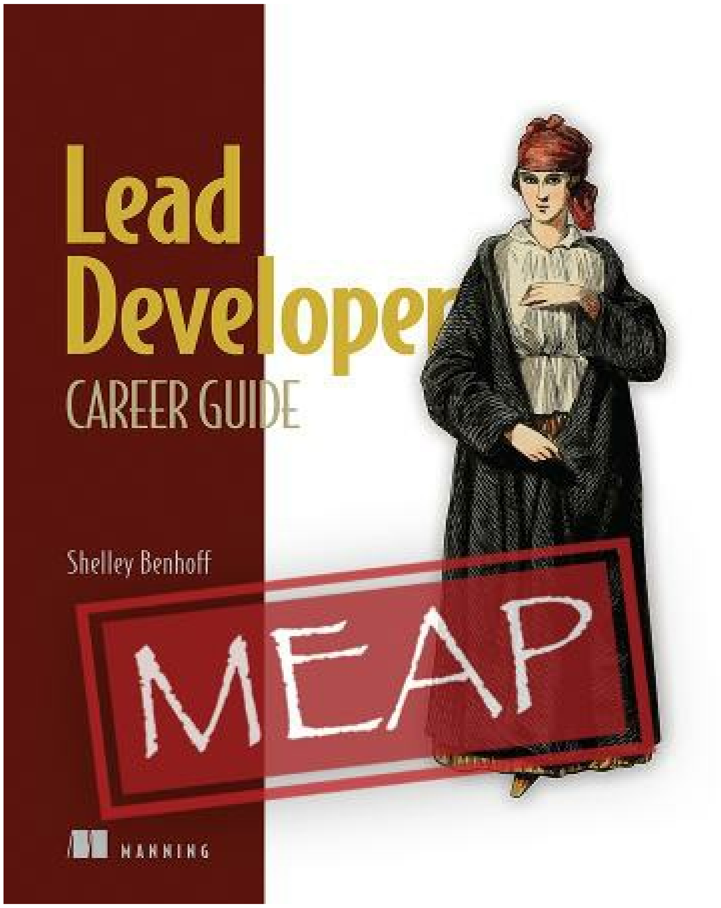
Lead Developer

CAREER GUIDE

Shelley Benhoff

MEAP

MANNING



Lead Developer

CAREER GUIDE

Shelley Benhoff

MEAP

MANNING

Lead Developer Career Guide MEAP V01

MEAP Edition

Manning Early Access Program

Lead Developer Career Guide

Version 1

Copyright 2023 Manning Publications

©Manning Publications Co. We welcome reader comments about anything in the manuscript - other than typos and other simple mistakes.

These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/lead-developer-career-guide/discussion

For more information on this and other Manning titles go to manning.com

welcome

Thank you for purchasing the MEAP for the *Lead Developer Career Guide*. As you embark on this journey, I assure you that this book is an excellent resource that will equip you with all the essential skills and knowledge you need to pursue a successful career as a Lead Developer.

It is true that there is a lack of references available specifically for Lead Developers, and most tech leadership references are made for Development Managers. While there are similarities in the roles, there are also distinct differences between the two positions. Development Managers are typically responsible for overseeing a team of developers and ensuring that projects are completed on time and within budget. On the other hand, Lead Developers are responsible for providing technical leadership and guidance to a team of developers, including setting technical standards and architecture, communicating with clients and stakeholders, and ensuring that projects are completed at a high level of technical quality. The Lead Developer Career Guide addresses this gap by providing targeted information and guidance specifically for Lead Developers, helping them develop the skills they need to excel in their role.

In this comprehensive guide, you will learn about the role of a Lead Developer and the tasks and expectations that come with it. The book delves into the Lead Developer career trajectory and provides guidance on how to transition from a Junior or Senior Developer to a Lead Developer or Lead Architect. You will also learn how to continuously improve your technical and soft skills. Learning leadership skills is a struggle for Lead Developers and this book covers everything you need to be a great leader including facilitating communication, being a mentor, and working with project teams.

This book includes stories and lessons I have learned in my career spanning over 20 years. I have been both a Lead Developer and Manager and I know the struggle of being new to leadership. I have also included case studies from tech influencers, authors, and technical professionals from prominent tech companies including Microsoft, Google, and Reddit.

Thank you for choosing The Lead Developer Career Guide. I am confident that this book will be a valuable asset to your career development and growth.

Please let me know your thoughts in the liveBook Discussion forum on what has been written so far and what you would like to see in the rest of the book. Your feedback will be invaluable in improving The Lead Developer Career Guide.

—Shelley Benhoff

In this book

1 What is a Lead Developer?

This chapter covers

- Defining what is a Lead Developer
- Comparing and contrasting a Senior vs. Lead Developer
- Understanding the tasks and responsibilities of a Lead Developer
- Listing the expectations of a Lead Developer beyond their day-to-day tasks

So, you want to have upward mobility in your career as a developer? The Lead Developer role may be for you. If you have done any research into the Lead Developer role or are a newly appointed Lead Developer, you probably encountered resources that focused on the technical expertise that you must achieve before being hired for this role. However, the title “Lead Developer” includes the word “lead”, and you must have a combination of both technical and leadership skills.

A Lead Developer, also known as a technical lead, is a senior-level software developer who takes on additional responsibilities within a development team. The Lead Developer is often responsible for guiding and mentoring other team members, making technical decisions, and coordinating the work of the development team. They represent the development team and work with project managers to prioritize development tasks. The Lead Developer may also play a key role in the planning and execution of software projects and may be involved in the management of the team and the development process. In my experience, the role of a Lead Developer varies in different types of organizations and there is no one-size-fits-all approach.

Lead Developers are expected to handle communication and support the development needs of project teams. To be a successful Lead Developer, you must have a high level of critical thinking and people skills. It is important to make quick and effective decisions and communicate those decisions among the team. There may be times when members of your team do not agree with your decisions, and you must be able to handle conflict and difficult conversations.

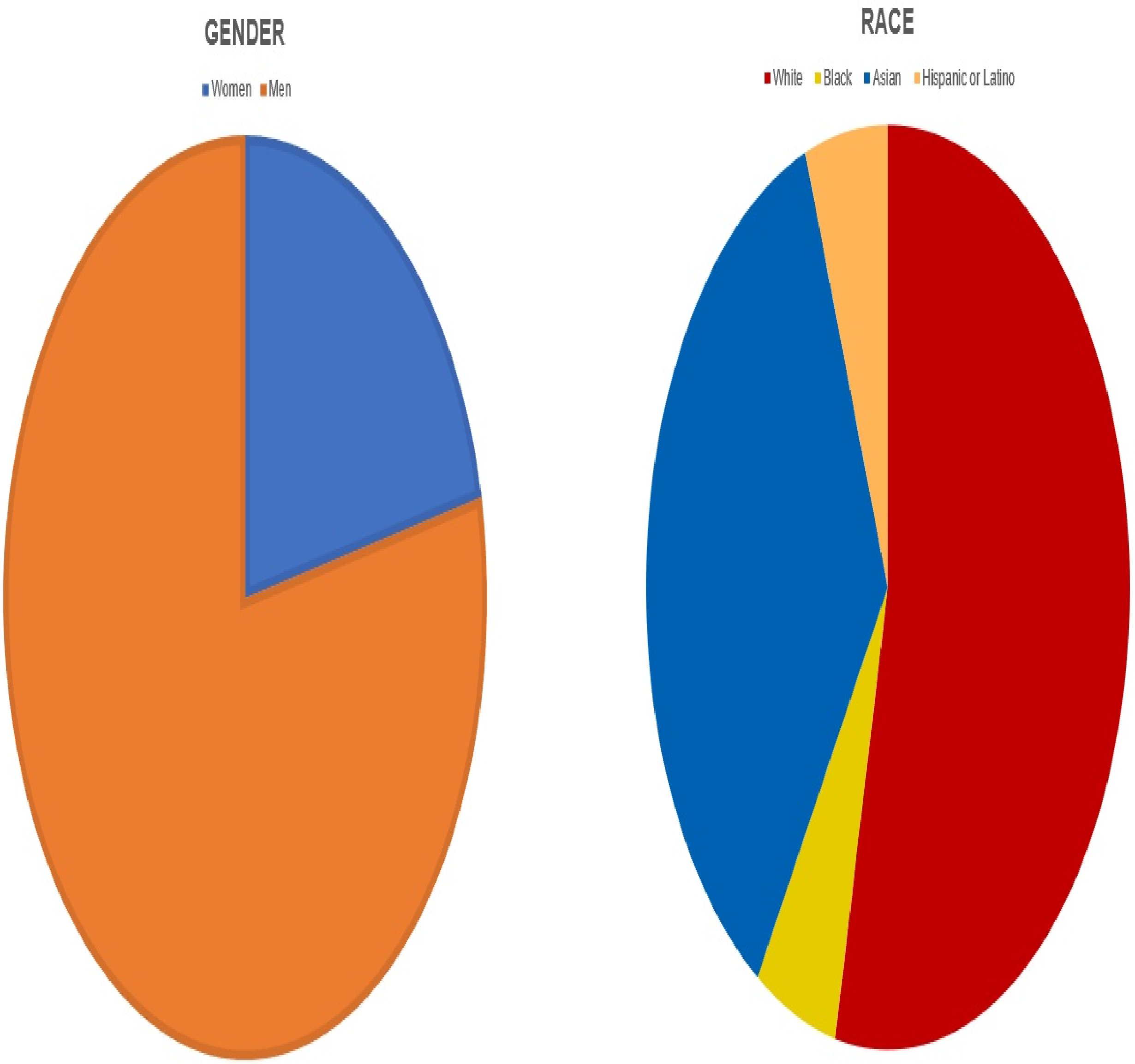
Many Senior Developers are promoted to Lead Developer positions based solely on their technical skills. The focus on soft skills is lacking for technical positions; however, when a developer is in a leadership position, they need support in mastering the soft skills needed for success in working on teams and interfacing with clients and stakeholders. I have witnessed this far too often, and it happened to me too. I felt like I was thrown into a pool without knowing how to swim and while it was difficult at first, I was able to learn on the job and I eventually learned how to be a great leader.

To start, you need to understand the demand for Lead Developers and who can be successful in this role.

1.1 Who Can Be a Lead Developer

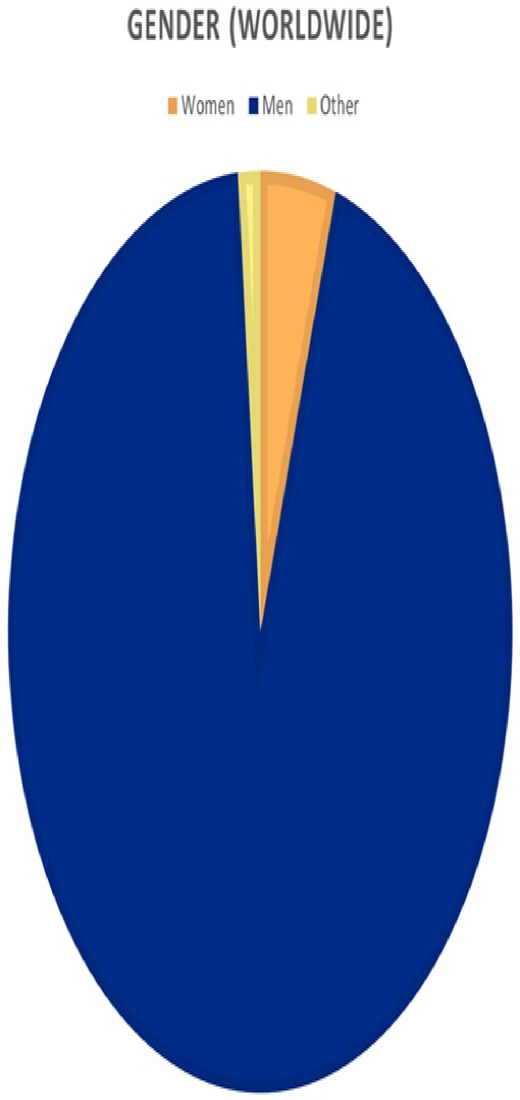
Lead Developers can be from anywhere in the world, have many different backgrounds, and represent all genders. Today, there is a need for more diversity in developer roles. According to The United States Bureau of Labor Statistics (BLS), only 21.5% of software developers in the United States are women. White developers equal 55.0% of the industry, Asians are 36.4%, followed by Black, Hispanic, or Latino at 5.7%. It is estimated that 8% of all developers are LGBTQIA+. Companies that champion diverse development teams can create innovation as they can create products and services for a wider audience. Being a champion for diversity is something you need to consider as a Lead Developer when you are hiring new developers.

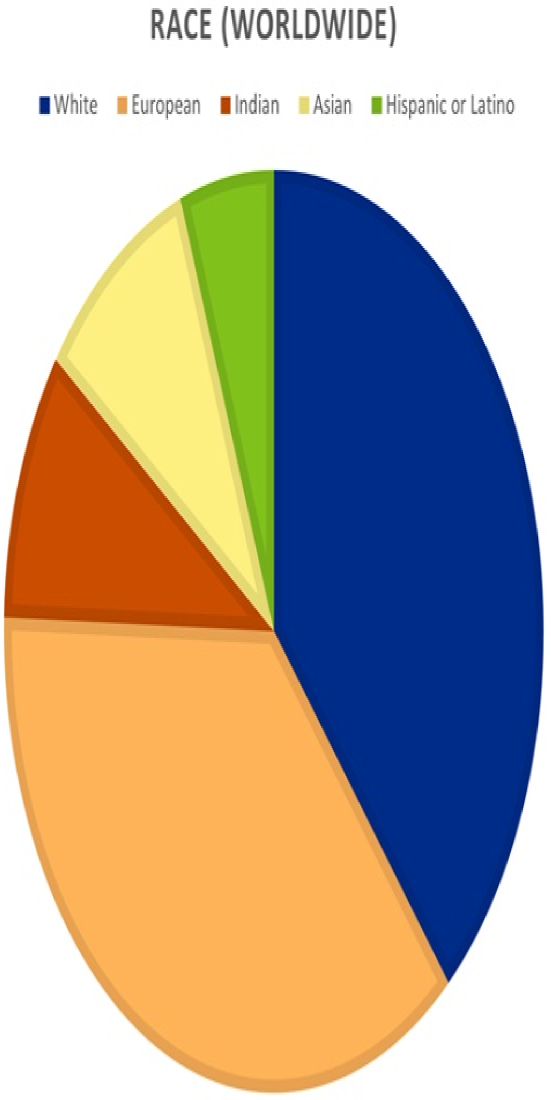
Figure 1.1 Developer Demographics - USA (https://www.bls.gov/cps/cpsaat11.htm)



The gender disparity is even greater according to the Stack Overflow Developer survey. This global survey suggests that almost 93% of developers are men, 4.8% are women, and 1.39% are non-binary, genderqueer, or gender non-conforming. In terms of race at the global level, 39.38% of all developers are white, 37.25% are European, 9.7% are Indian, 9.48% are Asian, and 5.71% are Hispanic or Latino.

Figure 1.2 Developer Demographics – Worldwide (https://survey.stackoverflow.co/2022/#demographics-gender-prof)





Most jobs I have had included a global team of both employees and contractors. In the United States, many tech companies outsource a portion of their software development staff, so I have had the pleasure of working with people from various backgrounds. However, most of the full-time employees I have worked with were white men. When I hired a team for the first time, I hired 3 white women and one white man. This was my own unconscious bias, and I should have done a better job of seeking out talent from different demographics. Do not make the same mistake that I did!

Another important demographic to note is education level. Many people think that a degree in computer science is necessary to be a developer and even more so for a Lead Developer. While this was true at one point, it is no longer the case. Patrick Collison, CEO of Stripe, and Sahil Lavingia, Founder of Gumroad do not have college degrees and they went on to excel in leadership and form successful tech companies. Not everyone can adhere to the strict approach to education offered by universities. Some people want to learn on the job and that’s fine! Many people also do not find coding until later in life and they studied something vastly different in college.

Many of the industry experts that are featured throughout this book come from different backgrounds including gender, race, and educational level. They are diverse, representing many groups in tech from around the world to show you that you can be successful as a Lead Developer no matter your background.

1.1.1 Who This Book Is For

This book is for aspiring & current Junior, Senior, and Lead Developers as well as Engineering or Development Managers. People in these roles can benefit from the information presented in this book to outline their career plans and have a clear path to achieving their goals using current industry best practices in technical leadership. If you are an Engineering or Development Manager, you can use the content presented in this book to help others achieve their goals and gain insight into how to help Lead Developers grow in their careers.

This book is not for non-developers and developers who do not work on a team. If you do not have a team to lead and mentor, then most of the concepts in this book will not apply to your role. Even if you interface with clients and stakeholders from time to time, the focus is more on technical skills with less focus on soft skills or leadership. For non-developers, this book is not a fit as it covers how to gain expert-level technical skills as well as leadership skills for development teams. Lead Developers must have a balance of technical and leadership skills and we will cover both throughout this book.

If you’re a Junior Developer, this book will help you learn how to effectively study and hone your technical skills. There are also many soft skills discussed in this book that you will benefit from in your current position such as communicating with your co-workers and using active listening to ensure that you are understanding not only what you are building, but also why you are building it. This goes a long way to help you create excellent software and websites with the end user in mind.

For Senior Developers, this book will help you prepare to move into a leadership role. You are already a technical expert, now you must learn how to discuss your technical approach for a project with project teams and get their buy-in. You also need to be an excellent mentor and support your team. These expectations may take you outside of your comfort zone and I encourage you to go outside your comfort zone. Some people don’t want to do this and that’s ok! However, I can tell you from experience that going outside of your comfort zone can lead you to opportunities that you never thought possible.

If you are an Engineering Manager with or without development skills, you will want to share this book with Junior, Senior, and Lead Developers on your team to help them grow in their careers. If your career includes development roles, then you may want to stay current with your skills so that you can communicate with your Lead Developers about their technical approaches. Even if your career did not include development roles and you have little to no development experience, as an Engineering Manager, you do have technical skills. You may not be able to sit down and code but you are able to understand technical concepts and this book will help you understand and empathize with your Lead Developers and their struggles.

1.1.2 Reviewing Top Industries for Lead Developers

As a Lead Developer, it is important to be aware of the industries that offer the most opportunities for employment. The most popular industries for Lead Developers include:

- Retail
- Technology
- Healthcare
- Banking/Fintech
- Manufacturing
- Business Services

- Government

I have worked in most of these sectors and historically, government jobs (at least in the USA) are the most stable and come with good benefits. The downside is that you will often not work on cutting-edge technology, instead, you will work with legacy code and systems. This can have its benefits as you will learn a bit of programming history. However, these jobs often go to developers who specialize in legacy systems. These systems are not going away any time soon and COBOL programmers are still in high demand.

I have also done a lot of work in the Healthcare and Business Services industries. In Healthcare, I learned a lot about security due to privacy laws dealing with patient information. And working in Business Services gave me the experience of working with external clients and customers. Each industry comes with its benefits, and I suggest you work for companies that provide you with opportunities to work on projects that interest you.

Each industry focuses on many different types of products and services. For example, in the Fintech industry, you would focus on consumer products and business solutions for handling payments or investments. Big Technology companies focus on innovative consumer products such as apps, streaming services, video games, and business-facing or B2B solutions. Manufacturing and Retail companies create the items that we use every day including cars and clothing which they sell on e-commerce websites.

Table 1.1 Top Industries and Companies for Lead Developers

Industry	Top Companies	Pros	Cons
Retail	Walmart, Amazon, Kroger, and Costco	Specialize in eCommerce, employee discounts	Prone to frequent layoffs based on sales, long hours during the holidays

Technology	Facebook/Meta, Apple, Amazon, Netflix, and Google	Learn from the best talent in the industry, excellent perks, high salaries, vesting options	Very high competition and expectations, stringent performance review process, long hours
Healthcare	CVS, UnitedHealth, Johnson and Johnson	Gain experience with high-level security systems and data privacy	Many codes to keep track of pertaining to health data
Fintech	American Express, PayPal, Mastercard, Fiserv, and Visa	Highly profitable, more job security, work on creative app solutions	Responsible for preserving sensitive data, constant hacking attempts
Manufacturing	Tesla, Samsung, Toyota, and Lockheed Martin	Variable project opportunities, learn different technologies and systems	Fewer remote jobs, not always on the cutting edge of technology

Whatever industry you are interested in, most companies need developers. You may work on different types of products and services such as building apps and websites or consulting for an agency; however, the role of the Lead Developer remains the same. The expectations and struggles for this role are applicable across all industries and sectors.

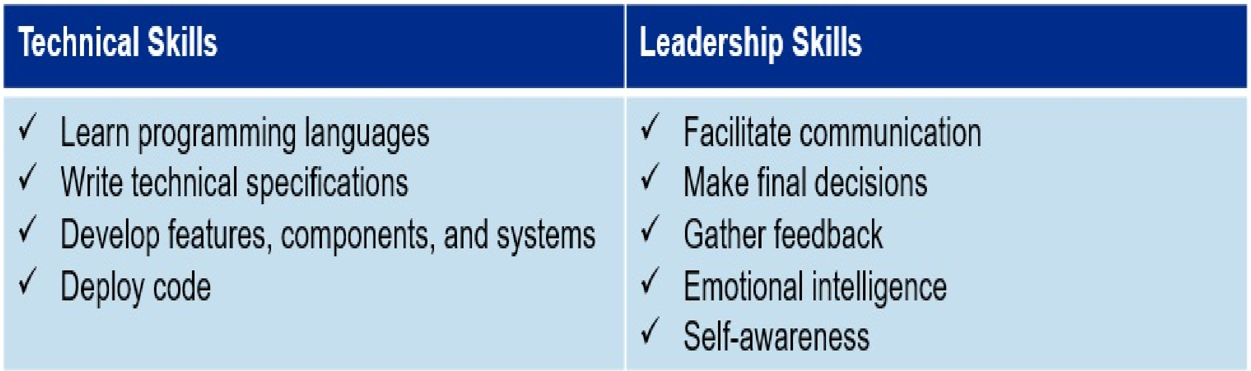
1.1.3 Being a Successful Lead Developer

A successful Lead Developer will learn technical skills throughout their career by working in roles such as Junior Developer, Senior Developer, or Developer Advocate/Developer Relations. In these positions, you will develop your technical skills as you build systems based on business or functional requirements and write technical requirements for tasks that are assigned to you. You will also understand the deployment process as you are responsible for deploying your code for testing before it goes into production.

Lead Developers are expected to have the technical expertise to build systems and lead the development team. You may work with a Lead Architect who is responsible for providing the technical requirements and approach. Lead Developers focus mainly on development tasks; however, these roles are often combined into one which is the focus of this book. It is best to know too much instead of too little to prepare you for anything.

When you move into a Lead Developer position, you will combine your technical skills with leadership skills. Facilitating communication is a key skill that you will need to master to ensure that the development team is productive and produces high-quality work. Decision-making is another leadership skill that takes some time to learn, especially making quick decisions in an emergency by gathering feedback from your team. You need empathy and self-awareness to build trust and cultivate relationships. When you build trust, your team will work together well, and you will see productivity increase. These relationships are important to not only your success but also your team's success.

Figure 1.3 Technical Skills vs. Leadership Skills



Technical Skills	Leadership Skills
✓ Learn programming languages ✓ Write technical specifications ✓ Develop features, components, and systems ✓ Deploy code	✓ Facilitate communication ✓ Make final decisions ✓ Gather feedback ✓ Emotional intelligence ✓ Self-awareness

The success of a Lead Developer is based on the success of your team. This includes ensuring that your team works within the projected timeline and that you take on minimal technical debt. An important metric to compare is the tasks to be completed in each iteration or release vs. what was completed. The quality of your team's work will be reflected in the open/close rates of tickets, issues, or bugs. When code is deployed to production, it should be the final version with no critical errors which will reflect an attention to detail by the team.

You do not have to have both technical expertise and leadership skills to apply for your first Lead Developer job. If you're lucky, you will be able to get promoted to a Lead Developer position at your current job. However, since you are reading this book, you are already ahead of the curve, and you will understand more about what lies ahead when you're finished. Many Senior Developers are thrown into Lead Developer positions and expected to "learn on the job". This only works if they are given the support that they need to navigate their new leadership position. Some companies offer access to learning platforms such as Pluralsight; however, you need mentoring as well. I suggest having a group of mentors from different backgrounds including gender, race, education level, and experience. When you have a diverse group of mentors, you will be inspired by their different points of view. That way, you can gain inspiration from them to find your own leadership style.

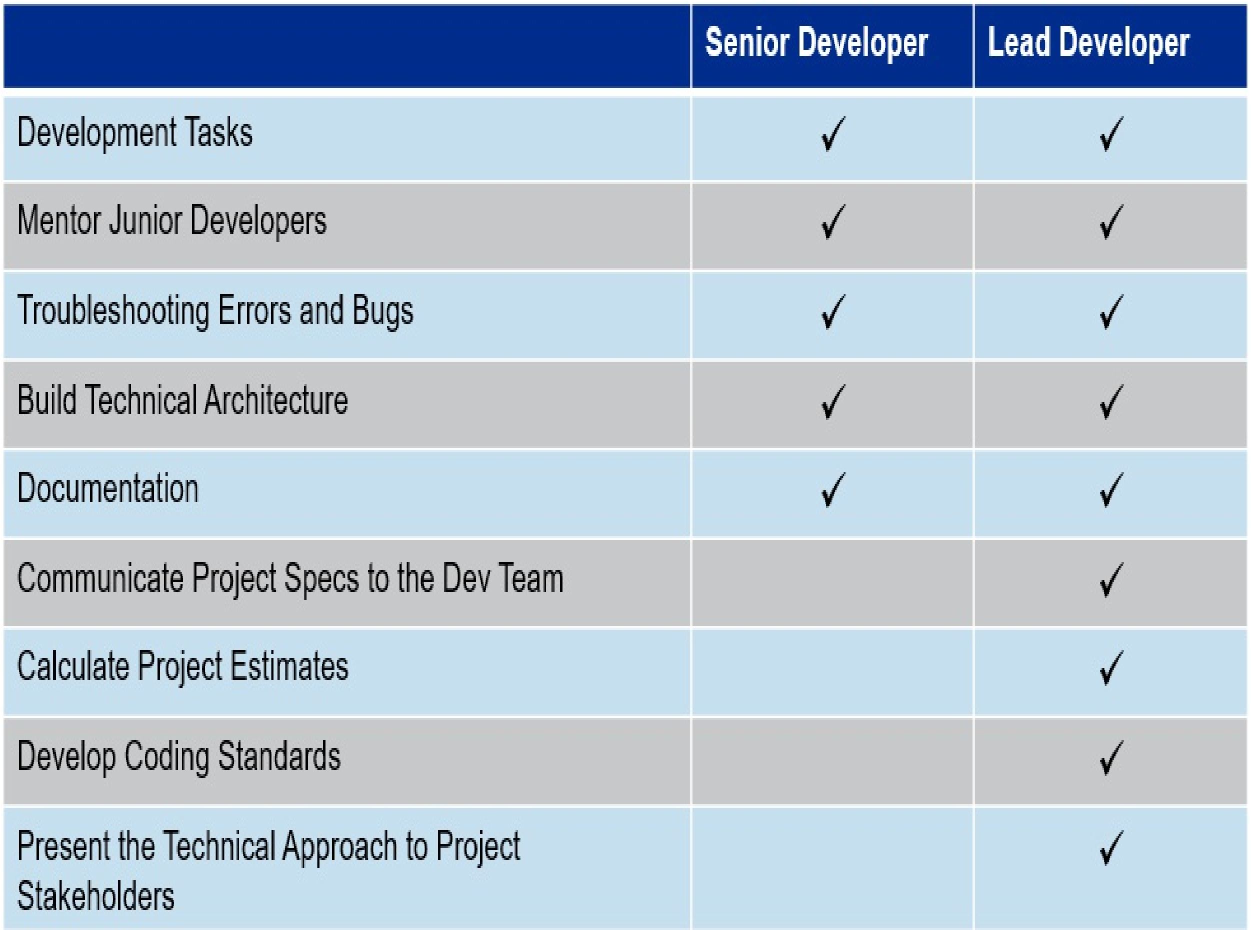
1.2 Lead Developer Tasks

Understanding the tasks and responsibilities of a Lead Developer is an important step in your career journey. Whether you are building products or offering services, you will have leadership responsibilities within project teams and potentially also at the organizational level. These can seem like daunting tasks but with the proper preparation and support, you will be able to take them head-on.

When you are a Senior Developer, your focus is mainly on your own work. Senior Developers may also mentor Junior Developers to assist them in learning new skills and collaborating with them on their work. The spotlight is on their technical skills and consistently improving their technical expertise. This role requires a high level of experience and knowledge that is incredibly valuable to development teams. Problem-solving skills help Senior Developers use their critical thinking to troubleshoot errors, fix bugs, and help remove blockers for other team members. This is a department-facing role with little to no direct communication with clients and stakeholders such as Project Managers, Development Managers, Marketers, etc.

In contrast, Lead Developers have the added responsibility of working with project managers, clients, and stakeholders. As a Lead Developer, you will represent the entire development team, gather their feedback, form your technical approach, and present it to the project team. The focus is more on business and communication skills as you are the face of the development team, and you act as the go-between to communicate requirements from the clients or stakeholders to the developers. This role requires some project management skills as well because the Lead Developer works closely with the Project Manager to calculate project estimates and keep the development team on track.

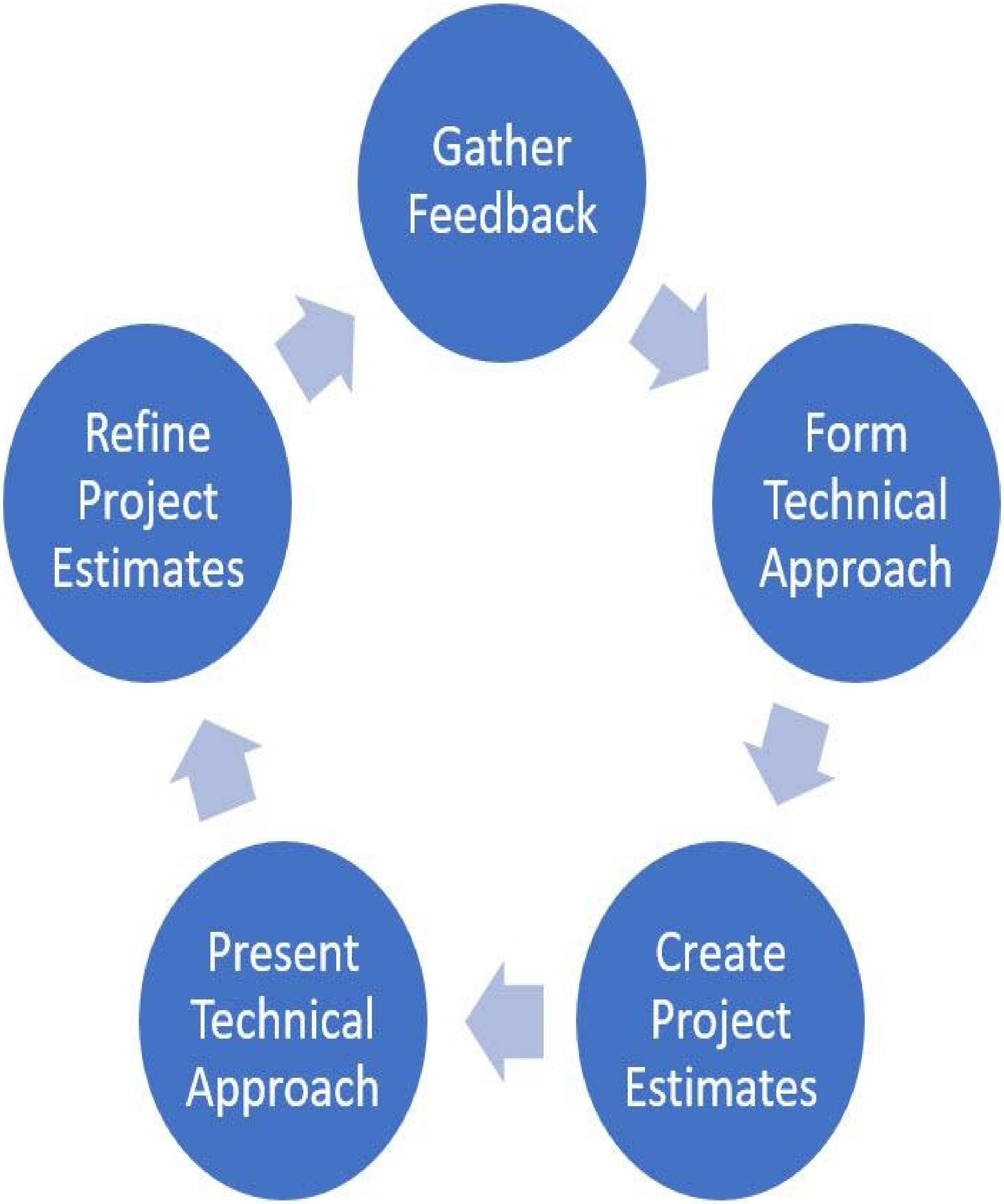
Figure 1.4 Senior vs. Lead Developer



	Senior Developer	Lead Developer
Development Tasks	✓	✓
Mentor Junior Developers	✓	✓
Troubleshooting Errors and Bugs	✓	✓
Build Technical Architecture	✓	✓
Documentation	✓	✓
Communicate Project Specs to the Dev Team		✓
Calculate Project Estimates		✓
Develop Coding Standards		✓
Present the Technical Approach to Project Stakeholders		✓

An important Lead Developer task is assisting in the project proposal phase when starting a new project. At the beginning of any project, the Lead Developer is tasked with forming a technical approach and creating project estimates. This process begins with gathering feedback from all project stakeholders including both business and technical requirements. This feedback supports the overall technical approach which will enable you to create the project estimates. After you present your technical approach, your estimates will need to be refined. Project estimates change frequently, and it is important for the Lead Developer to constantly gather feedback from the team.

