

**Medical Recommendation Chatbot**

A project submitted in partial fulfillment of the requirement for the award of the degree of

**BSc. (Hons.) Data Analytics and Artificial Intelligence**

Submitted to:

Dr. Amita Sharma Assistant Professor (SG) Department of CS&IT

Submitted by: Ms. Sharddha Laddha IISU/2023/ADM/36206

**Acknowledgment**

I would like to express my deepest appreciation to all individuals and organizations who have contributed to the successful completion of our project.

First and foremost, we extend our heartfelt thanks to our project supervisor, Dr. Amita Sharma, for her unwavering support and mentorship.

Special gratitude to **Dr. Anubha Jain (Director), Dr. Ruchi Nanda (Head of the Department), Dr. Navneet Sharma (Additional Head of Department),** all faculty members, and lab assistants whose contribution to stimulating suggestions and encouragement, especially in writing this report.

Additionally, I appreciate the support and resources provided by our educational institution, IIS (deemed to be) University, throughout the duration of this project.

Furthermore, I would also like to express our gratitude to our friends and family for their unwavering support, understanding, and encouragement throughout this endeavor.

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **S No.** |  | **Page No.** |
|  | Abstract | 4 |
| 1 | Introduction | 5 |
| 2 | Proposed Methodology | 6 |
| 3 | Results and Discussions | 7-8 |
| 4 | Comparative analysis | 9-10 |
| 5 | Conclusion | 11 |
|  | References | 12 |
|  | Annexure | 13-22 |

# Abstract

In recent years, healthcare systems have faced increasing pressure due to rising patient volumes, delayed consultations, and the need for immediate preliminary medical guidance. This project introduces a Medical Recommendation Chatbot — an intelligent, AI-powered conversational agent designed to provide users with basic health recommendations based on symptoms they describe in natural language. Leveraging Natural Language Processing (NLP) and rule-based symptom classification techniques, the chatbot simulates human-like interaction, asking relevant follow-up questions to refine symptom understanding and offer suitable recommendations.

The chatbot is developed with a focus on user-friendliness, scalability, and accessibility. It guides users through symptom identification and responds with general advice, possible conditions, or the suggestion to consult a doctor when needed. The system aims to reduce the burden on healthcare professionals by filtering out non-emergency queries and empowering users with timely health information. While it is not a substitute for professional diagnosis, it acts as an important support system — especially in areas with limited healthcare access or during times when consulting a doctor is not immediately feasible.

The project demonstrates the potential of conversational AI in healthcare, with promising results in symptom recognition and recommendation accuracy. Future improvements can include machine learning for smarter diagnosis, integration with user health records, multilingual support, and mobile app deployment for greater reach.

# Introduction

## Problem Statement

In many regions, especially in developing countries, access to immediate and reliable healthcare advice is limited due to overburdened medical systems, lack of infrastructure, and geographic constraints. Patients often struggle to identify the seriousness of their symptoms, leading to delayed or unnecessary hospital visits. There is a growing need for an intelligent, easily accessible system that can offer preliminary medical guidance and symptom-based recommendations without replacing professional medical consultation. This project aims to address this gap by developing a chatbot that can understand user-input symptoms using Natural Language Processing (NLP) and provide basic medical advice or suggest professional consultation where needed.

## Significance of the Problem

Many people lack quick access to medical guidance, leading to delayed treatment or unnecessary panic. A medical chatbot offers instant, preliminary advice based on symptoms, helping users make informed decisions, especially in areas with limited healthcare access. It also reduces the burden on medical professionals by handling basic queries.

# Proposed Methodology

## Dataset Acquisition and Preprocessing

The dataset used comprises a collection of symptom-disease pairs sourced from publicly available health repositories such as Kaggle and other open-source medical symptom databases. Each record includes a set of symptoms associated with a particular disease. To prepare the data for modeling, several preprocessing steps were performed. First, the dataset was cleaned by removing duplicate entries, null values, and irrelevant data. Next, the symptom text was tokenized to convert it into a structured and analyzable format. Encoding techniques such as one-hot encoding or label encoding were applied to represent symptoms and diseases numerically. The dataset was then split into training and testing subsets, typically using an 80:20 ratio. Finally, class imbalance issues were addressed through methods like Synthetic Minority Over-sampling Technique (SMOTE) or undersampling to ensure the model performs well across all disease categories..

