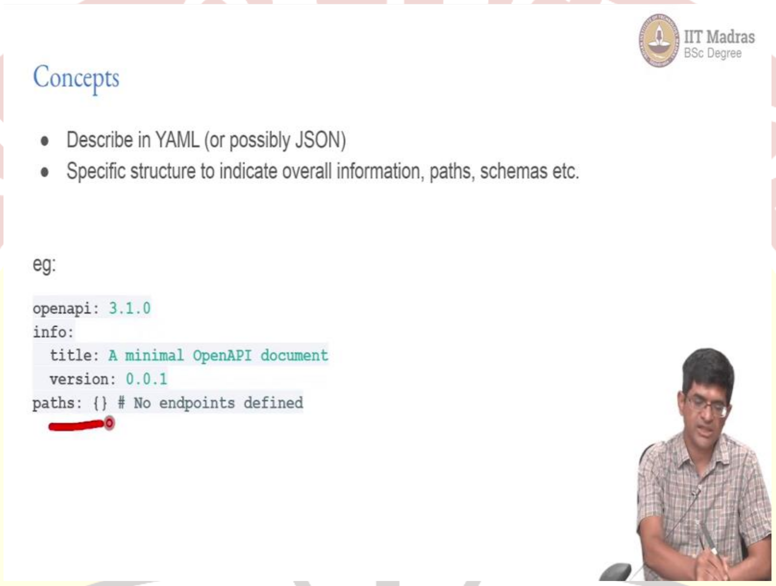
IIT Madras

ONLINE DEGREE

**Modern Application Development - I**
**Professor Nitin Chandrachoodaan**
**Department of Electrical Engineering**
**Indian Institute of Technology, Madras**
**Important Concepts of an API**

(Refer Slide Time: 00:16)



## Concepts

- Describe in YAML (or possibly JSON)
- Specific structure to indicate overall information, paths, schemas etc.

eg:

```
openapi: 3.1.0
info:
  title: A minimal OpenAPI document
  version: 0.0.1
paths: {} # No endpoints defined
```

Hello, everyone, and welcome to this course on Modern Application Development. So, let us look at the concepts and one of the things is mostly the open API specifications. So, you write a specification file. When you are creating a new API, you need to write the specification file for that and this is mostly done in YAML yet another markup language, it could potentially be done in JSON also, the important thing is it contains structure, which means that it can be sort of read by a machine without any ambiguity.

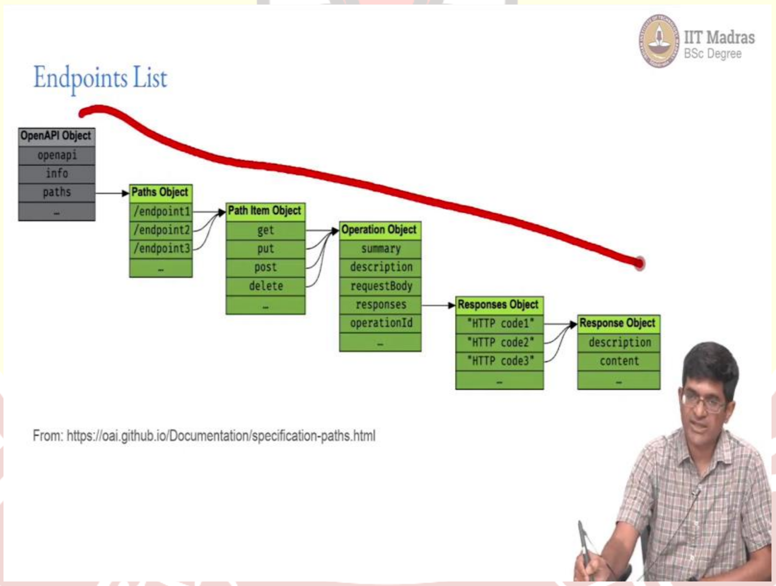
So, the machine will be able to get a clear picture of what you are trying to convey. The most sort of trivial example is something like this. So, this, what I have over here is an example of a YAML file. Exact details of how YAML is implemented, how it is meant to be passed and so on, I am not going to get into over here that is a little bit beyond the scope of what we can do now.

Broadly speaking, what YAML is doing is it is similar to the notion of the keys and values that where there in JSON or in general in a dictionary. So, what it has is, it has a key, which is openapi, which has this value 3.1.0, which is basically treated as a string but which gives the version number. Because it is got like two points over here, it cannot be a number. But so it is just stored as a string, essentially. What does it contain? It contains info, which in turn contains two sub pieces of information. One of them is title and one is version.

So, the title basically says that it is a minimal open API document, the version says 0.001 and then it has one thing paths, which in this case has just been defined as a null array or a null object, does not contain any entries, no endpoints have been defined. This by itself is a valid open API document.

What would happen, if you read through something like this, you only get the information that it is an minimal open API document with this version number using this version of open API, which does not have any endpoints. So, it is not a particularly useful document, but just serves to show that this is the minimal structure that you can expect to see.

(Refer Slide Time: 02:32)



## Endpoints List

From: https://oai.github.io/Documentation/specification-paths.html

So, most of what I am going to show over here are examples from the oai the open API documentation, which is available at oai.github.io and these pictures, as well as some of the succeeding things are all from there. So, what is does a open API by itself contain? You can see that, we already had this open API, the info and the paths. These are sort of the three main requirements of an open API object.

Now, in turn paths would contain paths object, which would have multiple endpoints, so it would have endpoint 1, endpoint 1 endpoint 3, an endpoint would have a path item object. So, what is an endpoint? In any API, what we refer to as the endpoint is that URL that you are going to access and the type of request that you need to send to that URL.

