In this lesson, we're going to go over the four primary ways to document a system and learn how to choose the one that's right for you.

Whether you've created a system or a routine that's very simple or very complex, you're probably going to need a way to store that information so that you can recall it when you need it.

That's especially true if what you've created isn't just for you, but also for other people who are going to need to know how it works and be able to reference it.

There are four primary ways to do that that range from quick and easy to more complex and time consuming.

You can memorize your system. Literally just store it in your brain. You can create a simple checklist to use each time the system needs to be run and check off each task as you do it. You can build flowcharts with all kinds of advanced logic to follow. And you can use project management software to store everything online and share with others.

None of these four options are inherently better or worse. Each one serves its own purpose and is better suited for some situations while a different one may be the better choice in other situations.

Also, none of them are mutually exclusive, so you may find that a combination of them is what works best for you.

Sometimes, you may only use one, and that will be more than sufficient. Other times, you might want to use all four in order to capture everything you need.

Try not to think of it as choosing one over another and, instead, think of it as choosing the best fit.

Consider the strengths and weaknesses of each option and put them together in a way that gets you as close as possible to the outcome you're looking for.

And to do that, we'll need to get familiar with those strengths and weaknesses, so let's go over each of them now and look at the different scenarios where they fit best.

So, first up we have memorization. In a nutshell, that means keeping all the pieces and details of your system right in your noggin.

That's really the default way to document and access any system because, well, you're already doing it. You have to do it to create a system in the first place.

And for some systems and routines, it's all you need. Memorization is great is because its accessible everywhere.

If you're running a simple system with just a few steps—and you need to be able to recall that system whether you're at your office or wading through the mud in the jungles of South America, you'll be able to.

When you store information in your brain, it comes with you anywhere you are. That's pretty obvious, so let's go over some of the other pros of memorization.

There's no tracking required. Keeping track of where you are in a process requires no outside resources. Again, you just keep it in your head and make a mental check mark each time you complete a step in your system.

Need to know where you are? Just take a moment to remember the steps you've already done, and you're back on track.

A memorized system is always up to date, too.

One important thing you understand about systems by now is that they usually need some tweaks and changes over time to be the best they can be.

If you're doing a really formal documentation process, you might hesitate to do the work to update a system because, well, it takes some effort to think through everything and redo that documentation.

But when it's all in your head, you can add, remove, or rearrange the steps of your system to your heart's content in the blink of an eye. You make a mental note, and off you go.

And memorized systems are quick to execute. You don't have to spend any time looking for information about what to do next or how to complete a step. It's all in your head and, when you want to run the system, you just… do it.

If you have a system that doesn't require too many steps, is easy to remember, and you're the only one who needs access to it, then memorization is the answer for you.

Memorization is also great where really complex decision making is required. You know, the kind of critical thinking that is incredibly difficult to get out of your head and onto a piece of paper.

But many of the things that make memorization a great solution for simple personal systems make it kind of awful for others.

For starters, they're fragile. They're really easy to forget. Most people don't have as great of a memory as they think they do. I know I don't.

If your system has a lot of steps, it can become really slow trying to remember where you are in the process. That's especially true if the system isn't one you'll completely execute in one sitting. If you have to come back to it over time to finish it, you might end up repeating steps you've already done or missing ones you forgot.

So, if accuracy is really important, memorization might not be the best choice.

And sometimes motivation is a problem. It's easier to just decide not to use a system that only exists in your head. Creating something physical to represent it can give it a kind of authority that you sometimes need to keep yourself on track.

And a memorized system is great if it's only for you or a few people with regular and easy access to each other. If you need to know the status of something, you can just have a quick conversation about it.

But memorization does not scale well. If you need to share a system or, worse, collaborate on one with more than a few people, trying to keep everyone in the loop while everyone keeps the info directly in their head is a recipe for chaos.

Memorized systems are opaque. You can't see into them from the outside. That's fine for some situations, but certainly not for others.

Next up, we have checklists. Checklists are basically just one step removed from memorization. Instead of keeping everything in your head, you write it down (or type it down) and now you have something physical to refer to.

They're usually really easy to access. You can keep them on a sheet of paper, a whiteboard, in your phone, almost anywhere.

And checklists can even handle some basic logic. You probably don't want to get too complex with it, but if you need to make sure certain steps happen in the right order or some simple decision-making needs to happen between steps, a checklist will serve you well.

