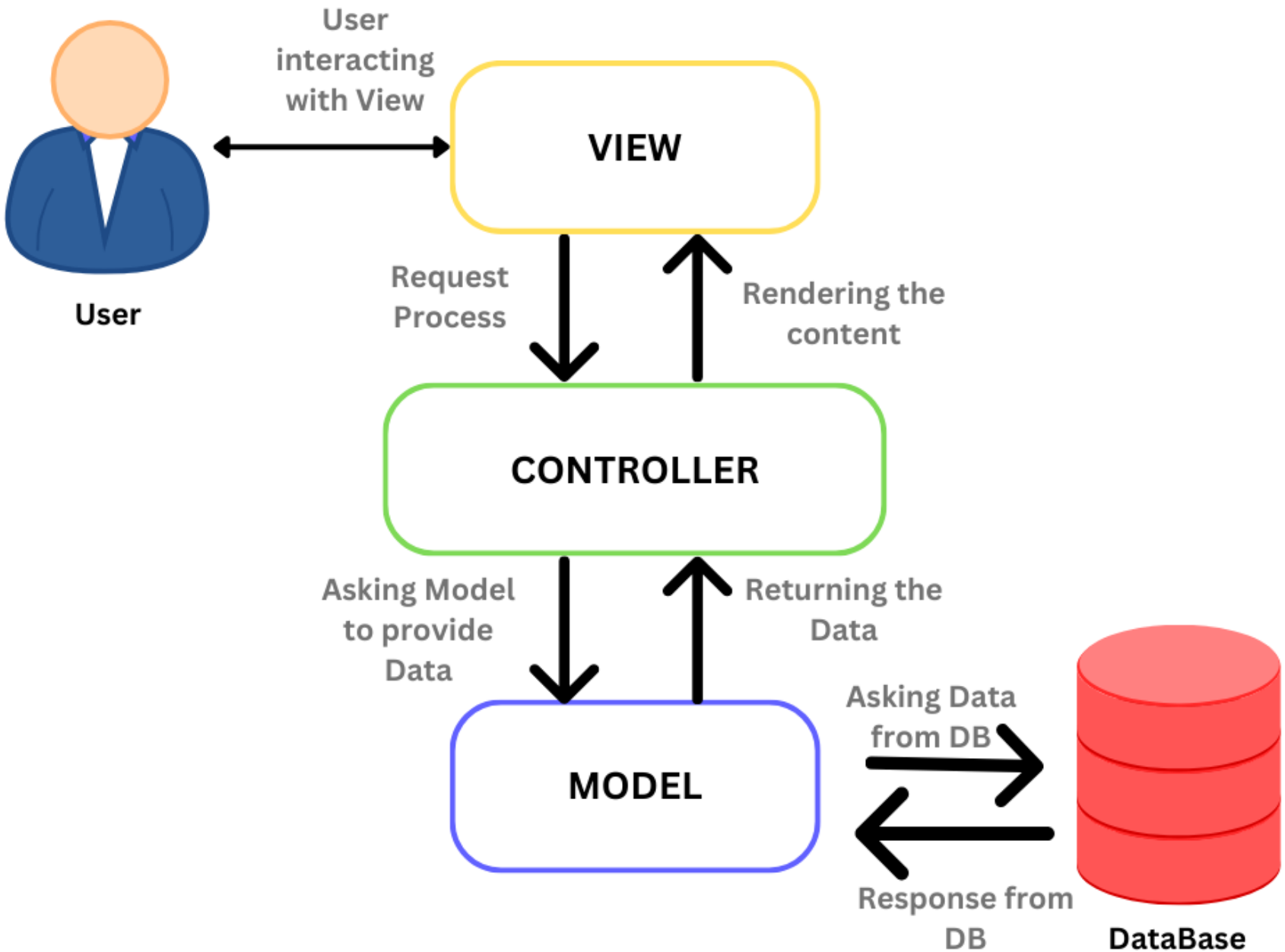
Model View Controller Architecture

A brief simple explanation of the complex concepts of the MVC Architecture and Frameworks

What is MVC architecture

- Def: Architectural paradigm followed by developers for the organization of the backend code into separate folders and files based on different purposes and jobs done by each part of our code, instead of having most of our code functionalities in just one file.
- M=> stands for the model which is a folder that includes files that contain just the part of our code to talk and interact with the database
- C=> stands for controller which is a folder that includes file which contains just the part of our code for interacting with both model and view upon listening to a request coming from the user then responding back.
- V=> stands for views which is a folder that includes files which contain just the part of our code for viewing and what stuff is going to be displayed and how as a final product for the user to see through a template that generates dynamic HTML elements.


User
interacting
with View
VIEW
User
Request
Process
Rendering the
content
CONTROLLER
Asking Model
to provide
Data
Returning the
Data
MODEL
Asking Data
from DB
Response from
DB
DataBase

Analogy

- When we are writing code at almost we try spontaneously to organize our code by separating each part by indenting a bunch of lines that are specified to do a specific thing from other parts of our code based on different purposes of each one just to make it easy for us to walk through our code and don’t take much time to know where to go when you want to edit something in it. To get it much better why MVC and what it helps us with let’s imagine a house sectioned into different kinds of rooms for different purposes so we know where to go when we want to sleep or prepare food instead of having a great open hall for doing a lot of different things.

Analogy

- just like that by following the MVC convention we can organize our code into specific folders for including files containing specific parts of our code responsible for doing similar things so we know where to go to edit or change a specific thing in our code.
- Instead of having all roads lead to Rome or our server.js like in our case, we have Model leads us to the DB, Views leads us to what we're going to display to the user to make requests, Controller leads to us both the Model and Views.

What scenario will MVC be best for?

- Makes it easy to work as a team on developing stuff at the same time
- We don’t worry when we need to change or edit something that our code could break and get confused about what and where the reason that causes it to break.
- Makes it easier to work together to build.
- If there’s a problem with one part of our code our entire code won’t break and it become much more easy to detect where the problem.

conclusion

- The whole point of the MVC paradigm is the separation of concerns

Because Modern web applications are very complex, and making a change can sometimes be a big headache.

- Managing the different parts of our code in smaller, separate components allows for the application to be scalable, maintainable, and easy to expand.