NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE

CZ3005: Artificial Intelligence

2020 Fall Assignment 4:
Implementing a Talking Box with Prolog

Question 4:
Patient with a Sympathetic Doctor

Name: Royce Ang Jia Jie
Matriculation Number: U1840416D
Tutorial Group Number: TSP2

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2020/2021 SEMESTER1

Table of Contents

Section 1. Introduction 3
Section 2. Overview 3
2.1 Introduction to KBS 3
2.2 Overview of KBS 4
2.3 Logic flow of Sympathetic Doctor Program 5
.... 5
2.4 Further Considerations 6
2.4.1 Pain Level and its effect on the biasness of diagnosis 6
2.4.2 Pain and Mood Level & their effect on the degree of sympathy 6
Section 3. Implementation and Explanation 7
3.1 Code Explanation 7
3.2 Logic Flow of Main Program 10
3.3 Logic Flow for Diagnosis 12
3.4 Heuristics/ Algorithm for conflicting diagnosis 14
3.5 Knowledge Database 14
3.6 Helper Functions 15
3.7 Further Considerations (Error Handling) 15
Section 4. Demonstration of Sympathetic Doctor Dialogue AI 16
Section 5: Additional Implementation (GUI) with Google Speech Recognition Engine 17
5.1 Motivation for GUI Implementation (Additional Feature) 17
5.2 Overall System Architecture Diagram of the GUI Implementation 17
5.3 Set up and Installation 18
5.4 Implementation of speech recognition on GUI 18
5.5 Screenshot of GUI Implementation 19
5.6 Demonstration of Sympathetic Doctor Dialogue AI with GUI (YouTube Link) 19
Works Cited 20

Section 1. Introduction

Prolog is a logic programming language associated with Artificial Intelligence and Computational Linguistics.

It is interesting to note that unlike other programming languages, Prolog has its roots in first-order logic, a formal logic, used to represent a powerful Knowledge-Based System (KBS). As such, logic programmers can ascribe the efficiency of the knowledge-based representation to its generic representation without having to deal with technical concepts (time, events). Additionally, it is a universal language, allowing us to express anything that can be programmed.

In this assignment, we will:

(1) Implement a *Knowledge Base System Dialogue AI* to explore the concept of separation of *Knowledge* (essential representation of the world) from the *Inference Engine* (Set of inference rules) in KBS using Prolog.

(2) As part of an additional feature, we will be implementing a *GUI* to simplify and have an active interaction with the KBS, while observing how the KBS does the update in the background.

Section 2. Overview

2.1 Introduction to KBS

The KBS is designed with the following considerations:

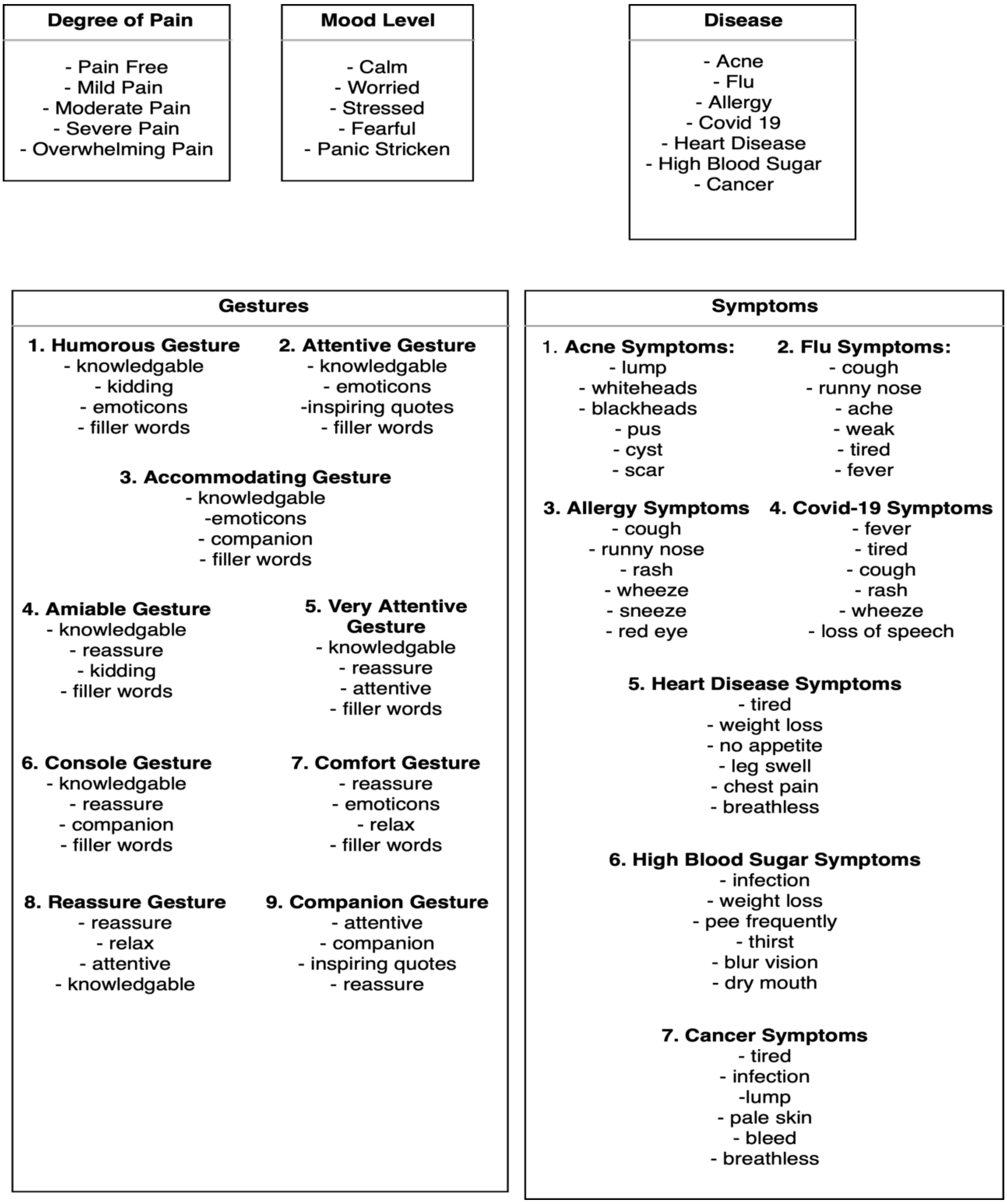
1. The KBS must be able to ask user(patient) questions.
2. The user(patient) must be able to reply “yes” or “no” only.
3. The KBS must be able to ask the user(patient) 5 or more degree of pain level.
4. The KBS must be able to ask the user(patient) 5 or more levels of mood.
5. The KBS must be able to ask questions in an appropriate demeanour based on the user’s(patient) pain and mood level.
6. The KBS must have a total of 5 diseases.
7. For each disease, there must be at least 5 or more symptoms associated to it.
8. The KBS must be able to diagnose the patient’s disease.

Some additional considerations during the design of the KBS:

1. When a user(patient) replies a “yes” to any question, KBS must be able to *assert* the given answer as true.
2. The KBS is assumed to be a *thorough* AI. In which, the KBS should be able to do a *comprehensive analysis* of various symptoms should be done before making a diagnosis.
   a. KBS should be able to iterate through all symptoms.
   b. A good heuristic or algorithm should be implemented for diseases with overlapping symptoms.
3. The KBS must be able to answer the user(patient) in a sympathetic manner.
   a. A combination of pain level and mood level should output a type of gesture.
   b. Gesture chosen should be able to distinctively show an appropriate level of sympathy.

2.2 Overview of KBS

To be able to separate the knowledge base from the inference engine, the KBS has the following 5 distinct "Databases". To ensure that the symptoms of each disease/illness is as *realistic* as possible, I have referenced them from *NHS disease database* [1]. Note that I have implemented a total of 7 diseases – each having 6 symptoms. Diseases with overlapping symptoms and their algorithm to resolve the diagnosis issue will be explained in *section **2.4*** and ***3.4***.



Degree of Pain

- Pain Free
- Mild Pain
- Moderate Pain
- Severe Pain
- Overwhelming Pain

Mood Level

- Calm
- Worried
- Stressed
- Fearful
- Panic Stricken

Disease

- Acne
- Flu
- Allergy
- Covid 19
- Heart Disease
- High Blood Sugar
- Cancer

Gestures

1. Humorous Gesture
- knowledgable
- kidding
- emoticons
- filler words

2. Attentive Gesture
- knowledgable
- emoticons
-inspiring quotes
- filler words

3. Accommodating Gesture
- knowledgable
-emoticons
- companion
- filler words

4. Amiable Gesture
- knowledgable
- reassure
- kidding
- filler words

5. Very Attentive Gesture
- knowledgable
- reassure
- attentive
- filler words

6. Console Gesture
- knowledgable
- reassure
- companion
- filler words

7. Comfort Gesture
- reassure
- emoticons
- relax
- filler words

8. Reassure Gesture
- reassure
- relax
- attentive
- knowledgable

9. Companion Gesture
- attentive
- companion
- inspiring quotes
- reassure

Symptoms

1. **Acne Symptoms:**
- lump
- whiteheads
- blackheads
- pus
- cyst
- scar

2. Flu Symptoms:
- cough
- runny nose
- ache
- weak
- tired
- fever

3. Allergy Symptoms
- cough
- runny nose
- rash
- wheeze
- sneeze
- red eye

4. Covid-19 Symptoms
- fever
- tired
- cough
- rash
- wheeze
- loss of speech

5. Heart Disease Symptoms
- tired
- weight loss
- no appetite
- leg swell
- chest pain
- breathless

6. High Blood Sugar Symptoms
- infection
- weight loss
- pee frequently
- thirst
- blur vision
- dry mouth

7. Cancer Symptoms
- tired
- infection
-lump
- pale skin
- bleed
- breathless

Figure 1 5 Distinct "Databases"

2.3 Logic flow of Sympathetic Doctor Program

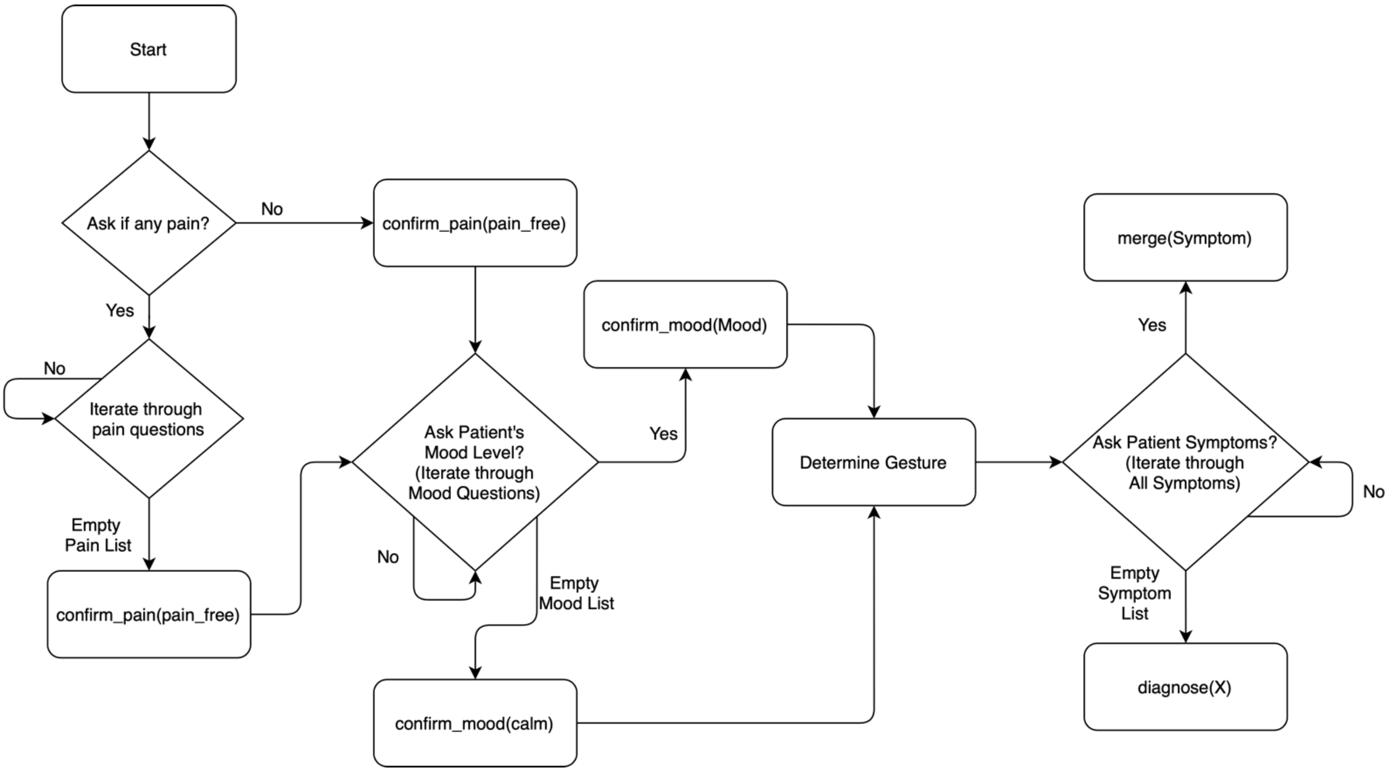


Figure 2 Sympathetic Doctor Logic Flow

Firstly, the Doctor will ask the patient if he/she is feeling any pain?

e.g. Doctor : “Hello! I am Doctor Box. Do you feel any pain?”

