Transcript

MIKE: Thank you all for joining us. My name is Mike Davidson. My pronouns are he/him/his. I'm a Librarian here at the National Library of Medicine (NLM). I have a bunch of different jobs here at NLM, but one of them is what I'm doing today, which is developing and delivering training to help people learn about and learn how to use our products and services. And today especially, I get to talk to you about one of my favorite things to teach people about, which is APIs. Also helping me out with today's session is Kate Majewski who will be helping me answer your questions a little bit later on and Michael Tahmasian who is helping with logistics and production.

So a few housekeeping matters before we get into sort of the meat of today's session. We are recording today's session to make it available to those who are not able to join us today or for you to review and refresh your memory later on. We've also made the slides available for your reference as well as a handout with some helpful links and reference information. And we're going to be putting the links to both of those documents in the chat. For those of you who are interested in MLA CE credit that is available for today's session, just make sure you fill out the evaluation that pops up when you leave, as that will give you access to the CE credit. And even if you're not interested in CE credit, we encourage you to please fill out that evaluation anyway. They really help us fine tune our training offerings to meet the needs of our audiences, of you, so please use those evaluations as a way to let us know what else you want to see from us.

Today's session will be interactive and in a little bit I will be walking you through a demo that if you'd like to follow along with you are more than welcome. In order to follow along with that, all you'll need is a web browser. We're also going to be using Zoom's polling feature, which we'll try out in just a little bit, and we will also be using the chat panel. So if you have any questions, feel free to drop them in chat at any time and Kate will make sure that I address those questions at the end of today's session. We also from time to time will be having questions that we're going to be asking you, and we'd like you to answer them in the chat. And when you're using that chat, make sure your messages are sent to Everyone, not just Hosts and Panelists, so our whole team and everyone else can see them as well.

So as a practice round for getting getting comfortable using the chat, I want to start off with this. What is your favorite fast food place? Go ahead and answer right now in the chat if you would please. And while y'all are putting your answers in, I'm going to turn off my video to avoid distractions, both yours and mine. Oh man, got some great answers in here. We got Chipotle, Panera, Subway, Taco Bell, Cavo. That's one of my favorites. Portillo's. Oh, very good choice. Another Panera. Nando's. Another favorite of mine. Chick-fil-A. Oh, Zaxby's. I've never been to a Zaxby's, but I've heard good things. OK, well great. So we all more or less know where the chat is and know how to sort of engage there. So I think we should go ahead and move forward.

Before we get started, again, with the meat, here's what we are hoping to accomplish with today's session. By the time we're done, you should be able to explain what an API is, how APIs,

can help you interact with systems, and why you might choose to use APIs to accomplish certain types of tasks. You'll be able to describe the basic mechanics of using an API, and you'll also be able to identify some of the APIs NLM makes available for our products and know when a specific NLM API might be useful.

As I said a few moments ago, my job is to help people learn about NLM products and services, which is a pretty big job because, well, NLM has a lot of products. There are lots of different types of health and biomedical science information out there and lots of different types of people who need access to that information. So our portfolio to achieve our goal of connecting the one with the other is pretty broad. What you see on the screen is only a small sample of what NLM offers. When a lot of people think about NLM, they think about accessing biomedical literature via PubMed, PMC, or Bookshelf. Others think of us as a collector and provider of consumer health information via MedlinePlus. NLM also does a lot of work in the medical terminology space. We provide a number of databases and tools related to molecular biology or bioinformatics and we are also a great source for drug and chemical information. Even more, we are also the home of ClinicalTrials.gov which provides access to information about clinical studies conducted around the world plus much, much more than can be listed on this slide.

So let's take this opportunity to do our first poll. Michael will go ahead and open up this poll. And what I want to know is which of these categories of NLM products do you use? And you can feel free to check off more than one category. And if the products that you use aren't listed here, feel free to tell us what other NLM products you use in the chat. Let me give you a few moments to answer that poll question. [Pause.] All right, give you another few seconds here just to just to get your pull answers in. "Do CE sessions count as a product?" says Jeanette Bradley, it looks like in the chat. Training I think is a perfectly valid product. LocatorPlus. Yep. "Do Clinical Trials count as literature or product?" You could count them either way. All right, we've got some great answers. All right, let's close out that poll right now and take a look at those results. So it looks like literature is, unsurprisingly, one of the most overwhelming responses here. Also a lot of folks with terminology probably getting into the MeSH stuff there. Consumer health is also popular as well, but some good votes for the drug and chemicals category as well.

So while many of you probably access these resources through our websites every day, some of our users who are pursuing new and innovative ways to use NLM information need a different type of access. So let's talk about some of those special cases. My first example is one that you may have used yourself. Many clinicians these days, from hospitals to doctors offices, use Electronic Health Record Systems (EHRs) that have a patient portal where you can review your own chart, test results, and medical history. However, a lot of the information that's available in an EHR may be confusing to the average patient without further explanation by a clinician. One solution that many EHRs use is an info button. The patient can click the info button next to a lab test, medication diagnosis, or medical procedure and get more information in an approachable, easy to read format. MedlinePlus, NLM's largest consumer health resource, is full of just this sort of authoritative, up to date information, designed for patients, families and healthcare,

providers. Developers and providers of EHRs can configure their systems to call up relevant MedlinePlus information automatically when users click on one of those info buttons, which helps patients stay informed about their health.

