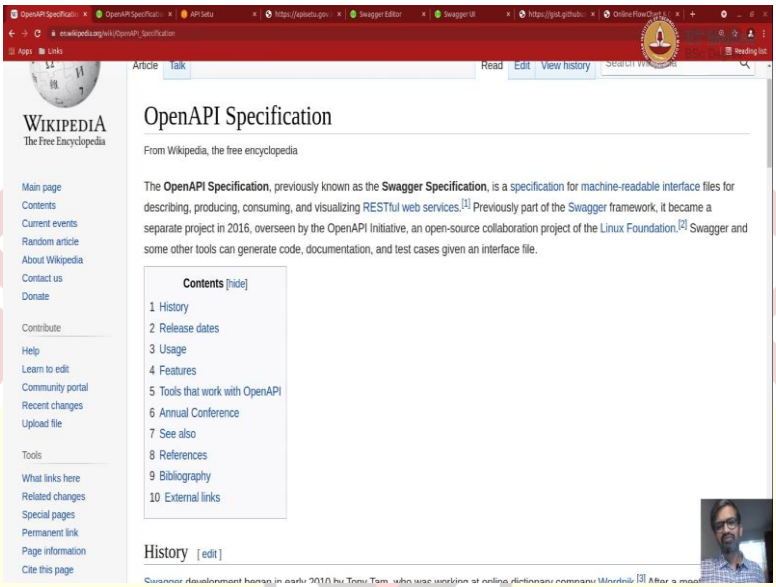
IIT Madras

ONLINE DEGREE

Moderrn Application Development – I
Professor Thejesh G. N.
Indian Institute of Technology, Madras
Documenting Open API

(Refer Slide Time: 00:15)

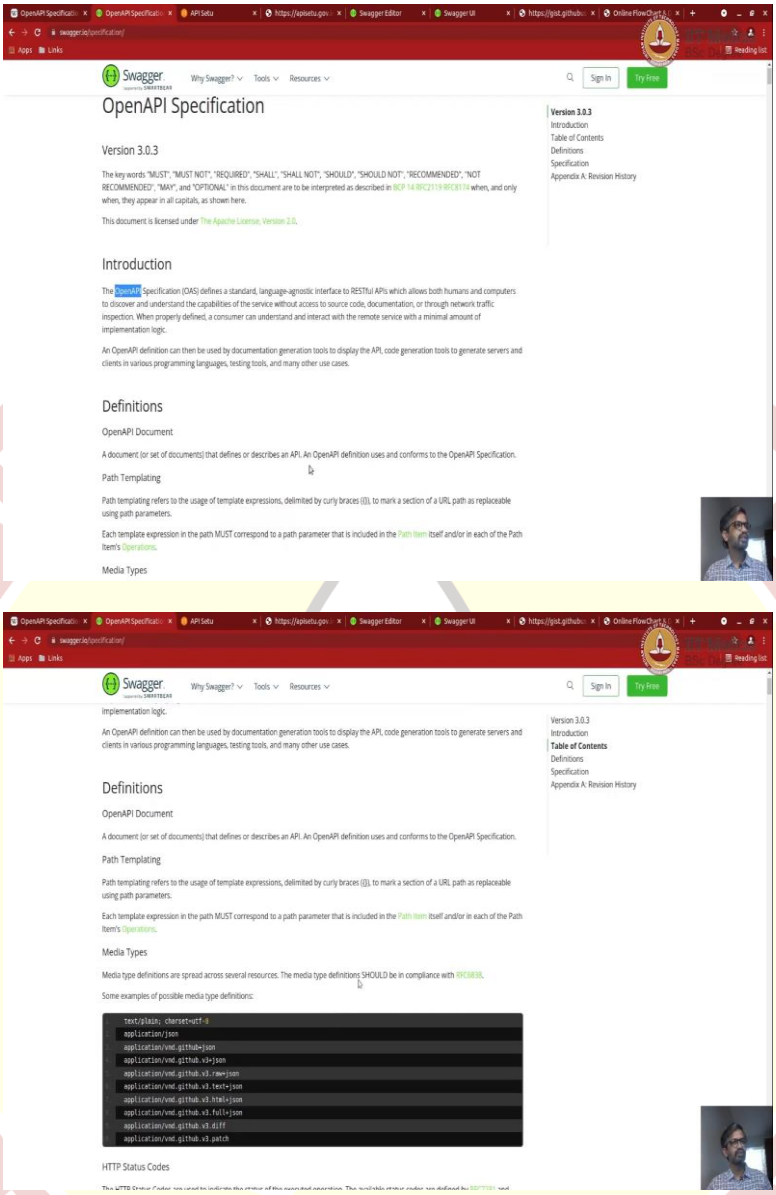


Welcome to Modern Application Development screencast. In this short screencast, we will learn how to document a RESTful API using open API specification. Also, we will try and explore some of the tools that we can use to explore the open API specification, so to execute some of the API calls. Okay.

First of all, let us go with our definition of what is open API specification. We can just get it from Wikipedia or any other place. It is just a machine readable format for describing producing and consuming, as well as visualizing RESTful web services, you know, if you are actually developing a RESTful web service, or you are designing a RESTful web service, then you need a way to document it, so that you can share it with the rest of the folks. And you know, open API specification is one such format that you could use to document it.

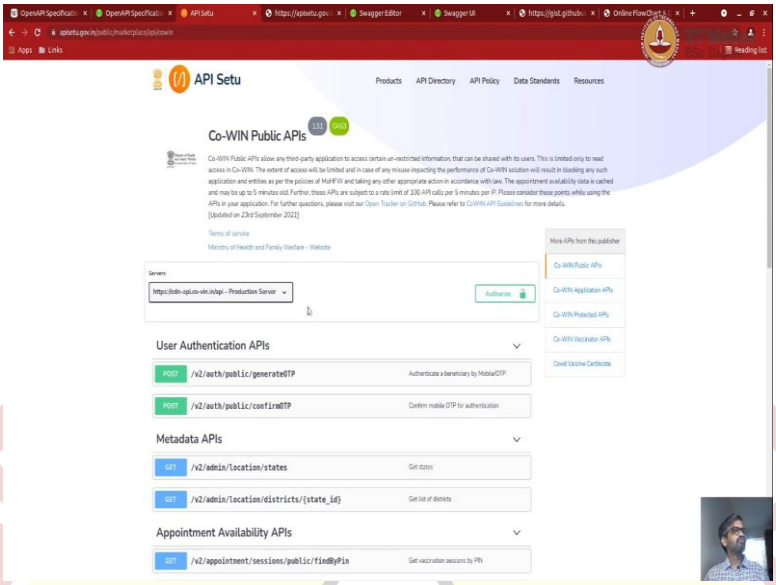
It beyond documentation, it can also be used by, you know, a machines to actually consume it, and do certain other things. It can even, automatically generate client codes or server side codes, based on the specification itself. There are many other uses to it, I think you can explore more. The specification itself was called swagger specification earlier, and then renamed to open API specification.

(Refer Slide Time: 01:59)



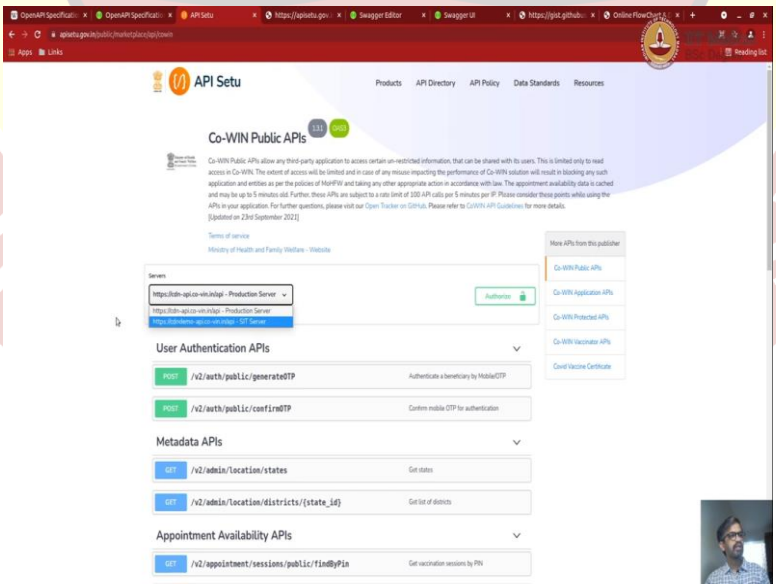
Currently it is in version 3.0.3. And here, you can find the details of it, if you really want to go through it here. Usually expressed as you know, JSON files are in yaml files.

(Refer Slide Time: 02:17)



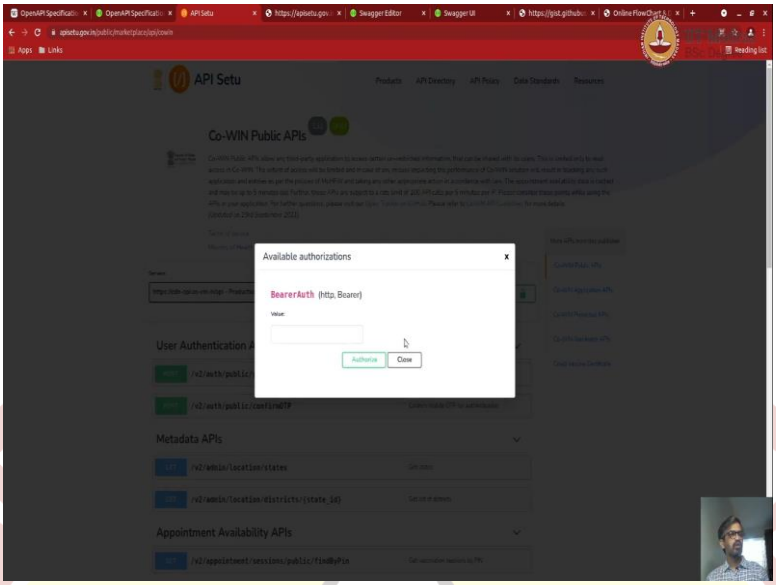
Let us see one of the API specifications. Most of you would have heard about CoWIN API's, you can see that the API specification or the documentation is open and is accessible at this URL. And you can see that the new API version is 1.3.1. And specification surface, we see OAS3. And you can read more about it. Let us just explore some of the particulars of this documentation.

