📝 Transcription:

These are the five steps I always use when building AI agents and they've helped me generate over $240,000 in the past six months. In this video, I'm going to be giving you the exact framework that I use so you can start to build some powerful AI systems even if you're not technical. I use this method to build all of the automations you've seen on my channel for the past seven months, but I've never really talked about the actual methodology. So let's get into it. Starting with step one is the foundations. Don't run before you walk. I see way too many people trying to jump into a very complex AI agent system, but they I still don't understand how to set up an API call. So before I started building agents, I wanted to make sure I understood the foundational elements. So I'm sure we all know what large language models are, but it's really important to at least at a high level understand how they work. You also want to understand some data foundations, APIs and HTTP requests, run Rags and vector databases. And this exact foundational course is one of them that I teach in my community called agent zero. So the first thing to dive into is large language models. Understanding that at their core, they're basically designed to predict the next word or token in a sentence. And this unlocks some really cool capabilities, but LLMs at their core cannot actually take action in anything. So it's really important to understand, when do we need to use AI or when do we not? From there, you really want to understand the difference between the most popular, large language model families and whether they're close source or open source. Because in our automations, we're going to switch out different large language models based on what we need, based on what the clients need, and based on what the actual automation is meant to do. Because all of these different large language models have different strengths and weaknesses. The next thing to dive into would be RAG and vector databases. Together, this creates a really powerful system where AI agents can search through a database of text based on similarity rather than keyword matching. They can then take back that data and create an answer for you based on actual text that we put into that vector database rather than hallucinating something that is in its training data. And so diving into vector databases, it's definitely important to understand the way that we take a document, we chunk it up, we run it through an embedding's model, and those chunks are placed into the vector database based on the actual meaning of the words in the chunk. However, when it comes to some of this stuff with large language models and vector databases in semantic search, I have a golden rule, which is just to understand enough to get the systems to provide value. Meaning we don't have to understand the different parameters and how the weights change within a large language model when we're interacting with it and when it's being trained. And we don't have to understand the Euclidean distance formula that goes into vector database semantic search. We just have to understand based on this input, what type of output are we getting? And how can we tweak the system prompt and build in certain parameters to make this perform the way we want it to and actually provide value. In my opinion, this one may be the most important, which is JSON and APIs, because I see way too many people trying to build these systems when they don't understand how to read JSON. And it's really quite simple. It's just key value pairs, almost like you're looking at an Excel sheet. So here's an example. Alice is a 25 year old student. We have name, Alice, age 25 is student, true. And here's an example where we have an array. We have a shopping list with apples, bananas, and milk. And over here we have the shopping list array with apples, bananas, and milk. But the great news about JSON is because it's so universal and it's been around for so long. Pretty much every large language model is trained on it, so it's really easy if you're getting stuck to talk to something like ChatGT or Claude and have it help you with your JSON. And it's really, really important to understand, first of all, what an API is, but actually how you set up an HTTP request to use an API to talk to some sort of third-party server. And for pretty much every single API that you want to interact with nowadays, you're going to be sending data to it using JSON and you're going to be getting data back using JSON. understanding the foundational elements of JSON and API calls before you get into those automations and you're in the weeds, it's gonna save you a lot of time, a lot of energy, and a lot of pain. Because if we're building automations or AI agents in something like NADN, we can only stay within the NADN environment unless we use some sort of API call to something like Gmail server, HubSpot server, or just wanna search the web in general. So once you have an understanding of NADN and you have an understanding of APIs and how to use them, the possibilities are truly infinite. Okay, the second step is actually identifying high ROI opportunities. So when you're looking for a process that you want to automate, it should tick these four boxes, which are, is it repetitive? Is it time consuming? Is it error prone? And is it scalable? And the main one I wanted to talk about here today is the scalable one. Because when you identify an opportunity for a scalable automation, the ROI there is just going to compound and compound over time, meaning as the business grows, the use of the automation grows, and more time is saved, and more money is saved. And so to actually show you what this means, is let's say we build out an AI system somewhere than the sales process. And what this does is it's going to take in 50 leads a week and it's going to save sales rep five hours a week of them doing, you know, lead nurturing or research and