Let's take a look at a second example, and this one is from the world of clinical oncology. I want to give a huge thanks to Philipp Unberath of Friedrich Alexander University, Erlangen–Nuremberg and his colleagues for this example. Increasingly, as genomic sequencing becomes cheaper and more available, clinicians are able to sequence cancer genomes and take a molecular based approach to cancer treatment. One tool used to help facilitate this molecular oncology approach to treatment is called cBioPortal. This is an open source platform that was developed by the Memorial Sloan Kettering Cancer Center, which helps clinicians analyze clinical and molecular patient data. But analyzing the data is only the beginning. What clinicians really need to do is turn that data into a treatment plan, and new treatments are constantly being developed and validated through clinical trials. In order to provide easy access to clinical trials that might enroll patients with a specific tumor profile, Philipp and his team integrated data from NLM's ClinicalTrials.gov directly into their instance of cBioPortal. Users can automatically see clinical trials that are registering new participants, and what's more, by using the patient's data from cBioPortal as a filter, the list of trials can be sorted and ranked, helping clinicians find studies that are the best match for a patient's age, sex, location, or specific cancer gene. All of this is done within the app, without the patient or clinician ever needing to visit the ClinicalTrials.gov website.

The third example I want to talk about is a research example, using literature citations themselves as data to better understand trends in scholarly publishing. This project was undertaken by a group of NLM associate fellows led by Luke Kudryashov to explore potential sources of bias among journal publishers. They specifically wanted to look at the nationality of authors and see if specific journals tended to publish authors from specific parts of the world more frequently. As you might know, author nationality is not a metadata field in PubMed, but author affiliation data is often available, and this data can often contain evidence of an author's home country. Luke and his team wanted to see if author nationality could be reliably identified using natural language processing and machine learning techniques on PubMed author affiliation data. They could then use those techniques to explore whether journals or publishers exhibited bias toward or against authors of certain backgrounds. But first, they needed a sizable set of PubMed citations, including that author affiliation, in order to train and test their NLP [natural language processing] algorithm. And what's more, they needed the citation data in a machine readable format to feed into the algorithm. Retrieving the requisite number of citations from PubMed would be rather cumbersome and time consuming using traditional methods, and doing so in a machine readable format would be even trickier. So the team explored other approaches of accessing PubMed data.

Now these are three very different projects, each accessing information from different NLM resources. We had MedlinePlus, ClinicalTrials.gov, and then PubMed. However, the thing these

projects have in common is that they each use information from NLM resources, but they don't use those resources' websites. In order to work the way they do and to be as successful as they are, they need a different type of access to these products. In the case of the first two examples, they need to access NLM information in another application, not in a normal web browser. They also need to retrieve their data without a human user interacting directly with the resource. They need to work automatically to some degree. All three examples need to access NLM information in a specific format, one which may not be available through the product's websites. And in the case of the last two examples, they need to access help information as data, not for a human to read, but for a machine to process and analyze.

So how do they accomplish all of this? As you may have suspected based on the title and subject of today's presentation, they do this using Application Programming Interfaces (APIs). Now before we get any further, I'm going to open up another poll just to get a sense of what previous experience with APIs you might have. Do you use them all the time? Maybe you use them periodically? Perhaps you've used them a bit in the past but not recently. Maybe you know something about APIs but haven't actually used them yourself. Or perhaps this is all new to you, which is totally fine as well, so I'll give you a few moments to put your answers in there. [Pause.]

All right, so since this is a quick question, we'll just give you a few more moments. [Pause.]

All right, let's wrap that one up and take a look at the results. OK, so we have a few people who use them all the time and periodically, some who have used them in the past. The majority there is Know about them but haven't used them. And then we also have some that this is all new to, which is again totally fine. Though I will say in some ways this is kind of a trick question. While you may not have worked with an API, you've almost certainly used one in some context. If you've ever checked a website to see if the local branch of a retail chain has a particular item in stock, or compared flight or hotel prices on a travel website, or used an app to get traffic updates or directions on a road trip or commute. If you've done any of those things, you've probably used an API. All of those systems use APIs to gather information from other online sources and present that data seamlessly to you, the user.

So what exactly is an API? It's a little tricky to define, since the term has been used to describe a number of different concepts in the world of computing and programming, dating all the way back to the 1960s. However, since a lot of the NLM APIs we're going to be talking about have some commonalities, we can come up with a good workable definition for our purposes today. So an API is a set of protocols for contacting a remote system and making requests. In other words, a set of protocols for interfacing with a remote system. APIs are designed to be used programmatically as part of a computer program, not directly by humans. In other words, an interface to be used when developing and programming applications, or an application programming interface. The APIs we're going to be talking about today typically include a computer somewhere, which we'll refer to as the server, that has information someone might want to access, plus a set of rules for making requests or calls to that server.

