Designing and implementing an EXPERT SYSTEM on

Medical Expert System

Submitted to the

Savitribai Phule Pune University

In partial fulfilment for the award of the Degree of

Bachelor of Engineering

in

COMPUTER ENGINEERING

by

SHRUTI NAGESH LONDHE

PRN NO :72029317M

SOHAM SAWANT

PRN NO :72029323F

ANUJ DWIVEDI

PRN NO : 72029103J

Under the guidance of

Mrs. Swati Mohite

Sinhgad Institutes

Department Of Computer Science

STES'S SINHGAD ACADEMY OF ENGINEERING

KONDHWA BK, PUNE 411048

2021-2022

Sinhgad Institutes

CERTIFICATE

This is to certify that the mini project report entitled **"Medical Expert System"** as AI's mini project being submitted by GROUP is a record of bonafide work carried out by them under the supervision and guidance of Mrs Swati Mohite in partial fulfilment of the requirement for TE – 2019 course of Savitribai Phule Pune University, Pune in the academic year 2021-2022.

Date: 09/05/22

Place: Pune

Guide Subject Coordinator Head of the Department

Principal

This Mini Project report has been examined by us as per the Savitribai Phule Pune University, Pune requirements at **SINHGAD ACADEMY OF ENGINEERING** Pune – 411048 on . . . .

Internal Examiner External Examiner

ACKNOWLEDGEMENT

We are going to make a project entitled **"Medical Expert System".** We all are 3 people. We are going to apply skills in solving one problem which is not too big but not less thann that once. We are happy to make a laptop medical expert system which helps to diagnose the diseases of a patient by asking questions about any symptoms. We would like to express our sincere gratitude to our project supervisor **Mrs SWATI MOHITE** who has in the literal sense, steered and supervised us. Their guidance deserves much more than the credit on the successful completion of our project and also for giving us an opportunity to implant our skills and for providing the ample help and co-operation to complete the report. We are indebted with a deep sense of gratitude for the constant inspiration and valuable guidance throughout the work. We are very grateful and indebted to our project guide **Mrs. SWATI MOHITE** for providing their enduring patience, guidance & invaluable suggestions. They were the ones who never let our morale down & always supported us. They were the constant source of inspiration for us & took utmost interest in our project.

(Students Name & Signature)

CONTENTS

Sr.No	TITLE
1.	Abstract
2.	Introduction
3.	Problem statement
4.	Objective
5.	Introduction
6.	Methodology
7.	Motivation
8.	Dependencies, installation, project layout
9.	Output
10.	Conclusion
11.	References

`

1. ***ABSTRACT***

This report includes a development presentation of an expert system for helping people diagnose diseases by asking questions about symptoms The system as such as it has been developed is called Medical Expert System. It consists of Python, HTML, CSS and python frameworks. The choice of the programming tools is individual and particular. An Medical expert system is useful for the healthcare department This is just for demonstration purposes. The symptoms and diseases are likely to match. It is an attempt to showcase how powerful these basic systems can get.

2. *INTRODUCTION*

Expert systems technology, as a category of computer based artificial intelligence, offers a number of possibilities for further automation and improvements in managing office documents. Because of its potential, expert systems represent an interesting subject for study and implementation. To design and develop an expert system application is a challenge. Different expert systems are already being applied in many areas of human activity like healthcare

An expert system is AI software that uses knowledge stored in a knowledge base to solve problems that would usually require a human expert thus preserving a human expert's knowledge in its knowledge base. They can advise users as well as provide explanations to them about how they reached a particular conclusion or advice. Knowledge Engineering is the term used to define the process of building an Expert System and its practitioners are called Knowledge Engineers. The primary role of a knowledge engineer is to make sure that the computer possesses all the knowledge required to solve a problem. The knowledge engineer must choose one or more forms in which to represent the required knowledge as a symbolic pattern in the memory of the computer.

The data in the knowledge base is added by humans that are expert in a particular domain and this software is used by a non-expert user to acquire some information. It is widely used in many areas such as medical diagnosis, accounting.

Problem:
An expert system that helps to diagnose the diseases of a patient.

Objective:

An expert system that helps to diagnose the diseases of a patient by asking questions about any symptoms

Methodology:

In this project an expert system is built that helps to diagnose the diseases of a patient by asking questions about any symptoms. Will Also display full diagnosis and cure for common diseases.

Using expert system shells for the development of Expert systems. These shells are empty Expert System knowledge bases, which have the structure and interface already written for us; all we need to do is to add the knowledge into the KB. Instead of writing an Expert System from scratch, for this problem we will be using a small Expert System shell to design and implement an expert system.