creating any sort of sales brief. And now that a sales person has gotten those five hours back per week, they can use those five hours to do higher impact activities which are going to actually grow the business rather than them sitting there and doing research. And as a direct result of the business growing, we're going to start to get more leads per week. So now if we start to get 200 leads per week going through the AI system, we're We're going to start to save more and more time per week. So 20 hours per week rather than five hours per week. And once again, the sales team getting 20 hours back per week is going to lead to the business growing even more, which in turn leads to more leads going through the system. So hopefully this really shows you how building scalable systems can compound over themselves over time. But maybe you're asking yourself what's a scalable system and what's not. So something like a personal assistant, let's say the business grows, you know, triples and revenue during the course of the year. Does that mean that you will be tripling the amount of times that you're using the personal assistant, maybe you'll start to use it a little more to set up more meetings or something, but really, that system is not going to compound on itself the same way that something like a lead AI system would. And sometimes truly, half the battle is picking the right process to actually automate and making sure that it's high ROI. Okay, now that we've identified the process, the third step is process mapping. I think this is something that's super, super overlooked, and whenever I get an idea for an automation or an AI agent that I want to build, I'll never just hop straight into N&N, pull up a new workflow, and just start building. That would be like opening a bag of all of these Legos and you know you need to make a parrot, but you don't have the instructions right next to you to build a parrot, or you dump out a puzzle on a table, but you don't know what the end picture is. The analogy here is that you definitely still could do it, but you're going to have a lot more trouble and it's going to take you a lot longer than if you would have just looked at the instruction manual to start with. So basically our job here in the process mapping step is to build out the instruction manual. And that means that we list every single step no matter how small. This is going to help make everything a lot more clear. We're going to be able to identify certain bottlenecks or edge cases before we even get into the build. It's going to help us with scalability and modularity for making sure that this is like, you know, a long-term solution. And of course, it's going to make your actual build time, your hands-on keyboard development in NADN or whatever you're building your agents. It's going to make that time a lot more efficient. So here's a very simple example of a process map and how that can transfer into a workflow. As you can see, we have step one over here, which is a new email received, and this is what triggers the automation. So we threw in a Gmail trigger. Step two is to determine if the email is customer support related. If not, forward to the correct team. If yes, support team reads the message. So notice how we list out what happens based on the decision as well. And here, we're able to identify right away, okay, we probably need AI for this classification. So we use an AI text classifier. We have it split out to the different paths. And then step three and four is to look up the information needed to answer the request, as well as draft to help for response. So we knew that we could use an AI agent right here to look up information in our pine cone vector database, as well as use its chat model to draft a response to the email. Step number five is to label the email for tracking. So we do that right here with the label email node. And then the sixth step is our final data destination, which is sending the reply right over here, again, using a Gmail node. And when you start to do process mapping and wireframing over and over, you're gonna notice that there's kind of five main things that you're looking out for, which is a trigger, which could be in the form of like a form submission or a manual trigger or a new email, whatever starts the process. And the easiest way to break this down is just think about how you, yourself, or someone on your team manually does this process. So we start with a trigger, which is like what happens in real life that makes me or my team go start to do this process. So in NADN, that could look like a form submission or a new email or a new record in your CRM or a new Slack message, whatever it is, identify the trigger. And then we have an element of data sources. So this can be multiple things. Probably from the trigger, you're gonna get some initial data and that's one of the data sources, but you could also be looking in like a Pinecon vector database or you could be searching the internet or you could be looking up some information about a customer in your CRM. And then there's probably gonna be some element of data transformation, whether that's filtering certain things or merging certain rows together, whatever it is, usually we have to do something with the data. It could even be as simple as just cleaning it up or removing duplicates, but we have to do some sort of data transformation. And by now, hopefully you guys understand the pattern, which is, you know, every step in this process has some sort of node associated with it in NADN. Pretty much every process, we have some sort of decision point, and this helps us identify what do we actually use to make the decision, what's the criteria, and then based on that decision, do different things happen for each of those outputs. And then from there, it's kind of whatever ends the process, which is the data destination, which is the end of that journey through this workflow. Step number four is a