the whole world is Raging about AI agents right now yet some of the biggest companies like apple and Amazon still struggle to ship effective AI features within their products last week Apple had to pull back Apple intelligence because it was hallucinating in the new summarizations that the product was providing and also Amazon still struggles to put AI features into Amazon Alexa because of the hallucinations yet if you look online on YouTube blog post everyone seems to be building these AI agents and everyone has their own ideas and tools and Frameworks on how to do so but here's the hard truth building effective and reliable AI agents is really hard and most of the examples that you will see online are really cool demos but they are just that they show what's possible they show where the future is going with AI agents but if you really put that into your product and let a lot of people use it it will just simply break down now in this video I want to share some practical tips and techniques for developers to build more effective and reliable ages now while I definitely don't have all the answers here I'm going to share some of the lessons that I've learned over the past 2 years building AI systems for our clients and I'm also going to share insights from some of the leading companies working on these Technologies and now if you're new to the channel my name is Dave abar I'm the founder of data Lumina I hold a bachelor and a master's degree in artificial intelligence and my journey we just started about 10 years ago and for the past 6 years I've been building custom data and AI solutions for my clients and next to that I also run a community with

over 100 freelance data nii developers
and I make this video to help you level
up and become a better engineer so
perhaps eventually you might want to
join us all right so let's first talk
about what AI agents actually are
because before we can build them we
first need to all be on the same page
about what they are right and depending
on who you ask you are going to get a
very different answer most of the time
and that is because a lot of people have
different ideas about what they are and
if right now you'll search for how to
build AI agents online you'll find some
tutorials almost all of the tutorials
that you'll find where they talk about
AI agents what they simply mean is you
have some piece of software you have
some operations where at some point
you're going to make an API call to a
large language model but now is it
really fair to say that such a system
can already be called an AI agent well
if you ask the experts the answer is no
yet why is then everybody talking about
AI agents like they are just dead well
that is simply because there is a lot of
hype and a lot of Buzz around that
particular word everyone wants to learn
what AI agents are how to build them but
really in the end what they're really
after is they want to learn or they want
to implement some kind of system that
can take some process and automate it
they want an automation that's
essentially what AI can do for us and
right now anytime that topic comes up we
say oh it's an AI agent but for this
video and for you as a developer I want
to dig a little bit deeper and show you
some of the different tools and
techniques that you can use to what we
refer to at dat Lumina as AI systems

rather than AI agents where not all AI systems are necessarily AI agents and now to make this more clear for you throughout the rest of this video I want to use a definition or distinction rather as introduced by entropic because this makes total sense to us and this is exactly also how we see the distinction between the different types of AI systems that you can make so so in this excellent blog post called how to build effective agents the under the section what are agents they talk about first how a lot of people have a lot of different ideas about it but then at entropic they make a clear distinction between workflows and agents where workflows quoting are systems where llms and tools are orchestrated through predefined code paths so this really aligns with what I was saying previously and what you often find online where you have a certain system some steps some change and at some point you make a call to an llm now agents on the other hand are systems where llms dynamically direct their own processes and Tool usage maintaining control over how they accomplish the task so there's a clear distinction between the two between workflows and agents and as a developer it's crucial to understand when to use which pattern and because of all the tutorials and information out there right now where everyone is using the word agents thinking that you need all kinds of tools and Frameworks in order to build agents there is a lot of confusion if we then come back to entropic's blog post and to the section when and when not to use agents and I in my opinion this really hits the spot so as a developer consider the following when building applications with llms we

