☑ Roadmap of Success

> **Software Engineer & DevOps Engineer** — A complete learning path from zero to professional.

🗺 Table of Contents

- Introduction & Duration
- About the Author
- Getting Started
- Basics of Software Engineering
- Front-End Roadmap
- Back-End Roadmap
- Microservices
- DevOps Roadmap
- Software Architecture
- Agile & Tools
- AI & Engineering Maturity
- Final Goal

📌 Introduction & Duration Overview

This roadmap is designed as a **long-term, serious learning path** for anyone who wants to become a **real Software Engineer & DevOps Engineer**, not just someone who knows tools.

⏳ Estimated Duration

There are **two main profiles** for learners:

Profile	Description	Minimum	Recommended
🟢 **Complete Beginner**	Starts from course 1, no programming background	3 years	3–4 years
🟢 **Knows Basics**	Understands loops, functions, OOP, data structures	1.5 years	1.5–2 years

📊 Total Learning Hours

Category	Hours
🎨 Front-End Development	~162h
💻 Back-End Development	~46h

Category	Hours
🔗 Microservices	~45h
⚙️ DevOps	~43h
🏛️ Software Architecture	~29h
📋 Agile & Tools	~1.5h
Total (Video Content)	**~327h**

> 💡 **Note:** This excludes the 27 Programming Advices courses (estimated 200-300h additional). Total learning time including practice, projects, and the foundational courses: **500-650 hours**.

> ⚠️ **This is not a race. Mastery matters more than speed.**

👤 About the Author

Hello and welcome! 👋 I'm **Ayoub Majjid**, a **fifth-year Computer Engineering student at EMSI**.

Current Roles:

- 🚀 **Tech Lead & Entrepreneur at Intellcap** — Leading 3 innovation projects
- 💼 **Capgemini TS PFE Intern** — Java & Microservices Engineer

This roadmap is built from:

- Academic foundations
- Real project experience
- Industry practices
- Mentorship mindset

🌐 **Website:** majjid.com

🎬 Introduction (Very Important)

Before starting, watch these foundational videos:

#	Video
1	Introduction Part 1
2	Introduction Part 2

🧠 Basics of Software Engineering

⏱️ **Estimated Duration:**

- Beginners (Course 1 → 13): **8–12 months**
- With basics (Course 13–27): **6–12 months**

What You Will Learn

Stage 1: Foundations & C++ (Courses 1–13)

Course	What You Will Learn
01 - Programming Foundations	Programming logic, computational thinking, how engineers solve problems
02 - Algorithms Level 1	Problem-solving basics, breaking down problems into steps

> **Before Course 03:** Learn VS Code setup (if you don't have a powerful PC for Visual Studio)
> 50 min — VS Code for C++ Development

03 - C++ Level 1	Variables, data types, operators, syntax fundamentals
04 - Algorithms L1 (Solutions)	Clean code practice, writing readable solutions
05 - Algorithms Level 2	Advanced problem-solving, pattern recognition
06 - C++ Level 2	Memory management, pointers, debugging techniques
07 - Algorithms Level 3	Complex logic, nested loops, multi-step solutions
08 - Algorithms Level 4	High-volume practice, speed & accuracy
09 - Foundations Level 2	Networks & OS basics, how computers communicate
10 - OOP Concepts	Object-oriented theory: classes, inheritance, polymorphism
11 - OOP Applications	Building real applications using OOP principles
12 - Data Structures L1	Arrays, linked lists, stacks, queues
13 - Algorithms Level 5	Capstone projects, applying everything learned

---

Stage 2: .NET & Enterprise Development (Courses 14–27)

Course	What You Will Learn
14 - C# Level 1	C# syntax, .NET fundamentals, transitioning from C++
15 - Database Level 1	SQL concepts, queries, relational database design
16 - OOP in C#	Professional OOP patterns in C#
17 - Database SQL Projects	Real database projects, complex queries, optimization
18 - C# & DB Connectivity	ADO.NET, connecting applications to databases
19 - DVLD Project	Full enterprise desktop application (Driving License System)
20 - C# Level 2	Advanced C# features, LINQ, async programming
21 - Database Level 2	T-SQL, stored procedures, triggers, programmability

Course	What You Will Learn
22 - Data Structures L2	C# collections, generics, performance optimization
23 - Algorithms Level 6	Advanced algorithms, competitive problem-solving
24 - Windows Services	Background tasks, scheduled jobs, system services
RESTful API	Building APIs, HTTP methods, REST principles
API Security	JWT authentication, roles, policies in ASP.NET Core
SOLID Principles	Maintainable design, dependency injection

After This Phase

You will be able to:

- Read and understand any code
- Solve beginner → intermediate problems alone
- Learn **any language or framework** with ease

Resources

- **Platform:** Programming Advices
- **27 courses** in total
- **Beginners:** Start with the first 13 courses

> **Parallel Learning:** Once you complete the **first 13 courses**, you can start the Front-End roadmap **in parallel** while continuing with courses 14–27!