If the patient replies a “*No*”, his reply would implicitly assert that the patient is not having pain or “***pain free***”.

Otherwise, if the patient replies a “*Yes*”, this would initiate the iteration of the remaining list of 4 pain questions, until it is empty.

e.g. Doctor: “*Do you feel* ***mild pain***” (*no*)→ “ *Do you feel* ***moderate pain***” (*no*)→
Doctor: “*Do you feel* ***severe pain***” (*no*)→ “*Do you feel* ***overwhelmingly severe pain***” (*no*)→ {*Empty List*}

Edge cases such as having an empty list is handled by implicitly asserting that the patient is not having pain or “***pain free***”.

The same procedure is repeated for asking *Patient’s Mood Level* (*Calm, Worried, Stressed, Fearful and Panic Stricken*).

The combination of pain level and mood level would determine how the Doctor should respond to the patient from one of the 9 varying types of gestures (*Humorous, Attentive, Accommodating, Amiable, Very Attentive, Console, Comfort, Reassure and Companion*).

e.g. Overwhelming Pain + Panic Stricken → Companion Gesture

Doctor: “***Let us go through this together*** (*Gesture*). Do you have a cough? (*Symptom*)”

2.4 Further Considerations

2.4.1 Pain Level and its effect on the biasness of diagnosis

To simulate a real-life situation, where a doctor would use pain level to bias towards a specific type of diagnosis.

An additional feature is added such that pain level would also induce the biasness of diagnosis. This is implemented by **adding** *additional weights* (*heuristic*) to *diseases* associated to specific pain level; thereby, giving these disease(s) a heard start.

Note: This will be explained in the *implementation and explanation section*.

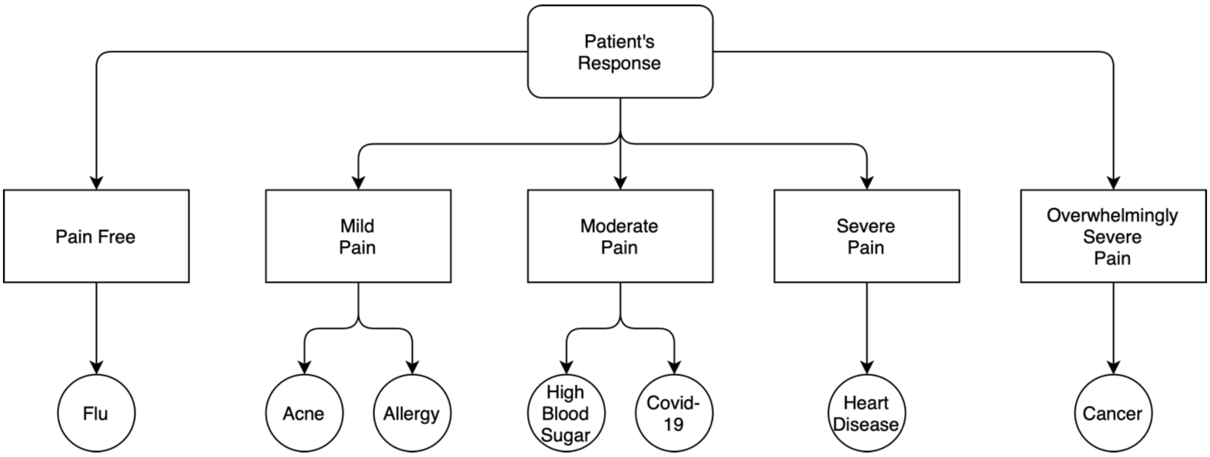


Figure 3 Weighted Disease(s) Associated to specific Pain Level

2.4.2 Pain and Mood Level & their effect on the degree of sympathy

Another challenge faced was the designing of gestures. A great emphasis is placed on good patient-physician communication skill to ensure the authenticity of the dialogue with the Doctor [2]. This is done by combining both non-verbal (e.g. *emoticons*) and verbal cues (e.g. *filler words*) for a single gesture.

e.g. *gesture(humorous) :- knowledgeable(A), emoticons(B), kidding(C), filler words(D)*

Note the mood level in the following order:

Calm < ***Worried*** < ***Stressed*** < ***Fearful*** < ***Panic Stricken***

The degree of seriousness in gestures are as follow:

Humorous < ***Amiable*** < ***Attentive*** < ***Very Attentive*** < ***Accommodating*** < ***Console*** < ***Comfort*** < ***Reassure*** < ***Companion***

Additionally, because of the extreme case, where a patient may experience (1) ***Overwhelmingly Severe Pain*** and (2) ***Calm Mood***, it would be inappropriate to have a gesture of ***Humorous***, but a more (in-between) serious degree of gesture should be picked instead, for instance, the gesture of ***Comfort***.

Therefore, when a patient experiences (1) ***Overwhelmingly Severe Pain*** *and (2)* ***Panic Stricken*** → ***Companion*** *Gesture* should be expected (since the degree of seriousness in the list of gestures is the highest), thereby, providing justified comfort to the patient.

Section 3. Implementation and Explanation

3.1 Code Explanation

Predicates/Variables used: (*More details in source code*)

1. pain_questions(L). – Provides a list of question for the patient to access level of pain.
2. knowledgable(L). – Provides a list of knowledgeable gestures a doctor would say.
3. emoticons(L). – Provides a list of non-verbal cues/gestures for a doctor to say.
4. kidding(L). – Provides a list of joking gesture for the doctor to say.
5. filler_words(L). – Provides a list of thinking gestures for the doctor to say.
6. companion(L). – Provides a list of hearty gestures for the doctor to say.
7. relax(L). – Provides a list of reassuring gestures for the doctor to say.
8. inspiring_quote(L). – Provides a list of inspiring gestures for the doctor to say.
9. attentive(L). – Provides a list of attentive gestures for the doctor to say.
10. count_acne(X). – “Heuristic” value for acne.
11. count_flu(X). – “Heuristic” value for flu.
12. count_allergy(X). – “Heuristic” value for allergy.
13. count_covid_19(X). – “Heuristic” value for covid-19.
14. count_heart_disease(X). – “Heuristic” value for heart disease.
15. count_high_blood_sugar(X). – “Heuristic” value for high blood sugar.
16. count_cancer(X). – “Heuristic” value for cancer.

Rules that govern the mood of the patient:

1. confirm_mood(calm) :- assert(mood(calm)).
2. confirm_mood(worried) :- assert(mood(worried)).
3. confirm_mood(stressed) :- assert(mood(stressed)).
4. confirm_mood(fearful) :- assert(mood(fearful)).
5. confirm_mood(panic_stricken) :- assert(mood(panic_stricken)).

Rules that govern all gestures the doctor can have determined by the <u>patient’s pain level</u> and <u>mood level.</u>

1. all_reactions(L) :- pain(pain_free),mood(calm), assertz(gesture(humorous)), all_gesture(L).
2. all_reactions(L) :- pain(mild_pain),mood(calm), assertz(gesture(humorous)), all_gesture(L).
3. all_reactions(L) :- pain(moderate_pain),mood(calm), assertz(gesture(humorous)), all_gesture(L).
4. all_reactions(L) :- pain(severe_pain),mood(calm), assertz(gesture(attentive)), all_gesture(L).
5. all_reactions(L) :- pain(overwhelming_pain),mood(calm), assertz(gesture(accommodating)), all_gesture(L).
6. all_reactions(L) :- pain(pain_free),mood(worried), assertz(gesture(amiable)), all_gesture(L).
7. all_reactions(L) :- pain(mild_pain),mood(worried), assertz(gesture(amiable)), all_gesture(L).

8. all_reactions(L) :- pain(moderate_pain),mood(worried), assertz(gesture(amiable)), all_gesture(L).
9. all_reactions(L) :- pain(severe_pain),mood(worried), assertz(gesture(very_attentive)), all_gesture(L).
10. all_reactions(L) :- pain(overwhelming_pain),mood(worried), assertz(gesture(console)), all_gesture(L).
11. all_reactions(L) :- pain(pain_free),mood(stressed), assertz(gesture(comfort)), all_gesture(L).
12. all_reactions(L) :- pain(mild_pain),mood(stressed), assertz(gesture(comfort)), all_gesture(L).
13. all_reactions(L) :- pain(moderate_pain),mood(stressed), assertz(gesture(comfort)), all_gesture(L).
14. all_reactions(L) :- pain(severe_pain),mood(stressed), assertz(gesture(comfort)), all_gesture(L).
15. all_reactions(L) :- pain(overwhelming_pain),mood(stressed), assertz(gesture(comfort)), all_gesture(L).
16. all_reactions(L) :- pain(pain_free),mood(fearful), assertz(gesture(reassure)), all_gesture(L).
17. all_reactions(L) :- pain(mild_pain),mood(fearful), assertz(gesture(reassure)), all_gesture(L).
18. all_reactions(L) :- pain(moderate_pain),mood(fearful), assertz(gesture(reassure)), all_gesture(L).
19. all_reactions(L) :- pain(severe_pain),mood(fearful), assertz(gesture(reassure)), all_gesture(L).
20. all_reactions(L) :- pain(overwhelming_pain),mood(fearful), assertz(gesture(reassure)), all_gesture(L).
21. all_reactions(L) :- pain(pain_free),mood(panic_stricken), assertz(gesture(companion)), all_gesture(L).
22. all_reactions(L) :- pain(mild_pain),mood(panic_stricken), assertz(gesture(companion)), all_gesture(L).
23. all_reactions(L) :- pain(moderate_pain),mood(panic_stricken), assertz(gesture(companion)), all_gesture(L).
24. all_reactions(L) :- pain(severe_pain),mood(panic_stricken), assertz(gesture(companion)), all_gesture(L).
25. all_reactions(L) :- pain(overwhelming_pain),mood(panic_stricken), assertz(gesture(companion)), all_gesture(L).

Rules that govern "heuristic" to resolve the issue of diseases with overlapping symptoms.

For instance, rule 1 is an overlapping symptom with acne and cancer, therefore, the increment weight for both diseases is 1. However, in rule 2, because "whiteheads" is a strong indicator of acne, the weight increment for acne is 2.

1. incr_lump :- (add_count_acne, add_count_cancer).
2. incr_whiteheads :- add_count_acne, add_count_acne.
3. incr_blackheads :- add_count_acne, add_count_acne.
4. *view more in the source code*

Rules that diagnose the patient based on the "heuristic". Highest value will serve as the diagnosis.

1. diagnose(acne) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
   A >= B, A >= C, A >= D, A >= E, A >= F, A >= G.

2. diagnose(flu) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
   B >= A, B >= C, B >= D, B >= E, B >= F, B >= G.

3. diagnose(allergy) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
   C >= B, C >= A, C >= D, C >= E, C >= F, C >= G.

4. diagnose(covid_19) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
   D >= B, D >= C, D >= A, D >= E, D >= F, D >= G.

5. diagnose(heart_disease) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
   E >= B, E >= C, E >= D, E >= A, E >= F, E >= G.

6. diagnose(high_blood_sugar) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
   F >= B, F >= C, F >= D, F >= E, F >= A, F >= G.

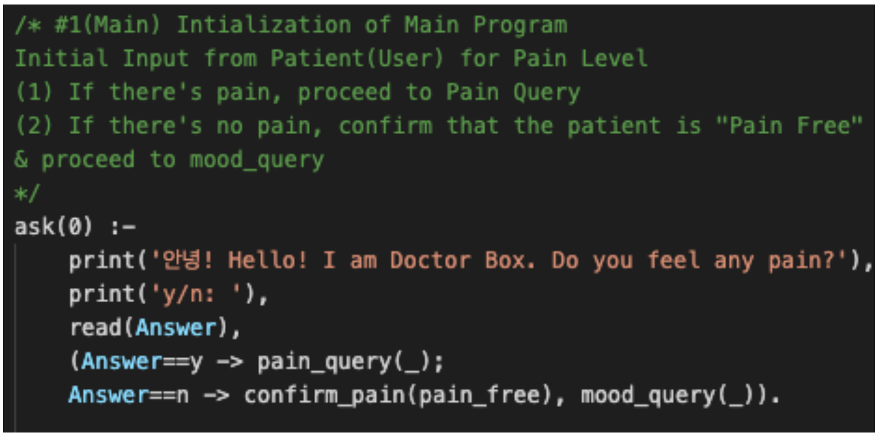
7. diagnose(cancer) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
   G >= B, G >= C, G >= D, G >= E, G >= F, G >= A.

Predicates that asserts symptom/pain and increments the appropriate counter.

