CAREER KIT

Software Engineer Career Kit: 90-Day Action Plan

How to Become a Software Engineer

2026-10 edition

Inside: at-a-glance summary · step-by-step path · costs & requirements · 90-day action plan · documents checklist · exams · interview questions · resume bullets · common mistakes · 13-week tracker · resources

howtobecome — independent career guides

SECTION 1

At a glance

To become a software engineer, you need to be able to write, test and ship working code, and prove it in technical interviews. The most common route is a bachelor's degree in computer science or a related field, but some people get hired after a bootcamp or by teaching themselves and building a strong portfolio. BLS reports a median salary of $135,980 for software developers (May 2025).

Timeline	1–4 years (bootcamp or self-taught at the short end, a degree at the long end)
Typical cost	$0–$120,000+ (self-taught to a four-year degree)
Education	Bachelor's degree in computer science or a related field is typical, not legally required
License	None required
Median pay	$135,980 per year (software developers)
Job outlook	+10% (2025–35), much faster than average

SECTION 2

Step-by-step path

1. Choose your route: degree, bootcamp or self-taught

Time: 1–4 weeks of research · Cost: $0

A computer science degree is the most widely accepted route, and BLS says software developers typically need a bachelor's degree. Bootcamps are shorter and cheaper than a degree but vary a lot in quality, and some employers still filter for degrees. Teaching yourself costs the least in money but takes the most discipline. Pick the route you can actually finish, and be honest about how much structure you need.

2. Learn one programming language well

Time: 3–6 months · Cost: $0–$500

Start with one general-purpose language such as Python, JavaScript or Java, and stick with it until you can build small programs without copying tutorials. Learn variables, functions, loops, data structures, error handling and how to read documentation. Depth in one language beats a shallow tour of five.

3. Learn the fundamentals employers test

Time: 3–12 months · Cost: $0–$500

Technical interviews focus on data structures (arrays, hash maps, trees, graphs), algorithms, and how fast your code runs. You also need Git, the command line, SQL basics, testing and how the web works (HTTP, APIs). Degree programs cover this in coursework; bootcamp and self-taught learners must study it on purpose.

4. Build real projects and put them on GitHub

Time: 3–6 months · Cost: $0–$200 for hosting

Build two or three projects that solve a real problem, such as a web app with a database, a tool for a local business or a contribution to an open-source project. Write a clear README for each, deploy it so people can try it, and write tests. Employers care far more about working, readable code than about certificates.

5. Get an internship, apprenticeship or first junior role

Time: 3–12 months of applying · Cost: Usually paid

Internships are the most common path into a first full-time job for degree students. Registered apprenticeships in software development also exist and pay you while you train; search Apprenticeship.gov. Other entry points include QA testing, IT support or a technical role at your current employer that lets you write code.

6. Prepare for technical interviews

Time: 1–3 months · Cost: $0–$300

Most companies use a coding screen, one or more live coding interviews and a behavioral interview; senior roles add system design. Practice solving problems out loud, explaining trade-offs and writing clean code under a time limit. Track the problems you miss and redo them a week later.

7. Keep learning and choose a specialty

Time: Ongoing

After your first job, grow into a specialty such as backend, frontend, mobile, cloud, data or security. Optional cloud certifications, such as AWS Certified Developer – Associate, can help for cloud-heavy roles. Code reviews, on-call work and reading other people's code will teach you more than any course.

SECTION 3

Costs & requirements

Estimated costs (USD)

Item	Low	High	Note
Bachelor's degree in computer science (total tuition)	$15,000	$120,000	Typical range, varies by school. In-state public schools and community college transfer credits are the cheapest route.
Coding bootcamp	$7,000	$20,000	Typical range, varies by provider. Some offer deferred tuition or income share agreements; read the contract carefully.
Self-taught courses and books	$0	$1,000	Many high-quality resources are free.
Computer	$500	$2,000	Any recent laptop that runs a code editor and a browser is enough to start.
Hosting and domain for portfolio projects	$0	$200	Many hosting platforms have free tiers.
Interview practice platforms	$0	$300	Typical range, varies by provider; free problem sets are enough for most people.
Optional cloud certification exam	$0	$150	AWS Certified Developer – Associate costs $150.
Total	**$22,500**	**$143,650**	

Requirements

Education	A bachelor's degree in computer science, software engineering, math or engineering is typical. It is not legally required, but many employers filter for it.
Programming skills	Strong command of at least one language, plus data structures, algorithms, Git, testing and SQL basics.
Portfolio	Public projects on GitHub or a portfolio site are especially important if you do not have a CS degree.
Interview performance	You must pass coding screens and live technical interviews, which test problem solving rather than memorized facts.
Licensing	No state license is required to work as a software engineer.
Work authorization	US employers will verify that you are authorized to work; some government and defense roles also require a security clearance.

SECTION 4

90-Day Action Plan

Tick each box as you finish the task. Do the phases in order; it is fine if a phase takes a little longer.

Phase 1: Foundations

Days 1–30

Goal: Pick a route and a language, and build a daily coding habit.

- ☐ Decide on your route (degree, bootcamp or self-taught) and write down your budget and weekly hours.
- ☐ Choose one language, Python or JavaScript, and code at least one hour every day.
- ☐ Install a code editor, set up Git and create a GitHub account.
- ☐ Finish an introductory course covering variables, functions, loops and basic data structures.
- ☐ Solve 20 easy practice problems and save your solutions to GitHub.
- ☐ Read the BLS software developer profile and three local job postings to see what employers ask for.

Phase 2: Build

Days 31–60

Goal: Ship your first real project and learn the fundamentals interviews test.