Now, this can all get pretty technical, so let's take a step back and talk about another way to think about APIs. Think back to your favorite fast food restaurant, which I asked about at the beginning of this session. If you want to order something from a fast food place, you can always park your car, go into the dining room, and order at the counter. That's like going to the website of your favorite NLM resource. Or you can go through the drive thru. The drive thru is like an API for your favorite fast food place. It is a quicker, streamlined way of getting the same stuff you get inside. When you pull up to the drive thru, you'll see something into the little intercom there. And based on what you say, you get a response back from the person on the other end. Either they tell you, oh, I'm sorry, I didn't understand what you asked for, please say that one more time. Or they tell you I'm sorry we don't have what you're asking for. Or hopefully they tell you to pull through to the next window to pick up what you want, you get what you want, and you never have to get out of your car. So APIs work basically the same way. Rather than going to the actual website, you or a computer program that you're using send a message to the remote system, to the server, requesting whatever it is you're looking for from that system. The remote system then responds. If you ask for something the system has available and you have formatted your request correctly, the system will respond by giving you exactly what you asked for.

APIs can save a lot of time and energy in certain situations because they're designed to be integrated into programs. Developers and programmers can incorporate these requests or calls to an API directly into their programs or applications, or even into other websites they're designing. Even better, developers can program the rules for properly creating API calls right into the application, and let the application create and send each actual request based on those hard coded rules. If you're making dozens or hundreds or thousands of requests a day, this is the only practical way to do it. In addition to being faster and easier, if a program can access data from an online database directly as part of its normal functions, it can do even more without needing human intervention. Rather than requiring a user go to the database's website, retrieve the information manually, and input it into the application, all of that process is handled automatically. Finally, some APIs provide access to data and information in different formats than the corresponding websites do. This could mean machine readable formats like XML or JSON that aren't always available to users through websites since they are less useful to the average human user. These machine readable formats may even include data that's not otherwise available through the website. The only way you can get it is via the API.

I don't want to get too technical about how APIs actually do what they do, mostly because I am not an expert in the technical nitty gritty myself. However, in order to see how APIs might help you or your users get things done, it can be useful to know a little bit about how APIs operate. So remember that the APIs we're talking about are composed of both a server somewhere, which is the remote computer you're trying to access, plus a set of rules or protocols for interacting with that remote computer. Going back to our drive thru analogy, the server is the specific fast food place you're ordering from, and the rules and protocols are like the menu board outside letting you know what's available to order. The way you or your computer

program interact with an API is via a URL. Now, these URLs look just like the kind of URL you'd visit in your web browser. However, the URL you use for an API call includes not only the address of the server for the API you're using, but also the details of what you're requesting from that API. What information you get back depends on how exactly you'd construct that URL.

We'll look at some examples in just a second, but for most of the APIs we'll be talking about, the URLs are made-up of two parts: a base URL which indicates which API you're using, and parameters which indicate what you're asking that API for. So as I said, first, there's the base URL. This is the address of the API's server. Think of this like the street address of the specific drive thru restaurant you're going to. Every time you go to that restaurant, it's going to be at the same address. Likewise, every time you use a specific API, the base URL for that API is going to be the same and tells your computer where to go to access the remote system for that API. You can see on the screen here a few base URLs, example base URLs from some of the NLM APIs we'll be talking about a little bit more later on.

The rest of the URL is made up of what we call parameters, and these include the details of what you're asking the server for. That can be things like search charms, if your request is about searching a database, or how many results you'd like to see, or what format you'd like those results in. This is like your actual order at the drive thru window, what you're actually asking for. And like at the drive thru, these will most likely be different for every request unless you always want to get the exact same results. On the screen here at the bottom, you can see an example string of parameters that I have used for retrieving some PubMed data from one of our APIs. Now in this case we have specified values for four different parameters separated by ampersands (&). We have a db parameter which equals (=) PubMed to indicate which database we want to retrieve data from. We have an id parameter which specifies a specific PMID for a record we want to retrieve. We have a retmode parameter which specifies that we want to retrieve results in XML, and a rettype parameter which equals full specifying that we want full records returned, not just summaries. Different APIs allow or require different sets of parameters, so you need to be familiar with how to use the parameters for the specific API you're going to be using.

So I know we've been talking pretty abstractly with a lot of metaphors about how APIs operate. So now let's take a moment to actually build and send an API request, and if you'd like, here's where you can feel free to follow along. In your own web browser, you can type this right into the address bar just like you would if you were going to any website. For this example, we're going to be using the MedlinePlus Web Service API. This API lets you search for and retrieve the information available via MedlinePlus Health Topics web pages, but in an XML format that can be more easily processed by machines. This allows you to embed this MedlinePlus health topic information on a website you're building. We'll start off with the base URL for the MedlinePlus Web Service API. It's https://wsearch.nlm.nih.gov/ws/query. The base URL for this API is always the same. Anytime you want to query the MedlinePlus Web Service, you'll start with the same base URL. And this base URL is on your handout, but make sure that you copy and paste it into

your browser's address bar instead of clicking on it in the PDF, because without the second part that we're about to talk about, the base URL won't do anything useful. And if maybe we could drop the handout link in there-- Oh great, Kate's already on top of it. All right, so that's the base URL.

