Système Expert de conseil en nutrition personnalisé

Réalisée Par : Khlifi Mohamed Rayen - Boukhris Ines
2ING01

PLAN

01 Présentation du sujet

02 Environnement logiciel

03 Réalisation

04 Démonstration

Introduction

- Un système expert est un type de système informatique conçu pour imiter le raisonnement humain dans un domaine spécifique. Il utilise des règles, des bases de connaissances et des algorithmes pour résoudre des problèmes ou prendre des décisions dans un domaine particulier.

- Les composants principaux d'un système expert incluent généralement :
  - Base de faits
  - Base des règles
  - Moteur d'inférence

01 Présentation du sujet

Utilité d'un système expert de conseil en nutrition personnalisée

Personnalisation des recommandations

Il peut prendre en compte les besoins spécifiques de chaque individu en fonction de son âge, de son poids, de son niveau d'activité physique, de ses objectifs de santé et de ses préférences alimentaires.

Conseils basés sur des données précises

En utilisant une base de connaissances étendue en nutrition, ce système peut fournir des conseils précis et actualisés sur les valeurs nutritionnelles, les portions recommandées, les habitudes alimentaires saines

Éducation et sensibilisation

Il peut informer les utilisateurs sur les principes de base de la nutrition, sur les types d'aliments à privilégier et sur l'impact de différentes habitudes alimentaires sur la santé.

02

Environnement Logiciel

Python est un langage de programmation populaire et polyvalent, souvent utilisé pour le développement d'applications, la création de sites web, l'analyse de données, l'intelligence artificielle.

Expera est une bibliothèque en Python utilisée pour la création de systèmes experts. Elle fournit des outils pour développer des systèmes basés sur des règles et des moteurs d'inférence, permettant ainsi de créer des applications.

Tkinter est une bibliothèque graphique intégrée à Python. Elle permet de créer des interfaces utilisateur graphiques (GUI). Avec Tkinter, les programmeurs peuvent concevoir des fenêtres, des boutons, des champs de texte et d'autres éléments d'interface graphique pour leurs applications Python.

Réalisation

Base des faits:

- Sexe
- Age
- Taille
- Poids
- Cholesterol
- Sugar
- Temperature
- Activity
- Regime

Base des règles:

- **rule_calculate_caloric_needs**: Calcul des besoins caloriques pour les femmes en fonction de l'âge, du poids, de la taille et de l'activité physique.
- **rule_recommandation_alimentaire_cholesterol**: Recommandations alimentaires en cas de taux de cholestérol élevé.
- **rule_recommandation_alimentaire_sugar**: Recommandations alimentaires en cas d'indice de glycémie instable.
- **rule_recommandation_hydratation**: Recommandations pour l'hydratation en cas de température élevée et d'activité physique intense.
- **rule_recommandation_supplements**: Recommandations pour des suppléments alimentaires en fonction des carences en vitamines (A, C, B12, Fer).

Réalisation

Captures d'écran du code

Les faits

```
from experta import *
import tkinter as tk
from tkinter import ttk

# Déclaration des faits
class Sexe(Fact):
    pass

class Age(Fact):
    pass

class Taille(Fact):
    pass

class Poids(Fact):
    pass

class Cholesterol(Fact):
    pass

class Sugar(Fact):
    pass

class Temperature(Fact):
    pass

class Activity(Fact):
    pass

class Regime(Fact):
    pass
```

Réalisation

Exemple des règles

```
# Définition des règles
class NutritionEngine(KnowledgeEngine):

    @Rule(Sexe(value=MATCH.sex),Age(int_value=MATCH.age),Taille(int_value=MATCH.taille),Poids(int_value=MATCH.poids),Activity(MATCH.activity))
    def rule_calculate_caloric_needs(self, sex, age, taille, poids, activity):
        if sex == "Female":
            m = 66.5 + (13.75 * int(poids)) + (5 * int(taille)) - (6.75 * int(age))
        elif sex == "Male":
            m = 655.1 + (9.563 * int(poids)) + (1.850 * int(taille)) - (4.676 * int(age))

        if activity == "Faible":
            w = m * 1.375
        elif activity == "Modéré":
            w = m * 1.55
        elif activity == "Intense":
            w = m * 1.725
        with open("results.txt", "a") as file:
            file.write(f"\n\n *** Vos besoins caloriques sont de {round(w, 2)} calories par jour. ***\n")
```

```
@Rule(Regime(value=MATCH.v))
def rule_recommandation_supplements(self, v):
    with open("results.txt", "a") as file:
        if v == "Vitamine A":
            file.write(" \n *** Les aliments riches en vitamine A *** \n")
            file.write(" -1- Les légumes à feuilles vertes, tels que les épinards, les blettes et le chou frisé. \n -2- Les carottes, les patates douces et les autres légumes orange ou
        elif v == "Vitamine C":
            file.write(" \n *** Les aliments riches en vitamine C *** \n")
            file.write(" -1- Les fruits frais, tels que les oranges, les citrons, les kiwis et les fraises. \n -2- Les légumes frais, tels que les poivrons, le brocoli et les choux de B
        elif v == "Vitamine B12":
            file.write(" \n *** Les aliments riches en vitamine B12 *** \n")
            file.write(" Les produits d'origine animale, tels que la viande, les fruits de mer, les œufs et les produits laitiers.\n")
        elif v == "Fer":
            file.write(" \n *** Les aliments riches en Fer *** \n")
            file.write (" -1- La viande rouge, la volaille et le poisson. \n -2- Les légumes à feuilles vertes, tels que les épinards et les blettes. \n -3- Les légumineuses, telles que
```