1. merge(lump) :- (assert(has(lump)), incr_lump).
2. merge(whiteheads) :- (assert(has(whiteheads)), incr_whiteheads).
3. merge(blackheads) :- (assert(has(blackheads)), incr_blackheads).
4. **view more in source code under <u>merge</u>**
5. confirm_pain(mild_pain) :- assert(pain(mild_pain)), add_count_acne, add_count_allergy.
6. confirm_pain(moderate_pain) :- assert(pain(moderate_pain)), add_count_high_blood_sugar, add_count_covid_19.
7. **view more in source code under <u>confirm_pain</u>**

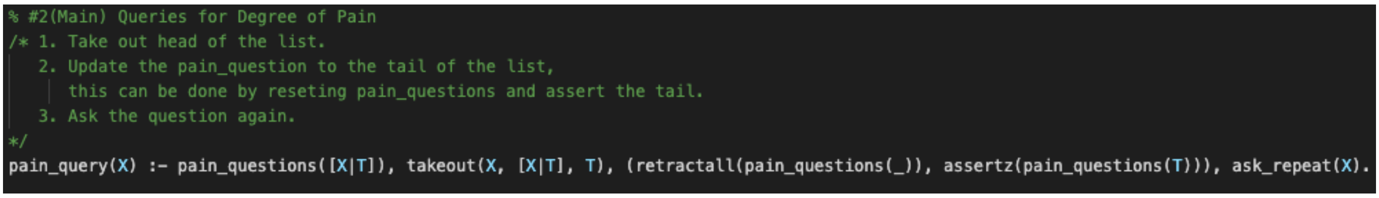
3.2 Logic Flow of Main Program

Step 1: Initialization of Main Program via *ask(0).*



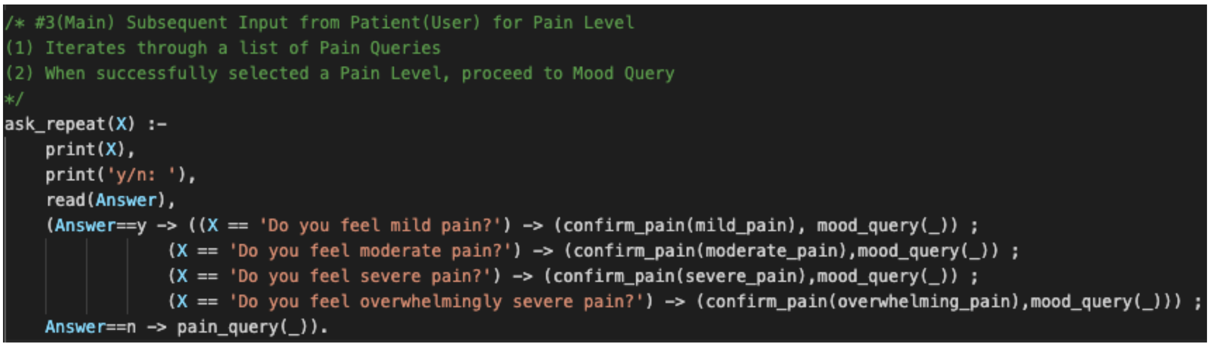
```
/* #1(Main) Intialization of Main Program
Initial Input from Patient(User) for Pain Level
(1) If there's pain, proceed to Pain Query
(2) If there's no pain, confirm that the patient is "Pain Free"
& proceed to mood_query
*/
ask(0) :-
    print('안녕! Hello! I am Doctor Box. Do you feel any pain?'),
    print('y/n: '),
    read(Answer),
    (Answer==y -> pain_query(_);
    Answer==n -> confirm_pain(pain_free), mood_query(_)).
```

Step 2: Selects pain question from pain question '*Database*' and return back to *ask_repeat* program.



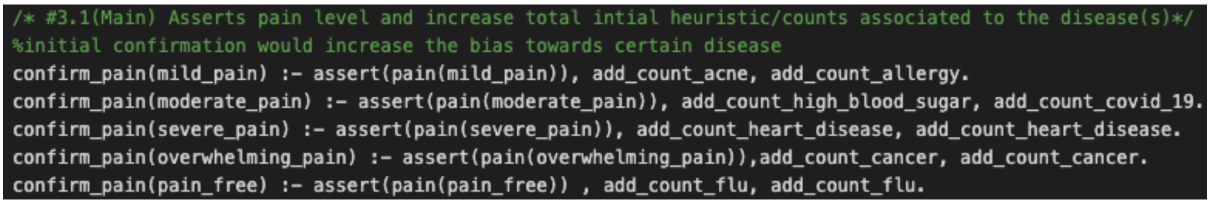
```
% #2(Main) Queries for Degree of Pain
/* 1. Take out head of the list.
   2. Update the pain_question to the tail of the list,
      this can be done by reseting pain_questions and assert the tail.
   3. Ask the question again.
*/
pain_query(X) :- pain_questions([X|T]), takeout(X, [X|T], T), (retractall(pain_questions(_)), assertz(pain_questions(T))), ask_repeat(X).
```

Step 3: When pain level has been selected, proceed to (1) *confirm_pain(pain)* level & (2) *query mood.*



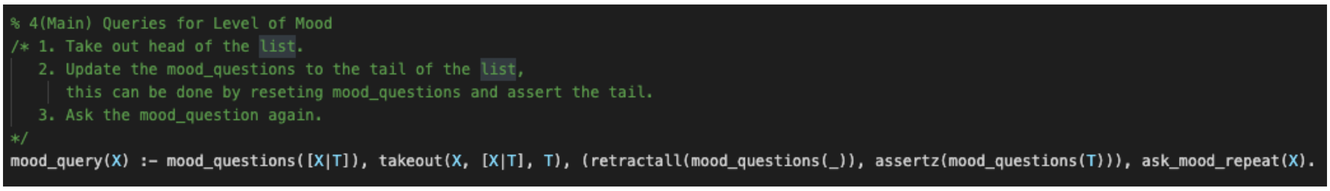
```
/* #3(Main) Subsequent Input from Patient(User) for Pain Level
(1) Iterates through a list of Pain Queries
(2) When successfully selected a Pain Level, proceed to Mood Query
*/
ask_repeat(X) :-
    print(X),
    print('y/n: '),
    read(Answer),
    (Answer==y -> ((X == 'Do you feel mild pain?') -> (confirm_pain(mild_pain), mood_query(_)) ;
                  (X == 'Do you feel moderate pain?') -> (confirm_pain(moderate_pain),mood_query(_)) ;
                  (X == 'Do you feel severe pain?') -> (confirm_pain(severe_pain),mood_query(_)) ;
                  (X == 'Do you feel overwhelmingly severe pain?') -> (confirm_pain(overwhelming_pain),mood_query(_))) ;
    Answer==n -> pain_query(_)).
```

Step 3.1: Confirm pain level by (1) *asserting the pain level* and (2) *increasing the total initial heuristics/counts* towards the associated disease(s). As explained previously in ***2.4.1*** , this is an additional consideration/feature to provide a head start for some diseases due to the extra initial weight provided, thereby, providing a more realistic approach towards diagnosing the disease.



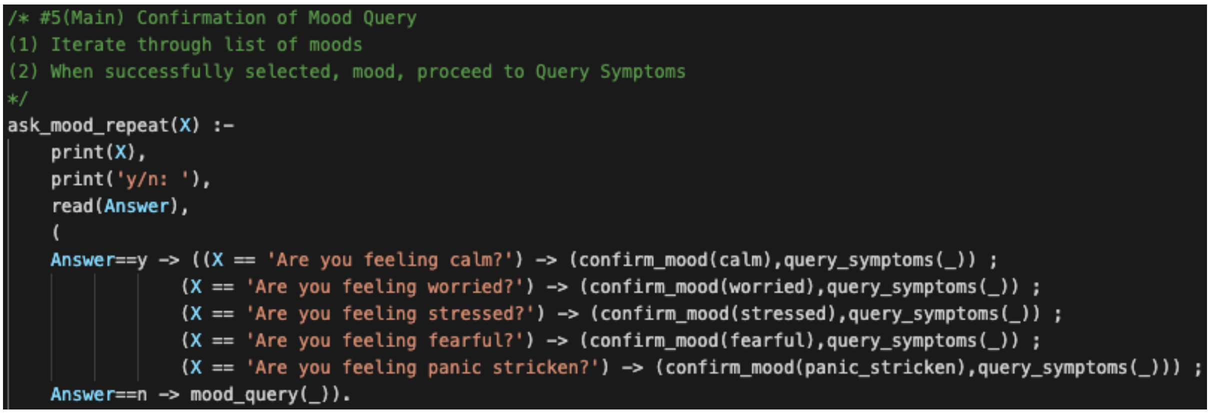
```
/* #3.1(Main) Asserts pain level and increase total intial heuristic/counts associated to the disease(s)*/
%initial confirmation would increase the bias towards certain disease
confirm_pain(mild_pain) :- assert(pain(mild_pain)), add_count_acne, add_count_allergy.
confirm_pain(moderate_pain) :- assert(pain(moderate_pain)), add_count_high_blood_sugar, add_count_covid_19.
confirm_pain(severe_pain) :- assert(pain(severe_pain)), add_count_heart_disease, add_count_heart_disease.
confirm_pain(overwhelming_pain) :- assert(pain(overwhelming_pain)),add_count_cancer, add_count_cancer.
confirm_pain(pain_free) :- assert(pain(pain_free)) , add_count_flu, add_count_flu.
```

Step 4: Selects mood question from the list of mood questions '*Database*' and return back to *ask_mood_repeat* program.



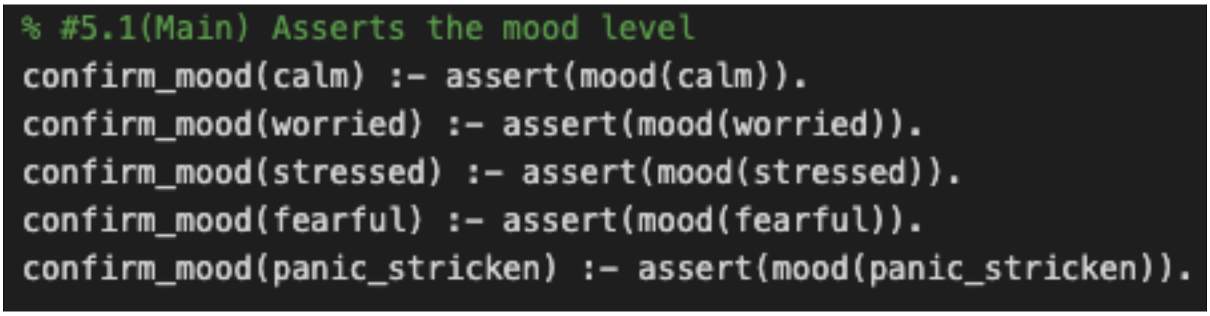
```
% 4(Main) Queries for Level of Mood
/* 1. Take out head of the list.
   2. Update the mood_questions to the tail of the list,
      this can be done by reseting mood_questions and assert the tail.
   3. Ask the mood_question again.
*/
mood_query(X) :- mood_questions([X|T]), takeout(X, [X|T], T), (retractall(mood_questions(_)), assertz(mood_questions(T))), ask_mood_repeat(X).
```

Step 5: When mood level has been selected, proceed to (1) *confirm_mood(mood)* and (2) *query_symptoms(X)* program.



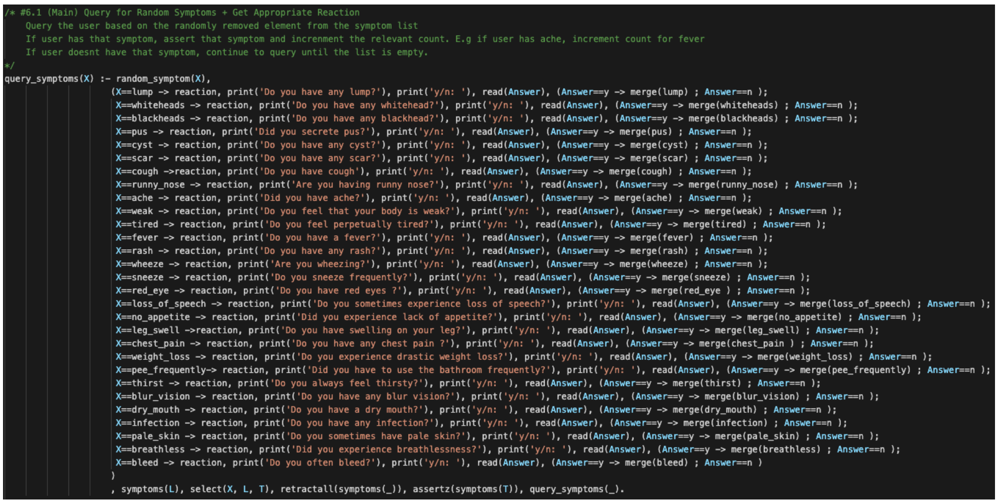
```
/* #5(Main) Confirmation of Mood Query
(1) Iterate through list of moods
(2) When successfully selected, mood, proceed to Query Symptoms
*/
ask_mood_repeat(X) :-
    print(X),
    print('y/n: '),
    read(Answer),
    (
    Answer==y -> ((X == 'Are you feeling calm?') -> (confirm_mood(calm),query_symptoms(_)) ;
                (X == 'Are you feeling worried?') -> (confirm_mood(worried),query_symptoms(_)) ;
                (X == 'Are you feeling stressed?') -> (confirm_mood(stressed),query_symptoms(_)) ;
                (X == 'Are you feeling fearful?') -> (confirm_mood(fearful),query_symptoms(_)) ;
                (X == 'Are you feeling panic stricken?') -> (confirm_mood(panic_stricken),query_symptoms(_))) ;
    Answer==n -> mood_query(_)).
```

Step 5.1: Confirms the mood by asserting the mood level provided by the patient.