(Refer Slide Time: 02:56)



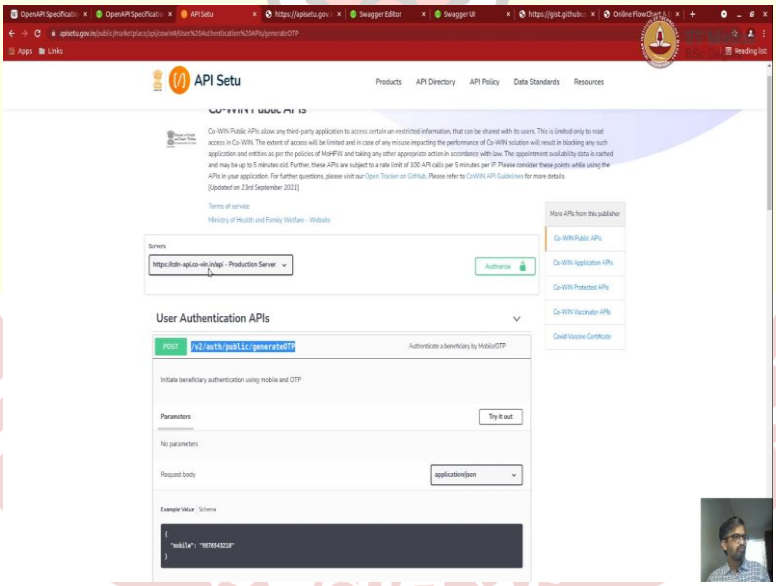
Here you can see that you know, the servers available are two servers, against to which you can run the API's. And there is some kind of authorization that is required to run the API's. You know, you can click on authorize, okay.

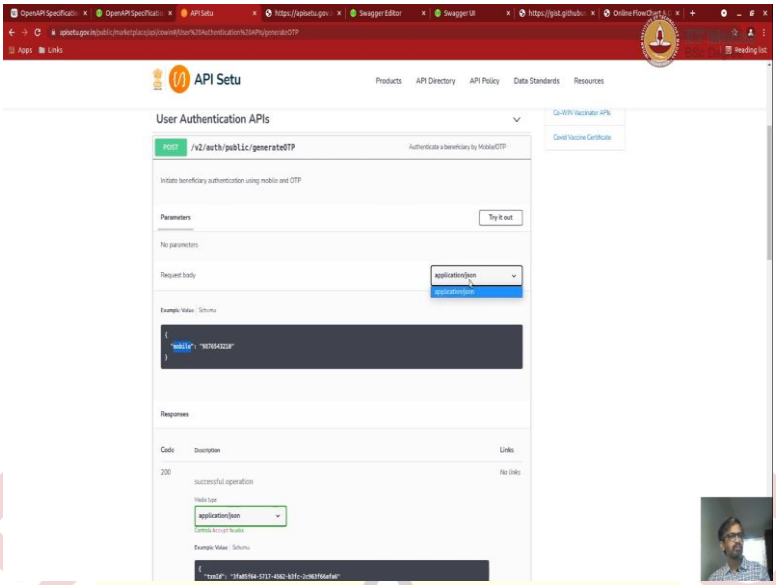
(Refer Slide Time: 03:05)



It requires a BearerAuth in the header to run the API's. Each API's is defined, method is defined, path is defined, path is attached to this path,

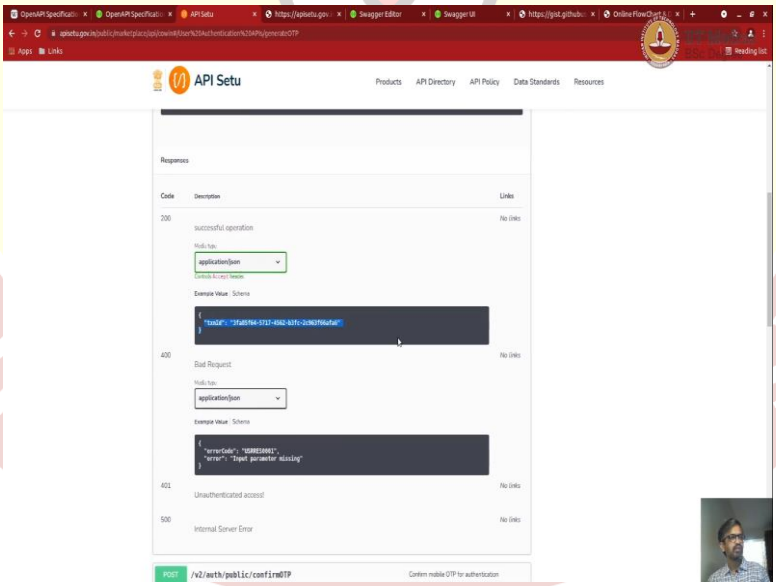
(Refer Slide Time: 03:21)





If you have https://cdn-api.co-vin.in/api /v2/auth/public/generateOTP is the full path. And then it has a description. It has a method called POST, whatever input parameters, if there are any input parameter, examples, you know, it is a JSON, takes the mobile number. You can see here the input type is application JSON.

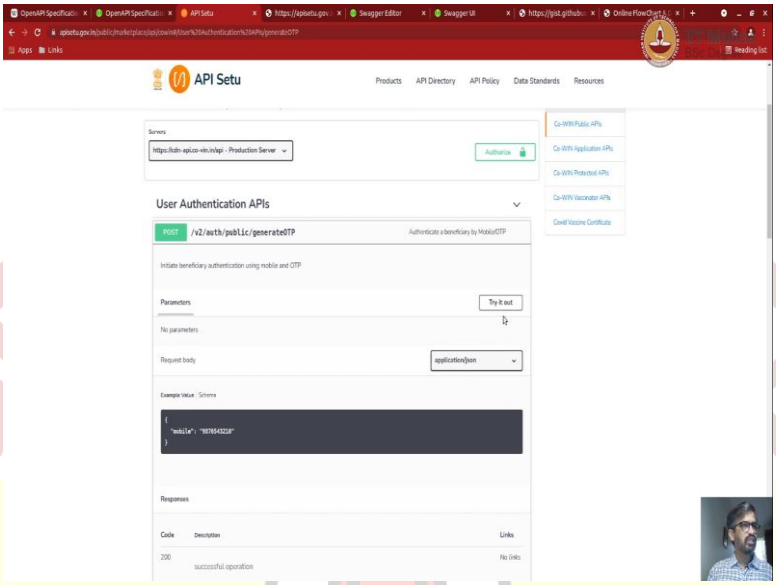
(Refer Slide Time: 03:51)

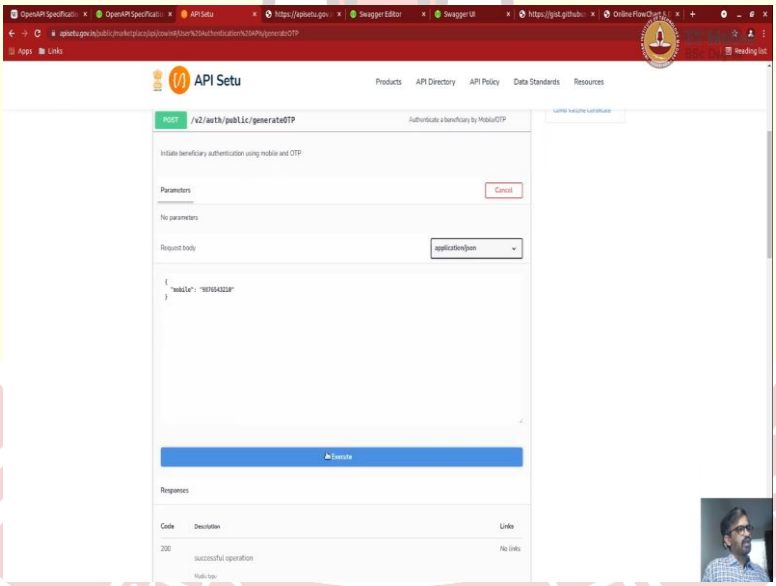


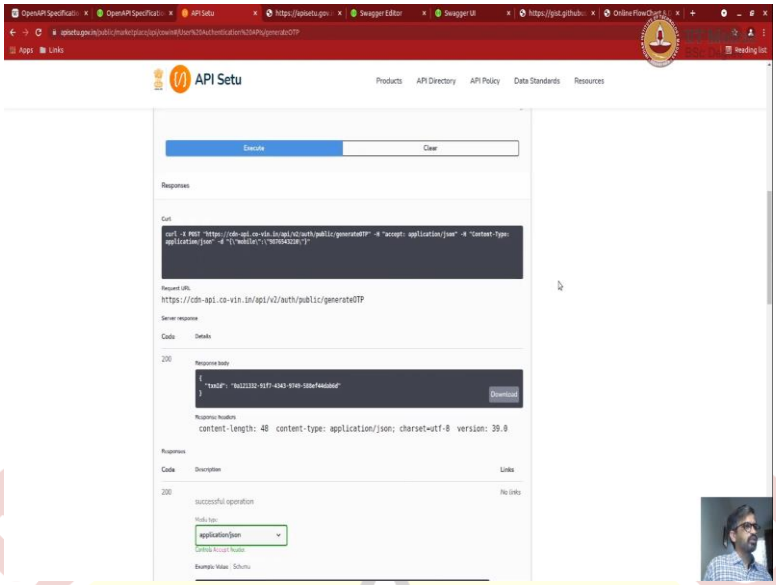
And the responses. If everything worked well, it will return 200 with you know, successful operation and also will return a transaction ID as a response. And the response is application/ json. Similarly, if there is something wrong with the input, it will have 400 as a status code, and it will return our application JSON, but it will return an error code and an error message or here with error, called ‘error’. Similarly, 401, if it is not authenticated, and 500, if it is an

internal server. This is for one of the POST requests and you can even try it out, if you had a real authorised key.