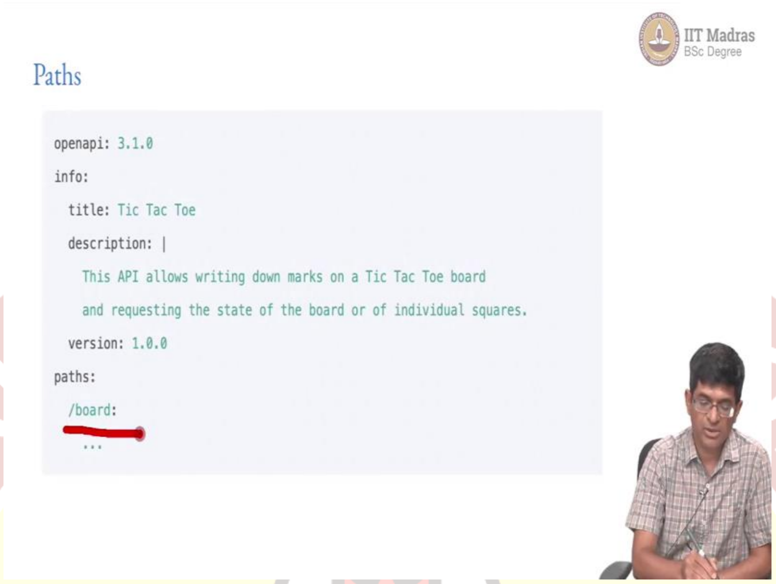
When you combine those two together, it is called an endpoint. So, a get request to wikipedia.org/rest.php/ something something is one particular end point on the Wikipedia API. So, each of these endpoints has a corresponding path item object, which basically tells the type of request.

The operation, each of these has an operation object or rather, the path item object has a list of operations. Each of these is an operation object, which has a summary a description, request, body responses, etc. So, in other words, what it is saying is corresponding to the get I will have one full operation object, that operation object, I would have a brief summary saying what it is supposed to do, a description that gives a little bit more detail on what I should, how I should use that get request to this particular endpoint.

And sorry, let me just correct that an endpoint is basically just a URL by itself and the operation to be performed on that URL, when combined with the endpoint will give you specific behaviours. So, in this case, the operation could be something like get, it will be summarised and described, it can say whether or not it has a request body request body usually refers more to the post response post kind of operations, where they actually have text in the body that needs to be sent across.

It describes what kind of responses you can expect for this operation, some more information, etc. Now, these responses, in turn could be different HTTP codes 200, 404, 400, something on that sort. For each of those response codes, it has more information, which could basically say, this is the description, status, okay, okay, everything is fine, I created a new shopping cart for you and the content which basically says okay, this is what are finally created, something like that. So, this is sort of the overall structure that you can expect to see inside an open API specification document. What does it look like in YAML format? We will look at a couple of examples.

(Refer Slide Time: 05:45)



## Paths

```
openapi: 3.1.0
info:
  title: Tic Tac Toe
  description: |
    This API allows writing down marks on a Tic Tac Toe board
    and requesting the state of the board or of individual squares.
  version: 1.0.0
paths:
  /board:
    ...
```

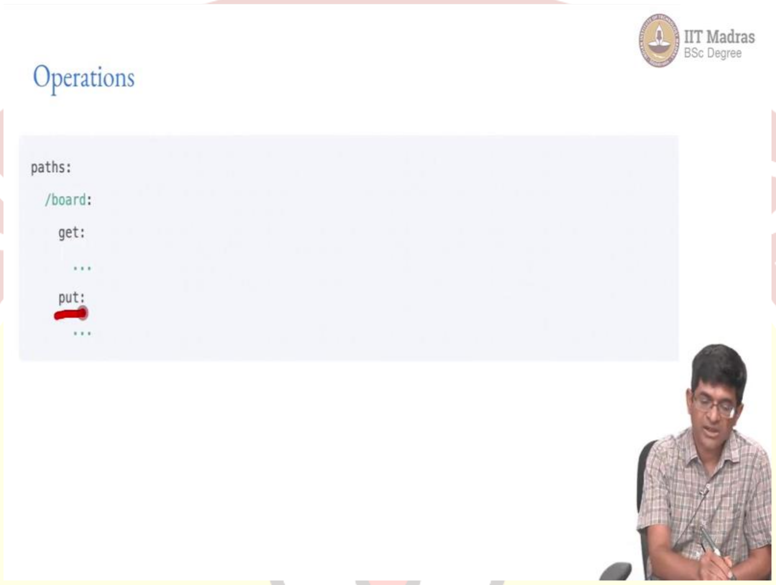
So, like I said, this is the same thing. Some of these examples are from the running example, which they have on the open API webpage, which is for a web app which plays the game Tic Tac Toe nought’s and crosses, so you would be familiar with that, I mean, you basically have like, grid out here and you have to put X's and O's such that you get 3 X‘s in a line or 3 O's in a line and whoever gets it right first will win.

So, they are sort of trying to say, how do we create an API that will allow playing such a game? Now, that is interesting, because if you think about it, what is it doing? This API is only encoding the functionality of the game, it is telling you that, for example, you can create a new game, you can put an X or an O in a specific location and finally, you can retrieve the state of the board and you can also find out who won.

It does not tell you anything about how to display the board, how to make an X or an O in a specific location, how to decide whose turn it is, none of that information is being specified as part of the API. So, that separation is very much there and what we are trying to do, as we will see is it basically tells you information about the functionality.

This part, the open API and the info is something that you already saw. So, we had a title and version earlier. Now, it also has a description and as you can see, this is all part of YAML, it just basically allows you to have a multiline description. So, all of this is sort of the header part, then come the paths previously, in the trivial example that we saw, it was just a null, basically saying there are no paths over there. Here, for example, we would have paths. So, there might be a path which say a /board that can be used, for example, to get the present state of the board.

