Fiche d’investigation de fonctionnalité

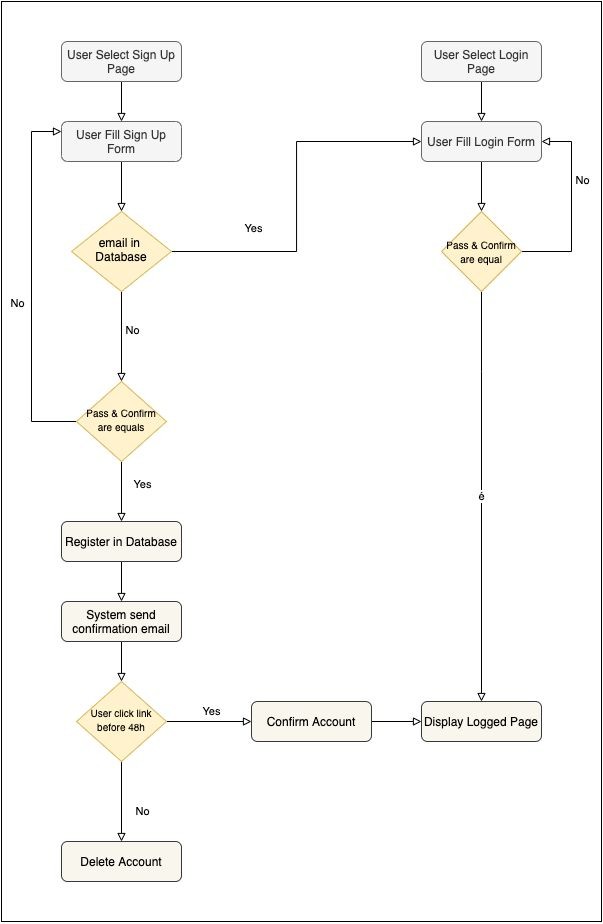
|  |  |
| --- | --- |
| **Fonctionnalité :** Filtre des recettes | **Fonctionnalité** |
| **Problématique :** La fonctionnalité de recherche (algorithme) doit être très performante, car les utilisateurs veulent une recherche la plus rapide possible, presque instantanée. | |

|  |  |
| --- | --- |
| **Option 1 : Méthode forEach avec Includes**  Ici, nous avons utilisé la méthode includes, laquelle vérifie si la saisie de l’utilisateur est incluse dans le tableau des recettes et renvoie soit vraie, dans ce cas cette dernière est ajoutée dans le nouveau filtré, soit faux et elle n’est pas ajoutée. | |
| **Avantages**  ⊕ moins de code  ⊕ syntaxe simple et pratique | **Inconvénients**  ⊖ temps d’exécution légèrement plus long |
| **Nombre de caractères minimum requis pour l’exécution de la recherche : 3** | |

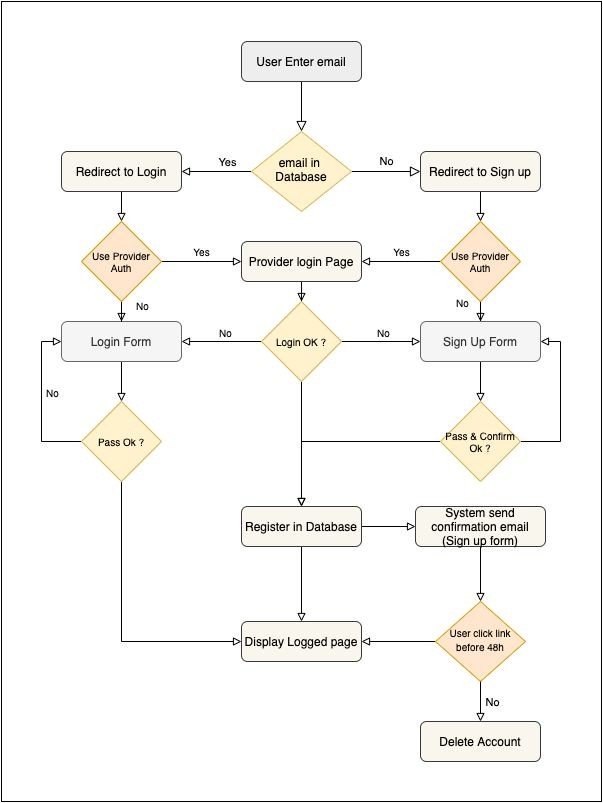
|  |  |
| --- | --- |
| **Option 2 : Boucle for of**  Dans cet algorithme, on utilise la méthode indeOf pour vérifier si la saisie de l’utilisateur existe dans le tableau des recettes afin de l’ajouter dans le nouveau tableau qui affiche les recettes filtrées. | |
| **Avantages**  ⊕ Rapidité d’exécution | **Inconvénients**  ⊖ syntaxe plus longue |
| **Nombre de caractères minimum requis pour l’exécution de la recherche : 3** | |

**Solution retenue :**

L’objectif étant d’avoir une recherche la plus rapide possible, l’algorithme 2 (celle de la boucle for native) est celle retenue car elle est la plus optimisée.

**Annexes**

# Figure 1 - Diagramme d’activité Classic login Workflow



**Figure 2 : Approche “Email First’ et usage de Google Identity Toolkit**