

# Implementing C# 11 and .NET 7.0

---

*Learn how to build cross-platform  
apps with .NET Core*

---

**Fiodar Sazanavets**



[www.bpbonline.com](http://www.bpbonline.com)

Copyright © 2023 BPB Online

*All rights reserved.* No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor BPB Online or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

BPB Online has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, BPB Online cannot guarantee the accuracy of this information.

First published: 2023

Published by BPB Online

WeWork

119 Marylebone Road

London NW1 5PU

**UK | UAE | INDIA | SINGAPORE**

ISBN 978-93-55513-281

[www.bpbonline.com](http://www.bpbonline.com)

**Dedicated to**

*To my mother, Liliya Sazanavets, and to the  
memory of my father, Dzmirty Sazanavets, who sadly  
isn't with us anymore. To my wife, Olga Sazanavets*

## About the Author

**Fiodar Sazanavets** is a Microsoft MVP (Most Valuable Professional) and a senior software engineer with over a decade of professional experience. He primarily specializes in .NET and Microsoft stack. He is enthusiastic about creating well-crafted software that fully meets business needs. He enjoys teaching aspiring developers and sharing his knowledge with the developers' community.

Throughout his career, he has built software of various types and various levels of complexity in multiple industries. This includes a passenger information management system for a railway, distributed smart clusters of IoT devices, e-commerce systems, financial transaction processing systems, and more.



## About the Reviewers

- ❖ **Vache Chek** is currently a senior software engineer with a specialization in backend and cloud computing.

Science and technology have been a constant source of fascination for him since his teenage years, when he began programming as a hobby on his Commodore 64 at the age of 13. Over time, his passion for programming grew, and he eventually pursued it as a career. Despite being self-taught, he has found that the most effective way to enhance his skills is by sharing his knowledge with others.

- ❖ **Kratika Jain** is an Enthusiastic Senior Software Developer eager to contribute to team success through hard work, attention to detail, and excellent organizational skills in leading and managing multiple projects while ensuring code quality, security, design pattern, and test cases with continuous integration build processes. She has participated in Agile project management and developed backend applications using Asp.NET, MVC, .NET CORE, Entity Framework, SQL server, and knowledge of software patterns and practices.

## Acknowledgement

I want to thank all the people who have supported and mentored me throughout my career. This includes Dikaios Papadogkonas, Vache Chek, Ian Turner, Paul Eccleston, Frank Lawrence, and all the other people I have worked or collaborated with.

## Preface

Welcome to this book about C# 11 and .NET 7! If you are a software developer, you have probably heard of C# and .NET, and you may have used them to create desktop, web, or mobile applications. C# is a modern, object-oriented programming language developed by Microsoft, while .NET is a powerful and flexible software framework for building applications for Windows, Linux, macOS, and other platforms.

This book is intended for developers who want to learn the latest features and enhancements in C# 11 and .NET 7. Whether you are a beginner or an experienced programmer, this book will provide you with the knowledge and skills you need to take advantage of the latest developments in C# and .NET.

In this book, you will learn about the new language features in C# 11, such as raw literal strings, improved date handling, and using generic maths. You will also discover the new APIs and improvements in .NET 7, such as a more intuitive command line interface, new functionality in the core libraries, and new project templates.

Moreover, this book will guide you through the development of practical applications using C# 11 and .NET 7. You will learn how to create web applications using ASP.NET Core, mobile applications using .NET MAUI, and compiled in-browser code by using Blazor. Although these technologies are not new, all of them have been enhanced quite significantly with the latest .NET update and this book will demonstrate these enhancements.

This book will also teach you how to use some more advanced .NET tools. You will learn how to build and run artificial intelligence models by using ML.NET. You will also learn how to build distributed applications by using the latest containerization capabilities of .NET.

I hope you will enjoy reading this book as much as I enjoyed writing it. Happy coding!

**Chapter 1: Getting Familiar with .NET 7 Application Structure** - introduces the reader to .NET 7 and provides a full set of instructions on how to get started, even if you have never used .NET before.

We will first set up our development environment. As you can build .NET apps on either Windows, Mac, or Linux, you will be shown what integrated development environments (IDEs) or a code editor you can install on the operating system of your choice.

We will then create a basic .NET application based on the Console Application template. Once the application has been created, we will examine the structure of a .NET project. Then we will write some code, which will enable us to get familiar with the basic C# syntax along with its inbuilt data types.

**Chapter 2: Overview of C# 11 Features** – demonstrates many exciting new features have been added to C# 11 to make the lives of developers easier and make the process of writing software more efficient. And this chapter will showcase all these features.

We will first cover struct auto-default, which allows struct-based objects to have their property values set to default values of their data types. This would prevent exceptions from being accidentally thrown. Next, we will cover generic attributes. This feature allows developers to use the generics feature of C# while defining attributes. This makes it easier to work with annotation in the code.

Afterwards, we will talk about sequence pattern matching. This feature gives developers more flexibility while comparing collections. Then we will move on to the new string-related features of the language. These include new raw string literals and multi-line interpolated strings. We will complete the chapter by looking at the required object members and static interface members.

**Chapter 3: What is New in .NET 7?** - focuses on the new features that have been added to the .NET platform itself, which consists of the SDK, build tools and the core libraries. We will start by going through the SDK and build tool improvements. The new features in these areas include the improvement to the command line interface, compiler optimization and so on.

Then we will cover various improvements to the core libraries, which come from Microsoft and System namespaces. The new features that have been added to these libraries include better observability improvements, new JSON features, improved RegEx, the ability to use TAR compression and several other improvements. Finally, we will go through the deprecated features of .NET 7 and breaking changes that have been introduced into the platform.

