ASP.NET 8 (core) & Angular Bootcamp

מבנה ומתכונת הקורס

- **520 שעות לימוד בהתאם לפירוט הבא:**
  - 400 שעות לימוד אקדמיות
  - ב- 120 שעות עבודה עצמאית על פרויקטים ומטלות

Overview:

NET 8 is the latest major release of .NET. The main programing language of .net is C#. In .NET 8 you can use C# Version 12.0 with all the improvements.

Angular is the next big deal, Angular 18 is faster and offers a flexible and modular development approach.

In this course, we will learn the best practices of ASP.NET, Angular and we will build a full application that use all this tools and features.

Course Contents:

Module Title	Hours
Flowchart	36
Basic .net & C#	32
OOP with C#	32
SQL	32
Entity framework 8	32
Advanced ASP.NET 8	32
Working with git	8
Docker & containers	24
Developing microservices with C# & .NET 8	40
HTML & CSS	36
Basic javascript	12
Typescript	32
Angular 18	52
Total	**400**

Module Title	Module Description
Flowchart	▪ Importance of flowcharts in problem-solving and algorithm design. ▪ Basic symbols and shapes used in flowcharts (e.g., start/end, process, decision, input/output, arrows). ▪ Introduction to variables and their representation in flowcharts. ▪ Using decision diamonds to represent if-else conditions and branching logic. ▪ Representing loops (while, for) in flowcharts, using connectors for repetitive processes. ▪ Representing reusable code blocks or procedures in flowcharts.
C#	▪ Overview of C# and .NET ▪ Setting up the development environment ▪ Structure of a C# program ▪ Compilation and execution process ▪ Overview of the .NET ecosystem ▪ Classes and objects ▪ Properties, methods, and fields ▪ Constructors and destructors ▪ Inheritance, Abstraction, Polymorphism, and Encapsulation ▪ Interfaces and abstract classes ▪ Comparison with TypeScript's OOP features ▪ Generics and Collections ▪ Asynchronous programming with async and await ▪ Nullable types and null-forgiveness operators
OOP	▪ *Introduction to OOP: Understanding the principles of Object-Oriented Programming in C#* ▪ *Classes and Objects: Defining blueprints and creating instances.* ▪ *Encapsulation: Controlling access with private, public, and protected members.* ▪ *Inheritance: Reusing code and extending functionality.* ▪ *Polymorphism: Overriding methods and dynamic method binding.* ▪ *Abstraction: Working with abstract classes and interfaces.* ▪ *Interfaces and Their Importance: Defining contracts for classes and achieving loose coupling.* ▪ *Constructors and Destructors: Managing object lifecycle and resource allocation.*

	▪ *Static Functions and Classes: Understanding the use of static members and their role in shared data.* ▪ *OOP Best Practices: Writing clean, maintainable, and scalable C# code.* ▪ *Using Linq*
SQL	▪ Introduction to database normalization and design principles. ▪ Primary keys, foreign keys, and unique constraints. ▪ Recap of SELECT, INSERT, UPDATE, DELETE. ▪ Basic filtering with WHERE, ordering with ORDER BY, and limiting results with LIMIT/TOP ▪ Basic joins (INNER, LEFT, RIGHT, FULL). ▪ Aggregation functions: COUNT, SUM, AVG, MIN, MAX. ▪ Understanding how indexes work (B-trees, clustered vs. non-clustered indexes). ▪ Best practices for designing relational databases.

Entity Framework 8	▪ Working with basic ADO
	  ○ Understand Connection, Command and Readers
	▪ Introduction to Entity Framework Core 8 and its architecture.
	▪ Setting up EF Core in an.NET 8 project.
	  ○ Code first
	  ○ Db first (basic examples)
	▪ Defining entities and relationships in EF Core.
	▪ Querying the database using LINQ and async operations.
	▪ Tracking vs. No-Tracking queries.
	▪ INSERT: Adding new rows to tables with single and multiple row inserts.
	▪ UPDATE: Modifying existing data with conditional updates using WHERE.
	▪ DELETE: Removing rows based on conditions and the difference between DELETE
	▪ Lazyloading related data
	  ○ Lazyloading data
	  ○ Use Include