Figure 1.5 Starting a New Project


Gather Feedback
Form Technical Approach
Create Project Estimates
Present Technical Approach
Refine Project Estimates

The following list contains the main responsibilities of a Lead Developer:

- Leading the Development Team
- Working with Project Teams
- Communicating with Clients and Stakeholders
- Setting Development Standards
- Building Technical Architecture

When you are working through these tasks, keep in mind that it will take practice. No one is comfortable with everything at first. You will learn as you go, and you will improve your skills. Communication is the key skill that every Lead Developer needs to be successful. When you are in a leadership position, your main job will be to facilitate communication within your team. Lead Developers play an important role to ensure that the development team is given the support that they need. This also goes a long way to retaining talent when your team is constantly learning, growing, and improving.

1.2.1 Leading the Development Team

Leading by example, building trust, and mentoring the development team is the first step on your path to leadership success. Lead Developers are responsible for supporting the development team to ensure their success both as a team and individually. This is ultimately how your success as a Lead Developer will be measured, not by your success alone but rather by your team.

When you’re in a leadership position, you can make or break an employee’s day. Leading by example means setting a positive tone for your team and coaching them through difficult times, especially in emergencies. For example, what do you do when a website goes down? Hopefully, you have an emergency plan in place that you can follow. If not, you should make one before emergencies happen. You must stay calm, formulate a plan of attack, and assign tasks to resolve the issue quickly. You should also tell everyone on the project team what the plan is step-by-step and keep them updated as each step is completed. If the resolution requires a deployment, you will need to get Quality Assurance (QA) involved and let them know the timing of each step so that they can prepare to start testing.

Building trust in your team will help you understand each other's points of view and improve your team's morale. If your team does not trust you, they will not come to you when they have a problem. If they can't come to you, they may not go to anyone, and their needs will never be met. When this happens, the employee can become disengaged from their work and potentially quit their job. Turnover is more expensive than you think, and it can take a developer up to a year to be fully acclimated to their job.

A study from LinkedIn (https://www.linkedin.com/business/learning/blog/learner-engagement/see-the-industries-with-the-highest-turnover-and-why-it-s-so-hi) states that the turnover rate in the tech industry is 13.2%. Furthermore, in recent years, 50% more people have voluntarily quit their jobs. According to a study by the Work Institute (https://info.workinstitute.com/hubfs/2020%20Retention%20Report/Work%2 the cost of employee turnover is approximately 33% of their annual salary. It is estimated that it can take up to two months to hire a new software developer and another six months to bring them up to speed. If there is no knowledge transfer, the costs are even higher which is why Lead Developers must ensure that they provide proper onboarding documentation which we will cover in depth later in this book. You want to avoid the loss of knowledge by retaining top talent and building trust in your team.

A good way to build trust is to back up your employees. When someone has an idea, and you think it is a good idea then you need to back them up. Defending the ideas of your team will make them feel listened to and appreciated, which will help you form a bond with them. If the idea is not exactly what you had in mind, you should keep an open mind and consider options that build upon the original idea. You can say, "That's a good start, but what do you think about [xyz]?" Asking questions is how you will draw out the talent within your team.

One of the most important behaviors you can exhibit is open communication. Clear, honest communication builds a foundation of trust and sets expectations for how your team interacts. If you make a promise, ensure you can deliver on it to show your team that you follow through on commitments. Being receptive to feedback shows that you value your team members' opinions and ideas. You must also act honestly and objectively and in the best interest of the community. Misusing information or treating people unfairly can erode trust and destroy morale. Respectfully handling information and requests and treating people with respect and fairness can help create a positive and productive team dynamic. By exhibiting these behaviors, you can build a culture of trust within your team, leading to greater collaboration, productivity, and success.

It is also important to ensure that your team is not stretched too thin and working too many hours because that results in burnout and lower productivity. You should promote work-life balance and encourage your team members to take breaks when needed. Additionally, it's essential to foster a culture of open communication and collaboration, allowing team members to share their workload and work together to achieve goals. Providing support and recognizing their hard work and achievements can also help boost morale and reduce stress levels. By taking a proactive approach to addressing burnout, you can help your team members feel valued, supported, and motivated, which can ultimately lead to higher productivity and better outcomes. It is of utmost importance to get feedback from your team on project estimates and coordinate timelines with the project manager.

1.2.2 Working with Project Teams

Lead Developers are a crucial part of the project team. They work closely with project managers, QA, clients, and stakeholders to provide their technical expertise and assist in project planning. Their skills directly contribute to the success of the project as they provide estimates and facilitate communication between the clients, stakeholders, and development team.

Working well with project managers is an essential skill for a Lead Developer. You should understand the project management framework and where you can assist. For new projects, you will provide estimates of the development work that needs to be done and the overall approach. As the work progresses, you will work closely with the project manager to organize tasks, priorities, bugs, and technical debt. The Lead Developer is the liaison between the project manager and the development team, and you must rely on their feedback to come up with estimates and to weigh in on priorities.

For example, if a developer on your team has been working on a product or feature and the priorities change, they should be given the time to get to a good stopping point, document where they left off, and then check it into a repository. The Lead Developer should ensure that their work is properly documented so that they or someone else can pick it back up in the future. This happens a lot because priorities change all the time. If that were not the case, we would not need project managers!

Another important task for a Lead Developer is working with QA to ensure that the quality of the development work matches the requirements, the code is ready for testing, and everything works as intended. This entails assisting with their testing strategy, ensuring that the technical requirements are well documented, and answering questions when necessary. The QA process is critical to a project's success and as a Lead Developer, you will be expected to work with them to figure out common issues so that you can adjust your development approach to avoid these issues in the future.

Working directly with clients and stakeholders is probably one of the hardest skills for a Lead Developer to master. I had a very hard time at first being the face of the development team which can be a daunting experience, especially if you are an introvert by nature. Lead Developers attend meetings including project kickoffs, sprint planning, and retrospectives to name a few. You may work with both internal and external stakeholders and vendors for services that you are using for the project. When you work directly with project stakeholders, they are getting the technical information from you and not the project manager. This goes a long way to avoid miscommunication and ensure that everyone is on the same page, which will help to minimize technical debt and improve productivity.

1.2.3 Communicating with Clients and Stakeholders

So, how do you communicate with clients and stakeholders? Everyone is different so you should start by assessing their level of comfort, understanding their point of view, and building relationships. Remember, clients and stakeholders are people with lives outside of work. You should always get to know the people you work with as human beings, consider their emotions, and attend to their needs.

You can assess a client’s level of comfort by observing how they interact with the rest of the project team. If the project is going poorly, are their arms crossed? Is there a frown on their face? Or are they open to suggestions and collaborating to resolve the issues? If they are not collaborative, this suggests that they do not trust that the project team has their best interests in mind. You can always tell the status of a working relationship when things go wrong. How do people react? Teams that work well in the face of adversity and show the client or stakeholder a unified front will ease their concerns and make them more comfortable with the status of the project.

Another key aspect of providing professional support for your clients and stakeholders is understanding their points of view. You should not design their systems until you have business requirements and you have spoken to the team to ensure that you are all in agreement. Thinking about not only how but why your team is building certain products or features will help the development team be more accurate when they begin working on new tasks. Developers need to test their work before it is deployed, and they need real-world examples to work with. You may also get feedback from the developers for improvements to run by the rest of the team. When you suggest improvements to clients and stakeholders and you understand their needs, you will provide them with added value, and they will think highly of the work that the team is doing.

Getting to know your clients as people helps to diffuse tension and build important relationships. Meetings should start with a quick “How is everyone doing?” and you should not shy away from asking people about their life outside of work. If you have a meeting on a Friday, you can ask, “What is everyone doing this weekend?”. Or ask the opposite on a Monday! You will often find things that you have in common with them and you can work them into the conversation from time to time. You may follow the same sports teams, watch the same movies, or read the same books. Even if you come from different backgrounds, you can often find something that you have in common with anyone.

Everyone struggles with communicating with clients and stakeholders, especially if you are providing a service to a client and not working on an internal project. Providing services such as consulting or working on business-to-business (B2B) solutions is a bit harder because the company is relying on you to ensure that the clients and stakeholders are happy so that their business is retained. I have worked for consulting agencies on projects including building enterprise websites for Fortune 500 companies and some of these projects had multi-million-dollar budgets. The level of responsibility was high for me as the Lead Developer on those types of projects and it was a struggle to find my voice at first.

When you are working on an internal project to build a product or support your organization's infrastructure, communication is a bit different. It is in no way less important, but it is often easier as you already have common ground working for the same company. You may also work with 3rd party vendors and that is a different relationship because they are business partners who work for your organization. You should offer everyone that you work with the same level of professionalism and consideration of their needs. I have worked on internal products where the project team was mostly internal with a few external vendors. The atmosphere during meetings was much less casual and there was more chit-chat, but we were still focused on the project and committed to building the best products possible.

This was just a quick summary of this topic. Later in this book, I devoted a full chapter to help you oversee clients and stakeholders with real-world examples.

1.2.4 Setting Development Standards

An especially important part of your job as a Lead Developer will be setting and maintaining development standards. You want to ensure that the development team is using the same coding style. This will bring symmetry to your codebase and the goal is to not be able to tell who the developer is just by looking at the code. One of the main reasons that organizations have standards is so that any developer can easily find the code they are looking for to fix a bug or improve a feature. This helps improve productivity as developers do not always have to reverse engineer code to find what they need.

Development standards should be as granular as possible including items such as where to place the curly brace after an if statement, adding comments before every function, and especially naming conventions. It is helpful to have a clear pattern in your file names, modules, functions, and features.

Whether you are working on products or services, you need to ensure that the development team adheres to the development standards. These standards must be well-documented and reviewed with every team member when they are being onboarded to the project. During the onboarding process, the Lead Developer should review the development standards with everyone on the team to ensure full adoption. During code review, standards should be enforced. You can also find common errors during code reviews that you can discuss with the entire team to ensure that everyone understands the development standards.

When you set development standards for a project, you may or may not have standards at the organizational level to rely on. If development standards exist, your priority is to learn and maintain them. Organizations may have custom libraries that should be included in all projects or a specific approach they take to technical architecture. These standards should change over time as the tech stacks your organization uses are upgraded or when new technologies come out that you want to adopt. Lead Developers can also influence the development standards at the organizational level, and you should never shy away from presenting new ideas for improvements.

Maintaining development standards is essential as technology is changing all the time. Lead Developers need to keep their skills current and alert the project team and potentially the organization about updates that need to be made. Subscribing to newsletters for the technologies that you’re using is a great way to keep up with the latest news and updates. Being active in the tech community and networking with people on social media is another way to keep up to date. Almost every technology has a community on Slack or Discord that is run either by a company or by community members.

I work with a lot of different technologies including Docker and they have an excellent community Slack and newsletters. Their Slack provides you with access to Docker Captains, Docker Community Leaders, and Docker employees who are always available to answer your questions. I am also in similar groups for .NET and Sitecore where we share current standards and updates. This helps me stay up to date with frequently changing standards and practices.

You can also rely on your development team to suggest improvements to the development standards. Every developer should have a say in the development approach. When people from different backgrounds come together to share ideas, this drives innovation. Including them in the standards process is a great way to motivate the team to keep their skills current by engaging in continuous learning.

1.2.5 Building Technical Architecture

Lead Developers are expected to understand how to build technical architecture from scratch using current industry standards and practices. You will need to work with project teams to ensure that the approach you are implementing fits within the budget but also provides a stable system. Technical architecture is usually illustrated as a flowchart in the project documentation so you will need to create flowcharts using the tools at hand.

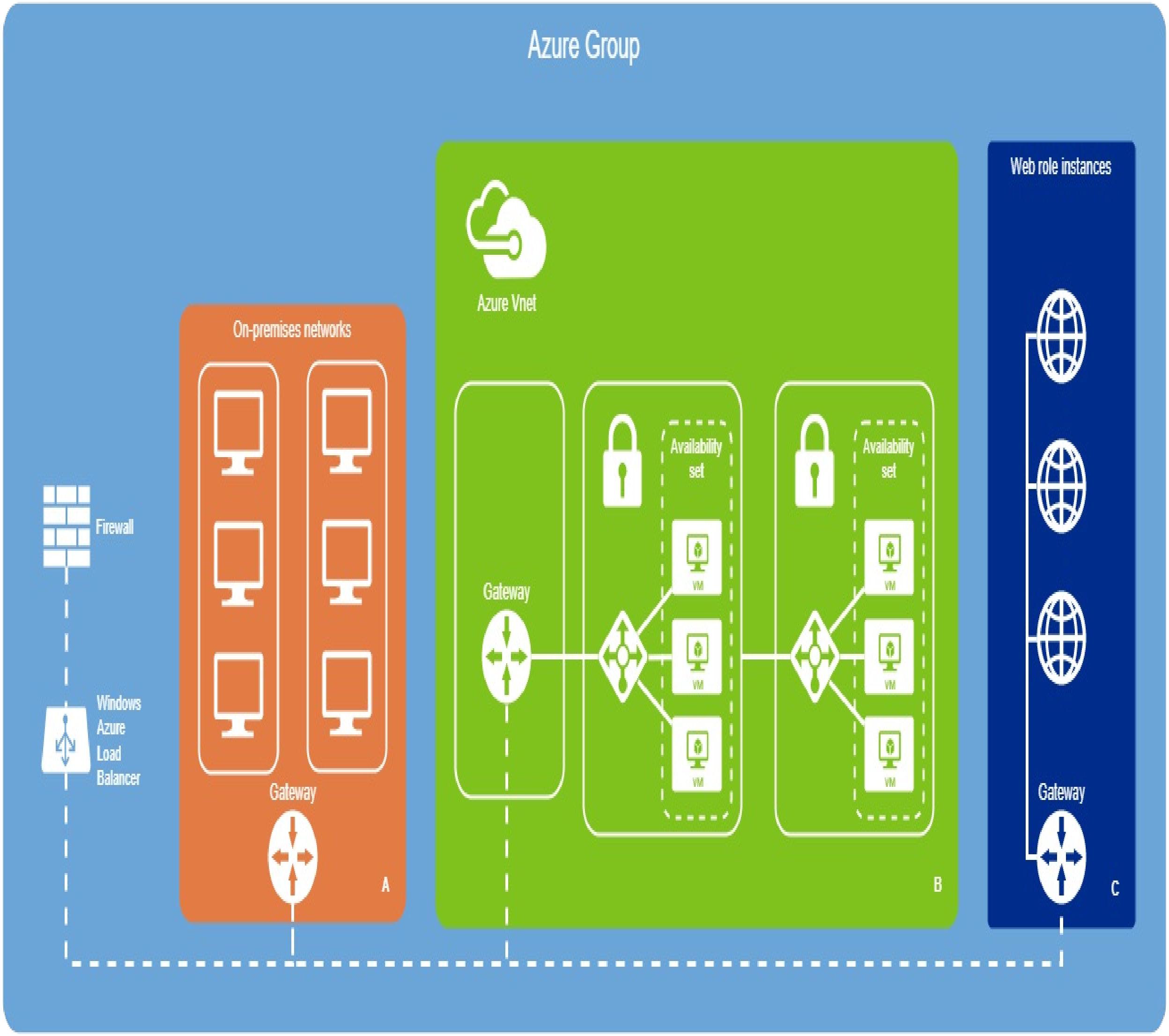
Being able to build out the technical architecture of a project from scratch is not a skill that is easy to come by. You may be able to build a simple application using a tutorial but when it comes to consumer products and enterprise solutions, you will need to understand much more advanced concepts. This takes time and experience and when we start as Junior Developers, our main task is to fix bugs or add features to already existing projects. The best way to learn how to build systems from scratch is to shadow and observe what the Lead Developer is doing. Ask them questions, get copies of the documentation they’re using, and work on prototypes or personal projects. Working on personal projects is a great way to learn new skills and it should be fun. You can get ideas for projects and guidance from boot camps, online courses, and architecture pattern books. Case studies are also a good resource, and you can find them by searching for the specific technology you are using. You can build yourself a portfolio website, app, or software that is related to your industry. If you become part of a community of developers, you should share your code with them for support.

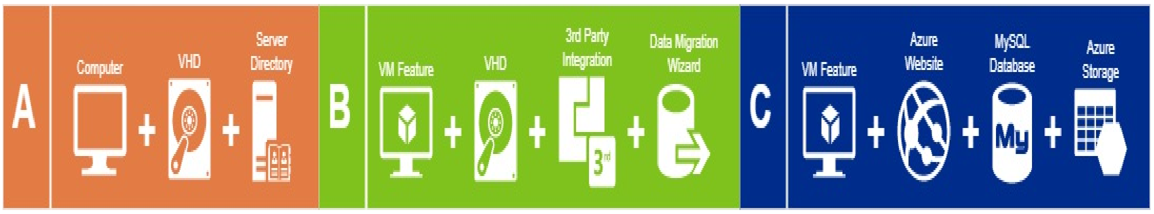
In my experience, I have learned a lot from other developers within the organization. I would ask them for advice about the project I was working on even if they were not working on that project. It is a good practice to get an outside opinion from someone who is impartial because they can often see things that you cannot. I was often asked to help on projects that I did not work on, and we would discuss what worked well with our projects and what did not. This helped me learn more about the organizational approach to their products and keep my skills up to date.

One of your tasks as a Lead Developer will be to help plan out the budget for the technical architecture. This includes items such as server costs, software, web hosting, databases, and security protocols. The Lead Developer must get as granular as possible with these costs upfront to avoid a price hike during the project. Furthermore, whenever you upgrade a project, you should reassess these costs and try to find solutions that will minimize the expense for your organization. To estimate server costs, there are tools you can use such as the AWS or Azure cost calculators. Most cloud providers have tools to help you with estimates including software and hosting as well. The budget for the technical architecture must include everything needed for every instance including development, testing, staging, and production.

The Lead Developer is also responsible for creating an illustration of the technical architecture, usually in the form of a flowchart or diagram. Doing this will help you envision everything that you need to estimate for the budget. The illustration also shows how systems interact with each other as well as the users. The overall purpose of a technical diagram is to plan how to build your project from the ground up including every stage of the software development lifecycle.

Figure 1.6 Example Technical Architecture Diagram





When you're creating technical architecture diagrams for projects, you should try to be as consistent as possible. Check to see if your organization already has standards for creating technical architecture diagrams that you can follow. Make sure that the usage of shapes, lines, colors, and legends follow the same pattern to avoid miscommunication. These diagrams are crucial to the project’s success as it provides a clear map to show the team how everything will be built from start to finish and support the budget. Throughout this book, we will discuss creating and presenting technical architecture diagrams to clients and stakeholders.

1.3 Lead Developer Expectations

Now that we have discussed Lead Developer tasks, we need to talk about the expectations for that role. It is important to understand the difference between responsibilities and expectations. Responsibilities are the tasks and duties that you're accountable for in your role, while expectations are the standards that others have of you. It's essential to take ownership of your responsibilities and ensure that you fulfill them to the best of your abilities. At the same time, you should also be aware of the expectations others have of you, such as being a good communicator, mentor, and leader. Balancing both your responsibilities and expectations can be challenging, but it's crucial for your success and the success of your team. Remember to always approach your role with a positive attitude and a willingness to learn and grow.

The Lead Developer role comes with a lot of responsibilities, and you will be held to a higher standard than Junior or Senior Developers. A successful Lead Developer must be highly communicative and show true leadership skills. This is a completely different skill set than development work and you will need to find a balance between your technical skills and leadership skills. When you’re working with project teams, they expect to be able to rely on you heavily to support everyone on the team, not just the development team. When you’re working with the development team, you are expected to deliver requirements from the project team to the developers who are implementing the features or fixing bugs.

The following list contains the main leadership expectations of a Lead Developer:

- Providing Team Support
- Forming Working Relationships
- Being a Leader

When you’re working with non-developers such as Marketing, QA, Product Designers, and Executives, they will expect you to speak their language. If you use too many technical terms, they may get frustrated if they don’t understand. Cultivating a language style that is technical but doesn’t go too far into the weeds of specific development tasks will take time and a lot of observations of how the non-developers on the team speak to one another. Remember, they do have technical skills and you can always coach them a bit to support them and help them learn more about the technologies you are using in a project.

1.3.1 Providing Team Support

One of your main expectations as a Lead Developer is to provide support to your team. This entails being communicative and answering questions as quickly as possible. This is a hard task as you are the technical expert and as such, you will most likely be inundated with questions from the team. You also have your work to do so you can’t stop everything you’re doing every time someone pings you with a question. You must ensure that both your needs and the needs of your team are met.

The hardest part about being communicative is when you’re concentrating on doing your development work. If you stop when you’re in the middle of something, you will lose productivity because it is hard to get started again, especially if you’re working on a very complex feature or task. My rule for myself and the Lead Developers I have hired is to try to answer questions within one hour. This will give you some time to wrap up what you’re doing or at least get to a good stopping point so that you don’t lose productivity. We live in a world where people are very busy and tend to want immediate answers. You must set some boundaries so that you can accomplish your tasks, but a compromise must be made to ensure the success of the team and you need to find your balance.

A good way to reduce the number of questions you get daily is to anticipate the needs of the team. When you are asked questions in a meeting, for example, you should explain the answer to its full and logical conclusion and don’t leave anything out. A lot of the time, people will ask a question in a meeting and then ask for further clarification after they have had time to think about it. Put yourself in their shoes and anticipate the information that they will need to get their tasks done. If someone is asking the same question over and over, don’t get frustrated. They are asking the same question because they may not understand the answer or there is no documentation. Taking the time to answer their questions and asking them to reiterate their understanding will help to mitigate this issue while increasing productivity.

Documenting FAQs in a central document repository and sharing it with the entire team is also a great way to ensure that everyone knows where to look for the information they need which will also help you balance your work. You should maintain separate FAQs for non-developers and developers on the project team. For developer FAQs, adding documentation directly to the code repository is a popular approach. The entire team should have access to add FAQs on their own and the Lead Developer should work with them to ensure that the answers are correct and there are no duplicates.

It also helps to create a glossary of terms to help non-developers understand the language that is being used. The terms should include everything that is included in the architecture diagram, and it should be linked to the diagram. This is a great reference for everyone on the team, including developers. The glossary should also include terms such as acronyms written in full including an explanation of what it is and what it does. Some acronyms are organization-specific so if anyone Googles that term, they may find other results with the same or similar acronym which will confuse them even more.

When you’re writing project documentation including a glossary, you should format it for readability and maximum retention. Most people organize a glossary in alphabetical order, but you can also group terms by industry vs. organizational terminology. I like to include a background knowledge section for industry-standard terms that already have their documentation. The background knowledge section should have links to existing documentation for industry terms and concepts to support different roles and varying degrees of expertise on the team.

When you go the extra mile to anticipate the needs of your team and provide them with the support, they need to be successful, you are also supporting your success as you can be more productive. Writing documentation and glossaries for an entire team is not an easy task as there are many different roles and experience levels within the team. A successful Lead Developer needs to understand how the people on the team think, what information they need to know, and how to help them learn.

1.3.2 Forming Working Relationships

Lead Developers must form working relationships so that there is cohesion, and the team works well together. This entails respecting the expertise of your team members, appreciating them, and having open communication. Developing trust and comradery amongst your team members can lead to a more productive and enjoyable work environment. When you take the time to get to know your colleagues, you're able to understand their strengths and weaknesses, communication styles, and work preferences. This can help you to better delegate tasks and foster a sense of collaboration, which can ultimately lead to more successful outcomes. When you have good working relationships, you will be better able to support the needs of your team.

Respecting the expertise of your co-workers seems like a no-brainer but you must be self-aware to be sure that you are being respectful. You should respect everyone you work with including developers, Project Managers, clients, etc. We are all prone to unconscious bias as it is built into our culture, and you must raise your level of self-awareness to ensure that you are being respectful. Unconscious bias is a social stereotype someone automatically forms about a person or group of people. It’s often unintentional and outside of their conscious awareness, so it can take time for people to realize that they’re creating unconscious biases toward others. You and your team need to know how to properly identify unconscious bias and immediately stop it.

The following list contains the most common examples of unconscious bias:

- Ageism – The idea older people slow down and therefore are not as competent or capable at work
- Beauty Bias – Unfairly judging people based on their appearance
- Conformity Bias - This is basic peer pressure, and it happens when a person is pressured to agree with the opinion of the entire group, even if they have stated that they disagree
- Affinity Bias – Being drawn to people who are like you

- Confirmation Bias - Actively seeking to confirm ideas or success they had in the past that has little to no bearing on the current situation
- Attribution Bias. Making assumptions about others based on their actions to understand why they're acting that way
- Gender Bias – Favoring one gender over another
- Ableism – Prejudice against individuals with disabilities or those who are perceived as having disabilities
- Cultural Bias – The tendency to judge or evaluate people, events, or behaviors based on the standards and values of one's own culture or group, without considering the cultural norms and values of other groups
- Racism - A system of prejudice, discrimination, and power based on the belief that certain races are superior or inferior to others

Being self-aware is your first step in combatting unconscious bias and being more respectful to your co-workers. Lead Developers should ensure that they have a diverse team so that there is an even amount of representation. You need the expertise of people from different backgrounds to create products and services that are usable by your target audience. A Lead Developer needs to be open to learning from anyone. You can learn a lot from both non-developers and developers and rely on their expertise when you keep an open mind.

Good working relationships are formed by appreciating your co-workers. Tell the project team during a meeting how successful someone was in building a feature or fixing a bug. Say "thank you" when someone answers your questions or helps you find the answers that you need. Take your team out to lunch occasionally. Or, if you're remote, schedule a fun activity that you can do together online such as a virtual happy hour, playing video games, or playing Dungeons and Dragons.

Open and consistent communication is a key component of every good working relationship. This can lead to having difficult conversations; however, moving through those conversations becomes easier when you already have a relationship with that person or group of people. When you're working with internal co-workers, don't lie and don't sugarcoat things. When there's a problem, let the team know and be honest and direct. It is so much worse when you try to cover things to protect yourself or your team. If you are not honest or consistent with your messaging, your team will lose trust in you. They may also follow your lead and avoid conflict or be dishonest themselves which will only lead to more conflict. When you are consistent in your honesty, then people will come to expect it and they will respect you for it.

1.3.3 Being a Leader

A leader is a person who is in a position of authority or influence, and who is able to guide and direct others. Leaders may be responsible for making decisions, setting goals, and developing strategies for achieving those goals. They may also be responsible for inspiring and motivating others, and for building and maintaining relationships within a team or organization. Leadership styles and approaches can vary widely and may depend on the specific goals and needs of an organization, as well as the personalities and abilities of individual leaders.

Finding your leadership style is important and throughout this book, we will review current best practices. As a Lead Developer, you can’t “set it and forget it”, in other words, you need to check back in on things from time to time. You will be expected to make decisions, ensure accountability, and lead change. You want to inspire your team and lead by example while at the same time understanding that you cannot please everyone.

Making decisions takes a high level of critical thinking. A Lead Developer must consider risk, form a plan, and execute that plan. You need to keep a level head during this process so that you are not making baseless decisions or going off course. Part of decision-making is ensuring that the action plan is carried out as you described. Therefore, you should check in throughout the project because people may deviate from the plan. When you have an action plan, you cannot set it and forget it. You must work together with your team to gather consistent feedback to ensure that everything is going as planned. When things are not going as planned, sharing information among the team can help you get back on track. Communicating the action plan should not include just telling your team what the plan is, but rather sharing information so that you can make informed decisions.

As a Lead Developer, everyone on your team is looking to you and you must lead by example. Establishing accountability for yourself is one of the first steps in becoming a true leader. You need to hold yourself accountable for deadlines and follow-through. If you promise to do something, then do it. If for some reason you cannot, then let the person know when they can expect what you promised. You should also apologize when you drop the ball on a project or offend someone unintentionally. Some people avoid apologizing because they don’t want to admit that they’re wrong and sometimes they fear that they may lose their job over it. This should never be the case and you need to show your team that apologizing when you’re wrong is a part of the team’s culture.

Leading by example is an important concept for all leaders to master. When you show your values through your behavior, then others will notice and follow your lead. At my first job as a Lead Developer, I was constantly promising that tasks would be done by a certain deadline and then I was so overloaded with work that I could not live up to that promise. Of course, you want to be enthusiastic about your work, but you must ensure that you are able to do the work in the amount of time given. I eventually learned how to push back on things that were not a priority so that my schedule was more realistic. In doing that, I found that my team felt comfortable coming to me when they felt overloaded so that we could work out a plan together.

We live in a world of change and people often do not like change because it can cause them anxiety. Anxiety is one of the most common mental health disorders and it impacts almost everyone at some level. Being a positive leader of change for your project team and organization will ensure your success as a Lead Developer and provide a safe and trusting workplace to mitigate anxiety among your team. Whether the change is suggested by you or another person on your team, you are expected to carry it out. You need to set an example by following the purpose and vision of the project while proposing necessary changes. Another component of this is to empower your team to be change leaders as well. They should feel comfortable sharing ideas, proposing new features, and streamlining processes. This will help raise their confidence level and give them the support that they need to develop their careers.

Inspiring the best in your team will help you draw out their talent. People are at their most productive when they feel supported, and you take an interest in their careers. However, being inspirational doesn’t mean that you will always agree with everyone. You do not have to be a people pleaser to be inspirational. Being a people pleaser is dangerous and it can cause unintentional conflict if you tell one person one thing, and another person another thing. They may feel lied to or even betrayed and then you need to sort that out. It is better to handle difficult conversations upfront and tell everyone why a decision must be made. If someone is not happy about it, you should take them aside and listen to their concerns. Even if they are not happy with the decision, when you listen to co-workers, they will feel respected and appreciated which will inspire them to treat others the same way.

Being a leader applies to other roles in development besides Lead Developer. Everyone at every level of their career should learn leadership skills to help prepare them for the next phase of their career. In the next chapter, we will talk about your career trajectory from Junior to Lead Developer and beyond.

1.4 Case Study

I have been a developer since 1998 and my career has spanned many different industries and job titles. I am currently the Co-owner at HoffsTech, LLC where I write and produce technical content including online courses with my husband, Jason Benhoff. My technical skills are mainly in web development using .NET/C#/MVC, Sitecore CMS, Azure DevOps, Docker, and Kubernetes. I am a 4x Sitecore MVP and Docker Community Leader. I speak at conferences around the world, and I am very passionate about mentoring future technology leaders. In the following case study, I will discuss my career leading up to and including being a Lead Developer to help you gain insight into the role and what it means to be a Lead Developer.

1.4.1 What Was Your First Experience as a Lead Developer?

I first became a Lead Developer after leaving a job where I was a manager, so my career trajectory was a bit different. Before I was a manager, I was a web developer and trainer. My responsibilities included creating courseware for developers to learn a specific platform. I had very complementary technical skills and I was highly organized, so I got promoted to management.

However, I had no management experience. I also had no project management experience, and I did not realize that I would manage both the team and the projects. I tried to ask management for training in these skills, but I never received it. So, I leveraged the knowledge of people on my team and other managers within the organization. I cannot say I was successful in that job (I was not really set up for success), but I did learn a lot.

When I left that company, I wanted to get back to my developer roots and I was offered a Lead Developer role at an agency specializing in web development. This was the classic Lead Developer role where I had a project team, and I represented the development team to project stakeholders. Even with my management experience, I felt a bit like a fish out of water because, unlike my previous job, I was now speaking with external clients. This came with an extra level of stress for me because I wanted to ensure that the clients were happy with our team's work.

I struggled in pretty much all aspects of being a Lead Developer at first. I have somewhat of an introverted personality (at least until I get to know people) so my main struggle was speaking up and asserting my opinion. I also tried to please everyone, which is a topic we will get into in a later chapter. My technical abilities were fine, but I struggled in meetings, and I avoided conflict at all costs which I learned only leads to more conflict.

Over the years, I found my way by engaging with multiple mentors and reading leadership publications such as the Harvard Business Review. I also learned to lean on my team for advice and not be afraid to tell someone that I do not know the answer to their question. I failed a lot, but I always learned valuable lessons from failures which helped me in the long run.

1.4.2 What Did You Enjoy the Most as a Lead Developer?

My greatest joy as a Lead Developer has been watching my mentees achieve success. Being a support system for the team comes with a high level of responsibility and when one person is successful, we are all successful. One of my greatest success stories was when I worked with a very talented developer who did not want to come out of his comfort zone. But he also wanted to be a Lead Developer, so I coached him in the areas he was struggling with including being confident in his technical approach and presenting his approach to the project team. He was not comfortable with presentations at first and I told him that I get nervous every single time I present. Sometimes, it is so bad that I have a hand towel on my desk because my hands get sweaty. He found that very relatable and told me that he was inspired by my willingness to go outside of my comfort zone. I told him that his career will take a major turn when he is able to go outside of his comfort zone. He is an engineering manager now at a big company and I am so proud of him!

1.4.3 How Did Your Experience as a Lead Developer Set You Up for Success in Your Current Job?

I am currently the Co-Owner at HoffsTech, LLC and we produce technical content. This is very different from being a Lead Developer because I am working with technical writers, editors, and producers instead of developers and clients. My experience as a Lead Developer has set me up for success in my new role as I oversee hiring and overall team management. I mentor everyone who works with me whether they are an employee of my company or a freelancer. I add mentoring opportunities to any job description that I post because I want potential candidates to know that they will be supported which often attracts top talent. I understand how to manage the day-to-day tasks of my team and ensure that our processes are reviewed and updated constantly.

Being a Lead Developer taught me not only how to lead but also what true leadership means. I learned how to stay on my toes and figure out how to handle constant changes while remaining (relatively) calm. I understand how to resolve conflict in my team without blaming anyone. The main thing that I learned that has helped me the most was the realization that people work with me, not for me. That change in my mindset helped me be a better mentor and leader because my job is not to tell people what to do. My job is to draw out their talent by working together to figure out what to do.