(Refer Slide Time: 07:45)

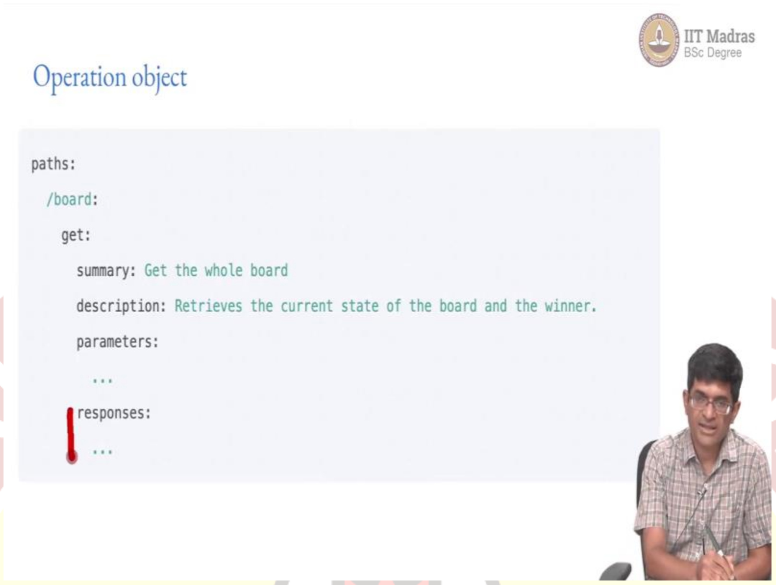


## Operations

```
paths:
  /board:
    get:
      ...
    put:
      ...
```

What are the kinds of things that we could have under board, I might have something to get, which will basically allow me to retrieve the present state of the board, I might have something called put, which allows me to create a new game or I could potentially even have put that allows me to create a partially filled game. it create the sort of nine square grid, but also pre populated with some values of X's and O's, that I do not know, it all depends on what you are trying to do. But the point is, it is specifying what kind of functionality can be done.

(Refer Slide Time: 08:18)

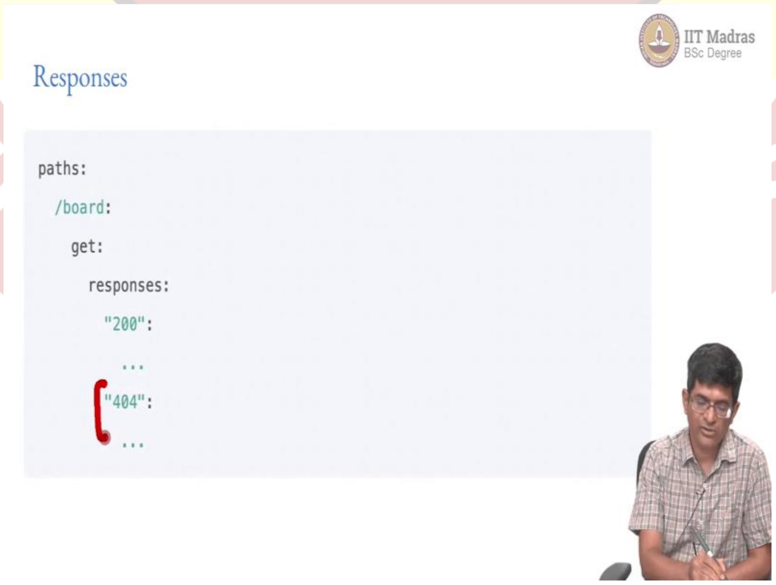


## Operation object

```
paths:
  /board:
    get:
      summary: Get the whole board
      description: Retrieves the current state of the board and the winner.
      parameters:
        ...
      responses:
        ...
```

Let us, get a little bit more detail on the get itself under /board, we have get, for get we basically get the summary. This is the whole board description could be longer it could be multiple lines. In this case, this is also just one line. Parameters and responses, are there any parameters that I need to provide to the board in order to get its current state and what kind of response can I expect to see from the board.

(Refer Slide Time: 08:46)

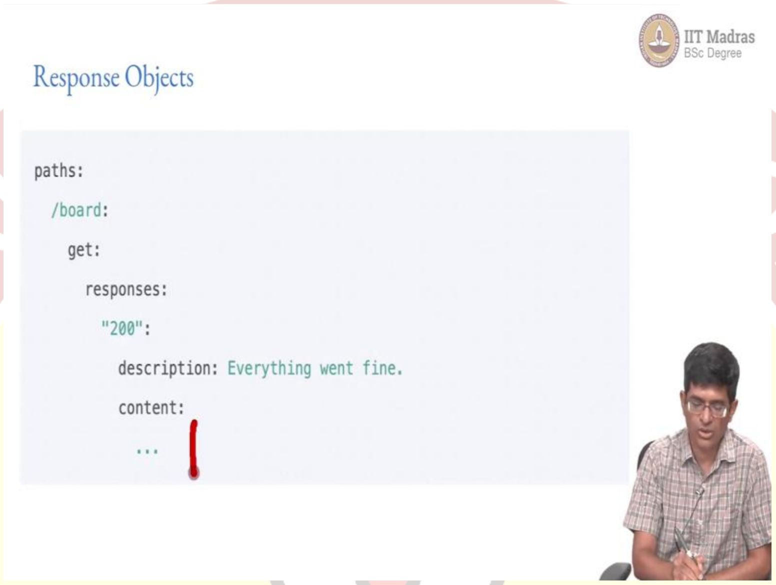


## Responses

```
paths:
  /board:
    get:
      responses:
        "200":
          ...
        "404":
          ...
```

What are the possible responses? One of them is 200, which basically says, successful request, this is the state of the board or it could be a 404, 404 is typically are not found error. It basically says, okay, there is no board, what are you asking me for?

(Refer Slide Time: 09:06)

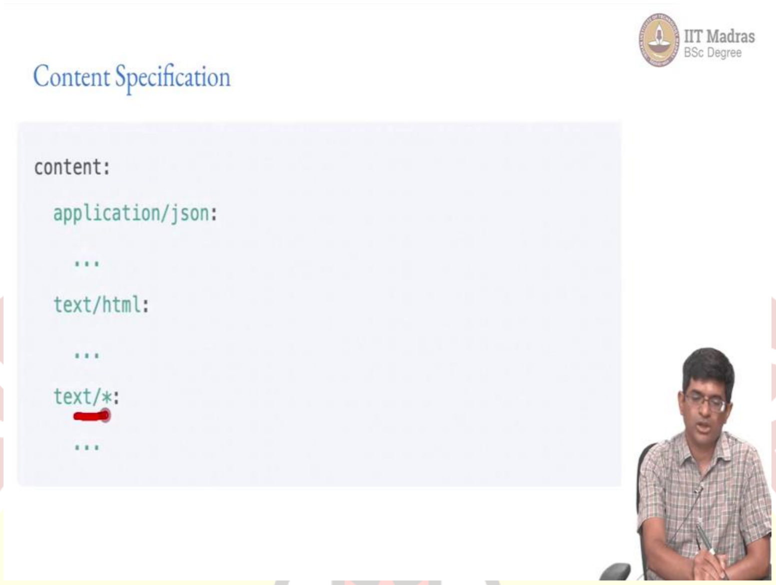


## Response Objects

```
paths:
  /board:
    get:
      responses:
        "200":
          description: Everything went fine.
          content:
            ...
```

Inside the responses, I could go further and say, okay, for 200 the description is everything went fine and content could be some more information, which is most likely the actual state of the board, this is not text, the content is most likely going to be something which has some other format, like, it could be JSON description of the board or something like that.