recommend finding the simplest solution possible something you should always do as an engineer and only increasing complexity when needed this might mean not Building agentic Systems at all and here's the for many applications however optimizing single llm calls with retrieval and in context examples is usually enough and we've been building AI systems for our clients across industry since the day the cat GPT API came out and I can definitely tell you this is true for many applications you don't need these agentic patterns you can build predefined simple workflows that solve a particular problem really well and then create a suite of tests and uations around it to really keep it in control and to really optimize it over time okay so then how do you actually build effective AI agents or rather AI systems I would say so the first step for you as a developer is really deciding on what you are going to use to build your AI system now if you have coding skills you probably want to use something like python or typescript or even JavaScript and if you don't have coding skills you probably want to look at something like make.com n8n or flow wise and it doesn't really matter what tool you use like if you do full coding or use these uh workflow Builders which they are essentially are you in both cases you can build reliable systems it's much more about the underlying patterns that you use to control the flow of your application and your data so that's what I want to dive in right now coming back again to the excellent blog post from entropic where they outline some of the different patterns that you can use to build applications around large language models so here are

some of the common building blocks that you can use when building AI systems whether that's workflows or agents and the basic building block that we start with that you all start with is what they call the augmented llm so we start with an llm a simple API call in the beginning and we can augment that we can enhance that by focusing on three things the first one is retrieval then we have tools and then we have memory starting with retrieval this is where your AI system pulls information from a different Source typically a database or a vector database and makes that available within the context of the large language model now in practice typically this is done through retrieval augmented generation or rag for short using a Vector database where you perform a similarity search on people often compare this to giving your llm application long-term memory because you can essentially offload all of the relevant contexts that you need to affect your database and whenever the llm needs it you can try to retrieve it and I'm saying try because with retrieval augmented generation or rag you never truly know what you're going to get back especially as the scale of your uh database grows so the second augmentation of llms is what they call tools and tools are essentially little services or apis that you can call within your application in order to get more information this could be for example making an API call to get the current weather data or to get the latest shipping updates on your parel using the uh trkking number and then the last part here is memory which in this context simply refers to the Past interaction that you've had with the llm

system so you can think of this for example when you're talking to cat GPT every time you send or ask something to the model that is then a new record the model will then respond that is a new record and the whole chain the whole sequence of all of those interactions together is what you call the memory so these are the three common Concepts that you'll encounter when you're working with llms and creating these agents in order to make the API call to the llm better this is all about enhancing the context providing more context in order to get better results and when you combine these three and put them together in Perfect Harmony and in sync they take your application Beyond just being a simple open AI or Chad GPT wrapper and really taking your app or automation to the next level just because it can get all of the right context at the right time all right and then up next let's look at a pattern that you can use to build very effective work workflows and often this is all you need and this is called prompt chaining so this is simply just chaining together multiple calls to an llm and typically using the previously generated information and then passing that to the next llm call with in the sequence and in this way you can break down a complex problem and instead of asking the AI write a block post you can break that down to First do some research and get clear on some ideas then dial in on a specific topic then provide an outline of what the um the essay or blog post could look like then give me all of the uh different chapters now write chapter one now write chapter two and every one of those steps could be a a chain within the whole application where at every

step you now have more control because at every step there is data and a prompt that you contr control and can tweak and this is guaranteed to improve your application and to make your system smarter overall all right and then let's get into the second pattern routing so while promp chaining can already get you great results if you focus on a single problem if the scope of the problem that you're trying to solve grows bigger and there are multiple scenarios multiple cases multiple solutions that is where routing comes in and what you essentially do with routing is given all the the data and context that is coming in you let the llm decide which way to go and you could clearly instruct the llm for the first step to categorize the incoming requests so is it a or is it B and we then capture that in a structured way so then within our application within our control flow we can use the so-called routers which in theory are practically just if statements or cases where we make a certain match and if output equals a we call that particular function with then which then goes into a particular direction and if the output was B for example we go in a different direction so that is routing and then real quick if you're a developer you got some technical skills and you've been thinking about starting as a freelancer maybe taking on some side projects to make a little bit extra money or learn more but you don't really know where to get started or struggle to land that first client you might want to check out the first link in the description it's a video of me going over how my company can help you with that we have a community with over 100 freelance data nii developers and we're all here to