1.5 Summary

- Lead Developers are responsible for leading and mentoring development teams as well as communicating with project teams.
- Lead Developers can come from any background including any gender, race, and education level.
- The day-to-day tasks of a Lead Developer include leading the development team, working with project teams, communicating with clients and stakeholders, setting development standards, and building technical architecture.
- The expectations of a Lead Developer are to provide team support, form working relationships, and be a leader.
- Lead Developers must possess technical expertise and be capable of writing technical specifications, developing features, troubleshooting errors, and deploying code.
- Lead Developers must learn leadership skills including facilitating communication, decision-making, gathering feedback, emotional intelligence, and self-awareness.
- The career path for a Lead Developer normally begins with Junior Developer, then Senior Developer, and Developer Advocate/Developer Relations.
- The success of a Lead Developer is reflected in the success of the development team and can be measured in the amount and quality of work completed in an iteration or release.

2 Lead Developer Career Trajectory

This chapter covers

- Understanding the possible roles leading up to the Lead Developer role.
- Reviewing options for management and technical positions that are available beyond the Lead Developer role.
- Writing high-quality resumes and cover letters that will get you interviews.
- Achieving success in technical interviews by asking the right questions and being prepared.
- Standing out above the competition with excellent writing, presentation, and interviewing skills.
- Reviewing in-demand technical skills.

Everyone must start somewhere, and no one gets a job as a Lead Developer with no experience. The most common path is Junior Developer > Senior Developer > Lead Developer > Management. However, this is not the only path, and, in this chapter, we will explore various options for your Lead Developer career trajectory.

Once you have a clear career path in mind, you must write a winning resume to get in the door for interviews. Not only does a strong resume help you stand out in a competitive job market, but it also helps you secure more lucrative job offers and better career opportunities. By highlighting your strengths and accomplishments, you can communicate your value proposition to potential employers and convince them that you're the right person for the job. Investing time and effort in creating a winning resume can pay off in numerous ways, both in terms of your career advancement and your personal growth.

Technical interviews are the hardest part about landing any Developer position. As a Lead Developer, you are expected to be a technical expert and you will be tested as such. Being prepared by asking the right questions beforehand will help you achieve success in these sweat-inducing interviews.

To start planning your career trajectory, you need to understand the Software Developer roles that are available to you and how these roles support your efforts in becoming a Lead Developer.

2.1 Reviewing Software Developer Roles

When you’re looking for jobs, it can be difficult to stick to one role especially after being laid off and needing another job quickly. It’s even harder for people who are on a work visa that is sponsored by their employer because they have the extra hurdle of not only looking for the right role but also needing a company that offers sponsorships for foreign visas.

When I first started, I could not find any Junior Developer positions that were open to people with 0 years of experience. I am very adamant that the Junior Developer role should not require any experience but sadly this is still an issue today. Instead of becoming a Junior Developer, I started out working Help Desk jobs. There are many people who champion this route because Help Desk Representatives are exposed to various technologies and processes. Help Desk work also helps you learn critical thinking skills by troubleshooting errors.

Throughout my career, I have deviated from the Developer career path several times, but I still landed Lead Developer positions in the end. I have had roles including Help Desk Representative, Web Developer, Courseware Designer, Manager, and Lead Developer, and now I’m a Business Owner! There is no career path that is set in stone. You just make it up as you go.

2.1.1 Starting as a Junior Developer

A Junior Developer is a person who possesses basic programming skills that they learned at university, by attending a boot camp, or by practicing on their own. They generally have 0-3 years of experience, but I wish employers would revise that to allow people with 0 years of experience. This means that you need to have experience in development before you get a job as a Junior

Developer.

The following list contains the main requirements for Junior Developer jobs:

- 0-3 Years of Experience
- Basic Programming Skills
- Detail Oriented
- Willingness and Ability to Learn New Skills
- Ability to Work on Teams

Companies are looking for someone who has a general knowledge of programming and development. They may not have the specific skills for the tech stack that the company is using but they should be able to learn quickly. They need to have the capacity to learn on the job and apply their knowledge to the task at hand. We are often placed in positions throughout our careers as Developers where we are asked to work with a technology that is not familiar to us. Learning how to break concepts down and find the documentation that you need are some of the key skills you will learn as a Junior Developer.

Since Junior Developers are new to the industry and to learning organizational processes, they often spend most of their time researching best practices, asking questions, and taking advice from Senior and Lead Developers. One of the hardest parts of being a Junior Developer is feeling like you know nothing and are a burden to your team. Junior Developers are a critical part of any organization. Without training and mentoring Junior Developers, we would never have any Senior or Lead Developers!

The following list contains the main responsibilities of a Junior Developer:

- Continuously Improve Programming Skills
- Assist the Development Team
- Attend Team Meetings
- Troubleshoot Errors and Fix Bugs

Junior Developers, focus on learning and applying their knowledge to assist the development team in software design and coding. They will learn a lot in this role and their skills will improve over the years as they gain experience. Critical thinking is a skill that anyone should master in this role as Junior

Developers work on bug fixes and errors. Tracking down the source of an error is not always easy and sometimes it seems impossible. Learning how to troubleshoot effectively will help Junior Developers achieve success when they move into a Senior Developer position.

2.1.2 Becoming a Senior Developer

A Senior Developer must have technical expertise as well as mentoring skills. As Junior Developers gain experience, they should start to help their peers learn new skills by guiding them to the proper resources. Learning to mentor your team is a skill that you will need to learn to become a Senior Developer.

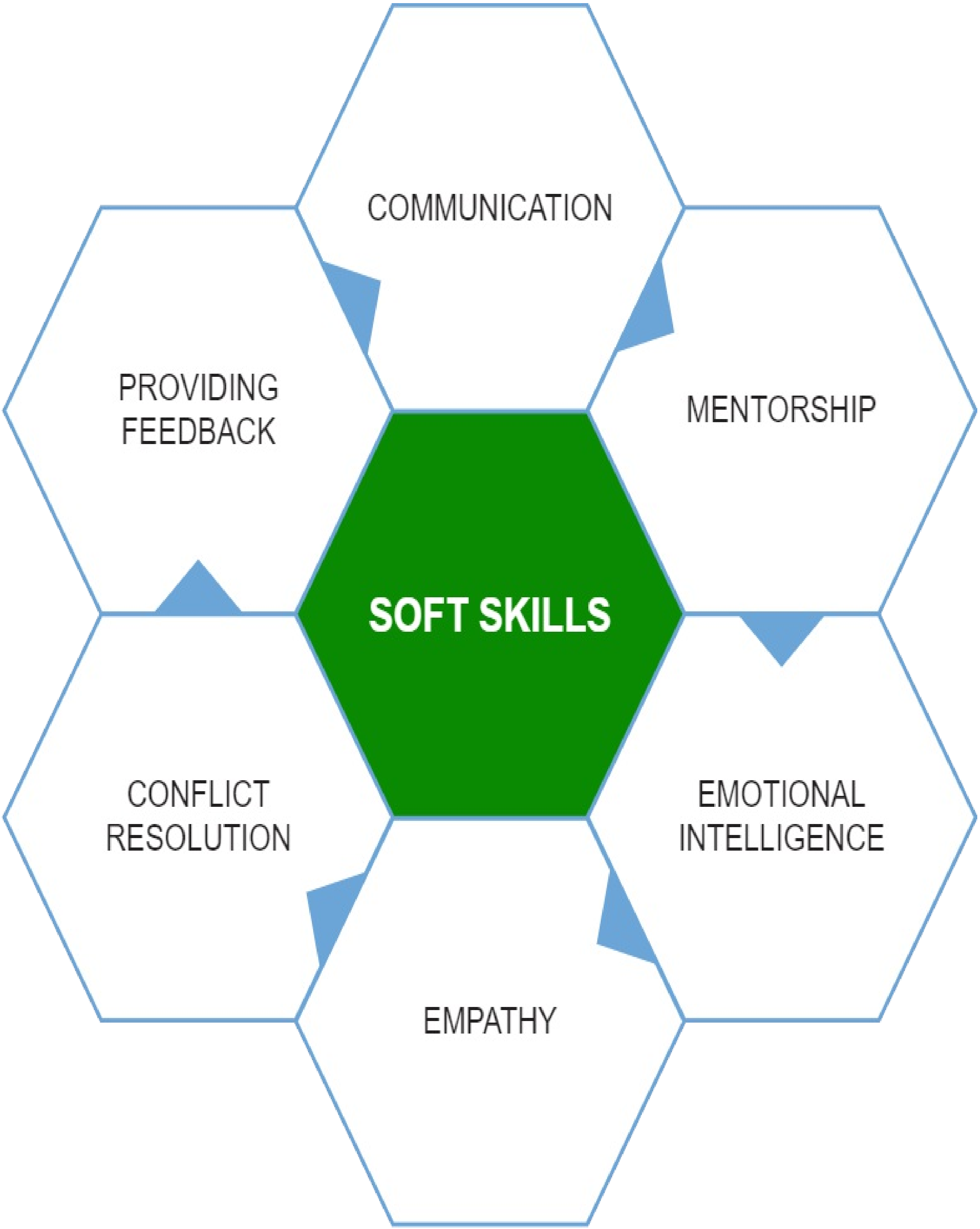
The following list contains the main requirements for Senior Developer jobs:

- More Than 3 Years of Experience
- Experience Working with Complex Projects
- Successfully Implemented High-Quality Features
- Navigate All Phases of the Software Development Lifecycle (SDLC)
- Self-Manage Tasks and Projects

These requirements are variable based on the organization. Some companies also offer a mid-level Developer role to help you move your career forward and this role is basically an extension of the Junior Developer role. The major difference is your level of experience and expertise, but your responsibilities are the same. I was a Junior Developer for 5 years before I was promoted to mid-level developer. I went straight to management from there which makes me think that I should have either been promoted from Junior to Senior or I should have received another promotion after becoming a mid-level developer. But I never asked to become a Senior Developer and promotions are rarely given automatically. This is a topic we will dive into in a later chapter.

When you are a Senior Developer, you are expected to mentor others, communicate your opinions on technical planning, and be independent. You also need to be comfortable speaking up in Development team meetings. Senior Developers must understand higher-level technical concepts and be able to apply them to build new features. Junior Developers are often assigned bugs to help them get acclimated to the programming languages used for specific projects. As you progress in your role as a Senior Developer, there are several key soft skills that you should focus on before becoming a Lead Developer.

Figure 2.1 Soft Skills



When I am hiring Senior Developers, I am looking for people who are willing and able to assist the team and support their success. They must be highly skilled in the tech stack the company is already using and have experience with the preferred programming language within the organization. This does not mean that they must be an expert in every programming language used by the company, and in my experience that is often hard to find. Instead of looking for someone who possesses 100% of the skills listed in the job description, companies look at their ability to learn new skills quickly.

Table 2.1 Junior Developer vs. Senior Developer Responsibilities

	Junior Developer	Senior Developer
Continuously Improve Programming Skills	✔	✔
Assist the Development Team	✔	✔
Attend Team Meetings	✔	✔
Troubleshoot Errors and Fix Bugs	✔	✔
Developing Basic Features		✔
Mentor Junior Developers		✔

As you grow in the Senior Developer role, it is important to observe the Lead Developers who work with you and learn from them. Watch how they create architectural diagrams and offer to help them write documentation. You can also find opportunities to show initiative by taking on responsibilities including presenting technical information and identifying areas of improvement. Reflecting on what went wrong with a project and lessons learned is a great way to improve your leadership skills. Showing that you have these soft skills will set you on your way to becoming a Lead Developer.

2.1.3 Moving to Lead Developer or Lead Architect

There are a few different variants for the Lead Developer position. A Lead Architect is like a Lead Developer, but they are mainly responsible for detailing what the technical architecture is and how to build it. They may also draw diagrams to illustrate how each component in the technical architecture is connected to each other. They are the people who plan what to do and how to do it. In contrast, a Lead Developer has the skills of a Lead Architect, but they also execute the plan and write code. They are responsible for maintaining coding best practices and standards which require the highest level of technical knowledge. Lead Developers must have full expertise in every aspect of a project at the technical level whereas Lead Architects do not have to be experts to plan projects.

Table 2.2 Lead Developer vs. Lead Architect Skills

	Lead Developer	Lead Architect
Create Technical Architecture Diagrams	✔	✔
Explain How the Technical Architecture Will Be Built	✔	✔

Illustrate Component/Module/Feature Relationships	✔	✔
Estimate Technical Tasks	✔	
Develop Infrastructure and Features	✔	
Mentor the Development Team	✔	

Some companies split these roles into two different positions but most of the time, it is a combined role. This is because generally a Lead Developer is expected to possess all the skills of a Lead Architect as well. Hiring managers are looking for Lead Developers who can build systems from the ground up and be able to conduct code reviews which require programming expertise. You may also be required to build the initial infrastructure for a project and write instructions for your team members to configure their local development environments.

As a Lead Developer, you need to understand the business side of things as well as the technical side. Part of project planning includes meetings with clients and stakeholders to list their business requirements. Understanding what they expect to see as a result is a skill that you must master. You will also work with the Project Manager to help them plan the budget and hopefully avoid increasing project costs during the development phase.

Lead Developers are expected to translate business requirements into technical requirements which is why they need to understand product domain. You will not be able to plan accurate technical requirements or estimates if you do not fully understand what the client or stakeholder needs. If your estimates are off, you need to work with your project team including the stakeholders to ensure that you are all on the same page. You need to think about not only the result but why the stakeholders need this result. What are their business drivers? Are they increasing brand awareness, supporting their end users, or driving sales? Usually, it’s a combination of things. If you are working at an agency or company delivering products and services to customers, you need to think about what the project you’re working on means to the company. The main goal in this situation is usually customer retention because you want people to be loyal to your company or brand.

The following list contains the business skills required for Lead Developer jobs:

- Assist in Project Planning
- Work with Budgets
- Understand Business Requirements
- Understand Business Goals

There are also important soft skills that a Lead Developer is expected to possess. When your team members ask questions or need help, you will be the person they look to for answers. If you don’t know the answer, you need to help your team find the answers they need from internal co-workers or external clients, stakeholders, and vendors. You can ask the Senior Developers on your team for their input as well and give them the opportunity to shine. It is also important to document the answers to avoid repeat questions. Lead Developers need to lead by example and provide their team with the skills they need to grow in their career. This is done by recognizing learning opportunities and facilitating communication between teams which will help the team members understand how to self-manage.

Lead Developers must be comfortable communicating with clients and stakeholders and presenting information to them. This may involve informal conversations or formal presentations. Becoming familiar with creating effective slides will help make your presentations engaging for the audience. Public speaking is not always easy for everyone, especially if you are an introvert. Employers are looking for Lead Developers who can communicate effectively with people both internally and externally. A Lead Developer is the face of the development team, and they are required to have excellent communication skills.

The following list contains the soft skills required for Lead Developer jobs:

- Mentoring Other Team Members
- Presenting Information to Project Teams
- Facilitating Communication Between Teams
- Communicating Directly with Clients and Stakeholders
- Leading with empathy and emotional intelligence
- Providing feedback
- Resolving conflict

The combination of technical, business and soft skills are what companies look for in a successful candidate for a Lead Developer position. You need to show that you have a proven track record of success in planning and implementing technical architecture, assisting with project planning, understanding the business side of things, and being able to lead teams. Learning business skills and soft skills to complement your technical expertise will help you be successful if you want to go into management and leadership.

2.1.4 Considering Management and Beyond

If you enjoy the business side of being a Lead Developer, you may want to consider a career in management and executive leadership. There are many opportunities available to you as an Engineering Manager, Technical Director, or even CTO. You will not write code as a part of your daily tasks, but your experience as a developer will greatly aid your success as a manager. If you do not want to lose your programming skills, you can work on personal projects or study skills with the development team. The best managers I’ve had asked questions about programming and were open to learning from anyone. Learning is a two-way street and managers can learn from any member of the team.

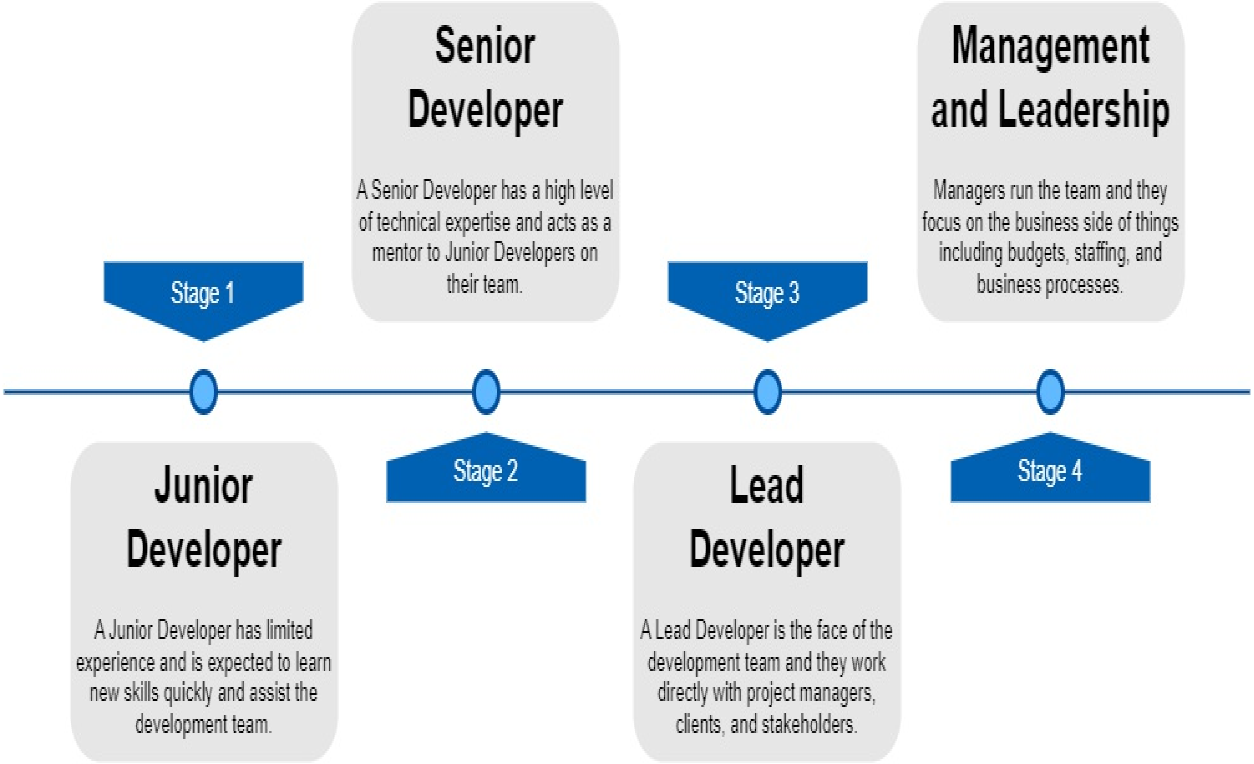
If you want to continue coding daily, then management may not be a fit for you and that’s fine! Even if you enjoy business skills, which doesn’t mean that you must push yourself into a management position. We often focus on an upwardly mobile career trajectory but keep in mind that once you become a Senior or Lead Developer, you can stay in that role for the remainder of your career. Management and Leadership require different skill sets, especially when you’re managing people.

As a manager, one of the hardest skills to master is hiring. You want to find the best talent for your team, which requires the ability to read people and assess their skills. They need to be a fit for your organization and your types of projects. You must hire developers at all levels whose skills complement one another. If you hire the wrong people or do not support the success of the right people, you will end up managing turnover for your team. Managers must be able to have difficult conversations and resolve conflicts. Lacking these skills will result in high turnover and that is costly as you lose talent.

Managers handle the budget for the entire team. This includes project costs and salaries. It is the Managers’ responsibility to ensure that projects are delivered within the proposed budget and the result is high quality. This involves a lot of spreadsheets and meetings with upper management. You must be prepared to state your case when project costs increase and work with the management team to make decisions to get the project back on track. This is a complex system that requires proactive thinking to avoid budget issues in the future.

Having solid business processes helps to keep your budget on track and increases the productivity of your team. Processes should be updated on a regular basis to keep up with changes in the industry. Holding on to outdated processes will set your team back, even if those processes have always worked. Managers must be up to date with best practices in development and implement new systems and technologies to ensure a high level of productivity.

Figure 2.2 Software Development Career Trajectory


Stage 1
Junior Developer
A Junior Developer has limited experience and is expected to learn new skills quickly and assist the development team.
Stage 2
Senior Developer
A Senior Developer has a high level of technical expertise and acts as a mentor to Junior Developers on their team.
Stage 3
Lead Developer
A Lead Developer is the face of the development team and they work directly with project managers, clients, and stakeholders.
Stage 4
Management and Leadership
Managers run the team and they focus on the business side of things including budgets, staffing, and business processes.

This career trajectory is the standard, but it is not the only path you can take as a software developer. There are many different paths that can set you up for success as a Lead Developer or Manager. Everyone is different so you should carve out the career path that is right for you.

2.1.5 Considering Your Career Options

A lot of people think that you must get a degree in Computer Science to be a developer. This is not the case anymore as boot camps and certification programs have become more popular. I know a lot of people who studied non-technical topics in college including history, philosophy, and psychology. You do not have to stick with the same career your entire life. You can switch it up and do different things that will enable you to bring a unique perspective to working on a development team. If everyone has the same background on any team, they will not be set up for success because there will be no innovation if everyone is of a similar mindset. Innovation comes from diverse teams with different perspectives who come together to create products and solutions that are applicable to many different types of people and businesses.

Starting a software development career as an intern can provide many benefits. Internships can be hard to find but they offer an opportunity to gain valuable experience and practical skills. Internships provide exposure to real-world projects and challenges, which can help you build critical thinking and problem-solving skills by working with experienced developers. I started my career as an intern for Black & Decker which I found through my college. Some boot camps may also offer job support and you can also find internships by building your professional network.

I am a proponent of beginning a career in tech by doing help desk or tech support work. This will get your foot in the door and allow you to network with tech professionals who can help you find a development position. You will also learn critical thinking skills as you help troubleshoot errors and find solutions for users. If you work with users who are not technically proficient, you can be a teacher and help them learn new skills so that they do not have to call or chat with the help desk in the future. When you work any type of customer service job, you gain valuable skills that are transferrable to any role as you are always seeking solutions for the customer.

Data Science is another field that lends itself to a development career. In this role, you extract meaning from and interpret data. This requires skills in statistics, machine learning, and software development skills. A Data Scientist uses tools to collect, clean, and organize data to find patterns and build models. This role collects data on product or app usage to quantify the health of a product for continuous improvement. You are expected to be detail-oriented to ensure the accuracy and uniformity of the data you extract for analysis.

Another role that lends itself well to a development career is technical training. I have experience in both writing and delivering training and I became a Lead Developer afterward. Writing technical training requires you to learn instructional design where you understand the learner, their goals, and learning objectives for the course. A key skill in instructional design is understanding how people learn and the different teaching methods they can use. Some people like to get training upfront before they begin a task and others prefer learning on the job. There are different formats for learning including written, video, and audio. You can learn from blogs, books, online video training, or audiobooks. Technical Training involves a lot of research, and you must be on your toes to troubleshoot and fix any issues that your students encounter during a class.

Technical writing is another role that can lead to a development career, and it is different from technical training. Technical training focuses on teaching people skills by providing them with instructions on how to build something. Technical writing is documenting the technical aspects of a product. Both roles require a lot of research; however, the format of the output is different. Instead of writing a how-to guide from start to finish, technical documentation is organized by feature or product. The purpose of this is to allow developers to find answers to specific questions, not to build something from scratch.

I have worked with many QA professionals who transitioned into a development career. This is a good career path because you will learn how to thoroughly test your code from the user’s perspective. Many developers (including me) have a tendency to ensure that the technical requirements have been met and only test for those. However, sometimes you will find that the technical requirements may not support the business requirements. You can uncover these issues by looking at the overall user experience to ensure that all requirements are met. QA professionals also provide regression testing which is an excellent skill for developers to have to avoid breaking code that was once working.

Being a Technical Marketer or Content Creator is a great way to learn development skills and move into a development position. You will work closely with software developers and product managers to understand complex technical concepts and communicate technical information to non-technical audiences. This role also provides an opportunity to develop marketing and communication skills, such as creating compelling technical content, developing marketing strategies, and building relationships with customers and clients.

Technical Marketers also work closely with sales, and you could become a Sales Engineer. Sales Engineers use their technical expertise to help customers solve complex problems and make informed purchasing decisions. This requires a deep understanding of software development processes, as well as the ability to communicate technical information to non-technical stakeholders. In this role, you will work closely with product managers, software developers, and customer support teams, which can provide exposure to different aspects of the software development life cycle. I have worked with many Sales Engineers who transitioned into developer roles, and they are often excellent at zeroing in on the problem to provide ideas for improvement to ensure customer satisfaction.

And finally, there is the Developer Advocate or Developer Relations role. This is something you can do either before or after you become a Lead Developer as a lateral move. This role requires a lot of public speaking so you must be comfortable speaking in front of a large group of people. Developer Advocates also create content to promote learning for a specific technology. This often involves filming live videos, screen sharing, webinars, and live streams. They are the face of the development community which gives them influence and they are role models. Developer Advocacy is great for people who enjoy development work, teaching, and connecting directly with an audience to help them learn new skills.

Whatever career path you take is your own choice. Don’t feel like you must stick to one path, we all have room to grow and sometimes that takes our careers in different directions. In my career, I have been a Help Desk Associate, Web Developer, Courseware Developer, Technical Trainer, Lead Developer, Author, Speaker, and Business Owner. I certainly did not take the traditional path even though I have a degree in Computer Science. There are many different roles that involve technical skills that will help you eventually land a job as a Lead Developer. It’s up to you to make it happen.

2.2 Moving Through a Software Development Career

Development positions are highly competitive, and it is hard to get your foot in the door. When you are applying for jobs through public job boards or recruiters, it’s even more difficult to get an interview. My number one tip for any professional in any career is to start networking early. The sooner you do not have to apply for jobs anonymously, the better off you will be. When you have a large professional network, you will find jobs more easily as people can suggest jobs to you and help you get an interview.

As a lead developer, it is important to continuously expand your professional network in order to stay up to date on industry developments, seek out new opportunities, and grow your career. One way to do this is to actively participate in online developer communities and forums, attend industry events and conferences, and join local meetups and networking groups. You can also reach out to other professionals in your field through LinkedIn, email, or other online channels to ask for advice, request introductions, or simply connect and exchange ideas. Building relationships with others in your field can also be a great way to learn new skills, get feedback on your work, and find new collaborators. Even if you find a job through your professional network, you still need to impress the Hiring Manager and show that you are a match for the position by successfully navigating the interview process.

2.2.1 Writing a Resume

Your resume is the main tool that will get you an interview. You should tailor your resume for different types of positions to highlight your experience that matches the job description. I do not generally tailor a resume for each job that I apply for. Instead, I keep copies of resumes for different job types. For example, I have different resumes for Lead Developer and Developer Advocate roles. While both positions have related skill sets, you must highlight the skills that are most applicable to the role.

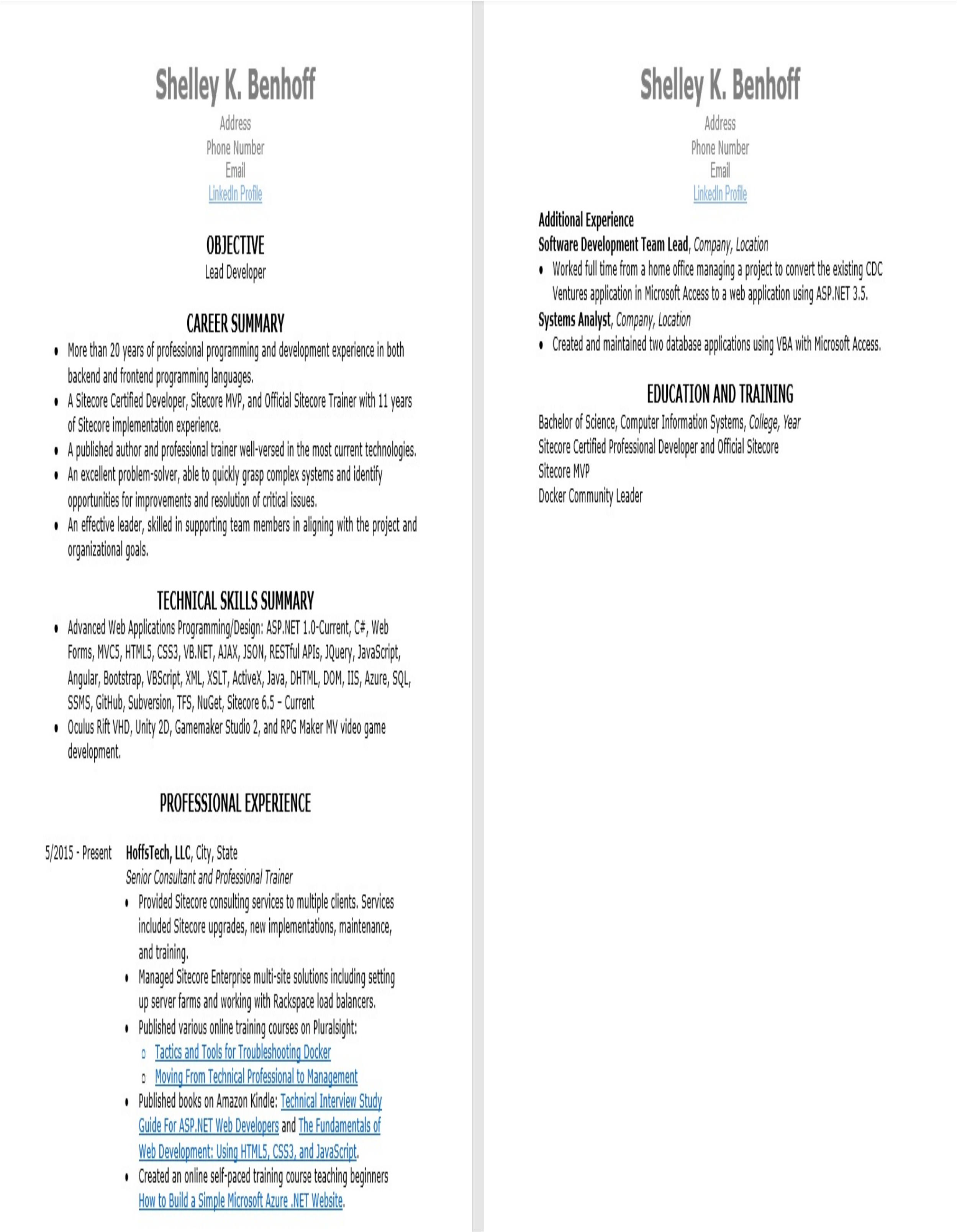
A good resume should include the following:

- Your name and contact information: phone number and email (optionally you can include your address, LinkedIn Profile, and portfolio website URL)
- Career Objective: keep it simple and list the position title you are applying for
- Career Summary: a bulleted list of high-level highlights from your career
- Technical Skills Summary: a bulleted list of programming languages,

methodologies, and tools

- Professional Experience: a table of your employment history over the past 5-7 years including company names, locations, dates, and a bulleted list of your job tasks
- Additional Experience: a shortened list of your job experience 7+ years in the past
- Education & Certifications: list everything you have accomplished, even if it is not related to the job description

Figure 2.3 Example Lead Developer Resume



Shelley K. Benhoff

Address
Phone Number
Email
LinkedIn Profile

OBJECTIVE

Lead Developer

CAREER SUMMARY

- More than 20 years of professional programming and development experience in both backend and frontend programming languages.
- A Sitecore Certified Developer, Sitecore MVP, and Official Sitecore Trainer with 11 years of Sitecore implementation experience.
- A published author and professional trainer well-versed in the most current technologies.
- An excellent problem-solver, able to quickly grasp complex systems and identify opportunities for improvements and resolution of critical issues.
- An effective leader, skilled in supporting team members in aligning with the project and organizational goals.

TECHNICAL SKILLS SUMMARY

- Advanced Web Applications Programming/Design: ASP.NET 1.0-Current, C#, Web Forms, MVC5, HTML5, CSS3, VB.NET, AJAX, JSON, RESTful APIs, JQuery, JavaScript, Angular, Bootstrap, VBScript, XML, XSLT, ActiveX, Java, DHTML, DOM, IIS, Azure, SQL, SSMS, GitHub, Subversion, TFS, NuGet, Sitecore 6.5 – Current
- Oculus Rift VHD, Unity 2D, Gamemaker Studio 2, and RPG Maker MV video game development.

PROFESSIONAL EXPERIENCE

5/2015 - Present **HoffsTech, LLC**, City, State

Senior Consultant and Professional Trainer

- Provided Sitecore consulting services to multiple clients. Services included Sitecore upgrades, new implementations, maintenance, and training.
- Managed Sitecore Enterprise multi-site solutions including setting up server farms and working with Rackspace load balancers.
- Published various online training courses on Pluralsight:
  - Tactics and Tools for Troubleshooting Docker
  - Moving From Technical Professional to Management
- Published books on Amazon Kindle: Technical Interview Study Guide For ASP.NET Web Developers and The Fundamentals of Web Development: Using HTML5, CSS3, and JavaScript.
- Created an online self-paced training course teaching beginners How to Build a Simple Microsoft Azure .NET Website.

Shelley K. Benhoff

Address
Phone Number
Email
LinkedIn Profile

Additional Experience

Software Development Team Lead, *Company, Location*

- Worked full time from a home office managing a project to convert the existing CDC Ventures application in Microsoft Access to a web application using ASP.NET 3.5.

Systems Analyst, *Company, Location*

- Created and maintained two database applications using VBA with Microsoft Access.

EDUCATION AND TRAINING

Bachelor of Science, Computer Information Systems, *College, Year*
Sitecore Certified Professional Developer and Official Sitecore
Sitecore MVP
Docker Community Leader

You should keep your resume relatively short which is why most people suggest only listing the past seven years of experience under the professional experience heading. An additional experience section is a place for you to summarize other positions you have held throughout your career that are applicable to the career objective. Hiring Managers see a lot of resumes and this format has landed me a lot of interviews over the years, even when I was applying for Junior Developer positions.