We used PyKnow Framework for building the concerned expert system.

Cure and Diagnosis is written down in Treatment/markdown files and converted to html with pandoc conversion tool.

Motivation:

Many rural areas in India have extremely limited access to medical advice. People travel long distances to clinics, or medical facilities and there is a shortage of medical experts in most of these facilities. This results in slow service, and patients end up waiting long hours without receiving any attention. Hence, medical expert systems can play a significant role in such cases where medical experts are not readily available. A Diagnosis Expert System can help a great deal in identifying those diseases and describing methods of treatment to be carried out.

Design a knowledge-based expert system that aims to provide the patients with medical advice and basic knowledge on various diseases. It should consider various symptoms and signs like chest pain, cough, fainting, fatigue, headache, back pain, sunken eyes, low body temperature, restlessness, sore throat, fever etc. along with its severity status and provide the patients with medical advice.

Getting Started

To get a local copy up and running follow these simple steps.

Dependencies

This following list of dependencies are present/used in the project

- experta

Installation

1. Clone the Medical-Expert-System

2.Install the experta package with pip (if not installed

pip install experta

3.Run the program

python expert.py

Project Layout

Root directory

```
├── ExpertSystemDesign # files with design of expert system
├── Treatment # files containing details of all diseases
│   ├── html
│   │   ├── Anemia.html
│   │   ├── ...
│   │   └── Tuberculosis.html
│   └── markdown
│       ├── Anemia.md
│       ├── ...
│       └── Tuberculosis.md
├── expert.py # main application
├── LICENSE
├── README.md
└── symptoms.ods # spreadsheet with diseases and symptoms
```

