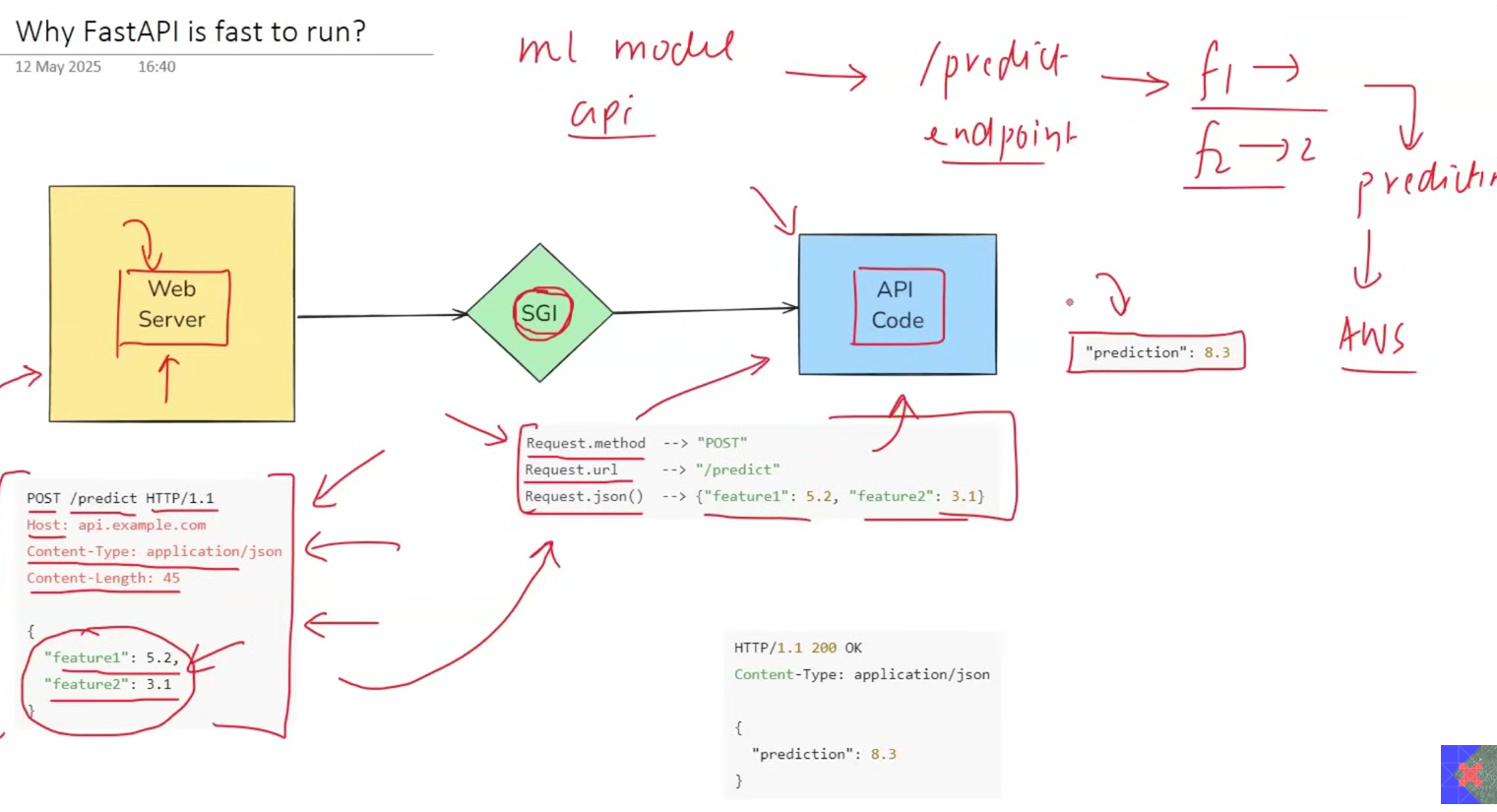


**Brief Picture of FASTAPI:**

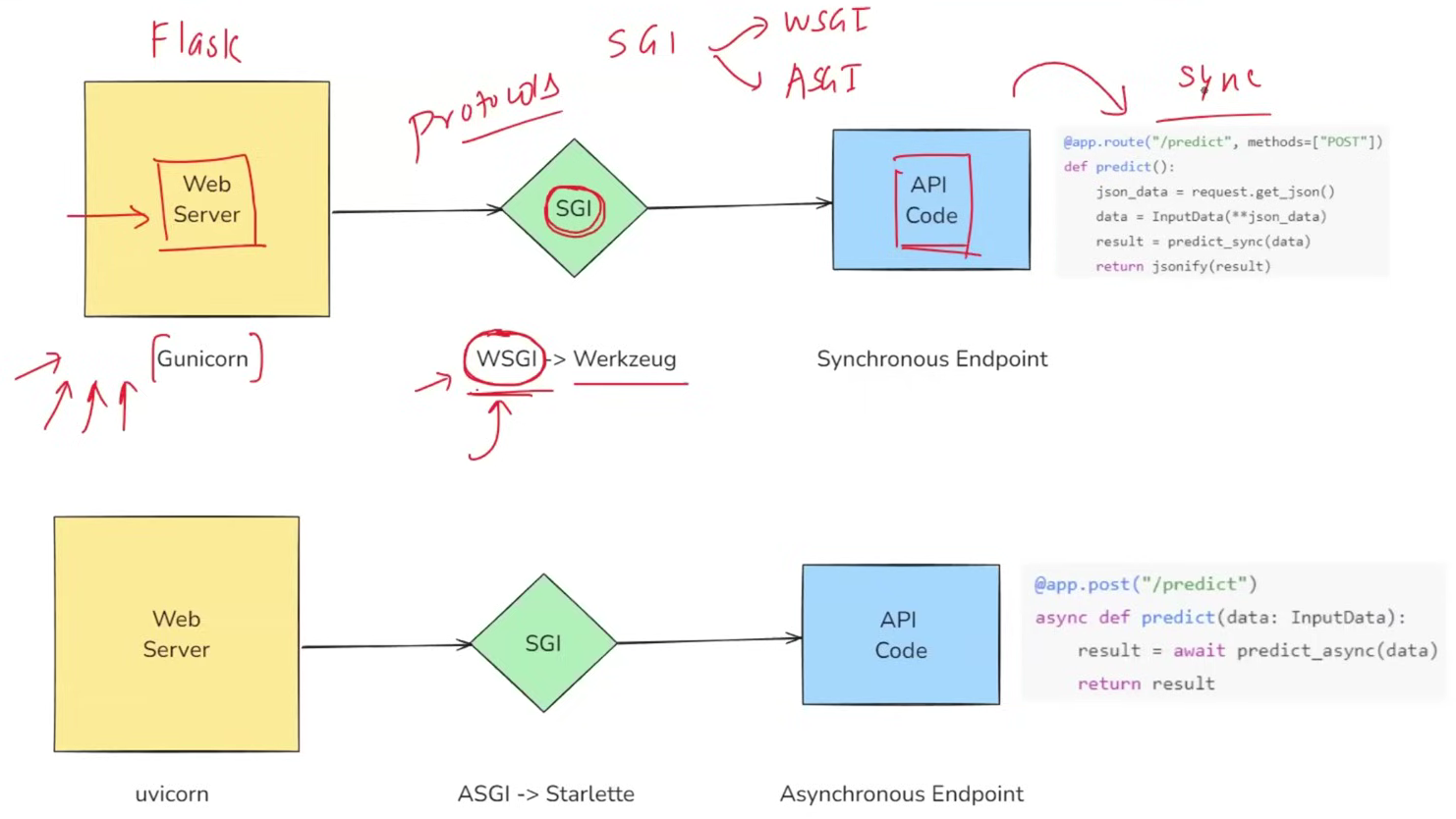
* We have three things at our AWS. (WebServer, API code and SGI [sever gateway interface])
* WebServer jo client se request ko leta hai aur process krta hai. (client ki request http form mei hoti hai)
* API code (jo ki python mei likha hua hai), aur yeh voh input leta hai jo API nei define ki hoti hai.
* SGI is a translator jo ki http request ko python mei convert krta hai and vice-versa.
* Taking one Example scenario how the things to be done, suppose we have one ML model which takes two input features and the predict function will predict the output based on those features, ok so now we understand how FASTAPI will process this request:
  + Client se request aya webserver ke paas http format mei (basically feature1 and feature2 ki value).
  + SGI nei http request ko python mei convert kr diya aur API code ke paas bhej diya (basically features ki values ko extract krke API code ko as an argument pass kr diya).
  + API code ne features ki values ko liya aur predict function ko de diya.
  + Predict function ne output diya than yeh user ko show krane ke liye isko http language mei convert krna pdega, so SGI nei convert kr diya and webserver ko wapis bhej diya.
  + And aage webserver nei predict value client ko dikha diya, SO this is the whole cycle of FASTAPI.