Now let's talk about the parameters. Now, these are really important as they're going to tell the MedlinePlus Web Service server what we are looking for. If we don't spell it out in the parameters, it won't be part of our request and won't inform our results. If you're following along with the demo, before you put in any parameters, make sure you put a question mark (?) in immediately following the base URL. This is really important because it separates these two halves and lets the server know where the address finishes and the parameters start. There are a few parameters that are required by the MedlinePlus Web Service. These have to be in every call. The db parameter here tells the web service which database we want to look in, the English Language Health Topics database or the Spanish language database. In this case, we've chosen to look for health topics in English. Now, if you're following along, make sure that you separate this parameter with from the next one, with an ampersand (&). In between each pair of each set of parameters, you need an ampersand. For our second parameter, we need to specify what we're actually searching for. Now for this example, I'm looking for health topics pages about acid reflux. And if you want to follow along with a different search query, feel free. Just know that if your search query has spaces in it like mine does, make sure that you replace each space with a plus sign (+) and again an ampersand after this parameter to separate it from the next one. There are additional optional parameters that we can use with this API which can help define what format we want our results in or other options. For this example I'm using the parameter called retmax to specify the maximum number of results I want to see. So for this demo, I'm going to limit the query to just the first 5 results. If you want to follow along, you can copy the string of parameters that are in your handout and paste it into the address bar of your browser right after the base URL and the question mark. Or just stay tuned until the next slide where we put all of the parts together.

So building the whole URL together, we start as always, with the base URL followed by that question mark to separate it from our parameters. Then we list out each of our parameters and corresponding values in order, separating successive parameters with an ampersand. And there we go. We have now built a MedlinePlus Web Service API call, and we can just put this URL into the address bar of any web browser to query this API and see what we get. And what I got when I went to this URL in my browser was this big chunk of XML. This XML contains the top five English language health topics in MedlinePlus that matched our search for acid reflux. That's what we asked for when we specified those parameters, so that's what we got.

Now at this point, some of you might be asking, why would I do any of this? The MedlinePlus website has a perfectly nice interface where I don't need to know all of this fancy syntax and it will return actual web pages that I can read without having to be fluent in XML. And the answer is you probably wouldn't do this. Not the way that we just did it. This is an interesting demo, but

it is not an efficient way to use an API. The best uses of APIs are by programs or applications or scripts that have the URL creation rules built into them. That way they can take input from a user, build the corresponding URL according to those rules, request the information from the API, accept the result in a machine readable form like XML, and then process that data into whatever format the application needs in order to share with the user.

Remember, you can't just walk up to the drive thru window, you need a car and you need to know how to drive that car. In this metaphor, the car is a programming or scripting language which, if you know how to drive it, can be used to automate the calls to the API. To extend this metaphor a little further, you might be able to roll up to the drive thru on a bicycle what we did before, building the URL by hand and putting it in the browser. That's the equivalent of cycling up to the drive thru window. It technically works, but it's not particularly efficient, and it's definitely not the way you are supposed to do it.

So what kind of car, what kind of programming language, do you need in order to make use of these APIs? Well, fortunately pretty much any modern programming or scripting language will be able to do it. If you're a developer, or you're already working with a developer, or you know a little bit of programming, whatever programming language you're already using will almost certainly be fine. If you're new to programming, you may want to look at either Python or R. Both of them are not easy to learn, but on the easier side to learn. Also, both have prebuilt tool kits called packages of commands designed to make some common tasks like using APIs easier. These packages can streamline the process of working with APIs in your program and make it so you don't have to worry as much about the mechanics. Sort of the equivalent of an automatic transmission in your car. It doesn't do anything that a manual transmission can't, it just makes the whole process easier. Some programming languages even have packages designed to work with specific APIs. So for example, if you're using the R programming language, the "rentrez" package, that package is designed specifically to help with the NLM E-utilities API. That's an API that provides access to data from PubMed, PMC, and other NLM products. And that package is designed specifically to help you use it. Just to be clear, the amount of programming required to use an API might be quite limited. We're not necessarily talking about developing a new iPhone app or creating a plugin for a clinical oncology platform. As we looked at in our earlier example, just creating a very rudimentary script that automates data retrieval using an API may be all that you need to achieve your goals. However, for some folks I understand any amount of programming may seem like too much programming. So this segues nicely into my next point.

If you don't know how to drive just yet, you have a few options. If programming or scripting is new to you, but you're interested and have a little time and patience, it may be worth your effort to learn. However, there is an alternative which is get a friend to drive you. If you aren't yourself a programmer, but you can find someone you know who is, you can ask them for help in building a project that uses an API. This may mean that you don't have to learn how to program yourself, but it can still be helpful to learn about APIs, how they work, and which APIs are available that might help you do what you want to do with NLM information. If you learn

which API options are available, you can better communicate with your programming friend about what you need using language that might be easier for them to understand. Going back to our drive thru analogy, if you can give your driver the address of the fast food place and good directions on how to get there and tell them exactly what you want to order, it makes the whole process go smoother. At the very least, if you can point your programming partner to a specific API that can access the specific data that you need, you can save them some time researching on a subject they might not be as familiar with.

All right, let's take this opportunity to do another quick poll just to see where we're all at in terms of prior programming experience. Let me know if you're familiar with any of the programming languages listed, and please feel free to select more than one option. If you have experience with the language that's not listed, please let us know that in chat as well. And if you haven't really had any programming experience yet, yeah, that is fine. Remember, you can always find a friend to help drive you where you need to go. [Pause.] I've seen a couple of votes for Java in the chat. Yes, Java, the one on that that should probably have been on this pole but was not. I've taken some Java classes myself but it has been a long time. BASIC and Pascal. I started out learning BASIC back in the day. All right let's close out this poll see what we got here. All right. A bunch of votes for Python. That makes sense. R looks like he's in second place. PHP or JavaScript, for those who've done any web development that that makes a lot of sense as well. Oh we got some got some shell scripting people in there too. That's one that I don't see as much of. But I I always I always find that super useful myself. And as I said, we had some votes for BASIC and for Java in the chat, so also some votes for HTML and CSS which that coding is useful, but without some sort of programming element like PHP or JavaScript that might have a little bit more difficulty using the APIs. But fortunately the JavaScript stuff does work really well with that in order to automate API calls if you're trying to pull in information as part of a website. All right, let us move along.