EXAMPLES OF COMMON RESUME ERRORS

The following list contains examples of common errors that you should avoid to ensure that your resume ends up in the “yes” pile:

- **Errors in spelling, grammar, and punctuation**
- **Missing contact information**
- **No technical skills section**
- **Not well organized**
- **Not tailoring a resume for the job requirements**

2.2.2 Creating a Cover Letter

The development industry is extremely competitive, and it can be difficult to get a Recruiter or Hiring Manager to review your resume. Recruiters and Hiring Managers are inundated with job applications when they post a new job. They hardly ever read every word of a resume and having a cover letter can help get you an interview by summarizing your qualifications and piquing their interest in a few short sentences.

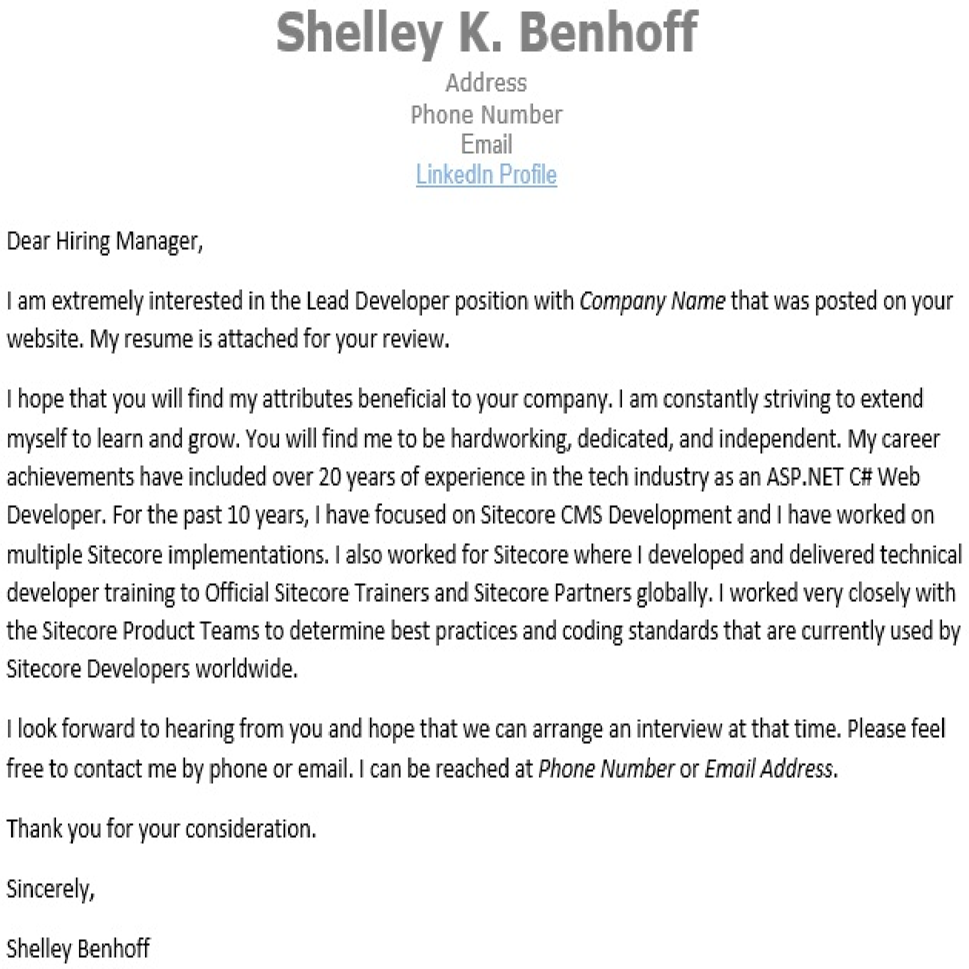
If you are a recent college graduate, congratulations, you are now competing against hundreds of thousands of Developers just like you. If you did not go to college, congratulations, you are now competing against hundreds of thousands of Developers just like you. Either way, you are going to have a lot of competition. So, how do you get potential employers to read your resume? As someone who has been both the interviewee and interviewer, I know it all starts with the cover letter. A well-written cover letter goes a long way to getting an employer to read your resume.

You should have the following items in a cover letter:

- The position title and company
- Where you found the job posting
- Highlights from your resume that match the job description
- Your availability for both an interview and a potential start date
- Your contact information

You should always keep a general cover letter template for yourself and then tailor it to each position you are applying for. If you send a generic cover letter to every employer, then it will look like you are spamming employers from job boards. To make the process easier, I suggest creating a Microsoft Excel spreadsheet listing each position, resume highlights, etc., and then using a Microsoft Word mail merge to easily automate your cover letter. Keeping a spreadsheet of your job applications will also help you to avoid applying for the same job multiple times which will land your resume in the “no” pile.

Figure 2.4 Example Lead Developer Cover Letter



Shelley K. Benhoff

Address
Phone Number
Email
LinkedIn Profile

Dear Hiring Manager,

I am extremely interested in the Lead Developer position with *Company Name* that was posted on your website. My resume is attached for your review.

I hope that you will find my attributes beneficial to your company. I am constantly striving to extend myself to learn and grow. You will find me to be hardworking, dedicated, and independent. My career achievements have included over 20 years of experience in the tech industry as an ASP.NET C# Web Developer. For the past 10 years, I have focused on Sitecore CMS Development and I have worked on multiple Sitecore implementations. I also worked for Sitecore where I developed and delivered technical developer training to Official Sitecore Trainers and Sitecore Partners globally. I worked very closely with the Sitecore Product Teams to determine best practices and coding standards that are currently used by Sitecore Developers worldwide.

I look forward to hearing from you and hope that we can arrange an interview at that time. Please feel free to contact me by phone or email. I can be reached at *Phone Number* or *Email Address*.

Thank you for your consideration.

Sincerely,

Shelley Benhoff

A well-written cover letter can be a great tool to use to get an interview and if you are applying for a job on a company’s website, they often require one. Understanding how to write a cover letter will set you up for success and move you through the interview process and on to the next step.

2.2.3 Achieving Success in Technical Interviews

Technical interviews are an important part of the interview process for developers, and they are crucial for the Lead Developer role. You are expected to be a technical expert, and this often requires a coding test followed by an interview where you discuss the coding test as well as best practices in the industry. You must show your skill level and stay calm during this phase of the interview process.

When you are given a coding test, make sure you understand what they are asking you to do. Ask for clarification if you don’t understand something about the coding test. You must go into it with a full understanding of the expected result. It is human nature to want to proceed without asking questions so that you appear self-sufficient but that is detrimental to your interview success. As a hiring manager myself, I like when people speak up because I am not looking to hire “yes” people. I want to hire people who will bring their opinions to the table to help the team learn and grow. The following list includes resources to help you practice skills for taking a coding test.

- LeetCode
- HackerRank
- Project Euler
- Interview Cake
- Cracking The Coding Interview

As you go through the coding test, make sure that you are detail-oriented, and that you follow each specific instruction to the letter. Developers must be detail-oriented to ensure that all requirements are completed so that clients are happy with the result. Hiring Managers want to see a high level of consistency and well-thought-out code when they’re hiring a Lead Developer. This will show that you not only possess technical expertise, but you are also a person who can look at every aspect of a task to drive high-quality results.

After you pass the coding test, you will move on to the interview where you will most likely be asked to answer questions about your approach to the coding test. A good thing to do is to walk them through your logic in solving the coding test and give well-rounded answers. Don’t just tell them how you did something, tell them why. What best practices have you learned and where did you learn them? Try not to second-guess yourself. Be confident in your approach! If you got an interview, then the Hiring Manager liked your approach, otherwise, you would not have moved on in the interview process.

You should also discuss other approaches you could have used to complete the coding test. This will show that you have an open mind and are able to see opportunities to go a different route. Companies tend to have set coding standards within the organization, and they may vary from company to company in terms of architecture and style. Being able to adapt to different coding standards is a good skill for any Lead Developer.

During the technical interview, if you are asked a question and you don't know the answer, talk about how you would go about finding the information that you need to get the answer. No one should be expected to know everything off the top of their head. The best interviews I have been a part of allowed people to use a laptop to look up answers if they were unsure. As Developers, most of our day is spent doing research anyway and I want to see that skill in practice when I'm hiring a Lead Developer.

Technical interviews can be very stressful, but you must remain calm. If you have spent time preparing for the interview, this will greatly increase your chances of success. You can outline your answers ahead of time based on questions you have been asked in previous technical interviews. You should always be prepared to walk through your coding test in detail to explain the logic in your approach. Going into a technical interview being prepared will help you remain calm, cool, and collected so that you are successful.

2.2.4 Interviewing For Development Leadership Positions

If you find that you have an interest in the business side of things, you can begin to think about the next steps in your career progression. Leadership interviews are different from technical interviews as they focus on things like business goals, budgeting, and staffing requirements. You may be asked some technical questions, but they will be more abstract at a higher level instead of walking through your approach to individual lines of code in a coding test. You should be prepared to talk about how you will lead teams and your approach to leadership in general. A good way to cultivate your leadership style is to work closely with Project Managers and Team Managers across the organization. Observe how they handle leading meetings, organizing projects, and maintaining budgets. Work with them to create solutions that will enable them to automate processes which will increase productivity and decrease costs. Showing that you are capable of decreasing costs is a key skill that Hiring Managers look for in a Development Manager.

There are generally multiple interviews included in an interview process for leadership positions. You may have to meet with executives and people you would not normally work with as a Lead Developer. When you meet with executives, I suggest that you prepare a slide deck. Part of being a manager and leader is the ability to organize engaging presentations. Even if you are not asked to prepare a presentation, prepare one anyway! This will show that you will go above and beyond to achieve success and that you are comfortable with public speaking. Your slides should not only walk through your resume but also discuss current industry trends and how you would ensure that your projects are following best practices. Also, list any applicable successes you achieved as a Lead Developer and focus on the success of your team, not yourself. Leaders support teams and must be team players, you should not be focused on yourself as this can show a tendency to take credit for the work of others. Hiring Managers are looking for leaders who give credit where credit is due.

If you have spent your entire career as a Developer, you must be prepared to shift gears when you apply for a leadership position. Leadership requires a different skill set, so it is important for you to keep a record of your success as a Lead Developer. This will make it easy to keep your resume updated in the future. Make sure that you focus on the skills required to manage development teams instead of being the technical expert on the team. You need to be forward-thinking and use data to predict future staffing needs and mitigate costs. These are skills that you will learn as a Lead Developer when you work closely with Project Managers on task estimates and statuses with the goal of reducing costs for the organization. This will help set you up for success as you move forward in your career and begin applying for leadership positions.

If you’re lucky, a position in Leadership may open at your current job that you can apply for. But before a position becomes open, you should begin to step up in your current job and show that you have leadership skills. When you are a Lead Developer, you will show your leadership skills by supporting the success of your team. You can do this by taking responsibility for mistakes that were made as the key decision-making is up to you. Mistakes happen and no one is infallible. Being a responsible person will show that you are leadership material.

Being proactive instead of reactive is another great skill to have as a leader. For example, you must schedule maintenance periods to avoid system failures. Your team should be empowered to think ahead and predict future issues, especially when large changes are made such as upgrading to a new version of the programming language you are using. Taking a proactive approach will reduce emergencies as well as costs due to time lost fixing critical errors that could have been prevented. If you find that the company is experiencing consistent emergencies across multiple projects, you should voice your opinion and offer solutions to avoid future emergencies. You can schedule nightly maintenance tasks that are automated and send out a status report once the maintenance is complete. Over time, keep track of the reduction in emergencies and communicate your team's success to the leadership team.

2.3 The Job Market for Lead Developers

Developers are in high demand and there are plenty of jobs available, however, there is also a lot of competition. Since you're reading this book, you are already ahead of the competition by taking an interest in your career plan! It also helps to know what your peers are doing to get jobs so that you can prepare for interviews accordingly. Being competitive includes observing the actions of others to inspire your own approach to maintaining your development career.

2.3.1 Assessing the Competition

As a Lead Developer, you should assess your competition in the job market to better position yourself to attract top talent and stay competitive in the industry. The following list includes some ways that you can assess your competition in the job market:

- Research job postings: You can review the job postings for Lead Developers and note the required qualifications, job responsibilities, and compensation packages. This will give you a better understanding of the skills and experience that are in demand in the current job market.
- Analyze job descriptions: You should pay attention to the specific technologies and programming languages that companies are looking for

in their job descriptions. This will help you identify the most important technical skills to develop and highlight in your own job postings.

- Track market trends: Joining online communities and forums where developers discuss their experiences and job opportunities will give you a better understanding of the current market trends and the demand for certain skills. Look for emerging technologies and programming languages that may become more popular soon.
- Monitor your competitors' online presence: You can look at your competitors' websites and social media accounts. Look for information about their team structure, the types of projects they're working on, and any recent hires or promotions. This can give insight into their hiring strategies and help you stay up to date on industry news.
- Attend industry events: Attending industry events and conferences gives you the opportunity to network with other developers and stay up to date on the latest technologies.

Another way to assess your competition is by speaking to mentors. Everyone needs a group of mentors to help them navigate their careers. Having mentors who have hired people in the past or who are current hiring managers helps a great deal as they can help you assess your competition. They can guide you, suggest skills to learn, and tell you what areas you excel in. It’s even better if you have multiple mentors who can do this for you because mentors should only guide you to career solutions that are right for you. We are all different and we should take inspiration from multiple people to forge our own path. Asking your mentors what drives them to read a resume, put it in the yes pile, and hire Developers will enable you to be competitive during the interview process and get the job that you want.

2.3.2 Standing Out Above the Crowd

How do you stand out to get job interviews when hiring managers receive piles of resumes for every job that they post? A great way to pique the hiring managers’ interest is to customize your resume for each job application. You should list your skills that match the job requirements listed in the job description. This will demonstrate your ability to apply your skills in real-world situations and draw attention to your relevant skills. You should showcase your relevant skills in an engaging way by using action verbs to describe your accomplishments and quantify any applicable results. For example, I like to use verbs like achieved, built, optimized, and automated. Hiring managers prefer to see a concise list of what you have achieved in your career so you should focus on actions.

Another good way to stand out from your competitors is to contribute to open-source projects. When you join tech communities, there are often channels where people post links to the community projects they’re working on. You can work with a team of Developers in a GitHub repository where you can report bugs, write code, and submit pull requests. Most of the time, you do not have to request to join community projects as the repositories are public so anyone can join at any time. However, it is good to get to know other people who are working on the project and discuss the most important features that need to be implemented and errors that need to be addressed. This will show Hiring Managers that you can work with a team and that you understand code repositories and how they relate to the Software Development Lifecycle (SDLC.) Being active in programming communities is a great sign that you are a team player and provide value by giving back to the community. Contributions to open-source projects are not often listed on the resumes that I have seen, so having this experience will help you stand out and land you an interview.

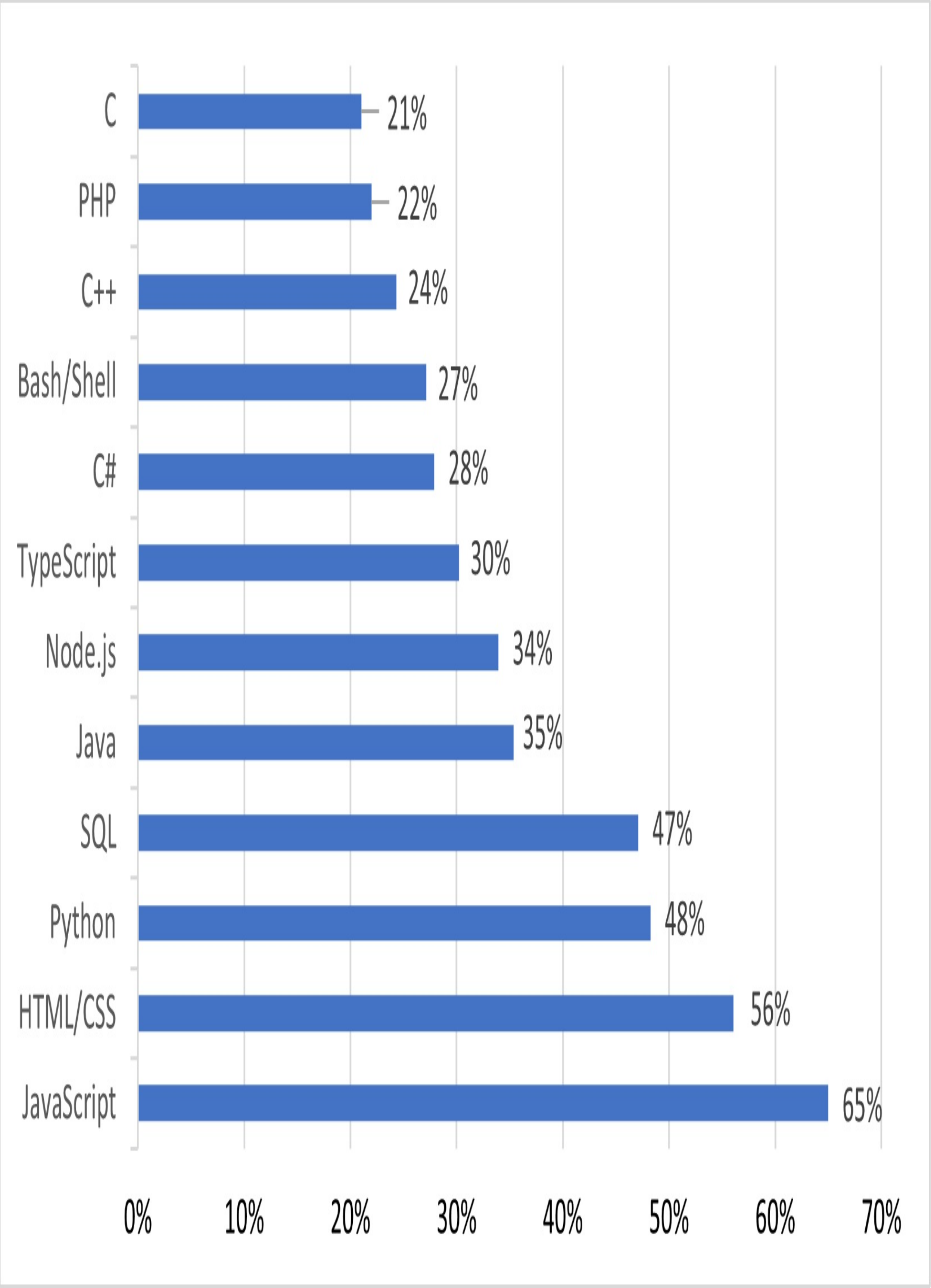
I suggest that any Developer should start a blog. This is another way to show that you give back to the community by posting helpful tips for your peers. The best blogs are the ones that explain an issue that the author encountered and how they fixed it. If you are struggling with a specific error, chances are that someone else is too and your blog post could be the resource that they need to resolve the error quickly. You can also blog about your work contributing to open-source projects and get others interested in joining the team. Plus, you can use your blog to help you find an answer that you need if you can’t think of it off the top of your head. Most Developers that I know often use their own blogs to help them if they’re struggling and I do this too! You can find my blog at hoffstech.com. Your blog can act as a notebook for you if you need to revisit an error or process that you previously documented.

As you contribute to open-source projects and write blog posts, you can build up your GitHub profile. Having an active GitHub account is a great way to show everything that you have learned in the past and are currently working on. When you write blog posts that contain walkthroughs of code, you should create a GitHub repo even if it's not a full project. Did you know that Hiring Managers can search GitHub to find interview candidates? GitHub settings allow you to add a job profile and check a box if you're available for hire. This is a little-known fact and it's a great way for you to get noticed by Hiring Managers and receive invitations to apply for jobs that you would have not known about otherwise. GitHub also has a feature for you to create your portfolio website and add it to your profile. You can even host your portfolio website on GitHub and include the link in your resume. Many Developers miss these features as they are not really promoted or intuitive.

2.3.3 Reviewing In-Demand Technical Skills

It is often confusing for Developers to figure out what programming skills, platforms, and technologies they should focus on. The number of skills that you can learn is overwhelming and you cannot be an expert in everything. It is good to specialize in related skills that complement each other such as JavaScript and frameworks and libraries including Angular and React. Keep in mind that these trends change frequently and just because a skill becomes less in demand does not mean that you should switch to another skill. You want to remain somewhat consistent throughout your career and build your skills according to your interests as well as what jobs are available.

Figure 2.5 Top In-Demand Programming Skills For 2022 According to Stack Overflow (https://survey.stackoverflow.co/2022/#programming-scripting-and-markup-languages)

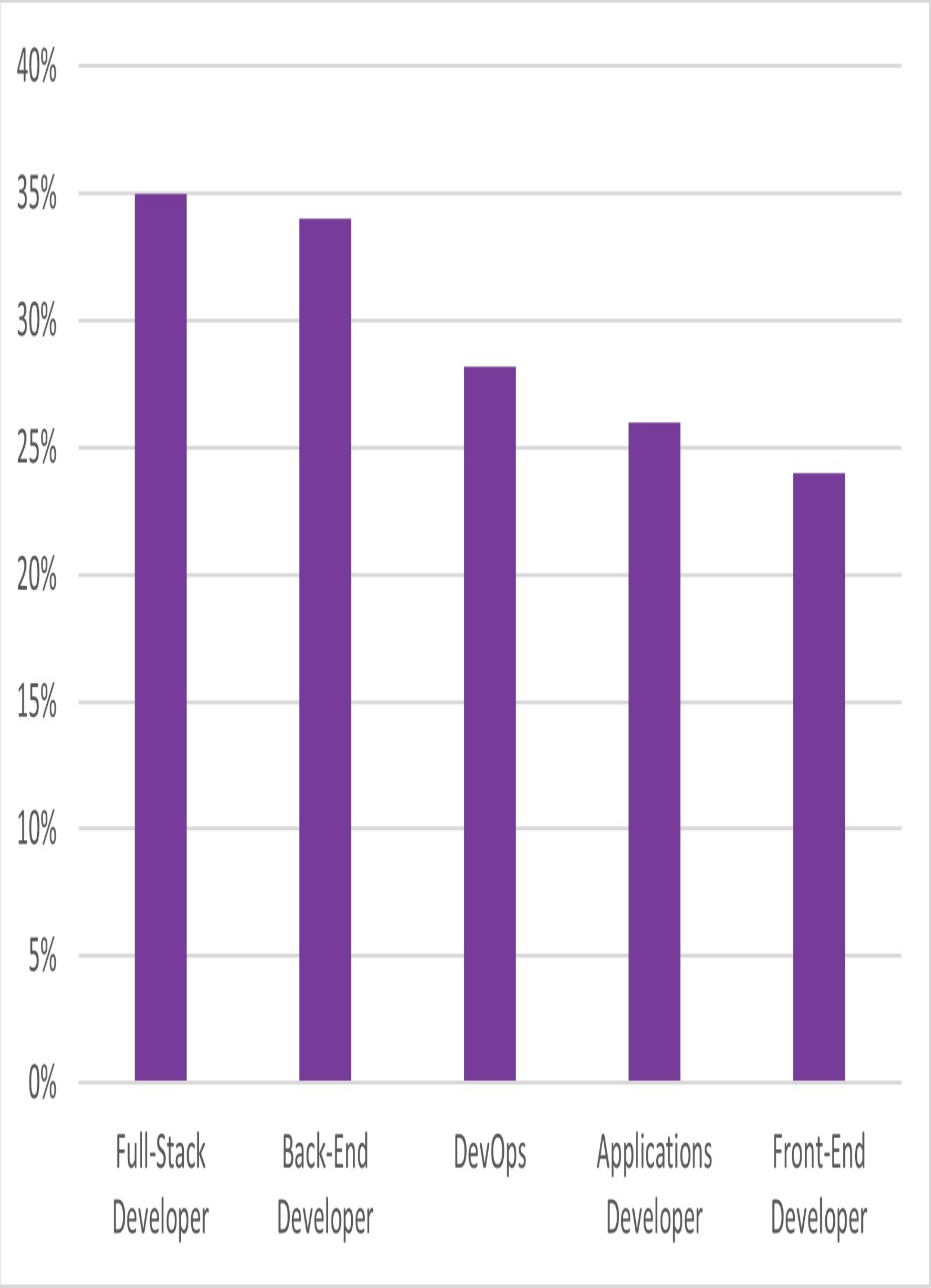

C
21%
PHP
22%
C++
24%
Bash/Shell
27%
C#
28%
TypeScript
30%
Node.js
34%
Java
35%
SQL
47%
Python
48%
HTML/CSS
56%
JavaScript
65%
0%
10%
20%
30%
40%
50%
60%
70%

The landscape in the tech industry for Developers is constantly changing and there are many different types of jobs and roles that you can specialize in. Full-Stack Developers are the most sought-after as they have well-rounded expertise. As a Lead Developer, you will be required to manage all aspects of the development process and you will work with different types of developers. If you are creating web applications, you will work with both Front-End and Back-End Developers, so you need to understand their day-to-day tasks. A Full-Stack Developer may not be an expert in everything, but that role will set you up for success as a Lead Developer if you are responsible for managing full teams.

Lead Developers must understand the CI/CD process and it is important to learn DevOps skills to manage releases. You should be able to configure all environments including a local development environment, QA, Staging, and Production. This requires expertise in the full SDLC and organization skills to work on multiple releases at the same time. Understanding cloud architecture is necessary to manage deployments so that there is minimal downtime for your users. This includes skills such as Amazon Web Services (AWS) and Microsoft Azure to integrate your development workflow and configure pipelines for CI/CD. Learning Docker and Kubernetes will help you streamline your deployment processes and automate a lot of the heavy lifting. The SDLC process varies between different companies and industries depending on the approach and methodology. Lead Developers should understand all aspects of the SDLC.

Application Developers are in high demand as they focus on building applications for multiple operating systems including Linux, Mac, Windows, iOS, and Android. You are required to have in-depth knowledge of managing code for each OS. You can specialize in building apps specifically for mobile devices and that is also a very in-demand role for Developers, however, most applications exist both on the web and as mobile apps so many companies look for Lead Developers who can manage both.

Figure 2.6 Top 5 In-Demand Development Jobs For 2022 According to Codename (https://www.codingame.com/work/blog/hr-news-trends/top-10-in-demand-it-jobs-2022/)


40%
35%
30%
25%
20%
15%
10%
5%
0%
Full-Stack Developer
Back-End Developer
DevOps
Applications Developer
Front-End Developer

Having a well-rounded skill set will help you achieve success as a Lead Developer as you are expected to manage the entire SDLC. You will work with people from different roles and when they need help, you should be able to point them in the right direction. Keeping your skills current is a great way to stay ahead of the curve and provide the organization with a proactive approach to future upgrades. Understanding current trends will help your team achieve success as they continue to learn and grow in their careers. We will discuss how to keep your technical skills current in a later chapter.

2.4 Case Study

Dan Wahlin founded Wahlin Consulting, which provides consulting and training services on JavaScript, Angular, Node.js, C#, ASP.NET MVC, Web API, Docker, and Kubernetes. He is a Google GDE (and former Microsoft MVP and Regional Director), Docker Captain, and speaks at conferences and user groups around the world. Dan is active on Twitter (@DanWahlin), blogs at https://blog.codewithdan.com, and adds a lot of code to his GitHub repositories at https://github.com/danwahlin. In the following case study, Dan provides advice and solutions for Developers who are setting and moving through a career plan and interviewing for jobs in a competitive market.

2.4.1 What Advice Do You Have for Developers Who Are Setting a Career Plan?

It is important to think through and visualize what achievements you want to reach in your career. As Stephen R. Covey said, “Begin with the end in mind.” You should read his book, “The 7 Habits of Highly Effective People”, if you have not already, it is an excellent career resource. He’s saying that you should make time to visualize what it would feel like to reach your goals. For example, athletes visualize themselves performing well in a game, salespeople visualize themselves giving a successful pitch and winning the sale, and visualization applies to many other scenarios including your career success. Visualize yourself in the role you want and think through what it would take to get to that level. Your inner voice may kick in and say, “You can’t do that, who are you kidding?". That’s normal and it happens to everyone no matter who they are. Keep at it and develop a belief that you can achieve what you’re visualizing. Planning and documenting the specific steps you need to take to achieve your career goals is a great start to begin your career plan.

This next one is extremely important! Build your plan into your daily routine. Want to manage a team someday? Research what highly successful managers do and start practicing what you learn in meetings, in interactions with colleagues and customers, and in other situations. You don’t have to be a manager to start applying the skills that a manager needs. Want to be a CEO? Learn what successful CEOs do, practice the skills you’ll need to get that type of job, learn about what path different CEOs took to get there, and apply what you learn to your current job and overall life. Don’t wait for the “perfect time”. The perfect time is now.

Be consistent and persistent. Have a plan to actively work toward your goals. Roadblocks will certainly be thrown at you along the way. Your muscles get stronger by applying consistent stress to them. You get stronger in life by viewing roadblocks or walls as opportunities that help you learn, improve, and move to the next level. You can stay at the bottom of the wall, or you can put in the effort to climb it.

Let me share a quick story about myself and my career journey. I can honestly say that way back in my early 20s I visualized myself doing things in my career such as:

- Managing a team.
- Writing a book.
- Releasing a training course.
- Giving corporate training around the world.
- Speaking at conferences.
- Publishing articles for magazines.
- Running a company.
- As well as other things.

While I experienced plenty of fear, doubt, and anxiety along the way, I knew I could achieve these goals and believed in myself. It took a lot of planning, consistent practice, persistence, and hard work, but I was able to reach all my key goals. I want to emphasize that it wasn’t because I was smarter than others or better than them. On the contrary, I view myself as pretty “normal”. But I was willing to visualize my goals and had the belief that I could achieve them. I’ll admit that I wasn’t as good at documenting the steps, but in hindsight, I realize how important the planning process is. Having said that, no amount of visualization or planning will help you achieve your career goals without consistent and active effort. Plan to roll up your sleeves, dig in and get dirty. Action speaks louder than words, so you must be willing to consistently work smart and work hard.

2.4.2 How Can Developers Stay on Top of The Competition in Today's Job Market?

Become a lifelong learner. I’ve had to learn many different concepts and technologies over my career. The secret to success in today’s job market? Be willing to step outside your comfort zone, learn something new, practice it, and increase your skills by actively using it.

Be willing to admit that you don’t know it all if pressed. “Imposter syndrome” is completely normal and everyone feels it from time to time. Push through it and be open to asking for help when you need it. Nobody knows it all. If you feel like someone does know it all, they’re just a good actor. Don’t act like a “know-it-all” in the interview.

It’s OK to be wrong and it’s OK to fail. Embrace that and take advantage of it in interviews. View failures as learning opportunities that you can leverage in future scenarios to help you stand above the crowd. Apply failures and lessons learned to scenarios you work with every day. Share how you failed, what you learned, and the approach you now take with others (in interviews, with your team, with your career community, etc.).

Actively contribute to your community. If you’re a developer, contribute your own projects to open source, submit pull requests as you find issues in projects you use, write about how you personally learn, lessons learned, etc. on your blog, in social media (if you use that), and other areas. If you do not have enough time to contribute to your community or work on personal projects, try asking your current employer if you can write blog articles for them detailing technical issues you have faced and how you fixed them.

Be willing to put yourself out there. If you enjoy sharing your knowledge, you can create videos, submit to speak at conferences, and offer to give “lunch and learns” at work. A little-known fact about me, when I began submitting to speak at conferences, I received nothing but rejections. It literally took a few years before I got my first “yes” and that changed everything. Although I’d argue it’s much easier to speak at conferences nowadays (there are a lot more of them), don’t take a rejection personally. Reach back out to the conference if possible and ask what you could do differently. Let them know that you’d really like to speak and that you’re willing to volunteer the first time around. Get creative!

2.4.3 How Can Developers Stand Out During the Interview Process?

Smile and be friendly. Is that stupid to mention? I don’t think so. It’s amazing how far a smile and a friendly and cooperative attitude will get you in life. I’m not saying that you should be fake, but I am saying that it’s easy to see when someone is truly excited to be there for the interview. If you feel the interview isn’t going well, do your best to continue having a positive attitude and demeanor. You may be doing way better than previous interviewees without knowing it. Some interviews are designed to test people’s resilience and ability to deal with stress so keep that in mind.

Be prepared and practice, practice, practice. Learn as much as you can about the interview process, have friends interview you, and give feedback. Practice how you’ll respond when you don’t know something. Practice what questions you’ll ask the interviewers if that opportunity presents itself. Learn how to deal with stress appropriately. The more you practice the more you can be yourself, which will instantly put you above the people who are nervous and struggling during the interview.

Another little-known fact about me is that I didn’t have the years of experience required for my first technology job interview. I got the job though, so what was the secret? I was extremely interested in the job and my enthusiasm showed through to the point that the interviewers were excited to give me a chance. They interviewed many people but remembered me the most I was told.

I had a project ready to show that related to the job I was interviewing for. Although I didn’t have an extensive background using the skills they wanted, I was able to demonstrate that I had all the skills needed by showing them a “real” project I built. The fact that I was prepared and had something ready in advance impressed them.

Be willing to say “I don’t know” when you’re not sure about something and practice how you’ll respond (as mentioned earlier). But - and this is very important - follow up any “I don’t know” statement with what you would do to figure it out and walk the interviewer through your problem-solving skills if they’ll let you.

Have project work, articles/blog posts, videos you’ve created, etc. that are directly related to the job ready to go for the interviewers. Be proactive and prepared rather than showing up with nothing aside from the ability to answer questions. If you really want the job and don’t currently have a project to show, put in the time to create the project. Some interviews require you to build a project and then turn it in later. I’d only do that if you’re very interested in the job (I’m not a big fan of that interviewing approach by the way, they should instead give preference to the proactive interviewees that have something ready to show in advance). If you do have that type of interview though and are very interested in the job, use that project as an opportunity to show off your skills and stand out above the crowd.

