**Search on backend or frontend**

Talking about trip searching , there are two ways to implement it .

1. filter logic is on front end

We get all the trip data from back end at once , and we filter it on the front end based on user search . In this case we don’t need to send the data to backend for every user search and wait for response . For every user search we simply filter and change the feedback on front end , which will certainly make our app reacts much faster .

2. filter logic is on back end

In this case we will call our backend every time when a user perform a search and the backend will do the filter job and only send back the trip data that we want . Since we are sending the data back end wait for the response for every time ,the speed could be slower in this case .

//https://stackoverflow.com/questions/52346685/filters-logic-should-be-on-frontend-or-backend

The truth is we should considering the amount of data . If we had a million trips in our database , and a hundred thousand users trying perform the search at the same time, would you really want to send a million records to EVERY user? It'd kill your server and user experience (waiting for a million records to propagate from the back end for every user AND then propagate on the front end would take ages when compared to just getting 20-100 records and then clicking a (pagination) button to retrieve the next 20-100). On top of that, then to filter a million records on the front-end would, again, take a very long time and ultimately not be very practical.

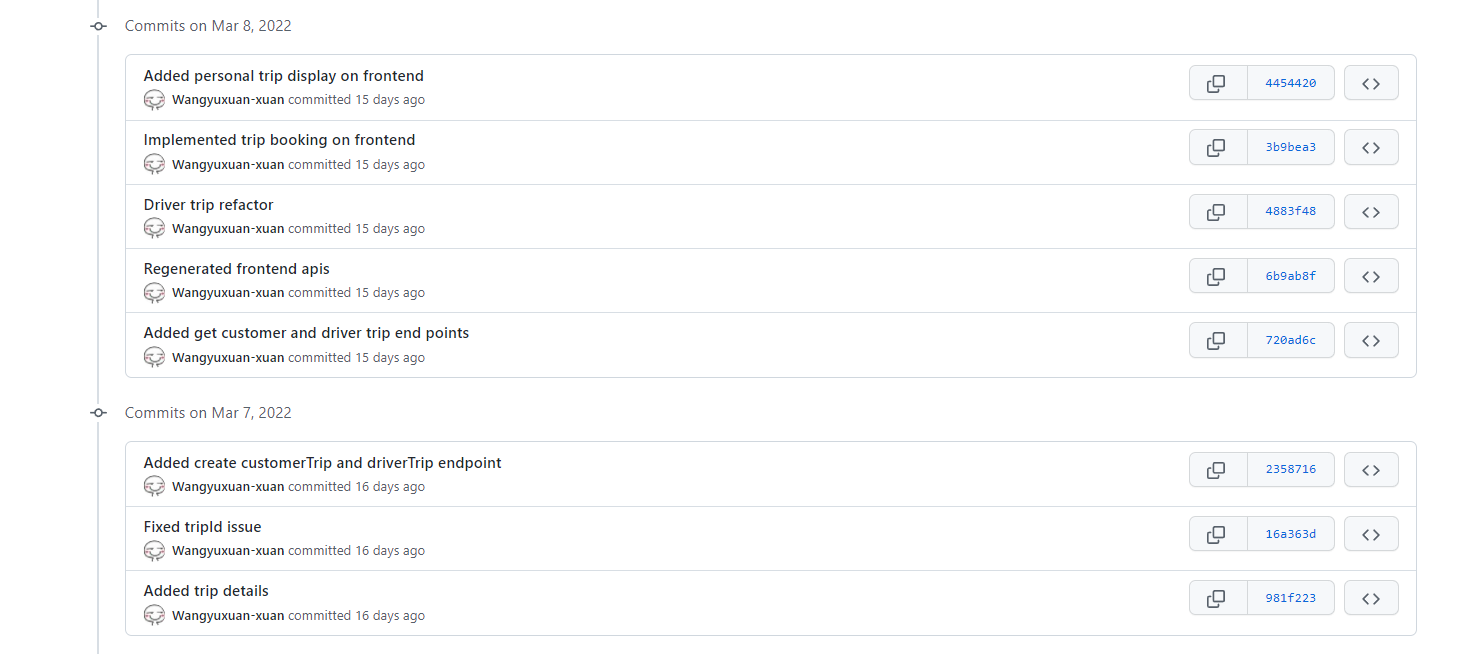
If we only have a small amount if data and are requesting frequently, they filtering on front end would be a better choice , since it avoids frequent requests and response and keeps user expirence .

From a real world stand point, most websites have some sort of record limit: Ebay = 50-200 records, Amazon = ~20, Target = ~20... etc. This ensures quick server responses and a smooth user experience for every user.

In our occation , we should keep our data filter in backend , which is the sever side . Since our trip data can be really big and we can have many users to request for it at the same time . It’s not wise to send all the data to every user .Would be a big burden for the sever .

#### Git Version Control

As we are developing our software , one thing we can not miss is git version control . Git is quite useful when we are developing large applications . It is more helpful and important when we are working in a team .



We can easily see our develop flow .

###### Commits

//https://www.freecodecamp.org/news/10-important-git-commands-that-every-developer-should-know/

This is maybe the most-used command of Git. Once we reach a certain point in development, we want to save our changes (maybe after a specific task or issue).

Git commit is like setting a checkpoint in the development process which you can go back to later if needed.

We also need to write a short message to explain what we have developed or changed in the source code.

###### Push & Pull

After committing your changes, the next thing you want to do is send your changes to the remote server. Git push uploads your commits to the remote repository.

The git pull command is used to get updates from the remote repo. This command is a combination of git fetch and git merge which means that, when we use git pull, it gets the updates from remote repository (git fetch) and immediately applies the latest changes in your local (git merge).

git pull <remote>

This operation may cause conflicts that you need to solve manually.

###### Branches

Branches are highly important in the git world. By using branches, several developers are able to work in parallel on the same project simultaneously. We can use the git branch command for creating, listing and deleting branches.

Creating a new branch:

git branch <branch-name>



//https://www.nobledesktop.com/learn/git/git-branches

When you've completed development in your branch and everything works fine, the final step is merging the branch with the parent branch (dev or master). This is done with the git merge command.