**Chapter 4: MAUI and Cross-platform Native Applications** - MAUI, which stands for Multi-platform App UI, is a framework that allows developers to build native applications that can be compiled to run on Windows, Mac OS, iOS, and Android. The intention behind this framework was that the same code base can be used to build an executable for any platform. And this includes both desktop computers and mobile devices.

In this chapter, you will learn how to use MAUI to build any type of a native application that the framework supports. You will learn how to set up your code base in such a way that you would then be able to compile your code into any type of executable. Some platforms supported by MAUI have some limitations in terms of what you can and can't do on them. And in this chapter, you will get to know those limitations.

**Chapter 5: Database Access with Entity Framework 7** - the server-side components of web applications are often required to access a database of some sort. Usually, this is done via an object-relational mapper (ORM), which abstracts away the database access and make it possible to manipulate data directly in the code. ASP.NET Core comes with its own ORM, which is known as Entity Framework Core. In this chapter, we will have a look at the latest version of this ORM - Entity Framework 7.

In this chapter, we will first examine the fundamentals of relational databases that Entity Framework 7 was designed to work with. Then you will learn how to use the ORM itself. There are a few ways you can set up the ORM inside your ASP.NET Core application. And in this chapter, we will have a look at them all: code-first and database-first.

**Chapter 6: Web Application Types on .NET** - introduces the reader to ASP.NET Core - the main framework on .NET that is designed for building web applications. We will also have a look at various types of web applications that ASP.NET Core supports.

First, we will start with ASP.NET Core fundamentals that are common to all ASP.NET Core application types. Following this, we will have a look at Web API, which is a type of an application that provides REST API for incoming HTTP requests but doesn't have any web pages. We will then move on to model-view-controller (MVC) applications, which allow the web pages to be rendered dynamically depending on the controller actions and the data from models. Finally, we will cover Razor Pages, which is a type of ASP.NET Core application where each web page has a server-side object associated with it.

**Chapter 7: Blazor and WebAssembly on .NET** - Blazor is a framework that allows developers to write .NET code that can be executed in browsers. This can be achieved in two ways - either by using Blazor WebAssembly or Blazor Server.

Blazor WebAssembly application is compiled into an executable that can run directly in browsers. It can also be hosted inside a standard ASP.NET Core application. Blazor Server, on the other hand, runs all its code on the server. In this case, the component in the browser will be communicating with the code on the server in real time via SignalR. Each of these hosting models has its pros and cons, even though the code would be very similar. This chapter provides an overview of all these hosting models.

**Chapter 8: SignalR and Two-way Communication** - introduces SignalR - an inbuilt ASP.NET Core library that allows the client and the server to engage in two-way real-time communication. The chapter shows how to build server-side components of SignalR, as well as demonstrating how to set up various types of its clients.

We will cover two types of SignalR clients - JavaScript and .NET. Both client types can be either used in-browser or as stand-alone applications. For example, JavaScript is a language that is native to in-browser applications. However, with technologies like Node.js, it can also be used in stand-alone applications. .NET is primarily used in stand-alone applications. But with a technology like Blazor, it can be executed in the browser as well.

**Chapter 9: gRPC on ASP.NET Core** - gRPC is a wrapper protocol that relies on HTTP/2 and enables efficient exchange of messages. This chapter demonstrates how to enable gRPC communication on ASP.NET Core. We will cover all the fundamental concepts of gRPC. You will learn Protobuf, which is the messaging protocol that gRPC uses. You will learn how to use Protobuf to set up both server and client-side gRPC components.

You will learn all four call types that gRPC supports, which are unary, server-streaming, client-streaming, and bi-directional streaming. Finally, you will get familiar with all the data types that Protobuf supports.

**Chapter 10: Machine Learning with ML.NET** - ML.NET is a library that allows developers to build machine learning application on .NET with relative ease. For example, the library makes it possible to select an ML algorithm and generate model for it in C# code. This model can then be re-used for multiple scenarios.

In this chapter, you will learn how to use ML.NET. First, we will go through its most fundamental features. Then, we will create some sample ML models by using some of its most popular algorithms. You will then learn the fundamentals of training and evaluating your ML model. And you will also be shown how to use a low-code model builder to build an ML model in a graphical user interface.

**Chapter 11: Microservices and Containerization on .NET 7** - Large-scale applications are often deployed as interconnected microservices that can be scaled out individually. And usually, the best way to deploy microservices is via containers. This will ensure that each service behaves consistently regardless of what environment it's deployed on. Because it runs in its own isolated environment, it won't be affected by any processes that happen on the operating system of the host machine, unless it has been explicitly exposed to such processes.

This chapter will show you how to apply orchestration to .NET 7 applications. It will walk you through .NET 7 Docker container images and the process of integrating Docker functionality with your .NET projects. Then you will learn the basics of container orchestration. For this purpose, we will have a look at Docker Swarm and Kubernetes.

## Code Bundle and Coloured Images

Please follow the link to download the *Code Bundle* and the *Coloured Images* of the book:

**<https://rebrand.ly/4xyu6op>**

The code bundle for the book is also hosted on GitHub at **<https://github.com/bpbpublications/Implementing-C-Sharp-11-and-.NET-7.0>**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **<https://github.com/bpbpublications>**. Check them out!

## Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

**[errata@bpbonline.com](mailto:errata@bpbonline.com)**

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at [www.bpbonline.com](http://www.bpbonline.com) and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at :

**[business@bpbonline.com](mailto:business@bpbonline.com)** for more details.

At **[www.bpbonline.com](http://www.bpbonline.com)**, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.



### **Piracy**

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

### **If you are interested in becoming an author**

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

### **Reviews**

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

## **Join our book's Discord space**

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



# Table of Contents

<b>1. Getting Familiar with .NET 7 Application Structure.....</b>	<b>1</b>
Introduction.....	1
Structure.....	2
Objectives.....	2
Setting up your development environment .....	2
<i>A suitable development machines .....</i>	<i>2</i>
<i>.NET 7 SDK .....</i>	<i>3</i>
<i>Setting up a code editor or an IDE .....</i>	<i>4</i>
<i>Installing a suitable code editor.....</i>	<i>4</i>
<i>Installing a suitable IDE.....</i>	<i>6</i>
<i>Microsoft Visual Studio 2022.....</i>	<i>7</i>
<i>Microsoft Visual Studio 2022 for Mac.....</i>	<i>8</i>
<i>JetBrains Rider.....</i>	<i>9</i>
Creating a .NET 7 application.....	10
<i>Creating an application via CLI .....</i>	<i>10</i>
<i>Creating an application via an IDE GUI.....</i>	<i>11</i>
.NET 7 project structure overview .....	14
<i>Adding a struct object.....</i>	<i>14</i>
<i>Adding an interface and a class.....</i>	<i>15</i>
<i>Modifying the entry point of the app.....</i>	<i>18</i>
C# 11 basics and inbuilt data types.....	21
<i>Inbuilt data types.....</i>	<i>21</i>
<i>Control flow .....</i>	<i>23</i>
<i>C# custom types .....</i>	<i>24</i>
<i>Access modifiers.....</i>	<i>25</i>
Conclusion.....	25
Points to remember .....	26
Multiple choice questions.....	26
<i>Answers .....</i>	<i>27</i>

---

Key terms .....	27
<b>2. Overview of C# 11 Features.....</b>	<b>29</b>
Introduction.....	29
Structure.....	29
Objectives.....	30
Prerequisites .....	30
Struct auto-defaults .....	30
Generic attributes.....	32
<i>Generic attribute example</i> .....	33
Sequence pattern matching .....	35
<i>Sequence pattern matching demonstrated</i> .....	36
<i>Sequence pattern matching with char span</i> .....	37
New string operations.....	39
Conclusion.....	44
Points to remember .....	44
Multiple choice questions.....	45
<i>Answers</i> .....	45
Key terms.....	46
<b>3. What is New in .NET 7? .....</b>	<b>47</b>
Introduction.....	47
Structure.....	47
Objectives.....	48
Prerequisites .....	48
SDK and build tool improvements .....	48
<i>CLI tools improvements</i> .....	48
<i>NativeAOT and enabling library trimming</i> .....	50
<i>Central package manager</i> .....	51
System and Microsoft library updates.....	52
<i>Microseconds and nanoseconds support</i> .....	52
<i>New JSON features</i> .....	55
<i>MaxDepth property of JsonSerializerOptions class</i> .....	55

---

Default JsonSerializerOptions configuration.....	57
JSON-specific HTTP PATCH.....	57
JSON polymorphism.....	58
Testing JSON features.....	61
New stream features.....	62
Regex improvements.....	63
Cryptography improvements.....	65
New TAR API.....	67
Observability improvements.....	69
New ways to monitor activity.....	70
Monitoring stopped activities.....	70
Current activity changed event.....	71
Enumerating activity properties.....	71
UpDownCounter metric.....	73
Breaking changes of .NET 7.....	74
Microsoft.Extensions nullability.....	74
Obsolete and non-nullable endpoints.....	75
PatternContext constraint.....	75
Multi-level lookup is disabled on Windows.....	75
MSBuild serialization of custom types.....	75
Conclusion.....	75
Points to remember.....	76
Multiple choice questions.....	77
Answers.....	77
Key terms.....	78
<b>4. MAUI and Cross-platform Native Applications.....</b>	<b>79</b>
Introduction.....	79
Structure.....	80
Objectives.....	80
Prerequisites.....	80
Introducing MAUI.....	80

---

<i>Enabling MAUI development environment</i> .....	81
<i>Creating a basic MAUI applications</i> .....	82
<i>MAUI XAML references</i> .....	86
<i>Working with Blazor on .NET MAUI</i> .....	89
<i>MAUI architectural patterns</i> .....	91
<i>Model-View-ViewModel</i> .....	91
<i>Patterns supported by MAUI via third-party frameworks</i> .....	94
<i>ReactiveUI</i> .....	94
<i>Model-View-Update</i> .....	94
Using MAUI to build desktop applications.....	95
<i>Preparing desktop development environment</i> .....	95
<i>Running a desktop app in a debug mode</i> .....	95
<i>Publishing a desktop app</i> .....	96
Using MAUI to build mobile apps.....	97
<i>Preparing mobile development environment</i> .....	98
<i>Running a mobile app on an Emulator</i> .....	99
<i>Publishing a mobile app</i> .....	99
Limitations of developing for Mac OS and iOS .....	100
<i>Extra tools required for publishing apps for iOS</i> .....	100
<i>Slightly lighter requirements for Mac OS apps</i> .....	101
Conclusion.....	101
Points to remember .....	102
Multiple choice questions.....	102
<i>Answers</i> .....	103
Key terms .....	103
<b>5. Database Access with Entity Framework 7</b> .....	<b>105</b>
Introduction.....	105
Structure.....	106
Objectives.....	106
Prerequisites .....	106
Introducing fundamentals of relational databases .....	106