## Feature Extraction

Feature extraction for the medical recommendation chatbot involved converting symptom text into numerical features using techniques like bag-of-words or TF-IDF. Disease labels were encoded numerically using label encoding. Additional domain-specific features, such as symptom severity or duration, were also considered. These structured features enabled the system to analyze symptom-disease relationships and provide accurate recommendations.

# Results and Discussions

## Model Performance

The medical recommendation chatbot's performance is evaluated based on its ability to identify symptoms and provide appropriate recommendations. The model uses keyword matching and symptom-based logic to suggest over-the-counter (OTC) medications and match diseases from a predefined dataset.

## Symptom Detection:

The chatbot effectively identifies symptoms based on keyword matching. The dictionary of symptoms (e.g., "headache," "fever," "cough") is used to extract relevant symptoms from the user's input. The chatbot's ability to recognize diverse terms and synonyms for each symptom (e.g., "head pain" for "headache") ensures comprehensive symptom extraction. This is crucial for accurate recommendation generation.

## Medicine Recommendation:

For common symptoms like fever, cough, and sore throat, the system generates relevant OTC recommendations. The recommendations are based on predefined rules associated with each symptom, offering users practical suggestions (e.g., "Paracetamol for fever" or "Cough syrup for cough"). This feature allows the chatbot to simulate medical advice on minor health concerns effectively.

## Disease Matching:

The chatbot also matches user inputs with diseases from a static medical dataset (e.g., "flu," "cold," "headache"). Once a disease is identified from the input text, the chatbot provides appropriate medical treatments or medications (e.g., "For flu, consider Oseltamivir"). This dataset-driven approach adds credibility and depth to the recommendations, enhancing the user experience.

## Fallback Mechanism:

When the user's input does not clearly match any known symptoms or diseases, the chatbot resorts to predefined responses such as greetings ("Hello!") or offers further assistance ("Ask me about common symptoms"). While this ensures continued interaction, improvements can be made to handle a wider range of input queries.

## User Interaction & Responsiveness:

The chatbot's response time is quick, with a slight delay (simulated with setTimeout) for better user experience. The conversation is smooth, and users can ask multiple queries without disruption. The design is user-friendly, with clear message distinction between the user and the bot.

## Challenges:

Handling Complex Inputs: While the system is effective for simple symptom descriptions, it struggles with complex or multi-symptom inputs. For example, when symptoms are mentioned ambiguously or in a non-standard way, the chatbot may not recognize them as intended.

Rare Diseases and Uncommon Symptoms: The current dataset includes common diseases and symptoms, but it lacks rare diseases or less common symptoms. Expanding the dataset with more disease-symptom pairs can improve its coverage.

## Future Improvements:

Integrating NLP Models: To improve understanding of complex symptoms, integrating advanced natural language processing (NLP) models like BERT or GPT could enhance the chatbot's ability to interpret and respond to diverse inputs.

Dynamic Dataset Updates: Regularly updating the disease and symptom dataset, including the latest medical research and guidelines, can ensure the chatbot remains current.

User Personalization: Incorporating user preferences or history for personalized medical advice could be another enhancement. For instance, keeping track of past symptoms or conditions can make future interactions more efficient.

# Comparative analysis

* 1. **Usability:**

User Experience: The chatbot provides an intuitive and simple interface with a clean design and seamless interactions. The form layout and input fields are responsive, ensuring that users across various devices can easily interact with the chatbot.

Response Time: The chatbot responds with a slight delay (300 ms), which enhances the user experience by simulating a more human-like response. However, this delay is kept short to ensure that it does not disrupt the flow of conversation.

Fallback and Error Handling: The bot handles unknown queries with a fallback message, asking users to consult a healthcare professional, thus maintaining a safe and responsible user experience.

* 1. **Functionality:**

Symptom Detection and Recommendations: The chatbot can successfully extract symptoms from user input using the extractSymptoms() function. It then offers over-the-counter (OTC) medicine recommendations for common ailments like headaches, coughs, and nasal congestion. This makes the bot relatively useful for basic medical symptom inquiries.

Dataset Matching: The checkDatasetDisease() function matches user input with pre-defined diseases and provides recommendations based on the dataset, such as specific medications. This gives a personalized touch to the bot's recommendations, although the dataset is limited in scope.