(Refer Slide Time: 04:31)

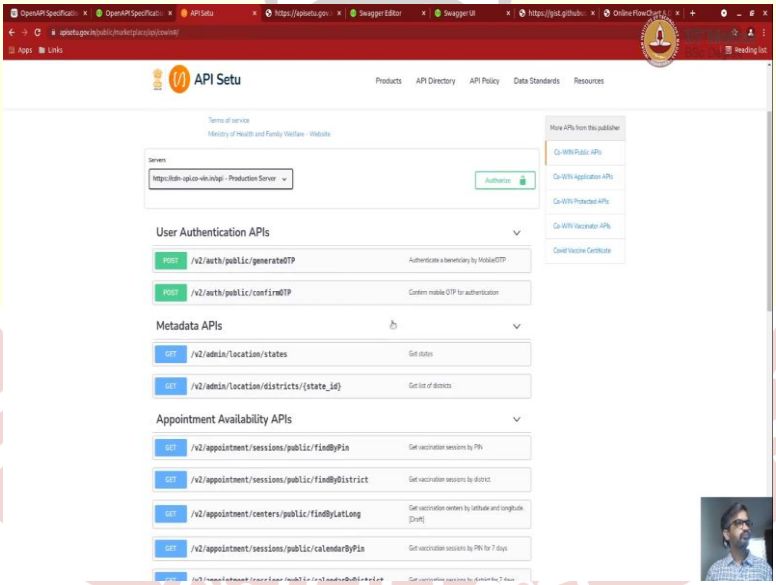






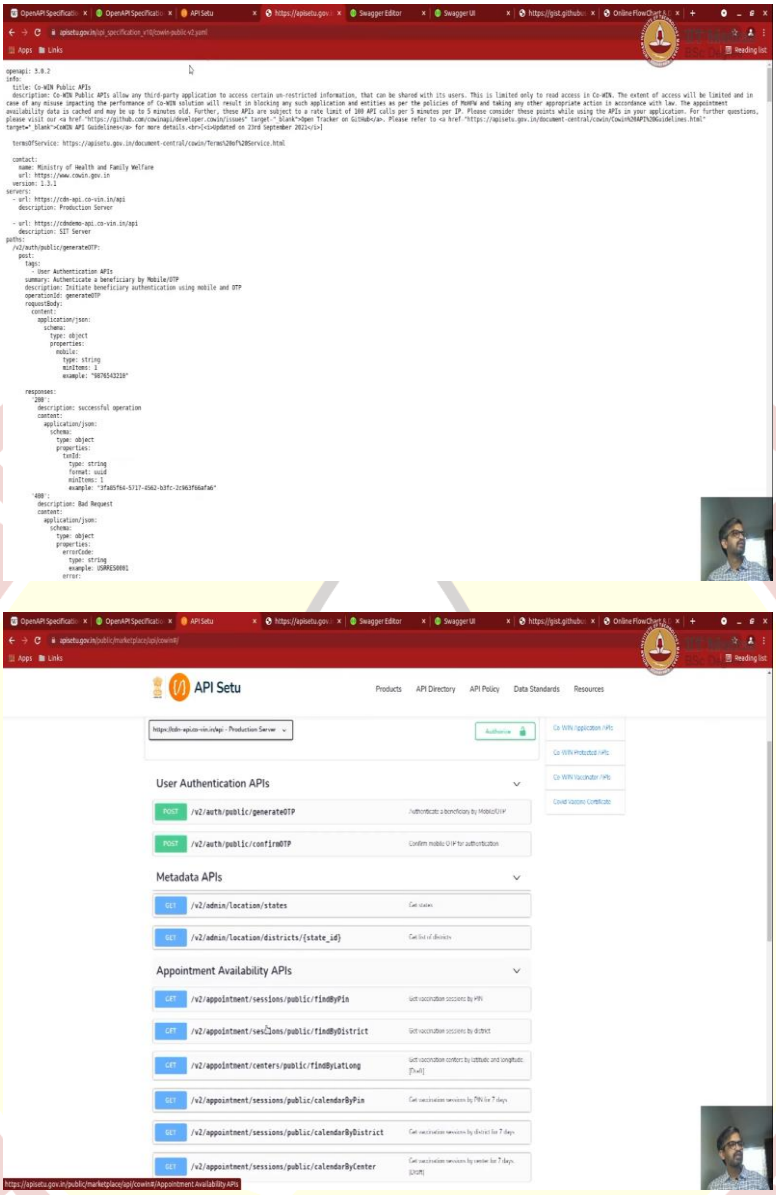
You can click on ‘try it out’. And you can execute, if I execute now, I think it might throw an error, because you know, we do not have an authorised key, otherwise, you can execute directly from here, from the documentation. So that is the best part.

(Refer Slide Time: 04:59)



And you can also explore other API's, there are many other available API’s. I think API Setu is one of those directories that you can go and explore by yourself. Now, this is a rendered format of the documentation has HTML, you know, application.

(Refer Slide Time: 05:19)



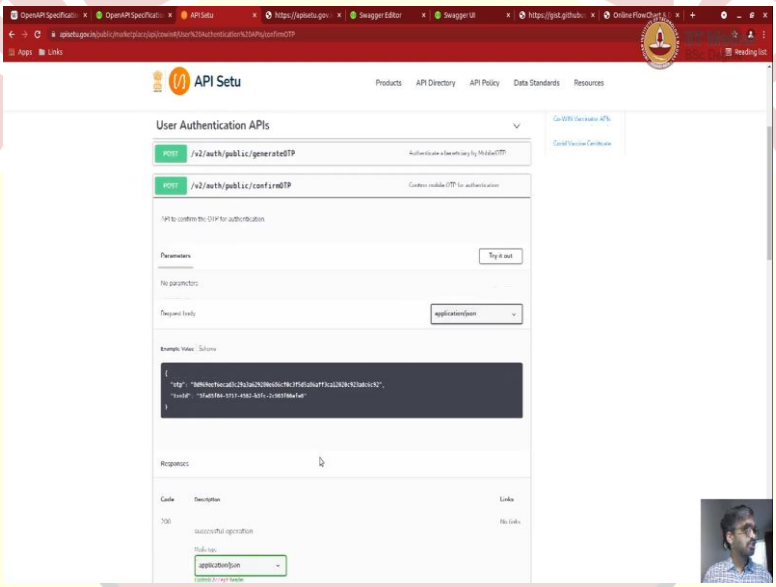
What is that machine readable file that we were talking about, whether it is in JSON or yaml format that is this. It is the same thing. This is the source code of this API description. And this is described as an yaml, you can see that you know, CoWIN public API's is the title of the API's that you can see, you can see it here, and then Terms of Service, or contact, other details of it, or have been rendered from the source code here.

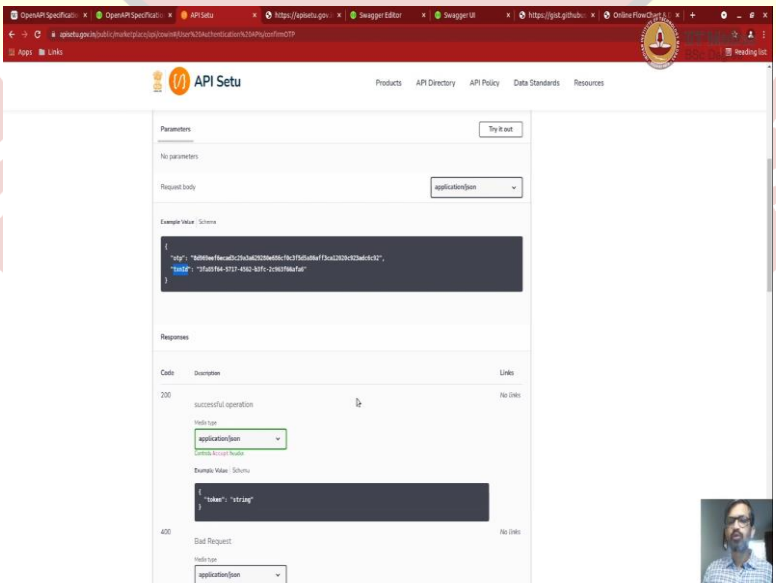
So virtually when you are actually trying to describe an API, or you know, document in API, you are going to write a document like this in yaml, or other format, JSON. To then you will render and display to folks or anyone else, so that they can see and explore the API's. So for example, if you are a developer, and you are expected to develop all these API's on the server side, then probably the designer of these API's would have sent you this document and asked

you to develop it because you know, the input, you know, the output, you know the authentication, and you know, where it needs to be hosted.

On the other side, if you are a consumer of this API, for example, if you are developing an application, or a mobile application, or a web application, and you want to use these API's, if you want to implement these API's, or the consume these API's, you can just access this documentation and get all the details that is required to unsume these API's. You do not need to go to the developer and ask, you know, what is output? How does it look?

(Refer Slide Time: 07:07)

