FICHE D'INVESTIGATION DE FONCTIONNALITÉ

Fonctionnalité : Barre de recherche principale	**Fonctionnalité #2**
Problématique : Offrir à l'utilisateur la possibilité d'effectuer une recherche de recettes via la barre de recherche principale avec des performances optimisées	

Option 1 : Algorithme utilisant uniquement des boucles natives (cf figure 1)

Dans cette option, l'algorithme de recherche génère le tableau des recettes filtrées en testant dans un premier temps chaque mot tapé par l'utilisateur dans la barre de recherche pour trouver les recettes contenant ce mot. Pour chaque mot testé, le résultat est envoyé dans un tableau intégré à l'élément *objet* de chaque recette.
Dans un second temps, une fonction est appelée pour générer le tableau des recettes dont le résultat des tests a retourné « *true* » pour tous les mots testés.

Avantages :	**Inconvénients :**
* Code clair * L'utilisateur peut effectuer une recherche en tapant plusieurs mots	* Nécessite de créer un tableau qui enregistre les résultats du test de match pour chaque mot recherché * Le test impacte les performances

Le résultat de la recherche est actualisé quand le premier mot recherché par l'utilisateur contient au **minimum 3 caractères**.
La recherche de correspondance s'effectue dans le **titre, la liste des ingrédients et la description** de chaque recette.

Option 2 : approche de l'algorithme de recherche en programmation fonctionnelle (cf figure 2)

Dans cette option, l'algorithme de recherche génère le tableau des recettes filtrées à l'aide de *array.protoype.filter()* et *array.protoype.every()*.
array.prototype.every() est utilisé dans la fonction *callback* appelée par *array.protoype.filter()* et garantit que chaque *objet* recette du tableau généré contient TOUS les mots tapés par l'utilisateur dans la barre de recherche principale.

Avantages :	**Inconvénients :**
***** Code simplifié (nécessite qu'une seule fonction, sans besoin de créer un tableau de test) ***** L'utilisateur peut effectuer une recherche en tapant plusieurs mots * La recherche est plus rapide (46 % plus rapide que l'option 1 au test sur JSBench)	***** Les performances de *array.prototype.filter()* sont généralement moins optimales qu'une boucle for ou for...of

Le résultat de la recherche est actualisé quand le premier mot recherché par l'utilisateur contient au minimum **3 caractères**.
La recherche de correspondance s'effectue dans le **titre, la liste des ingrédients et la description** de chaque recette.

Solution choisie :
Compte tenu de l'importance accordée à la performance de la fonctionnalité de recherche pour ce projet, **l'option 2** a été retenue. Les tests de performance effectués démontrent que l'algorithme en programmation fonctionnelle est 46 % plus rapide que celui de l'option 1. La recherche pour l'utilisateur est donc optimisée et le code est maintenable.

Figure 1 : Algorigramme de recherche avec boucles natives (illustration pour option 1)

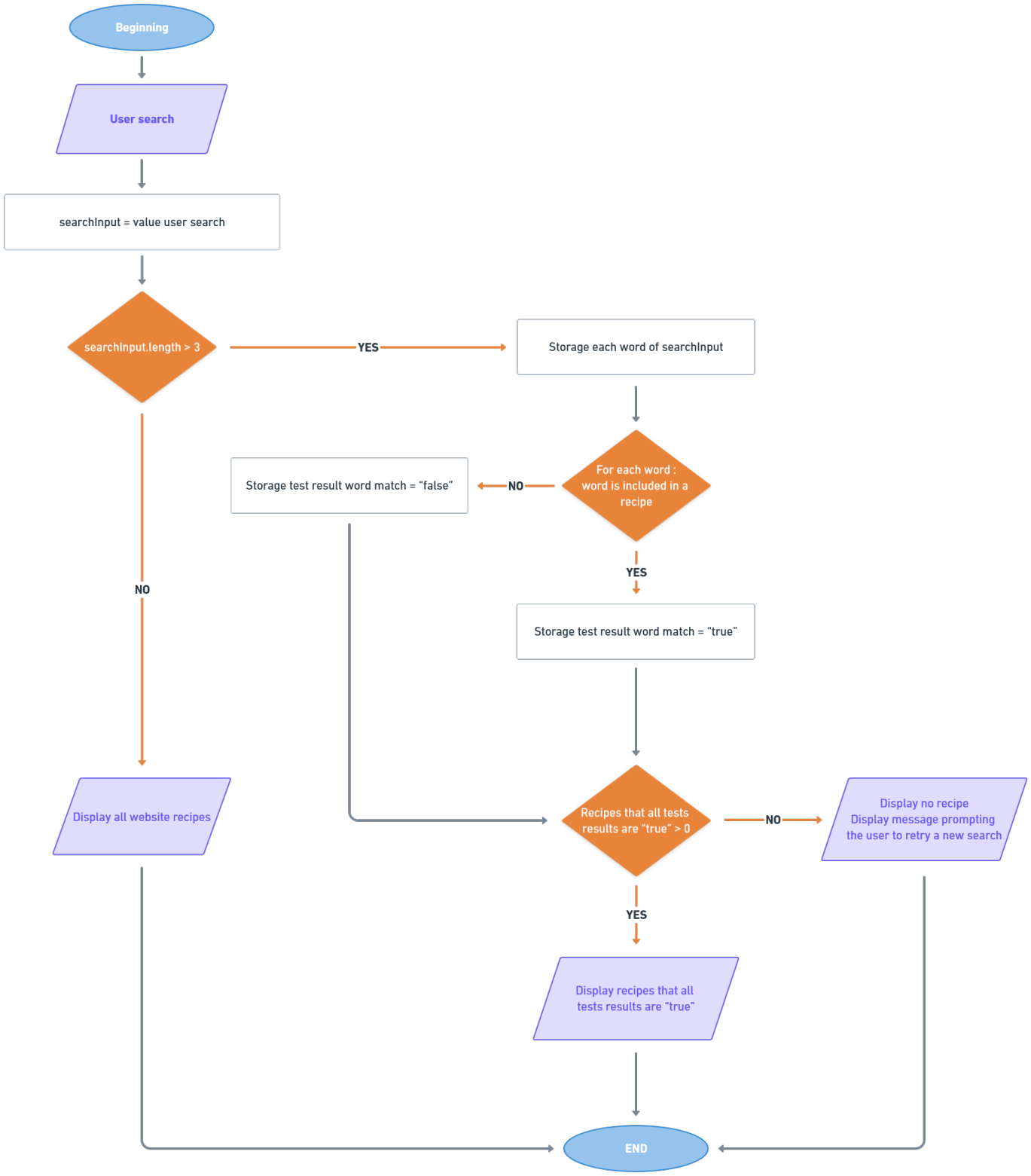

Figure 2 : Algorigramme de recherche en programmation fonctionnelle (illustration pour option 2)