ASP.NET 8	▪ Introduction
	▪ Understand ASP.NET pipelines
	▪ Design patterns & DI
	▪ Create custom middleware
	▪ Restful principles
	▪ Minimal Api & Controllers
	▪ Serialize and Deserialize messages
	  ○ Using system.text.json instead of Newtonsoft
	▪ Working with Tasks
	  ○ Tasks vs. Thread

	○ Use async/await ▪ Request validation ▪ Handle file & streams ▪ Monitoring and logging ○ Logging levels and categories: LogLevel, ILogger, and LogContext. ○ Configuring logging providers (Console, Debug, EventSource, cloud provider, Nlog, etc.). ▪ Error handling ▪ Configuration and IOptions, IOptionsMonitor, IOptionsFactory ▪ OpenAPI Specification & Swagger ▪ Use DTO and mappers ▪ ASP.NET Core Security ○ Authentication ○ Authorization ○ Using JWT
Working with git	▪ Git Basics: Overview of version control, Git installation, and setup, command-line essentials. ▪ Repository Management: Creating, cloning, and managing repositories, understanding the .git directory. ▪ Branching and Merging: Working with branches, merging strategies, resolving conflicts, using rebase. ▪ Collaborative Workflows: Understanding remote repositories, fetching, pulling, and pushing changes, managing pull requests. ▪ Git in Visual Studio: Integrating and using Git within the Visual Studio IDE, reviewing changes, committing, and syncing with repositories.
Developing microservices with C# & .NET 8	▪ Monolithic vs. Microservices ▪ Business Goals ▪ Key Benefits ○ Working with nuget ▪ Deriving Business Value ▪ Microservices Design Model ▪ Standardization and Coordination ▪ Focus on Communication ▪ API Gateway ▪ Async messages and queues
HTML & SCSS	▪ Introduction to HTML: Understanding the structure of web pages, elements, tags, and attributes. ▪ HTML5 Semantics: Using semantic elements like <header>, <footer>, <section>, and <article> to structure content meaningfully. ▪ Introduction to CSS: Styling web pages with basic CSS, including selectors, properties, and values.

	▪ CSS Layouts: Mastering layouts with Flexbox and CSS Grid for responsive design. ▪ Styling Text and Images: Working with fonts, colors, and media to enhance the look and feel of websites. ▪ Responsive Design: Creating responsive web pages using media queries and fluid layouts. ▪ Forms and Inputs: Building and styling forms, input fields, and buttons for user interaction. ▪ Animations and Transitions: Adding interactivity and visual effects with CSS animations and transitions. ▪ Working with sass ○ Sass concept ○ Scss variables ○ Nesting ○ mixins
Typescript	▪ Introduction & Differences from JavaScript: Understanding TypeScript's static typing and advantages over JavaScript. ▪ Type Annotations: Enforcing variable and function type safety. ▪ Interfaces: Defining structured contracts for objects. ▪ Classes and Inheritance: Using OOP concepts in TypeScript. ▪ Generics: Creating reusable components with type flexibility. ▪ Modules: Organizing code using imports and exports. ▪ Enums and Tuples: Working with specialized data types. ▪ TypeScript with JavaScript: Interoperability and type-checking in mixed projects. ▪ Session & Local storages
Advanced Angular development	▪ Angular bootstrap ▪ Components and Standalone ▪ DI & Injector Hierarchies ▪ Change detection. ▪ Interceptors ▪ Guards (CanActivate, CanLoad, etc.) ▪ Advanced Directive Development ▪ Advanced Component Development ▪ Advanced Reactive Patterns and Redux ▪ Angular Material ▪ Angular Reactive forms ▪ Implement out of zone code ▪ Working with rxjs ○ Observables ○ Subejct, BehaviorSubject, ReplaySubject ○ Map, Tap, Filter , etc ○ CombineLatest, Merge, MergeMap, etc. ○ Hot and Cold observables ▪ Routing & Nested routing & Lazy loading ▪ Using Signals ○ Use ngrx/signals