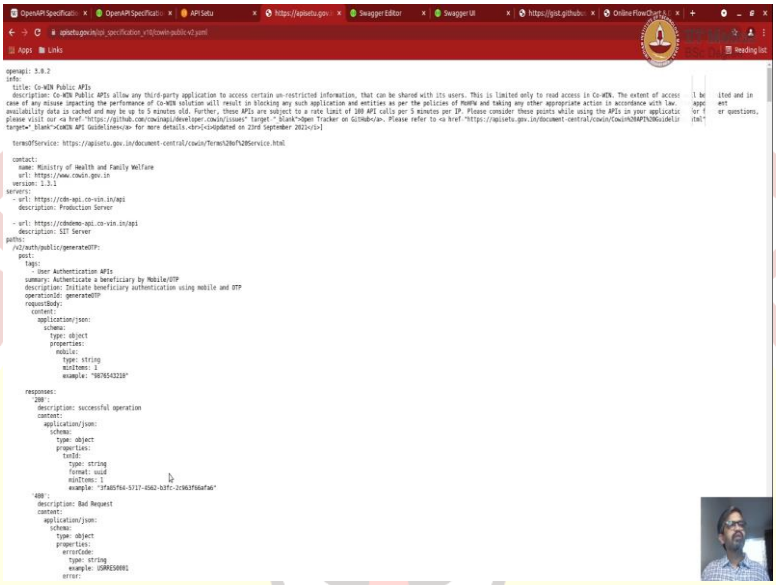


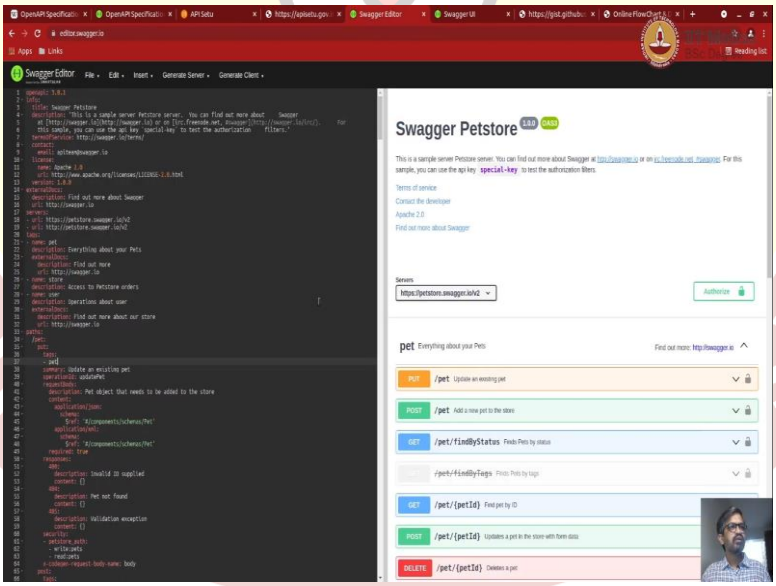


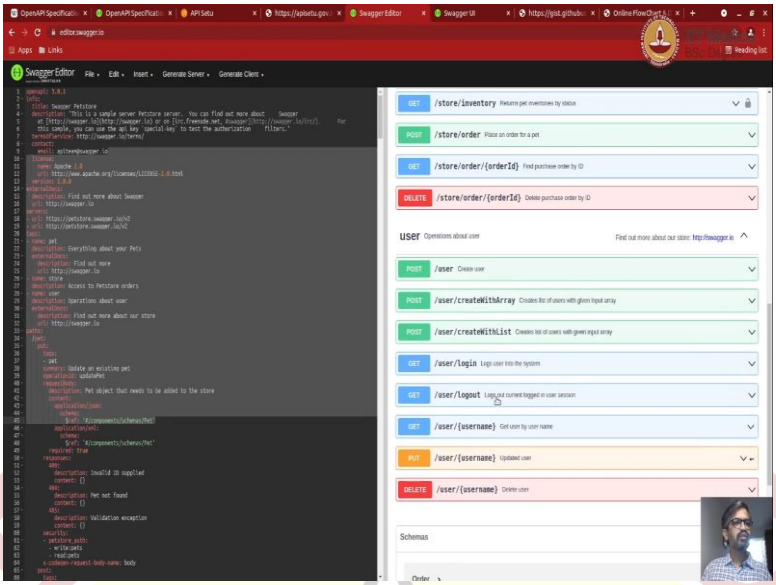
Because you can literally look at, for example, confirm OTP. What does this return. Okay, this returns a token. It is very clear. And what does it take as input parameters? It takes JSON

as input parameters with two elements. One is OTP, and other is transaction ID. How does it authorize? It needs an authorize. So all of this is very very clearly defined. So you can just directly consume it without, you know, even talking to developer of the service. That is the advantage of openAPI.

(Refer Slide Time: 07:40)





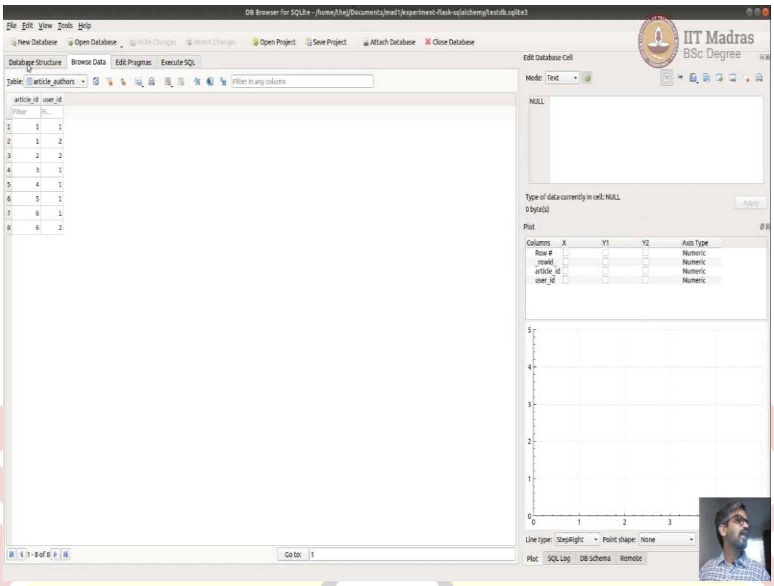


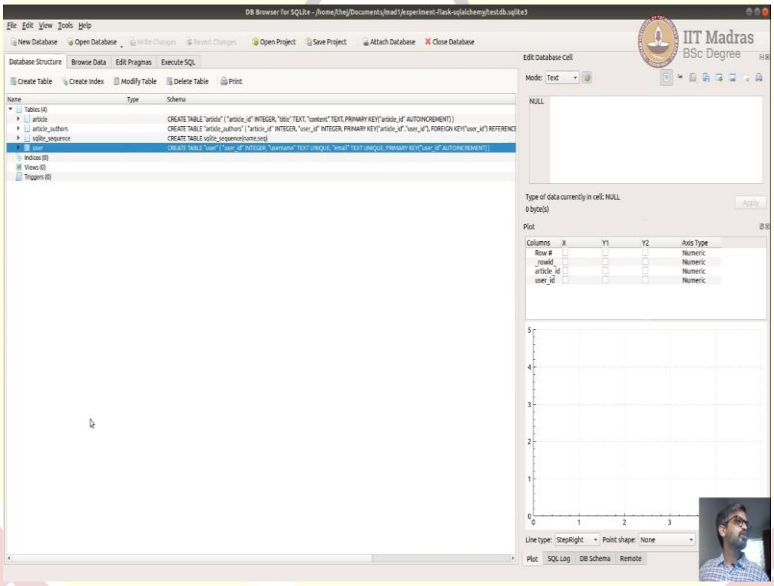
Now, you know, we know that we have to develop this, if you are a developer of the API services, you need to actually write a document like this. Can you write it manually? Yes, you can write it manually. But you can also use some tools to write it in an easy way. Swagger itself has an open source tool called Swagger Editor.

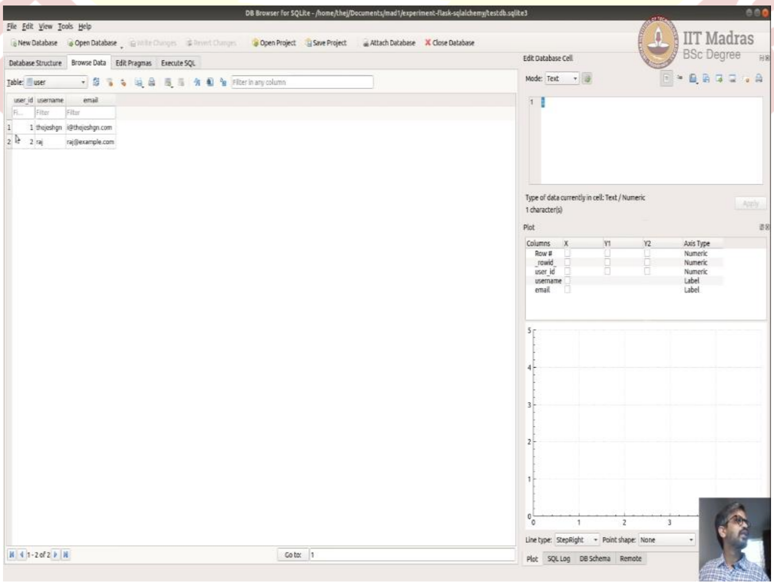
So, you know, web based editor, you can have a look at it here, you know, here is the code. And here is how the rendered version looks like. This is a generic Swagger Petstore, where they have different multiple APIs, you know, pet, a store, and the user. For example, user has log in, log out, get user, update user, etc, etc.

And similarly, for example, pet has an updated system PET, PUT is usually used for updating existing pet. POST is used for creating the petstore, GET is used for getting the pets, now, getting specific pet, getting specific pet by status, and then creating a pet and etc, etc. So there is, those are all defined here that you can use, and those are all rendered, based on the code here.

(Refer Slide Time: 09:15)



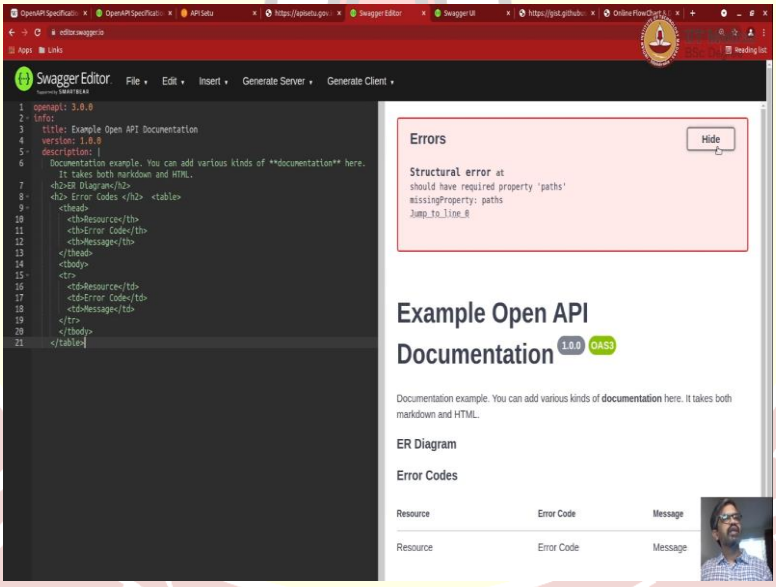




Now, let us clear this and start our own. Let us do our own. You can clear the editor. And then, you know, let us start by adding some higher level documentation for our database that we had application that we had, let us say, we were creating an API for this. Let us say, we were just creating an API for user (())(09:24) API for the user that the user table, has a very simple structure. It has only three columns, user_id, username and email id. user_id is, you know, auto generated, when the user name and email is product by the end consumer, who is going to create a user. So let us create an API for that.

