Complete CS Roadmap

2025 Edition

Your comprehensive guide to becoming a software engineer

From Beginner to Industry Ready

Based on industry best practices and real-world experience

Last Updated: December 2025

Table of Contents

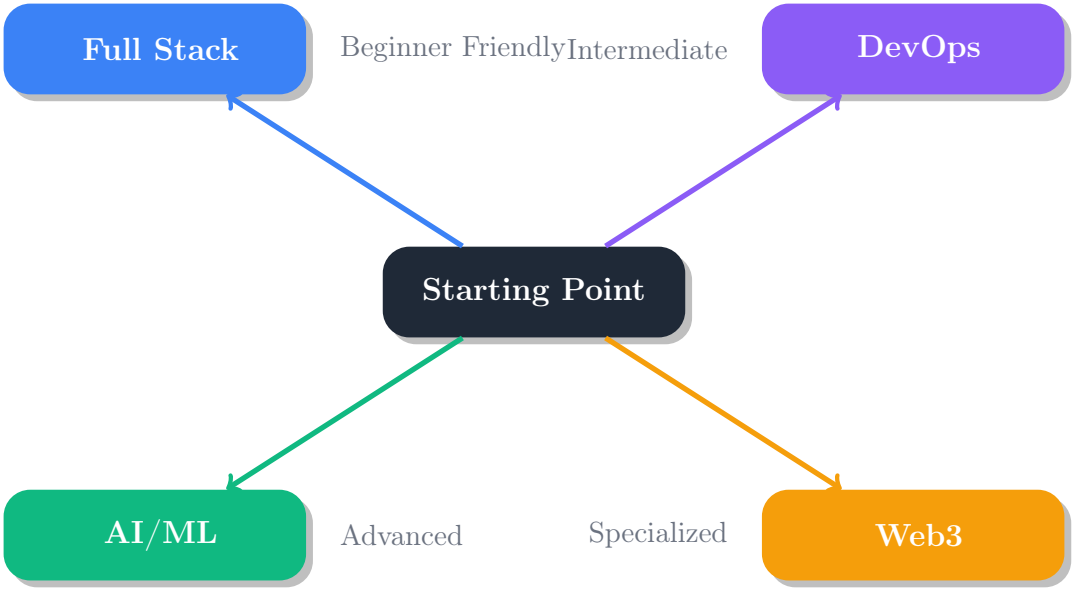
Contents

Introduction

Information

This roadmap covers **four major niches** in computer science where it's relatively easy to get an entry-level job and provide value to a company from day one. These are the areas that require engineers and human labor to execute quickly in companies.

Choosing Your Path



Important Note

For Complete Beginners: If you're just starting out, stick to **Full Stack Development**. It will probably take you 6-8 months to get decent at one of these niches. Once you have some programming experience and DSA knowledge, you can explore other areas.

Time Investment Overview

primaryBlue!20 **Track**	**Description**	**Duration**	**Difficulty**
Full Stack	Web development (frontend + backend)	2-4 months	
DevOps	Infrastructure, deployment, scaling	3-6 months	
AI/ML	Machine learning, LLMs, agents	4-8 months	
Web3	Blockchain, smart contracts, DeFi	4-8 months	

Full Stack Development

Full Stack Development

Pro Tip

The simplest track and the most commoditized. If you have decent CS knowledge, you can target the MERN stack in probably **2 months**. This is where everyone should start.

Learning Path

Syllabus

1. **HTML/CSS** — 1 week
   - Don't spend too much time here
   - One video or a week is more than enough
2. **JavaScript Basics** — 1-2 weeks
   - Syntax fundamentals
   - Variables, functions, objects, arrays
3. **JS Architecture** — 3-5 days
   - Event loop, call stack
   - Execution context
4. **Async JavaScript** — 1 week
   - Callbacks, Promises, async/await
   - Error handling
5. **Node vs Browser JS** — 2-3 days
   - Understanding runtime differences
   - Why server-side JavaScript exists
6. **HTTP and Express** — 2 weeks
   - HTTP protocol fundamentals
   - Building REST APIs with Express
   - Middleware, routing, error handling
7. **MongoDB** — 1 week
   - NoSQL basics
   - CRUD operations
   - Mongoose ODM
8. **PostgreSQL + ORMs** — 2 weeks
   - SQL fundamentals
   - Prisma or Drizzle ORM
   - Database design
9. **TypeScript** — 1-2 weeks