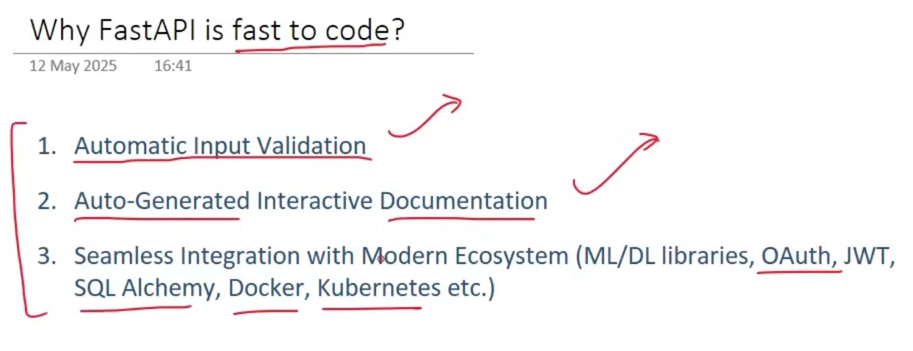


**Compare FLASK with FASTAPI:**

In FLASK the protocol is going to used is WSGI i.e. Web Server Gateway Interface, and it is based on the Werkzeug. The webserver used in FLASK is Gunicorn and FLASK has Synchronous endpoint which means at a time it will take only one API request and until the result will not come it will not encounter the another API request (even it the ML model is processing) so this was the biggest demerit of FLASK, so to resolve this synchronous endpoint issue we have FASTAPI.

FASTAPI protocol uses ASGI ie. Asynchronous SGI, and it is based on Starlette. The webserver is used in FASTAPI is uvicorn and best part of the FASTAPI is, its endpoint is Asynchronous in nature which means it handle multiple API requests at a single time. Like jab ML model process kr rha hoga thab yeh next API request ko consider krega. Aur image mei ‘async’ and ‘await’ dekh skte ho, iski help se multiple request ko consider krwa skte ho.