CODE:

```
!pip install experta
from experta import *

class Greetings(KnowledgeEngine):


    def __init__(self, symptom_map, if_not_matched, get_treatments, get_details):
        self.symptom_map = symptom_map
        self.if_not_matched = if_not_matched
        self.get_details = get_details
        self.get_treatments = get_treatments
        KnowledgeEngine.__init__(self)


    #code giving instructions on how to use the Expert System
    @DefFacts()
    def _initial_action(self):
        print("")
        print("This is a knowledge based bot to diagnose diseases")
        print("")
        print("Do you feel any of the following symptoms?")
        print("Reply high or low or no")
        print("")
        yield Fact(action="find_disease")


    #taking various input from user
    @Rule(Fact(action="find_disease"), NOT(Fact(headache=W())), salience=4)
    def symptom_0(self):
        self.declare(Fact(headache=input("headache: ")))


    @Rule(Fact(action="find_disease"), NOT(Fact(back_pain=W())), salience=1)
    def symptom_1(self):
        self.declare(Fact(back_pain=input("back pain: ")))


    @Rule(Fact(action="find_disease"), NOT(Fact(chest_pain=W())), salience=1)
    def symptom_2(self):
        self.declare(Fact(chest_pain=input("chest pain: ")))


    @Rule(Fact(action="find_disease"), NOT(Fact(cough=W())), salience=3)
    def symptom_3(self):
        self.declare(Fact(cough=input("cough: ")))


    @Rule(Fact(action="find_disease"), NOT(Fact(fainting=W())), salience=1)
    def symptom_4(self):
        self.declare(Fact(fainting=input("fainting: ")))


    @Rule(Fact(action="find_disease"), NOT(Fact(fatigue=W())), salience=1)
    def symptom_5(self):
        self.declare(Fact(fatigue=input("fatigue: ")))


    @Rule(Fact(action="find_disease"), NOT(Fact(sunken_eyes=W())), salience=1)
    def symptom_6(self):
        self.declare(Fact(sunken_eyes=input("sunken eyes: ")))


    @Rule(Fact(action="find_disease"), NOT(Fact(low_body_temp=W())), salience=1)
    def symptom_7(self):
        self.declare(Fact(low_body_temp=input("low body temperature: ")))


    @Rule(Fact(action="find_disease"), NOT(Fact(restlessness=W())), salience=1)
    def symptom_8(self):
        self.declare(Fact(restlessness=input("restlessness: ")))


    @Rule(Fact(action="find_disease"), NOT(Fact(sore_throat=W())), salience=1)
    def symptom_9(self):
        self.declare(Fact(sore_throat=input("sore throat: ")))
```

```
@Rule(Fact(action="find_disease"), NOT(Fact(fever=W())), salience=1)
def symptom_10(self):
    self.declare(Fact(fever=input("fever: ")))


@Rule(Fact(action="find_disease"), NOT(Fact(nausea=W())), salience=1)
def symptom_11(self):
    self.declare(Fact(nausea=input("nausea: ")))


@Rule(Fact(action="find_disease"), NOT(Fact(blurred_vision=W())), salience=1)
def symptom_12(self):
    self.declare(Fact(blurred_vision=input("blurred_vision: ")))


#different rules checking for each disease match
@Rule(
    Fact(action="find_disease"),
    Fact(headache="no"),
    Fact(back_pain="no"),
    Fact(chest_pain="no"),
    Fact(cough="no"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="high"),
    Fact(restlessness="no"),
    Fact(low_body_temp="no"),
    Fact(fever="low"),
    Fact(sunken_eyes="no"),
    Fact(nausea="high"),
    Fact(blurred_vision="no"),
)
def disease_0(self):
    self.declare(Fact(disease="Jaundice"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="no"),
    Fact(back_pain="no"),
    Fact(chest_pain="no"),
    Fact(cough="no"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="no"),
    Fact(restlessness="high"),
    Fact(low_body_temp="no"),
    Fact(fever="no"),
    Fact(sunken_eyes="no"),
    Fact(nausea="no"),
    Fact(blurred_vision="no"),
)
def disease_1(self):
    self.declare(Fact(disease="Alzheimers"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="no"),
    Fact(back_pain="high"),
    Fact(chest_pain="no"),
    Fact(cough="no"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="low"),
    Fact(restlessness="no"),
    Fact(low_body_temp="no"),
    Fact(fever="no"),
    Fact(sunken_eyes="no"),
    Fact(nausea="no"),
    Fact(blurred_vision="no"),
)
```

```
def disease_2(self):
    self.declare(Fact(disease="Arthritis"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="no"),
    Fact(back_pain="no"),
    Fact(chest_pain="high"),
    Fact(cough="low"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="no"),
    Fact(restlessness="no"),
    Fact(low_body_temp="no"),
    Fact(fever="high"),
    Fact(sunken_eyes="no"),
    Fact(nausea="no"),
    Fact(blurred_vision="no"),
)
def disease_3(self):
    self.declare(Fact(disease="Tuberculosis"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="no"),
    Fact(back_pain="no"),
    Fact(chest_pain="high"),
    Fact(cough="high"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="no"),
    Fact(restlessness="low"),
    Fact(low_body_temp="no"),
    Fact(fever="no"),
    Fact(sunken_eyes="no"),
    Fact(nausea="no"),
    Fact(blurred_vision="no"),
)
def disease_4(self):
    self.declare(Fact(disease="Asthma"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="low"),
    Fact(back_pain="no"),
    Fact(chest_pain="no"),
    Fact(cough="high"),
    Fact(fainting="no"),
    Fact(sore_throat="high"),
    Fact(fatigue="no"),
    Fact(restlessness="no"),
    Fact(low_body_temp="no"),
    Fact(fever="low"),
    Fact(sunken_eyes="no"),
    Fact(nausea="no"),
    Fact(blurred_vision="no"),
)
def disease_5(self):
    self.declare(Fact(disease="Sinusitis"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="no"),
    Fact(back_pain="no"),
    Fact(chest_pain="no"),
    Fact(cough="no"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="low"),
    Fact(restlessness="no"),
```

```
    Fact(low_body_temp="no"),
    Fact(fever="no"),
    Fact(sunken_eyes="no"),
    Fact(nausea="no"),
    Fact(blurred_vision="no"),
)
def disease_6(self):
    self.declare(Fact(disease="Epilepsy"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="no"),
    Fact(back_pain="no"),
    Fact(chest_pain="high"),
    Fact(cough="no"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="no"),
    Fact(restlessness="no"),
    Fact(low_body_temp="no"),
    Fact(fever="no"),
    Fact(sunken_eyes="no"),
    Fact(nausea="high"),
    Fact(blurred_vision="no"),
)
def disease_7(self):
    self.declare(Fact(disease="Heart Disease"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="no"),
    Fact(back_pain="no"),
    Fact(chest_pain="no"),
    Fact(cough="no"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="high"),
    Fact(restlessness="no"),
    Fact(low_body_temp="no"),
    Fact(fever="no"),
    Fact(sunken_eyes="no"),
    Fact(nausea="low"),
    Fact(blurred_vision="low"),
)
def disease_8(self):
    self.declare(Fact(disease="Diabetes"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="low"),
    Fact(back_pain="no"),
    Fact(chest_pain="no"),