really, really important one, which is workflows versus AI agents. And I actually did make a full video breaking this down, so if you wanna check that out, I'll link it right up here. Anyways, the whole theory here is AI is cool, and AI agents are cool, but never force AI, or never force an agent, into a process that doesn't actually need it. because all you'd be doing is increasing latency, increasing the cost, and increasing the risk of inconsistent outputs and lower quality outputs. So here's an example. We have a customer's port agent, which is going to trigger by a new email. The agent will then look up information and then send a reply. And that's cool. It works. It's an agent. But we could build out this exact same system for cheaper and probably it'll be quicker by using a workflow. You can see it starts with the same trigger. We have the same process of searching in knowledge base or filtering out some stuff or writing the email and sending the email. So it does the exact same thing, just a lot more efficient. And you may have noticed the difference here is that this is a linear process where it must go one, two, three, four, five, six. And we have basically guard rails that say stay on this path no matter what. But over here, it could go one, two, three, four, five, six, seven. We don't know what it's gonna do. So just to take a step back real quick, the evolution. We start with ChatGPT, which is a large language model, where we give it an input, at its core, it can't take any action, it doesn't have any tools, and it gives us an output. And then from here, we got into AI workflows, where we were implementing an LLM call somewhere within a workflow. Because automations have been around for a long time, which was just moving data from left to right. But now we have the really cool ability to use an LLM to either generate some sort of text for us or make decisions for us. So you can see here, there's an input, there's a tool call, there's an LLM, and then there's two other tools and then an output. And from here, we evolved into AI agents where we have the LLM basically is the brain of the whole thing, and it has access to the different tools that it needs to use to get the job done. So because this is super cool and there's been a lot of hype, we've seen a lot of people, myself included, using agents wrong, where a workflow would just get the job done a lot better. So here's another example where we have like a technical analyst agent, where we're talking to it in Telegram, and then it calls on a tool in order to actually go get the chart, download it, and make some sort of analysis. But because we know that this process is gonna happen in the same order every time, because we've process mapped it, we know that it makes way more sense to use an AI workflow, where we go one, two, three, four, five, six, seven, and it does the exact same thing as this AI agent over here, but it's quicker and cheaper. And then the fifth step is about POC, proof of concept, and guard rails. So there are three things that I like to say here. The first one is fail fast, debug often. No matter how experienced you are with NADN and building agents, when you start a workflow from scratch, you're going to have errors. And the quicker you can get to those errors and debug them, the quicker you can get to a POC. And getting to a POC quick is so, so important because moving on to bullet two, you don't know what you don't know. This basically means when we process map everything out, we can try to implement as many guardrails as possible, and we can try to predict certain scenarios that may happen and build in fixes for those. But at the end of the day, you don't know what you don't know, and there's gonna be multiple edge cases that are only going to present themselves once you've exposed this POC to different users, and you've started to test different prompts running through the system. And you're likely gonna have more and more edge cases pop up, and then your job is to basically identify the patterns of what's causing this error, what can we do in the workflow to fix it, and make sure that it doesn't happen. And from there, we move on to bullet three, which is there's no such thing as a finished product. And the reason I say this is because even if you've built something out and you've iterated upon it multiple times and you've built in different guardrails and you've identified different edge cases, there's always that element of a new chat model drops or you've learned something new that basically changes the way you structure your workflows or you wanna make something a sub workflow because it'll be quicker. There's just always opportunity to learn and implement new things and also so much opportunity to make these systems faster and cheaper. So there are many different types of guardrails that you can build in and different edge cases that you may be exposed to. And some of them are very simple in the sense that you may be able to predict them while you're doing your process mapping. So something like, okay, someone accidentally submits two forms, we're gonna clean up those emails by sort of removing duplicates. That's something that we may work in from day one. But of course, there are gonna be a lot of things that pop up that you didn't think about from day one, even if you did a really robust process map. So you may need to add certain conditional checks within your flow, or you may need to add some sort of loop because you're experiencing too much data, and then maybe within that loop, you need to have continuing on error with a certain different output for anything that avert. Or you may realize that this system works great for a hundred records, but what happens if I expose it to a thousand or 5,000? You may have to build in some loops, you may have to build in some sub workflows, you may have to store data somewhere third party, you may have to implement caching. There's just so much stuff that