(Refer Slide Time: 09:30)



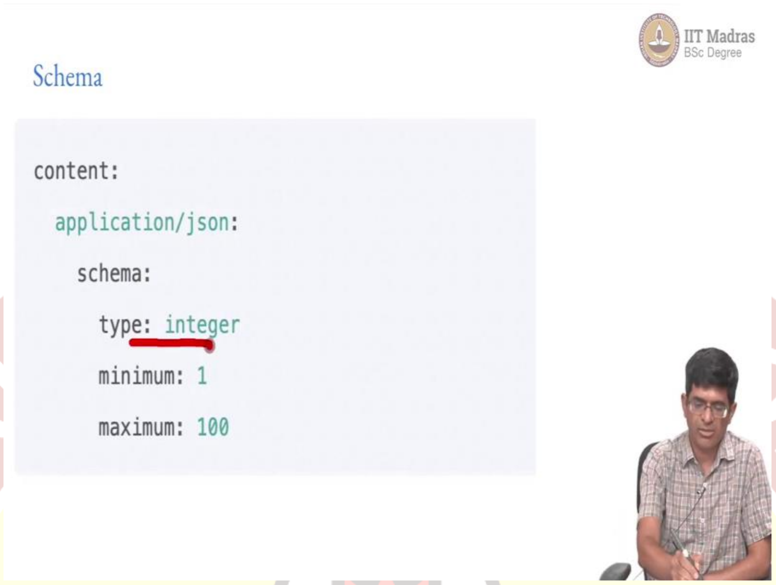
## Content Specification

```
content:
  application/json:
    ...
  text/html:
    ...
  text/*:
    ...
```

So, content could also be of multiple types. You could basically say if the person had requested saying they can accept application/json return a JSON representation of the board. If they have requested text slash HTML, you have to return it as some HTML format. So, this is interesting, what it is telling you is that as part of the API, I could also request different formats of data.

What ultimately happens over there is the controller will get this information from the server side, it is just going to get back data in some standardised format, the controller can then choose which view to return to you, depending on what type of content you are willing to accept. So, the server still does not have to know the server, the model, underlying model still does not need to know whether you are looking at HTML or JSON. It just returns data to the controller and the controller’s job is to pick the appropriate view and send it back.

(Refer Slide Time: 10:31)

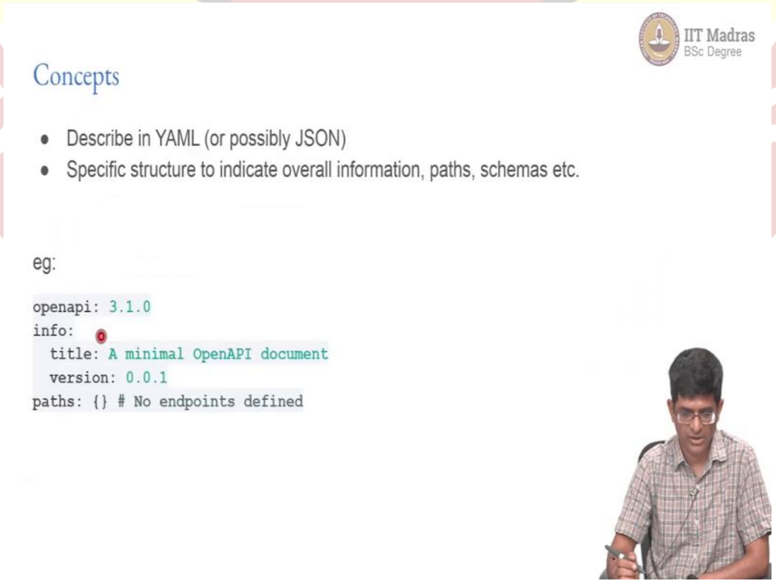


## Schema

```
content:
  application/json:
    schema:
      type: integer
      minimum: 1
      maximum: 100
```

And inside this thing, inside the content application JSON, you could actually have a schema which sort of gives information about what kind of information is going to be retrieved when you try to get the board. In this case, this is just an example. It says type integer minimum 1, maximum 100, that this is probably not relevant for the Tic Tac Toe board game. But in general, it would be something which says some information about what kind of content you can expect to get back when you do get requests on the board.

(Refer Slide Time: 11:11)

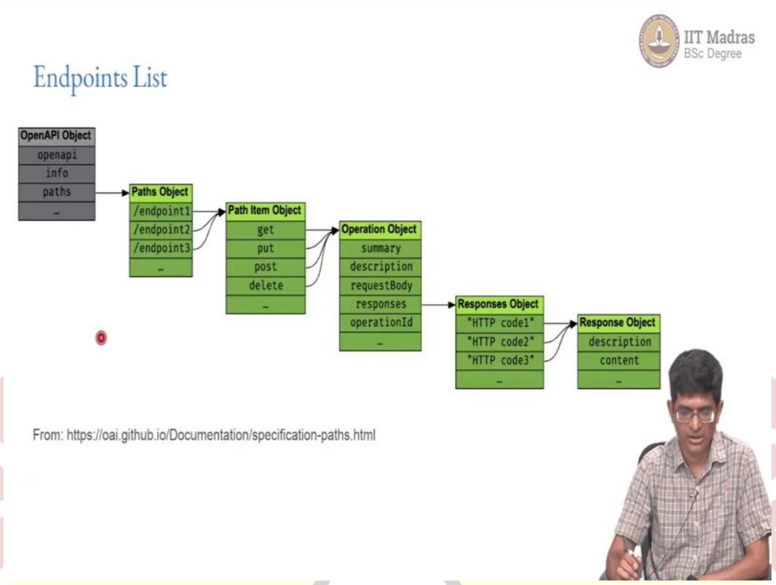


## Concepts

- Describe in YAML (or possibly JSON)
- Specific structure to indicate overall information, paths, schemas etc.