make more money work on fun projects and create Freedom so if that's something you're interested in you might want to check it out all right and the third workflow pattern is paralyzation and with paralyzation you also make similar to a prom chaining you make multiple llm calls but rather than doing doing them sequentially where one output might depend on the other one you do them in parallel and this is ideal for where you can really split up a certain uh certain task break it down but they are independent of each other and this can help you to speed up your application because the output would be the same if you sequence them sequentially but then you need to make multiple API calls and wait for that to finish then do the other one do the other one this is paralyzation is a way to do this async so typical example of this could be when you're implementing guard rails and you want to evaluate a certain output you might have one prompt that valuates accuracy or correctness you might have one prompt that evaluates uh harmfulness and you might have one prompt that is specifically targeted at uh capturing prompt injections so this can all be done in parallel it could then come together and that could act as your guardrail system all right and the next pattern is the orchestrator worker so this is still a workflow pattern but this this already gets a little bit more agentic because it requires a little less explicit programming of steps but it's still sequential and linear in nature and therefore also really predictable so here's an overview of what that looks like visually so another example of how you might want to use this let's take customer care again and

an email from uh a customer is coming in and uh you let an llm look at all of the contexts so the customer question you have the CRM data and maybe you also have some some other order data and you have the uh the the customer care guidelines you assess all of that and then ask an llm what's required in order to solve this particular problem and the llm might decide okay we need to look up an assess the right uh section within the customer care Playbook we need to do a look up of the particular order to check the status and we also need to uh call the shipping API in order to get the latest shipping information through this pattern we can make our application already more agentic and have less hardcoded uh Pathways in there all right and then the last workflow pattern is the evaluator Optimizer and this is simply where you let an llm create an output and then feed that into another llm call to review it and give feedback and then pass that to another llm to improve that so you could say write a blog post you have the blog post then you have another prompt that says critically review this blog post and make sure it aderes to XYZ a whole list of things that you would like to have there then give a detailed report of what we can improve the llm will do that you now have a list of feedback points and we feed that back to the llm here's the original output here's the feedback now process all of that all right so that were the workflow patterns but now let's look at what an agent pattern actually looks like and here is that displayed visually where we have a request from a human in this case case we make an llm call that llm is going to decide to take a certain action it's

going to check and assess the output of that action within a certain environment so this is mainly just the data that it has access to and the outputs and it's going to provide feedback back to the llm and it does that in a loop until it reaches a certain criteria either uh completing the tasks or a certain stopping criteria or an intermediate step where it's going to ask feedback from a human in the loop so as you can see this is really agentic there is almost no like hardcoded Steps in between just an environment and instructions and tools that an llm can use and it goes off in this Loop and that is one of the key distinctions compared to all of the other patterns that we've described where a workflow has clearly a start point and an end point whereas with a true agent system we don't know all that we know is that we have given it specific instructions to operate within a certain environment to reach a specific goal but if it's going to do that on the first try on the second try on the 100 try or never because it just keeps iterating stock in a loop we don't know yet and with this specific pattern agents can really handle sophisticated tasks but the implementation is often very straightforward we don't have to go to the drawing board and draw out a huge like diagram or workflow we just give it a set of instructions and rules and they are typically just llms using tools based on environmental feedback in a loop and so while these agents in this pattern is actually pretty straightforward to implement getting reliable results from them is really hard and something that is everyone is trying to figure out right now and for

most problems again I want to stress this for most problems you don't need a pattern like this and you don't want it you want far more control you want to start really small and work your way up building confidence with within the system and I think the best example of this was was Devin the AI software engineer that got a lot of hype I think already half a year ago and that you can now you can now rent it or buy it as like a junior engineer for your team it's pretty pricey and based on what I've seen so far and people that actually worked with it the results are pretty so out of I saw one example where out of 20 tasks that they would try I think only four or so would work and Deon is a perfect example of a true AI agent where you as a human give it a task you give it a goal let's say code this application or solve this BG and it will go off on itself and it will start to work in loops and it's going to perform unit tests try to run your code try to build see if it runs looks at the errors and then tries to correct those errors over and over again so that's a true agentic system but it doesn't work yet because it's really hard to pull off properly so those are the core patterns that you can use as a developer to build your AI systems and it doesn't really matter which tool or framework or platform you're using whether that's pure python or make.com almost all of these patterns can be implemented and really the key here is to start as simple as possible and build up the complexity over time only when that's necessary and now to conclude this video I want to give you some final tips next to these like core patterns that you should understand that was really the