Symptom Matching Limitations: The chatbot's ability to match symptoms to diseases or recommend medicines is limited by the dataset of symptoms and diseases. While the current dataset covers a range of common illnesses, the bot could fail to provide useful recommendations for less common symptoms or diseases.

Handling of Complex Queries: The chatbot handles basic symptom-based queries well, but it may struggle with more complex or ambiguous medical inquiries. The fallback responses are generic and can be further improved.

* 1. **Accuracy of Recommendations:**

OTC Recommendations: The chatbot provides sensible OTC medicine recommendations for common symptoms, such as headaches, sore throats, and fever. The recommendations appear to be sound, assuming that the user has mild symptoms and is not dealing with more serious underlying conditions.

Dataset Recommendations: While the dataset is useful, it is not exhaustive, and the chatbot may miss providing information about certain diseases or conditions not included in the list. This means the bot's ability to assist is limited by the quality and comprehensiveness of its dataset.

General Medical Advice: The bot avoids making potentially harmful recommendations, such as recommending specific treatments for serious conditions. This is a safety measure, as the bot guides users toward professional healthcare consultation when necessary.

**4.4 Potential Enhancements:**

Natural Language Processing (NLP) Integration: While the current symptom extraction mechanism works based on keyword matching, a more sophisticated NLP-based approach (such as using named entity recognition or machine learning models) could enhance the bot's ability to understand and respond to a wider variety of queries. This would make the chatbot more accurate in handling different ways users describe their symptoms.

Database Expansion: The current dataset only includes a limited number of diseases. Expanding this dataset with more diseases, medications, and treatment protocols would increase the bot's effectiveness and provide more accurate recommendations.

Personalized Recommendations: If the bot could ask users for more personalized information (e.g., age, gender, or underlying conditions), it could make more tailored recommendations, further improving the user experience.

**4.5 Comparative Performance (Hypothetical Benchmarks):**

Speed: Compared to traditional healthcare chatbots that rely on deep neural networks or extensive backend databases, the current system (based on simple keyword matching and a static dataset) is fast and efficient, with quick responses and minimal latency.

Accuracy: In terms of accuracy, the chatbot might fall short when compared to more advanced models that incorporate real-time medical data, user histories, and complex NLP for context understanding. For example, Google's MedPaLM or IBM Watson Health are capable of providing more accurate, nuanced medical advice based on vast amounts of data.

Scalability: The current chatbot could become cumbersome if the dataset expands too much, as it relies on a simple, static structure for storing medical conditions. Advanced models like GPT-3 or specialized healthcare systems can scale more effectively to handle large, complex datasets.

6. Suggestions for Improvement:

Integrating AI/ML Models: Incorporating AI models like GPT-3 or other natural language processing tools would help the chatbot provide better responses by understanding a wider range of inputs, including various medical terminologies and colloquial language.

Dynamic Learning: Allowing the chatbot to learn from user interactions (with proper safeguards) could help improve its accuracy over time, providing more personalized and relevant recommendations.

User Authentication and Data Privacy: For sensitive medical data, it would be beneficial to implement user authentication and ensure compliance with data privacy laws like GDPR or HIPAA when handling medical queries.

# Conclusion

The chatbot provides a useful and efficient solution for handling basic medical inquiries by offering symptom-based recommendations and OTC medicines. Its simplicity and speed make it accessible, and its ability to provide tailored responses based on a predefined dataset is valuable for addressing common conditions. However, the chatbot's limitations lie in its narrow dataset, inability to handle complex or rare conditions, and its reliance on basic keyword matching for symptom detection.

While it provides a safe and responsible approach to medical inquiries, the chatbot could benefit from improvements such as the integration of more sophisticated NLP techniques, a broader and more dynamic dataset, and enhanced user personalization. Incorporating machine learning models and allowing the chatbot to learn from interactions could significantly enhance its accuracy and relevance.

