**Implementation**

Here's a detailed implementation of the **Interactive Health Diagnosis System** using Python, incorporating `tkinter` for the GUI and `csv` for data handling. The system will allow users to input multiple symptoms, match those symptoms with a pre-defined CSV file, and provide potential diagnoses, treatments, and causes.

**1. Setup the CSV File**

First, create a CSV file named **`health\_data.csv`** in the same directory as the Python script. The file should contain information about symptoms, diagnoses, treatments, and causes.

symptom,diagnosis,treatment,causes

fever,Infection,Rest; Hydration; Antipyretics,Viral infection; Bacterial infection

cough,Common Cold,Rest; Fluids; Cough suppressant,Viral infection; Allergies; Smoking

headache,Migraine,Pain relievers; Rest; Dark room,Stress; Hormonal changes; Food triggers

fatigue,Anemia,Iron supplements; Diet change; Rest,Iron deficiency; Chronic disease

nausea,Food Poisoning,Hydration; Rest; Anti-nausea medication,Contaminated food; Bacterial infection

**2. Python Code Implementation**

Here is the complete Python implementation for the project:

import tkinter as tk

from tkinter import messagebox

import csv

# Load CSV data into a dictionary

def load\_health\_data(file\_path):

health\_data = {}

try:

with open(file\_path, mode='r') as file:

reader = csv.DictReader(file)

for row in reader:

symptom = row['symptom'].lower().strip()

diagnosis = row['diagnosis'].strip()

treatment = row['treatment'].strip()

causes = row['causes'].strip()

health\_data[symptom] = {

'diagnosis': diagnosis,

'treatment': treatment,

'causes': causes

}

except FileNotFoundError:

print(f"Error: {file\_path} not found.")

return health\_data

# Symptom matching function

def match\_symptoms(health\_data, symptoms\_input):

matched\_results = []

symptoms = [s.strip().lower() for s in symptoms\_input.split(',')]

for symptom in symptoms:

if symptom in health\_data:

result = health\_data[symptom]

matched\_results.append({

'symptom': symptom,

'diagnosis': result['diagnosis'],

'treatment': result['treatment'],

'causes': result['causes']

})

else:

matched\_results.append({

'symptom': symptom,

'diagnosis': 'Unknown',

'treatment': 'No information available',

'causes': 'No information available'

})

return matched\_results

# Display results in the GUI

def display\_results(results):

output\_textbox.config(state='normal')

output\_textbox.delete('1.0', tk.END) # Clear previous results

for result in results:

output\_textbox.insert(tk.END, f"Symptom: {result['symptom'].capitalize()}\n")

output\_textbox.insert(tk.END, f"Diagnosis: {result['diagnosis']}\n")

output\_textbox.insert(tk.END, f"Treatment: {result['treatment']}\n")

output\_textbox.insert(tk.END, f"Causes: {result['causes']}\n")

output\_textbox.insert(tk.END, "-" \* 40 + "\n")

output\_textbox.config(state='disabled')

# Function to handle symptom submission

def submit\_symptoms():

symptoms\_input = input\_field.get().strip()

if symptoms\_input:

results = match\_symptoms(health\_data, symptoms\_input)

display\_results(results)

else:

messagebox.showerror("Input Error", "Please enter at least one symptom.")

# Function to clear the input field and output box

def clear\_input\_output():

input\_field.delete(0, tk.END)

output\_textbox.config(state='normal')

output\_textbox.delete('1.0', tk.END)

output\_textbox.config(state='disabled')

# Main application window (GUI)

root = tk.Tk()

root.title("Interactive Health Diagnosis System")

root.geometry("600x500")

root.configure(bg="#E3F2FD")

# GUI title label

title\_label = tk.Label(root, text="Health Diagnosis System", font=("Arial", 18, "bold"), bg="#42A5F5", fg="white", pady=10)

title\_label.pack(fill=tk.X)

# Symptom input field

input\_label = tk.Label(root, text="Enter symptoms (comma-separated):", font=("Arial", 12), bg="#E3F2FD")

input\_label.pack(pady=10)

input\_field = tk.Entry(root, font=("Arial", 14), width=50)

input\_field.pack(pady=10)

# Submit and Clear buttons

button\_frame = tk.Frame(root, bg="#E3F2FD")

button\_frame.pack(pady=10)

submit\_button = tk.Button(button\_frame, text="Submit", font=("Arial", 12), command=submit\_symptoms, bg="#4CAF50", fg="white", width=10)

submit\_button.grid(row=0, column=0, padx=10)

clear\_button = tk.Button(button\_frame, text="Clear", font=("Arial", 12), command=clear\_input\_output, bg="#F44336", fg="white", width=10)

clear\_button.grid(row=0, column=1, padx=10)

# Output display box (text box)

output\_textbox = tk.Text(root, font=("Arial", 12), height=12, width=70, state='disabled', bg="#F1F8E9")

output\_textbox.pack(pady=10)

# Load health data

health\_data = load\_health\_data('health\_data.csv')

# Start the GUI event loop

root.mainloop()

**3. Running the Application**

1. Install Python: Make sure you have Python installed on your system.

2. Install Required Modules: The required modules (`tkinter`, `csv`) are built-in with Python.

3. Run the Script: Place the script and the `health\_data.csv` file in the same directory and run the Python script. You will see the GUI window appear.

This implementation provides a simple, user-friendly interface that allows users to input their symptoms and get preliminary health diagnoses, treatments, and probable causes based on a CSV file. The design can be expanded by adding more data to the CSV file or integrating more advanced AI techniques for symptom matching.