And checklists can even work really well for groups. If you've developed a team process that a lot of people need to follow, a checklist is a pretty simple way to make

sure everyone's on the same page. Post your checklist in a common area and have each person check off their steps as they complete them.

Ta da! Automatic tracking.

From doctors performing surgeries to families doing their weekend chores, checklists are a simple and versatile tool that can handle a lot of different systems.

But checklists also come with some downsides you'll need to consider.

First, they're just not very good at capturing complex logic. You can do some very simple logic like ordering or skipping steps, but if analysis and decision-making is an important part of moving between the steps in your system, you're probably going to find checklists to be lacking.

They can also be difficult to audit to ensure compliance. This is especially true if you're not actively involved in the system. Maybe you helped develop the system, but other people are using it. Trying to track down progress and make sure that steps were done correctly from afar probably isn't realistic with a basic checklist.

And that might be an important feature to have if you're working with a mission critical system that a lot of people are touching and it's important to know when different steps were completed or *how* they were completed.

A basic checklist can't really handle that gracefully.

If you keep your checklist in one place and everyone uses the same one, updating and distributing it isn't too hard. But if you have multiple copies of a checklist floating around in many people's hands and you need to make a change to it, good luck with that!

It might not be too bad if just a few people need to know the changes and be trained on the updates, but if we're talking about a big group of people distributed over a large area. Whoa. That's going to be tough. A checklist might not be the best solution here.

Finally, checklists don't give you much opportunity to provide context within them. What I mean by that is checklists work best when you can communicate everything that needs to be understood about a step in just a sentence or two.

But if there's a lot of information that needs to be communicated about how to perform a step, where are you going to put that on a checklist without making it really long and difficult to use?

Now, this can be overcome with training. For instance, doctors use checklists everyday to perform complex surgeries. One step on a checklist for them might be to “open the patient’s chest cavity.”

Well, there are a few years of medical school that need to be communicated in order to do that single step properly, right?

So, if you’re working with highly specialized people who have the training necessary to figure out all the complex stuff to do from a single sentence on a checklist, then that’s not such a problem. Or, if you’re willing to perform that training each time someone new interacts with your checklist, then it’s also not a problem.

But if you need lots of people with various backgrounds to be able to operate the system, then a basic checklist probably isn’t best. Though, you can still use a checklist—you’ll just need to pair it with another solution that allows you to document more complex info.

Alright, let’s talk about flowcharts. A flowchart, like you probably know, is a set of instructions for how to do something. It’s just like a checklist or even memorization in that sense.

But a flowchart has one tremendous advantage over those options, and that is it can handle complex conditional logic really well.

As you navigate a flowchart, there are lots of ways to insert information between steps that can tell who ever’s using it what to do next based on the outcome of what they just did.

So, flowcharts are really good at handling and providing instruction on decision-making throughout a system. When branching becomes necessary in a system, a flowchart is probably the best solution available to you.

Since a flowchart can handle branching really well, it can also handle multiple goals well since different branches within a system can lead to different outcomes.

If operating your system feels like playing Plinko on The Price is Right, then a flowchart is probably your best bet.

Flowcharts are also great for teams because they can provide context to everyone involved. It makes it easier for everyone using a really complex system to see how it flows and how decisions are made at different steps within it, even if they aren’t directly involved in those steps.

It helps them visualize where they fit into big picture.

That kind of transparency can help reduce team anxiety and increase motivation. Everyone can see how steps flow—or don't flow—to them, so they start getting better at knowing when to expect to be involved and how things happened before the process got to them.

But flowcharts come with some of the same challenges that checklists do. They're hard to audit. You can't usually look at a flowchart and see where you are at any given time when you're working with a team. It's a static thing—it can't really provide that kind of functionality.

And, just like checklists, it can be difficult to communicate changes. This problem is magnified with flowcharts. When one part of a flowchart changes, it usually affects many other parts downstream.

That's not good or bad by itself; it just makes it hard to update everyone when a change happens.

So, choose a flowchart to document your system when you need to capture a lot of decision-making. Just know that making changes when a lot of people are involved might require a little patience.

Alright, the last documentation system we'll talk about is project management software. I'm talking mostly about web-based software like Trello, Basecamp, or Asana, but many industries also have their own niche software that's specifically built for their needs.