---

<i>Overview of relational database management systems and SQL</i> .....	107
<i>Tables, relationships, and normalization</i> .....	108
<i>Introducing primary keys</i> .....	109
<i>Normalization and foreign key relationships</i> .....	110
Introducing entity framework 7 .....	112
Code-first approach in EF7.....	113
<i>Adding Entity Framework code</i> .....	114
<i>Adding entity objects</i> .....	114
<i>Adding database context</i> .....	117
<i>Adding database creation script</i> .....	120
<i>Creating the database by running the application</i> .....	125
Database-first approach in EF7 .....	126
<i>Creating EF7 models from an existing database</i> .....	126
<i>Looking at auto-generated code</i> .....	127
The latest features of EF7.....	128
<i>Controlling database-first via T4 templates</i> .....	128
<i>Guarded key</i> .....	129
<i>Table-per-concrete-type (TPC) mapping</i> .....	130
<i>Interceptors</i> .....	134
Conclusion.....	136
Points to remember .....	136
Multiple choice questions.....	137
<i>Answers</i> .....	138
Key terms .....	138
<b>6. Web Application Types on .NET</b> .....	<b>139</b>
Introduction.....	139
Structure.....	140
Objectives.....	140
Prerequisites .....	140
ASP.NET Core basics.....	140
<i>Basic ASP.NET Core application structure</i> .....	141

---

Web API on ASP.NET Core.....	143
<i>Web API with controllers</i> .....	144
<i>Minimal API endpoints</i> .....	150
<i>Adding open API metadata</i> .....	151
<i>Improvements to minimal API parameters</i> .....	151
<i>Minimal API and typed results</i> .....	152
<i>Uploading files to minimal API</i> .....	154
<i>The new in request processing middleware</i> .....	155
MVC on ASP.NET core.....	157
Razor Pages on ASP.NET Core .....	174
Conclusion.....	176
Points to remember .....	176
Multiple choice questions.....	177
<i>Answers</i> .....	178
Key terms.....	178
<b>7. Blazor and WebAssembly on .NET.....</b>	<b>179</b>
Introduction.....	179
Structure.....	180
Objectives.....	180
Prerequisites .....	180
Introducing Blazor.....	181
<i>Razor component example</i> .....	181
<i>@page</i> .....	182
<i>@onclick</i> .....	183
<i>@code</i> .....	183
<i>Razor keywords in Razor components</i> .....	183
<i>@using</i> .....	183
<i>@implements</i> .....	183
<i>@inherits</i> .....	184
<i>@inject</i> .....	184
<i>@layout</i> .....	184

---

@namespace.....	184
@preservewhitespace .....	184
@attributes .....	184
@bind.....	185
@ref.....	185
@typeparam.....	185
Blazor WebAssembly overview.....	186
Using code-behind approach.....	187
JavaScript Interop.....	190
Passing parameters to Razor components .....	192
Ahead-of-time compilation.....	193
Empty Blazor WebAssembly template.....	194
Hosting Blazor WebAssembly in ASP.NET Core .....	195
Adding a hoster Blazor WebAssembly to an existing ASP.NET core application.....	195
Form validation in Blazor.....	196
NavigationManager and passing state between pages.....	199
Setting up Blazor Server .....	200
Custom elements in Blazor .....	201
Razor component lifecycle .....	202
Empty Blazor server template.....	204
Conclusion.....	204
Points to remember .....	205
Multiple choice questions.....	205
Answers .....	206
Key terms .....	206
<b>8. SignalR and Two-way Communication.....</b>	<b>207</b>
Introduction.....	207
Structure.....	208
Objectives.....	208
Prerequisites .....	208
SignalR overview.....	209



---

<i>WebSocket</i> .....	209
<i>Server-sent events</i> .....	210
<i>Long-polling</i> .....	210
Creating SignalR Hub on the server .....	210
<i>Strongly-typed Hub</i> .....	214
<i>Dependency injection in SignalR Hub</i> .....	215
<i>JSON versus MessagePack</i> .....	216
JavaScript client for SignalR.....	216
<i>Adding HTML markup for SignalR client</i> .....	217
<i>Applying SignalR functionality in JavaScript</i> .....	221
.NET client for SignalR.....	224
Conclusion.....	228
Points to remember .....	229
Multiple choice questions.....	229
<i>Answers</i> .....	230
Key terms.....	230
<b>9. gRPC on ASP.NET Core .....</b>	<b>231</b>
Introduction.....	231
Structure.....	232
Objectives.....	232
Prerequisites .....	232
<i>gRPC overview</i> .....	232
Protobuf as the main message serialization protocol.....	233
Setting up gRPC server.....	234
<i>ASP.NET Core gRPC project structure</i> .....	234
<i>gRPC call types and data types</i> .....	237
<i>gRPC JSON transcoding</i> .....	240
Setting up gRPC client.....	244
<i>Using gRPC client factory and dependency injection</i> .....	248
Overview of gRPC data types.....	250
<i>Protobuf enums</i> .....	252