I want to spend most of the rest of this session going through some of NLM's most popular and well known products and then talking about some APIs that you can use to access those products information. This isn't going to be an exhaustive list of NLM APIs, however, it should hopefully serve as a sort of sampler platter, giving you an idea of what sort of APIs are available to help you interact with NLM's resources.

As you consider the options I'm about to talk about, it is important to remember that APIs are not one-size-fits-all. You can't just use any API you choose to access whatever remote system you want. Just like you can't go through the Starbucks drive thru and order a Big Mac, or through the Taco Bell drive thru and order a Pumpkin Spice Latte. You have to go to the drive thru that offers the food you want, and you have to use the API that offers the data you want. The first step in choosing an API is always going to be looking for one that can access the data that you need. In some cases, an online database might have multiple different APIs that can each retrieve the data in different ways or formats, and then you have some choice. We'll look at a couple examples of this in just a few moments. I will also mention that on that handout that

that we put a link to in the chat before, there are links on that handout to more information about each of these APIs. So if one of them catches your eye, you can go check it out in more depth after today's session.

So since we've already talked about MedlinePlus APIs a few times during this presentation, we're going to start there. You might remember our example of building the URL that was using the MedlinePlus Web Service API, which lets you search for and retrieve health topic information in XML. MedlinePlus also makes its data available via an API designed specifically to be used by electronic health records (EHRs) and patient portals. MedlinePlus Connect is a service that EHR developers can use to provide free, authoritative, up to date consumer health information to patients based on specific medications, procedures, lab tests, or medical conditions that are in that patient's health record. MedlinePlus Connect is the API used in the example that I talked about at the very beginning of today's session.

We can't very well talk about NLM products and services without talking about PubMed. In 2023, over 480 million users accessed PubMed's website and performed over a billion searches. However, that number does not even include API usage. If you include PubMed searches conducted via the API, you add an additional 2.5 billion searches on top of that on top of the billion-plus that were done via the website, and the overwhelming majority of these API searches were done using E-utilities. E-utilities is a suite of APIs that provide access to over 35 different databases and products developed and maintained by the National Center for Biotechnology Information (NCBI), which is the division of NLM responsible for PubMed. You can use E-Utilities to search PubMed and retrieve records in a variety of formats, including the full PubMed XML, which includes some data that isn't otherwise exposed to users on the PubMed website. E-Utilities can also let you grab lots of PubMed records quickly in a machine readable format, so you can treat PubMed citations like data. This is critical for those doing bibliometrics research and portfolio analysis. The example that we talked about earlier where the NLM Associate Fellows were trying to identify patterns in authorship nationality, they used the E-utilities API to retrieve a large number of PubMed records in XML format, which they then used as the training data for their machine learning project.

In addition to E-utilities, there are two other more specialized APIs that you can use to access PubMed data in specific ways. The Literature Citation Exporter API can be used to convert PMIDs or PMCIDs into formatted citation strings in a variety of citation formats. So if you need a way to quickly and automatically generate those strings, that API may be helpful for you. The sort of flipside of this is the Citation Matcher API that takes the citation matcher technology embedded in the search box on the PubMed website and makes it available for developers to use programmatically. Using this API to search for a citation string will return the set of PMIDs most likely to match that string.

Sticking with the literature theme, let's talk about two other NCBI literature products, PubMed Central (PMC) and Bookshelf. PMC is NLM's free full text archive of biomedical journal literature, while Bookshelf provides free online access to books and other documents. You can also use the

E-utilities API with both PMC and Bookshelf. These are two of the other 30-some databases that the E-utilities API can retrieve data from. However, there are some other API options for these databases as well, which are particularly suited for retrieving full text from articles in the PMC Open Access subset or full text from books in the NLM literature archive Open Access Subset. These APIs are built using a standard called Open Archives Initiative Protocol for Metadata Harvesting, which is almost always abbreviated OAI-PMH. OAI-PMH is a standard used by lots of different online digital depositories for their APIs. Now this standardization means that if you can find a program that's already written to use one OAI-PMH API or a programmer that's familiar with that standard, it's much easier to adapt that work and knowledge to retrieve full text from the PMC and Bookshelf Open Access collections.

