Functionality Investigation Sheet
Les petits plats
Feature No. 2 Search Engine

1.Problem:

The search bar does not filter in detail the results of the recipes by keyword categories:

- Ingredients.
- Device.
- Appliance.

2.Options

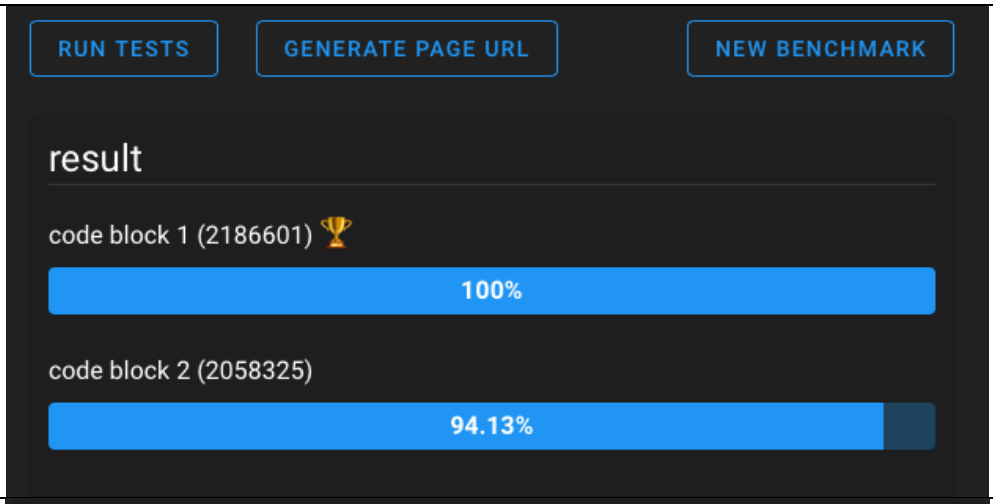
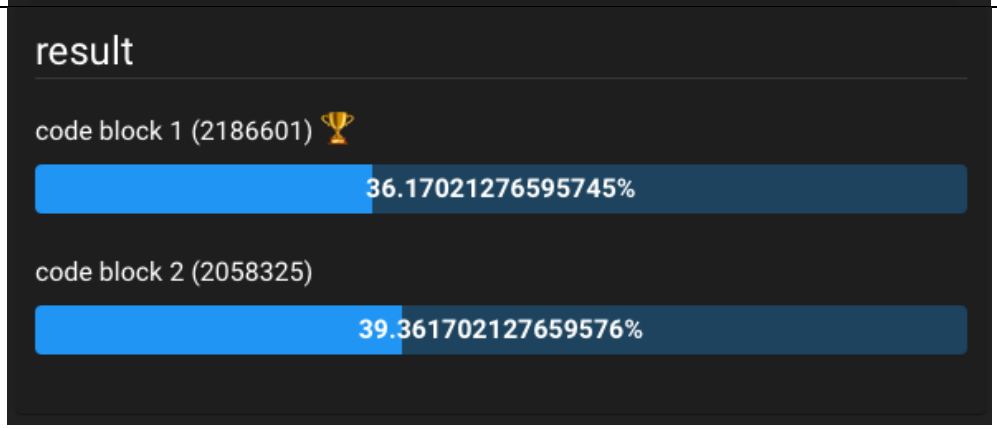
<table>
<tr><th>Option</th><th>Characteristics</th><th>Analysis</th></tr>
<tr><td>1</td><td>Wide search that algorithms covers every single character over the json file to find matching words. Delivers narrowed search results
<pre>
function functionSearch() {
    var inputSearch, filterSearch, cardsSearch, cardContainer, mySearch, i;//OK
    inputSearch = document.getElementById("searchFilter"); //Input search //OK
    filterSearch = inputSearch.value.toUpperCase();
    cardContainer = document.getElementById("items");//Container d//OK
    cardsSearch = cardContainer.getElementsByClassName("card");//Targeting Card //
    for (i = 0; i < cardsSearch.length; i++) {
        mySearch = cardsSearch[i].querySelector(".card-body"); //Modify lenght
        if (mySearch.innerText.toUpperCase().indexOf(filterSearch) > -1) {
            cardsSearch[i].style.display = "";
        } else {
            cardsSearch[i].style.display = "none";
        }
    }
}
</pre></td><td>Benefits:
Faster Search.
Disadvantages
Narrowed search results</td></tr>
<tr><td>2</td><td>Exhaustive algorithm that finds matches by the 3 first characters and refines the search by additional keyword matching from the attributes in json file and can be refined as much the user needs.
<pre>
function functionSearch() {
    var inputSearch, filterSearch, cardsSearch, cardContainer, mySearch, i;//OK
    inputSearch = document.getElementById("searchFilter"); //Input search //OK
    filterSearch = inputSearch.value.toUpperCase();
    cardContainer = document.getElementById("items");//Container d//OK
    cardsSearch = cardContainer.getElementsByClassName("card");//Targeting Card //
    for(let i = 0; i < cardsSearch.length; i++) {
        cardsSearch[i].classList.remove('hide')
    }
    if(inputSearch.value.length > 2) { //Defined lenght
        for (i = 0; i < cardsSearch.length; i++) {
            mySearch = cardsSearch[i].querySelector(".card-body"); //Modify lenght