---

<i>Enabling collections with a repeated keyword</i> .....	252
<i>Dictionary-like Protobuf functionality</i> .....	253
<i>Using the oneof keyword in Protobuf</i> .....	253
Well-known data types .....	253
Conclusion.....	255
Points to remember .....	256
Multiple choice questions.....	256
<i>Answers</i> .....	257
Key terms .....	257
<b>10. Machine Learning with ML.NET</b> .....	<b>259</b>
Introduction.....	259
Structure.....	260
Objectives.....	260
Prerequisites .....	260
ML.NET fundamentals .....	261
<i>Types of machine learning</i> .....	262
<i>Supervised learning</i> .....	262
<i>Unsupervised learning</i> .....	262
<i>Reinforcement learning</i> .....	262
<i>Getting started with ML.NET</i> .....	263
<i>Using ML.NET to create your first ML model</i> .....	265
Choosing a problem for ML.....	267
<i>Binary classification</i> .....	267
<i>Multiclass classification</i> .....	269
<i>Regression</i> .....	270
<i>Recommendations</i> .....	272
<i>Forecasting</i> .....	273
<i>Image classification</i> .....	275
<i>Clustering</i> .....	276
<i>Anomaly detection</i> .....	277
<i>Ranking</i> .....	277

---

Training and evaluating your model.....	277
<i>Binary classification metrics</i> .....	278
<i>Multiclass classification metrics</i> .....	278
<i>Log-Loss reduction</i> .....	279
<i>Image classification</i> .....	279
<i>Forecasting</i> .....	279
<i>Regression and recommendation</i> .....	279
<i>Clustering metrics</i> .....	280
<i>Anomaly detection metrics</i> .....	280
<i>Ranking metrics</i> .....	280
Using a low-code model builder .....	281
Conclusion.....	285
Points to remember .....	285
Multiple choice questions.....	286
<i>Answers</i> .....	286
Key terms.....	287
<b>11. Microservices and Containerization on .NET 7.....</b>	<b>289</b>
Introduction.....	289
Structure.....	290
Objectives.....	290
Prerequisites .....	290
Docker container fundamentals .....	291
<i>Containers and container images</i> .....	291
<i>Base images and layers</i> .....	292
<i>Network isolation and port mapping</i> .....	292
<i>File system isolation and bind mounts</i> .....	292
<i>Installing Docker on Linux</i> .....	293
<i>Installing Docker on Mac</i> .....	293
<i>Installing Docker on Windows</i> .....	293
Base Docker image for .NET 7 .....	294
<i>Creating an application with Docker support</i> .....	296

---

<i>Adding Docker support to an existing application</i> .....	298
<i>Dockerfile structure</i> .....	298
<i>Building and running a Docker container</i> .....	300
Orchestrating applications with Docker Swarm.....	301
<i>Basic orchestration with Docker compose</i> .....	302
<i>Starting Docker Swarm</i> .....	303
Orchestrating applications with Kubernetes .....	305
<i>Installing Kubernetes on Linux</i> .....	305
<i>Installing Kubernetes on Mac</i> .....	306
<i>Adding services to a Kubernetes cluster</i> .....	307
Conclusion.....	308
Points to remember .....	308
Multiple choice questions.....	308
<i>Answers</i> .....	309
Key terms.....	309
<b>Index</b> .....	<b>311-317</b>

# CHAPTER 1

# Getting Familiar with .NET 7

# Application Structure

## Introduction

From November 2022, .NET 7 is the latest version of a cross-platform software development framework called .NET, which is being developed and maintained by Microsoft. Although the framework supports a number of programming languages, the most popular .NET language is C#, and the new version of this language, C# 11, is an integral part of .NET 7.

The main benefit of using .NET 7 over some other software development platforms is that it can run on any of the major operating systems on PCs, which include Windows, MacOS, and Linux. In this version, that is, .NET 7, in particular, you will be able to build applications that run on both PCs and mobile devices. Later in this book, you will find out how.

This book will teach you how to use the latest features of both .NET 7 and C# 11. Whether you are an experienced .NET developer or you have only started using C# and .NET recently, this book will provide you with enough knowledge of these subjects so that you will be able to write your own .NET applications.

If you have used .NET and C# before, this book will give you a good introduction to the latest features of both the platform and the language. If you are a beginner to C#, you will be able to follow this book, but you should familiarize yourself with the basic C# syntax first. The primary focus of this book is to showcase the latest features

of C# 11 and .NET 7. However, we will still briefly recap all the fundamentals. Also, carefully selected links to the official language documentation will be provided at the end of this chapter. So, whether you are only starting to learn .NET or are already an experienced software engineer that specializes in .NET, you will find this book valuable.

## Structure

In this chapter, we will discuss the following topics:

- Setting up your development environment
- Creating a .NET 7 application
- .NET 7 project structure overview
- C# 11 basics and inbuilt data type

## Objectives

In this chapter, we will focus on setting up your development environment and creating a basic application by using .NET 7 templates. Then, once we have created our initial project, we will recap some basics of C#.

The following chapters will focus on the new and shiny features of .NET 7 and C# 11. But before we get there, we need to have our fundamental dependencies set up. So, let us begin.

## Setting up your development environment

To start working with .NET 7 and C# 11, you will need the following:

- A suitable machine is running either Windows, MacOS, or Linux operating system.
- .NET 7 SDK
- A suitable IDE or a code editor

If you do not have any of these prerequisites installed already, let us go through the steps you need to take to install them.