Now let's change gears a little bit and talk about some of the more terminology focused products, starting with Medical Subject Headings (MeSH). Many of you probably already know that many PubMed records are indexed with MeSH descriptors to help improve discoverability. Some libraries, including NLM, also use MeSH for descriptive cataloging. If you want to explore or download the MeSH vocabulary programmatically, you have a few different API options. If you'll recall the E-utilities API, which we talked about on the last two slides, that one works for MeSH too. However, because of the MeSH vocabulary's hierarchical nature, the E-utilities API is a little bit limited in how it can retrieve MeSH data. This is one of the reasons why we can't have a one-size-fits-all approach for APIs. The nature of the data we want to retrieve varies from resource to resource, so the tools and protocols we need to retrieve it have to vary as well. Fortunately, NLM provides another API called MeSH RDF which helps you query and retrieve mesh vocabulary information in a different way. Some of you may be familiar with RDF, which stands for Resource Description Framework. This is a standard that represents metadata as linked data. The MeSH RDF API can be used to query a linked data representation of MeSH. Using MeSH RDF is a little bit more complex than the other APIs I've discussed, as you or your programming partner need to be familiar with querying linked data, which is a slightly different skill set than say, searching PubMed or MedlinePlus. However, if you're looking for a way to dive deep into MeSH, MeSH RDF may be the API you need.

While we're discussing terminology APIs, I also want to give a brief mention to RxNorm and the RxNorm API. It may not be quite as well known as some of the other products, but it is a great resource with a particularly useful API. So as you may know, there are some unique challenges in designing computer systems to keep track of prescription drugs. The same medication can be called by different brand or generic names and may be identified by different codes in different drug terminologies. RxNorm provides a terminology of clinical drug names and codes that also serves as a crosswalk between many other drug vocabularies, including First Databank, Micromedex, and Gold Standard. Developers of pharmacy management software, hospital information systems, and even clinical and public health research tools can integrate the RxNorm API into their projects to ensure that their applications can keep clear and consistent track of each individual's medications, even if the drugs are entered using different naming schemes. In fact, the FDA has integrated calls to the RxNorm API into the workflow for their

Adverse Events Reporting System. By using the RxNorm API to map medications in Adverse Event Reports to standardized names, they were able to better identify patterns even when the same drug was referred to by different names in different reports. The reason I wanted to mention the RxNorm API specifically is that there's a particularly useful tool called RxMix that can make it easier to get started using the RxNorm API. RxMix, which you can see a screenshot of here on the slide, is a website where you can experiment with the different features of the RxNorm API. You can try out API calls and workflows and figure out how to make the API work well for you. Once you've found the right API calls for your project, you can take what you've learned and program it into your application. This makes it easier when developing and testing a project to tell whether any problems you have are in how you formed the API calls or maybe somewhere else in your code.

We talked about ClinicalTrials.gov earlier on, specifically when we looked at the example about clinical oncologists using cBioPortal to explore treatment options based on genetic tumor profiles. That project uses the ClinicalTrials.gov API to help clinicians find possible clinical studies which might be a good fit for their patients. The modified version of cBioPortal that they have uses the relevant patient information to formulate a search request to the ClinicalTrials.gov API. The results, which the API sends in a machine readable format, are then ranked by cBioPortal and converted into a human readable interface for the clinicians to review. Like with RxNorm, the ClinicalTrials.gov API also has a web-based graphical user interface that you can use to develop and test out your API calls.

So now that we've run down the menu of NLM APIs what looks appetizing, we're going to put up another poll here and I want you to let me know which of the APIs we've talked about are you most interested in learning more about or working with. Just pick your number one top choice, and if you're still not sure why or how you would use APIs, there is an option for that as well. [Pause.] I'll give folks a few more moments to pick their API that looks most interesting to them. I think we can wrap that up. All right. Yes, E-utilities, very popular choice that makes a lot of sense because E-utilities can access a lot of different databases. But I see my evangelizing about the RxNorm API has gotten some people interested there and MedlinePlus, that's another one, that's one that's kind of easy to work with. And if you need consumer health information, that's a really good choice. But I do see that we have some folks who are still not quite sure why they would use an API. Totally understand that.

So as we head toward the end of today's session, I want to go back over a few key concepts that we've covered. This is some of what I hope you take away with you today, aside from a deep craving for fast food. Sorry for those of you who this session is before lunch, on the East Coast I had my lunch beforehand. But this wrap up might also help those of you who aren't yet sure when or why you would need to use APIs. APIs (Application Programming Interfaces) are great ways to retrieve data from NLM resources. In order to make the best use of an API, you probably want to be using the API in the context of a program, script, or application that you're developing, as APIs are designed specifically to be used programmatically. APIs often provide

access to data in machine readable formats like XML or JSON. Since APIs query the same databases available on NLM's websites, they're great ways to get the most current, up to date data quickly and on demand. Also APIs are useful for when you're searching for or requesting specific things from a database, in other words, when you know more or less what you're looking for. Putting these last two points together, APIs are great at getting you exactly the data you need, exactly when you need it.

Now, despite how cool and useful I think APIs are, there are still sometimes when using API is probably not going to be the right choice. First of all, and this may seem obvious based on the last slide, but I do want to underline it, if you aren't programming or aren't working with someone who's programming, APIs are probably not the right choice since they work to their best advantage when integrated into programs or scripts. Also, as I've said many times before, APIs aren't one-size-fits-all. You have to use an API designed specifically for the resource you are trying to retrieve data from. If the resource you're trying to access doesn't have an API, you can't use an API to access that resource. Fortunately, as we've already discussed, many NLM resources are accessible via their own specific APIs. If you're not exactly sure what you're looking for in a resource and are still exploring or browsing or getting to know that resource, you may want to stick with the web version until you're a little bit more familiar with it. And the reason I say this is that because APIs require precise syntax to make specific requests, and because the output is often in machine readable, not human readable formats, you're often better off starting your exploration in the web interface and waiting to start building API requests until you have a better idea of what you're looking for. Finally, while APIs are really good at retrieving data in response to specific requests, if you're working on a project that needs access to the entirety of a database's data, like for example, if you want to do an analysis of all of PubMed's 37 million plus citations, retrieving that data via API may be a long and cumbersome process. Many APIs have throttles or gates to prevent people from abusing or overusing the system. These restrictions can make it difficult to do really massive bulk downloads.