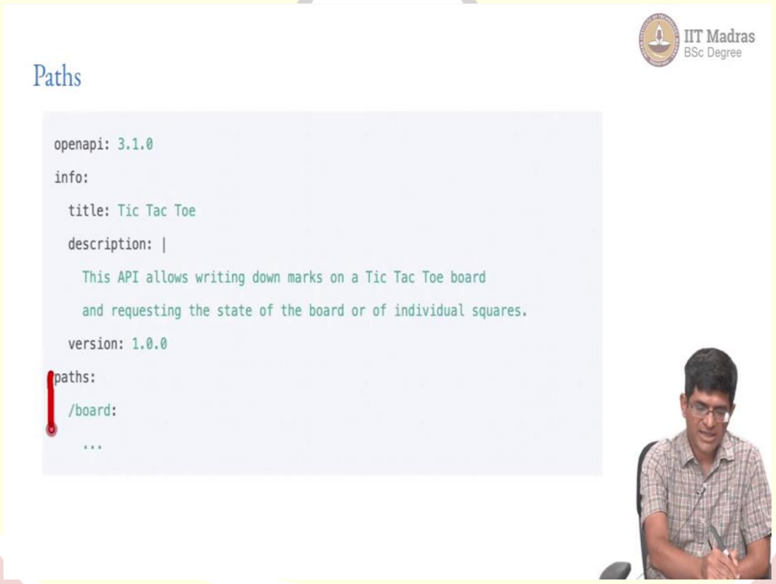
eg:

```
openapi: 3.1.0
info:
  title: A minimal OpenAPI document
  version: 0.0.1
paths: {} # No endpoints defined
```



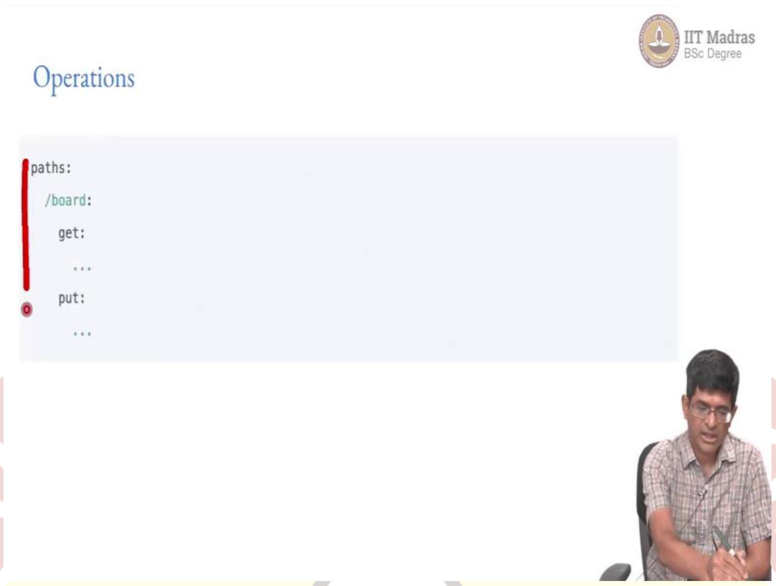
## Endpoints List

From: https://oai.github.io/Documentation/specification-paths.html


IIT Madras
BSc Degree

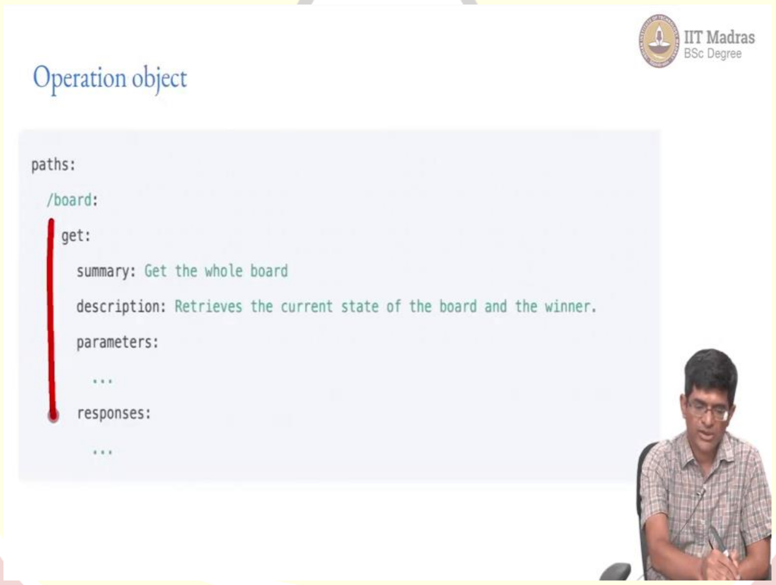
## Paths

```
openapi: 3.1.0
info:
  title: Tic Tac Toe
  description: |
    This API allows writing down marks on a Tic Tac Toe board
    and requesting the state of the board or of individual squares.
  version: 1.0.0
paths:
  /board:
    ...
```



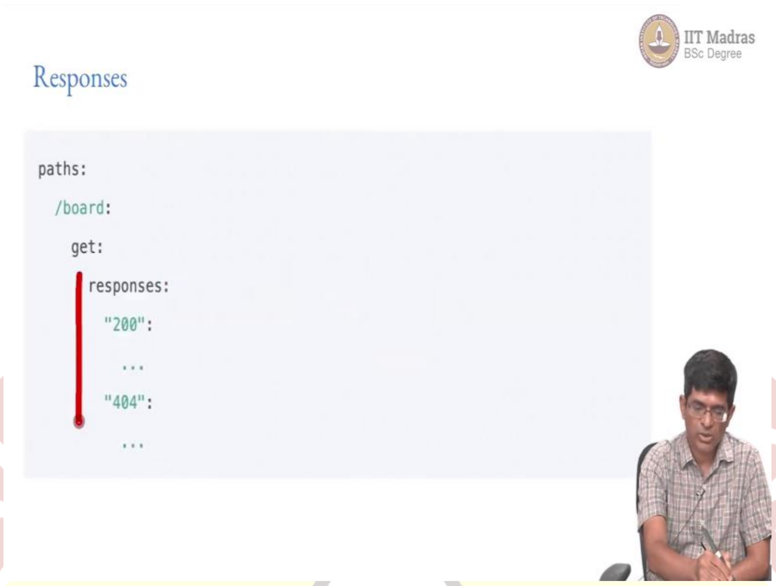
## Operations

```
paths:
  /board:
    get:
      ...
    put:
      ...
```