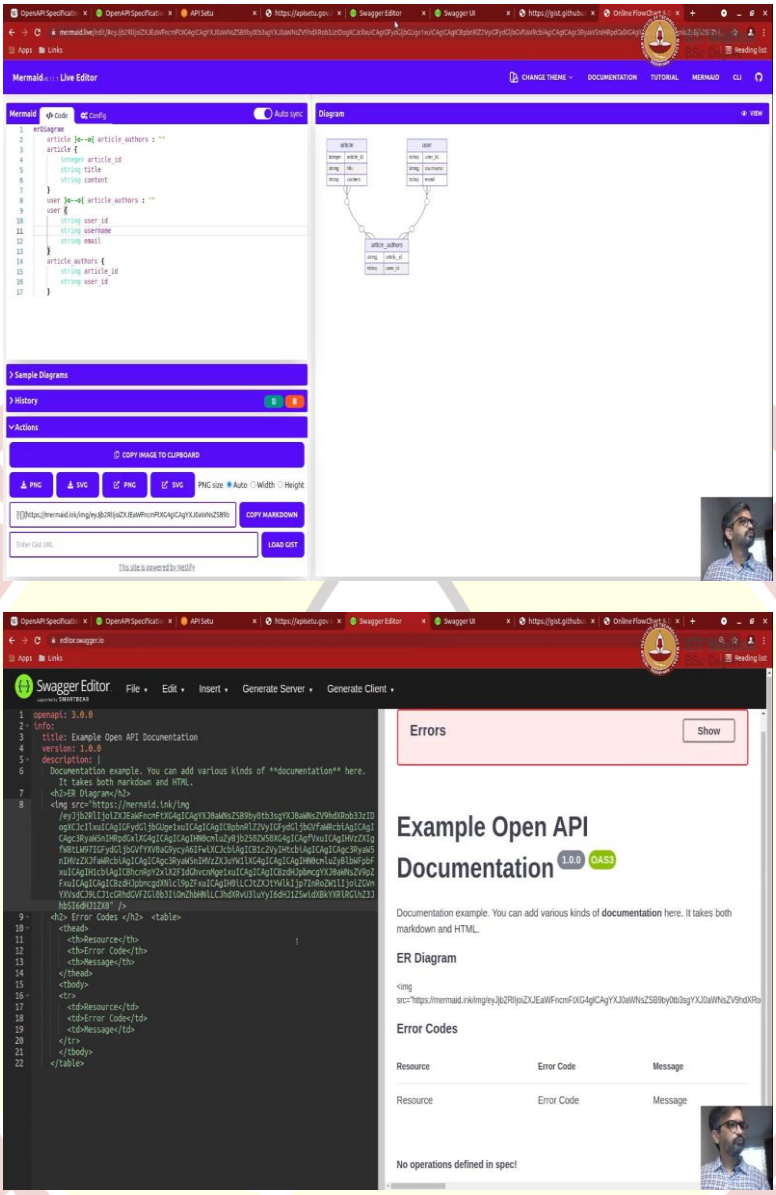
I am based on before we create an API, we actually like to document it. So let us create our documentation for the API. So let us start with adding on top level details. That would be, just give me a second, I will just paste it instead of typing, so that it will be faster. I am just going to increase the screen resolution. Okay, let us just add some top level details of an example open API.

(Refer Slide Time: 10:38)



Let me just saying it for a second just ignore there. We are saying I am going to define the open API 3.0. Like the documentation itself. And then name of my API's is example open API. And then some description I have added, you know, what kind of API it is, and stuff like that, here, I am just giving a general description. And then, you know, you can add some diagrams too, if you want to.

(Refer Slide Time: 11:11)

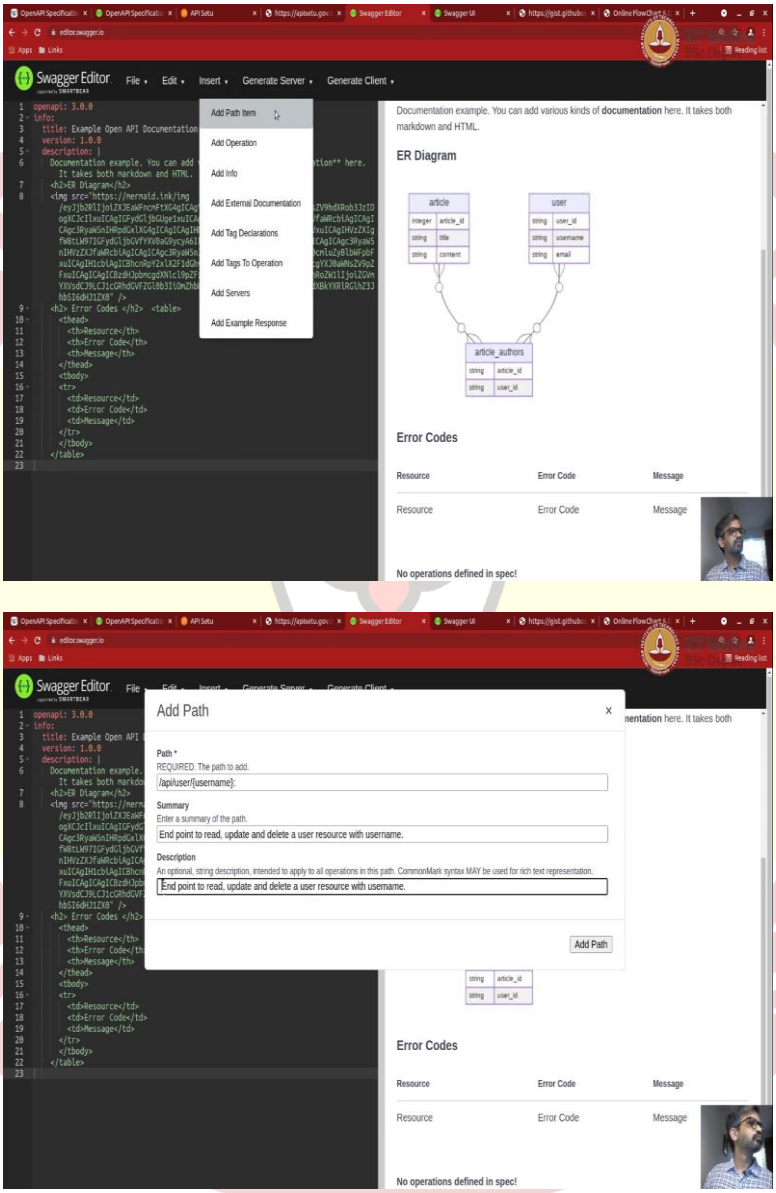


I can write a quick diagram, go to some Mermaid, a design a diagram, by writing the diagram code here, and then just copy paste the URL and embed it. So for example, I can just embed it. So, if this is a path. Since this path of the documentation takes HTML and markdown, I can just embedded an image here, so it embeds.

So, what I have done here, this is simple, you know, image creator, where you write image by code, and then it creates an image, it is called a mermaid, you can use any other image creator. But this is just an easy tool, and an open source tool, to create images. So for example, you can add an extra column here, string let us say, date, creation date, for example.

Creation date will be date, okay Things like that. You can just edit and make the table structure, so that it is easy for end user, how the table structure looks like. And then you can just copy the image path. And, you know, add it to your editor, so that, you know, your documentation has that.

(Refer Slide Time: 12:48)



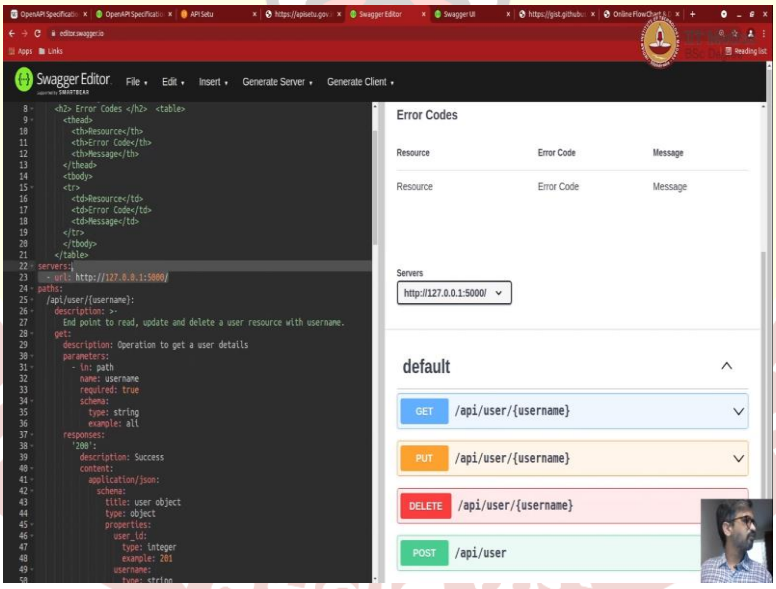
Then I have an error code's table, where I want to list error codes for, what resource for user resource, what is the error code and what is the error message. So that you know, the user knows what kind of error it throws. Currently, we have not added any operations. So it says, operations are not defined.

Let us add an operation. So I am just going to add an operation by adding a path. So I am just going to, not going to go through detail of everything, you might have to explore it yourself. Most probably you will, you can pick up like an existing API documentation and modify it. So that it is easy, or you can use like a completely GUI based editors, you know, so that it becomes easy, and I will share some links to the GUI based editors too.

So, you can either add it manually by typing it here, or you can just click on Insert, you know, add a path item and add the details here, and then our code appear here. For example, 'the path to add', we can say /api, I am just going to add a path. And description of the path will be.

And I am just going to paste same thing here. It has added all the details here. And you know, it is easy to do that. If you do not want to do that you can just I am just going to copy paste some of the details here directly so that you know I can just explain it to you without taking much time.

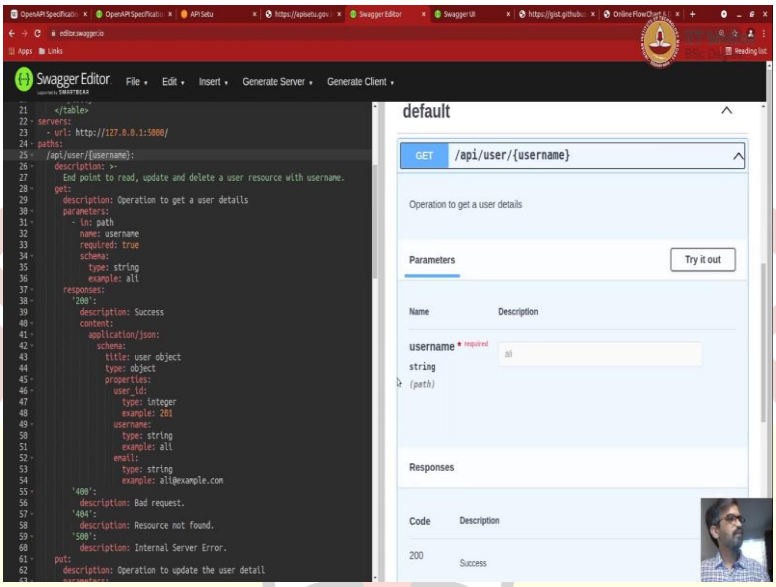
(Refer Slide Time: 14:42)



So, I have added here details, so that, for all the users, so it becomes easy. So here are defined server. You can define multiple servers, locally, usually generally. You have to run it at 127.0.0.1:5000/, so that is what we have added. For example, if you want to add other servers, you can just add more.