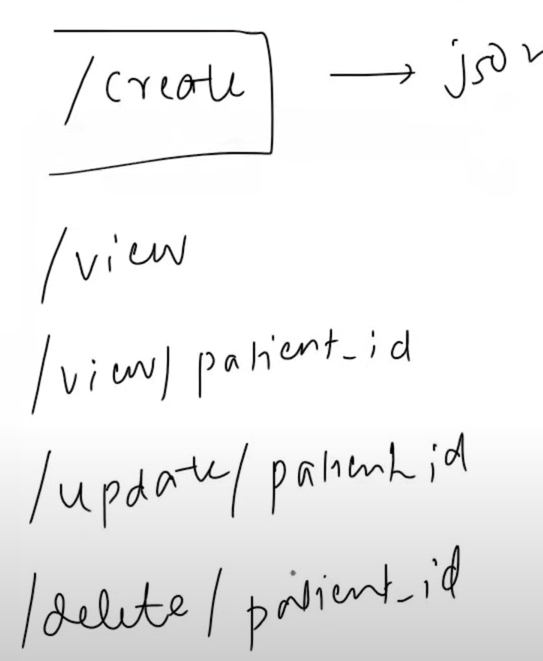


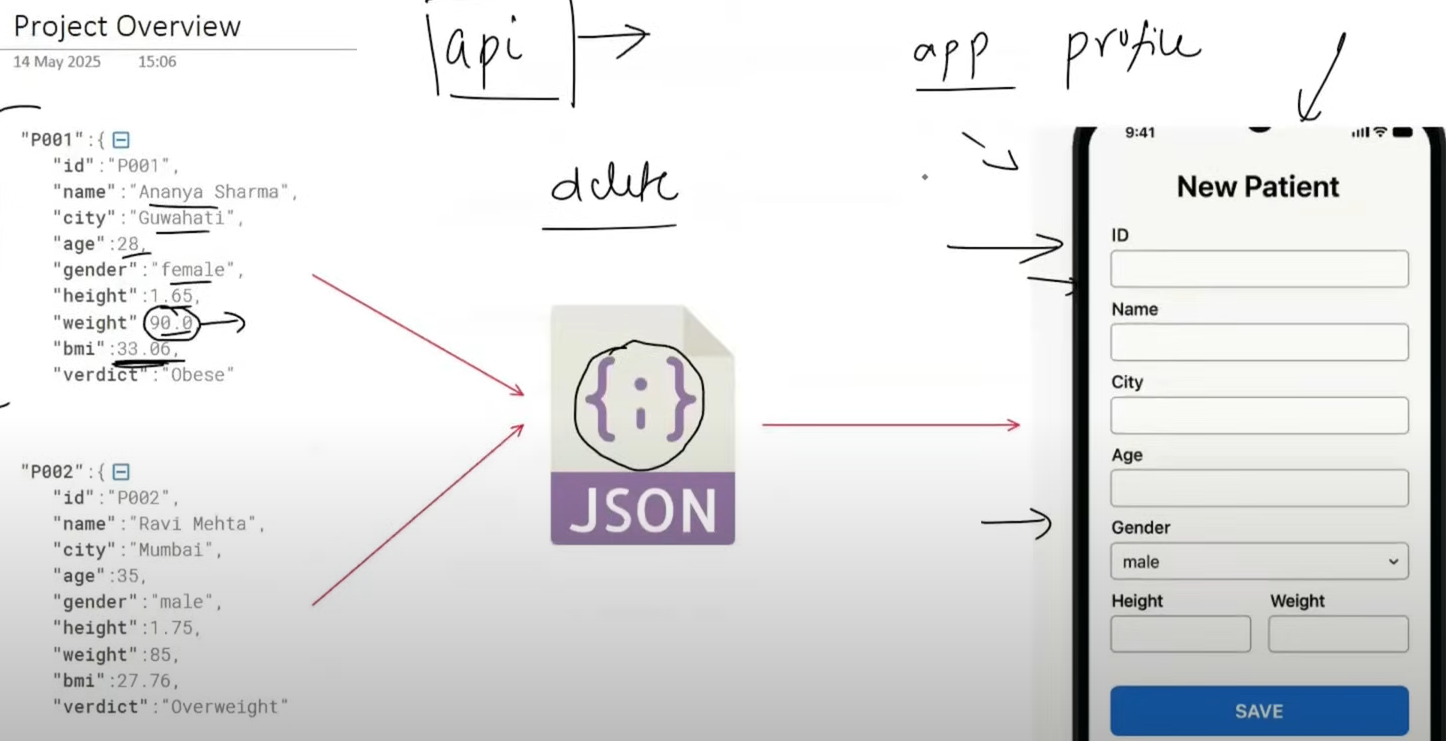
****

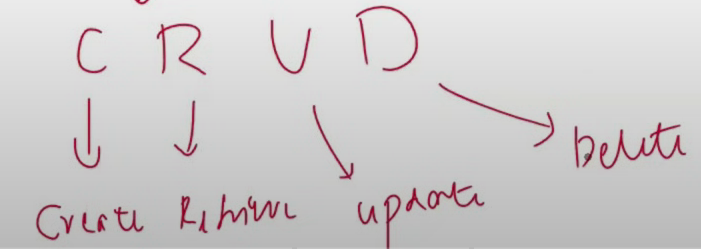
**Command to run FASTAPI Server**

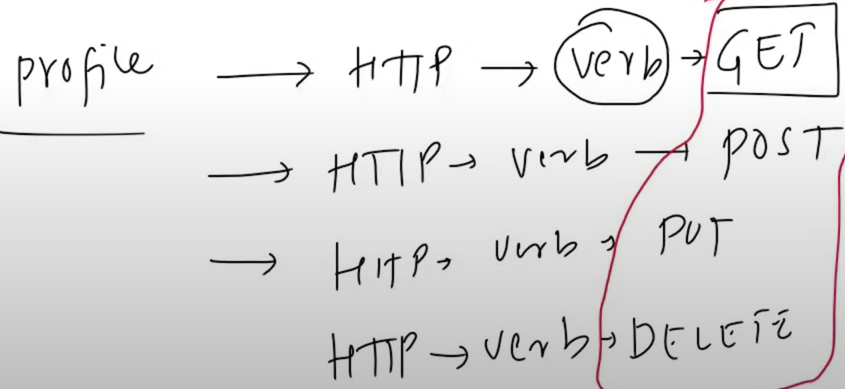
, here app is our FASTAPI object name and main is the python file name.