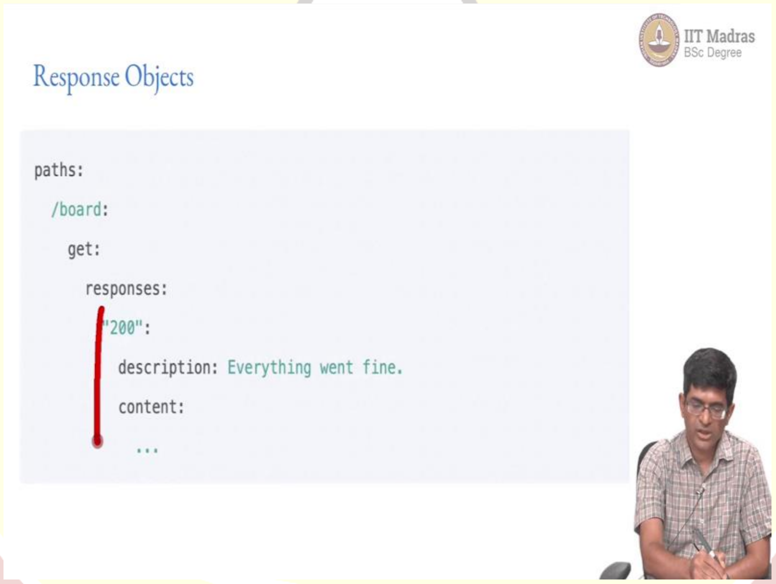
## Operation object

```
paths:
  /board:
    get:
      summary: Get the whole board
      description: Retrieves the current state of the board and the winner.
      parameters:
        ...
      responses:
        ...
```



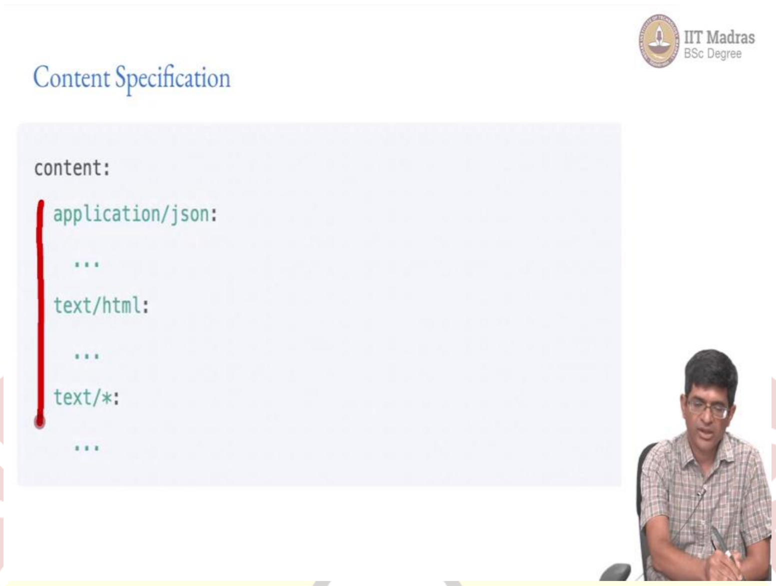
## Responses

```
paths:
  /board:
    get:
      responses:
        "200":
          ...
        "404":
          ...
```



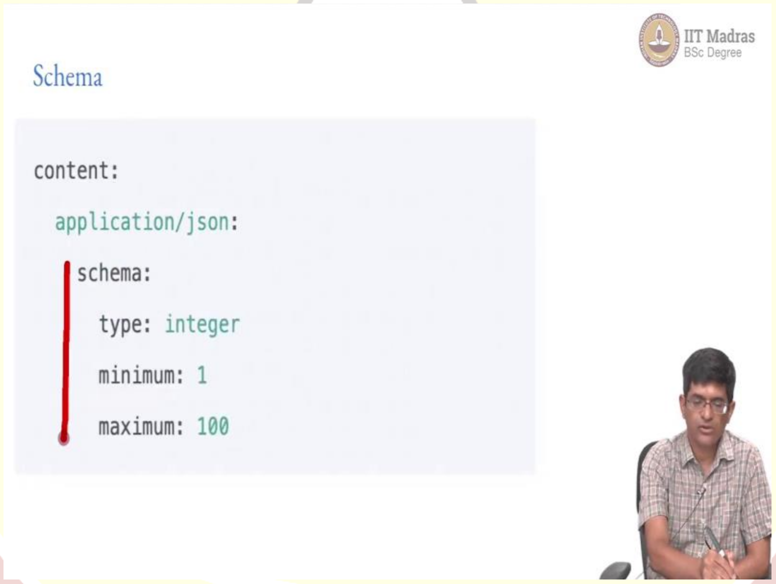
## Response Objects

```
paths:
  /board:
    get:
      responses:
        "200":
          description: Everything went fine.
          content:
            ...
```



Content Specification

```
content:
  application/json:
    ...
  text/html:
    ...
  text/*:
    ...
```