2.5 Summary

- The most common career trajectory for Developers begins with the Junior Developer role, then progresses to Senior Developer, and then Lead Developer.
- There are many positions that relate to development and will help you achieve success as a Developer such as Help Desk, Data Science, QA, Technical Training, Technical Writing, and Developer Advocacy.
- You must show that you have leadership and business skills to get promoted into management.

- Your resume should list your career objective, career summary, and technical skills summary at the top to highlight your career achievements in a condensed format to make it easier for Hiring Managers to read.
- Development positions have a high level of competition, and you can stand out by possessing certifications, working on open-source projects, starting a blog, and creating a GitHub portfolio.
- Achieving success in technical interviews can be accomplished by being detail-oriented and addressing every task required in the coding test and being able to walk through your approach to the coding test during the interview process.
- Lead Developers should focus on staying current with in-demand skills in programming and DevOps.

3 Learning Lead Developer Skills

This chapter covers

- Keeping your technical skills current and listing the resources you can use to study development skills
- Reviewing ways to study and practice soft skills needed for leadership
- Prioritizing learning and fitting it into your busy schedule
- Listing the most popular leadership styles and how to apply them
- Deciding what leadership style is right for you and your personality type
- Improving your presentation skills by creating professional slides and engaging with your audience

There are many key skills that you will need to learn to be a successful Lead Developer. Learning everything you must know takes a lot of time and it is important to know what skills to learn and prioritize them accordingly. Lead Developers are expected to strike a good balance between technical and soft skills and that is a very hard thing to do.

As Developers, we must keep our technical skills current if we want to be competitive in the job market. This is even more true for Lead Developers as you are expected to lead the technical aspects of a project. You must stay up to date with the latest technologies and trends. We are often using many different technologies in a single project, and it can be difficult to keep ourselves up to date. This is especially true when brand-new programming languages are developed. Throughout my career, I have probably worked with over 15 different languages and frameworks. I am constantly reading technical blogs, white papers, and announcements from technology companies to stay current with industry best practices. It helps me to study architectural patterns for every programming language before I begin working with them. That way, I have a reference for when I start a new project using that language.

Soft skills are often difficult to learn because you cannot always put yourself in specific situations so that you can practice soft skills such as conflict resolution. Plus, there’s no one-size-fits-all approach to each situation that will occur on your watch. You must learn how to navigate situations and communicate effectively to support the success of the team.

It may take time for you to find your leadership style as you learn and apply soft skills that are new to you. Leadership can be scary, especially when you are responsible for key decisions that impact the budget of a multi-million-dollar project. You must learn to embrace failure and not fear it. You will learn more from failure than you do when you are successful. The real failure is never trying in the first place. If you fear failure, you may opt for no decision vs. the right decision. Making no decision is a decision in and of itself and it can be very detrimental to the morale of your team. Some companies also defer decisions until the last minute as a best practice to allow time to consider all possible approaches and outcomes.

Public speaking is a skill that many people avoid at all costs. To be a good Lead Developer, you should continually work on improving your presentation skills. Learning best practices in slide design, cadence, and tone can make all the difference in keeping your audience engaged throughout your presentation. However, unlike most soft skills, you can practice presentations beforehand to ensure success when you’re presenting.

Being a lifelong learner is the key to being successful in your entire career, not just as a Lead Developer. I don’t believe that anyone ever reaches an ultimate understanding of 100% of any topic. There are always new things to learn, especially when you observe things from other people’s points of view.

3.1 Prioritizing Learning New Skills

Learning technical skills isn’t new to you as a Developer, you are constantly having to learn new skills to complete your day-to-day tasks. It’s often hard to find the time to set aside for studying but it must be done. When your skills are outdated, you may miss key upgrades that must be executed to avoid a security issue or bug. Lead Developers are not always given a heads-up when new technologies need to be implemented. Often, it is the Lead Developers who suggest the adoption of new technologies for the entire organization.

Learning and applying soft skills is equally important to keep your tech skills up to date. Leadership skills change over time as culture shifts and new processes are adopted. In recent years, emotional intelligence has become a key skill that all leaders must have but it was not that way back in the industrial era when they used the command-and-control style of leadership. Now, we are in the participation era and people want meaning and purpose in their work. Lead Developers must keep up with these changes so that they're leading using current best practices.

3.1.1 Learning Current Technical Skills

There are many ways that you can keep your technical skills current. When you do not prioritize learning, you will waste a lot of time muddling through difficult tasks. If you encounter an error that you do not recognize, the first thing you would do is search for that error. You may find quick answers but then you are faced with new errors, and you are nowhere close to figuring out the solution. At this point, you should look at the documentation for the technology that you're working with or find blogs that explain the error and the solution step-by-step.

An effective way to keep your technical skills current is to subscribe to get email notifications from any technology that you are currently using or may use in the future. There have been major shifts in the industry and companies often change their tooling and deployment processes to keep up with current best practices. Companies send out announcements and helpful blog posts when they have released an update to their technology. You should read the release notes to learn what has changed and think about how these changes will impact the projects your team is working on.

Tech communities also provide announcements for upcoming releases, known issues, and hotfixes. If you join tech communities that are run by a tech company, they usually also provide support so you can ask employees questions directly. Plus, joining tech communities is a great way to expand your professional network and work on community projects. This community interaction can also lead to awards as you give back to the community and help others learn new skills. You should never interact with a community to earn a reward; it should simply be the byproduct of your efforts. The point of joining a community is to provide as much support as you get.

POPULAR TECH COMMUNITIES

The following list includes popular tech communities where people work together to learn new skills and collaborate on projects:

- 100 Days of Code
- 100 Devs
- Hashnode
- Hacker News
- Women Who Code
- Black Tech Twitter
- Dev.to
- GitHub Community
- GitLab Community

I also like to follow tech influencers and developer advocates on social media because they often provide product updates as well as tips and tricks. To find qualified tech influencers, start by searching for keywords related to your industry on social media platforms like Twitter, LinkedIn, and YouTube. Attend industry events and conferences to meet tech influencers in person, and search for online communities and forums related to your industry to see who is actively contributing valuable insights and information. Focus on individuals who have a track record of producing high-quality content and engaging with their audience and be wary of those who seem more focused on self-promotion than on providing valuable information and insights. You should interact with them and start conversations with their followers who comment on their posts. This is a different type of community than joining developer Slack or Discord communities and your experience will vary across different platforms. Social media is full of opinionated people and Developers have curious minds and should not shy away from a public academic debate.

Keeping your technical skills current will enable you as a Lead Developer to provide ideas for continuous improvement of the projects you are working on. You can also pass your knowledge along to the development team and get their opinions on upgrading your current systems or adopting new technology. Making sure that your projects are using up-to-date technology will keep your systems running smoothly and avoid security issues. You will support the growth of your team, and this will result in successful projects.

3.1.2 Reviewing Necessary Soft Skills

While everyone who works in a professional setting must have some soft skills, Lead Developers are required to possess leadership soft skills to be successful. To be a great leader takes time and learning the necessary soft skills upfront will help ease you into a leadership position. When you are a Lead Developer, the development team looks to you for guidance not only in their daily tasks but also in their overall careers.

You must be prepared to communicate effectively by understanding the needs of your team members. Looking inward will raise your self-awareness and this will make you emotionally intelligent. You will be able to manage your own emotions in a positive way which will help you communicate effectively and resolve conflict. It is important to be able to put yourself in someone else’s shoes and see their point of view. This helps to gain the trust of your team as you are open to thinking about things from different perspectives. Being an empathetic leader also helps to manage conflict and resolve it effectively by considering everyone’s opinions. As a Lead Developer, you must keep an open mind and be able to have difficult conversations. It helps to be prepared and to understand what soft skills to focus on, why they are important, and how to learn them.

Table 3.1 Soft Skills for Lead Developers to Learn

Soft Skill	Why It’s Important	How To Learn
Communication	Lead Developers are expected to be the go-between for development and non-development teams.	· The Art of Communicating by Thich Nhat Hanh · Crucial Conversations: Tools for Talking When Stakes Are High by Kerry Patterson, Joseph Grenny, Ron McMillan, and Al Switzler
Mentoring	Developers need guidance to help them learn current skills and support their career success.	· The Mentor's Guide: Facilitating Effective Learning Relationships by Lois J. Zachary · Becoming an Effective Mentoring Leader: Proven Strategies for Building Excellence in Your Organization by William J. Rothwell and Peter Chee
Emotional Intelligence	Being self-aware and managing your behavior in positive ways will help you communicate effectively and overcome challenges.	· The Emotionally Intelligent Manager: How to Develop and Use the Four Key Emotional Skills of Leadership by David R. Caruso and Peter Salovey · The Ride of a Lifetime by Bob Iger, CEO of Disney
Empathy	Seeing things from others’ points of view will help you form	· Applied Empathy: The New Language of Leadership by Michael Ventura · Empathy Works: The Key to

	quality working relationships.	Competitive Advantage in the New Era of Work by A. Sophie Wade
Conflict Resolution	Effectively resolving conflict will support a positive working environment to help you attract and retain top talent.	· Difficult Conversations: How to Discuss What Matters Most by Douglas Stone, Bruce Patton, and Sheila Heen · The Anatomy of Peace: Resolving the Heart of Conflict by The Arbinger Institute
Providing Feedback	Learning to provide both positive and negative feedback will help your team improve and grow in their careers.	· Radical Candor: Be a Kick-Ass Boss Without Losing Your Humanity by Kim Scott · Thanks for the Feedback: The Science and Art of Receiving Feedback Well by Douglas Stone and Sheila Heen

One of the first steps in developing your leadership skills is to take a step back and reflect on your strengths and weaknesses. What are your natural talents and areas of expertise? What are the areas in which you need to improve? Reflecting on these questions can help you identify where you need to focus your efforts.

Another important step in developing your leadership skills is to seek feedback from others. This can include colleagues, managers, and even team members. Ask for their honest feedback on your leadership style and be open to constructive criticism. This feedback can help you identify areas for improvement and give you a better understanding of how others perceive you as a leader. Reading books, articles and blog posts on leadership can be a great way to learn from others and gain new perspectives. You can also attend workshops or training sessions, or even take a course on leadership. The key is to stay curious and open to learning new things.

Leadership is not just about giving orders or making decisions, it's also about leading by example. Take on additional responsibilities, be a role model for your team, and set the standard for excellence. Be a good listener, be responsive, and be open to feedback. One of the best ways to learn leadership skills is to surround yourself with good mentors. Look for individuals who have experience and are skilled in leadership. Learn from their experiences and ask for guidance and advice. Mentors can provide valuable feedback and can help you see the bigger picture. When you take the time and effort to learn these soft skills, you may find it hard to apply them right away. When you’re facing new responsibilities, it is important to jump right in to figure out solutions and not approach things tentatively. Applying soft skills is very different from applying technical skills because there is no predefined script for what you should say in every single situation that you encounter. You will have to learn as you go.

3.1.3 Practicing Soft Skills On-the-Job

Part of learning soft skills is putting them into practice in the workplace. When you are leading teams, you must embrace failure. A friend of mine, Hetal Dave, Sitecore MVP, once told me, “You should fail. Without failure, you will never know what success looks like.” You will fail at times and that’s ok. You learn far more from failure than you do from success. Understanding what not to do and why is a powerful skill to have.

One of the hardest skills to master is conflict resolution. The first time you resolve conflict on your team it will probably feel uncomfortable. You must try your best to listen to all sides of the conflict and keep an open mind. You may go into a mediation session with a predetermined solution, but you should allow everyone to be heard. This is especially difficult when leadership is involved, and you want to impress them with your handling of the situation. You must remember, it’s not about you and your success. Your goal should be the success of the team.

Conflict is not always reported so a good way to learn how to resolve conflict is learning how to recognize unspoken conflict. Try paying attention to nonverbal cues, such as body language, tone of voice, and changes in behavior that indicate underlying tension. If team members are not listening to each other, they may go off on their own and take their own approach despite what was agreed upon. Also, notice when people have their arms crossed or appear indifferent when a team member is speaking. Once unspoken conflict is recognized, it is essential to address it in a constructive manner. You should encourage open communication and listen to team members' concerns to find a mutually beneficial resolution. By recognizing and addressing unspoken conflict, teams can work more effectively together and achieve their goals. It took me a long time to get used to resolving conflict. You should not expect to master this skill right away, it just takes practice. For more information on this topic, you can read my eBook, The Conflict Management Playbook.

Applying emotional intelligence and empathy is something you can do in your personal life as well as in the workplace. When you are always self-aware and can apologize if you have done something wrong, people will feel listened to. You should ask questions to find out how people view events or tasks they’re working on. One of the responsibilities of a Lead Developer is to reduce technical debt, however, this should not come at the expense of your team. You must observe how much they are working and ensure that they are not getting burnt out. Understanding that people have lives outside of work will help you to gain their trust and they will be more honest with you.

You can practice emotional intelligence and empathy by actively listening to team members' concerns, acknowledging their feelings, and responding with compassion and understanding. This could involve scheduling regular one-on-one meetings with team members to discuss their concerns, provide constructive feedback, and recognize their accomplishments. You should take the time to understand the challenges and pressures that team members may be facing outside of work and provide support and flexibility as needed. You can start the conversation by telling your team about your life and your own struggles. Some people may not be comfortable sharing personal information, so it is up to you to provide an example for them to follow. Lead Developers should build strong relationships with their team members and build trust.

Trust goes a long way to establishing honesty in your team. When people are honest with you and you do not have to read between the lines to determine how they feel, this is a great step in establishing proper communication. Communicating openly among your team makes it easier to provide frequent feedback whether it’s positive or negative. If there is a sense of trust between you and your team members, they will be empowered to share their opinions, and this will help you drive innovation on your projects.

3.1.4 Setting Aside Time for Learning

It’s easy to say, “You should set aside one hour for learning every day.” But will you be able to stick to that? I know I can’t! You can block off time in your calendar, but priorities are constantly changing, and you may find that you are not able to study every single day. I also suggest that you do not spend too much of your personal time studying as that is your time to relax. If you study in your personal time, you should read books and spend some time away from the monitor and keyboard. This is important for both your physical and mental health.

So, how do you stick to a learning schedule? I like to plan learning objectives for the week to support timeboxing my learning schedule. We are often overly positive when planning our goals for the week so you should aim to achieve 70% or more of your goals. Outrageous goals are called “moonshot goals” and while they are useful to inspire you to do your best, keep in mind that you should not make yourself feel bad if you do not achieve your goals 100%.

EXAMPLE WEEKLY LEARNING OBJECTIVES

The following list contains example learning objectives for a developer’s learning schedule:

- Create a GitHub repository

- Create a simple website from scratch including a heading, paragraph text, and dropdown box.

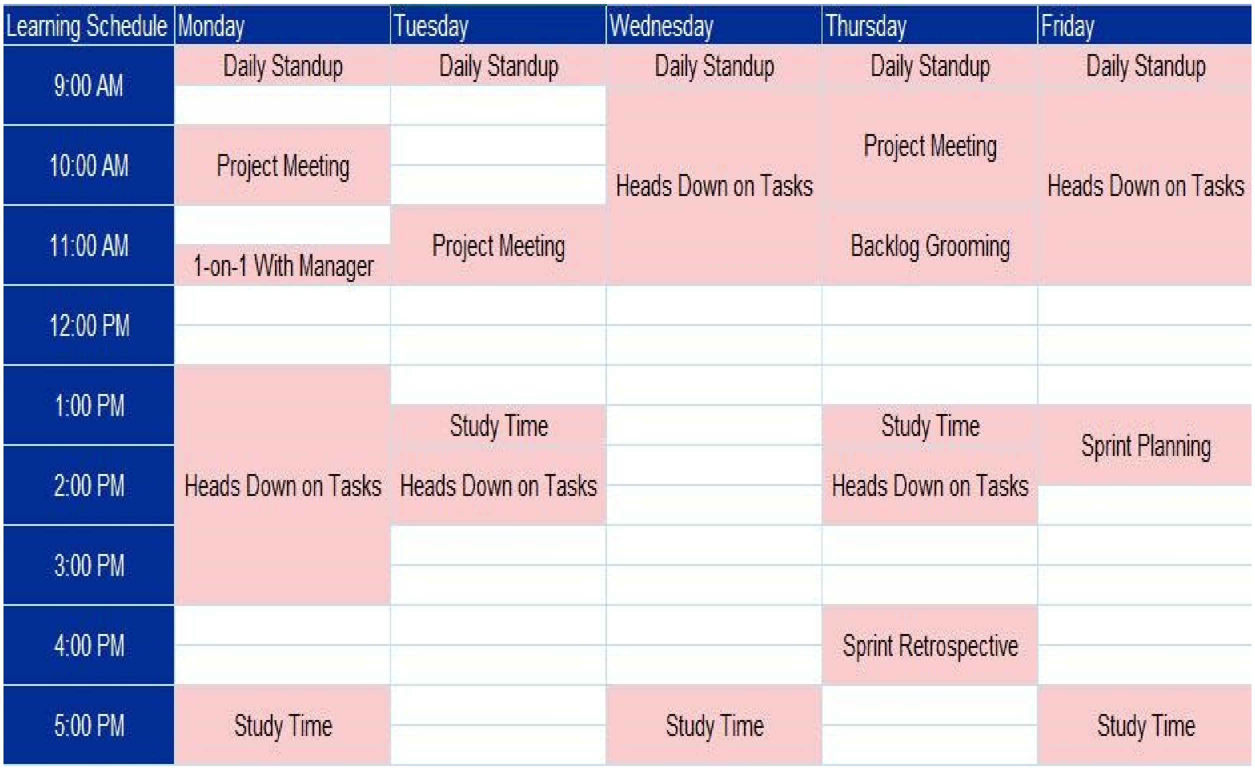
- Add CSS to a simple website and style all page elements

- Commit the completed website to the GitHub repository

If you have time blocked off in your calendar for learning, you should try your hardest to stick to it. If you’re in a meeting that is going over, ask yourself if it’s necessary for you to speak up and end the meeting. Everyone has a busy schedule, and they may also want the meeting to end on time so that they can move on with their day. This is where you can use emotional intelligence and observe the body language of the meeting attendees to assess if they have checked out of the meeting already. When people are clearly multi-tasking or have already packed up to leave, these are signs that they want the meeting to end. They will thank you for speaking up, especially if they are already late for their next meeting.

When your calendar is blocked off, you are not required to provide instant answers when people send you messages. Turn off notifications, and close your email, Slack, Microsoft Teams, and all social media apps. Multi-tasking is the killer of productivity. When you concentrate on one thing for even 15-30 minutes, you will do more in that time than you would if you allowed distractions and your task ended up taking three times longer.

Figure 3.1 Example Learning Schedule



Learning Schedule	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 AM	Daily Standup	Daily Standup	Daily Standup	Daily Standup	Daily Standup
			Heads Down on Tasks	Project Meeting	Heads Down on Tasks
10:00 AM	Project Meeting				
11:00 AM		Project Meeting		Backlog Grooming	
	1-on-1 With Manager				
12:00 PM					
1:00 PM	Heads Down on Tasks				
		Study Time		Study Time	Sprint Planning
2:00 PM		Heads Down on Tasks		Heads Down on Tasks	
3:00 PM					
4:00 PM				Sprint Retrospective	
5:00 PM	Study Time		Study Time		Study Time

You must learn to set boundaries for your study schedule. You are allowed to tell people no when they ask you to do things right away that are not emergencies. In these cases, the person requesting something from you may not be aware of your existing workload. You can put it back on them and say that you would be happy to work on their task, but you already have a full workload. Then, ask them what you can cut from your workload to make the time you need for them. You can also delegate the task to someone else on the team who is equally capable.

Making time to study new skills is very hard when you’re new to the Lead Developer role. If you cannot find large blocks of time, you can study for 15-30 minutes at a time. If you are always finding time to learn, you are doing what is necessary to be a successful Lead Developer.

3.2 Finding Your Leadership Style

Many leadership styles have been identified and studied over the years. These styles are often updated, and it is difficult to find a list containing every leadership style and variant that exists because there are so many. Leadership styles vary across industries and cultures. What works for one person may not work for another, which is why every leader must find their perfect fit.

The best leaders test out multiple leadership styles for various situations and personality types. Your approach to leadership should not be defined by one leadership style, rather it should include many different leadership styles to support both your needs and the needs of your team. Understanding your own personality traits and how they apply to different leadership styles will help you figure out what styles are best for you. You should also observe the personality types of the people on your team. This will help you support them and their needs so that they perform at a high level and produce quality work.

3.2.1 Reviewing Popular Leadership Styles

As I have studied leadership over the years, I have come across up to 30 different leadership styles in various industries. For the purposes of this book, I will narrow it down to the top ten most popular leadership styles for tech companies. Every leader should master several leadership styles that match their personality type and business needs. As you read through this list, think about how you already interact with your co-workers and how they respond to the leadership styles that are used in your workplace.

The following list includes the top ten most popular leadership styles:

- Autocratic: This style involves a leader making decisions on their own, without input from team members.
- Democratic: This style involves leaders seeking input and feedback from team members before making decisions.
- Transformational: This style involves leaders inspiring and motivating team members to be their best selves, and to work towards a shared vision.
- Transactional: This style involves leaders setting clear goals and expectations for team members, and providing rewards or consequences based on performance.
- Servant: This style involves leaders focusing on the needs of their team members and using their own skills and resources to help team members grow and develop.

- Visionary: This style involves leaders setting a clear and compelling vision for the future, and inspiring team members to work towards that vision.
- Coaching: This style involves leaders providing guidance and support to help team members develop their skills and reach their goals.
- Laissez-faire: This style involves leaders taking a hands-off approach and allowing team members to make their own decisions.
- Affiliative: This style involves leaders focusing on building strong relationships and creating a harmonious team environment.
- Commanding: This style involves leaders taking charge and issuing orders, often in high-pressure or emergencies.

Learning to combine these leadership styles and utilize them in different situations is difficult because there is no correct answer in your approach to each situation. You will learn as you go but it does help to understand the strengths and weaknesses of each leadership style by discussing the pros and cons of each.

Table 3.2 Pros and Cons of Popular Leadership Styles

Leadership Style	Pros	Cons
Autocratic	Decisions can be made quickly and efficiently.	May lead to low morale and a lack of creativity and innovation.
Democratic	Can lead to higher job satisfaction and commitment to the team.	Decision-making may be slower.
Transformational	Can inspire and motivate team members.	May not be effective in situations where quick decision-making is

		necessary.
Transactional	Can be effective in achieving specific goals and improving performance.	May not foster a long-term commitment to the team or company.
Servant	Can lead to a strong team culture and high levels of job satisfaction.	May not be effective in situations where quick decision-making is necessary.
Visionary	Can inspire team members and lead to long-term success.	May not be effective in times of crisis or when quick decision-making is necessary.
Coaching	Can help team members develop their skills and reach their goals.	May not be effective in situations where quick decision-making is necessary.
Laissez-faire	Can foster creativity and innovation.	May not be effective in achieving specific goals or in times of crisis.
Affiliative	Can create a harmonious team environment.	May not be effective in achieving specific goals or in times of crisis.

Commanding	Can be effective in emergencies or when quick decision-making is necessary.	May lead to low morale and a lack of innovation.

While the autocratic leadership style is listed, I do not recommend that any Lead Developer make decisions without the input of their team members regularly. However, sometimes you may need to make quick decisions and you will not have time to rely on your team. In this case, it is good to have autocratic leadership skills so that you can make effective decisions.

The same can also be said about the commanding leadership style, you do not want to implement the old school “command and control” leadership style by telling people what to do 100% of the time. However, this is a good skill to have in emergencies when people are scrambling, and you need to regain order quickly and formulate a plan. As a Lead Developer, you should expect emergencies to happen, even if you are at the top of your game, system errors and failures happen, and you need to be prepared to handle them.

Lead Developers are in a support role so you will focus mainly on supportive leadership styles including transactional and servant leadership. You must ensure the productivity of your team and support them by answering questions and removing blockers so that they can complete their tasks. Part of supporting your team includes mentoring which is included in the coaching and transformational leadership styles.

As a Lead Developer, you may not need higher-level leadership styles in your current job, but it is good to practice these if you want to move into managerial roles in the future. Being a visionary leader is usually something that is attributed to the CEO of a company as they are responsible for communicating the vision and direction of the company. However, you can do this at the team level, and this is something that you would do as an Engineering Manager or Director.

The following list includes popular leadership books to help you learn more about different leadership styles:

- "The Art of War" by Sun Tzu
- "Good to Great: Why Some Companies Make the Leap and Others Don't" by Jim Collins
- "The 7 Habits of Highly Effective People" by Stephen Covey
- "Leadership and Self-Deception: Getting Out of the Box" by The Arbinger Institute
- "Drive: The Surprising Truth About What Motivates Us" by Daniel H. Pink
- "Start with Why: How Great Leaders Inspire Everyone to Take Action" by Simon Sinek
- "The Power of Servant-Leadership" by Robert K. Greenleaf
- "The Five Levels of Leadership: Proven Steps to Maximize Your Potential" by John C. Maxwell

There are many learning resources to help you in your journey to becoming a successful Lead Developer. The books listed above are excellent sources and should be a staple in any leader’s library. You may also want to listen to popular leadership podcasts such as Leadership and Business by The Harvard Business Review and The Leadership Lab by The Center for Creative Leadership. Understanding the main leadership styles will help you as you navigate your responsibilities as a Lead Developer. You should give every single one a go and observe the results to assess what works for you and what does not. Not all leadership styles are a fit for everyone and you will need to consider your own personality type when forming your own personalized leadership style.

3.2.2 Assessing Your Personality Type

Most people believe that you must be an extrovert to be an effective leader but that is simply not true. I have introverted tendencies and have known many Lead Developers who were effective in that role and were introverts. Many developers are introverted and therefore they think that leadership is not for them. Research has shown that introverts can bring unique strengths to leadership positions, such as the ability to listen carefully, think deeply, and communicate clearly. It is important to remember that leadership is not only about your personality type, but rather about your abilities and skills. Both introverts and extroverts can be successful leaders, provided they can effectively communicate, inspire, and motivate their team.

Several personality assessment tools are commonly used in the industry for leadership development and selection. Some of the most well-known and widely used ones include the following:

- The Myers-Briggs Type Indicator (MBTI): This is a widely used personality assessment tool that helps individuals understand their personality preferences and how they perceive and make decisions.
- The Five-Factor Model (FFM): This is a personality assessment tool that measures five broad dimensions of personality, including openness, conscientiousness, extraversion, agreeableness, and neuroticism.
- The DISC assessment: This is a personality assessment tool that measures four dimensions of personality, including dominance, influence, steadiness, and compliance.
- The Leadership Styles Inventory (LSI): This is a personality assessment tool that measures an individual's leadership style and helps them understand their strengths and areas for development.
- The Emotional Intelligence Quotient (EQ) assessment: This is a personality assessment tool that measures an individual's emotional intelligence, which includes their ability to recognize and understand their own emotions and the emotions of others.

The results of a personality assessment can be a helpful tool for understanding your own leadership style and how you might be perceived by others. By understanding your personality traits, you can gain insight into your strengths as a leader and areas where you may need to work on developing your skills. For example, if you score high in conscientiousness, you may be organized, reliable, and detail-oriented, which could be beneficial for your leadership style. If you score high in extraversion, you may be confident, energetic, and skilled at motivating others.

To determine your leadership style using the results of a personality assessment, you might try the following steps:

- Review the results of your personality assessment and identify your dominant personality traits.
- Reflect on how these traits might influence your leadership style. For

example, if you score high in agreeableness, you may be inclined to build consensus and foster positive relationships with team members.
- Consider how your personality traits might be both strengths and challenges in your role as a leader.
- Identify any areas where you would like to develop your leadership skills and consider what specific actions you can take to do so.
- Seek feedback from others, such as team members or colleagues, about your leadership style and how it is perceived by others.

Use this information to adapt and refine your leadership style as needed. It is not accurate to say that certain personality types make better leaders than others. Leadership may be influenced by one's personality type, but it is your abilities and skills that will make you a good leader. Both introverts and extroverts can be successful leaders, and each personality type has its own unique strengths and weaknesses.

Some research has suggested that certain personality traits may be more conducive to leadership. For example, individuals who are confident, decisive, and able to adapt to changing circumstances may be more likely to succeed as leaders. These personality traits are not necessarily tied to any specific personality type, but rather can be found in individuals of all types. A personality trait is a specific characteristic or aspect of a person's behavior, thoughts, and feelings that describes how they tend to behave or respond to certain situations, whereas a personality type is a broader categorization that groups individuals based on a combination of several traits, often in a systematic way as in the case of Myers-Briggs Type Indicator. In simpler terms, a personality trait is a building block, and a personality type is a collection of those building blocks put together to form a bigger picture. A successful leader does not have to be a specific personality type, but they do need to have personality traits that complement leadership skills.

Ultimately, the most effective leaders are those who can understand and work with their own strengths and limitations, and who can adapt their leadership style to the different personalities of the people on their team and the situation at hand. Leadership is always in flux and your style will change and adapt with experience.

3.2.3 Observing Personalities on Your Team

Understanding team members' personalities can help you communicate and interact with them more effectively. For example, if you know that a team member is introverted, you might approach them differently than you would an extroverted team member. Observing team members' personalities can help you to identify and address any potential conflicts or misunderstandings within the team. By understanding how team members approach problem-solving and decision-making, you can help to facilitate more productive and harmonious interactions.

Understanding team members' personalities can also help you tailor your leadership style to better suit the needs and preferences of the team. If you know that a team member is highly motivated by recognition and praise, you might make a point of praising them more frequently. Observing team members' personalities can also help to identify potential areas for growth and development. By understanding what drives and motivates your team, you can help to create a supportive environment that encourages personal and professional growth.

There are many ways that a Lead Developer can observe the personalities of their team members. You should pay attention to how team members communicate and interact with one another. Do they tend to be more introverted or extroverted? Do they prefer to work independently or as part of a team? You need to adjust your communication style to suit the preferences of different team members. For example, an introverted team member might prefer written communication or one-on-one conversations, while an extroverted team member might thrive in group discussions.

You should take notice of how team members approach problem-solving and decision-making. Do they tend to be more analytical or intuitive? Do they prefer to consider all the options before they make a decision, or do they prefer to act quickly? An analytical team member might appreciate access to data and research materials, while an intuitive team member might benefit from more hands-on learning opportunities. By providing tailored support and resources, a leader can help team members to feel more supported and valued.

Consider how team members respond to feedback and criticism. Do they tend to be more open and receptive to feedback, or do they become defensive? Supporting team members who are receptive to feedback and those who are not can be a challenge for any leader. For team members who handle criticism poorly, it can be helpful to provide it in a private setting, rather than in front of the whole team. This can help to reduce feelings of embarrassment or shame and create a more open and supportive environment for feedback. I have handled giving negative feedback in a private setting many times for my own employees and I have witnessed employers who handled it publicly. Public shame kills morale, and it does not help the person feel supported. If a team member is struggling to accept and respond to feedback, consider offering additional support and resources to help them improve. This might include coaching, mentorship, or access to training and development opportunities.

Take the time to notice how team members handle their workload. Do they tend to become overwhelmed, or do they remain calm and composed under pressure? Different team members may have different preferences when it comes to how they work best. Some may prefer a more structured environment, while others may thrive in a more flexible setting. By offering flexibility and allowing team members to work in ways that suit their personalities, you can help to create a more positive and productive work environment.

Pay attention to team members' values and motivations. What is important to them, and what drives them to succeed? Different team members may have different goals and areas for growth and development. By providing opportunities for learning and personal development, and by offering support and guidance as needed, a leader can help team members to reach their full potential.

By observing these and other aspects of team members' personalities, you can gain valuable insights into how best to support and manage your team.

3.3 Improving Your Presentation Skills

Public speaking can be a daunting task, but it does not have to be. Whether you're presenting to a group of coworkers, clients, or students, it's natural to feel anxious about speaking in front of others. However, with proper preparation and the right mindset, you can calm your nerves and deliver a confident and effective presentation. In this section, we will explore some strategies for overcoming nervousness before a presentation, including practicing and visualizing your success, getting feedback, arriving early and setting up equipment, and relaxing with breathing exercises and positive self-talk. While it may take time to master the art of being calm before, during, and after a presentation, with practice, you can become an excellent public speaker.

Some desired outcomes from professional presentations can include:

- A clear and concise message that is easily understood by the audience
- A single takeaway or a few key points that the audience can remember and apply
- Increased understanding or knowledge of the topic at hand
- Inspiration or motivation for the audience to take action
- Changed minds or perspectives on a particular issue or topic
- Increased interest or engagement in the subject matter
- Building credibility and establishing oneself as an expert in the field

It really depends on the purpose of the presentation and the audience, whether the outcome should be a single takeaway, a few key points, or to change minds. Ideally, a good professional presentation should aim to achieve a combination of these outcomes. Being able to present information is a requirement for conducting demos and walkthroughs or training. When you are mentoring developers, you must be able to discuss technical architecture in detail. These discussions should be short and to the point so you must organize the information so that it is easily consumable. If you give a bad presentation, you may lose trust in your team or even worse, with clients and stakeholders. When you cannot communicate effectively, this can lead to miscommunication and time wasted working on the wrong things. The following list includes examples of effective presentations that have helped me in my career.

- .NET Overview & Roadmap by Scott Hanselman and Scott Hunter
- The Art of Computers by Scott Hanselman

- Everything You Thought You Already Knew About Orchestration by Laura Frank Tacho
- Forward-thinking: What’s next for AI by IBM

Effective communication is crucial for Lead Developers to effectively convey technical information to their team and stakeholders. You need to be prepared and remain confident throughout your presentation, which is not an easy task at first. As time goes on, you will get better with practice, trust me. Learning how to engage with and manage an audience is a very important skill that Lead Developers need to convey the appropriate message to both developers and non-developers.

3.3.1 Calming Your Nerves

The best advice anyone ever gave me about public speaking is that you should be nervous. You feel nervous because you want to do a good job and convey your message in its entirety. If you aren’t nervous, kudos to you! Most of the public speakers in my network get nervous before speaking at an event or giving a presentation. You are not alone!

