Name :Akhilesh

Rollno :232010037

Implementation of Medical Diagnostic system using Dynamic Database in Prolog

Code :

**% -------------------------------**

**%** Medical Diagnostic System in Prolog

**% -------------------------------**

**%** Enable dynamic predicates to allow runtime modifications

:**-** dynamic symptom**/**2.

:**-** dynamic disease**/**1.

**% -------------------------------**

**%** Knowledge Base: Diseases **and** Symptoms

**% -------------------------------**

**%** Define diseases disease(flu). disease(malaria). disease(migraine).

**%** Define symptoms associated with each disease

**%** symptom(Symptom, Disease).

**%** Symptoms *for* Flu symptom(fever, flu). symptom(cough, flu). symptom(headache, flu). symptom(sore\_throat, flu).

**%** Symptoms *for* Malaria symptom(fever, malaria). symptom(chills, malaria). symptom(sweating, malaria). symptom(nausea, malaria).

**%** Symptoms *for* Migraine symptom(headache, migraine). symptom(nausea, migraine). symptom(visual\_disturbance, migraine). symptom(sensitivity\_to\_light, migraine).

**% -------------------------------**

**%** Dynamic Update Predicates

**% -------------------------------**

**%** Add a **new** symptom to a disease add\_symptom(Symptom, Disease) :**-**

( disease(Disease) **->**

( \**+** symptom(Symptom, Disease) **->**

assertz(symptom(Symptom, Disease)),

format('Symptom "~w" added to disease "~w".~n', [Symptom,

Disease])

;

format('Symptom "~w" already exists for disease "~w".~n',

[Symptom, Disease])

)

;

format('Disease "~w" does not exist. Please add the disease first.~n',

[Disease])

).

% Remove a symptom from a disease remove\_symptom(Symptom, Disease) :-

( symptom(Symptom, Disease) ->

retract(symptom(Symptom, Disease)),

format('Symptom "~w" removed from disease "~w".~n', [Symptom, Disease])

***;***

format('Symptom "~w" is not associated with disease "~w".~n', [Symptom, Disease])

***).***

% Add a new disease add\_disease(Disease) :-

( disease(Disease) ->

format('Disease "~w" already exists.~n', [Disease])

***;***

assertz(disease(Disease)),

format('Disease "~w" has been added to the knowledge base.~n',

[Disease])

***).***

% Remove a disease and its associated symptoms remove\_disease(Disease) :-

( disease(Disease) -> retractall(symptom(\_, Disease)), retract(disease(Disease)),

format('Disease "~w" and its symptoms have been removed from the knowledge base.~n', [Disease])

***;***

format('Disease "~w" does not exist in the knowledge base.~n', [Disease])

***).***

***%***

% Diagnosis Predicate

***%***

% Diagnose diseases based on a list of symptoms

% Usage Example: diagnose([fever, cough, sore\_throat], Diseases). diagnose(Symptoms, PossibleDiseases) :-

findall(Disease, ( disease(Disease),

subset\_symptoms(Disease, Symptoms)

), Diseases),

sort(Diseases, PossibleDiseases).

% Helper predicate to check if all symptoms of a disease are present in the provided symptoms

subset\_symptoms(Disease, Symptoms) :-

\+ ( symptom(Symptom, Disease), \+ member(Symptom, Symptoms) ).

***%***

% User Interaction Predicates

***%***

% Interactive diagnosis: Prompts user to enter symptoms and returns possible diseases

% Note: SWISH may have limitations with interactive prompts. It's recommended to provide symptoms as a list.

interactive\_diagnose :-

write('Enter your symptoms one by one. Type "done." when finished.'), nl, read\_symptom\_list(SymptomList),

diagnose(SymptomList, PossibleDiseases), ( PossibleDiseases \= [] ->

format('Based on the symptoms, you might have the following disease(s): ~w~n', [PossibleDiseases])

***;***

write('No diseases matched the provided symptoms.'), nl

***).***

% Recursive predicate to read symptoms from the user read\_symptom\_list(Symptoms) :-

write('Enter symptom (or type done. to finish): '), read(Input),

( Input == done -> Symptoms = []

***;***

read\_symptom\_list(Rest), Symptoms = [Input | Rest]

***).***

Output :



