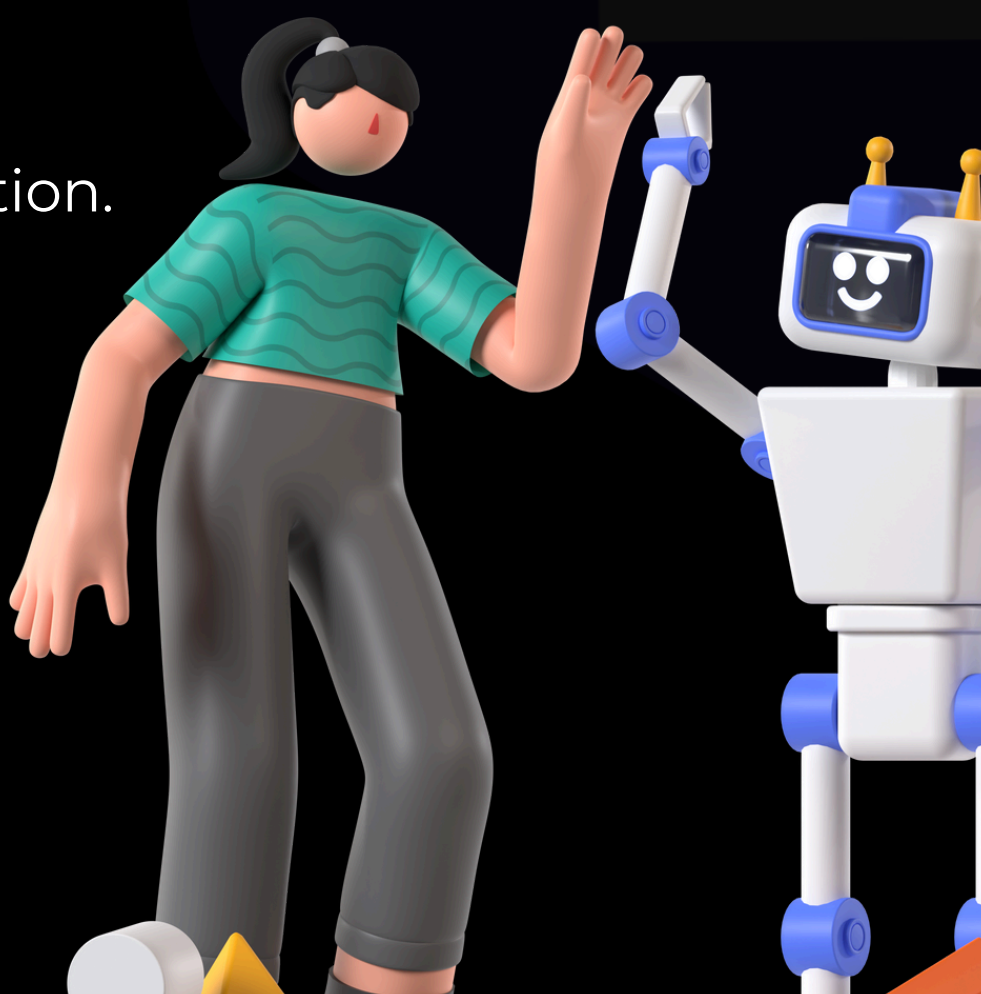
Learn to work with AI!

"AI won't replace you, but developers who know how to work with AI will!"

Nowadays, integrating AI tools into your workflow is not just a luxury but a necessity for staying competitive and productive. With the rapid advancements in AI technology, tools are becoming more accessible, versatile, and capable of automating mundane tasks, analyzing large datasets, and enhancing decision-making processes.

AI Tools can help generate source code, explain code, generate documentation, and help solve problems way easier for developers. It’s all about learning to ask the right questions.

- ChatGPT / Claude.
- GitHub CoPilot.
- Cursor AI for source code generation.



mukesh murugan
@iammukeshm

codewithmukesh

Congrats!