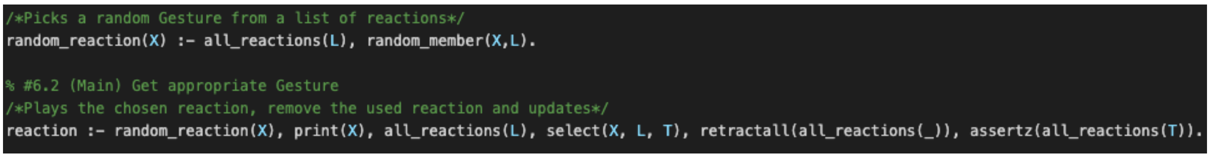
```
% #5.1(Main) Asserts the mood level
confirm_mood(calm) :- assert(mood(calm)).
confirm_mood(worried) :- assert(mood(worried)).
confirm_mood(stressed) :- assert(mood(stressed)).
confirm_mood(fearful) :- assert(mood(fearful)).
confirm_mood(panic_stricken) :- assert(mood(panic_stricken)).
```

Step 6.1: Query for a random symptom and with the appropriate gesture from the Doctor, ask the patient sympathetically.



```
/* #6.1 (Main) Query for Random Symptoms + Get Appropriate Reaction
    Query the user based on the randomly removed element from the symptom list
    If user has that symptom, assert that symptom and increnment the relevant count. E.g if user has ache, increment count for fever
    If user doesnt have that symptom, continue to query until the list is empty.
*/
query_symptoms(X) :- random_symptom(X),
                    (X==lump -> reaction, print('Do you have any lump?'), print('y/n: '), read(Answer), (Answer==y -> merge(lump) ; Answer==n );
                     X==whiteheads -> reaction, print('Do you have any whitehead?'), print('y/n: '), read(Answer), (Answer==y -> merge(whiteheads) ; Answer==n );
                     X==blackheads -> reaction, print('Do you have any blackhead?'), print('y/n: '), read(Answer), (Answer==y -> merge(blackheads) ; Answer==n );
                     X==pus -> reaction, print('Did you secrete pus?'), print('y/n: '), read(Answer), (Answer==y -> merge(pus) ; Answer==n );
                     X==cyst -> reaction, print('Do you have any cyst?'), print('y/n: '), read(Answer), (Answer==y -> merge(cyst) ; Answer==n );
                     X==scar -> reaction, print('Do you have any scar?'), print('y/n: '), read(Answer), (Answer==y -> merge(scar) ; Answer==n );
                     X==cough ->reaction, print('Do you have cough'), print('y/n: '), read(Answer), (Answer==y -> merge(cough) ; Answer==n );
                     X==runny_nose -> reaction, print('Are you having runny nose?'), print('y/n: '), read(Answer), (Answer==y -> merge(runny_nose) ; Answer==n );
                     X==ache -> reaction, print('Did you have ache?'), print('y/n: '), read(Answer), (Answer==y -> merge(ache) ; Answer==n );
                     X==weak -> reaction, print('Do you feel that your body is weak?'), print('y/n: '), read(Answer), (Answer==y -> merge(weak) ; Answer==n );
                     X==tired -> reaction, print('Do you feel perpetually tired?'), print('y/n: '), read(Answer), (Answer==y -> merge(tired) ; Answer==n );
                     X==fever -> reaction, print('Do you have a fever?'), print('y/n: '), read(Answer), (Answer==y -> merge(fever) ; Answer==n );
                     X==rash -> reaction, print('Do you have any rash?'), print('y/n: '), read(Answer), (Answer==y -> merge(rash) ; Answer==n );
                     X==wheeze -> reaction, print('Are you wheezing?'), print('y/n: '), read(Answer), (Answer==y -> merge(wheeze) ; Answer==n );
                     X==sneeze -> reaction, print('Do you sneeze frequently?'), print('y/n: '), read(Answer), (Answer==y -> merge(sneeze) ; Answer==n );
                     X==red_eye -> reaction, print('Do you have red eyes ?'), print('y/n: '), read(Answer), (Answer==y -> merge(red_eye ) ; Answer==n );
                     X==loss_of_speech -> reaction, print('Do you sometimes experience loss of speech?'), print('y/n: '), read(Answer), (Answer==y -> merge(loss_of_speech) ; Answer==n );
                     X==no_appetite -> reaction, print('Did you experience lack of appetite?'), print('y/n: '), read(Answer), (Answer==y -> merge(no_appetite) ; Answer==n );
                     X==leg_swell ->reaction, print('Do you have swelling on your leg?'), print('y/n: '), read(Answer), (Answer==y -> merge(leg_swell) ; Answer==n );
                     X==chest_pain -> reaction, print('Do you have any chest pain ?'), print('y/n: '), read(Answer), (Answer==y -> merge(chest_pain ) ; Answer==n );
                     X==weight_loss -> reaction, print('Do you experience drastic weight loss?'), print('y/n: '), read(Answer), (Answer==y -> merge(weight_loss) ; Answer==n );
                     X==pee_frequently-> reaction, print('Did you have to use the bathroom frequently?'), print('y/n: '), read(Answer), (Answer==y -> merge(pee_frequently) ; Answer==n );
                     X==thirst -> reaction, print('Do you always feel thirsty?'), print('y/n: '), read(Answer), (Answer==y -> merge(thirst) ; Answer==n );
                     X==blur_vision -> reaction, print('Do you have any blur vision?'), print('y/n: '), read(Answer), (Answer==y -> merge(blur_vision) ; Answer==n );
                     X==dry_mouth -> reaction, print('Do you have a dry mouth?'), print('y/n: '), read(Answer), (Answer==y -> merge(dry_mouth) ; Answer==n );
                     X==infection -> reaction, print('Do you have any infection?'), print('y/n: '), read(Answer), (Answer==y -> merge(infection) ; Answer==n );
                     X==pale_skin -> reaction, print('Do you sometimes have pale skin?'), print('y/n: '), read(Answer), (Answer==y -> merge(pale_skin) ; Answer==n );
                     X==breathless -> reaction, print('Did you experience breathlessness?'), print('y/n: '), read(Answer), (Answer==y -> merge(breathless) ; Answer==n );
                     X==bleed -> reaction, print('Do you often bleed?'), print('y/n: '), read(Answer), (Answer==y -> merge(bleed) ; Answer==n )
                    )
                    , symptoms(L), select(X, L, T), retractall(symptoms(_)), assertz(symptoms(T)), query_symptoms(_).
```

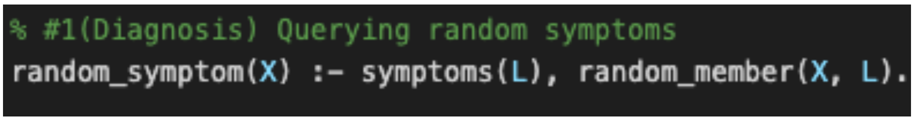
Step 6.2: Selects an appropriate Gesture before returning to *Step 6.1* to ask the patient sympathetically.



```
/*Picks a random Gesture from a list of reactions*/
random_reaction(X) :- all_reactions(L), random_member(X,L).

% #6.2 (Main) Get appropriate Gesture
/*Plays the chosen reaction, remove the used reaction and updates*/
reaction :- random_reaction(X), print(X), all_reactions(L), select(X, L, T), retractall(all_reactions(_)), assertz(all_reactions(T)).
```

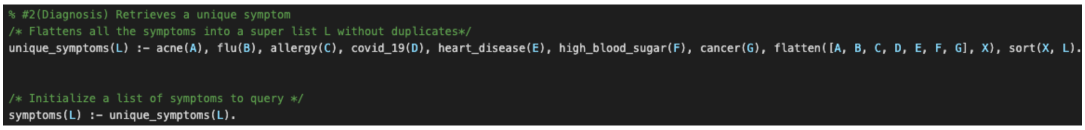
3.3 Logic Flow for Diagnosis

Step 1: Query for random symptom



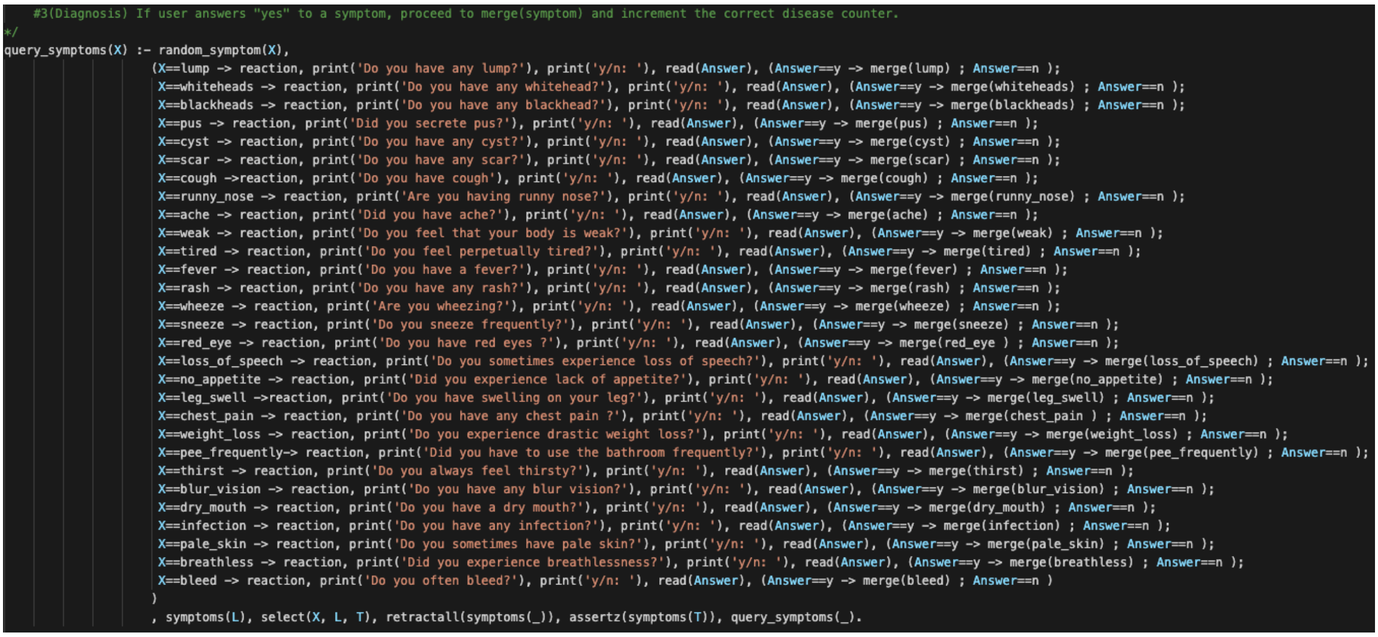
```
% #1(Diagnosis) Querying random symptoms
random_symptom(X) :- symptoms(L), random_member(X, L).
```

Step 2: Selects a unique symptom for Step 1



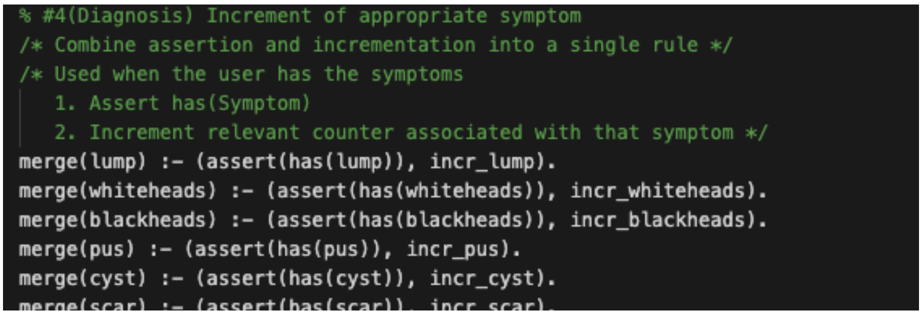
```
% #2(Diagnosis) Retrieves a unique symptom
/* Flattens all the symptoms into a super list L without duplicates*/
unique_symptoms(L) :- acne(A), flu(B), allergy(C), covid_19(D), heart_disease(E), high_blood_sugar(F), cancer(G), flatten([A, B, C, D, E, F, G], X), sort(X, L).


/* Initialize a list of symptoms to query */
symptoms(L) :- unique_symptoms(L).
```

Step 3: If patient answers “*Yes*” to a symptom, (1) *assert that symptom* and (2) *proceed to merge(symptom) program*.