</pre></td><td>Benefits:
High refined search for the user according to his/ her tailormade needs.
Disadvantages
It shows tons of results according to the keyword matches but still good for a DB with only 51 objects.</td></tr>
</table>

```
            if (mySearch.innerText.toUpperCase().indexOf(filterSearch)
> -1) {
                cardsSearch[i].classList.remove('hide')
            } else {
                cardsSearch[i].classList.add('hide')
            }
        }
    } else if (inputSearch.value.length === 0) {
        for (i = 0; i < cardsSearch.length; i++) {
            cardsSearch[i].classList.remove('hide')
        }
    }


    //filter ingredients
    let ingredientsArray = tags.filter(tag => tag.category ===
'ingredient').map(r => r.name)
    for(let i = 0; i < recipes.length; i++) {
        if(cardsSearch[i].classList.contains('hide') === false) {
            let ingredientsMatchs = []
            for(let m = 0; m < ingredientsArray.length; m++) {
                ingredientsMatchs.push(recipes[i].ingredients.map(x =>
x.ingredient).includes(ingredientsArray[m]))
            }
            if(ingredientsMatchs.includes(false)) {
                cardsSearch[i].classList.add('hide')
            }
        }
    }



    // filter device
    let deviceArray = tags.filter(tag => tag.category ===
'appliance').map(r => r.name)
    for(let i = 0; i < recipes.length; i++) {
        if(cardsSearch[i].classList.contains('hide') === false) {
            if(deviceArray.includes(recipes[i].appliance) === false &&
deviceArray.length !== 0) {
                cardsSearch[i].classList.add('hide')
            }
        }
    }

    // filter ustensils
    let ustensilsArray = tags.filter(tag => tag.category ===
'ustensils').map(r => r.name)
    for(let i = 0; i < recipes.length; i++) {
        if(cardsSearch[i].classList.contains('hide') === false) {
            let ustensilsMatchs = []
            for(let m = 0; m < ustensilsArray.length; m++) {

ustensilsMatchs.push(recipes[i].ustensils.includes(ustensilsArray[m]))
            }
            if(ustensilsMatchs.includes(false)) {
                cardsSearch[i].classList.add('hide')
            }
```

```
        }
    }
}
```