You should always practice your presentation beforehand. The more comfortable you are with your material, the less nervous you'll be. This also helps you keep the pace and ensure proper timing, so your presentation is not over or under the allotted time. As you practice, visualize yourself giving a successful presentation. Imagine yourself feeling confident and in control. This will go a long way to ensure your success because if you think you will succeed, then you are in a mindset to support your own success. If you think you will fail, you are setting yourself up for failure. Instead, focus on your audience. Remember that they want you to succeed and are rooting for you.

When you practice your presentation, you can also record yourself and watch it back to assess your performance. You may notice that some parts of your presentation are confusing or that you need to rearrange certain sections. Practicing and recording your presentation allows you to identify and fix these issues before you give the actual presentation. Seeing yourself give the presentation can help you feel more confident and prepared when it's time to give the actual presentation. It also allows you to get feedback from others.

Showing your practice presentation to a friend or colleague can give you valuable feedback on your delivery, content, and overall effectiveness. It can help you feel more confident.

Arriving early and setting up your equipment beforehand will help you feel more in control and prepared. Doing this will also mitigate any technical issues that may arise in the conference room or video conference. If you're doing an in-person presentation in a conference room, make sure that someone who knows the equipment is there to help you in case you need assistance. If you're doing a video conference and you are not the host, ask the host to join 15 minutes early to help you get set up to ensure everything is working properly before the meeting starts.

On the day of your presentation, try to participate in relaxing activities such as breathing exercises and meditation. Taking a few moments to sit quietly and think of nothing at all is a proven method that will reduce stress and tension. If you are tense at the beginning of your presentation, it will be hard for you to give a good performance throughout. Try taking deep breaths. Inhale slowly and exhale slowly to help calm your nerves. Use positive self-talk. Tell yourself that you are prepared and that you can do this. Try to relax your body. Tense and release different muscle groups to help release tension.

Being calm before, during, and after your presentation will take time for you to master so don't be worried if it doesn't happen immediately. I used to get so anxious before a presentation that my hands would shake and it was hard to operate the mouse and keyboard. Practicing by yourself or with friends is different from giving a presentation to a room full of people that you may not know very well. But the more you do it, the more you will improve to become an excellent public speaker.

3.3.2 Creating Effective Slides

As a Lead Developer, it is important to communicate technical information effectively to your team and other stakeholders. One powerful tool for achieving this is the use of visual aids, such as slides. When done well, slides can help you clarify your points, illustrate complex concepts, and engage your audience. However, creating effective slides requires careful planning and attention to detail.

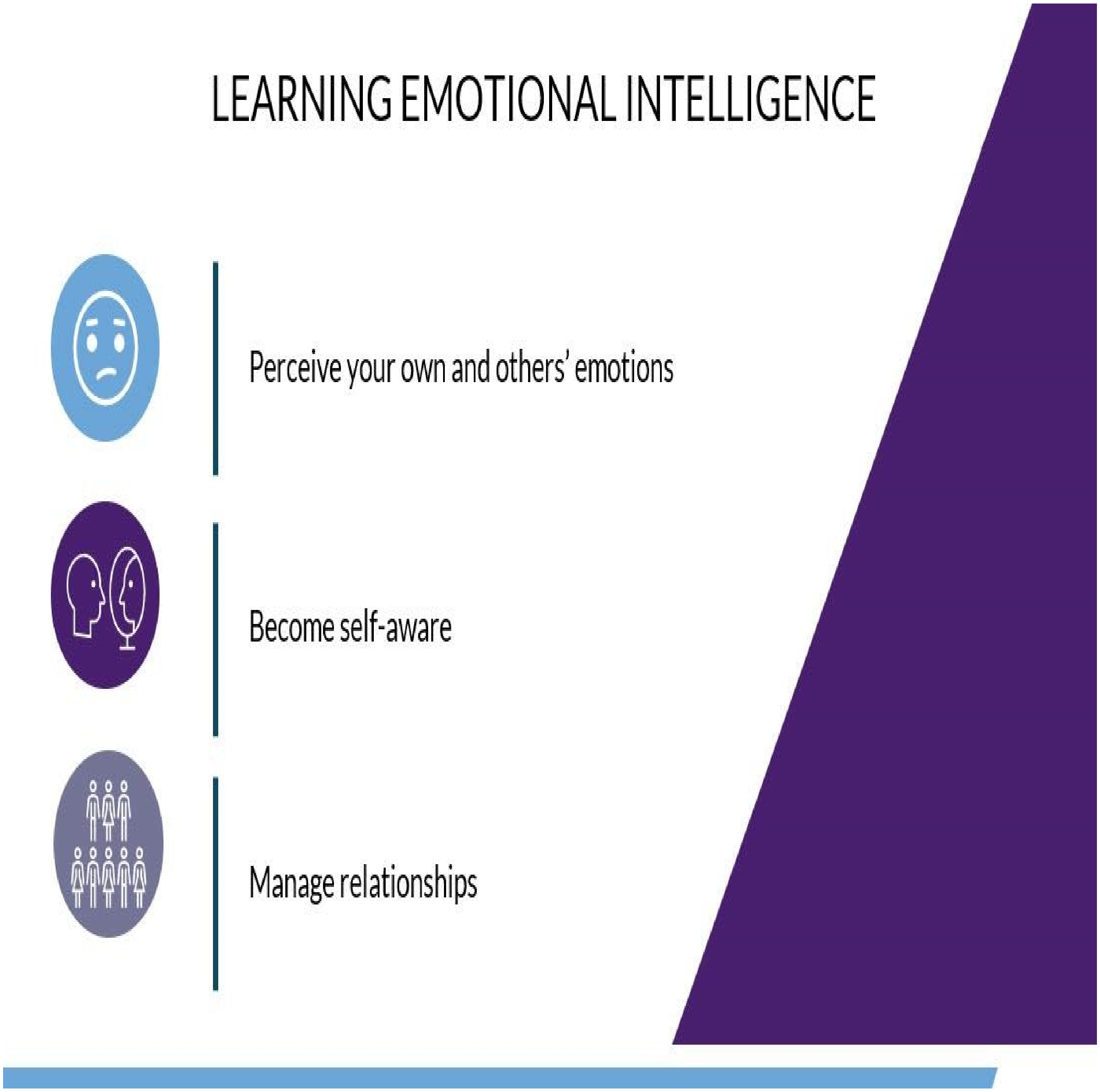
To create effective slides, it's important to keep them simple and focused. Avoid cluttering your slides with too much text or too many graphs and charts. Instead, choose a few key points and use clear, concise language to convey them. Use headings, subheadings, and bullet points to organize your content and make it easy to follow. If your company has a style guide for presentations, you should study it and ensure that you follow any guidance provided.

Another important factor to consider is the use of fonts and images. Choose a font that is easy to read from a distance and use a large enough font size to avoid straining your audience's eyes. When it comes to images, be sure to select ones that are relevant to your content and of high quality. Poorly designed or low-resolution images can distract from your message and make your presentation appear unprofessional.

In addition to these general tips, there are a few considerations specific to development teams that you should keep in mind when creating your slides. For example, you may want to include code examples or debugging output to illustrate specific points. In these cases, it's important to use a font and layout that make the code easy to read and understand.

Finally, use color and layout effectively to highlight important points and create a visually appealing presentation. Choose a limited color palette and use it consistently throughout your presentation, such as the color palette for the company brand. Use headings and subheadings to organize your content and make it easy for your audience to follow along.

Figure 3.2 Example of an Effective Slide



LEARNING EMOTIONAL INTELLIGENCE

Perceive your own and others' emotions

Become self-aware

Manage relationships

While you may be comfortable speaking to other developers, communicating technical information to non-developers can be more challenging. Before diving into the technical details, it's important to provide some context and establish a common understanding. Begin by explaining the problem you are trying to solve or the goal you are trying to achieve. Use clear, concise language and avoid using jargon or technical terms that may be unfamiliar to your audience.

Use analogies and examples, as they can be a powerful tool for explaining complex concepts to a non-technical audience. For example, you might use an analogy to explain how a particular software component works by comparing it to something in the real world, such as a car engine or smart home device. Similarly, you can use examples to illustrate how a particular technology or process works in practice.

By following these tips, you can create effective slides that will help you communicate your message effectively and professionally to your development team. Remember, the purpose of your slides is to support and supplement your presentation, not to distract from it. With careful planning and attention to detail, you can create slides that will help you effectively communicate your message and engage your audience.

3.3.3 Giving a Great Performance

A passive audience is not a good audience. You must keep your audience engaged by asking questions, soliciting feedback, and encouraging discussion. Use a variety of delivery techniques, such as storytelling or anecdotes, to keep your audience interested. At the beginning of your presentation, you should ask the audience leading questions so that you can tailor your material for them. If you’re working on a project team or consulting gig, it is best to ask the team about your audience beforehand, but you should also ask at the beginning of your talk to mitigate any misinformation you may have been given about your audience beforehand. For example, ask everyone what they hope to learn from your presentation. Hopefully, their answers will be in line with what you have prepared but you should be ready to improvise on the fly.

Improvisation is a difficult skill to master. While most people suggest that you join a class or group to learn public speaking skills, I suggest that you take an improv class. These classes are mostly geared toward performers and comedians, but you need to think of your presentations as performances. Whenever you’re in front of an audience, it is a performance. People expect to engage with what you’re saying and if they are not engaged, then you will need to quickly change gears to improve your performance which is where improv skills come in handy. For example, if you’re speaking about an advanced programming topic, you may find that some people in your audience are beginners. In this case, you should try to explain things in full without leaving out key information so that the beginners can understand what you are talking about, even if it’s at a lower level.

Confidence is key to giving a great performance. Remember that you are the expert on the topic you are presenting, and your audience is looking to you for guidance and information. Believe in yourself and your abilities, and your confidence will shine through in your delivery. Do not let imposter syndrome get to you. You were selected to present because of your expertise. We will discuss imposter syndrome in detail later in this book.

It's also helpful to seek support and feedback from colleagues and mentors. They can provide valuable insights and encouragement as you work to build your confidence as a public speaker. Finally, remember that it's okay to make mistakes or not know all the answers. No one expects you to be perfect and acknowledging that you don't have all the answers can make you more relatable to your audience. If someone asks you a question and you do not know the answer, tell them that you will research the topic and get back to them after your presentation is over. When you find the answer, you should let everyone know, not just the person who asked because someone else may be wondering the same thing.

Thinking of your presentations as performances will help keep your audience engaged and interested in what you’re talking about. You don’t have to be a professional actor to learn the art of performance or improvisation. But learning a bit about these skills will help your confidence level improve as you gain experience.

3.4 Case Study

Maureen Josephine is a Software Engineer and a Google Developer Expert in Flutter & Dart. She was the first female Flutter Google Developer Expert in Sub-Saharan Africa and the first flutter Google Developer Expert in Kenya. She is a Technical Speaker, Technical Writer, and the lead Community Organizer for Flutter Kisumu. She is a recipient of the McKinsey & Company Next Generation Women Leader Award 2020. She enjoys exploring and learning new things. Besides coding, she is passionate about fashion and design, and she gets her inspiration from nature, African culture, and Technology. In this case study, she provides insight into her experience learning lead developer skills.

3.4.1 What soft skills have you learned that helped you the most in your career?

Communication skills have helped me a lot because no one can know what you want or need other than you. You must effectively communicate your needs to others and not be afraid to ask questions. Good communication goes two ways, you also need to be able to receive feedback from others. If you receive good feedback, that’s great! No feedback is not good because without feedback, how will you improve? And negative feedback can be difficult to take but you cannot take it personally. Put yourself in their shoes and think about the negative feedback with an open mind so that you can assess the need for your own professional development.

You should ask people what they need if you notice them struggling. Some people shy away from asking tough questions because they are afraid of looking like they do not know something. I studied CS at university, and I felt afraid to ask questions at my first internship. You learn a lot of theory at the university but then you get out into the real world that requires more practical work, and you realize that you do not know that much. I got stuck trying to complete tasks based on the reading assigned to me by my mentor because I did not ask questions. I almost quit because of this! Finally, I discovered the need to communicate with people and be proactive about learning new skills and asking questions.

3.4.2 How do you prepare for giving a presentation?

The most important thing is to be confident. People are attending your presentation to listen to you and learn something new. It helps to do your research so that the information you are presenting is accurate. One time, I gave a talk on AI, Flutter, and Machine Learning. I had to read a lot about Machine Learning, and I learned a lot. But it was good that I did that because I was able to share my insights so that the audience learned something from my presentation.

As you’re preparing for a presentation, you should not procrastinate on creating your slides. Make them about two weeks before your talk. I regret procrastinating on my slides for some of my previous presentations because I did not have enough time to think about the slide design and make good slides. It’s also nerve-wracking to be making slides a few days or even hours before your presentation.

Right before your presentation, you can calm your nerves by massaging your jaw and cheeks for a few minutes. This helps to relax your jaw and reduce anxiety. You can also write down or tell yourself positive affirmations such as “you can do it” or “you look great”. Looking good will also boost your confidence because you do not have to worry about what people think of your appearance.

Do not be afraid to engage with your audience. You should start your presentation by asking the audience if they know about the topic. This is a good way to get to know the audience before you start talking. When I did my first talk in Hamburg, Germany, it was the first time I gave a tech talk physically in Europe coming from an African background, and I felt nervous because I did not know what to expect from the audience. When I asked them what they knew about my topic, I got some really good feedback that I incorporated into my presentation.

3.4.3 Why is it important to observe personalities on your team?

When you’re working in a global community, you must realize that people are very different based on their culture. You may think you’re making jokes but, in some cultures, what you are saying is offensive. You should observe your environment and how things work considering the energy that the team gives off. When you or someone else on your team is giving feedback, you can assess if a person takes that feedback well or not. You cannot treat everyone equally as to how they react to things. People are different and you will have to approach giving feedback differently for different types of personalities. Putting yourself in their shoes helps to gain insight into where they are coming from. The last thing you want to do is accidentally create enemies.

3.4.4 What advice do you have for Lead Developers with a busy schedule to help them prioritize learning new skills?

When you are busy, it seems like every day something new comes up. It is important for you to maintain discipline. You should learn using a schedule that suits you. Some people learn better and are more alert in the morning, while others prefer the evening. Do not set a schedule based on someone else’s schedule. Their life is probably much different than yours and their schedule may not work for you. Your schedule can be as little as 30 minutes per day, but the important thing is to be consistent.

Also, think about what is next after you learn the new skill. What is the end goal or main objective? What is driving your motivation toward learning that skill? When you are learning new skills, you need to also practice your new skills. You need practical skills and textbook knowledge to implement what you have learned. You can make your own project to practice and share your learning with others. When you learn something new, you can create videos, blogs, articles, or even Tweets. I see a lot of people sharing knowledge on Twitter and I think that’s really cool! I have written many articles and answered questions on Stack Overflow, then later I forget the information on that topic, and I Google it only to find my own work. That goes to show that you can help not only the community but also your future self by sharing the information that you have learned.

If I’m researching something for a problem at work, I would take time from my workday to learn that specific topic. Sometimes you need a small bit to make something work, and if you have a deadline, you need to find the solution for your specific problem quickly. But afterward, you can take the time to learn a more in-depth overview of how it works overall. This will help you balance your learning between your work hours and personal time.

3.5 Summary

- Lead Developers must have a balance of both technical and soft skills.
- Being a lifelong learner and prioritizing learning vs. your daily tasks is essential for success as a Lead Developer.
- There are 10 popular leadership styles for tech companies: autocratic,

democratic, transformational, transactional, servant, visionary, coaching, laissez-faire, affiliative, and commanding.

- Lead Developers should prioritize supportive leadership styles such as transactional, servant, coaching, and transformational.
- Lead Developers should also be prepared to use higher-level leadership styles such as visionary and commanding in the future or emergency situations.
- It's normal to feel nervous before giving a presentation but practicing beforehand can help reduce nerves and ensure a successful performance.
- Visual aids, like slides, can help clarify points and engage an audience, but it's important to keep them simple and focused, and to use clear language and high-quality images.
- To communicate technical information effectively to a non-technical audience, it's important to provide context, use analogies and examples, and avoid jargon. It can also be helpful to seek support and feedback from colleagues and mentors.

4 Learning Any Developer Skill

This chapter covers

- Reviewing different learning styles and content types that are applicable to learning development skills
- Assessing your learning style and evaluating content types that are best for you
- Understanding how mental and physical health work to boost your memory and enable you to retain more information
- Examining different learning blockers and how to overcome them
- Discussing how community projects help developers collaborate with their peers and learn new skills quickly
- Developing personal projects to gain hands-on experience and build your portfolio
- Setting attainable goals for learning new skills

Learning new technical skills is a crucial part of being a developer, but it can also be a challenging and overwhelming process. Whether you're just starting out or you're a seasoned pro, there's always something new to learn and ways to improve your skills.

One of the biggest problems that developers face when learning new skills is feeling overwhelmed by the amount of information available. With so many resources and tutorials at our fingertips, it can be difficult to know where to start and what information is most relevant. When developers find the information that they need, they may lack the time needed for practice or application. It's one thing to read about a new concept or technology, but it's another thing entirely to put it into practice. Developers often struggle to find the time or opportunity to apply what they're learning in real-world scenarios.

Developers also face the problem of imposter syndrome. It's easy to feel like you're the only one who doesn't understand a new concept or technology, especially when you're surrounded by more experienced developers. Plus, working with experienced developers can be intimidating and you want to impress them with your skill level. This happens to everyone, even me (especially me)! Just remember you are not alone. We will discuss this topic more later in this chapter when we talk about physical and mental health.

Another problem that developers face is a lack of patience and perseverance. Learning new technical skills takes time and effort, and it's easy to get discouraged when progress is slow. The goals that developers set during one-on-one meetings with their manager are often too aspirational, so they are not set up for success. It’s great to have goals but they can also make you feel like a failure when you do not achieve them. This often prevents you from making progress or even ditching your learning goals entirely.

These are the problems that face many developers, but you can overcome them. Even if we set attainable goals, life happens and priorities shift. Remember that learning is a lifelong journey, and the most important thing is to enjoy the ride.

4.1 Improving Your Learning Methods

It is important to consider your own learning style and ways to reflect on the format which makes you comfortable and helps you retain the information better. Assessing your preferred content types is important in effectively learning new technical and soft skills and improving your learning methods. The most popular content types include textbooks, blog posts, online courses, instructional videos, and audiobooks. With so many options available, it can be overwhelming to know where to start and which types of content will be the most effective for you. When you encounter learning blockers, it can be difficult to overcome them if you do not know where to go to find the information that you need. Many developers (me included) will often try to break through learning blockers by just continuing to work without a break until the solution presents itself. However, there are other, more healthy ways of dealing with learning blockers without overworking yourself.

You can retain information at a high level while you are learning new skills by using tactics such as mnemonic devices. A mnemonic device is a tool or technique that helps you remember something. It's a simple and fun way to make learning new information more efficient and effective. Mnemonic devices can take many forms, such as a catchy phrase, rhyme, or acronym, which make it easier to recall complex information or a list of items. They're like little memory helpers that can make it a lot easier to remember important facts, figures, and dates.

Lead Developers must balance their time between their work responsibilities and keeping their skills up to date. This can be difficult without guidance and setting attainable goals. As discussed in the previous chapter, you must make the time to learn new skills. It is a good idea to block off your calendar to ensure that you will have the ability to learn new skills on a regular basis. When it comes to learning new skills, it is ultimately up to you to balance your time and keep your mind sharp so that you learn quickly and retain what you have learned.

4.1.1 Understanding How You Learn

Sifting through the endless resources available for learning new technical and soft skills can take a lot of time and patience. With so many options available, it is hard to know where to start, and more importantly, which types of content will be the most effective for you.

To determine your preferred content types, you should experiment with different formats and evaluate which ones work best for you. Some common formats include:

- Video tutorials
- Written tutorials and documentation
- Interactive coding exercises
- Code samples
- Podcasts
- Audiobooks

By trying out a variety of formats, you can gain insight into which content types are most effective for you and use that knowledge to guide your future learning. I prefer written tutorials which people find interesting because I create online courses that are video based. My preference comes from beginning my development career before video tutorials were available. Now that they are available, I like having a combination of a video and written instructions. One way that you can assess if video tutorials are for you is by watching a tutorial on a topic that interests you. After watching the video, try reading an article or documentation about the same topic. Take note of how you process and retain the information from each format. Are you able to absorb more from the video tutorial? Or does reading the documentation make it easier for you to understand the material?

Another effective method is to practice what you have learned. Try working through some interactive coding exercises that apply the concepts you've learned. This will give you a chance to apply what you've learned in a hands-on way and see if you can successfully implement the concepts on your own. As you work through the exercises, take note of how you feel and how well you're able to understand the material. Are you feeling more confident in your abilities? Or are you having a hard time following the instructions? Code samples can also be a great way to assess yourself. Try reading through code samples in a language or technology you're trying to learn and see how well you understand the code. Take note of any areas where you're struggling and plan to study those concepts more thoroughly.

It's also important to consider your own learning style. Every individual is different in the way they process information. Some people are visual learners, who prefer to see information presented in diagrams, illustrations, or pictures. Some are auditory learners, who prefer to hear information in lectures, podcasts, or audiobooks. Lastly, some people are kinesthetic learners, who prefer to use their hands to touch and manipulate objects while they learn. Reflect on the format which makes you comfortable and helps you retain the information better.

By trying out different content types, practicing what you've learned, and reflecting on your own learning style, you'll be able to gain a deeper understanding of which types of content will be the most effective for you in the future.

4.1.2 Boosting Your Memory

Boosting your memory while learning new skills can help you retain information more effectively and become a more efficient and effective developer. This will help you reduce the time you spend studying as you are working smarter, not harder. You will be better equipped to use critical thinking and keep in mind similar skills you have learned in the past or overcome common errors you have encountered before.

Practicing active recall is the process of retrieving information from your memory without the aid of prompts or cues. Instead of passively reading through documentation or tutorials, try to actively recall what you've learned by testing yourself or teaching the material to someone else. This helps strengthen the connections in your brain and makes it more likely that you'll remember the information later. As you gain experience, you may see the same errors occurring frequently. When this happens, you will be able to quickly fix those errors especially if you have seen them before and it took you and your team a long time to fix them. Trust me, you will remember how to resolve errors like this off the top of your head when you have been through the stress of trying to figure them out in the past.

Mnemonic devices are memory aids that can help you remember information more easily. For example, you could use an acronym to remember a list of programming terms or create a mental image to help you remember a specific syntax. Mnemonic devices can be especially helpful when you're trying to remember something that's difficult to visualize, such as a complex algorithm or data structure. There are several different types of mnemonic devices that developers can use to retain information about programming skills.

Table 4.1 Mnemonic Devices with Examples for Developers

Mnemonic Device	Definition	Example
Acronym	Word made up of the first letters of a phrase or name	SOLID: Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, and

		Dependency Inversion
Acrostics	A sentence or phrase in which the first letter of each word represents a letter in another word or phrase	CRUD (Create, Read, Update, Delete) application = "Careful Reading Uncovers Deeper Meaning"
Chunking	The process of breaking down information into smaller, more manageable chunks	Break down long code strings into smaller groups, like a specific function or a group of related variables
Imagery	The use of visual images to help remember information	An image of a tree to represent the hierarchical structure of a program, with the trunk representing the main function and the branches representing sub-functions
Rhymes	Correspondence of sound between words or the endings of words	“Agile projects go round and round, with planning, doing, checking, and adjusting to being found” = key stages in Agile project management

Large amounts of information can be overwhelming and difficult to retain. Instead of trying to take everything in at once, break complex concepts down into smaller, more manageable chunks. Try focusing on one aspect of a new skill or technique at a time, and gradually building on what you've learned. If the concept is building a new mobile application, the main components could include developing the backend functionality and components. You can create a task list or a to-do list for each component, with smaller, more specific tasks that need to be accomplished in order to complete that component.

4.1.3 Taking Care of Your Physical and Mental Health

Research has shown that sleep, stress level, and mental health play an important role in the consolidation of new information in our memory. Ensure you are getting a good amount of restful sleep and managing your stress level as they can impact your physical health making it harder to learn and retain information. Your mental health is equally as important to keep yourself balanced and alert. You should exercise regularly, watch your nutrition, and not overwork yourself. Being healthy will help to enhance your ability to learn and retain new information.

Taking care of your physical and mental health is important for overcoming imposter syndrome. Make sure you're getting enough sleep, eating healthy, and taking breaks when you need them. Exercise can also be a great way to relieve stress and improve your mood. Additionally, consider talking to a therapist or counselor if you're struggling with imposter syndrome or other mental health issues. I still get imposter syndrome and something that helps me is reframing my negative thoughts into positive thoughts. Instead of thinking, "I don't know what I'm doing," try thinking, "I'm still learning, and that's okay." Instead of focusing on your mistakes, focus on what you've learned from them. Remember that everyone makes mistakes, and they're an opportunity to grow and improve.

As developers, we often find ourselves in a fast-paced and ever-changing environment. While this can be exciting and stimulating, it can also be stressful. Stress can take a toll on our physical and mental health, and it’s important that we take the time to manage it effectively. It’s easy to get caught up in the hustle and bustle of our work, but it’s crucial that we make self-care a priority. This can include things like exercise, meditation, and hobbies. Make sure you have time for yourself and your loved ones. Don’t hesitate to reach out to colleagues, friends, or a professional if you’re feeling overwhelmed. Remember that it’s okay to take a step back, take a deep breath and regroup. You are not expected to know everything, and it is important that you ask for help. Your well-being is important and taking care of yourself will ultimately help you to be more productive, effective, and fulfilled.

It is also important to set boundaries and be able to say no to people. Lead Developers are often approached numerous times per day to answer questions or engage in project conversations. If you have a full workload and someone comes to you with a request, it is a good practice to tell them that you want to help them, but you cannot help them right now. When people make requests like this, they are often not expecting immediate action anyway. Project Managers exist to help you manage your workload and you should always refer to them if you are feeling overwhelmed.

I was never good at saying no to people early in my career. I could not set boundaries and I was a people pleaser. I once had a boss who requested that I take a train from London to Belgium at 8 PM at night to meet with a client the next day and I agreed to it. Not only was it dangerous for a young woman to arrive in an unfamiliar country late at night by herself, but I didn't get to sleep at all that night because I had to prepare for the meeting. I had every right to decline my boss' request.

When you do not set proper boundaries, the stress of overworking yourself will take a toll on your physical and mental health. I was hospitalized for a week once because I had been traveling internationally twice a month for too long and I got very sick. Even after that, I didn't get a wake-up call until 4 years later and I realized I was not getting better because I went right back to work and continued to overwork myself. I had a mental breakdown at that point which is what forced me to finally take time off of work. It changed my life and now I put my health first. You cannot help other people until you help yourself.

4.1.4 Teaching While You Learn

Finding opportunities to teach others while you learn new programming skills can be a great way to solidify your understanding of the material and give back to the community. Let's say you're working on a new project, and you are setting up a local development environment. You continue to get the same networking errors and you try to implement fixes from Stack Overflow, but nothing is working. Or you are pair programming with a Junior Developer and neither of you can figure out how to fix an error and what is causing it. When you do figure out the solution, you should create content to help others who are facing the same errors. You can write a blog post, make a tutorial video, or give a talk detailing both the problem and the solution.

One way to find opportunities to teach is to look for meetups or user groups online or in your area that focus on the programming languages or technologies you're interested in learning. These groups often need speakers to give presentations or lead workshops on specific topics, and as a member, you can volunteer to lead a session or give a talk about the errors you have experienced and how you fixed them. These are my favorite types of presentations because they are engaging and can gain an emotional response from the audience if they have also experienced the same errors.

You can also create your own opportunities by offering to teach a class or workshop online or at a local college, university, or community center. This can be a great way to give back to your community and share your knowledge with others. If you can’t find a user group, you can always create one! Don’t be afraid to create your own community as being the creator of a community will teach you leadership skills. You have to organize your group, schedule events, and moderate discussions which are all great skills to have as a Lead Developer.

Another way to find opportunities is to look for mentorship programs or apprenticeships. Many companies and organizations have mentorship programs for developers looking to learn new skills and gain experience. As a mentor, you'll be responsible for guiding a more junior developer through the process of learning a new technology or skill set. The wonderful thing about being a mentor is that you have the opportunity to learn from your mentees as well. I have learned so much from my mentees over the years as they are studying the latest and greatest technologies that I hadn’t studied yet. This will help them reinforce their confidence when they’re able to teach new skills to an experienced Lead Developer.

Additionally, you can also leverage the power of social media and other online platforms. Creating a blog, a YouTube channel, a podcast, or a Twitch channel about your own learning experience can help you attract followers and students that want to learn from your experience and methods. And remember, you do not always have to share technical skills that you have learned, you should also share leadership skills. Telling stories about situations you found yourself in, how you handled them, and how you would handle them now is a great way to connect with your audience and get feedback on your leadership skills.

Teaching while learning new skills can be a mutually beneficial experience. It allows you to deepen your understanding of the material while giving back to the community. There are many ways to find opportunities to teach, including volunteering to speak at meetups or user groups, participating in mentorship programs or apprenticeships, creating online content, or teaching a class or workshop at a local institution. As you prepare your teaching materials, you may also uncover other teachable moments that you did not think of before. All of this will go a long way to ensuring that you are consistently improving your skills and you are set up for success in your career as a Lead Developer.

4.2 Applying Your Skills

Working on personal projects, community projects, and creating prototypes are valuable skills for any Lead Developer to have in their career. Prototypes are early versions of a product or feature that are used to test and validate design concepts. They can take many forms, from rough sketches on paper to highly detailed mockups built with software. Creating a prototype is a great way to explore different design options and get feedback on your ideas. There are many tools and techniques you can use to build prototypes, including wireframing, mockups, and even coding.

When it comes to learning new skills, working on community projects can be extremely helpful. They allow developers to experiment with new technologies and techniques in a supportive environment, and to get a sense of how something will work before committing to a full implementation. Additionally, working on a personal or community project can help to develop problem-solving skills, as developers learn to identify and work through challenges that arise. Prototypes allow you to test and validate your ideas quickly and cheaply, and they also help you to communicate your vision to stakeholders and team members. When you’re starting a new project, a prototype can be a great way to explore different design options and identify potential issues early on. This can save a lot of time and resources in the long run, as changes are much more difficult and expensive to make once a project is well underway. A good example of a simple prototype that you can build for any programming language or framework is a CRUD application which stands for Create, Read, Update, and Delete.

Figure 4.1 Example CRUD Prototype

CRUD APPLICATIONS

C

Create

Developing a new project from scratch will help you learn how to go from conceptualization to building applications. Understanding how to build applications from scratch is a skill that helps developers understand application architecture and software design.

R

Read

Understanding how to read data from an API or database is one of the fundamental skills that every developer needs. Reading data is the foundation of every application to support user interaction, data analysis, and reporting.

U

Update

Learning how to update data and send it back to the user interface is a skill that developers need to build features in a software or web application. It teaches developers how to perform business logic and send data back using a secure API or database connection.

D

Delete

Creating a secure method for deleting data is a necessary skill for developers building applications to ensure that the correct data is deleted while current data is preserved. This teaches developers how to differentiate data that should be backed up vs. sensitive data that should be deleted forever.

In addition to helping with the development process, prototypes can also be used to demonstrate the value of a project to stakeholders. By providing a tangible example of what the final product will look like, it can be much easier to secure funding, gather feedback, and build buy-in from key decision-makers.

Prototypes can also be a great tool for team collaboration and communication. They can help developers clearly communicate their ideas to the rest of the team and gather feedback and input from other team members. This can lead to a more efficient and effective development process, as everyone is on the same page and working towards the same goal. You should take the time to learn and practice creating prototypes, it will pay off in the long run.

4.2.1 Developing Personal Projects

Working on personal projects can be incredibly beneficial for a career in development. Not only do they allow you to gain hands-on experience with various technologies and tools, but they also demonstrate your passion and commitment to your craft. Personal projects allow you to experiment with new technologies and tools that you may not have the opportunity to work with in your day-to-day job. This can help you stay current with the latest industry trends and improve your skills.

The following list includes some of the benefits of working on a personal project:

- Building a portfolio: Personal projects can be used to create a portfolio that showcases your abilities and skills to potential employers. Having a portfolio of real-world projects can be a great way to stand out from other candidates and demonstrate your expertise.
- Improving problem-solving skills: Personal projects can be a great way to challenge yourself and improve your problem-solving skills. Building something from scratch requires you to think critically, identify and fix bugs, and come up with creative solutions to problems.
- Increasing confidence: Working on personal projects can help you build

confidence in your abilities as a developer. Completing a project, no matter how small is a great feeling and can give you the motivation you need to tackle bigger and more complex projects.

Personal projects are a great way for developers to gain hands-on experience, build a portfolio, and showcase their skills to potential employers. But it can be difficult to come up with ideas for personal projects. The following table lists a few examples of personal projects that developers can work on.

Table 4.2 Examples of Personal Projects for Developers

Project	Benefits
Build a website or web application	Gain experience with front-end technologies such as HTML, CSS, and JavaScript, and web frameworks such as React or AngularJS.
Create a mobile app	Gain experience with mobile development technologies such as Swift for iOS and Kotlin for Android. Developers can also use cross-platform development tools such as React Native or Flutter to build apps that run on both iOS and Android.
Create a game	Learn game development technologies such as Unity or Unreal Engine as well as game development frameworks such as Pygame or Godot to create 2D games.
Create a personal	Learn natural language processing and machine learning technologies using platforms such as Dialogflow or Rasa to create an AI-powered

assistant	personal assistant that can help with tasks such as scheduling, reminders, and more.
Create a chatbot	Gain experience with natural language processing and machine learning technologies using platforms such as Dialogflow or Botkit to create a chatbot that can interact with users in natural language.