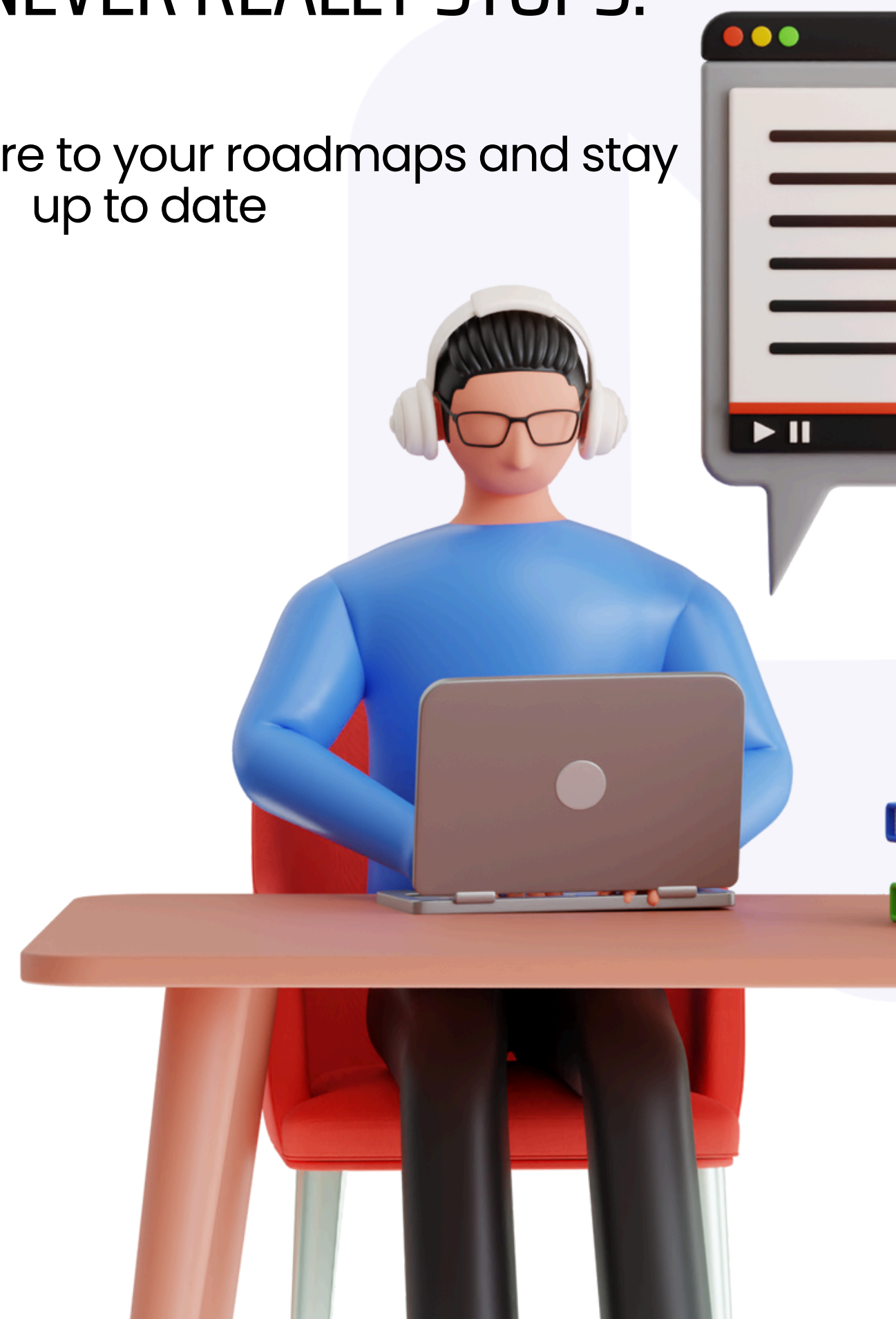
You are just one interview away!

mukesh murugan
@iammukeshm

codewithmukesh

LEARNING NEVER REALLY STOPS!

Keep adding more to your roadmaps and stay up to date



mukesh murugan
@iammukeshm

codewithmukesh

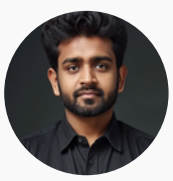
.NET Zero to Hero Advanced Course (FREE)

Build Production-Ready .NET Solutions with Confidence and Level Up Your Career.

We'll begin with the fundamentals and progressively dive into advanced topics such as Authentication, Logging, OpenTelemetry, Validation, CQRS, Vertical Slice Architecture, Docker, integrating External Identity Providers, and much more!

By the end of this course, you'll have the expertise to design, build, and deploy robust, scalable .NET Web APIs like a pro.

Course Link in Description!



mukesh murugan
@iammukeshm

codewithmukesh

WAS THIS HELPFUL?

Share with a friend who needs it!

mukesh murugan
@iammukeshm

Follow me for more!