## A suitable development machines

Since .NET is a cross-platform software development framework, it will work on either Windows, Linux, or MacOS. Therefore, a machine running either of these

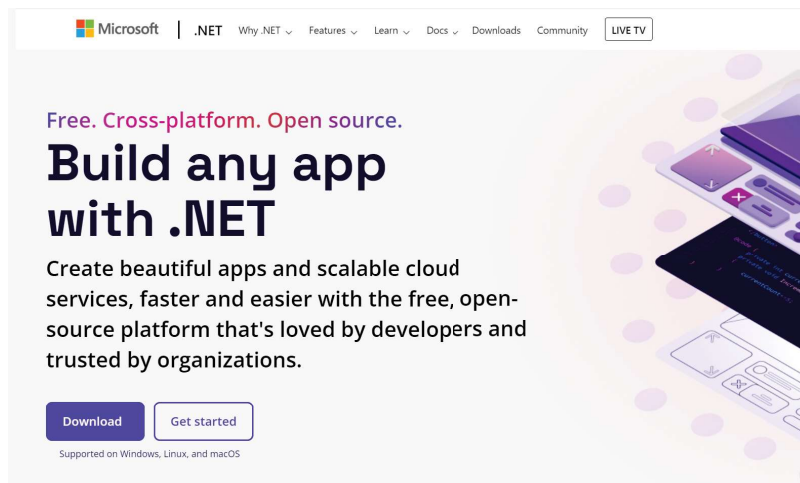
operating systems will be suitable. .NET is also compatible with a variety of CPU architectures. It will work with either Intel/AMD or ARM.

Regarding the processing power, disk space, and memory size, any average consumer-grade laptop or desktop would do. You do not need an extra-powerful machine to run your .NET code on. However, I would recommend a machine with at least 8 GB of RAM.

## .NET 7 SDK

.NET software development kit (SDK) contains everything that you need as a .NET developer, including the platform, the compiler, and all supporting tools. The latest .NET 7 SDK can be installed via the following steps:

1. To obtain .NET 7 SDK, you will need to visit the following page:  
<https://dotnet.microsoft.com/>
2. Once on the page, you will need to click on the **Download** tab, as per *figure 1.1*:



*Figure 1.1: Download tab on Microsoft .NET page*

3. You will then be taken to the download page, where you will need to choose the latest .NET 7 SDK to download and make sure that the option that you choose is SDK rather than Runtime. The .NET 7 runtime will allow you to run .NET 7 applications on your machine, but you will not be able to build them. SDK, on the other hand, contains both the runtime and all the development tools that you will need, including command line interface (CLI) tools that we will cover in detail in *Chapter 3: What is new in .NET 7*.

4. Next, we will set up a suitable IDE or a code editor if you have not done so already.

## Setting up a code editor or an IDE

It does not matter whether you will choose a code editor or an IDE for your application development. You will be able to use either. And you will be able to perform all exercises in this book regardless of this choice. However, it would be useful to know the difference between the two, so you can decide which tool would be more suitable for you.

A code editor allows you to write the code and navigate through it. It comes with a variety of code formatters and highlighters, so your code will be easy to read. But this is pretty much what the capabilities of a code editor are limited to. Typically, you will have to use some external tools or install additional plugins to be able to build your application from the source code. However, because code editors are limited in their capabilities, they tend to be substantially lightweight and faster to load than IDEs.

**Integrated Development Environment (IDE)** can do absolutely everything a code editor can do and much more. Things like creating new projects from various templates, running and debugging your applications, and building your source code into a deployable application are available out of the box. And all of these things can be managed via **Graphical User Interface (GUI)**. But all of this comes at the expense of performance. Typically, an IDE would be slow to install, occupy a reasonably large chunk of disk space, and noticeably slower to load than a code editor, especially if you are running it on a slower machine.

So, which one should you choose for application development with .NET 7? Well, you can choose either of these, depending on your preferences. If you do not mind building and testing your application via a CLI, which comes with .NET 7 SDK, a code editor would probably be sufficient. However, if you prefer the comfort of using a GUI for everything and you do not mind longer loading times and occasional dips in performance, then you should probably use an IDE. Also, I would recommend that you use an IDE if you are a beginner.

Let us now review which code editors and IDEs would work with .NET 7 and C# 11.

## Installing a suitable code editor

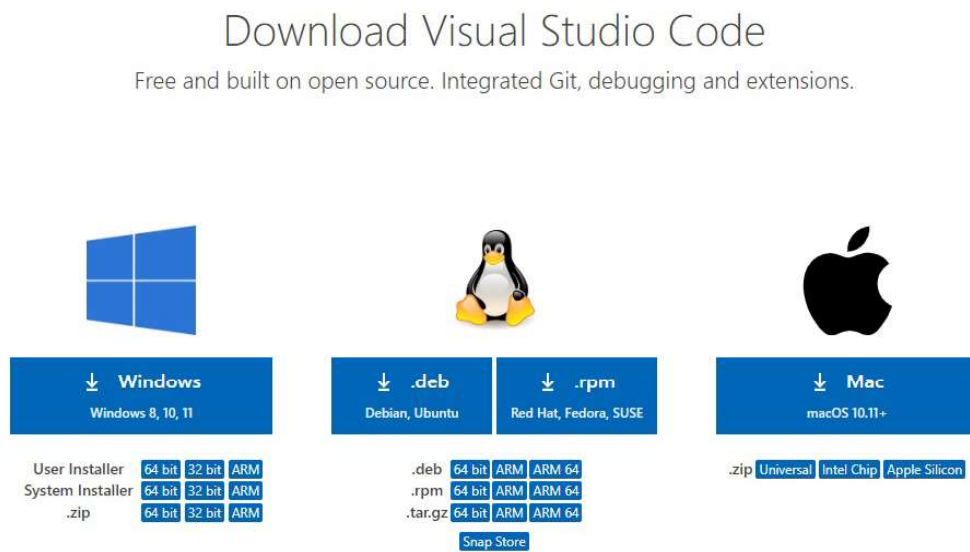
If you want to use a simple code editor, then the choice would be Visual Studio Code. And it does not matter which OS your development machine runs. There is a version of Visual Studio Code available for all supported operating systems.