- ☐ Build and deploy one small full project, such as a web app with a database.
- ☐ Write a clear README with screenshots, setup steps and what you learned.
- ☐ Add automated tests to your project.
- ☐ Study arrays, hash maps, stacks, queues and basic sorting and searching.
- ☐ Learn SQL basics and write ten queries against a sample database.
- ☐ Join one developer community, meetup or open-source project and introduce yourself.
- ☐ Search Apprenticeship.gov for software development apprenticeships near you.

Phase 3: Apply

Days 61–90

Goal: Polish your portfolio and start interviewing.

- [ ] Build a second, more ambitious project that solves a real problem for someone.
- [ ] Write a one-page resume that leads with projects, skills and measurable results.
- [ ] Practice two interview problems a day, explaining your thinking out loud.
- [ ] Do at least two mock interviews with a friend or a free peer platform.
- [ ] Apply to 30 or more internships, apprenticeships and junior roles, and ask for referrals.
- [ ] Make one small open-source contribution, such as a documentation fix or bug fix.
- [ ] Follow up on every application within a week.

SECTION 5

Documents checklist

Gather these early. Keep originals in one folder and scans in the cloud.

- [ ] One-page resume focused on projects and skills
- [ ] GitHub profile with pinned projects and READMEs
- [ ] Portfolio website or links to deployed apps
- [ ] College transcripts or bootcamp certificate, if you have them
- [ ] Government photo ID and proof of work authorization for onboarding
- [ ] LinkedIn profile URL
- [ ] Two or three references who have seen your code or work
- [ ] List of target companies and application tracker

SECTION 6

Exams & how to prepare

Technical interviews (no licensing exam)

What it is: There is no state license or required exam for software engineers. Instead, employers use coding screens and live interviews on data structures, algorithms, code quality and, for senior roles, system design.

How to prepare: Practice problems in your main language, explain your thinking out loud and review the time and space cost of your solutions.

AWS Certified Developer – Associate (optional)

What it is: A 130-minute, 65-question exam on building and deploying applications on AWS. It costs $150 and is valid for three years.

How to prepare: Useful mainly for cloud-focused roles. Build a small app on AWS first, then use AWS's official exam guide and practice questions.

SECTION 7

Interview questions

1. Tell me about a project you built.

Tip: Explain the problem, your technical choices, one hard bug and what you would do differently.

2. Reverse a linked list, or find duplicates in an array.

Tip: Talk through a simple solution first, then improve it, and state the time complexity.

3. What happens when you type a URL into a browser?

Tip: Cover DNS, the HTTP request, the server response and how the page renders.

4. What is the difference between a list and a hash map, and when would you use each?

Tip: Focus on lookup speed and give a practical example.

5. How do you debug a problem you cannot reproduce?

Tip: Mention logs, narrowing inputs, adding tests and asking teammates.

6. How do you make sure your code works?

Tip: Talk about unit tests, edge cases and code review.

7. Tell me about a time you disagreed with a teammate.

Tip: Show you listened, used data or a quick prototype and reached a decision.

8. How do you learn a new technology quickly?

Tip: Describe reading docs, building a tiny example and asking good questions.

9. Explain what an API is to a non-technical person.

Tip: Use a simple analogy and keep it under a minute.

10. What is Git, and how do you resolve a merge conflict?

Tip: Describe pulling the latest changes, editing the conflicting lines and testing before committing.

11. How would you design a simple URL shortener?

Tip: Start with requirements, then the data model, API and how you would handle scale.

12. Why do you want to work here?

Tip: Mention something specific about their product or engineering team.

SECTION 8

Resume bullet examples

Adapt these to what you actually did. Use numbers wherever you can.

- Built and deployed a full-stack task app in React and Node.js with a PostgreSQL database and 80% test coverage.
- Wrote a Python script that automated a weekly report and saved the team three hours a week.
- Contributed a bug fix to an open-source library that was merged by the maintainers.
- Designed a REST API with authentication and documented it with OpenAPI.
- Solved more than 150 data structures and algorithms problems in Python.
- Set up continuous integration with GitHub Actions to run tests on every pull request.
- Built a scheduling tool for a local nonprofit used by 25 volunteers each week.
- Completed a 12-week bootcamp capstone as part of a four-person agile team.

SECTION 9

Common mistakes to avoid

1. Jumping between languages and tutorials without finishing any real project.
2. Paying for a bootcamp without checking audited placement data and financing terms.
3. Building only tutorial clones instead of projects that solve a real problem.
4. Skipping data structures and algorithms, then struggling in technical interviews.
5. Applying to a handful of jobs and stopping, instead of applying widely and asking for referrals.
6. Leaving GitHub projects without READMEs, tests or a live demo.
7. Ignoring communication skills; interviewers want to hear how you think.

SECTION 10

13-week progress tracker

Fill this in by hand each week.

Week	Focus	Done?	Notes
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			

SECTION 11

Resources, salary & disclaimer

Resources

- **BLS Occupational Outlook Handbook: Software Developers**

https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm

- **freeCodeCamp**

https://www.freecodecamp.org/

- **Apprenticeship.gov Job Finder**

https://www.apprenticeship.gov/apprenticeship-job-finder

- **GitHub Docs: Get Started**

https://docs.github.com/en/get-started

- **AWS Certified Developer – Associate**

https://aws.amazon.com/certification/certified-developer-associate/

Salary

Bottom 10%	Median	Top 10%
Less than $82,460	**$135,980 per year (software developers)**	More than $214,670

Outlook: Employment of software developers, quality assurance analysts and testers is projected to grow 10% from 2025 to 2035, much faster than the average for all occupations, with about 106,100 openings each year for the group.

Source: BLS Occupational Outlook Handbook, Software Developers, Quality Assurance Analysts, and Testers, May 2025 data (software developers) — https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm

Disclaimer. Requirements vary by state and change over time. Verify with the official sources listed. This guide is not legal or career advice.