SMART HEALTH DIAGNOSIS SYSTEM

A PROJECT REPORT

Submitted by:
Govind(23BCS10198)
Aman Kumar(23BCS14314)
Prakram Pundeer(23BCS12232)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

Nov, 2025

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 What This Project Is About

In today’s fast-moving world, people often feel unwell but avoid going to the doctor due to time, money, or just uncertainty. Our project, the Smart Health Diagnosis System, helps tackle this problem by acting like a basic first-aid advisor. Built using Java Swing, this desktop app allows users to enter their symptoms and personal info. It then tells them which disease they might have, along with basic treatment and prevention tips.

One of the coolest features is that it can even provide a link to nearby hospitals or pharmacies using Google Maps based on the user’s location. This tool doesn’t replace doctors, but it’s a quick and easy way to raise awareness and guide people in the right direction.

1.2 Why We Chose This Problem

We noticed that many people ignore common symptoms or waste time searching for information online — which can be confusing or even misleading. We wanted to create something that anyone can use, even without internet access, to get a quick idea of what might be wrong.

By using Java (which works on almost any system) and keeping the app offline-friendly, we made a solution that’s simple, smart, and helpful.

CHAPTER 2: BACKGROUND STUDY

2.1 Existing Systems and Their Drawbacks

There are already websites and apps out there like WebMD or health portals, but:

- They need the internet all the time.
- Some ask for money or show too many ads.
- They're often too complex for normal users.
- They focus on mobile or web — rarely on desktop PCs.

We realized that there isn't a lightweight, offline, Java desktop app that could help users with basic health predictions — so we decided to build one.

2.2 The Problem We're Solving

To sum it up:

We wanted to build a simple tool that asks users for a few symptoms, checks those against a small database, and gives back possible diseases with helpful advice. It should be clean, offline-capable, and easy enough for anyone to use — especially in areas with poor internet.

2.3 What We Aim to Achieve

Our main goals were:

- Let users input symptoms and get instant diagnosis suggestions.
- Show them what to do next — basic treatment and prevention.
- Help them find nearby hospitals or pharmacies.
- Keep the app light, simple, and free of clutter.
- Make it beginner-friendly but powerful.

CHAPTER 3: SYSTEM DESIGN & PROCESS

3.1 How We Picked Technologies and Features

We chose Java because it works well on all platforms and we're familiar with it.

For the GUI, we used Java Swing, which is great for building desktop apps.

We didn't use a database — instead, we created a hardcoded list of diseases using Java classes and objects. This made the app fast and easy to modify later.

3.2 Features We Finalized and the Limitations We Faced

Here's what we added:

- Name, age, gender, location inputs
- Symptom selectors (with dropdowns to avoid typos)
- Google Maps integration to show nearby medical help
- Matching logic to check user symptoms against stored disease data
- A neat output box showing the results

Limitations:

- It can't diagnose very complex or rare diseases.
- It doesn't support dynamic updates from real-world medical data.
- It's only for static (hardcoded) conditions for now.

3.3 How the System Works (Design Flow)

```
+---------------------+
|   USER INPUT        | <== Name, Age, Gender, Location, Symptoms
+---------------------+
           |
           v
+-------------------------+
| SYMPTOM MATCHING ENGINE |
| (Search in Disease DB)  |
+-------------------------+
           |
           v
+--------------------------+
|  DIAGNOSIS RESULT BUILDER|
|  (Disease + Advice + Map)|
+--------------------------+
           |
           v
+--------------------+
|  RESULT DISPLAY     |
|  (Scrollable Output)|
+--------------------+
```

Here's the flow of the app:

1. User fills in name, age, gender, symptoms, and location.
2. The app compares the symptoms with its stored disease data.
3. It lists matching diseases, treatments, and prevention tips.
4. A Google Maps link is created showing nearby help.
5. All info is shown clearly in a scrollable text box.

Each part of the app is modular — so adding features later will be easy.

CHAPTER 4: RESULTS AND TESTING

4.1 How We Built the Project

We worked in teams, divided responsibilities, and tested features step by step :

- Member 1 handled the GUI design and user experience
- Member 2 built the logic for symptom-to-disease matching
- Member 3 ran all test cases and debugged issues
- Member 4 wrote the documentation and handled the final report

We used a disease class to hold data like:

```
new Disease("Flu", Arrays.asList("Fever", "Cough", "Body Ache"), "Paracetamol,
```

This format made it easy to add or change diseases later.

4.2 Testing, Output & Observations

We tested with several combinations of symptoms:

- Fever + Cough + Loss of taste → Matched to Covid-19
- Rash + Muscle Pain + Fatigue → Detected Dengue
- Dizziness + Fatigue → Matched Anemia

All outputs were correct. Even when no match was found, the app showed a helpful message like "Please consult a doctor."

The Google Maps link worked perfectly too — opening a browser and showing nearby hospitals based on the location entered.

CHAPTER 5: CONCLUSION AND FUTURE PLANS

5.1 Wrapping It Up

We successfully created a smart, simple, and practical desktop health tool. It shows how programming can help solve real-world problems in health, education, and daily life. We used basic Java and OOP skills to deliver a working, helpful tool — and learned a lot along the way.

This project improved our teamwork, UI design skills, and understanding of how to apply logic to real-world situations.

5.2 What We'd Add in the Future

If we had more time, we'd love to add:

- User login system (to track reports or generate PDFs)
- Database storage for saving user entries
- Mobile app version using Flutter or Android Studio
- Dynamic disease updates from the internet (via APIs)
- More diseases and symptoms in the dataset
- Multi-language support for people who prefer regional languages