**Patient Management API Project**

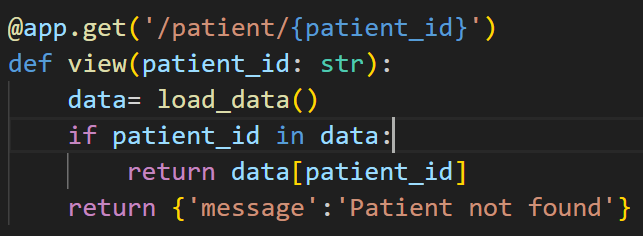
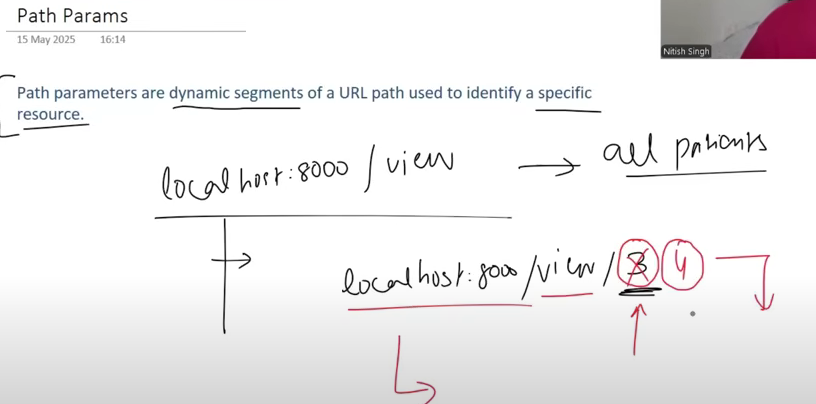
 These are the http request API we are going to create.



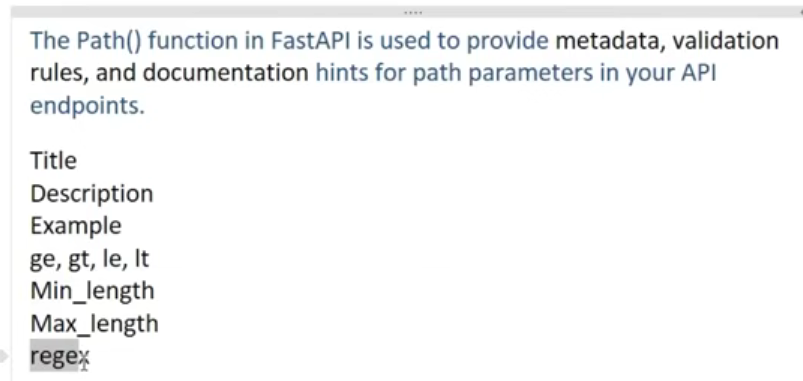
 Dynamic Software has only these four operations.