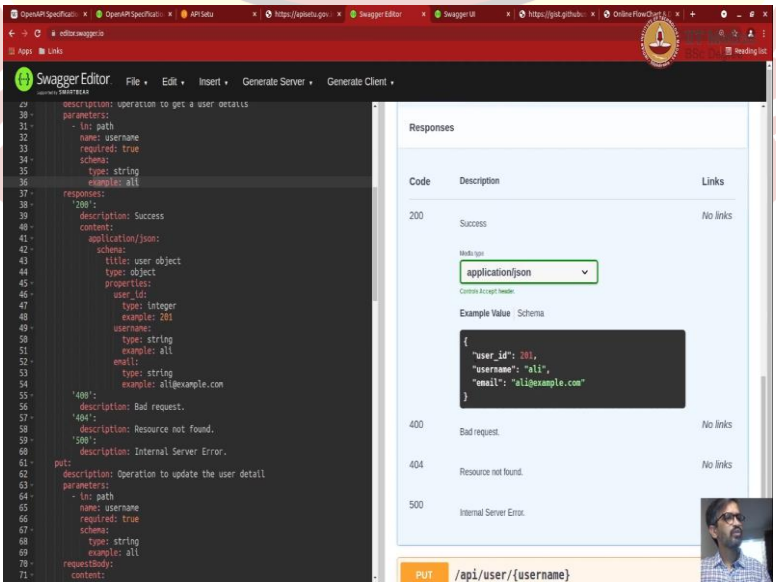
The next one is, you know, paths, like I said, you can insert basic here, I have just pasted it here. So the path will look like a /api/user /user name is a variable that we are going to pass to the API's to get the details of the thing. So here is the username, for example.

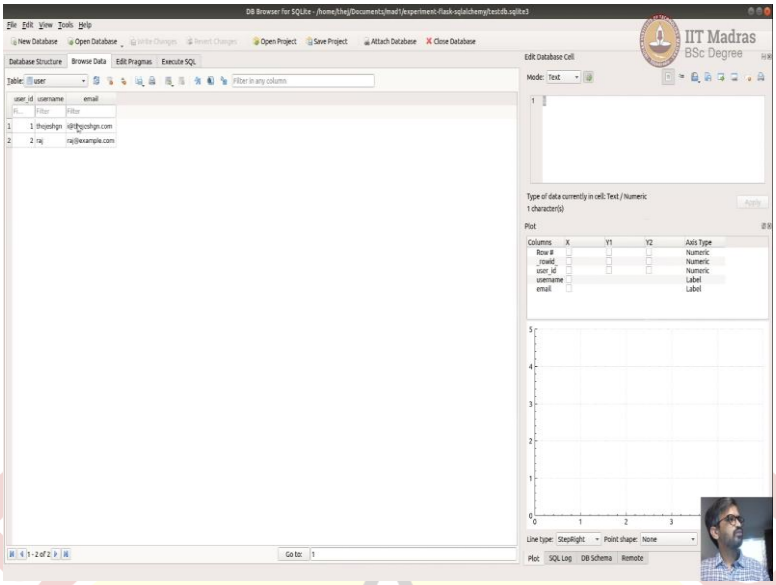
(Refer Slide Time: 15:28)



So, then you can give the description which appears here, then you can say in parameters that this API requires, and in where in path, so this is a parameter in path of the URL and it is required. And what is the kind of, what is the type of variabilities, which is, you know, user name is a string, and an example which is Ali, so you can see it here.

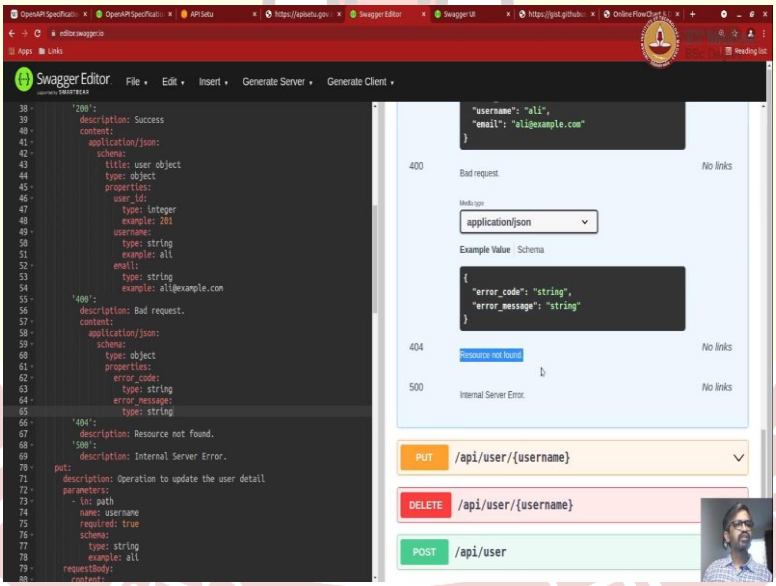
(Refer Slide Time: 16:02)





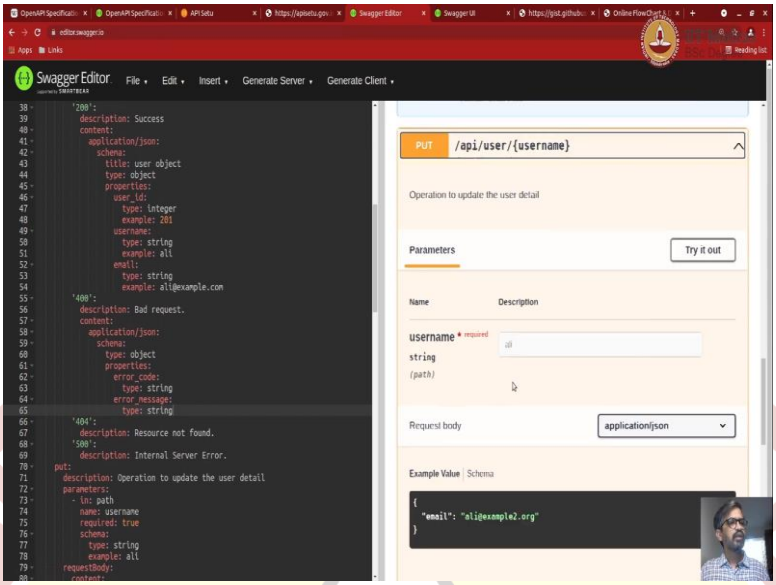
And what happens on success. On success, your return user_id, username and email, that is what this table has, once successfully inserted, it will return the whole thing.

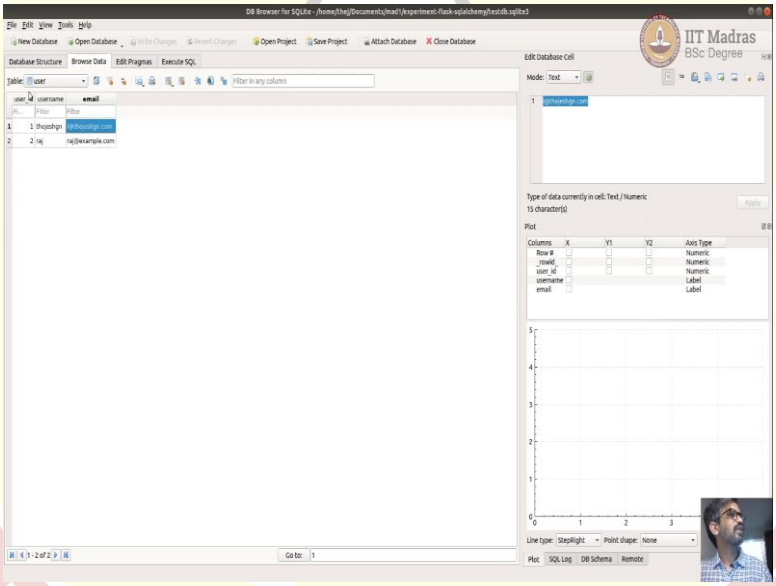
(Refer Slide Time: 16:43)

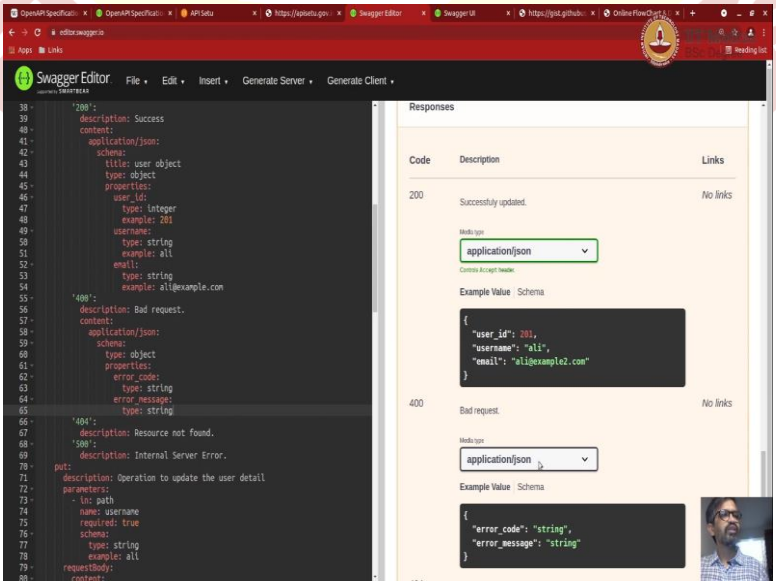


And then if there is a bad request, what are we going to return, etc, etc. We are going to return an error message. So I think I must have missed it, how does an error message look like, for example, I can add it to the 400 here, so that, whenever there is a bad request, it will return a JSON with the error code and the error message. And then if it is, if there is a URL not found, if there is a mistake, then it will return 404 or the user is not found, it will return 404. And then, you know, 500, if there is an internal server error. So these are the standard, you know, path.