Visual Studio Code can be downloaded from the following Web page:

<https://code.visualstudio.com/download>

You will then need to choose the download option that is relevant for your OS and your CPU architecture, as per *figure 1.2*:



*Figure 1.2: Visual Studio Code download page*

Once downloaded, you will then just need to follow the installation instructions that are specific to the OS you are using. Once installed, you will need to download a C# plugin for the code editor to make sure that all C# code is highlighted correctly. Either you can do it now, or you will be prompted to do so the first time you use the editor to open any file with the .cs extension. If you choose to do it now, you will need to open Visual Studio Code and click on the Extensions bar on the left-hand side, which is represented by a symbol containing four squares. Then, you will just

need to type **C#** in the search panel and install the first plugin that comes up in the results, as per *figure 1.3*:

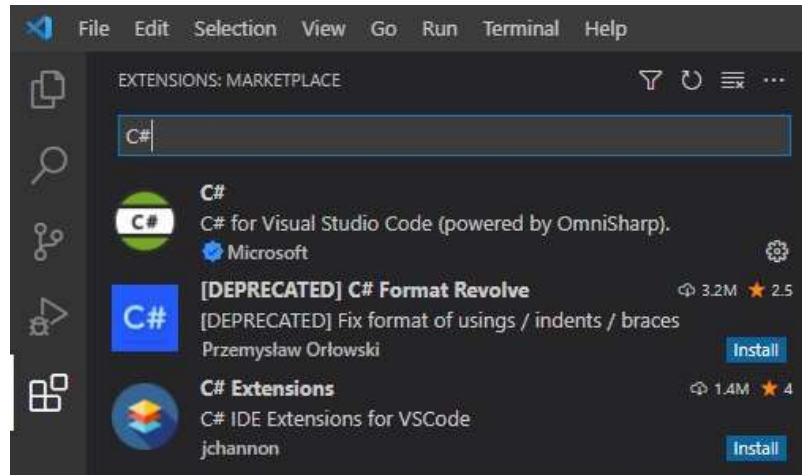


Figure 1.3: C# extension in Visual Studio Code

And this is all you need to start building your apps. However, if you prefer an IDE, then these are the steps you would need to take.

## Installing a suitable IDE

Installing an IDE is not as simple as installing a code editor. Visual Studio Code is the only recommended code editor for .NET development, regardless of which OS you are using. But when it comes to an IDE, different operating systems have different options available. The options can be summarized as follows:

### Windows

- Microsoft Visual Studio 2022
- JetBrains Rider

### MacOS

- Microsoft Visual Studio 2022 for Mac
- JetBrains Rider

### Linux

- JetBrains Rider

So, as you can see, the only common IDE is JetBrains Rider. But it has its own caveats too, so it might not be the best option for everyone. In fact, every IDE from this list has its own pros and cons. We will now examine each of the options, so you can decide which IDE to choose.

## Microsoft Visual Studio 2022

This is the official .NET IDE from Microsoft. Although it was mainly designed to work with .NET, it supports a range of different platforms, languages, and technologies. Despite its name, it is not related to Visual Studio Code in any way. It looks different and feels different. The only common things between the two is that both are made by Microsoft, and both can be used for writing code.

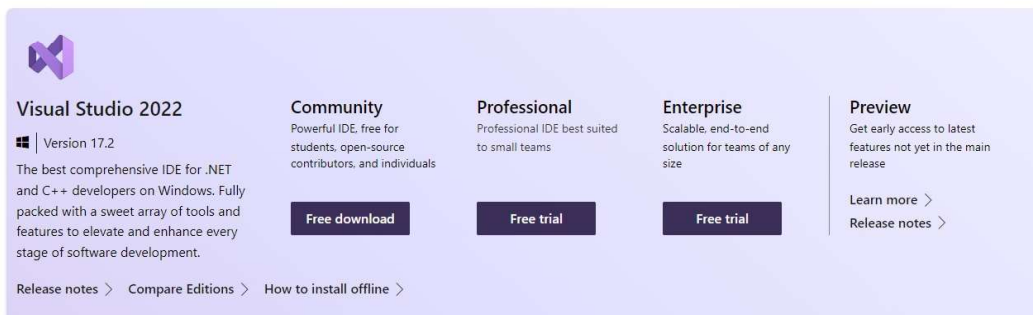
It comes with all the tools that you need. And you can also get it for free, as it has the so-called *community edition*. There are also premium *professional* and *enterprise* editions that you have to purchase a license for. They come with more tools than the free community edition. However, even the community edition comes with a sufficient amount of tools for developing your .NET applications. You will definitely not need anything more than the community edition to follow the exercises in this book.

The biggest advantage of using Visual Studio 2022 over any other IDE is that it is kept up to date with .NET updates. So, whenever .NET SDK gets updated (even if it is only a preview version of it), an update for Visual Studio will be made available immediately to make it compatible with it. So, you can be certain that your IDE will always be able to handle the latest .NET features.

To download Visual Studio 2022, you can visit its official page via the following link: <https://visualstudio.microsoft.com/downloads/>

You will be greeted by the following screen, which is illustrated in *figure 1.4*, where you can choose the version to download. Choose the **Community** option if you are not sure which version you will need. You can always upgrade later if you have to.