Réalisation

Fonction d'exécution du moteur d'inférence

```
# Declaration des attributs
Nom, sex, age, taille, poids, cholesterol, sugar, temperature, activity, regime = None, None, None, None, None, None, None, None, None, None


def execution():
    with open("results.txt", "a") as file:
        file.write("**** Bienvenue dans le système expert de conseil en nutrition personnalisée! **** \n\n ")

    global Nom
    global sex
    global age
    global cholesterol
    global sugar
    global temperature
    global activity
    global regime
    global taille
    global poids

    engine = NutritionEngine()
    engine.reset()
    engine.declare(Sexe(value=sex))
    engine.declare(Age(int_value=age))
    engine.declare(Taille(int_value=taille))
    engine.declare(Poids(int_value=poids))
    engine.declare(Cholesterol(cholesterol))
    engine.declare(Sugar(sugar))
    engine.declare(Temperature(temperature))
    engine.declare(Activity(activity))
    engine.declare(Regime(value=regime))

    # Exécution du moteur d'inférence
    engine.run()
```

Exemple de code pour l'interface graphique

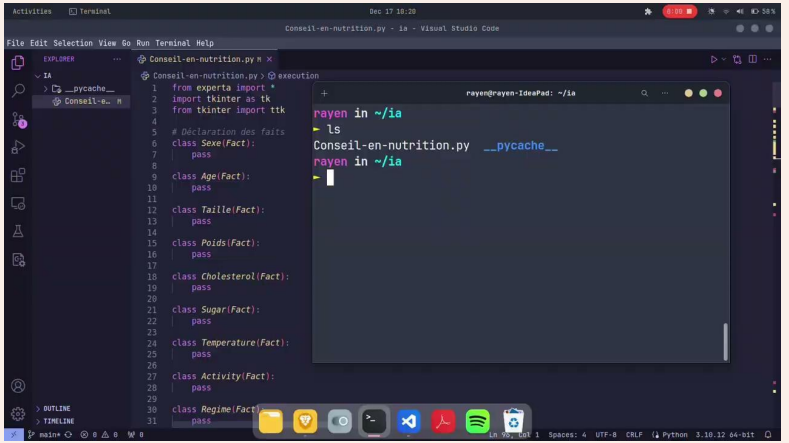
```
def start_GUI():

    def submit():

        global Nom
        global sex
        global age
        global cholesterol
        global sugar
        global temperature
        global activity
        global regime
        global taille
        global poids

        Nom = entry_name.get()
        sex = sex_combo.get()
        age = entry_age.get()
        cholesterol = chol_combo.get()
        sugar = sugar_combo.get()
        temperature = temp_combo.get()
        activity = activity_combo.get()
        regime = vitamins_combo.get()
        taille = entry_taille.get()
        poids = entry_poids.get()

        # Create a label inside the frame
        label_tkinter = tk.Label(
            window,
            text="**** Que ton aliment soit ta seule médecine ****:",
            bg='#d5e6e6',
            fg="#374151",
            font=("Arial", 12, "bold")
        )
        label_tkinter.grid(row=12, column=0, sticky="nsew", padx=(0, 5))

        return True


    # Create the main window
    window = tk.Tk()
    window.title("Système expert de conseil en nutrition personnalisée")
```

```
# Create an entry widget (text box) inside the taille_frame
entry_taille = tk.Entry(window, font=("Arial", 12))
entry_taille.insert(0, "Taille en cm.")
entry_taille.configure(fg='gray')
entry_taille.grid(row=3, column=1, sticky="nsew")

# Create a frame to hold the label and entry
age_frame = tk.Frame(window)
age_frame.grid(row=4, column=0, padx=10, pady=10)

# Create a label inside the frame
label_age = tk.Label(
    age_frame,
    text="Age:",
    fg="#374151",
    font=("Arial", 12, "bold")
)
label_age.grid(row=4, column=0, sticky="nsew", padx=(0, 5))

# Create an entry widget (text box) inside the age_frame
entry_age = tk.Entry(window, font=("Arial", 12))
entry_age.grid(row=4, column=1, sticky="nsew")

# Create a frame to hold the label and entry
poids_frame = tk.Frame(window)
poids_frame.grid(row=5, column=0, padx=10, pady=10)
```

04 Démenstartion

1 er Cas de test :

2eme Cas de test :



➔ Résultat de l'exécution est sauvegardé dans un fichier results.txt enregistré dans la même répertoire que le fichier conseil-en-nutrition.py