main message of this this video to dive a little bit deeper and understand that understand the differences and now the first one one we already covered but I want to sayate it one more time be very careful with agent Frameworks they can get you up and running really quickly but make sure that you understand everything that's going on and you probably don't need them and it will make you a better engineer if you learn how to build these basic core components from the ground up within your own code base and now the second tip is to to prioritize deterministic workflows over complex agent patterns start really simple and really isolate the problem and build it from the ground off so you first focus on a particular workflow particular problem where you can really optimize that and nail that so it works almost 100% of the time and not 80% of the time and how you can do this look at the problem overall look at all the data that you need to process and as the first step within your system create a categorization step where you essentially only select a very small portion of their problem so let's take the customer care example let's say out of all the tickets that you have first focus on the where's my order question so all the tickets that are coming in you build a router only if the llm decides this is about an order throw it into the AI workflow and solve that all the other tickets just send it to the human agents and this can be extrapolated this idea to almost any problem that you solve take all of the data get categorize it focus on what you want to solve first go vertical on that then scale horizontal to other problems as well once you understand the full

scope of the problem you can start to think about introducing more agentic patterns but you first need to understand how you as a human would break it down step by step before you can instruct an AI to do so and then tip number three is don't underestimate what happens when you scale your application going from the demo to hey it works here is cool to now we're going to put this in front of hundreds or potentially thousands or millions of users is such a big difference and that chaos that we were talking about if you like scale chaos that is is just going to get insane and you will have hallucination so just don't underestimate that it's really hard so scaling rag is also really hard if you have your Factor database and you increasingly add more and more and more data that's also going to be a challenge on its own so be very careful with scaling your application and putting it in front of a lot of users too soon otherwise you'll end up with something that apple had with apple intelligence and the only way to do this systematically and properly is to have a proper testing and evaluation system in place which is tip number four start with this from the beginning don't neglect it a lot of people simply like don't know how to do this um and that's why they don't have it in place again ask yourself the question if you were to change your system prompt right now could you say for sure that it's going to improve your application Beyond just a f check this is something that you have to keep in mind if the answer is no you should look into that and figure out what it means for your application to set something up like this okay so the next one is to put proper guard rails

within your application and this is
actually really simple to do but a lot
of people just skip this step and it's
really a quick win so make sure before
sending your data or the output back to
the customer to the application you have
at least another llm do a quick check on
whether we can actually send this and
while it may sound very straightforward
this is something that for example a
company like Amazon didn't manage to put
in one of their customer support chat
Bots where if you would ask it whether
it was an AI or a human it would Clearly
say no this is a human you're talking to
and it would even give you a name but
then someone tried to ask the customer
support to give a code function back so
within that same trat first the said yes
I'm human and it was like can you
explain me how to build XYZ function in
Python and it would give just the whole
explanation and write out all the codes
and this is just like so embarrassing
and even the biggest companies out there
right now um fail to do this properly so
really put those guard rails in place
and perform various checks in order to
protect your brand and to protect your
reputation all right and then the final
optional tip that I want to give you the
other ones weren't optional this is
optional is if you want to learn more
about how we at data Lum structure our
projects and build these AI systems all
the way from how we structure our code
within the repository to the
infrastructure and deployments that we
use then you can check out the link in
the description to our generative AI
Launchpad we wrapped this into a product
and made it available in a giup
repository where this is essentially our
entire company's IP that we worked on in

the last two years and we make that available for AI Engineers who want to learn how to build and deploy generative AI apps and now this is a paid product just want to be transparent here but if you're serious about AI engineering and you want to speed up your learning join our community and get in our Discord with myself and the other engineers at data Luma to ask questions then you might want to check it out all right and that's it for this video now thank you very much for watching if you found it helpful please leave a like and consider subscribing and then next up I recommend to check out this video where I go over 17 python libraries that we use for our AI engineering projects