These are just a few examples of personal projects that developers can work on. The possibilities are endless, and the key is to choose a project that aligns with your interests and skills. However, it is also a good idea to work on projects that are outside of your comfort zone to broaden your skill set. To come up with ideas for personal projects, you can explore current trends or participate in coding challenges. You can also create something that solves a problem you or someone you know is having. For example, one of the featured contributors for this book, Jamie Maguire, works on his own personal projects using the Twitter API. He was interested in creating a robust tool to manage his account, so he learned how to work with the tools provided by Twitter. By working on a personal project, developers can gain valuable experience, improve their skills, and build a portfolio of work that can help them advance their careers. So, take some time to explore new technologies and tools and start working on a personal project today!

4.2.2 Working on Community Projects

Working on community projects can be incredibly beneficial for developers for a number of reasons. For one, it allows you to collaborate with other like-minded individuals, which can lead to the creation of innovative and high-quality projects. Additionally, participating in community projects can help you build a strong network of professional contacts, which can prove invaluable in your career.

But how do you get started? The first step is to find a project that aligns with your interests and skills. There are many different types of open-source projects, from software development to documentation and translation. Browse through popular open-source platforms like GitHub, SourceForge, and GitLab to find projects that catch your eye. You can also become a contributor to many libraries and frameworks such as React. You can address any problems that you find and suggest new features.

Before you reach out to contribute to a project, it's a good idea to read through the documentation to get a sense of the project's goals and how it is organized. This will help you understand how you can contribute and what skills are needed. Once you've found a project that you're interested in, reach out to the project's maintainers or community members. Introduce yourself and explain why you're interested in the project. Be sure to mention any relevant experience or skills you have that could be helpful to the project.

When you're first getting started, it's best to start with small, manageable tasks. This will help you get familiar with the project's codebase and workflow, and it will give you a chance to build a relationship with the other members of the community. Open-source projects are built on collaboration, so it's important to communicate effectively with other members of the community. Be sure to ask questions when you need help and provide updates on your progress.

One of the most popular tech communities for developers is GitHub, a platform that allows developers to share, collaborate on, and review code. It is widely used by developers of all skill levels and offers a wide variety of projects to work on, including those related to open-source software, machine learning, and data science. GitHub also provides developers with a range of tools to help them collaborate more effectively, including code reviews, issue tracking, and project management.

Another popular community for developers is Stack Overflow, a platform where developers can ask, and answer questions related to programming. Stack Overflow is a great resource for developers looking for help with specific programming problems, and it also provides a wealth of information on best practices and common pitfalls. Additionally, Stack Overflow has a reputation system that rewards users who contribute high-quality answers and questions, which can help to build your reputation as a developer.

A third popular community for developers is the Linux community, which is dedicated to the development and promotion of the Linux operating system. Linux is an open-source operating system that is widely used in servers, desktops, and mobile devices. The Linux community is made up of a diverse group of developers, system administrators, and users, who work together to create and maintain the operating system and they are always looking for contributors.

Remember, getting involved in open-source projects is not just about writing code, it's about building a community. By contributing to a project, you're helping to improve the project and you're also building a network of like-minded developers who can help you grow both personally and professionally. So don't hesitate to take the first step and reach out to get involved today!

4.2.3 Creating Prototypes

As a developer, one of the most important skills you can have is the ability to quickly learn new technologies and programming languages. One of the best ways to do this is by creating prototypes that can be adapted to a variety of different situations. One of the main benefits of creating prototypes is that they allow you to build a solid foundation of knowledge in a particular area. For example, if you are learning a new programming language, you can start by creating a simple project that demonstrates the basic syntax and structure of the language, such as a CRUD application. As you become more proficient, you can then adapt the project to include more advanced features and functionality. This process helps you to gradually build your skills and become more confident in your ability to work with new technology.

Something that helped me learn a new language was recreating code that was written in one language into another language. I learned C# this way because when C# was becoming popular, I was a VB.NET programmer. I used tools including a VB to C# code converter to help me learn the differences between the two languages. I also read blogs and textbooks to further hone my skills to become proficient in C#.

Another benefit of creating prototypes is that they can serve as a reference point for future projects. For example, if you are working on a project that requires the use of a specific library or framework, you can refer to a previous project where you used the same technology. This can save you a lot of time and effort, as you will already be familiar with the basics of how the technology works. Additionally, you can also reuse the code that you created for the previous project, which can help to speed up development time and reduce the risk of errors.

Creating reusable projects can also be a great way to demonstrate your skills to potential employers or clients. If you are a Lead Developer looking for a new job, you can create a portfolio of reusable projects that demonstrate your abilities in different programming languages or technologies. This can be a powerful tool for landing new jobs and building a successful career.

One example of a prototype is creating a library or framework for a specific language or technology. For example, a developer who is interested in learning Python can create a library for handling CSV files that can be used in multiple projects. This not only helps you improve your Python skills, but it also saves time for future projects that require CSV file handling.

Another example is creating a boilerplate or starter project for a specific use case. If you are interested in learning React, you can create a starter project that includes a basic file structure, a few commonly used libraries, and some example code. This can save a lot of time for developers who are just getting started with React and want to quickly set up a new project.

Creating prototypes also helps to improve code quality and maintainability. By writing code that can be reused across multiple projects, developers can ensure that the code is well-structured, easy to understand, and easy to maintain. Additionally, prototypes can be shared with other developers, which can lead to collaboration and feedback that can improve the overall quality of the code. You can also make your prototype an open-source project to get community feedback and contributions. That way, you are giving back to the community and allowing others to learn new skills.

Prototypes are a great way for developers to learn new skills and improve development time. They allow developers to apply their knowledge to real-world problems, provide a sense of accomplishment and satisfaction and improve code quality and maintainability. So, next time you are looking to improve your skills and save time, consider creating a prototype.

4.3 Overcoming Learning Blockers

Lead developers often face a variety of challenges when it comes to learning new skills and technologies. You may encounter learning blockers which are a common and natural part of the learning process. They occur when our brain needs a break from information overload or when we hit a snag in our understanding. This doesn't mean that we're not capable or intelligent, but rather our brain needs time to process and make connections. Taking a step back, trying a different approach, or seeking help can often break through these blocks and help us continue our learning journey with renewed energy and understanding.

4.3.1 Taking Breaks

As developers, we often find ourselves in the middle of a difficult problem or stuck trying to learn a new concept. It can be frustrating and demotivating, and it's easy to get caught up in the feeling that we should just push through and work harder., Taking a break is one of the best things you can do when you encounter a learning blocker.

Taking a break can help you see the problem from a different perspective. When you've been staring at the same piece of code or trying to understand the same concept for hours, it can be hard to see the bigger picture. Stepping away from the problem for a bit can help you come back to it with fresh eyes and a renewed perspective.

Just like your body needs rest to function properly, your brain needs a break too. When you're feeling mentally exhausted, it can be hard to focus and absorb new information. Giving your brain some time to rest and recharge can help you come back to the problem feeling refreshed and ready to tackle it again.

Taking a break can help you avoid burnout. When we're feeling blocked and frustrated, it's easy to fall into the trap of working longer and harder in an effort to make progress. You may find yourself working on the same thing over and over but never getting anywhere. This approach can lead to burnout, which can make it even harder to make progress in the long run. By taking a break and giving yourself permission to step away from the problem, you're allowing yourself the time and space you need to come back to it with renewed energy and focus.

So, when should you take a break? The answer will vary depending on the individual and their specific needs, but a good rule of thumb is to take a break every hour or so. This will give your brain a chance to rest and recharge, and help you stay focused and productive when you return to your studies. During your breaks, it's important to engage in activities that are different from studying. This could be something as simple as taking a walk outside, stretching your muscles, or chatting with a friend. Doing something that is enjoyable and diverting your focus from your studies helps you to come back to your studies with renewed energy.

Another important thing to consider is the time of day. If you're a morning person, you may find that you're most productive in the early hours of the day and need more frequent breaks. On the other hand, if you're a night owl, you may be able to push through for longer periods of time before needing a break.

Ultimately, the key to successful studying is finding a balance between hard work and rest. By taking regular breaks, you can ensure that you're getting the most out of your study sessions and avoid feeling overwhelmed or burnt out. So, the next time you're hitting the books, make sure to take a break every now and then. Your mind and body will thank you!

4.3.2 Setting Attainable Goals

As a developer, you're constantly learning new skills and technologies to stay ahead in the field. But with so many options out there, it can be overwhelming to decide what to focus on next. That's why setting attainable learning goals is so important. When setting your goals, it's important to be realistic about what you can accomplish in a set amount of time. If you're just starting out, for example, it's unlikely that you'll be able to master a complex framework like React in a week. Instead, start with something smaller, like learning the basics of JavaScript or a particular library.

Another important aspect of setting goals is to make sure they align with your overall career aspirations. If you're working towards a specific job or role, make sure your goals are in line with the skills required for that position. For example, if you're interested in becoming a full-stack developer, make sure you're learning both front-end and back-end technologies.

You should also make sure to set both short-term and long-term goals. Short-term goals can be things like learning a specific language or library, while long-term goals might be more general, like becoming an expert in a particular field or earning a specific certification. Having a mix of both will help you stay motivated while also giving you a sense of direction in the long term.

When you set learning goals for yourself, you're able to stay on track and make sure that you're focusing on the skills that are most important to you. This can help you to achieve your career aspirations, whether you're looking to move into a new role, advance in your current position, or simply stay up to date with the latest technologies. Having attainable learning goals also helps to keep you motivated and engaged. When you're working towards a specific goal, you're more likely to stay focused and make steady progress. This can help you to avoid feeling overwhelmed or burned out, which is all too common in the fast-paced world of development.

Another benefit of setting learning goals is that it allows you to measure your progress. By setting specific and measurable goals, you can track your progress and see how far you've come. This can be incredibly motivating and help you to stay on track as you work towards your goals. A good way to stay on track is to break down your goals into smaller, more manageable tasks. For example, instead of setting a goal to "learn React," you might break it down into smaller tasks like "learn the basics of React," "build a simple app using React," "learn advanced features of React," etc. This will help make the goal more manageable and help you track your progress more easily.

A moonshot goal is a highly ambitious and transformative target that aims to make a significant positive impact. Having a moonshot goal is a great way to inspire creativity and innovation, push boundaries, and drive progress. By setting a moonshot goal, you are setting your sights high and challenging yourself to aim for something truly meaningful and impactful. Having a moonshot goal can also provide a sense of purpose, motivation, and direction in your personal or professional life. Just keep in mind that these are the end goals, and you should focus on achieving success with attainable goals that will lead you to your moonshot goals.

Table 4.3 Example Moonshot vs. Attainable Goals

Project	Moonshot Goal	Attainable Goals
Build an app	Build the next big social media app	· Learn the basics of React · Build a “Hello, world!” app · Learn advanced React skills · Develop user interactions
Create Artificial General Intelligence (AGI)	Create the next ChatGPT	· Learn the basics of AI · Build a simple AI algorithm capable of performing a specific task · Improve the accuracy and efficiency of the algorithm

		· Study cognitive psychology to create human intelligence models

Setting attainable learning goals is essential for any developer looking to improve their skills and advance their career. It helps you stay on track, stay motivated, and measure your progress. Make sure they align with your career aspirations, break them down into smaller tasks, and seek out resources and support. With a solid plan in place, you'll be well on your way to achieving your goals and becoming a better developer.

4.3.3 Seeking Help

As a lead developer, it's natural to want to be self-sufficient and to be able to handle any challenge that comes your way. But, let's face it, no one has all the answers, and even the most seasoned developers encounter learning blockers from time to time. In these moments, seeking help can be the key to overcoming those obstacles and moving forward with your learning journey.

Let's talk about why seeking help is so important. It can save you a lot of time and frustration. When you're stuck on a problem or don't know how to proceed, it can be easy to get bogged down and waste hours or even days trying to figure things out on your own. By seeking help, you can get a fresh perspective and some guidance that can help you make progress much more quickly. Seeking help allows you to learn from the experience of others and can expose you to new ways of thinking and working that you might not have discovered on your own.

So, how do you go about seeking help? One of the best ways is to reach out to more experienced colleagues or mentors. They can provide valuable insights and guidance that can help you overcome the obstacle you're facing. You should approach experienced colleagues or mentors with respect and a clear understanding of their needs. Before asking for help, take the time to research and try to solve the problem on your own. This will show that you are motivated and capable of tackling the issue at hand. When asking for help, you should be specific about the issue you are facing and provide any relevant background information. It also helps when you are open to feedback and willing to take constructive criticism. It's important to remember to be gracious and thank the colleague or mentor for their time and assistance.

Additionally, you can also look for help within the developer community. There are a variety of online forums and communities where developers come together to share knowledge and help each other out.

The following list includes developer communities you can use to ask for help:

- Stack Overflow: A popular Q&A platform for programmers to ask and answer technical questions.
- GitHub: A platform for developers to host and share their code, as well as collaborate on open-source projects. Many developers use GitHub's "Issues" feature to ask for help with specific problems.
- Reddit: The /r/learnprogramming and /r/programming subreddits are both popular communities where developers can ask for help and share their knowledge with others.
- Quora: A Q&A website where developers can ask and answer questions on a wide range of programming-related topics.
- Dedicated Communities: Many programming languages and frameworks have dedicated communities where developers can ask for help and share their knowledge with others. Examples include the Docker community, the React community, and the Angular community.
- Meetup: Many cities have meetup groups for developers where they can meet, network, and ask for help from others in their field.

When you reach out for help, it's important to be specific about what you're looking for. Be clear about what you're trying to accomplish and what you need help with. This will make it easier for the person you're asking to give you the guidance you need. It's also a good idea to be open to feedback and to be willing to try different approaches. Remember, seeking help is not a sign of weakness, it's a sign of strength. It shows that you're willing to learn and grow and that you're not afraid to ask for help when you need it.

Seeking help is an essential part of the learning process for Lead Developers. It can save you time, expose you to new ways of thinking, and help you overcome obstacles that might otherwise hold you back. By reaching out to more experienced colleagues or mentors, or by finding help within the developer community, you can get the guidance you need to keep moving forward with your learning journey. So, next time you're faced with a learning blocker, don't be afraid to ask for help. It could be the key to unlocking your full potential.

4.4 Case Study

Scott Hanselman has been a developer for 30 years and has been blogging at hanselman.com for 20 years! He works in Open Source on .NET and the Azure Cloud for Microsoft out of his home office in Portland, Oregon. Scott has been podcasting for over 15 years and he has produced 800 episodes of the Hanselminutes podcast and 700 episodes of the Azure Friday podcast. He's written a number of technical books and spoken in person to over one million developers worldwide! He's also on TikTok, which was very likely a huge mistake. In this case study, Scott shares his thoughts on learning any developer skill.

4.4.1 How can a Developer find community projects and how can they get involved?

One of the things that I struggle with when I’m coaching people is the unspoken vibe when you meet someone who you know is going places. Like if you’re at Chipotle and you’ve seen someone a few times and you notice them. There’s just something about them and you think to yourself that they have the potential to do more. We think these things, but we never say them. But now that I’m an old man with big Dad energy, I do say these things! I’ll talk to them and say something encouraging and watch them light up.

There are so many Developers who are going places and they are so desperate to get there, and it shows. They do things like send me random emails saying, “let me pick your brain” or they spam my LinkedIn inbox. It doesn’t come from a bad place; they are clear self-starters who want to achieve success, but they are going about it in the wrong way. Starting with negative interactions doesn’t make me want to help you. On the other hand, there are these other people that you just have a feeling about and you want to help them and offer them mentorship. So, how do we prepare people for these moments? A lot of it is just plain luck.

When you’re reaching out to someone to get involved in the community and find community projects, make sure you are prepared first. Have you searched for community projects on your own? Have you looked for relevant social media accounts or interacted with anyone else in the community? You need to do these things first to be prepared for when the opportunity becomes available. Luck is opportunity plus being prepared. I could point you to a project that you can contribute to, but you must show up and do the work.

There are soft skills involved in finding community projects and interacting with other developers. You must warm up to people and not try to fast forward the process of getting to know someone. Offer your help first before asking for help. With community projects, a lot of the tasks can be grunt work, but it is still something that you will learn from. Having a positive attitude and making helpful comments will establish you as someone who is a good community contributor. Think about the humans behind the project. What do they need? What problems are they trying to solve? How can you help them? Offer to do the dirty work. That’s how you build trust.

4.4.2 How can a Developer find the help they need to overcome learning blockers?

I think that it is common for Developers to get overwhelmed with all the things that we must learn in tech. This happens to people who are new in their careers and people like me who've been doing this for many years. What do you do when you're completely overwhelmed? There’s a concept called rubber ducking where you get a rubber duck and explain the problem to them. Say you’re trying to reverse sort a linked list and then you just explain it to the duck and in the process of getting it out of your mouth back into your ears into your brain you can solve the problem. Explaining it to even a non-technical partner, spouse, or roommate is also something you can do.

You should wake up and be mentally present and intentional. I’m a big believer in deliberate practice. If you’re searching for an error and staring at the screen looking for the answers you need and not finding them, it’s time to talk to somebody. If you can’t find somebody, then talk to a rubber duck. Put it on top of your monitor and when you have a problem, explain it to the duck. You can also go for a walk and talk to yourself. That’s also rubber duck debugging. You don’t need an actual duck.

Overcoming learning blockers is a waking-up process. If you don’t understand the answers you’re getting from Google or Stack Overflow, you need to take a step back. If you’re trying to copy and paste code you found on Stack Overflow for unit testing and you don’t know what async means but you are trying to do async programming, then you need to step back and learn about async programming first. I’m a big fan of going one level below your comfort zone. You don’t have to go all the way down to assembly, but it does help if you understand the overall concepts that you are implementing in your code.

4.4.3 What would you tell a Developer who feels inadequate because they are not working on as many community projects as other Developers in their network?

I am inadequate compared to Rihanna and Beyoncé. I can give you a list of all the people I am inadequate when compared to. The only person you should be comparing yourself to is yourself. You don’t know what’s going on in other people’s lives. Do you have kids? Then you won’t have as much time to work on community projects as some college kid who lives with their parents. You have different life experiences and responsibilities. We need to stop apologizing for what we are not doing and start thinking about what we are doing and what we can be doing and only comparing ourselves to our past selves.

You should also think about how you value your time. I like to play video games so in my off hours, I’m playing God of War. But what are my colleagues doing for those extra 3-4 hours a day? Some of them spend that time coding and that can make me feel bad. But coding helps them relax and unwind so their experience is different. I’m not going to apologize for playing video games, it’s what helps me relax. We are all on our own journey and should not apologize for what we are not doing.

4.4.4 How do you like to learn new skills?

I like to learn in public but that is a privilege because I am less likely to have mean people comment on my work. I think if you’re learning in public, you need to be able to ignore those comments and delete those unwanted DMs and just move on. My favorite way to learn in public is through trusted mentoring groups. One-on-one mentoring can be threatening but if you can find a group of three to four, you can keep each other accountable. Codenewbies.org is a great community and they are very welcoming. You can create a mini study group where everyone is on the same level so you can collaborate with people who are similar to you without feeling like you are at the bottom of the food chain. Everyone is equal in their skills and accountability, and everyone has the opportunity to be both the mentor and the mentee.

4.5 Summary

- Assessing your preferred content types is important to determine which formats are most effective for you.
- Experimenting with different content formats, practicing what you've learned, and reflecting on your own learning style can help you gain insight into which types of content will be the most effective for you in the future.
- Boosting your memory while learning new skills can help you retain information more effectively and become a more efficient and effective developer. This can include practicing active recall, testing yourself, teaching the material to someone else, and using mnemonic devices.
- Creating prototypes can save time and resources by allowing you to test and validate ideas early on and communicate your vision to stakeholders and team members.
- Prototypes can be used to learn new skills and demonstrate the value of a project to stakeholders to improve collaboration and communication within a team.

- Personal projects can be beneficial for a career in development by providing hands-on experience with new technologies, building a portfolio, networking opportunities, improving problem-solving skills, and increasing confidence.
- Overcoming learning blockers can be achieved by taking regular breaks, setting clear and attainable goals, and seeking help and guidance from more experienced colleagues or mentors.
- Taking breaks can help overcome learning blockers by providing a fresh perspective, recharging the brain, and avoiding burnout.
- Setting attainable goals can help provide direction and focus, a sense of accomplishment when met, and helps to navigate the learning process effectively.

5 Writing Technical Documentation

This chapter covers

- Reviewing workflow and project management practices when there is proper technical documentation vs. little to no technical documentation
- Defining technical debt and how to avoid it with technical documentation
- Using technical documentation to onboard new developers so that they get up to speed quickly
- Listing best practices in structuring documentation and how to apply them
- Understand how to write an outline for technical documentation
- Using a style guide and writing documentation to get right to the point
- Managing technical documentation including keeping it updated

Lead developers must learn how to write and manage technical documentation because it helps to clearly communicate the design and functionality of a software system to both internal and external stakeholders. This includes other developers on the team, as well as project managers, QA engineers, and other non-technical team members.

Technical documentation also helps to ensure that a software application is easy to maintain and update over time. Clear documentation can make it easier for new developers to understand the codebase and quickly become productive. It also can help to prevent errors and confusion by providing a clear and accurate reference for how the system is supposed to work. Additionally, technical documentation can be used as a training tool for new team members, and it can also be used to create user manuals and other forms of end-user documentation.

Technical documentation is important for communication, maintainability, and clarity. It helps others to understand the system, prevents errors and confusion, and can be used during the onboarding process for new team members. When you invest time into creating and maintaining technical documentation you will improve the developer workflow. You will also provide project teams with a centralized location to reference when they have questions. When there is turnover on your team, you will not lose important knowledge as it is already documented. With clear and concise technical documentation, your life as a Lead Developer is made easier with proper organization and communication.

5.1 Setting the Team Up for Success

When there is proper technical documentation, developers have a clear understanding of the codebase and how it should be maintained. As a result, the codebase is more consistent, easy to read, and maintainable. Additionally, developers can quickly find the information they need, which can help to reduce the amount of time spent on resolving issues or debugging code. Furthermore, proper technical documentation also helps to ensure that the decisions are not lost over time, which can help to improve the long-term maintainability of the codebase.

Table 5.1 Impact of Proper Technical Documentation on the Development Workflow

With Proper Technical Documentation	Without Proper Technical Documentation
Developers have a clear understanding of how the code works and how to use it.	Developers are left to figure things out on their own, which can lead to confusion and mistakes.
Developers can easily find the information they need to complete their tasks.	Developers may have to spend a lot of time searching for information and experimenting to figure out how things work.
Developers can quickly identify and	Developers may have a harder time

fix bugs and make changes to the code.	identifying and fixing bugs and making changes to the code.
Developers can easily collaborate with other team members.	Developers may have difficulty working with other team members because they are not on the same page.
Developers can easily onboard new team members.	Developers may have difficulty onboarding new team members because they lack the information they need to get started.

When there is little to no documentation, developers may have a hard time understanding the codebase and how it should be maintained. This can lead to a lack of consistency in the codebase, making it difficult to read and keep updated. Additionally, developers may have a hard time finding the information they need, which can lead to delays or mistakes. Without proper documentation, decisions may be lost over time, which can lead to a higher level of technical debt and increased maintenance costs.

5.1.1 Documenting Everything

As a Lead Developer, one of the most important things you can do to set your team up for success is to document everything having to do with developer processes. This includes documentation on coding standards, development workflow, and deployment processes. There are several collaboration tools that you can use to manage your documentation including Confluence, Microsoft Word, Google Docs, or GitHub Wiki pages.

Developing and maintaining coding standards is an essential aspect of any software development project. It ensures that the codebase is consistent, easy to read, and maintainable. Coding standards serve as a guide for developers to follow when writing code, and they also provide a consistent reference for code review and maintenance. Some companies have coding standards at the organizational level which must be reviewed and updated frequently. Establishing and documenting coding standards can also help to prevent common coding mistakes and improve the overall quality of the codebase.

It is important to ensure that the coding standards are easily accessible to all team members as a part of your technical documentation. This can be accomplished by making the document available on a company intranet or shared drive, or by including the standards in the team's code review process. You should also keep the coding standards up to date. As new technologies and best practices emerge, you should review the standards and make any necessary updates. Additionally, the team should be encouraged to provide feedback on the standards, and any issues or concerns should be addressed in a timely manner.

Documenting the development workflow is an essential part of ensuring that your team is working efficiently and effectively. By creating clear and detailed documentation of the development process, you can help to streamline the workflow and improve communication among team members. To start, it's important to gather input from the team on their current workflow and identify any areas that may be causing delays or confusion. Once you have a clear understanding of the existing workflow, you can begin to document it in a step-by-step format, including information on how to handle bugs and issues that arise, how to review and merge code, and how to test and deploy code.

An example of a development workflow may include the following steps:

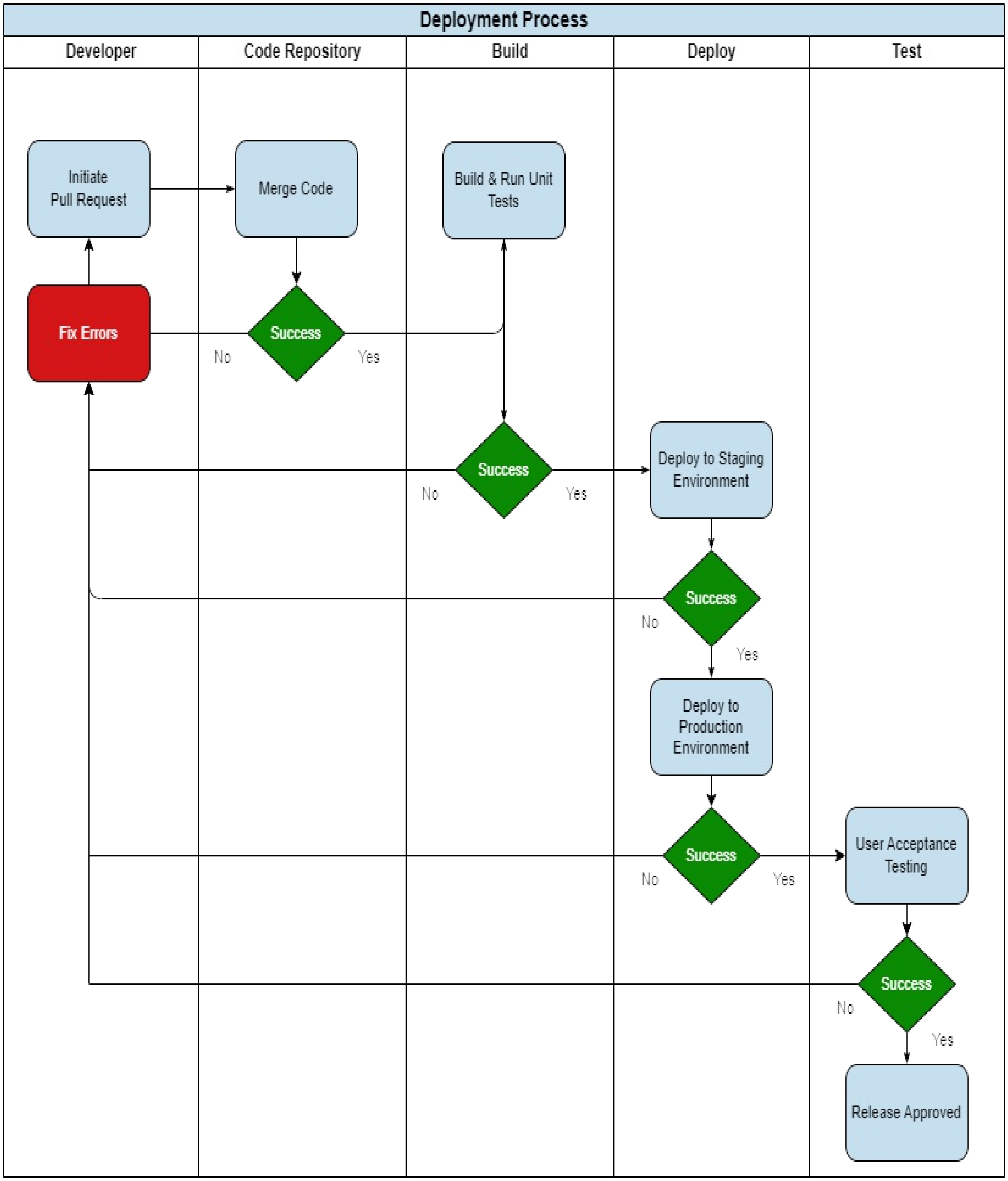
- Planning: Team members gather to discuss new features or updates for the project and create a plan for implementation.
- Development: Team members work on implementing the new features or updates, following established coding standards and best practices.
- Code Review: Team members review each other's code and provide feedback, ensuring that the code is of high quality and adheres to the established coding standards.
- Testing: Team members test the new features or updates to ensure that they are functioning correctly and fix any bugs that are found.
- Deployment: Team members deploy new features or updates to a

staging environment for further testing and validation before being released to production.
- Maintenance: Team members continue to monitor and maintain the codebase, addressing any issues that arise and implementing any necessary updates.

Development workflows can vary depending on the specific project or organization and you need to find what works best for you. Documenting the workflow is important as it helps to ensure that everyone is aware of the different steps and the team is aligned in the same direction. Gitflow is a popular software development methodology that provides a development workflow for managing the Git version control system. The main branch contains the production-ready code, while the development branch is where all the new features are developed. You should document the process for naming feature and hotfix branches, pushing code to QA, and merging code into the main branch once the feature is complete. You can also document how to handle merge errors, cherry-picking code changes to commit, and deployment procedures.

Documenting the deployment process is an important step in ensuring that your team can quickly and efficiently deploy new features and updates to production. By creating clear and detailed documentation of the deployment process, you can help to reduce the risk of errors and improve communication among team members. To start, you should gather information on the existing deployment process, including the steps involved, the tools and technologies used, and any best practices that have been established. Once you have a clear understanding of the existing process, you should document it in a step-by-step format, including information on how to handle any issues or problems that may arise during deployment. It's also a good idea to include diagrams or flowcharts to help visualize the process. Effective documentation of the deployment process is key to ensuring that your team can deploy new features and updates quickly and efficiently. The following figure shows an example of documenting the deployment process.

Figure 5.1 Example Deployment Process Chart



Documentation not only helps new team members understand how things are done within your organization, but it also serves as a valuable reference for experienced developers when they encounter a problem or need to troubleshoot an issue. By providing clear and detailed documentation, you can help your team work more efficiently and effectively, which in turn will help the organization. Your team will be set up for success as there are transparent standards for everyone to follow. In addition, proper documentation can also help to reduce the risk of errors and improve communication within the team. When everyone is on the same page and understands the processes that are in place, it's easier to identify and correct any issues that may arise.

5.1.2 Managing Technical Debt

Technical debt is a term used to describe the cost of maintaining and updating code that has been developed in a hurry or without proper attention to quality. It's important to understand that technical debt is not always a bad thing, as it allows teams to quickly deliver features and updates to meet deadlines or respond to changing business needs. However, when not managed properly, technical debt can quickly accumulate and become a significant burden on a project. Think of technical debt as a loan, like a mortgage or a credit card debt. It allows you to achieve your goals faster, but it also means you will have to pay interest in the form of added complexity, increased maintenance costs, and longer development cycles. Just like financial debt, technical debt can be manageable if it is monitored and paid off over time. On average, developers waste 33% of their time managing technical debt. By writing proper documentation including coding standards and the deployment process, you will manage technical debt which helps to reduce time spent researching “unknowns”, re-implementing work that does not meet standards, or introducing new errors.

As a Lead Developer, it's important to be aware of technical debt and to take steps to manage it effectively. When you take the time to document your code and systems, you are effectively creating a roadmap for your team to follow. You are also creating a record of what has been done and how it has been done. This can be incredibly valuable when it comes time to make updates or changes, as you will have a clear understanding of what needs to be done and how to do it. This can help to reduce confusion and errors and can also make it easier for new team members to get up to speed quickly.

One way to effectively manage technical debt is to maintain a historical record of decisions made in the technical documentation. By keeping track of why certain decisions were made and what trade-offs were considered, you and your team will have a better understanding of the codebase and be able to make more informed decisions in the future.

For example, if a certain design pattern was chosen because it was the best fit for a specific use case, documenting that decision can help the team understand the reasoning behind it and make similar decisions in the future. Similarly, if a certain technology was chosen because it offered better performance, documenting that decision can help the team understand the reasoning behind it and make similar decisions in the future.

Including this historical record in the technical documentation also helps to ensure that the decisions are not lost over time as team members come and go. This means that anyone new to the project will have access to the context and understanding of the decisions made in the past, which will help them make better decisions going forward. Retaining this information and documenting it will help you reduce technical debt when there is turnover and you lose expertise on your team. There should be a knowledge transfer before a person leaves the team but that is not always the case so it is best to document everything. Preserving a historical record allows Lead Developers and the team to make more informed decisions, understand the codebase better, and ensure that the decisions are not lost over time. It is not only beneficial for the present but also for the future of the project.

5.1.3 Onboarding New Developers

As a Lead Developer, one of the most important responsibilities you have is ensuring that new developers are able to quickly and easily onboard to your team. One of the best ways to do this is by writing clear and comprehensive documentation. Proper technical documentation can help new developers understand the overall architecture of the project, as well as the specific details of how different components and systems work together. This can save a significant amount of time and reduce bugs, as new developers will not have to spend as much time trying to figure out how everything works on their own.

In addition, clear and well-organized technical documentation can also make it easier for new developers to identify and fix any bugs that they come across. This can be especially useful if the documentation includes detailed information about how different parts of the code are supposed to work, as well as any known issues or edge cases. Writing proper technical documentation is a simple but powerful tool that can help you as a Lead Developer to reduce the time it takes to onboard new developers and make the process smoother and more efficient for everyone involved.

The following list includes topics that should be included in developer onboarding documentation.