you don't know until your system actually starts to have users and you expose it to more situations. And once you've identified certain patterns that are causing the errors or poor quality responses in your system, there's almost always a fix for those. So here's a quick practical example. I had a sub workflow where the main agent was sending data to it. You can tell because I have a one executed by another workflow trigger. And every one out of 50 runs or so, I was seeing this error where an empty query was coming through for no reason, which basically just broke the system. So all I had to do was a quick conditional check where I was saying does the query exist? If no, we're gonna send it this way and it's gonna retry, but if yes, then we're fine and we'll just continue down the rest of the process. So you may have to get creative, but if you've identified a pattern, you can do something about it because when it comes to automations, predictable is better. So if you know something happens in a certain way, you can pretty much build out something to handle that exact case every time. When it comes to automations, boring is better. So that was my five step AI agent playbook that I've been using for the past months, building these systems, starting with the foundations, identifying high ROI opportunities, process mapping, workflows versus agents, and then POC and guard rails. And I really, really believe in taking this iterative approach where you learn things that build on top of each other, especially if you're not coming from a coding or technical background. And this is exactly why I created the course from 10 hours to 10 seconds, which is basically the exact methodology that I just covered with you guys, but in much more detail. There's over 11 hours of content in here. We talk about picking the process, we talk about mapping the steps, we talk about the foundations of NADN, triggers, nodes, different elements of a workflow, AI nodes, and also smarter NADN workflows. So if this sounds like something that you wanna dive into, and it interests you, then definitely check out my paid course in community using the link down in the description. What I talked about today were these two courses, Agent Zero, which is the foundational course, and then 10 hours to 10 seconds. And of course, we've got a great community of over 1200 members, it's growing really quick. And everyone in here is also going through those same courses and building automations every day with NADN. We also have five live calls per week, where we bring in guest speakers, Q&As, coffee chats, tech support, just to make sure you're never getting stuck, but also meeting people in this space. So that's gonna do it for today's video. I hope you guys enjoyed and I hope you learned something new. If you did, please give it a like, definitely helps me out a ton. And as always, I appreciate you guys sticking around to the end of the video. I'll see you in the next one. Thanks everyone.

 Final Transcription:

These are the five steps I always use when building AI agents and they've helped me generate over $240,000 in the past six months. In this video, I'm going to be giving you the exact framework that I use so you can start to build some powerful AI systems even if you're not technical. I use this method to build all of the automations you've seen on my channel for the past seven months, but I've never really talked about the actual methodology. So let's get into it. Starting with step one is the foundations. Don't run before you walk. I see way too many people trying to jump into a very complex AI agent system, but they I still don't understand how to set up an API call. So before I started building agents, I wanted to make sure I understood the foundational elements. So I'm sure we all know what large language models are, but it's really important to at least at a high level understand how they work. You also want to understand some data foundations, APIs and HTTP requests, run Rags and vector databases. And this exact foundational course is one of them that I teach in my community called agent zero. So the first thing to dive into is large language models. Understanding that at their core, they're basically designed to predict the next word or token in a sentence. And this unlocks some really cool capabilities, but LLMs at their core cannot actually take action in anything. So it's really important to understand, when do we need to use AI or when do we not? From there, you really want to understand the difference between the most popular, large language model families and whether they're close source or open source. Because in our automations, we're going to switch out different large language models based on what we need, based on what the clients need, and based on what the actual automation is meant to do. Because all of these different large language models have different strengths and weaknesses. The next thing to dive into would be RAG and vector databases. Together, this creates a really powerful system where AI agents can search through a database of text based on similarity rather than keyword matching. They can then take back that data and create an answer for you based on actual text that we put into that vector database rather than hallucinating something that is in its training data. And so diving into vector databases, it's definitely important to understand the way that we take a document, we chunk it up, we run it through an embedding's model, and those chunks are placed into the vector database based on the actual meaning of the words in the chunk. However, when it comes to some of this stuff with large language models and vector databases in semantic search, I have a golden rule, which is just to understand enough to get the systems to provide value. Meaning we don't have to understand the different parameters and how the weights change within a large language model when we're interacting with it and when