(Refer Slide Time: 17:08)

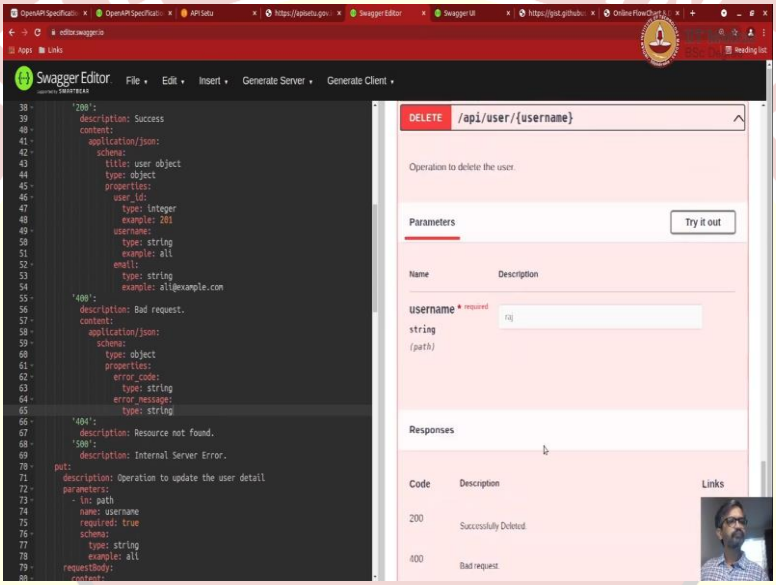


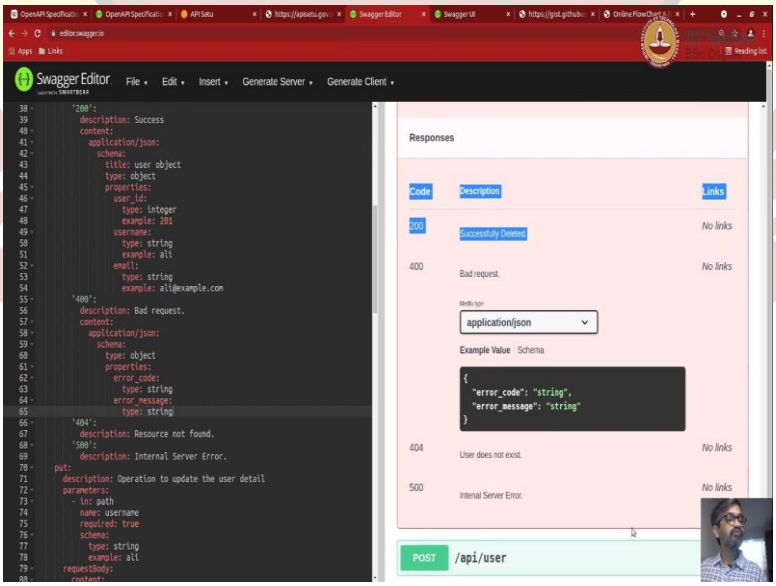


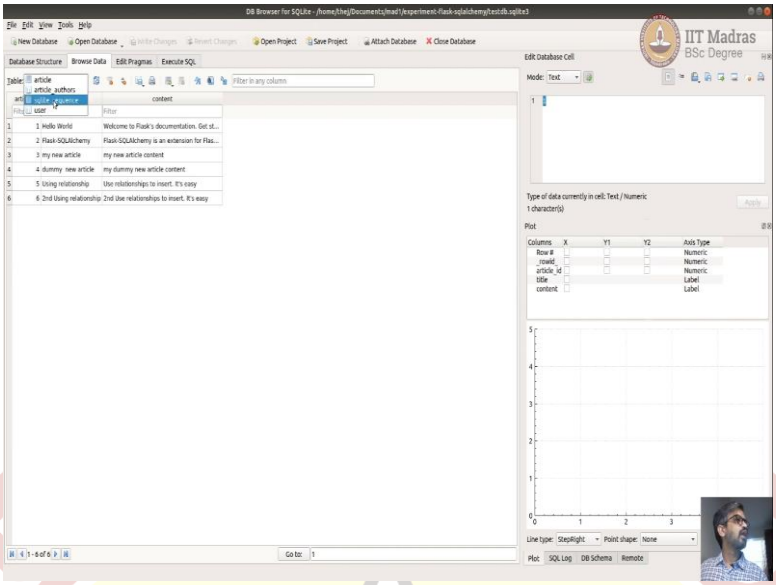


Similarly, ‘put’, ‘put’ is an update, it updates the user given by this username. And then, you know, username is input. And currently, we can just update only the email, because that is the only other thing that allowed to be updated. user_id is, you know, auto generated, you can not update and user name usually, we consider it as unique and you know, nonalterable. So that cannot be updated. Once it is updated. If everything goes down, it will return the updated object. If there is, if it could not update for some reason, for example, we are updating the duplicate email id then it will show in message.

(Refer Slide Time: 16:43)



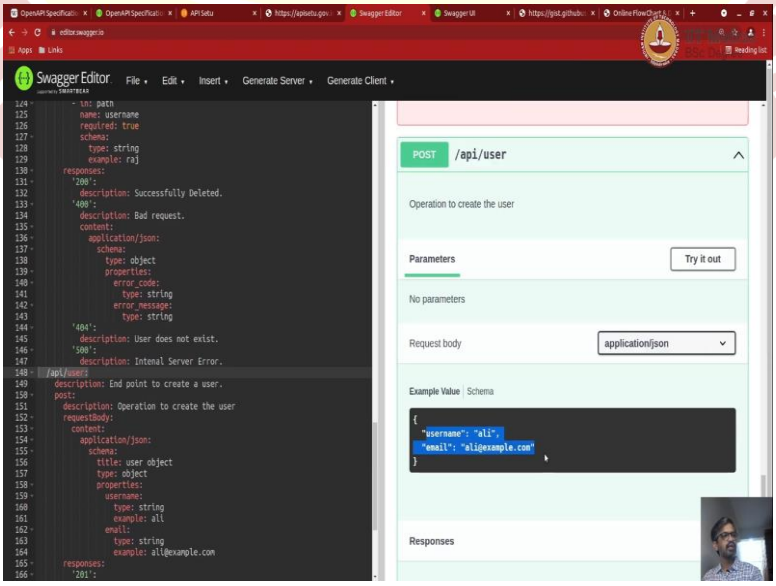


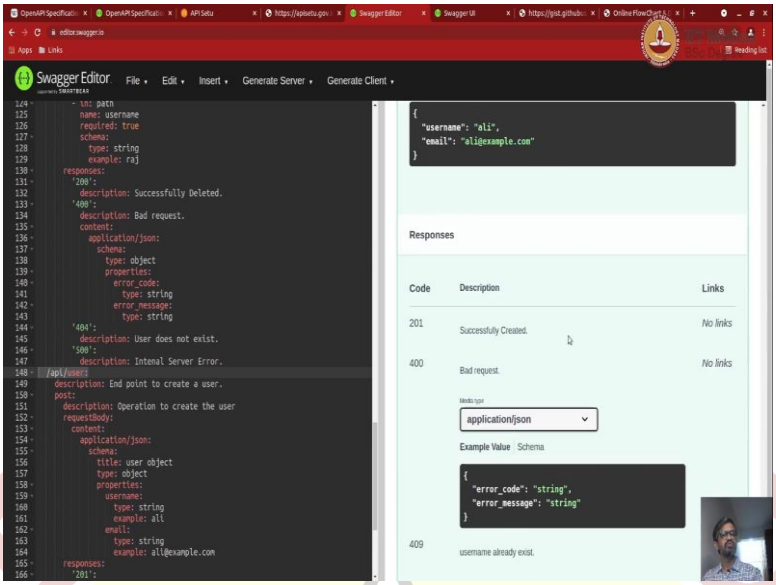


Similarly delete, it can, if it deletes, you know, it will just return 200 successfully deleted if it cannot delete, for example, if you are trying to delete a user name, 'thegeshgn', but it has an article, you cannot delete it, because you know, you will have to have a relationship in 'article_authors' that will not allow you to delete basically, you have to delete everything if you delete an article user.

So basically, you might want to send an error, saying, you know, it has relationships, we can delete, delete articles first and then you can delete the user. So basically, to delete a user, you should not have any articles. So you can have multiple other conditions that can be coded as error messages.

(Refer Slide Time: 18:49)



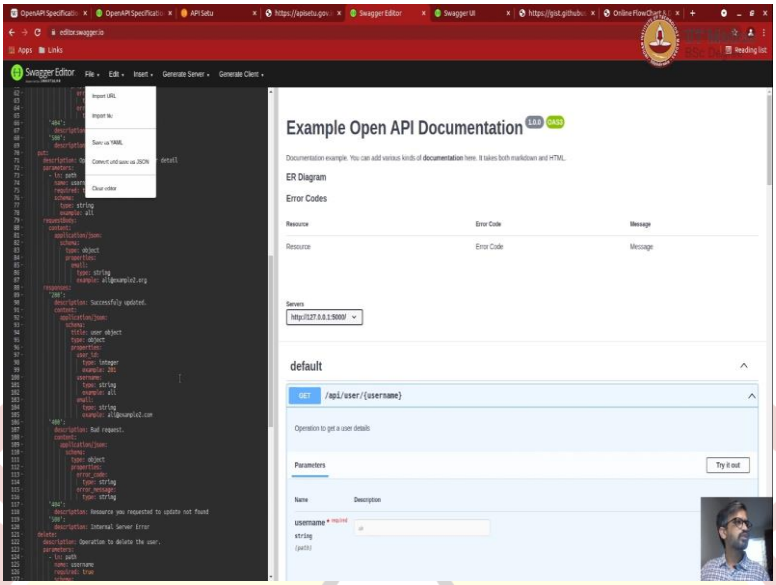


Similarly, ‘POST’, ‘POST’ is at /user. This is a different path. For example, that way, there will be a second path defined, I will just expand it here. This is the second path defined, in which there is no user name, because we do not have the user name. And that takes user name, and you know, email as input. Because if you are trying to insert into user table, we need user name and email, and then it returns if everything goes well, it should, it returns 201.

Or it can even return sometimes a whole object, based on how someone has designed the API. And basically, if you have designed like that, then you will return that, if there is an error, for example, duplicate email, or duplicate username, or username has space, all those error will be returned as 400 status code. And then error message and error code.

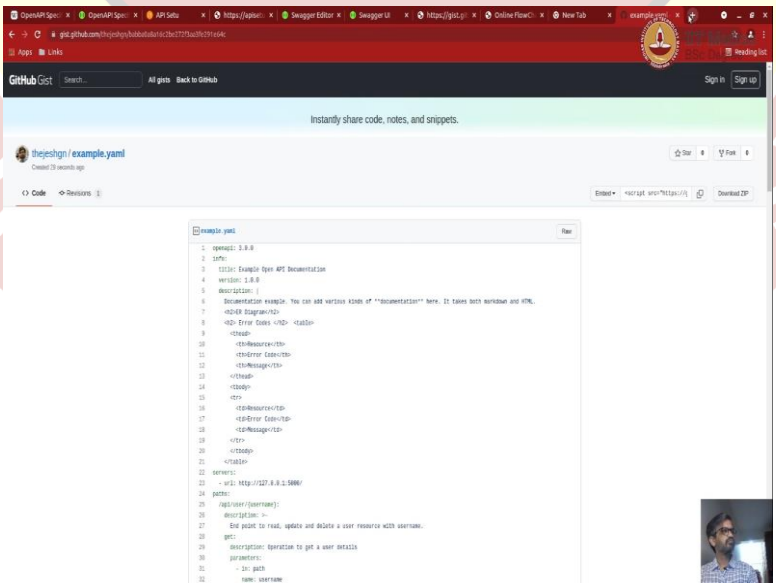
This is how we usually describe the APIs. Now if you are going to use this API, you know how to use it. In this API, we have not defined any authorization or like authentication or authorization. Hence, that is not showing up. Otherwise, it would have shown up.