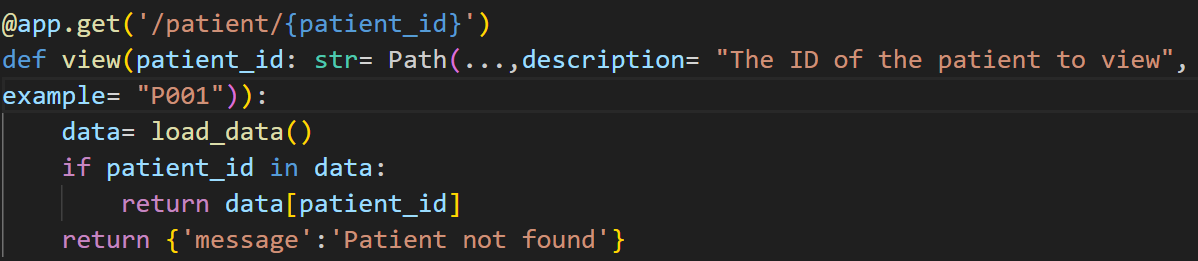
**Path Parameters:**

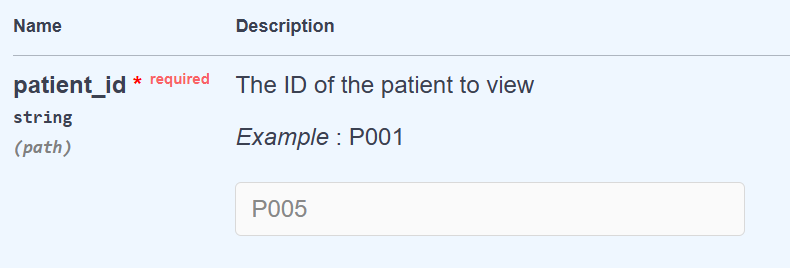


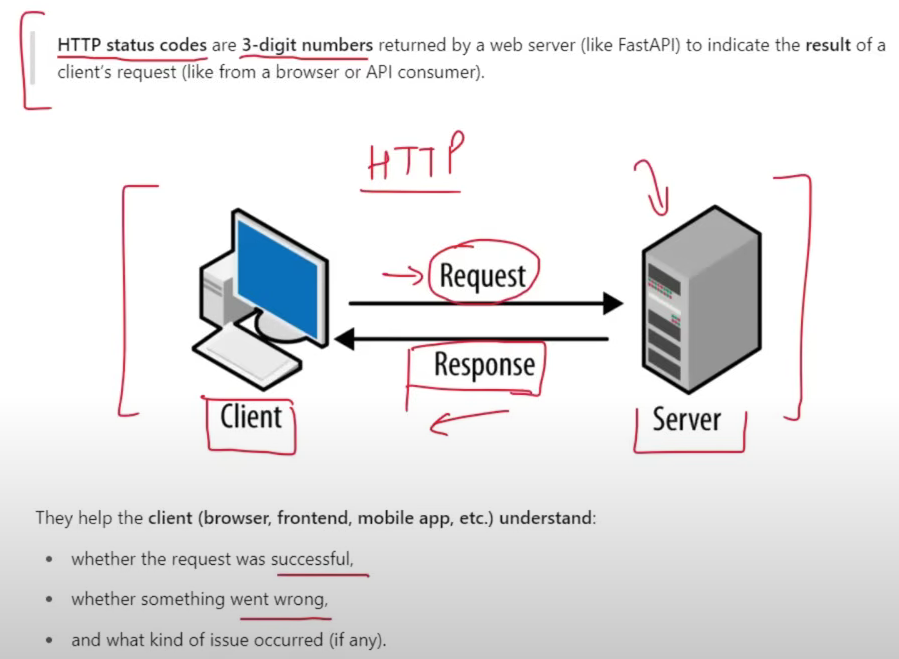
Here in the code you can see that patient\_id variable is dynamic in the url means particular value of patient id it will display the patient detail of that particular patient.

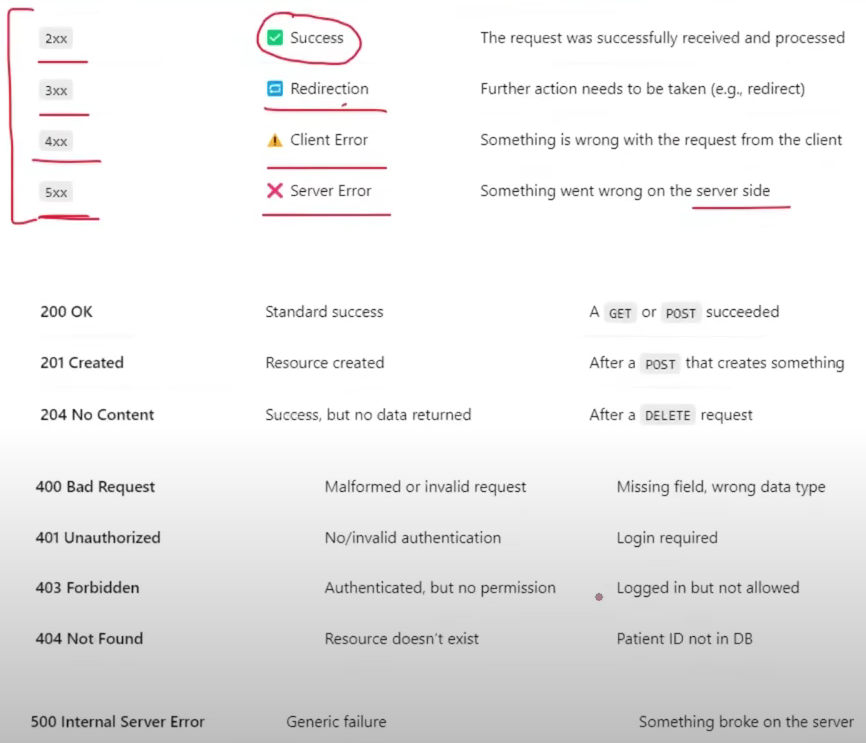
**Path function:** ge means greater than, lt means less than similarly gt and le. This is used to apply validaton or add constraint in any function.****

Three dots in the Path fxn means it is required. Similarly if we want to add more validation like ge, gt so we can add that simply by separating with comma in Path function.(Below code)

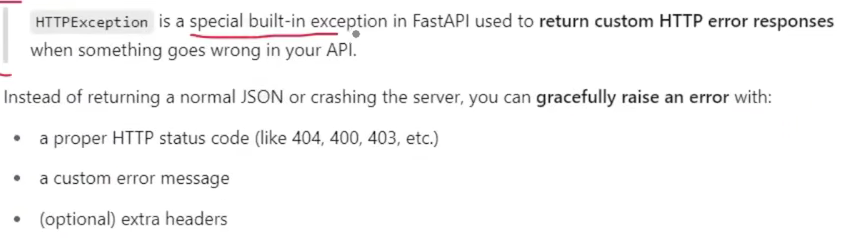
 For this code we get the below output, by using description we are helping our customer to knew him what is required in this particular field.