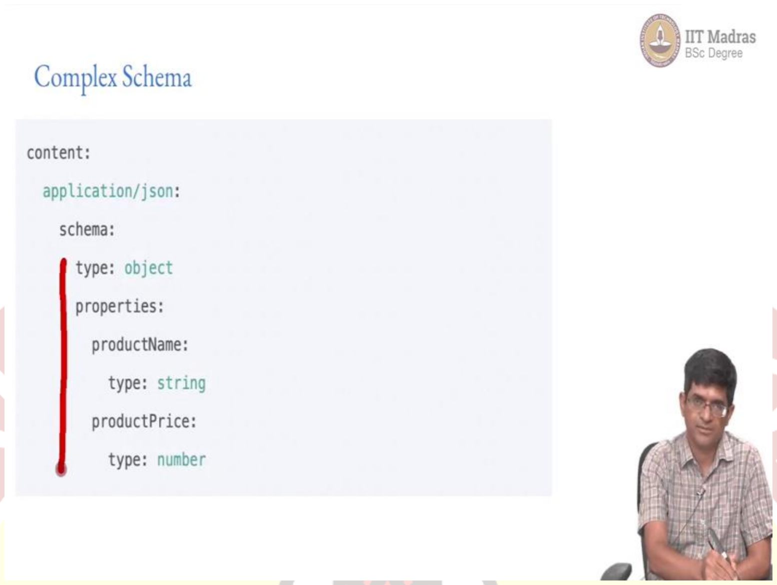
Schema

```
content:
  application/json:
    schema:
      type: integer
      minimum: 1
      maximum: 100
```

So, you can sort of see the flow over here we started all the way back from here, minimal document. We start off with this and we basically say these are the paths inside a path, you could have multiple different types of requests. Inside a request, you have different parts that describe it in more detail.

For the responses, these are the different codes that you can have. For a given code, you could have more information, including what type of content it takes for the content, you could have different types of application formats or content formats that it will accept and inside one of those, you might actually give a specification with a schema saying how the data is actually being represented.

(Refer Slide Time: 11:52)

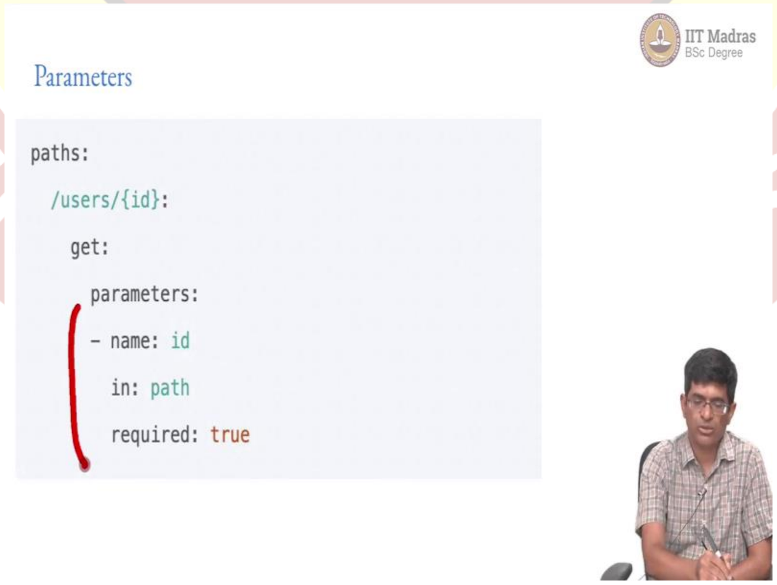


## Complex Schema

```
content:
  application/json:
    schema:
      type: object
      properties:
        productName:
          type: string
        productPrice:
          type: number
```

So, you could in fact, go a bit further than that with this schema, you could also have something which says this is actually a complex data type, which actually has, it is a type object with the following properties, it has a product name, which is of type string a product price, which is of type number and so on. In other words, the JSON object that you get back would be some kind of a complicated object, which will follow this scheme.

(Refer Slide Time: 12:19)

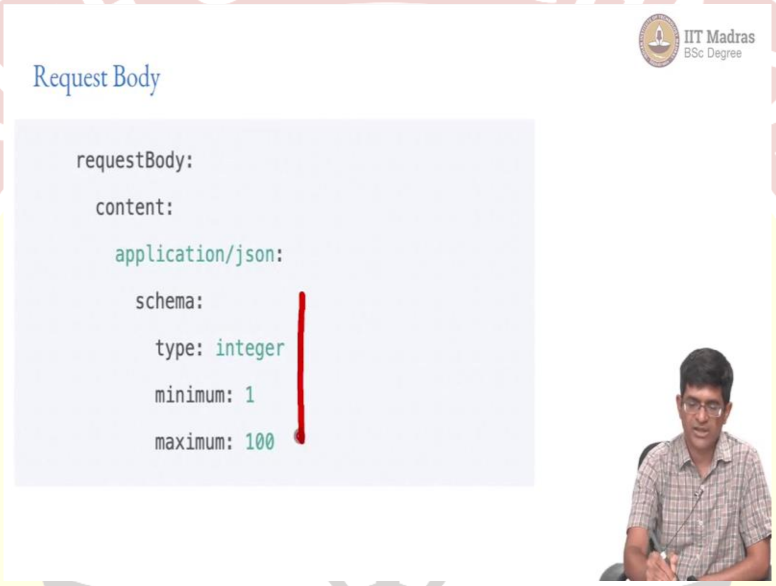


## Parameters

```
paths:
  /users/{id}:
    get:
      parameters:
      - name: id
        in: path
        required: true
```

Along with all of this, these are the responses that I was talking about, you might also have things which in the path can specify parameters. So, /users/{id} now says id is a parameter which is present in the path and is required. If you do not have it, if you just give /users without a /{id}, then there should be some error in the API or at least you should, the response will be that, I did not get the id this was required.

(Refer Slide Time: 12:48)

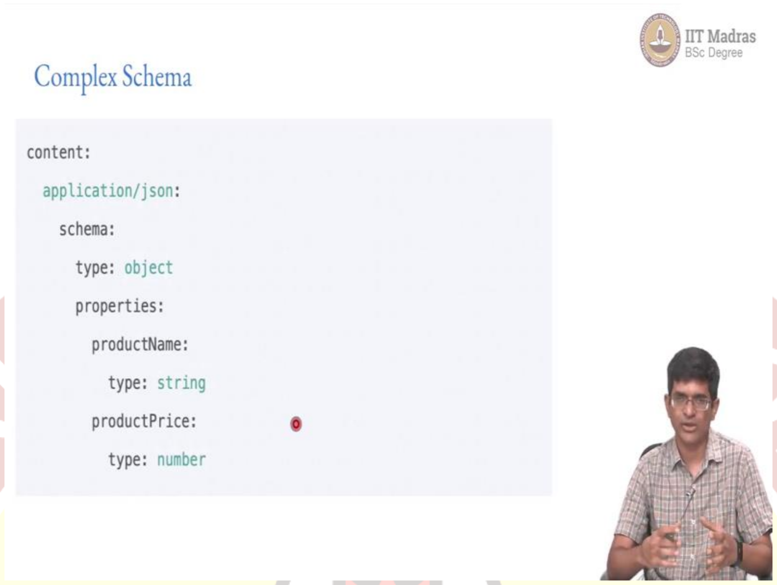


Request Body

```
requestBody:
  content:
    application/json:
      schema:
        type: integer
        minimum: 1
        maximum: 100
```

Similarly, you could also have something which says that the request body could have a certain kind of structure, it might itself be it is expected to be a JSON blob of text, which has some kind of a structure, which basically, in this case, is an integer. So, with all of this, let us sort of try and summarise where we are. What would be the main purpose of having a document like this?

