DISEASE ANALYSIS USING PYTHON

PROJECT REPORT

Submitted by

JEEBAN SWARUP JAGDEV

[EC2332251010140]

Under the Guidance of

Umamaheswari. Km

(Assistant Professor, Directorate of Online Education)

in partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

DIRECTORATE OF ONLINE EDUCATION
SRM INSTITUTE OF SCIENCE AND
TECHNOLOGY KATTANKULATHUR- 603 203
JAN 2023

DIRECTORATE OF ONLINE EDUCATION

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

This Mini Project Work Report titled **"Disease analysis using python"** of **JEEBAN SWARUP JAGDEV [EC2332251010140]**, who carried out the Mini Project Work under my supervision along with the company mentor. Certified further, that to the best of my knowledge the work reported herein does not form any other internship report or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to Director DOE, SRM Institute of Science and Technology, Prof. **Dr Manoranjan Pon Ram,** for his invaluable support. SRM Institute of Science and Technology, for her support throughout the project work. We want to convey our thanks to Programme Coordinator **Dr. G. Babu**, Directorate of online Education, SRM Institute of Science and Technology, for their inputs during the project reviews and support. Our inexpressible respect and thanks to my guide, **Umamaheswari. Km**, Assistant Professor & Programme Coordinator Directorate of online Education,, SRM Institute of Science and Technology, for providing me with an opportunity to pursue my project under his mentorship. He provided me with the freedom and support to explore the research topics of my interest. His passion for solving problems and making a difference in the world has always been inspiring. We sincerely thank the Directorate of online Education, staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Jeeban Swarup Jagdev

TABLE OF CONTENTS

1. ABSTRACT

This project involves the development of an interactive health diagnosis system using Python, aimed at providing users with a preliminary understanding of potential health conditions based on their symptoms. The system utilizes a graphical user interface (GUI) built with the `tkinter` library to offer a user-friendly experience, allowing individuals to input multiple symptoms at once and receive instant feedback in the form of possible diagnoses, treatments, and underlying causes.

The program is designed to be both educational and informative, helping users recognize common health issues by referencing a comprehensive CSV file containing a broad range of symptoms and associated medical data. The CSV file serves as the knowledge base for the application, enabling it to cross-reference user input with stored health information, including diagnoses, recommended treatments, and probable causes for each symptom.

Key features of the system include an easy-to-navigate GUI, real-time processing of multiple symptoms, and interactive prompts that guide users through the diagnosis process. The GUI is enhanced with a colorful and intuitive design to ensure accessibility and engagement, making it suitable for users with varying levels of computer literacy.

The expanded dataset in the CSV file covers a wide array of common symptoms, offering a more accurate and comprehensive preliminary diagnosis. While the system is not intended to replace professional medical advice, it serves as a useful tool for initial self-assessment, helping users make informed decisions about when to seek medical attention.

Overall, this project demonstrates how simple AI techniques and structured health data can be combined to create a practical, interactive application that empowers users to better understand their health conditions.

2. SYSTEM ANALYSIS

System analysis involves understanding the problem the project aims to solve, identifying the requirements, and proposing a solution that fulfills those needs. This section delves into the problem description, objectives, and proposed solution for the **Interactive Health Diagnosis System Using AI**.

2.1 Problem Description

Health is a critical aspect of human life, but access to timely and accurate medical consultation can sometimes be limited due to various factors such as geographic location, cost, or unavailability of healthcare professionals. Often, individuals experience symptoms but are unsure whether they should seek immediate medical care or treat the symptoms with home remedies. Many users turn to the internet for self-diagnosis, which may provide inconsistent or incorrect information.

Furthermore, traditional health diagnostic methods rely on physical consultations and diagnostic tools, which may not always be immediately accessible. This project aims to provide a preliminary solution by offering users an **interactive, AI-driven health diagnosis system** that gives them probable diagnoses, treatments, and causes based on their symptoms. This helps bridge the gap between experiencing symptoms and seeking proper medical advice.

2.2 Proposed Solution

The solution involves developing a Python-based **Interactive Health Diagnosis System** that can:

- Accept multiple symptoms from users through a graphical user interface (GUI).
- Match the symptoms with a pre-existing health dataset stored in a CSV file, which contains information on symptoms, probable diagnoses, treatments, and causes.
- Return potential health conditions, suggested treatments, and possible causes for the symptoms entered by the user.
- Offer a simple and efficient user experience with an easy-to-use GUI powered by Python's Tkinter library.
- Be flexible and maintainable for future updates, allowing the dataset to be expanded with more health conditions and medical data as needed.