Type system, interfaces
Generics, utility types

10. **Turborepo** 2-3 days
    Monorepo management
    Build optimization
11. **Bun.js** 2-3 days
    Alternative to Node.js
    Performance benefits
12. **React** 2-3 weeks
    Components, props, state
    Hooks (useState, useEffect, useContext, etc.)
    State management
13. **Tailwind CSS** 3-5 days
    Utility-first CSS
    Responsive design
14. **Next.js** 2 weeks
    SSR, SSG, ISR
    API routes, file-based routing
    App router
15. **WebSockets + WebRTC** 1 week
    Real-time communication
    Socket.io implementation
16. **Queues/Pub-Sub** 1 week
    Redis, RabbitMQ, Kafka basics
    Message queue patterns

Projects to Build

Projects to Build

Beginner

1. Todo App

Basic CRUD operations

2. E-commerce App

User auth, cart, payments

Intermediate

3. Paytm-like Clone

Transactions, wallets

4. Trading App

Real-time data, WebSockets

Advanced

5. Lovable Clone

AI integration, LLM APIs

6. Codeforces Clone

Code execution, queues

Resources

Resources

YouTube Channels

- 100xDevs YouTube Channel
- Traversy Media
- Web Dev Simplified

Courses

- Angela Yu's Web Development Course (Udemy) - Great for beginners
- 100xDevs Cohort - Comprehensive paid option

Documentation

- react.dev - Official React documentation
- nextjs.org/docs - Next.js documentation
- prisma.io/docs - Prisma ORM

Open Source Practice

- GSOC Organizations (JavaScript ecosystem)
- Cal.com, Documenso, Formbricks (Next.js)

DevOps Engineering

DevOps Engineering

Important Note

Hiring Reality: DevOps is hard to get hired for as an early engineer because it's critical for companies. Most grow into DevOps by joining as a full-stack engineer first. Budget **$100**+ for cloud practice on AWS/GCP.

Learning Path

Syllabus

1. **Bash/Terminal** — 1 week
   - Basic bash commands
   - File/folder navigation
   - Git CLI proficiency
   - Optional: Vim/Neovim
2. **VMs and Bare Metal** — 3-5 days
   - Cloud environments (AWS/GCP)
   - Virtual machines vs bare metal
   - EC2/Compute Engine basics
3. **Process Management + Reverse Proxies** — 1 week
   - PM2, systemd
   - Nginx as reverse proxy
   - Multiple apps on single server
4. **SSL Certificates** — 2-3 days
   - HTTPS setup
   - Let's Encrypt/Certbot
   - Certificate renewal
5. **Auto Scaling Groups/MIGs** — 1 week
   - Horizontal scaling
   - Load balancers
   - Scale up/down policies
6. **Containers + Container Runtimes** — 3-5 days
   - Container concepts
   - OCI standards
   - Runtime options
7. **Docker** — 2 weeks
   - Basic commands

Writing Dockerfiles
Multi-stage builds
Docker Compose

8. **Kubernetes (Part 1)** — 2 weeks
   - Pods, Deployments, Services
   - ConfigMaps, Secrets
   - kubectl basics
9. **Kubernetes (Part 2)** — 2 weeks
   - Ingress controllers
   - Helm charts
   - StatefulSets
   - Advanced networking
10. **CI/CD** — 1 week
    - GitHub Actions
    - GitLab CI
    - Automated testing and deployment
11. **Monitoring/Observability** — 2 weeks
    - Prometheus + Grafana
    - Log aggregation (ELK/Loki)
    - Alerting systems
    - DataDog basics
12. **Infrastructure as Code (IaC)** — 2 weeks
    - Terraform
    - Pulumi
    - Multi-cloud deployments
13. **CDNs + Object Stores** — 3-5 days
    - S3, CloudFront
    - Static asset deployment
14. **Sandboxing/Firecracker** — 1-2 weeks
    - MicroVMs
    - Lambda/Cloudflare Workers internals
    - E2B sandboxes

Projects to Build

Projects to Build

E2B Clone
- Build sandbox environments on demand
- Use Firecracker for lightweight VMs
- API for starting/stopping sandboxes

Replit Clone
- Code execution environment
- Multi-language support
- Collaborative features

Cloudflare Workers Clone
- Edge computing platform
- Serverless function deployment
- Request routing

Pro Tip

DevOps is more about deploying applications than building them. Focus on: "I have a full-stack app, how do I deploy it using different methods (PM2, Docker, K8s, etc.)?"

Resources

Resources

Blogs
- e2b.dev/blog - Sandboxing and Firecracker
- modal.com/blog - Serverless computing

Documentation
- AWS/GCP official docs
- Kubernetes.io documentation
- Docker official docs

Practice Platforms
- KillerCoda (free K8s labs)
- AWS Free Tier
- GCP Free Credits

AI/ML Engineering

AI/ML Engineering

Information

This roadmap covers both **research-oriented topics** (understanding transformers) and **applied AI** (building agents, RAG systems). Focus based on your goals - most jobs are in applied AI.