(Refer Slide Time: 13:15)


## Complex Schema

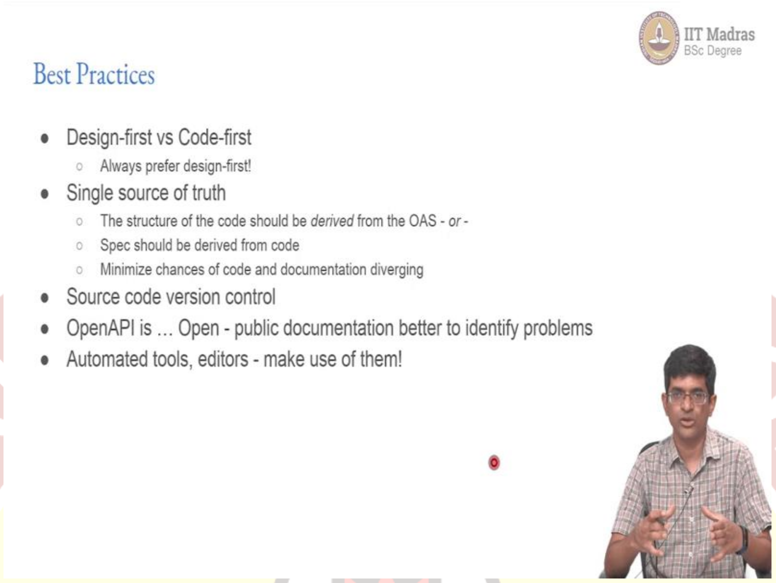
```
content:
  application/json:
    schema:
      type: object
      properties:
        productName:
          type: string
        productPrice:
          type: number
```

Let us, say that I have got to this point where I have all of this schema and everything else in place. As you can see, I could probably write a script, which will take in this YAML file and pretty much generate a lot of Python code for me, which implements almost all of these functions, except for the exact details of how to implement certain things.

So, for example how to create a board, how to display it, the final view, those kinds of things are still open. But the fact that the board needs to be created, that the function is needed for that API call is required over there, all of that information is already taken care of in the YAML specification. Not just that a lot of documentation is also present here, because the descriptions are present and from the descriptions, you can just basically extract out and create a, the kind of documentation pages that we saw for the CoWin and Wikipedia API's.

So, that essentially, is the problem that swagger is trying to solve. It has combined the documentation as well as the code generation into a single file. What is the catch? Once you try doing all of that YAML itself is complicated to write. It is not as bad as for example, writing XML. But on the other hand, it is not easy to get it right without any errors on its own.

(Refer Slide Time: 14:40)


IIT Madras
BSc Degree

## Best Practices

- Design-first vs Code-first
  - Always prefer design-first!
- Single source of truth
  - The structure of the code should be *derived* from the OAS - *or* -
  - Spec should be derived from code
  - Minimize chances of code and documentation diverging
- Source code version control
- OpenAPI is ... Open - public documentation better to identify problems
- Automated tools, editors - make use of them!

So, which brings us finally to what are known as best practices. And over here, finally, what this is sort of to summarise everything that we have about open API. The sort of core idea of something like an open API specification is to say, try and do design first, rather than writing code first. Design your system to have the right kind of API's, right kind of interfaces so that somebody else can use it before you start writing any code that is always important.

In general, the thing is you spend 20 minutes or 20 days or 20 months, whatever on design, versus 1 day on coding, it is much better than trying to write 5 days of coding without having thought about it, because you will then spend the next month or two, trying to debug your code and fix problems with it. The more time that you spend in the design and trying to get that path clean, the better will be your ability to debug it and to get it right the first time around.

Now, something like an open API specification, which can be read by humans, as well as by machines, in order to generate code, if necessary, is what is called a single source of truth. So, the idea behind a single source of truth is that if I just have one document over there, I can actually derive most of the structure of the code from that document using some kind of a script, if necessary or the documentation itself can be obtained from that document.

The other possibility is that I say my code is primary. The code is the most important thing, I already started writing the code this is great I know that I have done everything correctly.

Therefore, that is the single source of truth and all your documentation has to then come from that. Either way, avoid a possibility where you are writing code in a certain way documentation in a different way and they are diverging from each other.

Things like specification documents are ideally suited for version control. So, things like git should be used in order to manage specification documents, because they are things that can change with time and you do want to know, how they evolve with time what feature was added when, by whom and hopefully, if there is a suitable comment, for what reason.

And one other important thing about things like open API, remember that it is open and it is meant to sort of be public. Now, why is that important, because you might be designing an app, which only you intend to use, in which case, of course, you would cannot really be expected to make your API's public.

But just like Wikipedia, CoWin, GitHub, Google Cloud, all of those are showing, by making your API's public, it allows not just public documentation and people to understand easily what is happening, but it allows things to grow in a much more organic fashion, other people start to use your API's in ways that you could not have come up with or you maybe do not have the time to support.

And finally, coming back to the important problem, what I just mentioned, which is that YAML itself is difficult to write, do not try to write all of it. There are automated tools, there are editors, I am not sort of going to be discussing any of them specifically as part of these lectures. But using such editors and tools is an important part of making sure that you are able to get such the API standards done correctly.

In a way that makes it easy for somebody else to use. So, using such tools and sort of constructing code from it, constructing documentation from it, making sure all of them are in sync is a very important part of how you can design a scalable and an easy to maintain web application.