2.3 Requirements Analysis

2.3.1. Functional Requirements

1. **Symptom Input**: Users should be able to input one or more symptoms into the system through the GUI.
2. **Diagnosis Matching**: The system should match the input symptoms with the symptom-diagnosis data stored in the CSV file.

3. **Treatment Suggestions**: The system should provide treatment suggestions for the diagnosis made.
4. **Probable Cause**: For each diagnosis, the system should return a probable cause (e.g., bacterial or viral infection).
5. **User Interface**: The GUI should be easy to navigate, allowing users to enter their symptoms and view the results clearly.
6. **Error Handling**: The system should handle cases where no matching symptoms are found and provide appropriate feedback to the user.

2.3.2. Non-Functional Requirements

1. **Performance**: The system should provide quick responses, delivering results within seconds.
2. **Usability**: The system should be intuitive enough for users of various age groups to use without additional guidance.
3. **Scalability**: The system should be capable of handling larger datasets in the future as new health conditions are added.
4. **Maintainability**: The system should allow easy updates to the health dataset and the diagnosis logic.
5. **Security**: Although this system does not deal with personal data, it must handle user input securely and provide accurate and verified health information.

3. REQUIREMENTS SPECIFICATIONS

3.1 Hardware Requirements

- **Processor**: 1 GHz or faster processor.
- **Memory**: Minimum of 2 GB RAM.
- **Storage**: At least 100 MB of free storage for the dataset and application.
- **Display**: A monitor to display the graphical interface.

3.2. Software Requirements

- **Operating System**: Windows, macOS, or Linux.
- **Programming Language**: Python 3.x.
- **Libraries/Modules**:
    - Tkinter for GUI development.
    - csv for handling the health data file.
    - unittest for testing.
    - webbrowser to fetch the web pages
- **Data Source**: CSV file containing health conditions, symptoms, diagnoses, treatments, and probable causes.

4. TECHNOLOGY

This project uses several technologies to create an interactive health diagnosis system that is functional, user-friendly, and efficient. Here's a breakdown of the technologies employed:

4.1. Python Programming Language

- **Overview**: Python is a high-level, versatile programming language that is known for its simplicity and readability. It has a rich ecosystem of libraries that support various functionalities, making it ideal for rapid development.
- **Usage in Project**: Python is used to implement the entire backend logic for this system. Its libraries handle data management, user interface design, and testing.

4.2. CSV (Comma-Separated Values) Format

- **Overview**: CSV is a lightweight, tabular file format used for storing data. It allows data to be organized in rows and columns, making it easy to handle and process.
- **Usage in Project**: In this project, CSV is used to store the health dataset, which includes information on symptoms, probable diagnoses, treatments, and causes.

4.3. Tkinter (Python's Standard GUI Library)

- **Overview**: Tkinter is Python's built-in library for creating graphical user interfaces. It is lightweight, easy to use, and works across multiple platforms.
- **Usage in Project**: Tkinter is used to create an interactive, user-friendly interface where users can enter symptoms and view diagnoses. The interface includes input fields, buttons, and display areas for showing diagnosis results.

4.4. Unittest Library (Python's Standard Testing Library)

- **Overview**: unittest is Python's built-in library for testing. It offers a framework to write and execute unit tests, ensuring each part of the system works as expected.
- **Usage in Project**: The unittest library is used to create test cases that validate the functionality of key components, such as the diagnosis matching and symptom input modules.

4.5. webbrowser

- **Overview**: To open external links (e.g., video tutorials and articles).

4.6. Data Storage and Management

For this project, a simple CSV file acts as the data storage solution. This file contains rows of data corresponding to different symptoms, diagnoses, treatments, and causes. The CSV format is highly compatible with Python and can be processed with minimal overhead, which is ideal for a lightweight application focused on delivering quick diagnostic results.

CSV Files:

- **health_data.csv** for storing symptoms, diagnoses, treatments, and causes.
- **patient_history.csv** for saving patient registration details.
- **search_log.csv** for keeping a log of symptom searches and diagnoses.