Now, I'm not going to get into specific platforms here because it's pretty fast-moving technology and, by the time I finish this video, one will probably have closed shop, two more will have been created, and one will have completely overhauled its feature set.

Instead, I'll go over the general pros and cons of putting your systems in a project management platform.

The primary benefits of using an online platform are really big ones.

First, most platforms are mainly designed to be a kind of meta-system. A place to warehouse and store many different systems.

So, if you've been doing the work throughout this course so far, then you've probably built up several different systems. Some of them might be just for you. Some others might be for you to share with your family or a team. And some might be simple systems while others are relatively complex.

A software platform gives you one place to store and recall all those different systems. So, whichever one you need at a given time, it's usually just a few clicks away.

That can solve your organizational problems if you've been thinking, "where do I keep all this information?"

A web platform also makes updating and distributing information trivial. In most cases, when you need to change something in one of your systems, you make that change in one place and it's automatically updated everywhere else it needs to be.

Project management platforms can also integrate with many other online systems you use, and that can be a big benefit.

So, if you use your email to keep track of tasks or your team uses a chat system to talk to each other, or you store files in a cloud app, you can often find a project management system that will integrate with all those tools so that information across different platforms becomes easier to access.

It can help you make a system faster if you often have to look for information in different places as you go from step to step.

And probably the biggest benefit for teams is that status and deadline management is much better in a shared, online platform.

With everyone working on the same copy of something and the software's ability to automatically track progress and create alerts for deadlines and other important events, anyone can see where you're at in a system at any time.

The software can do most of the admin work for you so that you can focus on making sure the really important stuff is getting done faster.

Those are the primary benefits. But what about the drawbacks?

In my opinion, there really aren't that many. At least not many that outweigh the benefits. Here are some to consider, though.

First, a project management tool is probably overkill if all you have are a few simple systems for yourself. These tools are pretty robust, and if you don't need the features, you might just be weighing yourself down unnecessarily.

Another risk is that, if you stop using it, your data can become siloed, meaning that you have to either keep using the platform, or abandon everything you've built on it. Lots of tools come with an export function so you can take your information with you when you leave but, in my experience, they don't do a very good job of organizing that data in a way that would be very useful if you needed to leave.

Of course, there's also the general risk that, anything you expose to the internet could be hacked. I don't worry much about this myself because if my systems were stolen, it wouldn't really harm me or anyone else.

This is really only worth worrying about if your systems require working with really sensitive data that would cause catastrophic loss if it were compromised. Probably doesn't apply to most of us, but good to remember.

Then, of course, data can be stolen when it's paper, too. So...

Most of all, what you need to understand about project management tools is that they kind of exist in their own category outside the other three options we discussed.

Any decent project management tool can create checklists, and some of them can do some limited flowcharting.

So, it's not really about choosing a PM tool over one of the other options.

Instead, it's about figuring out when you need the benefits that come from a PM tool, and using it in conjunction with those others to get everything you need. Think of it as a way to add helpful functionality to the other three solutions.

In practice, you could even find yourself using all four documentation solutions to some extent in a more complex system.

Maybe part of a system needs a flowchart to navigate really well, but other parts work fine with just a checklist. And maybe a few sub-steps are really simple, so you plan to memorize them rather than write down something that should be obvious.

And this is just one system of several you have, so you store it online with your others and make use of deadlines and assign different team members to different steps when you need to.

It's all about mixing and matching to find the combination that works best for you. You don't have to try to cram a system into one solution or try to use the same solution for all your systems.

Instead, when you need to document a system, look at it on it's own and decide which parts need which treatments. Then, decide if it'll be helpful to manage it on a project management platform, or if it doesn't need that and can live somewhere else.

So, here's what learned in this lesson.

First, we talked about when and why you might want to document a system you've created. Then, we went over the four primary ways to create that documentation. There's memorization, simple checklists, flowcharts, and project management tools. And we looked at the pros and cons of each solution to figure out when each might be more or less useful.

Finally, we learned that these four solutions are not mutually exclusive. You can combine them in any way that's most helpful to you, and you might use just one or even all four of them on a single system to get the level of detail you want.

My challenge to you today is to think about a system you've worked on in this course, and decide which of the documenting solutions would work best for you.

Think about what's needed most at each step as well as overall. Then, try to match those needs up with one or more of the solutions we just talked about, and try to actually create something. Get something down on paper or loaded into your project management platform.

When you've finished that, I'll see you in the next lesson.