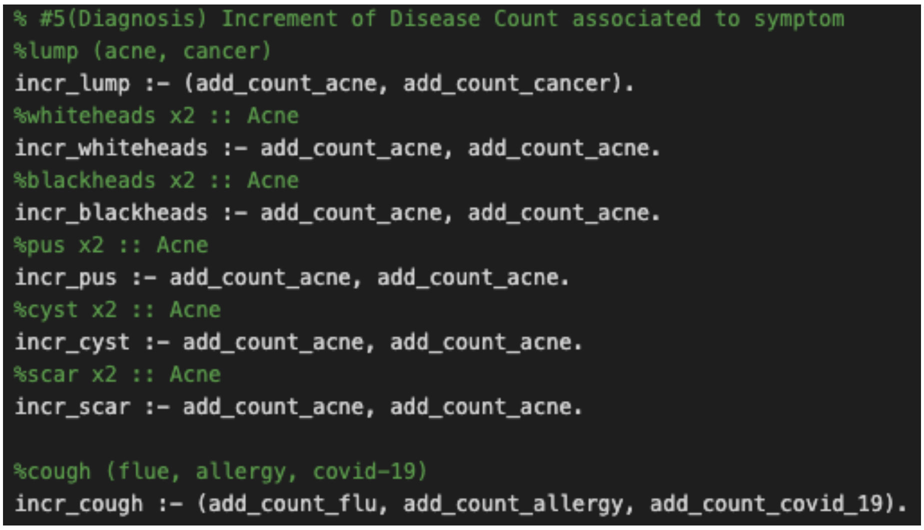
```
    #3(Diagnosis) If user answers "yes" to a symptom, proceed to merge(symptom) and increment the correct disease counter.
*/
query_symptoms(X) :- random_symptom(X),
                    (X==lump -> reaction, print('Do you have any lump?'), print('y/n: '), read(Answer), (Answer==y -> merge(lump) ; Answer==n );
                     X==whiteheads -> reaction, print('Do you have any whitehead?'), print('y/n: '), read(Answer), (Answer==y -> merge(whiteheads) ; Answer==n );
                     X==blackheads -> reaction, print('Do you have any blackhead?'), print('y/n: '), read(Answer), (Answer==y -> merge(blackheads) ; Answer==n );
                     X==pus -> reaction, print('Did you secrete pus?'), print('y/n: '), read(Answer), (Answer==y -> merge(pus) ; Answer==n );
                     X==cyst -> reaction, print('Do you have any cyst?'), print('y/n: '), read(Answer), (Answer==y -> merge(cyst) ; Answer==n );
                     X==scar -> reaction, print('Do you have any scar?'), print('y/n: '), read(Answer), (Answer==y -> merge(scar) ; Answer==n );
                     X==cough ->reaction, print('Do you have cough'), print('y/n: '), read(Answer), (Answer==y -> merge(cough) ; Answer==n );
                     X==runny_nose -> reaction, print('Are you having runny nose?'), print('y/n: '), read(Answer), (Answer==y -> merge(runny_nose) ; Answer==n );
                     X==ache -> reaction, print('Did you have ache?'), print('y/n: '), read(Answer), (Answer==y -> merge(ache) ; Answer==n );
                     X==weak -> reaction, print('Do you feel that your body is weak?'), print('y/n: '), read(Answer), (Answer==y -> merge(weak) ; Answer==n );
                     X==tired -> reaction, print('Do you feel perpetually tired?'), print('y/n: '), read(Answer), (Answer==y -> merge(tired) ; Answer==n );
                     X==fever -> reaction, print('Do you have a fever?'), print('y/n: '), read(Answer), (Answer==y -> merge(fever) ; Answer==n );
                     X==rash -> reaction, print('Do you have any rash?'), print('y/n: '), read(Answer), (Answer==y -> merge(rash) ; Answer==n );
                     X==wheeze -> reaction, print('Are you wheezing?'), print('y/n: '), read(Answer), (Answer==y -> merge(wheeze) ; Answer==n );
                     X==sneeze -> reaction, print('Do you sneeze frequently?'), print('y/n: '), read(Answer), (Answer==y -> merge(sneeze) ; Answer==n );
                     X==red_eye -> reaction, print('Do you have red eyes ?'), print('y/n: '), read(Answer), (Answer==y -> merge(red_eye ) ; Answer==n );
                     X==loss_of_speech -> reaction, print('Do you sometimes experience loss of speech?'), print('y/n: '), read(Answer), (Answer==y -> merge(loss_of_speech) ; Answer==n );
                     X==no_appetite -> reaction, print('Did you experience lack of appetite?'), print('y/n: '), read(Answer), (Answer==y -> merge(no_appetite) ; Answer==n );
                     X==leg_swell ->reaction, print('Do you have swelling on your leg?'), print('y/n: '), read(Answer), (Answer==y -> merge(leg_swell) ; Answer==n );
                     X==chest_pain -> reaction, print('Do you have any chest pain ?'), print('y/n: '), read(Answer), (Answer==y -> merge(chest_pain ) ; Answer==n );
                     X==weight_loss -> reaction, print('Do you experience drastic weight loss?'), print('y/n: '), read(Answer), (Answer==y -> merge(weight_loss) ; Answer==n );
                     X==pee_frequently-> reaction, print('Did you have to use the bathroom frequently?'), print('y/n: '), read(Answer), (Answer==y -> merge(pee_frequently) ; Answer==n );
                     X==thirst -> reaction, print('Do you always feel thirsty?'), print('y/n: '), read(Answer), (Answer==y -> merge(thirst) ; Answer==n );
                     X==blur_vision -> reaction, print('Do you have any blur vision?'), print('y/n: '), read(Answer), (Answer==y -> merge(blur_vision) ; Answer==n );
                     X==dry_mouth -> reaction, print('Do you have a dry mouth?'), print('y/n: '), read(Answer), (Answer==y -> merge(dry_mouth) ; Answer==n );
                     X==infection -> reaction, print('Do you have any infection?'), print('y/n: '), read(Answer), (Answer==y -> merge(infection) ; Answer==n );
                     X==pale_skin -> reaction, print('Do you sometimes have pale skin?'), print('y/n: '), read(Answer), (Answer==y -> merge(pale_skin) ; Answer==n );
                     X==breathless -> reaction, print('Did you experience breathlessness?'), print('y/n: '), read(Answer), (Answer==y -> merge(breathless) ; Answer==n );
                     X==bleed -> reaction, print('Do you often bleed?'), print('y/n: '), read(Answer), (Answer==y -> merge(bleed) ; Answer==n )
                    )
                    , symptoms(L), select(X, L, T), retractall(symptoms(_)), assertz(symptoms(T)), query_symptoms(_).
```

Step 4: Assert the appropriate symptom and increment the symptom accordingly.(*This is a snippet*)



```
% #4(Diagnosis) Increment of appropriate symptom
/* Combine assertion and incrementation into a single rule */
/* Used when the user has the symptoms
    1. Assert has(Symptom)
    2. Increment relevant counter associated with that symptom */
merge(lump) :- (assert(has(lump)), incr_lump).
merge(whiteheads) :- (assert(has(whiteheads)), incr_whiteheads).
merge(blackheads) :- (assert(has(blackheads)), incr_blackheads).
merge(pus) :- (assert(has(pus)), incr_pus).
merge(cyst) :- (assert(has(cyst)), incr_cyst).
merge(scar) :- (assert(has(scar)), incr_scar).
```

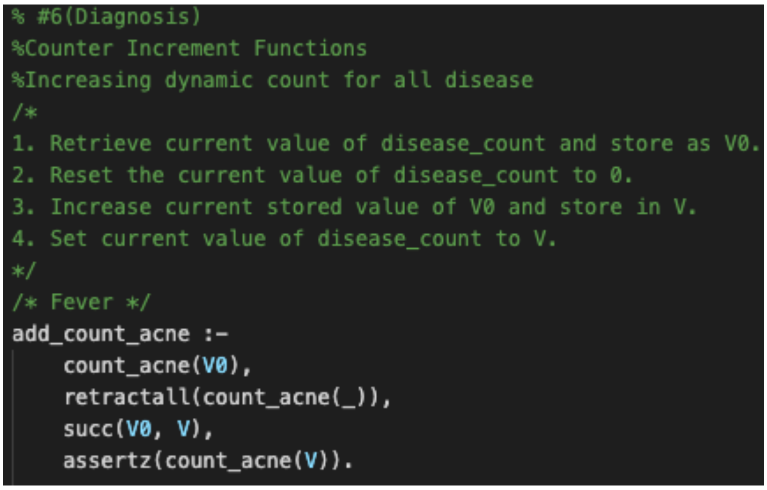
Step 5: Increments Disease Counter (*This is only a snippet, please refer to the code for the full implementation*)



```
% #5(Diagnosis) Increment of Disease Count associated to symptom
%lump (acne, cancer)
incr_lump :- (add_count_acne, add_count_cancer).
%whiteheads x2 :: Acne
incr_whiteheads :- add_count_acne, add_count_acne.
%blackheads x2 :: Acne
incr_blackheads :- add_count_acne, add_count_acne.
%pus x2 :: Acne
incr_pus :- add_count_acne, add_count_acne.
%cyst x2 :: Acne
incr_cyst :- add_count_acne, add_count_acne.
%scar x2 :: Acne
incr_scar :- add_count_acne, add_count_acne.

%cough (flue, allergy, covid-19)
incr_cough :- (add_count_flu, add_count_allergy, add_count_covid_19).
```

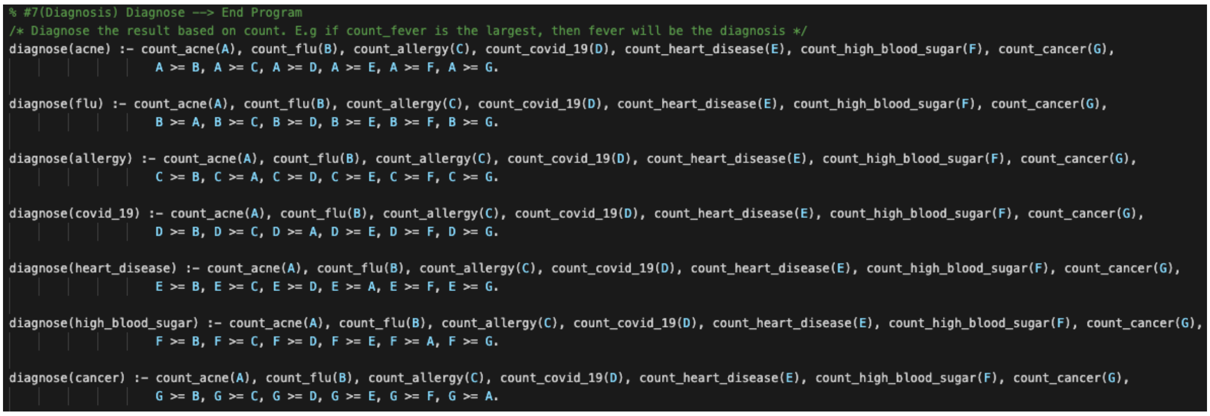
Step 6: Increments "heuristic"/count_value for a specific disease.

(*This is only a snippet, please refer to the code for the full implementation*)



```
% #6(Diagnosis)
%Counter Increment Functions
%Increasing dynamic count for all disease
/*
1. Retrieve current value of disease_count and store as V0.
2. Reset the current value of disease_count to 0.
3. Increase current stored value of V0 and store in V.
4. Set current value of disease_count to V.
*/
/* Fever */
add_count_acne :-
    count_acne(V0),
    retractall(count_acne(_)),
    succ(V0, V),
    assertz(count_acne(V)).
```

Step 7: When symptom list is empty, proceed to diagnose program. Disease with the *highest heuristic* will be considered for the diagnosis of the patient.



```
% #7(Diagnosis) Diagnose --> End Program
/* Diagnose the result based on count. E.g if count_fever is the largest, then fever will be the diagnosis */
diagnose(acne) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
                    A >= B, A >= C, A >= D, A >= E, A >= F, A >= G.

diagnose(flu) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
                    B >= A, B >= C, B >= D, B >= E, B >= F, B >= G.

diagnose(allergy) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
                    C >= B, C >= A, C >= D, C >= E, C >= F, C >= G.

diagnose(covid_19) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
                    D >= B, D >= C, D >= A, D >= E, D >= F, D >= G.

diagnose(heart_disease) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
                    E >= B, E >= C, E >= D, E >= A, E >= F, E >= G.

diagnose(high_blood_sugar) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
                    F >= B, F >= C, F >= D, F >= E, F >= A, F >= G.

diagnose(cancer) :- count_acne(A), count_flu(B), count_allergy(C), count_covid_19(D), count_heart_disease(E), count_high_blood_sugar(F), count_cancer(G),
                    G >= B, G >= C, G >= D, G >= E, G >= F, G >= A.
```

3.4 Heuristics/ Algorithm for conflicting diagnosis

One of the problems faced is the possibility of 2 diseases with overlapping symptom, for instance, count_flu = X and count_cancer = X. However, patient is diagnose with flu instead of cancer mainly due to the order. As such, to resolve this problem, I've implemented a ***weighted/heuristic system*** whereby the count will be incremented more if the symptom is a stronger indicator for that specific disease.