4.7. Basic AI Concepts

While this system uses rule-based matching rather than advanced machine learning, the AI component lies in:

- **Rule-Based Matching**: The system uses pattern matching based on pre-defined data entries in the CSV file. When symptoms are entered, the system cross-references them with the dataset to find corresponding diagnoses, treatments, and probable causes.
- **Decision-Making Logic**: The application makes decisions based on symptom entries to suggest probable health conditions, effectively mimicking basic AI-driven diagnosis logic.

4.8. Software Development Tools

Integrated Development Environment (IDE)

- **Examples**: PyCharm, VS Code, or Jupyter Notebook.
- **Usage in Project**: IDEs simplify coding, debugging, and testing in Python, offering a streamlined development process with features like syntax highlighting, code suggestions, and error checking.

These technologies collectively make the **Interactive Health Diagnosis System Using AI** feasible by offering a balanced approach to data handling, user interface design, testing, and basic diagnostic logic. Each tool contributes to a reliable, accessible, and efficient user experience, laying the foundation for future system enhancements.

5. CODING

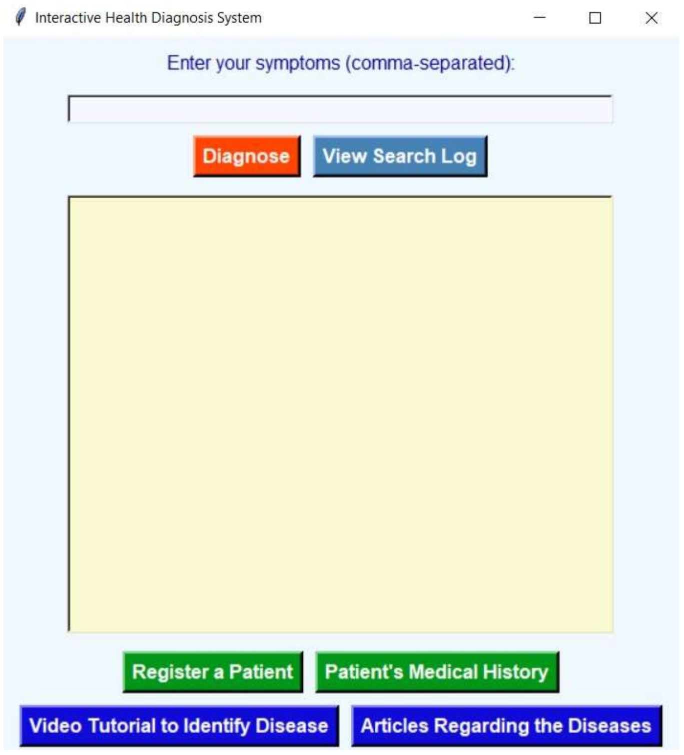
```
import csv
import tkinter as tk
from tkinter import messagebox, Toplevel, scrolledtext, StringVar
import webbrowser
# Function to load health data from a CSV file
def load_health_data(file_path):
    health_data = {}
    with open(file_path, mode='r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            symptom = row['symptom'].strip().lower()
            health_data[symptom] = {
                "diagnosis": row['diagnosis'].strip(),
                "treatment": row['treatment'].strip(),
                "causes": [cause.strip() for cause in row['causes'].split(';')]
            }
    return health_data
# Function to diagnose based on multiple symptoms
def health_diagnosis(symptoms, health_data):
    symptoms = [symptom.strip().lower() for symptom in symptoms.split(',')]
    diagnosis_result = ""
    for symptom in symptoms:
        diagnosis_info = health_data.get(symptom, None)
        if diagnosis_info:
            diagnosis_result += f"\nSymptom: {symptom.capitalize()}\n"
            diagnosis_result += f"  Diagnosis: {diagnosis_info['diagnosis']}\n"
            diagnosis_result += f"  Treatment: {diagnosis_info['treatment']}\n"
            diagnosis_result += f"  Probable Causes: {', '.join(diagnosis_info['causes'])}\n"
        else:
            diagnosis_result += f"\nSymptom: {symptom.capitalize()}\n"
            diagnosis_result += "  Diagnosis: Symptom not recognized. Please consult a healthcare professional.\n"
    return diagnosis_result
# Function to save patient data to a CSV file
def save_patient_data(data):
    with open('patient_history.csv', mode='a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(data)
# Function to handle diagnosis button click and log search
def on_diagnose_click():
    symptoms = entry.get()
    if not symptoms:
        messagebox.showwarning("Input Error", "Please enter at least one symptom.")
        return
    result = health_diagnosis(symptoms, health_data)
    # Save to search log
    with open('search_log.csv', 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([symptoms, result])
    text_result.config(state=tk.NORMAL)
    text_result.delete(1.0, tk.END)
    text_result.insert(tk.END, result)
    text_result.config(state=tk.DISABLED)
# Function to open patient registration form
def open_register_patient():
    register_window = Toplevel(root)
    register_window.title("Register Patient")
    fields = {
        "Patient Name": StringVar(),
        "Age": StringVar(),
        "Sex": StringVar(),
        "Height": StringVar(),
        "Previously Diagnosed Diseases": StringVar(),
        "Currently Diagnosed Disease": StringVar(),
        "Symptoms": StringVar(),
        "Family History": StringVar()
    }
    for idx, (label_text, var) in enumerate(fields.items()):
        label = tk.Label(register_window, text=label_text, font=("Arial", 10))
        label.grid(row=idx, column=0, padx=5, pady=5)
        entry = tk.Entry(register_window, textvariable=var, width=40)
        entry.grid(row=idx, column=1, padx=5, pady=5)
    def on_save_data():
        data = [var.get() for var in fields.values()]
        if not all(data):
            messagebox.showwarning("Incomplete Data", "Please fill out all fields before
saving.")
        else:
            save_patient_data(data)
            messagebox.showinfo("Data Saved", "Patient data saved successfully.")
            for var in fields.values():