it's being trained. And we don't have to understand the Euclidean distance formula that goes into vector database semantic search. We just have to understand based on this input, what type of output are we getting? And how can we tweak the system prompt and build in certain parameters to make this perform the way we want it to and actually provide value. In my opinion, this one may be the most important, which is JSON and APIs, because I see way too many people trying to build these systems when they don't understand how to read JSON. And it's really quite simple. It's just key value pairs, almost like you're looking at an Excel sheet. So here's an example. Alice is a 25 year old student. We have name, Alice, age 25 is student, true. And here's an example where we have an array. We have a shopping list with apples, bananas, and milk. And over here we have the shopping list array with apples, bananas, and milk. But the great news about JSON is because it's so universal and it's been around for so long. Pretty much every large language model is trained on it, so it's really easy if you're getting stuck to talk to something like ChatGT or Claude and have it help you with your JSON. And it's really, really important to understand, first of all, what an API is, but actually how you set up an HTTP request to use an API to talk to some sort of third-party server. And for pretty much every single API that you want to interact with nowadays, you're going to be sending data to it using JSON and you're going to be getting data back using JSON. understanding the foundational elements of JSON and API calls before you get into those automations and you're in the weeds, it's gonna save you a lot of time, a lot of energy, and a lot of pain. Because if we're building automations or AI agents in something like NADN, we can only stay within the NADN environment unless we use some sort of API call to something like Gmail server, HubSpot server, or just wanna search the web in general. So once you have an understanding of NADN and you have an understanding of APIs and how to use them, the possibilities are truly infinite. Okay, the second step is actually identifying high ROI opportunities. So when you're looking for a process that you want to automate, it should tick these four boxes, which are, is it repetitive? Is it time consuming? Is it error prone? And is it scalable? And the main one I wanted to talk about here today is the scalable one. Because when you identify an opportunity for a scalable automation, the ROI there is just going to compound and compound over time, meaning as the business grows, the use of the automation grows, and more time is saved, and more money is saved. And so to actually show you what this means, is let's say we build out an AI system somewhere than the sales process. And what this does is it's going to take in 50 leads a week and it's going to save sales rep five hours a week of them doing, you know, lead nurturing or research and creating any sort of sales brief. And now that a sales person has gotten those five hours back per week, they can use those five hours to do higher impact activities which are going to actually grow the business rather than them sitting there and doing research. And as a direct result of the business growing, we're going to start to get more leads per week. So now if we start to get 200 leads per week going through the AI system, we're We're going to start to save more and more time per week. So 20 hours per week rather than five hours per week. And once again, the sales team getting 20 hours back per week is going to lead to the business growing even more, which in turn leads to more leads going through the system. So hopefully this really shows you how building scalable systems can compound over themselves over time. But maybe you're asking yourself what's a scalable system and what's not. So something like a personal assistant, let's say the business grows, you know, triples and revenue during the course of the year. Does that mean that you will be tripling the amount of times that you're using the personal assistant, maybe you'll start to use it a little more to set up more meetings or something, but really, that system is not going to compound on itself the same way that something like a lead AI system would. And sometimes truly, half the battle is picking the right process to actually automate and making sure that it's high ROI. Okay, now that we've identified the process, the third step is process mapping. I think this is something that's super, super overlooked, and whenever I get an idea for an automation or an AI agent that I want to build, I'll never just hop straight into N&N, pull up a new workflow, and just start building. That would be like opening a bag of all of these Legos and you know you need to make a parrot, but you don't have the instructions right next to you to build a parrot, or you dump out a puzzle on a table, but you don't know what the end picture is. The analogy here is that you definitely still could do it, but you're going to have a lot more trouble and it's going to take you a lot longer than if you would have just looked at the instruction manual to start with. So basically our job here in the process mapping step is to build out the instruction manual. And that means that we list every single step no matter how small. This is going to help make everything a lot more clear. We're going to be able to identify certain bottlenecks or edge cases before we even get into the build. It's going to help us with scalability and modularity for making sure that this is like, you know, a long-term solution. And of course, it's going to make your actual build time, your hands-on keyboard development in NADN or whatever you're building your agents. It's going to make that time a lot more efficient. So here's a very simple example of a process map and how that can transfer into a workflow. As you can see, we have step one