However, there may be another option for projects that require you to download all of the data in an NLM database. It's not really the point of this presentation, so I'm just going to touch on it briefly. The data from many NLM products are available in bulk downloads via FTP. This includes PubMed data, MeSH, NLM catalog records, the Open Access subset of PMC, MedlinePlus Health topics, ClinicalTrials.gov records, and much more. For the right project, these bulk downloads can be invaluable, but they can take quite a while to download and require a lot of space on your computer to store. Also, in some cases you'll need to set up your own database system to be able to search and query these data sets. Depending on your project, you'll also need to consider how often you need to stay up to date. The bulk downloads are a snapshot of the database at the time you download them. If your project requires the newest data, you'll have to continually download the latest updates. Going back to our drive thru metaphor, bulk downloads are kind of like the equivalent of going to the grocery store, buying all of the ingredients, then bringing them back home and cooking the meal yourself. You might well end

up with something way tastier and better than what you can get at the fast food place, but you have to do a lot more work to get your meal, and you'll probably have to go back to the store again next week to get fresh ingredients.

So hopefully this presentation whetted your appetite to learn more about and maybe even start working with NLM APIs. If you're trying to figure out what your next step is, here are a few suggestions. If you aren't already familiar with programming or scripting, that might be a good place to start. There are plenty of online courses that teach the fundamentals of R or Python, including the basics of how to use those languages to engage with APIs. Another good starting point is Library Carpentry. They host intensive multi-day workshops that cover a range of data and technical skills aimed at librarians and information professionals. Some of these workshops include an intro to R or to Python. So look at these workshops, take a look at the curriculum and see if you can find a workshop that works for you. Also ask around at your institution to see if basic programming classes are available, or if perhaps one of your coworkers already has the programming skills you need. Something else that can help if you have a project in mind is thinking very carefully about the project before you get started. Identify what information you already have going in, be it a search term, an author's name, a set of clinical trial criteria, list of PMIDs, whatever it is you have already, and then also identify what information it is you are trying to retrieve. Because API requests require specific syntax to retrieve specific information, a little planning ahead of time can save you a lot of trial and error work in formulating the right request. And finally, make sure you've identified the API that is actually relevant to your project.

We've talked about a bunch of APIs today that provide access to NLM data, but there are many more as well. To help you find the right API for you, you may want to check out NLM Data Discovery. The Data Discovery portal serves as a directory of NLM data sets and access points, including APIs and bulk downloads. Additionally, some NLM data sets that don't have dedicated APIs can be retrieved directly from Data Discovery using the web portal or Data Discovery's own API.

Once you've found the API you think will work for you, settle in to spend some time reading the APIs documentation. Now these docs can be dense and occasionally confusing, but often contain a wealth of genuinely useful information. API documentation usually tells you what an API can and can't do, so you'll know if you're spending your time on the right API. Docs will also outline the syntax for creating an API request, including the base URL and all of the possible and required parameters. Most APIs have guidelines or restriction for usage which are also documented. These guidelines prevent the system from being overloaded with too many calls at once and help protect our resources against malicious code. Finally, most API documentation, at least most of the good ones, include example API calls you can review, modify, or pop into a web browser to test out. The handout that we distributed has links to help you find the documentation for all of the APIs we've discussed today.

So that's most of what I wanted to cover today. But we also want to hear from you about what else you feel like you need in order to get started with APIs. We'll throw up this poll, but if

there's other things not listed here that would help you get going with APIs go ahead and put them in the chat. Also if you have any questions, hopefully you've been putting them in the chat right along, but go ahead and throw them in there as well and we can get to answering them after I take a sip of water. [Pause.] All right, let's take a look at that poll and we can get into some questions here. I see somebody in the chat says courage. I totally understand that in terms of what you need in order to get started with APIs. That is I think the biggest stumbling block that I had starting out and that a lot of other people have I've also seen starting out. One of the things that I always tell people when I'm teaching them skills like this is assume that it can be done. A lot of things can be done. APIs can do a lot of things. Programming can do a lot of things and you can do a lot of things. So assume that it can be done and then go out and you know you'll be proven right probably. Oh, I should turn my video back on. All right. Let's wrap up that poll and then we can get into some questions.

All right. Help with programming and more examples of APIs in action that seems like the most common needs which again totally understand. There are examples out there and we're working to collect more of them. Hopefully the ones that we shared with you today were helpful, but there are also others as well and we are definitely working to gather more of those. And as far as more information on available APIs, I would say Data Discovery is a place to check for that because they're all listed there and then programming help we already talked about a little bit. So all right, Kate, do we have any questions?