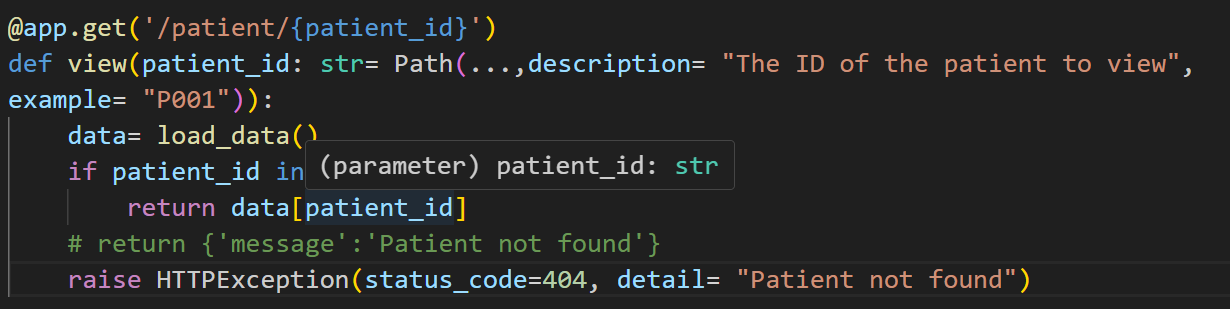
****

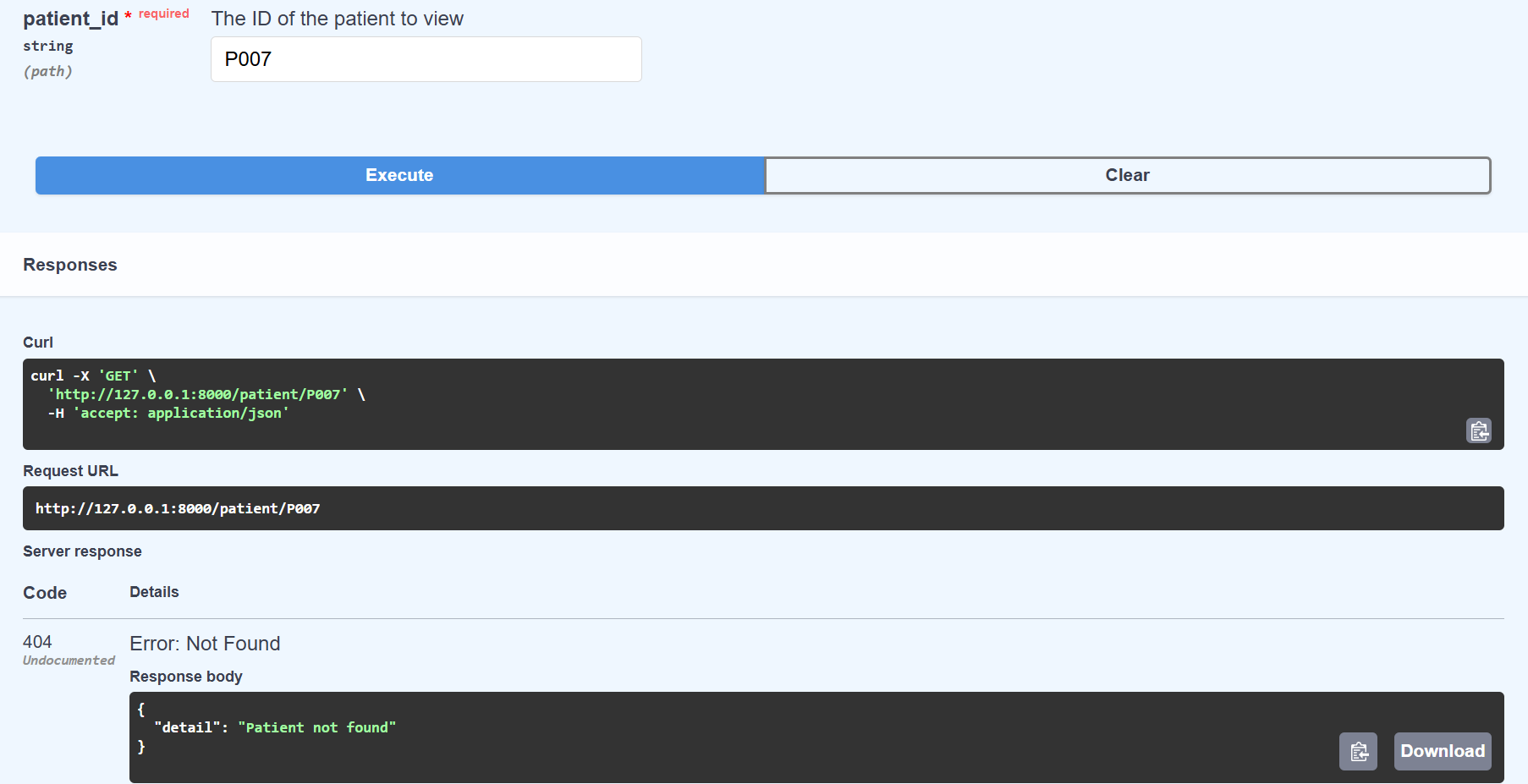
**Status Code**

****

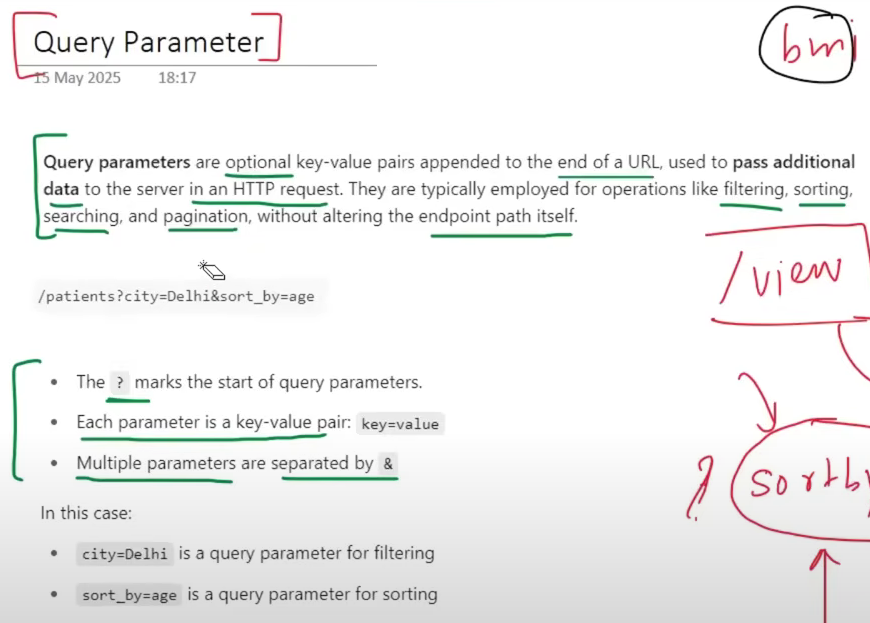
**HTTPException:** HTTPException hum thab use krte hai jab hume output mei eek relevant exception dikhana ho aur python