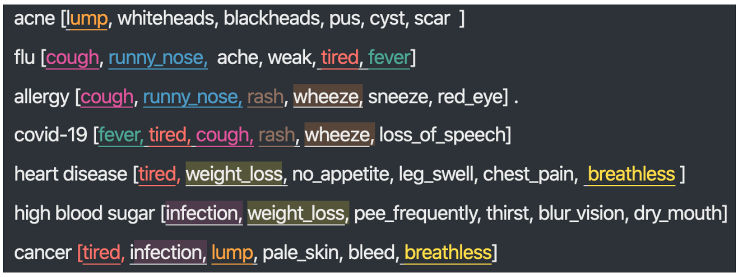
The heuristic is governed by the following algorithm:

(1) For disease with non-unique symptoms, the increment weight is 1.

(2) For disease with total unique symptoms > 2, the increment weight is 2.

(3) For disease with total unique symptoms <=2, the increment weight is 3.

(More explanation on implementation can be found in source code under Heruistics section)

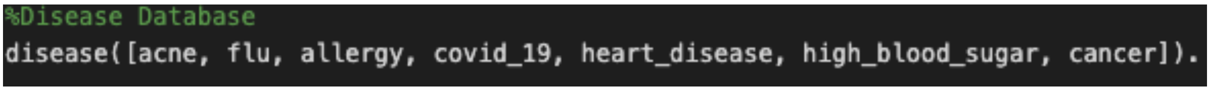


acne [lump, whiteheads, blackheads, pus, cyst, scar]
flu [cough, runny_nose, ache, weak, tired, fever]
allergy [cough, runny_nose, rash, wheeze, sneeze, red_eye] .
covid-19 [fever, tired, cough, rash, wheeze, loss_of_speech]
heart disease [tired, weight_loss, no_appetite, leg_swell, chest_pain, breathless]
high blood sugar [infection, weight_loss, pee_frequently, thirst, blur_vision, dry_mouth]
cancer [tired, infection, lump, pale_skin, bleed, breathless]

Figure 4 Unique symptoms == Stronger Indicator (Non-Highlighted & Non-Underlined)

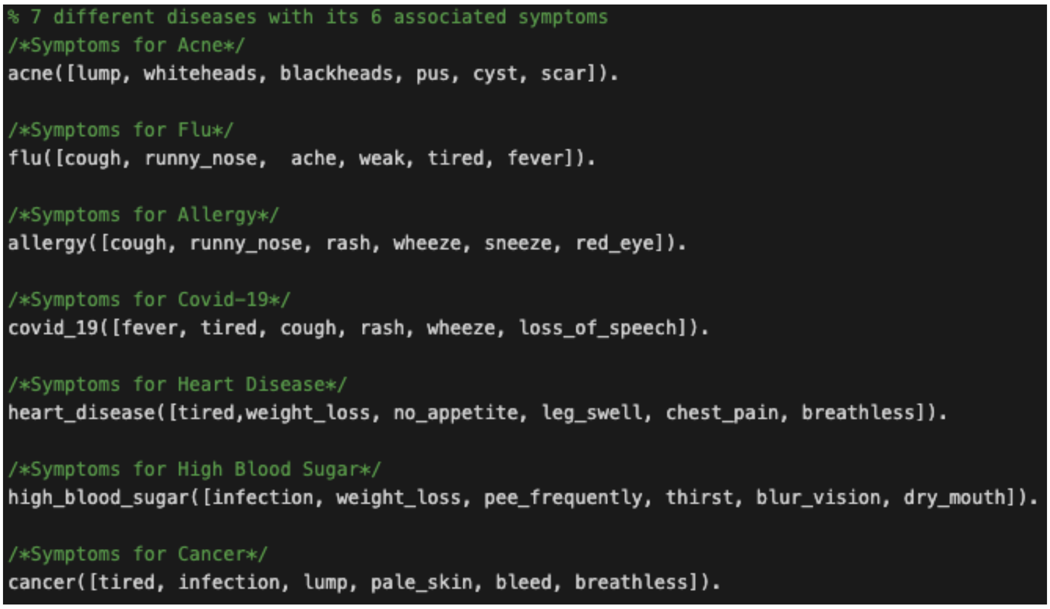
3.5 Knowledge Database

Disease Database



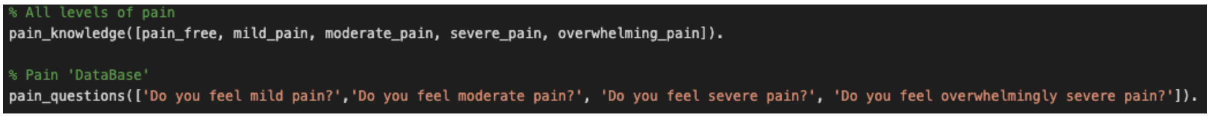
```
%Disease Database
disease([acne, flu, allergy, covid_19, heart_disease, high_blood_sugar, cancer]).
```

Symptoms Database



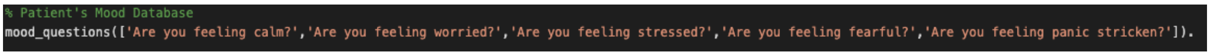
```
% 7 different diseases with its 6 associated symptoms
/*Symptoms for Acne*/
acne([lump, whiteheads, blackheads, pus, cyst, scar]).

/*Symptoms for Flu*/
flu([cough, runny_nose,  ache, weak, tired, fever]).

/*Symptoms for Allergy*/
allergy([cough, runny_nose, rash, wheeze, sneeze, red_eye]).

/*Symptoms for Covid-19*/
covid_19([fever, tired, cough, rash, wheeze, loss_of_speech]).

/*Symptoms for Heart Disease*/
heart_disease([tired,weight_loss, no_appetite, leg_swell, chest_pain, breathless]).

/*Symptoms for High Blood Sugar*/
high_blood_sugar([infection, weight_loss, pee_frequently, thirst, blur_vision, dry_mouth]).

/*Symptoms for Cancer*/
cancer([tired, infection, lump, pale_skin, bleed, breathless]).
```

Pain Database (Note: ‘pain_free’ question is asked in ask(0) program of “Do you feel any pain?”)



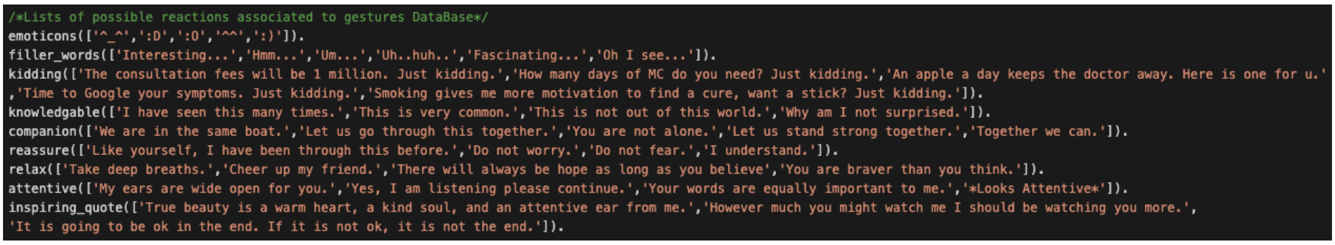
```
% All levels of pain
pain_knowledge([pain_free, mild_pain, moderate_pain, severe_pain, overwhelming_pain]).

% Pain 'DataBase'
pain_questions(['Do you feel mild pain?','Do you feel moderate pain?', 'Do you feel severe pain?', 'Do you feel overwhelmingly severe pain?']).
```

Mood Database



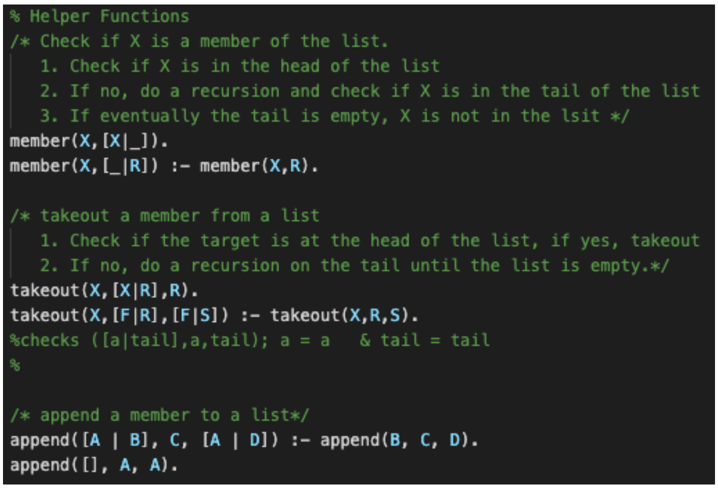
```
% Patient's Mood Database
mood_questions(['Are you feeling calm?','Are you feeling worried?','Are you feeling stressed?','Are you feeling fearful?','Are you feeling panic stricken?']).
```

Gestures Database



```
/*Lists of possible reactions associated to gestures DataBase*/
emoticons(['^_^',':D',':O','^^',':)']).
filler_words(['Interesting...','Hmm...','Um...','Uh..huh..','Fascinating...','Oh I see...']).
kidding(['The consultation fees will be 1 million. Just kidding.','How many days of MC do you need? Just kidding.','An apple a day keeps the doctor away. Here is one for u.'
,'Time to Google your symptoms. Just kidding.','Smoking gives me more motivation to find a cure, want a stick? Just kidding.']).
knowledgable(['I have seen this many times.','This is very common.','This is not out of this world.','Why am I not surprised.']).
companion(['We are in the same boat.','Let us go through this together.','You are not alone.','Let us stand strong together.','Together we can.']).
reassure(['Like yourself, I have been through this before.','Do not worry.','Do not fear.','I understand.']).
relax(['Take deep breaths.','Cheer up my friend.','There will always be hope as long as you believe','You are braver than you think.']).
attentive(['My ears are wide open for you.','Yes, I am listening please continue.','Your words are equally important to me.','*Looks Attentive*']).
inspiring_quote(['True beauty is a warm heart, a kind soul, and an attentive ear from me.','However much you might watch me I should be watching you more.',
'It is going to be ok in the end. If it is not ok, it is not the end.']).
```

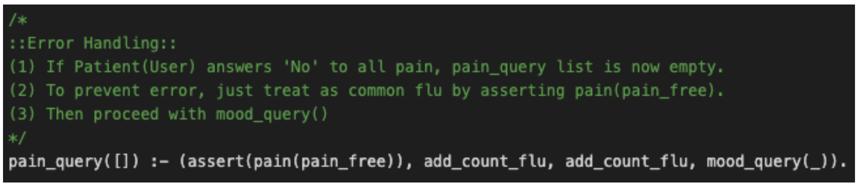
3.6 Helper Functions



```
% Helper Functions
/* Check if X is a member of the list.
   1. Check if X is in the head of the list
   2. If no, do a recursion and check if X is in the tail of the list
   3. If eventually the tail is empty, X is not in the lsit */
member(X,[X|_]).
member(X,[_|R]) :- member(X,R).

/* takeout a member from a list
   1. Check if the target is at the head of the list, if yes, takeout
   2. If no, do a recursion on the tail until the list is empty.*/
takeout(X,[X|R],R).
takeout(X,[F|R],[F|S]) :- takeout(X,R,S).
%checks ([a|tail],a,tail); a = a   & tail = tail
%

/* append a member to a list*/
append([A | B], C, [A | D]) :- append(B, C, D).
append([], A, A).
```

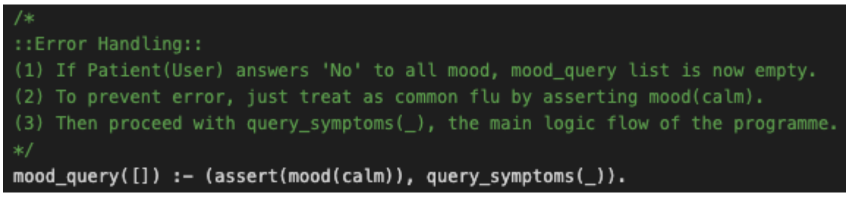
3.7 Further Considerations (Error Handling)

Rules governing pain query empty list (error handling).



```
/*
::Error Handling::
(1) If Patient(User) answers 'No' to all pain, pain_query list is now empty.
(2) To prevent error, just treat as common flu by asserting pain(pain_free).
(3) Then proceed with mood_query()
*/
pain_query([]) :- (assert(pain(pain_free)), add_count_flu, add_count_flu, mood_query(_)).
```