                var.set("")
    def on_clear_all():
        for var in fields.values():
            var.set("")
    save_button = tk.Button(register_window, text="Save Data", command=on_save_data,
bg="#008000", fg="white", font=("Arial", 10, "bold"))
    save_button.grid(row=len(fields), column=0, padx=5, pady=10)
    clear_button = tk.Button(register_window, text="Clear All", command=on_clear_all,
bg="#978803", fg="white", font=("Arial", 10, "bold"))
    clear_button.grid(row=len(fields), column=1, padx=5, pady=10)
```

```
# Function to search and display patient's medical history
def search_patient_history():
    search_window = Toplevel(root)
    search_window.title("Search Patient History")
    search_label = tk.Label(search_window, text="Enter Patient Name:", font=("Arial", 10))
    search_label.grid(row=0, column=0, padx=5, pady=5)
    search_entry = tk.Entry(search_window, width=40)
    search_entry.grid(row=0, column=1, padx=5, pady=5)
    result_text = scrolledtext.ScrolledText(search_window, width=80, height=20,
font=("Arial", 10))
    result_text.grid(row=2, column=0, columnspan=2, padx=10, pady=10)
    def on_search():
        name = search_entry.get().strip().lower()
        if not name:
            messagebox.showwarning("Input Error", "Please enter a patient name to search.")
            return
        found = False
        result_text.config(state=tk.NORMAL)
        result_text.delete(1.0, tk.END)
        try:
            with open('patient_history.csv', mode='r') as file:
                reader = csv.reader(file)
                for row in reader:
                    if row[0].strip().lower() == name:
                        result_text.insert(tk.END, f"Patient Name: {row[0]}\n")
                        result_text.insert(tk.END, f"Age: {row[1]}\n")
                        result_text.insert(tk.END, f"Sex: {row[2]}\n")
                        result_text.insert(tk.END, f"Height: {row[3]}\n")
                        result_text.insert(tk.END, f"Previously Diagnosed Diseases: {row[4]}\n")
                        result_text.insert(tk.END, f"Currently Diagnosed Disease: {row[5]}\n")
                        result_text.insert(tk.END, f"Symptoms: {row[6]}\n")
                result_text.insert(tk.END, f"Family History: {row[7]}\n\n")
                found = True
                break
        if not found:
            result_text.insert(tk.END, "No records found for this patient.")
    except FileNotFoundError:
        result_text.insert(tk.END, "No patient history file found.")
    result_text.config(state=tk.DISABLED)
  search_button = tk.Button(search_window, text="Search", command=on_search,
bg="#4682b4", fg="white", font=("Arial", 10, "bold"))
  search_button.grid(row=1, column=0, columnspan=2, pady=10)
# Function to open video tutorial link
def open_video_tutorial():
  webbrowser.open("https://www.youtube.com/watch?app=desktop&v=Plse2FOkV4Q")
# Function to open article link
def open_article_link():
webbrowser.open("https://ncvbdc.mohfw.gov.in/index4.php?lang=1&level=0&linkid=407&l
id=3683")
# Function to view search log
def view_search_log():
  log_window = Toplevel(root)
  log_window.title("Search Log")
  log_text = scrolledtext.ScrolledText(log_window, width=80, height=100, font=("Arial",
10))
  log_text.pack(padx=10, pady=10)
  try:
    with open('search_log.csv', 'r') as file:
      reader = csv.reader(file)
      for row in reader:
        log_text.insert(tk.END, f"Search: {row[0]}\nResult: {row[1]}\n\n")
  except FileNotFoundError:
    log_text.insert(tk.END, "No search log file found.")
# Load the health data from the CSV file
file_path = 'health_data.csv'
health_data = load_health_data(file_path)
# Create the main application window
root = tk.Tk()
root.title("Interactive Health Diagnosis System")
root.configure(bg="#f0f8ff")
label = tk.Label(root, text="Enter your symptoms (comma-separated):", bg="#f0f8ff",
fg="#00008b", font=("Arial", 12))
label.pack(pady=10)
entry = tk.Entry(root, width=50, font=("Arial", 12), bg="#f8f8ff", fg="#2f4f4f", bd=2)
entry.pack(pady=5)
# Custom colors for each button
button_colors = {
    "Diagnose": "#FF4500",
    "View Search Log": "#4682B4",
    "Register a Patient": "#039719",
    "Patient's Medical History": "#039719",
    "Video Tutorial to Identify Disease": "#110bd8",
    "Articles Regarding the Diseases": "#110bd8"
}
# Function to create hover effect for buttons
def on_enter(e):
    e.widget['background'] = '#2f4f4f'
    e.widget['foreground'] = 'white'
def on_leave(e):
    button_text = e.widget['text']
    e.widget['background'] = button_colors.get(button_text, "#4682b4")
    e.widget['foreground'] = 'white'
# Create button frame for upper buttons
```