mei yeh eek special class hai jise use krke hum easily HTTPException use kr skte hai.

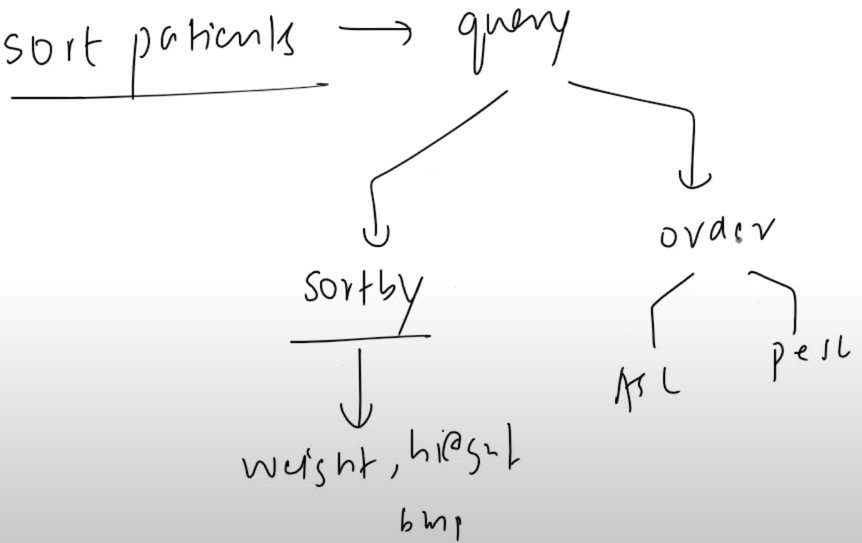






**Query Parameters**



Idea to build in code: 

Just like Path function in path parameter, we have Query function in query parameter.

