Service side process – This is when a **server** return the query of a user. For example: The user requested to give the results of the filtered data which are (Pediatrician and Male). A DMBS will perform a query:

SELECT FROM TABLE DOCTOR criteria Male and a Pediatrician.

A DBMS will perform this searching and sorting process **EVERY TIME** the user requested. Which means, every time a new user will come to request for that specific query, the database will update itself over and over again.

The problem arises when there are 10000 users accessing and requesting different query at the **same time**. The DBMS will then reach its threshold and will bottleneck.

Can the DB handle 1000 concurrent users sorting at the same time?
Can the DB handle 1000 concurrent users searching at the same time?

It has been found that databases have certain limitations when performing operations

Database Server busy. Request timeout.

Client side process – This is when **a user** filter (sort) out the data which are (Pediatrician and Male) instead of the server with its database functions. All of the calculations be done on the client side.

This is proven to be a solution when a dbms server had certain limitations and can't accept concurrent request to thousands of users.

This work by a client having all the data, which then be sorted through his device in any way he likes. The user's device is the one be responsible to the sorting and searching process.

As per stackoverflow:
First, there's the issue of the "right" place to do it from a design point of view. Most of the time the order of data **isn't a business rule thing** but rather a **end-user convenience thing** (end-user preference) , so I view it as a function of the presentation, and I don't like to push presentation issues into the database. (Which means we don't need to sort the database over and over again as per the user request. Because everytime a user requests for a specific details using server side process (database function) it will update itself (for example sort by name) and expensive and time consuming.

We choose client side process because of:

Cost – Database adds to business cost. For example a web server using Amazon Relational Database System (Amazon RDS) it is a pay as you go pricing. If you pay for the database then it will cost more. Let the client to the calculation. (e.g. sort via client side.)

Data Transfer – This business model will work on companies where data transfer cost is neglicted. (In client side process, the user needs to download all the data, which is an amount of data transfer)

Faster Round trip –

In server side =

Request -> Query -> Transfer data -> display

In client side =

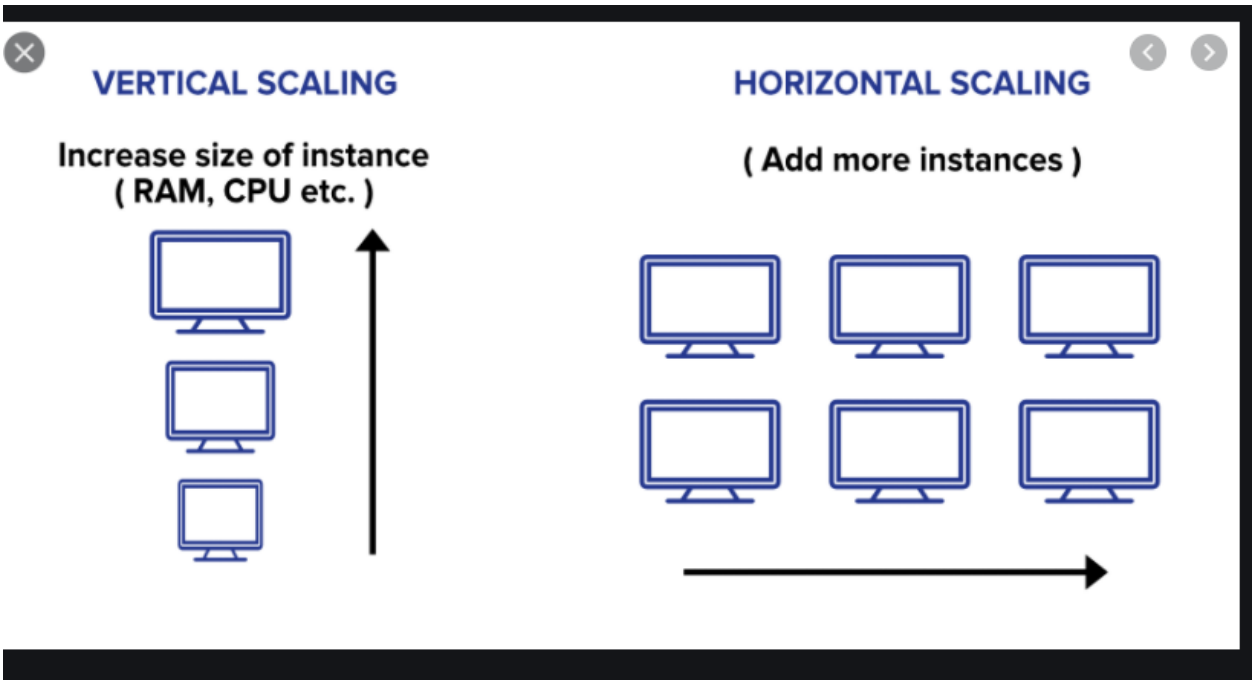
Transfer data -> display

Scalability - will scale to more clients since you don't need to consider dbms limits. Let the client download the data and sort it through their side.

As per stackoverflow:

The second reason I prefer sorting in the client layer is one of performance. Web servers scale horizontally, that is, if I

overload my web server with users I can add another, and another, and another. I can have as many frontend servers as I need to handle the load and everything works just fine. But, if I overload the database I'm screwed. Databases scale vertically, you can throw more hardware at the problem, sure, but at some point that becomes cost prohibitive, so I like to let the DB do the selection, which it has to do, and let the client do the sorting, which it can to quite simply.



And most importantly, we can’t use built in functions to our project since the project required us to develop our algorithm without the help of databases or any other programs.

If you let a database server sort through your data it is a server side process.

If you let you let your mobile device or computer sort though your data it is a client side process.

And we don’t want built in functions sort our data, since we need to implement our algorithm.

Javascript is a client side language. Meaning all of the queries are run through the user. Which means you need to develop the algorithm via JS.

PHP is a server side language. Meaning all of the queries are executed in the server.

On the project:

https://github.com/kishan0725/Hospital-Management-System

- uses server side query in sorting, search, inserting, etc. data. It is mainly done it using a database.

System Weakness:

Uses database function in operations.

What can we improve:

Implement our algorithms , replacing the database functions in this system. Sorting and Sorting.

REALTALK: Mas mabilis ang sorting process ng mga database systems, as in sobrang bilis. Sobra optimize na yan. Kasi anjan na lahat ng mga matatalinong tao na nagtulong tulong to make the accessing of data optimized. As an example Client side 28 seconds using XML or JS for sorting, and Server side sort total load time 3 seconds. **Yes, but we are focus on making a more efficient client sorting process.**

Ngayon ang gagawin natin, instead na lalamangan natin ang speed ng database functions which are a server side process. We will discover and propse the best algorithm naman without using database functions and server side process.

Parang main goal ay: best algo implementation in client side process.

So here we are proposing: instead of using built in functions from a database (like sorting) and storing the data. We will let our algorithm sort it **Oyelami Sort** by reading JSON data.

All of the data should be in JSON format. So it can be downloaded to the user side and sort it via Oyelami Sort developed using Javascript. Binary search too.

You can't use JAVA or Python or C in web based apps. JS is the only language that will work. So algorithm needs to be in JS (javascript) form.

You need to learn how to read from JSON data.

You need to learn how to write from JSON data.

You need to learn how to sort from JSON data.

You need to learn how to implement Oyelami Sort.

You need to learn how to implement Binary Search.

UI is done. We just need to improve the system by switching to client side process and developing our algorithms.