- Code of conduct: A list of the team's expectations for behavior and communication, including guidelines for working with others and resolving conflicts.
- Coding standards: Detailed information on the team's coding standards, including naming conventions, formatting, commenting, and best practices.
- Development workflow: A step-by-step guide to the team's development workflow, including information on how to handle bugs and issues, how to review and merge code, and how to test and deploy code.
- Deployment process: A step-by-step guide to the team's deployment process, including information on how to handle any issues or problems that may arise during deployment.
- Tools and technologies: A list of the tools and technologies used by the team, including any required software or libraries.
- Communication channels: Information on the team's communication channels, including email, chat, and video conferencing tools, and how to use them effectively.
- Resources and documentation: A list of useful resources and documentation for the team, including any internal wikis, documentation, or tutorials.
- Onboarding checklist: A list of tasks that new developers should complete as part of their onboarding, including training, code reviews, and other requirements.
- Technical debt management: Information on the team's approach to managing technical debt and how new developers can help to minimize

it.
- Mentorship program: Information on the team's mentorship program, including how new developers can find a mentor and what is expected of them.

It's important to note that the specific items on this list may vary depending on the organization or project, but these are some of the most common items that should be included in developer onboarding documentation. The most important thing is that the documentation should be complete, clear, and easy to understand, and it should be reviewed regularly to make sure that it stays up to date.

I have worked at many companies that lacked technical documentation and I took it upon myself to write documentation so that future developers would not struggle with onboarding as I did. Many organizations do not document the process for setting up your local development environment and this causes confusion and frustration. One time, it took me two whole weeks to reverse engineer a development environment configuration because there was no documentation and the previous developers who worked on the project had all left the company. When I went through the process over the two-week period, I wrote down step by step what I had to do to configure my environment and the errors I encountered along the way. When I hired new developers for my team, it took them one day to set up their local development environments versus two weeks.

Do not be afraid to invest time in this task and remember that keeping your documentation updated is a key aspect of reducing the time it takes to onboard new developers, as the project and team evolve over time. It is a good practice to have new developers edit the documentation as they go through the onboarding process so that it is always up to date. This has the added benefit of giving them experience in maintaining documentation so that they can contribute regularly. Lead Developers do not bear the full responsibility of maintaining documentation, it is a team effort. With proper documentation in place, you will be able to onboard new developers more efficiently and ensure that your entire team is working together effectively.

5.2 Structuring Documentation

Structuring technical documentation is extremely important for ensuring that the information is easy to understand and navigate for developers and other users. Well-structured documentation makes it easy for developers to find the information they need quickly and get up to speed with a new project or technology. This can save a lot of time and frustration and help to ensure that projects are completed on time and to a high standard.

When documentation is not structured, it can be difficult to find the information you need, which can lead to confusion and mistakes. This can be especially problematic when working on a large or complex project, where there may be a lot of different documentation to navigate. Poorly structured documentation can also make it difficult for new developers to get up to speed with a project, which can slow down the development process and lead to delays.

5.2.1 Chunking the Content

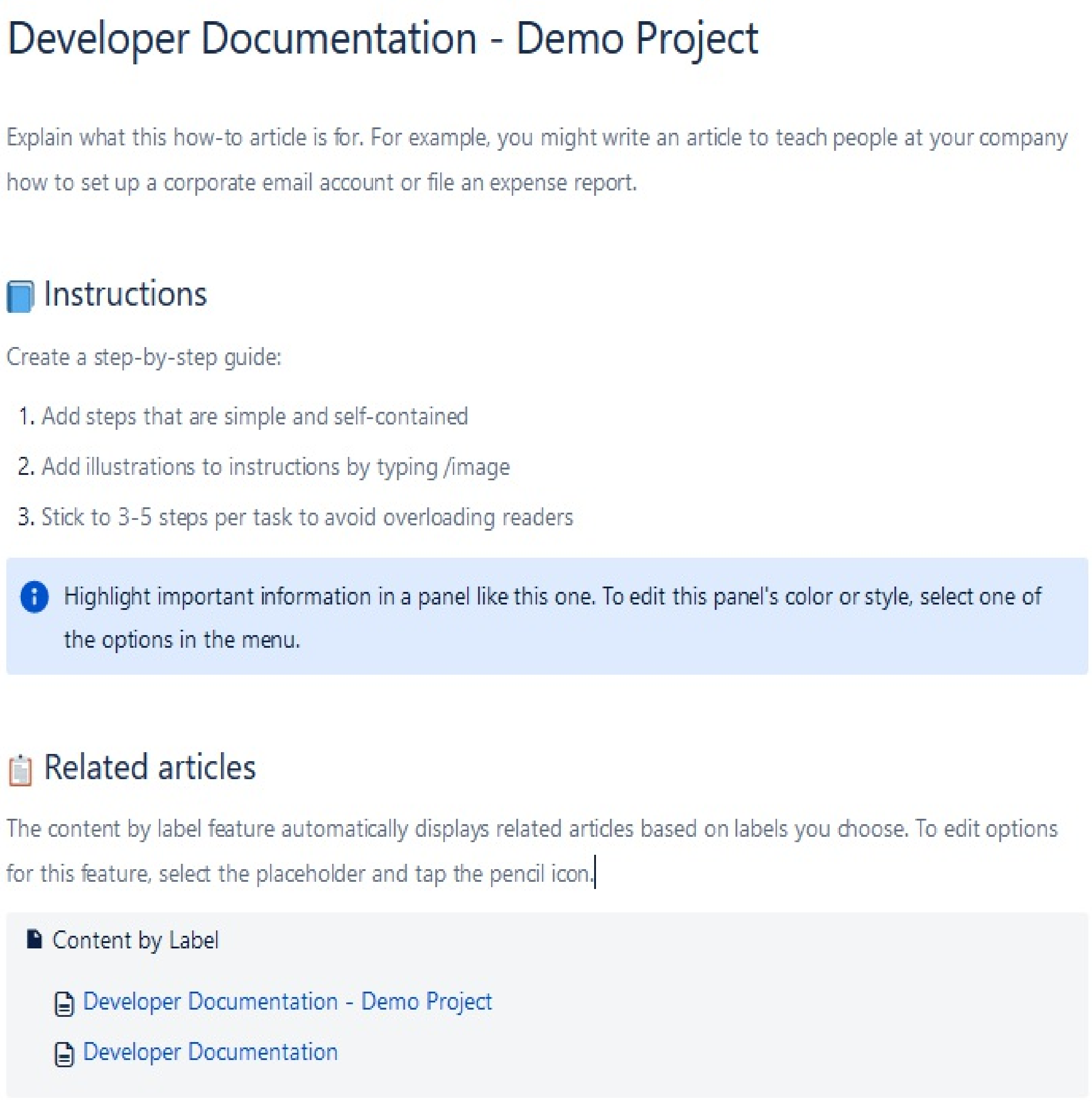
When it comes to technical documentation, it's important to keep in mind that your audience is made up of busy developers who want to quickly find the information they need. One of the best ways to make this happen is by breaking up your documentation into easily consumable chunks, or "chunking" as it's often called. Chunking is a technique for organizing information into small, manageable pieces so that it's easy for readers to find what they need and understand it quickly.

The following list includes tips for structuring your documentation using chunking:

- Headings and Subheadings: Use headings and subheadings to break up your document into sections. This makes it easy for readers to scan and find the information they're looking for.
- Bullet Points: Use bullet points and numbered lists to organize information into bite-sized chunks. This makes it easy for readers to quickly absorb the information and understand the main points.
- Images and Diagrams: Use images and diagrams to supplement your text. This helps to break up the text and make the document more visually appealing, which can make it easier for readers to understand.

- Examples and Case Studies: Use examples and case studies to illustrate key concepts. This helps to make the information more concrete and easier to understand.
- Simple Language: Keep your language simple and easy to understand. Avoid using jargon or technical terms that your readers may not be familiar with.

Figure 5.2 Example Developer Documentation in Confluence



Developer Documentation - Demo Project

Explain what this how-to article is for. For example, you might write an article to teach people at your company how to set up a corporate email account or file an expense report.

Instructions

Create a step-by-step guide:

1. Add steps that are simple and self-contained
2. Add illustrations to instructions by typing /image
3. Stick to 3-5 steps per task to avoid overloading readers

Highlight important information in a panel like this one. To edit this panel's color or style, select one of the options in the menu.

Related articles

The content by label feature automatically displays related articles based on labels you choose. To edit options for this feature, select the placeholder and tap the pencil icon.

Content by Label

Developer Documentation - Demo Project

Developer Documentation

By following these tips, you can help to ensure that your technical documentation is structured in a way that makes it easy for your readers to find the information they need and understand it quickly. Remember, the goal of technical documentation is to help developers be more productive and effective, so make it a priority to make it easy for them to use.

5.2.2 Using Visual Aids

Visual aids have a positive impact on technical documentation in several ways. They can help to make complex information more easily understandable by breaking it down into smaller, more manageable chunks. Visual aids can also make technical documentation more engaging and interesting to read, which can help to keep the reader's attention. They can help to convey the overall structure and flow of a system or program, making it easier for the reader to navigate and understand the codebase. Using visuals helps to break up text and make the documentation more visually appealing. The use of visual aids can greatly enhance the effectiveness of technical documentation and make it easier for developers to understand and work with the codebase.

Table 5.2 Examples of Visual Aids

Visual Aid	Purpose
Flowcharts	Flowcharts are a great way to visualize the flow of control in a program or system. They can be used to show the different paths a user can take through the system or explain the logic of a particular function or module.
Diagrams	Diagrams can be used to show the overall architecture of a system or to illustrate the relationships between different components. They can also be used to show the flow of data or control between different parts of the system.

Screenshots	Screenshots are a great way to show what a user interface looks like and how it works. They can also be used to show the output of a program or the results of a particular test.
Code Snippets	Code snippets are an essential part of technical documentation. They should be used to illustrate specific examples of how to use a particular function or class.

When I’m writing technical documentation, I rely heavily on screenshots and code snippets. As you are listing the steps to walk a developer through a task such as setting up their development environment or connecting to a test server, it helps to see a visual representation of what the developer sees on their screen. It also helps if you draw their attention to specific areas of the screen using callouts such as boxes and highlighting. This can be difficult to keep updated as developer tools are changing all the time so you should keep that in mind as you update your documentation. We will discuss this in detail later in this chapter.

5.2.3 Including an Introduction and Summary

When it comes to technical documentation, the introduction is crucial. It should provide an overview of the purpose of the document, as well as any background information that may be useful. The introduction should also include a list of any assumptions or prerequisites that the reader should be aware of before diving into the document. The following is an example of an introduction to developer documentation.

Welcome to the developer documentation for setting up a local development environment. This document will provide a step-by-step guide for setting up a local environment on your computer for development purposes. This guide assumes that you have a basic understanding of computer systems and software development. Additionally, it is important to note that the instructions provided in this document are for a Windows operating system.

After the introduction, you should include a summary of the document's contents. This summary should give the reader a high-level understanding of what the document covers and what they can expect to learn from it. This can be a great way to help the reader decide if the document is relevant to their needs and can also help them navigate the document more easily. The following is an example of a summary for developer documentation.

This document provides step-by-step instructions on how to configure a development environment on a local machine, including the installation of required software, tools, and libraries, as well as the configuration of relevant settings. It includes an overview of the development environment, explaining its purpose and how it differs from production environments. It then provides a list of prerequisites needed for the installation, such as the operating system and hardware requirements. The document covers the installation of software dependencies and tools, including compilers, code editors, and package managers. It explains how to set up environment variables and configure the necessary paths. Finally, this document includes a troubleshooting reference including common errors.

In addition, it is essential to include a conclusion summary at the end of the document including the key takeaways. The conclusion should summarize the main points of the document and provide any additional context or background information that may be useful. It's also good practice to include a section for further reading, this could be resources, relevant documents, or external links that can help the reader understand the subject matter more in-depth. There are different types of conclusions depending on the subject matter of the documentation. If the document covers deep technical detail or has iterated through several topics, a more complete conclusion is needed. The following list includes examples of a conclusion for a document including deep technical detail.

- Install a local web server, such as Apache or Nginx
- Install and configure a database management system, such as MySQL or PostgreSQL
- Install a programming language runtime, such as PHP or Python
- Install a version control system, such as Git
- Clone the project repository to your local machine
- Set up a virtual host or local domain for the project

- Configure environment variables and settings specific to your local development environment
- Run any necessary setup or installation scripts, such as database migrations
- Test the project to ensure it runs correctly in your local environment
- Related Documents and Resources
  - Contact List
  - Project Stakeholders
  - API Documentation

If you are writing technical documentation that is a reference vs. a how-to guide, you can write a more general conclusion or wrap-up where applicable. An example of this is API documentation that provides detailed information on the endpoints, parameters, and responses of the API, enabling developers to integrate it into their applications. This type of documentation is not a walkthrough of specific steps, so it is not appropriate to have a conclusion with bullet points. Instead, you should summarize the key takeaways in a paragraph. The following is an example of a conclusion for reference documentation.

This API reference document has provided a reference of the API's capabilities. We understand that developers may have additional questions or require further assistance during the integration process, and we encourage you to reach out to our support team for help. Additionally, we are continuously updating and improving our API to provide the best possible experience for our users, so we recommend regularly checking our documentation for updates.

By using introductions and summaries, as well as clear and concise writing, diagrams, code examples, and conclusion sections, developers can ensure that their documentation is easy to understand and navigate. With these best practices in mind, Lead Developers can help ensure that their documents are well structured to maximize their effectiveness.

5.3 Creating the Content

As a Lead Developer, you have a lot on your plate. With so much to do, it can be easy to feel overwhelmed when it comes time to create technical documentation. You may ask yourself where you should even begin. It helps to delegate content creation to your team and not take on everything yourself. Creating technical documentation can be a daunting task, but it doesn't have to be overwhelming. It's important to remind yourself that creating technical documentation is a necessary and important task, it not only helps the team to understand the project better but also serves as a reference point for future projects. Having complete documentation will help your team be more productive by providing them with a place to find the answers to their questions. That way, it frees up more of your time to focus on your leadership duties. By investing time and effort in creating technical documentation, you are helping to ensure the success of your project and your team.

5.3.1 Starting with an Outline

One of the most important steps in creating technical documentation is writing an outline. An outline is a roadmap for the documentation that helps to organize the information and ensure that it is presented in a logical and easy-to-follow manner. Before writing an outline, it's important to understand the purpose of the documentation. Who is the audience? What is the goal of the documentation? What information needs to be included? Understanding the purpose of the documentation will help to ensure that the outline is focused and relevant.

Once the purpose of the documentation is understood, the next step is to break the documentation down into sections. This can be done by identifying the main topics that need to be covered. For example, if the documentation is a user manual for a software application, the sections might include an introduction, getting started, using the application, troubleshooting, and references.

After the sections have been identified, it is important to create subheadings for each section. These subheadings will help to organize the information within each section and make it easier for developers to find the information they need. Under the "Getting Started" section, the subheadings might include "Installing the application," "Creating an account," and "Navigating the user interface."

Use bullet points and numbered lists: Bullet points and numbered lists are an effective way to present information in a clear and easy-to-follow manner. These can be used to organize information within each section and subheading. If you have a subheading named "Installing the application" subheading, the bullet points might include "Download the installation file," "Run the installation wizard," and "Activate the application."

Once the outline is complete, it's important to review and revise it. The outline should be clear and easy to follow, and it should include all the information that is necessary for the documentation. It's also a good idea to ask for feedback from other developers to ensure that the outline is clear and easy to understand.

An outline for a developer guide of an application could look like this:

- Introduction
  - Description of the application
  - Audience and purpose of the guide
- Summary
  - High-level overview
  - Key takeaways
- Getting started
  - Setting up the development environment
  - Cloning the repository
  - Installing dependencies
- Application architecture
  - High-level overview of the application structure
  - Detailed description of each component
- Building and deploying the application
  - Building for different environments
  - Deployment options
  - Best practices
- Troubleshooting and maintenance
  - Common issues and solutions
  - Upgrading the application
  - Maintenance tasks
- Conclusion

    - Main points of the document
    - Additional context or background information
- Appendices
    - References and external resources
    - Glossary of terms

Writing an outline for technical documentation can be a time-consuming task, but it is well worth the effort. A well-written outline will help to ensure that the documentation is clear, concise, and easy to understand, which will save developers time and frustration and help to ensure that projects are completed on time and to a high standard.

5.3.2 Writing Specific Instructions

Writing specific instructions for technical documentation can be a challenging task, but it is essential for ensuring that developers have the information they need to work efficiently and effectively. Specific instructions should be clear and concise, with no room for confusion. Use active voice, and be specific about what action should be taken, and what the expected outcome should be.

Writing step-by-step instructions is an effective way to ensure that developers know exactly what to do, and when to do it. Each step should be clearly numbered, and the instructions should be presented in a logical order. When giving instructions, be as specific as possible. Instead of saying "configure the settings," say "configure the settings in the 'config.txt' file." It's also a good idea to include screenshots or diagrams to help illustrate the instructions.

The following list includes an example of step-by-step instructions for a developer to pull a Docker image:

1. Open a command line interface (CLI) on your local machine.
2. Make sure you have Docker installed by running the command `docker -version.` If the command runs successfully, and you can see the version of Docker it means you have installed Docker.
3. Search for the image on the Docker Hub registry by running the

   command `docker search <image_name>`, where "image_name" is the name of the image you want to pull.

4. Once you have found the image you want to use, pull it down to your local machine by running the command `docker pull <image_name>`.
5. Verify that the image has been pulled successfully by running the command `docker images` and checking that the image is listed.

In the previous example, notice that I have the code for the developer to execute in a different font to clearly define the executable instructions vs. documentation text. This makes it easy for a developer to copy and paste the code into the CLI of their choice which saves time and reduces misspellings due to human error. You want to make sure that you write in full sentences and be as descriptive as possible.

Along with the specific instructions, it's a good idea to include tips and best practices to help developers work more efficiently and effectively. You could include tips on how to troubleshoot common issues or best practices for working with certain tools or technologies. Examples and use cases are effective ways to illustrate how the instructions and best practices should be used. These can help to clarify the instructions and make it easier for developers to understand how the instructions apply to their specific situations.

Before publishing the documentation, it's important to test the instructions to ensure that they are accurate and complete. Have other developers who are not familiar with the content follow the instructions and provide feedback to ensure that the instructions are accurate, clear, and easy to follow. That way, you know that the documentation that is available to the entire team is targeted to the right audience and will help to set them up for success.

5.3.3 Getting Right to the Point

When it comes to technical documentation, it's important to get right to the point. Not only is this approach more efficient, but it also helps to ensure that the instructions are clear and easy to follow. One of the biggest problems with poor technical documentation is that it can be overly wordy. This can make it difficult for readers to understand what is being asked of them and can even lead to confusion and frustration. Consider the following instructions for setting up a new software development environment.

Table 5.3 Overly Wordy Documentation vs. Clear and Concise Documentation

Overly Wordy Documentation	Clear and Concise Documentation
In order to properly set up your new development environment, you will need to first install the necessary software components. This includes the latest version of the programming language that you will be using, as well as any associated libraries and frameworks. Once these components have been installed, you will then need to configure your environment to properly connect to any necessary external dependencies. This may include setting up access to a remote database or configuring your environment to use a specific version control system. Finally, you will need to run any necessary tests to ensure that your environment is properly set up and configured.	To set up your new development environment, install the programming language, libraries, and frameworks. Then, connect to the remote database. Finally, run tests to ensure everything is working properly.

As you can see, the clear and concise version is much easier to understand and follow. It eliminates any unnecessary information and gets straight to the point. In contrast, the overly wordy version includes too much information. You must strike a balance between providing specific instructions and keeping them short enough that the content is easily consumable. I hate to break it to you, but most developers will skim your documentation looking for the key instructions that they need to get the job done. When you include too much information, this will slow the reader down and cause confusion.

When writing technical documentation, it's important for Lead Developers to be clear and concise. By getting right to the point and eliminating any unnecessary information, you can help to ensure that the instructions are easy to understand and follow. This will ultimately save time and lead to a more efficient and effective development process.

5.3.4 Using a Style Guide

When it comes to technical documentation, having a consistent and clear writing style is crucial for ensuring that your readers can easily understand and utilize the information you provide. One of the most effective ways to achieve this consistency is by using a style guide.

A style guide is a set of guidelines for writing and formatting documents. It typically includes rules for grammar, punctuation, and word usage, as well as recommendations for document structure and layout. By adhering to a style guide, writers can ensure that their documents are clear, consistent, and easy to read. Two of the most widely used style guides for technical documentation are those provided by Google (https://developers.google.com/style) and Microsoft (https://learn.microsoft.com/en-us/style-guide/welcome). Both guides are designed to help writers create clear and consistent documentation, but they do have some key differences.

The Google Technical Writing Style Guide is focused on helping writers create clear and concise documentation that is easy to understand. It includes rules for grammar, punctuation, and word usage, as well as recommendations for document structure and layout. The guide includes guidelines for writing for different audiences and for different types of documents.

The Microsoft Writing Style Guide is designed to help writers create documentation that is both clear and professional. It includes rules for grammar, punctuation, and word usage, as well as recommendations for document structure and layout. The guide also includes guidelines for writing for different audiences and for different types of documents, and it also puts emphasis on the importance of accessibility and inclusivity in language usage.

Both guides are comprehensive and provide a wealth of information for writers. However, the Google guide is more focused on clarity and concision, while the Microsoft guide puts more emphasis on professionalism and accessibility. If your organization focuses on Microsoft technologies and programming languages such as .NET, you will want to use the Microsoft style guide. Otherwise, use the Google style guide.

Using a style guide for technical documentation is essential for ensuring that your documents are clear, consistent, and easy to understand. Whichever style guide you choose, it will help you create high-quality, user-friendly documentation that your readers will appreciate.

5.4 Implementing a Documentation Maintenance Cycle

As a lead developer, you understand the importance of keeping your codebase up-to-date and maintainable. The same principle applies to technical documentation. A documentation maintenance cycle is a regular process of reviewing, updating, and improving technical documentation. This ensures that the documentation remains accurate, user-friendly, and easy to understand.

It's important to keep in mind that technical documentation is not a one-time task. As the codebase evolves, so too should the documentation. Keeping documentation up to date is crucial for maintaining the efficiency and productivity of your team. When documentation is outdated, developers may spend more time trying to understand the codebase and troubleshoot issues, which can lead to delays and mistakes. The documentation review process should include testing the documentation, soliciting feedback, and publishing regular updates.

It is essential to keep technical documentation up to date to ensure that developers have access to accurate and relevant information. Outdated documentation can lead to confusion, errors, and inefficiencies, as users may rely on inaccurate information. Additionally, technology solutions are continually evolving and improving, so keeping documentation up to date ensures that users have access to the latest features and functionalities.

Regularly updating technical documentation also demonstrates a commitment to quality, which can improve user satisfaction and promote customer loyalty.

Testing the Documentation

As a lead developer, you know how important it is to ensure that your code is thoroughly tested and working correctly before it is released to users. But did you know that it's just as important to test your technical documentation? Testing technical documentation helps ensure that it is accurate, easy to understand, and user-friendly. This is especially crucial for developers, as clear and accurate documentation can save a significant amount of time and effort when it comes to understanding and implementing new features or troubleshooting issues. So, how can you go about testing your technical documentation?

Make sure to test the documentation in the same environment that it will be used. This will help you identify any issues or inaccuracies that may arise when the documentation is being used in a real-world scenario. If issues are found, it is important to update the documentation to address these issues before it is published to the whole team. This will help to avoid miscommunication and confusion due to inaccurate instructions.

You should also ask anyone who tests your documentation to list any errors they encountered that were environmental but the instructions in the documentation were correct. These types of errors should be included at the end of the document in a section for “Known Issues” or “FAQs”. I have worked for many companies that attempted to provide every full-time developer with the same exact local development environment, but contractors had different environments because their equipment is not provided by the company. If this is the case in your organization, you should have at least one contractor test your documentation so that any error they may encounter is covered.

There are several automation tools available that can help you test your documentation, such as DocTest and Doxygen. These tools can automatically check for syntax errors, broken links, and other issues that may arise. They can save a lot of time if you use them before you even have anyone else test your document. You should also use a spelling and grammar tool such as Grammarly to ensure that your documentation is written properly. This is especially important when you work with people who do not speak English as a first language. I like to avoid using contractions and I keep in mind that the English language is hard to learn. Wherever possible, have non-native English speakers test your documentation to ensure that it is accessible to them.

Testing technical documentation is just as important as testing the code itself. By following these best practices, you can help ensure that your documentation is accurate, user-friendly, and easy to understand, which will save time and effort for developers and end-users alike. Remember that clear and accurate documentation is essential for successful development and user experience. So, always take the time to test and improve your technical documentation.

5.4.1 Getting Feedback

When it comes to technical documentation, feedback can be especially valuable. It can help you identify inaccuracies, areas that are unclear, or even areas where the documentation could be improved in terms of usability. By incorporating this feedback, you can help ensure that your documentation is accurate, user-friendly, and easy to understand. But how do you go about getting this feedback?

When asking for feedback, make sure you're asking the right questions. For example, you might ask "Is this documentation clear and easy to understand?" or "Did you find everything you were looking for in the documentation?". Some people need a prompt so that they know what type of feedback you are looking for and how best they can help. You can add these prompts using collaboration tools for your documentation such as Confluence, Microsoft Word, Google Docs, or GitHub.

You should also aim to get feedback from different stakeholders as they can provide different perspectives on your documentation. This may include feedback from other developers, QA testers, Project Managers, or end-users that can help you identify different issues. Each person has a different perspective based on their role so you may get varied opinions. Keep in mind that feedback, even if it may not be directly applicable, may be beneficial. If feedback from an end-user is not applicable to the technical documentation, it could still be useful for improving the user experience. In these cases, it's important to keep an open mind and consider the feedback in the context of the overall user experience. For example, if an end-user suggests a feature that isn't currently in the documentation, it could be an indication that the feature is missing from the codebase as well.

Getting feedback on your technical documentation is an important part of the development process. By asking the right questions and being open to feedback, you can help ensure that your documentation is accurate, user-friendly, and easy to understand. Remember that feedback is an ongoing process, and you should be prepared to iterate and improve as needed. Keep in mind that feedback can be beneficial even if it may not be directly applicable, it could be an indication of areas that need improvement in other aspects of the project.

5.4.2 Setting a Documentation Maintenance Window

Even the best documentation can become outdated or inaccurate if it is not properly maintained. That's where a documentation maintenance window comes in. A documentation maintenance window is a dedicated time during which the team can focus on updating, reviewing, and improving the project's documentation. This can include tasks such as fixing typos, updating code samples, and ensuring that the documentation is still accurate and relevant.

To form a successful documentation maintenance plan, you should schedule regular maintenance windows. Depending on the size and complexity of your project, it may make sense to schedule maintenance windows on a weekly or monthly basis. This will help ensure that the documentation stays up to date and that small issues are addressed before they become big problems. Most of the documentation that I’ve written was updated at least quarterly or after a large update. It helps to put a reminder in your team’s calendar so that you do not forget these important updates.

During the maintenance window, assign specific tasks to team members to ensure that all areas of the documentation are covered. One team member could focus on code samples, while another could focus on user documentation. As a Lead Developer, you should learn how to delegate responsibilities to others so that you are not overworked. This has the added benefit of empowering your team to take on new tasks and learn new skills to help them level up their careers.

You can also use a documentation tool like Docusaurus, Read the Docs, or Mkdocs to manage the documentation process. These tools make it easy to collaborate on documentation, track changes, and ensure that the documentation is always up to date. If those tools are not available to you, there are also good collaboration features in Microsoft Word and Google Docs which are commonly used in organizations.

It is a good idea to measure and track progress by keeping a record of the tasks that were completed during each maintenance window. This will help the team see what has been accomplished and identify areas where additional work is needed. I like to use a changelog or release notes to list all changes that have been made to technical documentation. You can even incorporate these changes into the release notes for a deployment.

Having a documentation maintenance window is an essential part of any software development project. By scheduling regular maintenance windows, assigning specific tasks to team members, and using documentation tools, you can ensure that your project's documentation stays accurate, relevant, and up to date. By following these tips, you can create a successful documentation maintenance plan that will help your team work more efficiently and effectively.

5.5 Case Study

Edidiong Asikpo is a Senior Developer Advocate based in Lagos, Nigeria. She is passionate about sharing her knowledge of DevOps through technical articles, videos, and social media. Edidiong has given over 100+ talks at tech events worldwide and continues to play a significant role in building developer communities in Africa. She is a Certified Kubernetes Application Developer and an Open-Source contributor. When she's not doing anything tech-related, she travels across the world, takes beautiful pictures, and analyzes movies. In this case study, Edidiong shares her experience with technical documentation and provides advice for writing and maintaining documentation.

5.5.1 How has writing proper documentation helped you set your team up for success?

When I worked as a Developer Advocate at Hashnode, we didn’t have product documentation. This led our users to be unaware of certain features or how to implement the ones they were aware of. To solve this, users often contacted us via Discord to ask questions and share their concerns. But as a small team, we could answer only so many questions while trying to complete our normal day-to-day tasks. So, we decided to create documentation to address this problem!

Creating this documentation didn’t just set up our team for success, it also gave our users and community members a one-stop place to find every piece of information they needed about Hashnode. They no longer had to wait for minutes or hours to get a response, and we (the team) no longer had to spend hours of our day responding to questions via Discord or retyping responses we had already shared with someone else the previous week. The documentation also eliminated confusion within the team - if you were unsure about the right answer to give a user about a question, you could head over to the documentation to read about it before reverting to the user.

Another example of how writing proper documentation helped me set up my team for success was when I worked as a Software Engineer in the Developer Relations team at Interswitch. For context, Interswitch is the biggest Fintech company in Africa, so we were over 300+ people in the company. One of the negative impacts of having so many people in a company is that sometimes two teams could be working on the same thing without the knowledge of the other team. For us, it was APIs. Several teams will build APIs that did the same thing or almost the same thing. This significantly affected our documentation and the developer experience of external developers integrating with our APIs.

We needed to fix this, and we knew having better documentation was the first step to take. So, we set up meetings with all Engineering managers, compiled all these APIs, merged the ones that were similar, and ensured every team took responsibility for updating their documentation when a change was made to the API. In addition to this, we also added a quick description of each API so that developers could quickly find the right API, and also code snippets of how to implement these APIs with several programming languages.

In summary, this updated documentation and structure of updating made our APIs more discoverable, and easier to keep up to date and increased the sales we made as a company.

5.5.2 Have you received feedback from other developers about your documentation? Did they suggest any improvements and what were the suggestions?

Yes, I have. During the process of updating the Interswitch API documentation, we would reach out to external and internal developers and ask them for feedback on what we could do or improve. One of the feedback items they shared was the outdatedness of the documentation in relation to the APIs. So, we thought through some possible solutions and decided to use a tool called Swagger which allowed us to automatically publish an update to the documentation when the codebase of the API was updated through the deployment pipeline. This completely removed forgetfulness of humans to update the documentation when the API was changed, and it ensured that the API was always up to date.

Another feedback they shared was how the API documentation wasn't properly organized based on the different types of APIs. To solve this, we adopted a tool called Readme to make it easier to collaborate with all Engineering teams and add as many code snippets as possible to ensure every developer had a good developer experience while using our documentation and APIs. Making the documentation open source also allowed external developers to contribute to the documentation as well.

5.5.3 What if someone has never written technical documentation before? What is your advice to help them get

started?

Five things - Take a course. Read More. Follow a style guide. Understand the goal of the documentation. Always proofread before publishing.

1. I always recommend taking a course first because it enables you to discover several things to do when writing technical documentation. I highly recommend the technical writing course by Google. It was written by some of the best technical writers and developers in the industry. The great thing about this course is that it is free. So, you don’t have to spend any more to learn how to write better technical documentation.
2. Reading is essential because it will help you enrich your vocabulary, keep you abreast of current trends, discover what's going on in the writing world, and also helps keep the spirit of writing alive. This quote by Lisa See “Read a thousand books, and your words will flow like a river.” says it all. The more you read other technical documentation or articles, the better you’ll be able to express yourself or structure your documentation in a better way.
3. Technical writing isn’t like any other form of writing. There are specific rules and formatting to think about. Following a style guide will help define things like when to use sentence case vs. title case and writing introductions and summaries. Both Google and Microsoft have style guides that are industry standard and good to start with.
4. Before you begin writing, think through what you’re going to write. What is the purpose of the documentation? Is it explaining a process or answering a specific question? If it’s the latter, you want to make sure the documentation only answers the question without any extra information that would be distracting. An example would be someone searching for information about how to center a div in HTML. You just want to give them the code they need, instead of beating around the bush.
5. My final piece of advice is that when you finish writing the documentation, don’t just go and publish it right away. Take an hour or so and do other things, and then come back and read it again. That helps you notice things that maybe you could have crafted better, are not necessary, or that you need to add. You should read your documentation

thinking about the reader's mindset and what they need to accomplish. Doing all these things and taking into account the readers’ perspective will help you to write outstanding technical documentation.

5.6 Summary

- Proper technical documentation leads to a clear understanding of the codebase, consistency, readability, maintainability, efficient problem resolution, and improved long-term maintainability.
- Lack of documentation can lead to a lack of understanding of the codebase, lack of consistency, difficulty in reading and updating the code, delays, and mistakes in problem resolution, loss of decisions over time, and increase technical debt and maintenance costs.
- Documentation not only helps new team members understand how things are done within your organization, but it also serves as a valuable reference for experienced developers when they encounter a problem or need to troubleshoot an issue.
- Clear and well-organized technical documentation can make it easier for new developers to identify and fix bugs, by providing detailed information about the code, it is working, and known issues, which can help to reduce onboarding time and make the process smoother and more efficient for everyone involved.
- Technical documentation should be structured to make it easy for developers to quickly find the information they need by using proper headings, an introduction and summary, visual aids, code samples, and case studies.
- Writing an outline is an important step in creating technical documentation, as it is a roadmap that helps organize the information and present it in a logical and easy-to-follow manner.
- Testing and revising technical documentation are just as important as testing the code, by getting feedback from users, testing the documentation in the same environment that it will be used, asking for any known issues, and incorporating the feedback during a documentation maintenance window.