KATE: All right, yeah, well while folks are formulating their questions, we do have one that came in earlier from Tian and it's an excellent question: **Are there security concerns with using APIs? From what it sounds like APIs only allow one directional information transfer. For example, I can only pull data from the database, but can data be pulled from my end?**

MIKE: Interesting question. There are APIs that are bidirectional, and I guess all APIs are bidirectional it to a degree right? Because you are sending a request, so information is going out that way. But one of the things that's nice about APIs is that they are targeted, right? You are sending a request to do a specific thing. So unless you are sending more information than you want, there's there is generally not a way for the API to pull additional information from you. It is entirely reliant on what you are sending it. Hopefully that answers that question.

KATE: Alright. And **another question is specifically about the ClinicalTrials.gov API, about how in using the API I get a lot of excess information I don't need. Is there a way of just pulling what I need or do I have to take all the information and sort through it on my end?**

MIKE: So I have not used the ClinicalTrials API myself that much. I've dabbled with it a little bit. There's a couple of ways that you can go about this. One is see what parameters are available for limiting results but if what you're talking about is sort of accessing specific data elements as opposed to like an entire record sometimes that is available from APIs, often times it is not because the assumption is that these APIs will be used as part of a program. So I think the assumption is, you know, we'll send you all of the data and you sort it out. That's sort of an

oversimplification. But hopefully, if you're trying to retrieve specific information and you're doing this from the context of a script or program and you have those machine readable data, either XML or JSON or something like that, the data should be well structured enough that you can use additional parts of your program to pick and choose the bits that you want and display it in the way that you want. So when I use the E-utilities API for example to retrieve PubMed information, by default the E-utilities API is only going to give me back XML or JSON or plain text. But if I take that XML and run it through sort of an XML parsing processor to parse out the bits that I want, that way I can get sort of more specifics, a more specific set of data arranged in a way that I want. Hopefully that answers your question great.

KATE: And I'll just tag on there that I have heard from the ClinicalTrials.gov staff that they have heard that request in the past or several requests in the past to offer more flexibility in their API to only extract certain fields. So they're aware that people are asking for that and so they're putting it on their wishlist and discussing it. So thank you.

OK, so then **there are a couple of questions/ comments about grabbing data from PubMed to post on library websites or institutional websites**. (Sorry, I'm going to post evaluation for folks who have to leave. Let me just get back to the question here.) OK, so **can you speak to pros and cons of using the API versus the RSS feed to push PubMed results to a web page?**

MIKE: Hmm, that's a good question. For PubMed results specifically, right? So my gut reaction to that question is the RSS is easier but the API is more flexible. Now I have to like dig into that a little bit more to be sure of that. But because there's a lot of tools already designed for processing RSS feeds. The RSS is already presenting the information in a specific arrangement and sort of just sort of sharing it up for you and then you can grab that. That makes things easier, right? There's a lot of things that are already working to make that an easier process. If you're talking about like LibGuides, I think there's a LibGuide module for RSS feeds, so like that's easier. But if the RSS feed is not giving you what you want, I know that one thing that that many people have sort of not complained about but have sort of asked for from the PubMed development team over the years is more customization with the RSS feed. That's not sort of technically feasible for a number of reasons as far as I understand it. But the API is sort of the answer to that, right? It means the API, you can ask for whatever you want, whatever, search and get the results back, you know, process those results and display them however you want. It gives you that additional flexibility.

KATE: Great, thank you. Question from Katie who asks **would you think about offering a workshop just on RxNorm database API in the future?**

MIKE: Interesting. I guess we could certainly consider it. One of the things that we have to balance is sort of demand, you know if there's a groundswell of support for that that's certainly something we consider. I love RxNorm, I love talking about RxNorm, and their API is really interesting. One of the things about RxMix is, as I said, they have those sort of workflows that you can sort of test out in a web browser which I think makes it a really easy one to get to start

using. The sort of downside to that is that the use cases for RxNorm are a little bit more specific, at least for at least for librarians, which is a lot of our usual audience. But it's certainly something we could we could think about.

KATE: Curious, do you think that the one hour RxNorm class would be useful for someone who is looking into using the API?

MIKE: Absolutely. I would say that yeah, I don't know if you if you can dig up a link to that but the-- oh there you go (https://www.nnlm.gov/training/class-catalog/drug-terminologies-and-rxnorm).

That goes along with what I was saying before about make sure you know how to use the resource before you start trying to use the API. And it's not just a sort of walk before you can crawl thing. It's because a lot of times if you are sort of not as familiar with the resource and how to use it and what's in it and like sort of how RxNorm works and like what the, you know how the data in there is arranged, you'll get really frustrated trying to make API calls and not understanding how your results are coming out or why you are getting certain things in a certain way, why you're not getting other things. That's why I always say like make sure you are very familiar with the resource itself and then go explore the API to sort of you know go up the next level and access things in a different way.

KATE: Great. It's great to hear about interest in that. Wonderful. I don't see any other questions at this point.

MIKE: We have a few more minutes. So if anybody has any last questions or thoughts-- I did see one thing and I'm not as familiar with this so I'm not going to be able to give you a solid answer, or as solid an answer, but **somebody was mentioning LitSense a couple of times which is I think a newer experimental NCBI search system**. I'm not, like I said, I'm not as familiar with it, but it does have an API. So if you are interested there is a link on their page to information about their API. And one of the things because again it's a newer resource and the search is resource intensive, they are limiting usage to one request per user per second and that's a great example of one of those sort of throttles or gates that they don't want to overtax the system. So hopefully that was of use to whoever was asking about LitSense. I guess that's probably it.