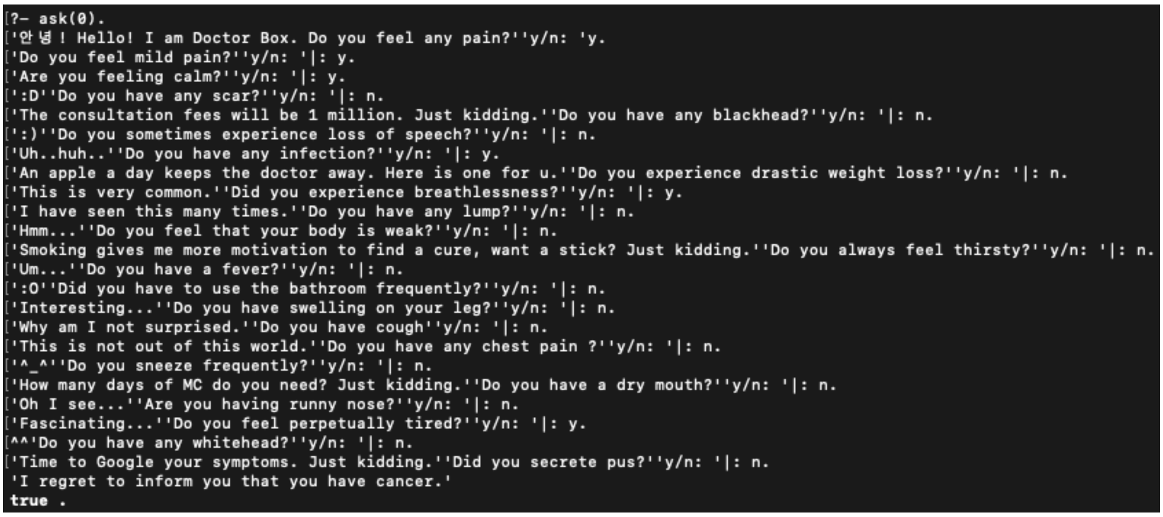
Rules governing mood query empty list (error handling).



```
/*
::Error Handling::
(1) If Patient(User) answers 'No' to all mood, mood_query list is now empty.
(2) To prevent error, just treat as common flu by asserting mood(calm).
(3) Then proceed with query_symptoms(_), the main logic flow of the programme.
*/
mood_query([]) :- (assert(mood(calm)), query_symptoms(_)).
```

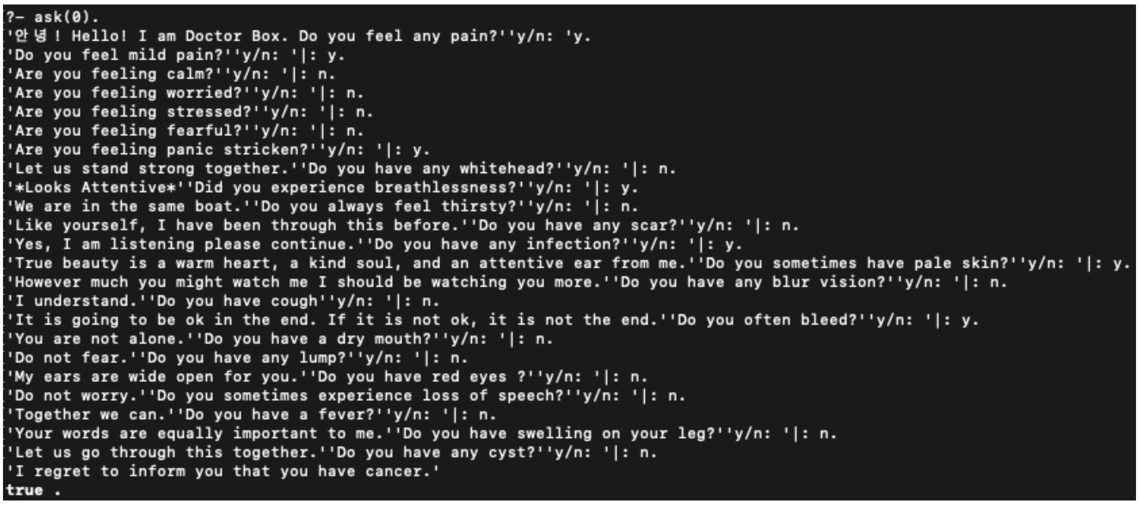
Section 4. Demonstration of Sympathetic Doctor Dialogue AI

Case 1: Cancer disease with (1) **mild pain** and (2) **calm mood**. (Notice the gesture)



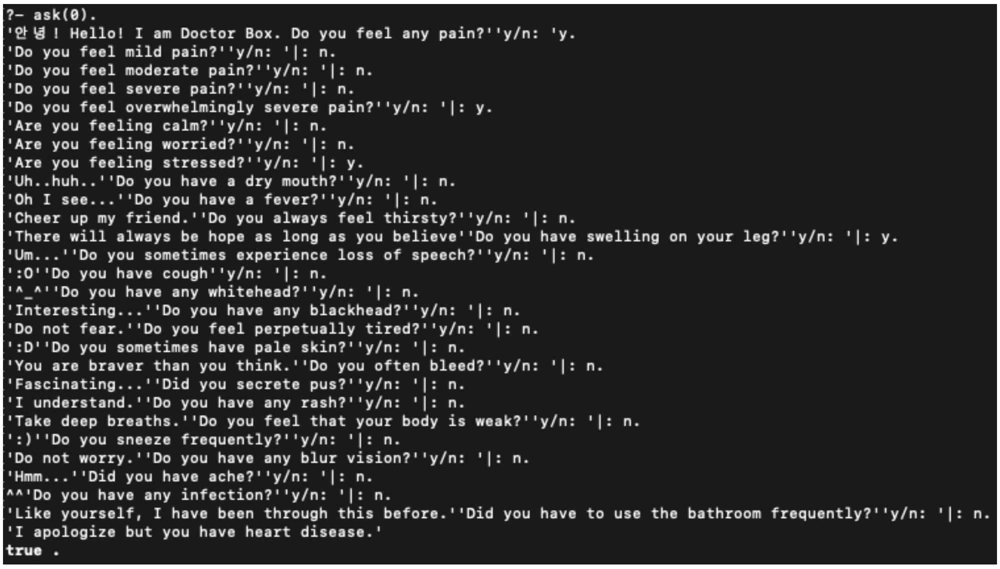
```
[?- ask(0).
['안녕 ! Hello! I am Doctor Box. Do you feel any pain?''y/n: 'y.
['Do you feel mild pain?''y/n: '|: y.
['Are you feeling calm?''y/n: '|: y.
[':D''Do you have any scar?''y/n: '|: n.
['The consultation fees will be 1 million. Just kidding.''Do you have any blackhead?''y/n: '|: n.
[':)''Do you sometimes experience loss of speech?''y/n: '|: n.
['Uh..huh..''Do you have any infection?''y/n: '|: y.
['An apple a day keeps the doctor away. Here is one for u.''Do you experience drastic weight loss?''y/n: '|: n.
['This is very common.''Did you experience breathlessness?''y/n: '|: y.
['I have seen this many times.''Do you have any lump?''y/n: '|: n.
['Hmm...''Do you feel that your body is weak?''y/n: '|: n.
['Smoking gives me more motivation to find a cure, want a stick? Just kidding.''Do you always feel thirsty?''y/n: '|: n.
['Um...''Do you have a fever?''y/n: '|: n.
[':O''Did you have to use the bathroom frequently?''y/n: '|: n.
['Interesting...''Do you have swelling on your leg?''y/n: '|: n.
['Why am I not surprised.''Do you have cough''y/n: '|: n.
['This is not out of this world.''Do you have any chest pain ?''y/n: '|: n.
['^_^''Do you sneeze frequently?''y/n: '|: n.
['How many days of MC do you need? Just kidding.''Do you have a dry mouth?''y/n: '|: n.
['Oh I see...''Are you having runny nose?''y/n: '|: n.
['Fascinating...''Do you feel perpetually tired?''y/n: '|: y.
[^^'Do you have any whitehead?''y/n: '|: n.
['Time to Google your symptoms. Just kidding.''Did you secrete pus?''y/n: '|: n.
 'I regret to inform you that you have cancer.'
 true .
```

Case 2: Cancer disease with (1) **mild pain** and (2) **panic stricken mood**. (Notice the difference in gesture from case 1)



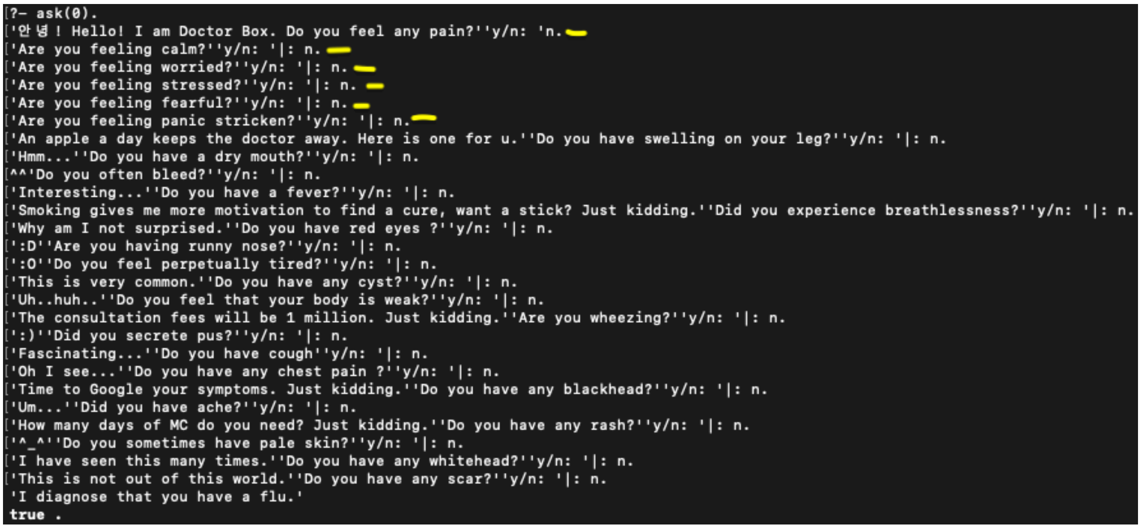
```
?- ask(0).
'안녕 ! Hello! I am Doctor Box. Do you feel any pain?''y/n: 'y.
'Do you feel mild pain?''y/n: '|: y.
'Are you feeling calm?''y/n: '|: n.
'Are you feeling worried?''y/n: '|: n.
'Are you feeling stressed?''y/n: '|: n.
'Are you feeling fearful?''y/n: '|: n.
'Are you feeling panic stricken?''y/n: '|: y.
'Let us stand strong together.''Do you have any whitehead?''y/n: '|: n.
'*Looks Attentive*''Did you experience breathlessness?''y/n: '|: y.
'We are in the same boat.''Do you always feel thirsty?''y/n: '|: n.
'Like yourself, I have been through this before.''Do you have any scar?''y/n: '|: n.
'Yes, I am listening please continue.''Do you have any infection?''y/n: '|: y.
'True beauty is a warm heart, a kind soul, and an attentive ear from me.''Do you sometimes have pale skin?''y/n: '|: y.
'However much you might watch me I should be watching you more.''Do you have any blur vision?''y/n: '|: n.
'I understand.''Do you have cough''y/n: '|: n.
'It is going to be ok in the end. If it is not ok, it is not the end.''Do you often bleed?''y/n: '|: y.
'You are not alone.''Do you have a dry mouth?''y/n: '|: n.
'Do not fear.''Do you have any lump?''y/n: '|: n.
'My ears are wide open for you.''Do you have red eyes ?''y/n: '|: n.
'Do not worry.''Do you sometimes experience loss of speech?''y/n: '|: n.
'Together we can.''Do you have a fever?''y/n: '|: n.
'Your words are equally important to me.''Do you have swelling on your leg?''y/n: '|: n.
'Let us go through this together.''Do you have any cyst?''y/n: '|: n.
'I regret to inform you that you have cancer.'
true .
```

Case 3: Different level of pain and different diagnose result.



```
?- ask(0).
'안녕 ! Hello! I am Doctor Box. Do you feel any pain?''y/n: 'y.
'Do you feel mild pain?''y/n: '|: n.
'Do you feel moderate pain?''y/n: '|: n.
'Do you feel severe pain?''y/n: '|: n.
'Do you feel overwhelmingly severe pain?''y/n: '|: y.
'Are you feeling calm?''y/n: '|: n.
'Are you feeling worried?''y/n: '|: n.
'Are you feeling stressed?''y/n: '|: y.
'Uh..huh..''Do you have a dry mouth?''y/n: '|: n.
'Oh I see...''Do you have a fever?''y/n: '|: n.
'Cheer up my friend.''Do you always feel thirsty?''y/n: '|: n.
'There will always be hope as long as you believe''Do you have swelling on your leg?''y/n: '|: y.
'Um...''Do you sometimes experience loss of speech?''y/n: '|: n.
':O''Do you have cough''y/n: '|: n.
'^_^''Do you have any whitehead?''y/n: '|: n.
'Interesting...''Do you have any blackhead?''y/n: '|: n.
'Do not fear.''Do you feel perpetually tired?''y/n: '|: n.
':D''Do you sometimes have pale skin?''y/n: '|: n.
'You are braver than you think.''Do you often bleed?''y/n: '|: n.
'Fascinating...''Did you secrete pus?''y/n: '|: n.
'I understand.''Do you have any rash?''y/n: '|: n.
'Take deep breaths.''Do you feel that your body is weak?''y/n: '|: n.
':)''Do you sneeze frequently?''y/n: '|: n.
'Do not worry.''Do you have any blur vision?''y/n: '|: n.
'Hmm...''Did you have ache?''y/n: '|: n.
^^'Do you have any infection?''y/n: '|: n.
'Like yourself, I have been through this before.''Did you have to use the bathroom frequently?''y/n: '|: n.
'I apologize but you have heart disease.'
true .
```