over here, which is a new email received, and this is what triggers the automation. So we threw in a Gmail trigger. Step two is to determine if the email is customer support related. If not, forward to the correct team. If yes, support team reads the message. So notice how we list out what happens based on the decision as well. And here, we're able to identify right away, okay, we probably need AI for this classification. So we use an AI text classifier. We have it split out to the different paths. And then step three and four is to look up the information needed to answer the request, as well as draft to help for response. So we knew that we could use an AI agent right here to look up information in our pine cone vector database, as well as use its chat model to draft a response to the email. Step number five is to label the email for tracking. So we do that right here with the label email node. And then the sixth step is our final data destination, which is sending the reply right over here, again, using a Gmail node. And when you start to do process mapping and wireframing over and over, you're gonna notice that there's kind of five main things that you're looking out for, which is a trigger, which could be in the form of like a form submission or a manual trigger or a new email, whatever starts the process. And the easiest way to break this down is just think about how you, yourself, or someone on your team manually does this process. So we start with a trigger, which is like what happens in real life that makes me or my team go start to do this process. So in NADN, that could look like a form submission or a new email or a new record in your CRM or a new Slack message, whatever it is, identify the trigger. And then we have an element of data sources. So this can be multiple things. Probably from the trigger, you're gonna get some initial data and that's one of the data sources, but you could also be looking in like a Pinecon vector database or you could be searching the internet or you could be looking up some information about a customer in your CRM. And then there's probably gonna be some element of data transformation, whether that's filtering certain things or merging certain rows together, whatever it is, usually we have to do something with the data. It could even be as simple as just cleaning it up or removing duplicates, but we have to do some sort of data transformation. And by now, hopefully you guys understand the pattern, which is, you know, every step in this process has some sort of node associated with it in NADN. Pretty much every process, we have some sort of decision point, and this helps us identify what do we actually use to make the decision, what's the criteria, and then based on that decision, do different things happen for each of those outputs. And then from there, it's kind of whatever ends the process, which is the data destination, which is the end of that journey through this workflow. Step number four is a really, really important one, which is workflows versus AI agents. And I actually did make a full video breaking this down, so if you wanna check that out, I'll link it right up here. Anyways, the whole theory here is AI is cool, and AI agents are cool, but never force AI, or never force an agent, into a process that doesn't actually need it. because all you'd be doing is increasing latency, increasing the cost, and increasing the risk of inconsistent outputs and lower quality outputs. So here's an example. We have a customer's port agent, which is going to trigger by a new email. The agent will then look up information and then send a reply. And that's cool. It works. It's an agent. But we could build out this exact same system for cheaper and probably it'll be quicker by using a workflow. You can see it starts with the same trigger. We have the same process of searching in knowledge base or filtering out some stuff or writing the email and sending the email. So it does the exact same thing, just a lot more efficient. And you may have noticed the difference here is that this is a linear process where it must go one, two, three, four, five, six. And we have basically guard rails that say stay on this path no matter what. But over here, it could go one, two, three, four, five, six, seven. We don't know what it's gonna do. So just to take a step back real quick, the evolution. We start with ChatGPT, which is a large language model, where we give it an input, at its core, it can't take any action, it doesn't have any tools, and it gives us an output. And then from here, we got into AI workflows, where we were implementing an LLM call somewhere within a workflow. Because automations have been around for a long time, which was just moving data from left to right. But now we have the really cool ability to use an LLM to either generate some sort of text for us or make decisions for us. So you can see here, there's an input, there's a tool call, there's an LLM, and then there's two other tools and then an output. And from here, we evolved into AI agents where we have the LLM basically is the brain of the whole thing, and it has access to the different tools that it needs to use to get the job done. So because this is super cool and there's been a lot of hype, we've seen a lot of people, myself included, using agents wrong, where a workflow would just get the job done a lot better. So here's another example where we have like a technical analyst agent, where we're talking to it in Telegram, and then it calls on a tool in order to actually go get the chart, download it, and make some sort of analysis. But because we know that this process is gonna happen in the same order every time, because we've process mapped it, we know that it makes way more sense to use an AI workflow, where we go one, two, three, four, five, six, seven, and it does the exact same thing as this AI agent over here, but it's quicker and cheaper. And then the fifth step is about POC, proof