3. Solution retention

According to the comparison of the algorithms by the tool https://jsben.ch/TG6y6
And it results, I have decided to implement the second algorithm listed on this document that is the one that shows the trophy.

Results	Evidence
The first comparison shows that the first algorithm is 100% more trustable related to search performance compared to the 2nd one what is under 6.37% percent of performance.	
The second comparison of both algorithms shows with a numerical result, that the first algorithm is better because it shows that is faster to search because it gets the search done in 36.17s that takes to run all the function; compared to the second algorithm that takes a total of 39.36 seconds to run	

4. Flow chart of option one

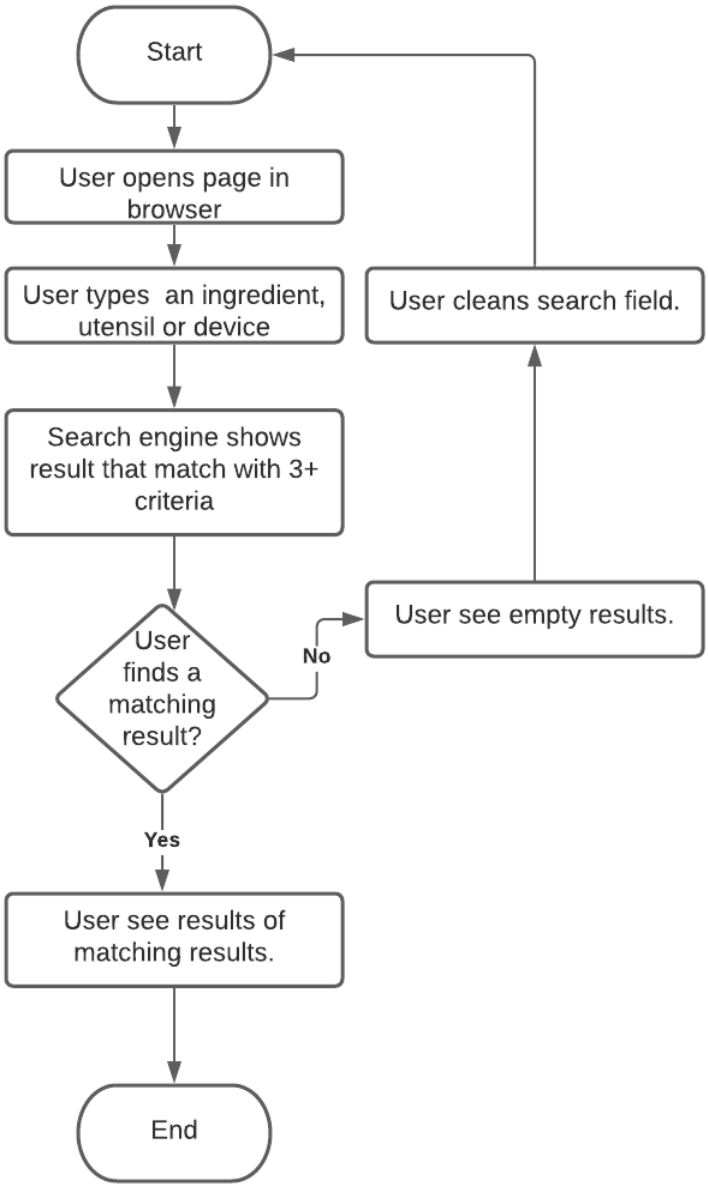


Fig. 1. Algorithm flowchart for search option one.

4.1 Flow chart of option 2

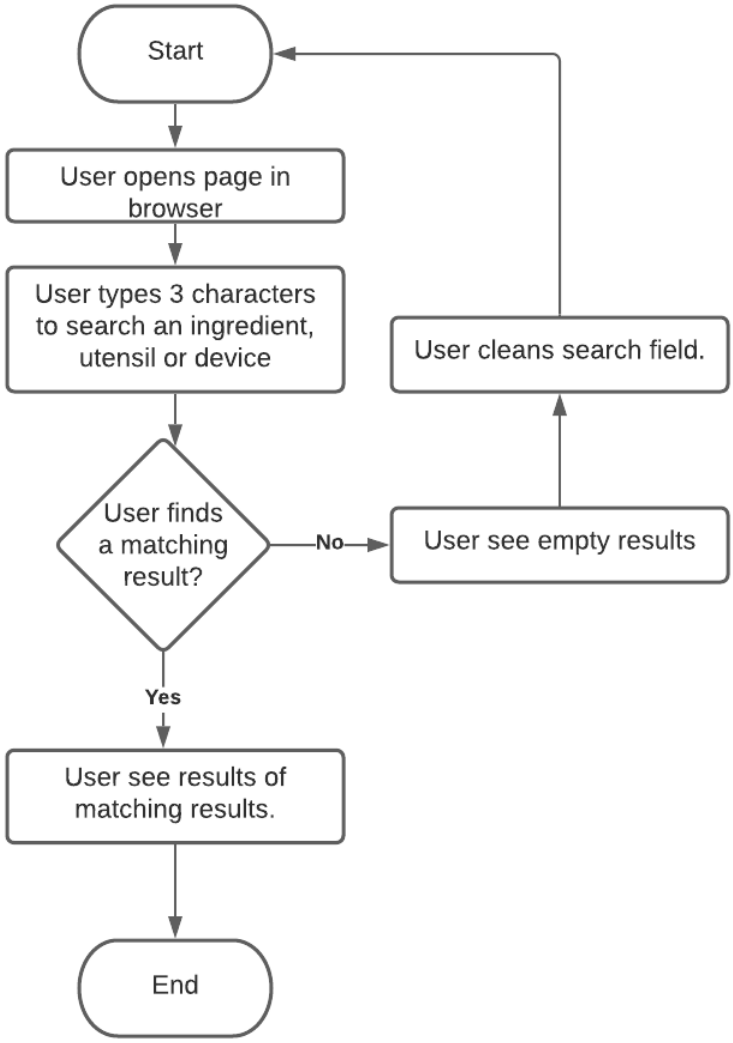


Fig. 1. Algorithm flowchart for search option two.