## Downloads



The screenshot shows the Visual Studio 2022 download page. On the left, there is a logo and the text: "Visual Studio 2022", "Version 17.2", and "The best comprehensive IDE for .NET and C++ developers on Windows. Fully packed with a sweet array of tools and features to elevate and enhance every stage of software development." Below this are links for "Release notes", "Compare Editions", and "How to install offline".

The main content area features four columns representing different editions:

- Community**: Powerful IDE, free for students, open-source contributors, and individuals. Button: "Free download".
- Professional**: Professional IDE best suited to small teams. Button: "Free trial".
- Enterprise**: Scalable, end-to-end solution for teams of any size. Button: "Free trial".
- Preview**: Get early access to latest features not yet in the main release. Buttons: "Learn more" and "Release notes".

Figure 1.4: Visual Studio 2022 download page

Once the download begins, you will just need to follow the installation instructions. But you will need to be aware that both the downloading and the installation may take a while, as Visual Studio 2022 is a fairly sizeable IDE.

Even though Visual Studio 2022 provides you with all the tools that you need, the main caveat is that it is only available for Windows. There is an IDE called Visual Studio for Mac, but despite its name, it is not a Mac version of the same IDE. It is a completely different piece of software. And this is what we will have a look at next.

## Microsoft Visual Studio 2022 for Mac

If you use Mac instead of Windows, Visual Studio 2022 for Mac might be a good IDE option. This IDE comes with sufficient tooling to build your .NET applications, but it is more basic than the Windows version of Visual Studio 2022. It is also that the GUI of the IDE looks completely different, so if you have previously been using Visual Studio on Windows and you have now switched to using Mac, it will take you some time to get used to it.

Another caveat of using Visual Studio for Mac is that its development lags somewhat behind the development of Visual Studio for Windows. And it does not keep up with the evolution of .NET. Sometimes you even have to wait months before you can start using any new .NET features. Sometimes the only way to use any new .NET features in this IDE is to install the preview version of it, which, as a piece of software that has not yet been signed off for an official release, may have some bugs.

You can download Visual Studio for Mac from its official page, which can be accessed via the following link:

<https://visualstudio.microsoft.com/vs/mac/>

Because there is only one version of this IDE, you will be presented with a single download button, as illustrated in *figure 1.5*:



*Figure 1.5: Visual Studio 2022 for Mac download page*

Then you just need to follow the download and the installation instructions, which should be self-explanatory. There is also a third IDE option. There is an IDE made by JetBrains called Rider. And it is worth examining regardless of the OS you are using.

## JetBrains Rider

The main advantage of Rider over any other IDEs is that it comes with a lot of inbuilt tooling by default. It will automatically find potentially problematic code, and it will provide refactoring suggestions. It will be able to decompile third-party libraries, so you will be able to see the original code they were written in. And the list goes on.

If you are a Windows user, then you will get a much richer IDE than Visual Studio at a relatively low price. It will be even more noticeable if you are a Mac user. And if you happen to be a Linux user, this will be your only option. The IDE will look the same and have the same functionality regardless of the OS you run it on.

However, it comes with its own caveats. But there are only two I can think of. There is no free version of it. After the initial 30-day trial, you must purchase the license. However, the price of it tends to be cheaper than either the *professional* or the *enterprise* edition of Visual Studio. The second caveat is that, since it is made by a third party rather than Microsoft, it sometimes lags slightly behind when new .NET SDK updates get released. However, the Rider development team tends to work fast, so these delays do not tend to be big. It tends to get updated quicker than Visual Studio for Mac.

Rider can be obtained from its official **Download** page, which can be found via the following link:

<https://www.jetbrains.com/rider/>

The Web page should automatically detect which OS you are on, so you will be presented with the download link that is specific to your OS:



Figure 1.6: Rider downloads page

Then all you have to do is just follow the instructions.

By now, you have chosen and installed either an IDE or a code editor that is right for you. Now, we are ready to start creating our first .NET 7 applications and examining their structure.

## Creating a .NET 7 application

When you write a .NET application, you work with projects and solutions. A project is a collection of code files that will later be built into a single executable file or a single reference library. These files form an application. An application can consist of a single executable file or have the main executable file alongside some other files that provide additional functionality. The latter types of files are known as libraries, and they can be shared between different applications. The libraries are also represented in the source code by projects.

A solution is something that holds multiple projects together. It is represented by a file with a `.sln` extension that gets placed alongside the project folders in the source code. Although you do not strictly require a solution, having one is helpful if you are using an IDE, as it would make it easier to manage and organize related projects.

Both projects and solutions can be created either via the CLI or via IDE GUI. And in this section, we will go through both of these methods. Later in the book, we will be primarily using the CLI commands, as they will be the same on all operating systems. Plus, .NET CLI comes with the .NET SDK, so if you have the SDK installed, you have the CLI too.

## Creating an application via CLI

The most basic type of a .NET application is known as a console application. It does not have any GUI. The only way it can interact with the outside world is via a textual interface, such as the one provided by CMD, PowerShell, Terminal, Shell, and so on. And because it is so basic, it is a perfect application type to use for our demo.

And now, we will go ahead and create our solution. To do so, you can open any command line terminal of your choice, create a folder in which you want to place your solution, navigate to this folder, and execute the following command:

```
dotnet new sln
```

This will create a file inside this folder with the same name as the name of the folder and the `.sln` extension. So, assuming that your folder is called **BasicApp**, the solution file will be called **BasicApp.sln**.

Now, we will create a project. For demonstration purposes, we will call it **BasicConsoleApp**. We will do so by executing the following command:

```
dotnet new console -o BasicConsoleApp
```