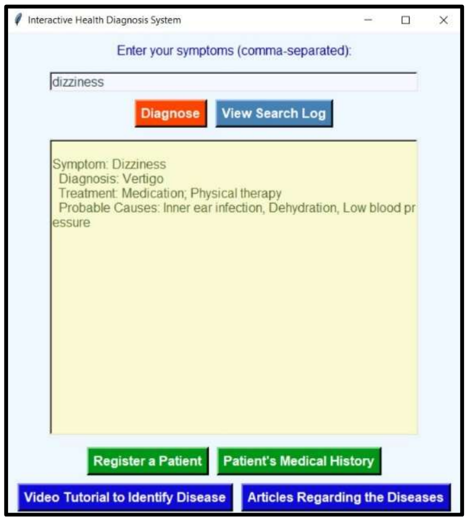
```
frame1 = tk.Frame(root, bg="#f0f8ff")

frame1.pack(pady=5)

# Upper buttons

def create_button(text, command, frame):

    button = tk.Button(frame, text=text, command=command, bg=button_colors[text], 
fg="white", font=("Arial", 12, "bold"), bd=3)

    button.bind("<Enter>", on_enter)

    button.bind("<Leave>", on_leave)

    button.pack(side=tk.LEFT, padx=5)

create_button("Diagnose", on_diagnose_click, frame1)

create_button("View Search Log", view_search_log, frame1)

# Output text area

text_result = tk.Text(root, height=20, width=50, state=tk.DISABLED, font=("Arial", 12), 
bg="#fafad2", fg="#556b2f", bd=2)

text_result.pack(pady=10)

# Lower button frame for additional buttons

frame2 = tk.Frame(root, bg="#f0f8ff")

frame2.pack(pady=5)

frame3 = tk.Frame(root, bg="#f0f8ff")

frame3.pack(pady=5)

# Lower buttons

create_button("Register a Patient", open_register_patient, frame2)

create_button("Patient's Medical History", search_patient_history, frame2)

create_button("Video Tutorial to Identify Disease", open_video_tutorial, frame3)

create_button("Articles Regarding the Diseases", open_article_link, frame3)

for widget in root.winfo_children():

    widget.pack_configure(padx=10)

root.mainloop()
```