Front-End Roadmap

Estimated Duration: 8–12 months (can run in parallel with basics from course 14)

> **Tip:** After completing courses 1–13, start Front-End while working on courses 14–27 simultaneously. This maximizes your learning efficiency!

Introduction

Topic	Link
What is Front-End?	Watch
How Websites Work	Watch
HTTP vs HTTPS	Watch

Tools

Tool	Duration	Link
VS Code	⏱ 50 min	Watch
Live Server	⏱ 50 sec	Watch

🌐 HTML & CSS

Topic	Duration	Link
HTML	⏱ ~4h	Watch
CSS	⏱ ~11.5h	Watch

🎨 Templates Practice

Template	Duration	Link
Template 1	⏱ ~1.5h	Watch
Template 2	⏱ ~3.5h	Watch
Template 3	⏱ ~6h	Watch
Template 4	⏱ ~6.5h	Watch

🌬 Tailwind CSS

Topic	Duration	Link
Tailwind CSS	⏱ ~1.5h	Watch

🔧 Git & GitHub

Topic	Videos	Link
Git Playlist	📺 19 videos 04:35:06	Watch

📌 **Best Practices:**

- Document everything using **Markdown (.md)**
- Use VS Code extensions: Office Viewer, Markdown to PDF
- Publish your projects
- Write code **by hand** (use AI only for understanding)

⚙ JavaScript

Topic	Videos	Link
JavaScript (Basics)	📺 99 videos 20:38:02	Watch

Topic	Videos	Link
JavaScript (Advanced)	38 videos 24:29:33	Watch
JavaScript Architecture	35 videos 4:02:27	Watch

TypeScript

Topic	Duration	Link
TypeScript	~2h	Watch

React

Topic	Videos	Link
React Course	131 videos 21:50:40	Watch
React Projects	36 videos 67:49:54	Watch

Next.js (Advanced)

Level	Videos	Link
Level 1	18 videos	Watch
Level 2	18 videos	Watch

Back-End Roadmap

Estimated Duration: 8–10 months

Java

> ⚠ Note: The link below is an intro video. For a full Java course, consider additional resources.

Watch Introduction

Spring Boot

Resource	Duration/Videos	Link
Course	42 videos 08:49:49	Watch
Project 1	~5.5h	Watch
Project 2	~23.5h	Watch

Practice heavily: Code With Zosh Playlists

Docker

Resource	Videos	Link
Full Course	28 videos 08:33:49	Watch

Microservices (Next Level)

Estimated Duration: 4–6 months

Resource	Duration/Videos	Link
Course	17 videos 12:06:56	Watch
Project 1	~6.5h	Watch
Project 2	14 videos 14:41:55	Watch
Project 3	~12h	Watch

DevOps Roadmap

Estimated Duration: 4–6 months

Level	Duration/Videos	Link
◇ Basics	~5h	Watch
◇ Deep Dive	59 videos 37:43:01	Watch

Software Architecture

Estimated Duration: 3–4 months

Resources

Resource	Link
Tech Mentors	View Courses

Udemy Courses

Course	Link
Software Architecture Essentials 9h	Udemy
Domain-Driven Design (Arabic) 5h	Udemy
Microservices Architecture 11h	Udemy

Course	Link
Docker & Kubernetes Essentials (Arabic) 4h	Udemy

Agile & Tools

Estimated Duration: 1–2 months

Topic	Duration	Link
Scrum Basics	~20 min	Watch
Jira Basics	~1.25h	Watch

Useful YouTube Channels

Channel	Link
Bashmohandes Mazen	YouTube
Programming Advices	YouTube

AI, Learning Curve & Engineering Maturity

How AI Fits Into This Roadmap

AI is a **powerful tool**, but **only after you build strong fundamentals**.

Phase 1: Learning Phase (Beginner → Intermediate)

During most of this roadmap:

❌ Don't	☑ Do
Rely on AI to write code for you	Use AI to explain concepts
Copy-paste AI solutions	Use AI to clarify errors
Skip the struggle	Use AI to understand*why* something works

You must:

- Write code **by hand**
- Struggle with bugs
- Build mental models

> This phase builds your **engineering brain**.

🚀 Phase 2: Post Learning Curve (Senior Mindset)

Once you complete this roadmap and build multiple projects:

☑ AI can now:

- Handle **50–60% of repetitive coding tasks**
- Speed up boilerplate
- Assist with refactoring and documentation

> This is **engineering leverage**, not dependency.

📺 **Reference (Very Important)** ⏱ ~30 min:
Antigravity – AI Explained

🏁 Final Goal

🎓 **Title Achieved:**

> **Software Engineer & DevOps Engineer**

🧠 **With:**

- Strong fundamentals
- System thinking
- Architecture mindset
- Smart use of AI (not blind reliance)

You don't compete *against* AI.
You compete with people who don't know how to use it correctly.

Created by Ayoub Majjid