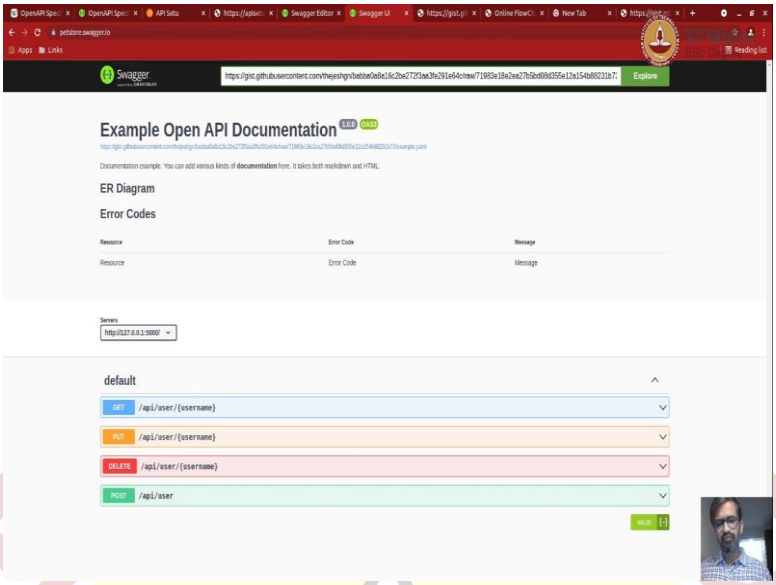
(Refer Slide Time: 19:55)



Now you have the whole thing as documented. It is much, much much clearer. You do not need to look into code to see what is doing. You have the API is clearly defined here as a documentation. Now you can also, you know, share this yaml with anyone, you can just download the file, you know, and share it with someone, or you can just you know, if it is not a secretive API, or not a private API, you can just add it to GitHub, go to gist and add it and share the URL.

(Refer Slide Time: 20:29)



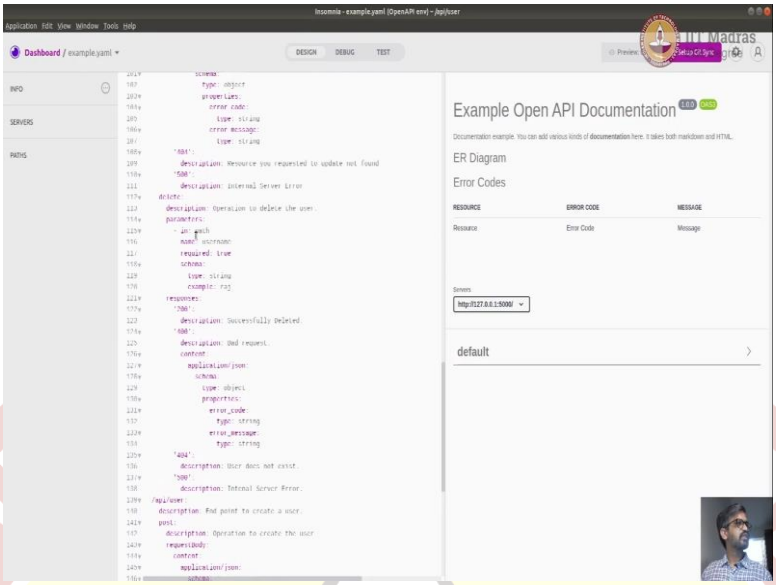


For example, here, Simple yaml file. If you want to view it, you know, you can just copy, click on the raw here and copy the raw data. And then, you know, paste it here. Or you can just go to swagger UI, which is by default it shows, you know, a pet store, you can just copy this yaml file URL, and, you know, paste it here and click on explore. It just shows the whole API documentation here, which is easy to view.

There are many other places, where you can view this, you know, many other tools available to view this. Couple of examples are, you know, there is one called rapid Doc, let me just show that to you. Just as an example, there is another one which can, again, you have to give an URL to it, I will just copy paste this URL, paste it here, and click on open. And, you know, it renders and shows it to you, how the API looks like. Then all might render in different ways. But they all have similar functionalities.

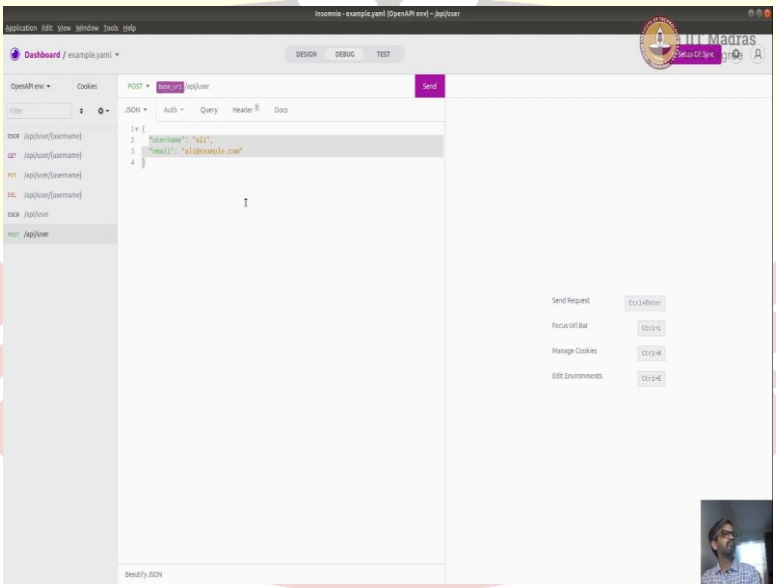
But I think I would say best is would be to use swagger UI, or swagger editor to see and explore, or Mrindock is also quite nice, you can use it. There are other ones, you can also use, you know, a tickclient, or a local installed client called insomnia. You know, let me just open that quickly. For you to explore it. You can search, insomnia, REST API. This is an open source tool that you can use, install locally. And you can use.

(Refer Slide Time: 22:30)



For example, when you open it locally, it looks like this. Here I have pasted the same yaml file, and it renders here on your right side, you can try it out, you can try it out the same thing, it is locally done. You can also do other things in this. This is very extensive tool.

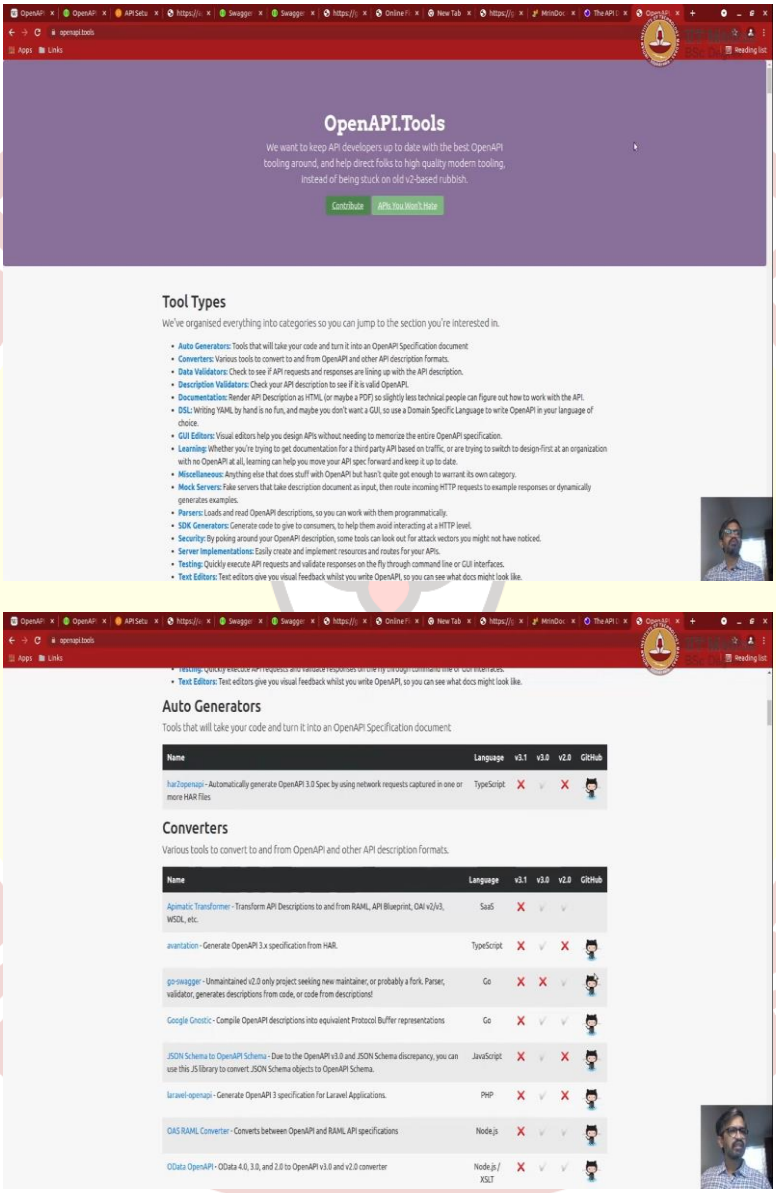
(Refer Slide Time: 23:01)



Or to explore this, you can also click on test, and then create new test suit, or click on debug and get all these API's here. For example, when I go to post, it auto populates, example, JSON, then auto populates the base URL from the environment defined in the yaml. And then you know gives all the things.

You can even try and send. Currently no server is running, you know, in 127, locally at 5000 port, no server is running. If it was running, it would have access and shown you the result. So you can use insomnia too. There are many other tools to explore the APIs, I think I would want you to go and explore yourself.

(Refer Slide Time: 23:45)



There is the site, which lists all the open source tools or even some of the closed source tools, which are related to open API. So you know, you can go explore so many tools available. Like I think the one which has GitHub is mostly open source. So you can you can try them, you know, and explore.

So most probably what is going to happen, the thing that is going to happen is someone is going to design this documentation and give you this documentation and ask you to implement the API's.

So, what would you do, you would take this document, and then read to this document, see the details of the documentation, create the table, create the flask RESTful API s and test that against this API documentation. So, this API documentation is all the input and output for you to test. And also, he would have defined the error codes for you to return an error, whenever there is an error. Things like that is one case.

The other case is you already I know built this APIs and you want someone else to use then you will actually write this documentation yourself and tell them here. My API's are exposed to the world, to be used if you are authenticated or authorized. And here are the details of the API, those are the two scenarios, which you will probably use this API documentation most probably, and you would actually find many of these documentation in the, you know, in the open world on internet on any of the service providers.

I would suggest you to go to API setu and explore them. There are also other commercial players, who expose their API's is open API documentation. I would I suggest you to explore them as well. That is it. Thank you so much for watching.