    Fact(cough="no"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="no"),
    Fact(restlessness="no"),
    Fact(low_body_temp="no"),
    Fact(fever="no"),
    Fact(sunken_eyes="no"),
    Fact(nausea="high"),
    Fact(blurred_vision="low"),
)
def disease_9(self):
    self.declare(Fact(disease="Glaucoma"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="no"),
    Fact(back_pain="no"),
```

```
    Fact(chest_pain="no"),
    Fact(cough="no"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="high"),
    Fact(restlessness="no"),
    Fact(low_body_temp="no"),
    Fact(fever="no"),
    Fact(sunken_eyes="no"),
    Fact(nausea="low"),
    Fact(blurred_vision="no"),
)
def disease_10(self):
    self.declare(Fact(disease="Hyperthyroidism"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="high"),
    Fact(back_pain="no"),
    Fact(chest_pain="no"),
    Fact(cough="no"),
    Fact(fainting="no"),
    Fact(sore_throat="no"),
    Fact(fatigue="no"),
    Fact(restlessness="no"),
    Fact(low_body_temp="no"),
    Fact(fever="high"),
    Fact(sunken_eyes="no"),
    Fact(nausea="high"),
    Fact(blurred_vision="no"),
)
def disease_11(self):
    self.declare(Fact(disease="Heat Stroke"))


@Rule(
    Fact(action="find_disease"),
    Fact(headache="no"),
    Fact(back_pain="no"),
    Fact(chest_pain="no"),
    Fact(cough="no"),
    Fact(fainting="yes"),
    Fact(sore_throat="no"),
    Fact(fatigue="no"),
    Fact(restlessness="no"),
    Fact(low_body_temp="high"),
    Fact(fever="no"),
    Fact(sunken_eyes="no"),
    Fact(nausea="no"),
    Fact(blurred_vision="no"),
)
def disease_12(self):
    self.declare(Fact(disease="Hypothermia"))

@Rule(
    Fact(action="find_disease"),
    Fact(headache="high"),
    Fact(back_pain="no"),
    Fact(chest_pain="high"),
    Fact(cough="high"),
    Fact(fainting="no"),
    Fact(sore_throat="high"),
    Fact(fatigue="high"),
    Fact(restlessness="no"),
    Fact(low_body_temp="no"),
    Fact(fever="high"),
    Fact(sunken_eyes="no"),
    Fact(nausea="no"),
    Fact(blurred_vision="no"),
)
def disease_13(self):
    self.declare(Fact(disease="Coronavirus"))
```

```
#when the user's input doesn't match any disease in the knowledge base
@Rule(Fact(action="find_disease"), Fact(disease=MATCH.disease), salience=-998)
def disease(self, disease):
    print("")
    id_disease = disease
    disease_details = self.get_details(id_disease)
    treatments = self.get_treatments(id_disease)
    print("")
    print("Your symptoms match %s\n" % (id_disease))
    print("A short description of the disease is given below :\n")
    print(disease_details + "\n")
    print(
        "The common medications and procedures suggested by other real doctors are: \n"
    )
    print(treatments + "\n")


@Rule(
    Fact(action="find_disease"),
    Fact(headache=MATCH.headache),
    Fact(back_pain=MATCH.back_pain),
    Fact(chest_pain=MATCH.chest_pain),
    Fact(cough=MATCH.cough),
    Fact(fainting=MATCH.fainting),
    Fact(sore_throat=MATCH.sore_throat),
    Fact(fatigue=MATCH.fatigue),
    Fact(low_body_temp=MATCH.low_body_temp),
    Fact(restlessness=MATCH.restlessness),
    Fact(fever=MATCH.fever),
    Fact(sunken_eyes=MATCH.sunken_eyes),
    Fact(nausea=MATCH.nausea),
    Fact(blurred_vision=MATCH.blurred_vision),
    NOT(Fact(disease=MATCH.disease)),
    salience=-999
)
def not_matched(
    self,
    headache,
    back_pain,
    chest_pain,
    cough,
    fainting,
    sore_throat,
    fatigue,
    restlessness,
    low_body_temp,
    fever,
    sunken_eyes,
    nausea,
    blurred_vision,
):
    print("\nThe bot did not find any diseases that match your exact symptoms.")
    lis = [
        headache,
        back_pain,
        chest_pain,
        cough,
        fainting,
        sore_throat,
        fatigue,
        restlessness,
        low_body_temp,
        fever,
        sunken_eyes,
        nausea,
        blurred_vision,
    ]
    max_count = 0
    max_disease = ""
    for key, val in self.symptom_map.items():
        count = 0
```

```
            temp_list = eval(key)
            for j in range(0, len(lis)):
                if temp_list[j] == lis[j] and (lis[j] == "high" or lis[j] == "low" or lis[j] == "yes"):
                    count = count + 1
            if count > max_count:
                max_count = count
                max_disease = val
        if max_disease != "":
            self.if_not_matched(max_disease)
!pip install greetings
from greetings import greetings
diseases_list = []
diseases_symptoms = []
symptom_map = {}
d_desc_map = {}
d_treatment_map = {}

#loads the knowledge from .txt files into variables to allow the code to use it
def preprocess():
    #global diseases_list, diseases_symptoms, symptom_map, d_desc_map, d_treatment_map
    diseases = open("/content/drive/MyDrive/diseases.txt")
    diseases_t = diseases.read()
    diseases_list = diseases_t.split("\n")
    diseases.close()


    for disease in diseases_list:
        disease_s_file = open("/content/drive/MyDrive/Disease symptoms/"+disease+".txt")
        disease_s_data = disease_s_file.read()
        s_list = disease_s_data.split("\n")
        diseases_symptoms.append(s_list)
        symptom_map[str(s_list)] = disease
        disease_s_file.close()


        disease_s_file = open("/content/drive/MyDrive/Disease descriptions/"+disease+".txt")
        disease_s_data = disease_s_file.read()
        d_desc_map[disease] = disease_s_data
        disease_s_file.close()


        disease_s_file = open("/content/drive/MyDrive/Disease treatments/"+disease+".txt")
        disease_s_data = disease_s_file.read()
        d_treatment_map[disease] = disease_s_data
        disease_s_file.close()



def identify_disease(*arguments):
    symptom_list = []
    for symptom in arguments:
        symptom_list.append(symptom)


    return symptom_map[str(symptom_list)]



def get_details(disease):
    return d_desc_map[disease]



def get_treatments(disease):
    return d_treatment_map[disease]



def if_not_matched(disease):
    print("")
    id_disease = disease
    disease_details = get_details(id_disease)
    treatments = get_treatments(id_disease)
    print("")
```

```
    print("The most probable disease that you have is %s\n" % (id_disease))
    print("A short description of the disease is given below :\n")
    print(disease_details + "\n")
    print(
        "The common medications and procedures suggested by other real doctors are: \n"
    )
    print(treatments + "\n")


#driver function
if __name__ == "__main__":
    preprocess()
    #creating class object
    engine = Greetings(symptom_map, if_not_matched, get_treatments, get_details)
    #loop to keep running the code until user says no when asked for another diagnosis
    while 1:
        engine.reset()
        engine.run()
        print("Would you like to diagnose some other symptoms?\n Reply yes or no")
        if input() == "no":
            exit()
```