of concept, and guard rails. So there are three things that I like to say here. The first one is fail fast, debug often. No matter how experienced you are with NADN and building agents, when you start a workflow from scratch, you're going to have errors. And the quicker you can get to those errors and debug them, the quicker you can get to a POC. And getting to a POC quick is so, so important because moving on to bullet two, you don't know what you don't know. This basically means when we process map everything out, we can try to implement as many guardrails as possible, and we can try to predict certain scenarios that may happen and build in fixes for those. But at the end of the day, you don't know what you don't know, and there's gonna be multiple edge cases that are only going to present themselves once you've exposed this POC to different users, and you've started to test different prompts running through the system. And you're likely gonna have more and more edge cases pop up, and then your job is to basically identify the patterns of what's causing this error, what can we do in the workflow to fix it, and make sure that it doesn't happen. And from there, we move on to bullet three, which is there's no such thing as a finished product. And the reason I say this is because even if you've built something out and you've iterated upon it multiple times and you've built in different guardrails and you've identified different edge cases, there's always that element of a new chat model drops or you've learned something new that basically changes the way you structure your workflows or you wanna make something a sub workflow because it'll be quicker. There's just always opportunity to learn and implement new things and also so much opportunity to make these systems faster and cheaper. So there are many different types of guardrails that you can build in and different edge cases that you may be exposed to. And some of them are very simple in the sense that you may be able to predict them while you're doing your process mapping. So something like, okay, someone accidentally submits two forms, we're gonna clean up those emails by sort of removing duplicates. That's something that we may work in from day one. But of course, there are gonna be a lot of things that pop up that you didn't think about from day one, even if you did a really robust process map. So you may need to add certain conditional checks within your flow, or you may need to add some sort of loop because you're experiencing too much data, and then maybe within that loop, you need to have continuing on error with a certain different output for anything that avert. Or you may realize that this system works great for a hundred records, but what happens if I expose it to a thousand or 5,000? You may have to build in some loops, you may have to build in some sub workflows, you may have to store data somewhere third party, you may have to implement caching. There's just so much stuff that you don't know until your system actually starts to have users and you expose it to more situations. And once you've identified certain patterns that are causing the errors or poor quality responses in your system, there's almost always a fix for those. So here's a quick practical example. I had a sub workflow where the main agent was sending data to it. You can tell because I have a one executed by another workflow trigger. And every one out of 50 runs or so, I was seeing this error where an empty query was coming through for no reason, which basically just broke the system. So all I had to do was a quick conditional check where I was saying does the query exist? If no, we're gonna send it this way and it's gonna retry, but if yes, then we're fine and we'll just continue down the rest of the process. So you may have to get creative, but if you've identified a pattern, you can do something about it because when it comes to automations, predictable is better. So if you know something happens in a certain way, you can pretty much build out something to handle that exact case every time. When it comes to automations, boring is better. So that was my five step AI agent playbook that I've been using for the past months, building these systems, starting with the foundations, identifying high ROI opportunities, process mapping, workflows versus agents, and then POC and guard rails. And I really, really believe in taking this iterative approach where you learn things that build on top of each other, especially if you're not coming from a coding or technical background. And this is exactly why I created the course from 10 hours to 10 seconds, which is basically the exact methodology that I just covered with you guys, but in much more detail. There's over 11 hours of content in here. We talk about picking the process, we talk about mapping the steps, we talk about the foundations of NADN, triggers, nodes, different elements of a workflow, AI nodes, and also smarter NADN workflows. So if this sounds like something that you wanna dive into, and it interests you, then definitely check out my paid course in community using the link down in the description. What I talked about today were these two courses, Agent Zero, which is the foundational course, and then 10 hours to 10 seconds. And of course, we've got a great community of over 1200 members, it's growing really quick. And everyone in here is also going through those same courses and building automations every day with NADN. We also have five live calls per week, where we bring in guest speakers, Q&As, coffee chats, tech support, just to make sure you're never getting stuck, but also meeting people in this space. So that's gonna do it for today's video. I hope you guys enjoyed and I hope you learned something new. If you did, please give it a like, definitely helps me out a ton. And as always, I appreciate you guys sticking around to the end of the video. I'll see you in the next one. Thanks everyon