Learning Path

Syllabus

Part A: Foundations and History

1. **History of AI** — 2-3 days
   - How we arrived at transformers
   - Pre-transformer approaches
2. **Deep Learning Basics** — 1 week
   - Backpropagation, sigmoid functions
   - Classical ML concepts
3. **Neural Networks + PyTorch** — 2 weeks
   - Building neural nets from scratch
   - PyTorch fundamentals
   - Tensors, autograd
4. **RNNs, LSTMs, Sequential Models** — 1 week *(Optional)*
5. **CNNs** — 1 week *(Optional)*

Part B: Transformers and Attention

6. **Coding Simple Attention** — 1-2 weeks
   - "Attention is All You Need" paper
   - Implementing attention from scratch
7. **Attention Optimizations** — 2 weeks
   - KV Cache, MQA, GQA
   - MLA (Multi-head Latent Attention)
   - DSA (DeepSeek's architecture)
8. **Hugging Face Ecosystem** — 1 week

Part C: Applied AI

9. **Instrumenting LLM Calls** — 3-5 days
10. **Vector DBs and RAG** — 2 weeks
11. **Context Engineering** — 2 weeks
12. **Agents from First Principles** — 2 weeks

13. **Agent Frameworks** 1-2 weeks
14. **Memory Systems** 1-2 weeks
15. **MCP (Model Context Protocol)** 1 week

Part D: Advanced Topics

16. **Computer Use and Multimodal Agents** 2 weeks
17. **Fine-tuning Basics** 1-2 weeks
18. **Fine-tuning for Specific Use Cases** 2 weeks
19. **RL Fine-tuning (RLHF)** 2-3 weeks
20. **Evals and Testing Agents** 1-2 weeks
21. **Other Modalities** Optional

    Voice, Images, Video generation

Projects to Build

Projects to Build

Agent Framework - Build your own LangGraph-like system
RL Fine-tuning Project + Evals - Pick a domain, create training environment
Devin Clone - AI coding assistant with sandbox execution
Memory Framework - Build Mem0-like system

Resources

Resources

YouTube: Andrej Karpathy, 3Blue1Brown
Courses: Coursera (Andrew Ng), Fast.ai
Blogs: cognition.ai, anthropic.com/engineering, blog.langchain.com

Web3 / Blockchain Development

Web3 Development

Important Note

Pick **either AI or Web3** if you're intermediate level - both require significant time. This roadmap focuses on **Solana** development.

Learning Path

Syllabus

Part A: Blockchain Fundamentals

1. **Intro to Blockchains** — 1 week
2. **Cryptography Basics** — 1 week
3. **Solana Architecture** — 2 weeks
4. **Solana Jargon** — 1 week
5. **PDAs (Program Derived Addresses)** — 1-2 weeks

Part B: Client-Side Development

6. **@solana/web3.js or Gill** — 1 week
7. **Solana Wallet Adapter** — 3-5 days
8. **Data Model on Solana** — 1-2 weeks

Part C: Smart Contract Development

9. **Token Program** — 2 weeks
10. **DeFi Concepts (AMM, DLMM, CLMM)** — 3-4 weeks
11. **Rust (Easy + Advanced)** — 4-5 weeks
12. **Anchor Framework** — 2-3 weeks
13. **Common Contracts (Staking/Escrow)** — 2 weeks

Part D: Advanced Topics

14. **Indexing** — 1-2 weeks
15. **MPC and Shamirs** — 1-2 weeks
16. **Partially Centralized Contracts** — 1-2 weeks

Projects and Resources

Projects to Build

DEX - AMM or order book based
CEX - Order matching engine
Wallet Application
Prediction Market - Polymarket-like

Resources

Solana Foundation Curriculum
Bitcoin Whitepaper
Jon Gjengset (YouTube) - Rust

Conclusion

Your Journey Starts Now

Pro Tip

For Beginners: Start with Full Stack. Build projects. Get a job. Then explore DevOps, AI, or Web3 based on your interests and company needs.

primaryBlue!20 **Phase**	**Duration**	**Goal**
Learning	2-4 months	Complete syllabus
Projects	1-2 months	Build 2-3 projects
Job Search	1-3 months	Apply and interview
primaryGreen!20 **Total**	**4-9 months**	**Entry-level job**

Important Note

Reality Check: 90% of people drop off before completing their first project. Consistency beats intensity. The ones who persist have a much higher chance of success.

Good Luck on Your Journey!

Remember: *Learning to code is a marathon, not a sprint.*
Take breaks, build projects, and never stop learning.