OUTPUT

```
Requirement already satisfied: greetings in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: Click>=6.0 in /usr/local/lib/python3.7/dist-packages

This is a knowledge based bot to diagnose diseases

Do you feel any of the following symptoms?
Reply high or low or no

headache: high
cough: no
nausea: yes
fatigue: yes
blurred_vision: no
back pain: yes
fever: yes
sore throat: no
low body temperature: high
chest pain: yes
sunken eyes: yes
restlessness: yes
fainting: no

The bot did not find any diseases that match your exact symptoms.


The most probable disease that you have is Heat Stroke

A short description of the disease is given below :

Heatstroke generally occurs when an individual has been too hot for too long, whethe

Also known as sunstroke, heatstroke is a serious condition and must be considered an

If the body temperature rises above 40° Celsius, and the body loses the ability to c

If left untreated, damage to internal organs can occur. The longer it is left, the m

Heatstroke can be brought on by physical exertion in hot conditions, or simply by be

Treating heatstroke centers around bringing body temperature down.

Heatstroke can occur as a result of:

Exposure to a hot environment.

In a type of heatstroke, called nonexertional (classic) heatstroke, being in a hot e

It occurs most often in older adults and in people with chronic illness.

Strenuous activity. Exertional heatstroke is caused by an increase in core body temp

Anyone exercising or working in hot weather can get exertional heatstroke, but it's

The common medications and procedures suggested by other real doctors are:
```

Conclusion

In this report, an expert's system's development has been presented. It was emphasized on the basic steps, consequently taken during the project's development course as a particular attention was turned to the basic operative functions performed upon the data. The report's content comprises the whole task solution, starting from the programming environments have been selected, the applications analyse and construction, and finishing with the code-implementation and test-samples.

References.

http://www.geeksforgeeks.org/expert-systems/
http://www.ijesrt.com/