5. RESULT

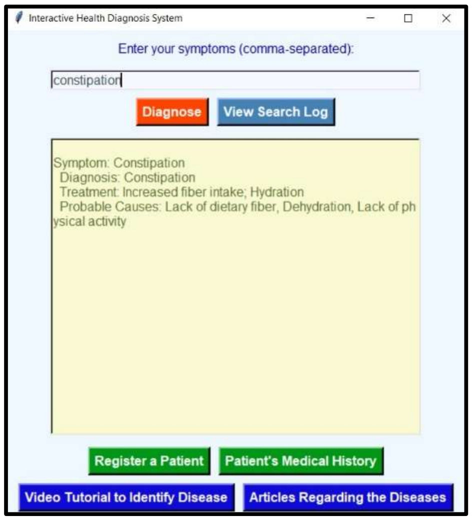
Starting window:


Interactive Health Diagnosis System
Enter your symptoms (comma-separated):
Diagnose
View Search Log
Register a Patient
Patient's Medical History
Video Tutorial to Identify Disease
Articles Regarding the Diseases

Diagnose:

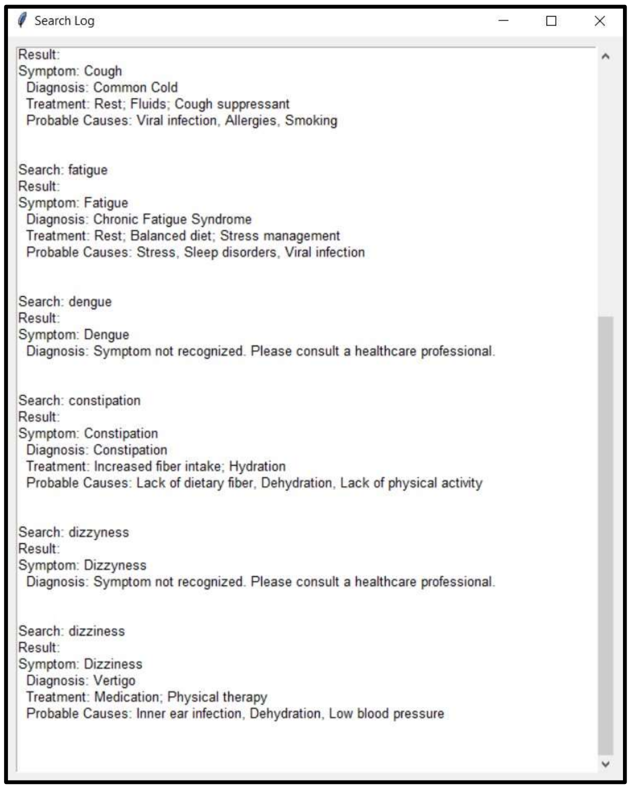

Interactive Health Diagnosis System
Enter your symptoms (comma-separated):
dizziness
Diagnose
View Search Log
Register a Patient
Patient's Medical History
Video Tutorial to Identify Disease
Articles Regarding the Diseases

```
Symptom: Dizziness
  Diagnosis: Vertigo
  Treatment: Medication; Physical therapy
  Probable Causes: Inner ear infection, Dehydration, Low blood pr
essure
```


Interactive Health Diagnosis System
Enter your symptoms (comma-separated):
constipation
Diagnose
View Search Log
Register a Patient
Patient's Medical History
Video Tutorial to Identify Disease
Articles Regarding the Diseases

```
Symptom: Constipation
  Diagnosis: Constipation
  Treatment: Increased fiber intake; Hydration
  Probable Causes: Lack of dietary fiber, Dehydration, Lack of ph
ysical activity
```

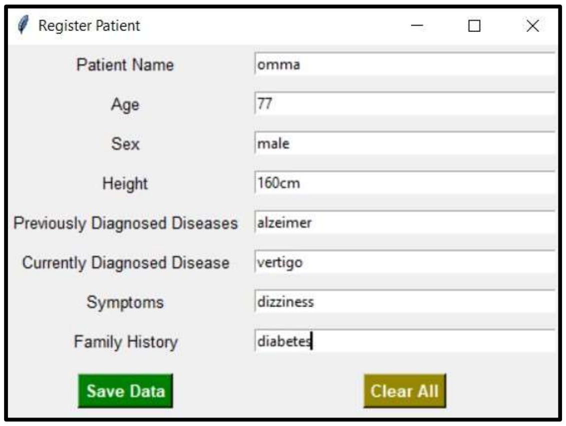
View Search Log:



Search Log

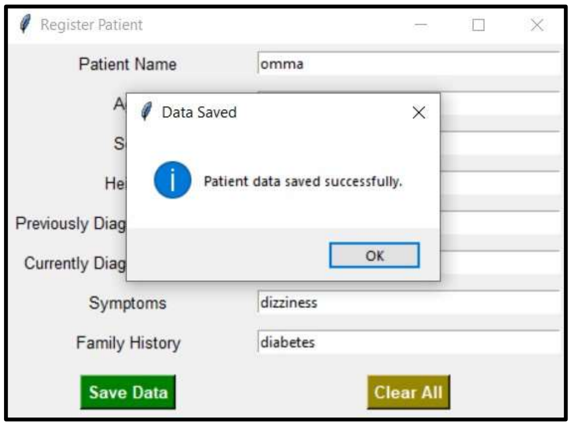
```
Result:
Symptom: Cough
  Diagnosis: Common Cold
  Treatment: Rest; Fluids; Cough suppressant
  Probable Causes: Viral infection, Allergies, Smoking


Search: fatigue
Result:
Symptom: Fatigue
  Diagnosis: Chronic Fatigue Syndrome
  Treatment: Rest; Balanced diet; Stress management
  Probable Causes: Stress, Sleep disorders, Viral infection


Search: dengue
Result:
Symptom: Dengue
  Diagnosis: Symptom not recognized. Please consult a healthcare professional.


Search: constipation
Result:
Symptom: Constipation
  Diagnosis: Constipation
  Treatment: Increased fiber intake; Hydration
  Probable Causes: Lack of dietary fiber, Dehydration, Lack of physical activity


Search: dizzyness
Result:
Symptom: Dizzyness
  Diagnosis: Symptom not recognized. Please consult a healthcare professional.


Search: dizziness
Result:
Symptom: Dizziness
  Diagnosis: Vertigo
  Treatment: Medication; Physical therapy
  Probable Causes: Inner ear infection, Dehydration, Low blood pressure
```

Register a Patient:



Register Patient

Patient Name	omma
Age	77
Sex	male
Height	160cm
Previously Diagnosed Diseases	alzeimer
Currently Diagnosed Disease	vertigo
Symptoms	dizziness
Family History	diabetes