<u>Case 4</u>: <u>Handle Empty List Error</u>; when all pain level and mood level are answered with “no”.



```
[?- ask(0).
['안녕 ! Hello! I am Doctor Box. Do you feel any pain?''y/n: 'n.
['Are you feeling calm?''y/n: '|: n.
['Are you feeling worried?''y/n: '|: n.
['Are you feeling stressed?''y/n: '|: n.
['Are you feeling fearful?''y/n: '|: n.
['Are you feeling panic stricken?''y/n: '|: n.
['An apple a day keeps the doctor away. Here is one for u.''Do you have swelling on your leg?''y/n: '|: n.
['Hmm...''Do you have a dry mouth?''y/n: '|: n.
[^^'Do you often bleed?''y/n: '|: n.
['Interesting...''Do you have a fever?''y/n: '|: n.
['Smoking gives me more motivation to find a cure, want a stick? Just kidding.''Did you experience breathlessness?''y/n: '|: n.
['Why am I not surprised.''Do you have red eyes ?''y/n: '|: n.
[':D''Are you having runny nose?''y/n: '|: n.
[':O''Do you feel perpetually tired?''y/n: '|: n.
['This is very common.''Do you have any cyst?''y/n: '|: n.
['Uh..huh..''Do you feel that your body is weak?''y/n: '|: n.
['The consultation fees will be 1 million. Just kidding.''Are you wheezing?''y/n: '|: n.
[':)''Did you secrete pus?''y/n: '|: n.
['Fascinating...''Do you have cough''y/n: '|: n.
['Oh I see...''Do you have any chest pain ?''y/n: '|: n.
['Time to Google your symptoms. Just kidding.''Do you have any blackhead?''y/n: '|: n.
['Um...''Did you have ache?''y/n: '|: n.
['How many days of MC do you need? Just kidding.''Do you have any rash?''y/n: '|: n.
['^_^''Do you sometimes have pale skin?''y/n: '|: n.
['I have seen this many times.''Do you have any whitehead?''y/n: '|: n.
['This is not out of this world.''Do you have any scar?''y/n: '|: n.
 'I diagnose that you have a flu.'
 true .
```

Section 5: Additional Implementation (<u>GUI</u>) with <u>Google Speech Recognition Engine</u>

5.1 Motivation for GUI Implementation (Additional Feature)

While it is entirely feasible to have Prolog KBS to be the forefront of the application, it still presents its form of inconvenience to users. For instance, replying to KBS requires the user to spell both "no" and "yes" correctly. And as humans, we tend to err easily, and this could prove to be catastrophic if the KBS is used to run on a larger scale. Misspelling “no” and “yes” would cause a system crash in the KBS.

As such, to resolve this issue. Having an interactable GUI would serve as a form of layered abstraction, thereby, hiding the internal logic of the KBS from the user. The chances of a system crash would be reduced significantly.

5.2 Overall System Architecture Diagram of the GUI Implementation

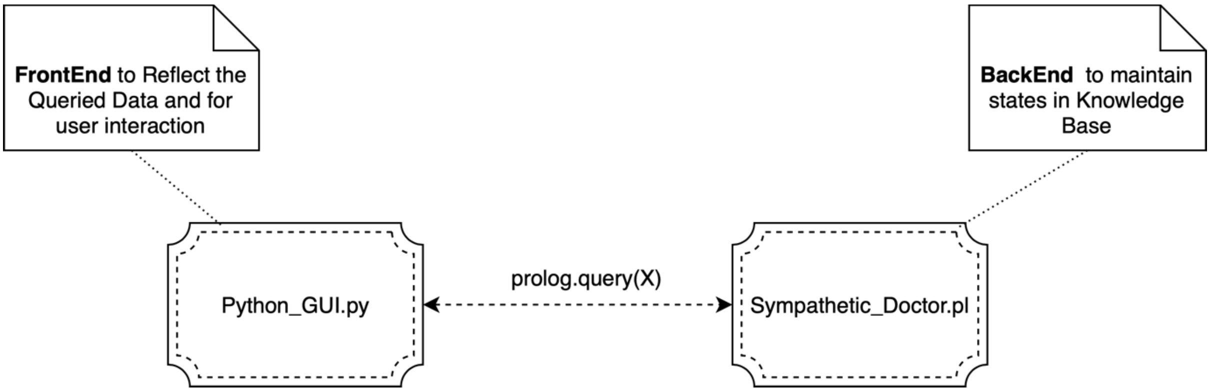

FrontEnd to Reflect the Queried Data and for user interaction
BackEnd to maintain states in Knowledge Base
Python_GUI.py
prolog.query(X)
Sympathetic_Doctor.pl

Figure 5 System Architecture Diagram of Implemented GUI

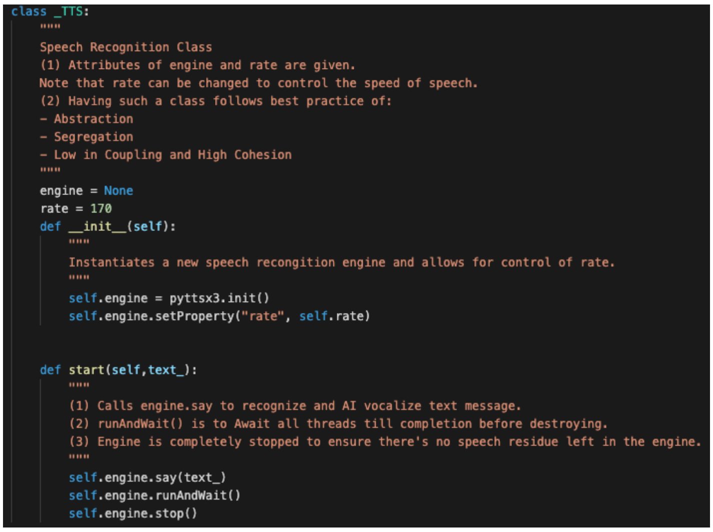
5.3 Set up and Installation

1. Ensure that you have Python and SWI-Prolog installed on your computer.
    a. Python 2.7 or 3.4 and higher.
    b. SWI-Prolog 7.2.x and higher.
    c. Both must have similar bit architecture. (E.g both 64 bit).
2. Install PySwip using the command: pip install pyswip
3. Install PySimpleGUI using the command: pip
4. Install PySimpleGUI Pyttsx3 using the command: pip install pyttsx3
    a. Additionally, for windows, install Pypiwin32 using the command: pip install pypiwin32
5. Install SpeechRecognition using the command: pip install SpeechRecognition
    a. Install pyaudio using the command: pip install pyaudio
    For python 3.7 and above, download the appropriate file from
    https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyaudio
    b. Example Python 3.8, 64 bit -> download PyAudio-0.2.11-cp38-cp38-
    c. win_amd64
    d. Go to the directory where the file is and install it using the command: pip install [file_name].whl

5.4 Implementation of speech recognition on GUI

(*Full implementation can be found in source code-* ***Sympathetic_Doctor_GUI.py***)

Here, I will only highlight the class I've used to instantiate a speech recognition engine used in this GUI. Also note that, running a speech recognition is thread-intensive. As such, I've only implemented the speech recognition to instantiate twice. Once at the beginning of the program, "Hello! I am Doctor Box. Do you feel any pain?", and the other at the end of the program, when the Doctor diagnoses, "I diagnose that you have a….".

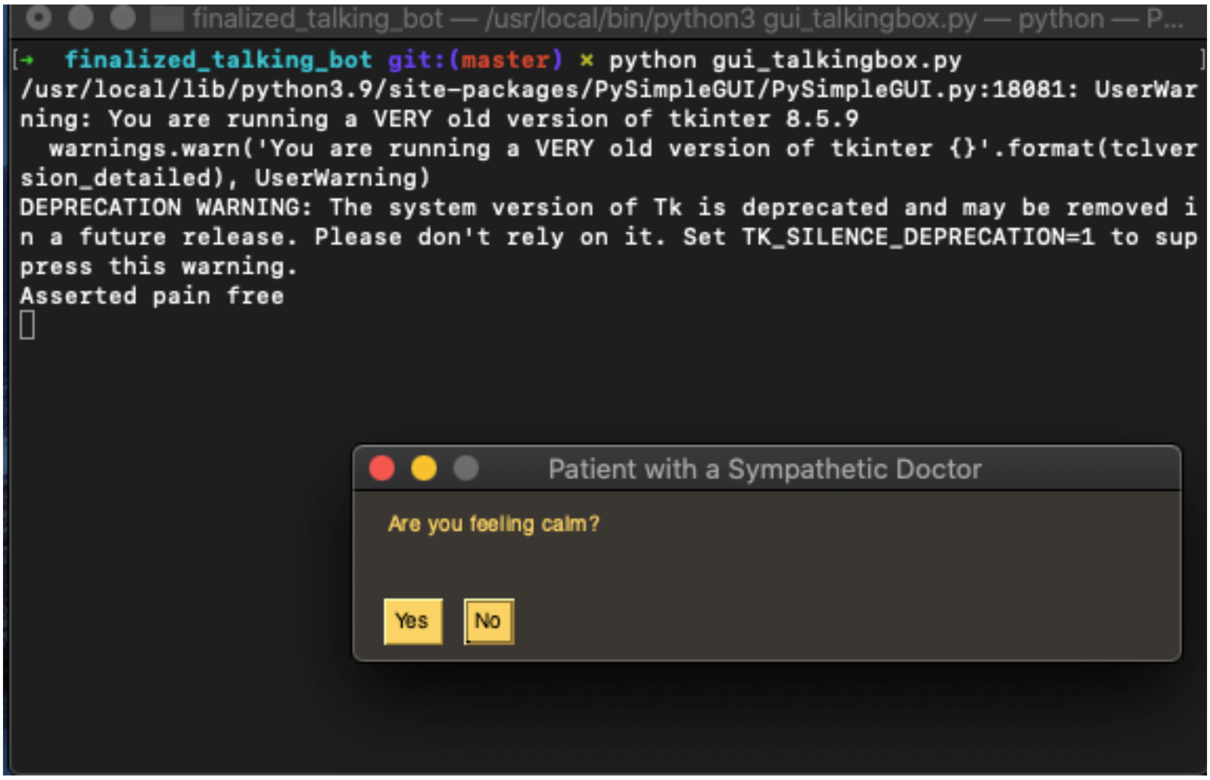


```
class _TTS:
    """
    Speech Recognition Class
    (1) Attributes of engine and rate are given.
    Note that rate can be changed to control the speed of speech.
    (2) Having such a class follows best practice of:
    - Abstraction
    - Segregation
    - Low in Coupling and High Cohesion
    """
    engine = None
    rate = 170
    def __init__(self):
        """
        Instantiates a new speech recongition engine and allows for control of rate.
        """
        self.engine = pyttsx3.init()
        self.engine.setProperty("rate", self.rate)


    def start(self,text_):
        """
        (1) Calls engine.say to recognize and AI vocalize text message.
        (2) runAndWait() is to Await all threads till completion before destroying.
        (3) Engine is completely stopped to ensure there's no speech residue left in the engine.
        """
        self.engine.say(text_)
        self.engine.runAndWait()
        self.engine.stop()
```

Figure 6 Speech Recognition Class

5.5 Screenshot of GUI Implementation

Notice that a “*Yes*” and “*No*” buttons are present for the patient to interact with, without having to worry about the internal logic of the KBS.


finalized_talking_bot — /usr/local/bin/python3 gui_talkingbox.py — python — P...
Patient with a Sympathetic Doctor
Yes
No

```
[→  finalized_talking_bot git:(master) ✗ python gui_talkingbox.py                  ]
/usr/local/lib/python3.9/site-packages/PySimpleGUI/PySimpleGUI.py:18081: UserWar
ning: You are running a VERY old version of tkinter 8.5.9
  warnings.warn('You are running a VERY old version of tkinter {}'.format(tclver
sion_detailed), UserWarning)
DEPRECATION WARNING: The system version of Tk is deprecated and may be removed i
n a future release. Please don't rely on it. Set TK_SILENCE_DEPRECATION=1 to sup
press this warning.
Asserted pain free
```

Are you feeling calm?

Figure 7 GUI with "Yes"/"No" button with Speech Recognition

5.6 Demonstration of Sympathetic Doctor Dialogue AI with GUI (YouTube Link)

The demonstration of the Sympathetic Doctor GUI Video can be found in the zipped folder or via this YouTube link:

https://www.youtube.com/watch?v=IfDK-hPf6fs

The following video will demonstrate on the following three cases:

- Case 1: Demonstration on the Doctor’s Diagnosis for Flu
- Case 2: Demonstration on the Doctor’s Diagnosis for Heart Disease
- Case 3: Demonstration on the Doctor’s Diagnosis for Cancer

(Note that the source code can be found in Sympathetic_Doctor_GUI.py)

Section 6. Conclusion

Through this assignment, we can see the advantages of using Prolog as a higher-level computer programming language that are closer to the programmer than to the computer. And with the rising trend for enterprises that want to be successful with using high information technology, I have no qualms that Prolog would be a significant and efficient contribution towards the development of Artificial Intelligence-related applications.

Works Cited

[1] A. K. A. C. Piyush Ranjan, "How can Doctors Improve thier Communication Skills?," *NCBI,* vol. 9, no. 3, pp. 01-04, 2015.

[2] NHS, "NHS inform," [Online]. Available: https://www.nhsinform.scot/illnesses-and-conditions/immune-system/allergies. [Accessed 19 November 2020].