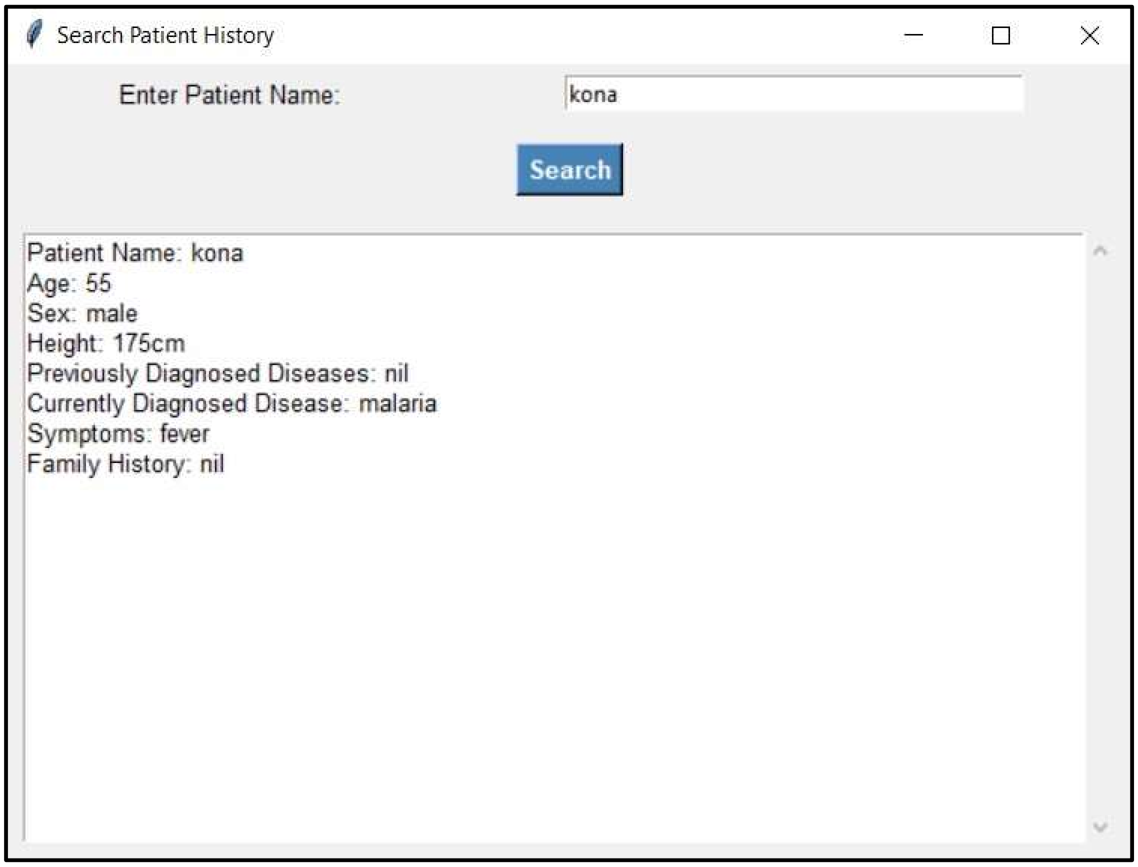
Register Patient

Patient Name	omma
Symptoms	dizziness
Family History	diabetes

Data Saved

Patient data saved successfully.

Patient's Medical History:



```
Patient Name: kona
Age: 55
Sex: male
Height: 175cm
Previously Diagnosed Diseases: nil
Currently Diagnosed Disease: malaria
Symptoms: fever
Family History: nil
```

6. CONCLUSION

The **Interactive Health Diagnosis System** is a versatile tool designed to assist users in preliminary health assessments by providing diagnosis suggestions based on symptoms, alongside treatment information and potential causes. By utilizing a simple and accessible GUI, the system enables easy navigation for non-technical users, allowing them to register patients, view medical histories, and access educational resources. This tool offers significant value in supporting initial health inquiries, promoting health awareness, and facilitating better-informed discussions with healthcare providers.

Though it provides valuable guidance, this system is not a substitute for professional medical advice and should be viewed as an informative resource to encourage proactive health management. With potential future enhancements, such as machine learning for symptom-diagnosis mapping and integration with real-time health databases, the application could further support personal and public health initiatives.

7. REFERENCES

7.1 Python Official Documentation:

- Python's official documentation provides comprehensive guides on various Python modules used in the project, such as csv for reading CSV files and unittest for testing.
- **Link**: https://docs.python.org/3/

7.2 Tkinter GUI Documentation:

- For the graphical user interface (GUI) design, Tkinter is a built-in Python library. The official Tkinter documentation offers insights into the different widgets and methods available for creating an interactive user interface.
- **Link**: https://docs.python.org/3/library/tkinter.html

7.3 Unittest Framework Documentation:

- The unittest module is part of the Python Standard Library and is useful for writing tests. The official documentation provides all necessary details on how to structure and write unit tests.
- **Link**: https://docs.python.org/3/library/unittest.html

7.4 CSV Module Documentation:

- The csv module in Python is used to handle reading from and writing to CSV files. The official Python documentation offers an in-depth guide on how to work with CSV files in Python.
- **Link**: https://docs.python.org/3/library/csv.html

7.5 Machine Learning Algorithms for Healthcare:

- A good resource to understand how AI and machine learning can be applied to healthcare diagnosis.
- **Link**: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6225786/

7.6 Introduction to AI in Healthcare:

- This article discusses the role of artificial intelligence in healthcare, providing insights into its applications for diagnostics and treatment recommendations.
- **Link**: https://healthitanalytics.com/news/the-role-of-artificial-intelligence-in-healthcare-advancements

7.7 Human-Computer Interaction and AI in Healthcare:

- This academic paper outlines the potential and challenges of AI-driven health systems that provide diagnostics through user interfaces.
- **Link**: https://www.frontiersin.org/articles/10.3389/fcomp.2020.00023/full

7.8 AI and Diagnosis: Emerging Applications:

- An in-depth article on how AI and machine learning models can assist with the diagnosis of diseases and health conditions.
- **Link**: https://www.sciencedirect.com/science/article/pii/S2589537019300177

7.9 Kaggle Datasets for Healthcare:

- Kaggle hosts numerous healthcare-related datasets that can be used to train and test machine learning models for healthcare diagnosis systems.
- **Link**: https://www.kaggle.com/datasets

7.10 An Overview of Health Informatics:

- This book provides insights into health informatics and AI technologies that can assist in medical diagnosis and treatment.
- **Link**: https://link.springer.com/book/10.1007/978-3-030-52348-9