In conclusion, while the chatbot serves as a helpful tool for basic health-related queries, its scope is limited, and it can be further developed into a more comprehensive, intelligent system by leveraging advanced AI technologies and expanding its knowledge base

# References

* Chawla, S., & Gera, P. (2020). A review on the development and deployment of healthcare chatbots. International Journal of Computer Applications, 975, 11-18.
* This paper discusses the development of healthcare chatbots, including design methodologies, challenges, and various applications in the medical field.
* Bhat, S., & Shaikh, M. (2021). Medical recommendation system using machine learning and artificial intelligence techniques: A review. Health Information Science and Systems, 9(1), 1-11.
* This paper reviews the use of machine learning and AI techniques in medical recommendation systems, which is relevant to building chatbots that recommend healthcare solutions.
* Xu, J., & Li, Y. (2019). Chatbot for medical recommendations: The state of the art. Proceedings of the 2019 2nd International Conference on Artificial Intelligence and Health Informatics, 94-100.
* This conference paper covers the state-of-the-art techniques used in chatbots for medical recommendations, offering insights into their applicati

Annexure

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Medical Recommendation Chatbot</title>

<meta name="viewport" content="width=device-width, initial-scale=1" />

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: linear-gradient(135deg, #74ebd5, #ACB6E5);

margin: 0;

padding: 0;

display: flex;

flex-direction: column;

height: 100vh;

}

#app {

max-width: 600px;

margin: auto;

flex-grow: 1;

display: flex;

flex-direction: column;

background: white;

box-shadow: 0 8px 24px rgba(0,0,0,0.2);

border-radius: 12px;

overflow: hidden;

height: 90vh;

}

header {

background-color: #4b6cb7;

background-image: linear-gradient(315deg, #4b6cb7 0%, #182848 74%);

padding: 1rem 1.5rem;

color: white;

font-size: 1.5rem;

font-weight: 700;

text-align: center;

user-select: none;

}

#chat {

flex-grow: 1;

overflow-y: auto;

padding: 1rem 1.5rem;

box-sizing: border-box;

}

.message {

margin-bottom: 1rem;

max-width: 80%;

line-height: 1.4;

font-size: 1rem;

word-wrap: break-word;

border-radius: 16px;

padding: 0.6rem 1rem;

box-shadow: 0 1px 4px rgba(0,0,0,0.1);

white-space: pre-line;

}

.user {

background: #daf1fd;

align-self: flex-end;

border-bottom-right-radius: 0;

}

.bot {

background: #e4e9f2;

align-self: flex-start;

border-bottom-left-radius: 0;

color: #222;

font-weight: 600;

}

form {

display: flex;

padding: 1rem;

border-top: 1px solid #ddd;

background: #f8f9fb;

}

input[type=text] {

flex-grow: 1;

border: 1px solid #bbb;

border-radius: 25px;

padding: 0.7rem 1.2rem;

font-size: 1rem;

outline: none;

transition: 0.3s border-color ease;

}

input[type=text]:focus {

border-color: #4b6cb7;

box-shadow: 0 0 8px #4b6cb7aa;

}

button {

background: #4b6cb7;

color: white;

border: none;

border-radius: 25px;

padding: 0 1.5rem;

margin-left: 1rem;

font-weight: 700;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover:not(:disabled) {

background: #365298;

}

button:disabled {

cursor: not-allowed;

opacity: 0.6;

}

@media (max-width: 600px) {

#app {

height: 100vh;

border-radius: 0;

box-shadow: none;

}

#chat {

padding: 0.8rem 1rem;

}

input[type=text] {

font-size: 1rem;

}

button {

padding: 0 1rem;

}

}

</style>

</head>

<body>

<div id="app">

<header>Medical Recommendation Chatbot</header>

<div id="chat" aria-live="polite" aria-atomic="false"></div>

<form id="inputForm" aria-label="User input form">

<input type="text" id="userInput" autocomplete="off" placeholder="Describe your symptoms or ask about diseases..." required maxlength="500" aria-describedby="desc"/>

<button type="submit" id="sendBtn">Send</button>

</form>

</div>

<script>

// Symptom keywords dictionary

const symptomKeywords = {

"headache": ["headache", "head pain"],

"fever": ["fever", "high temperature"],

"body ache": ["body ache", "muscle pain", "general pain"],

"cough": ["cough", "coughing"],

"dry cough": ["dry cough", "non-productive cough"],

"productive cough": ["productive cough", "cough with phlegm"],

"sore throat": ["sore throat", "throat pain", "scratchy throat"],

"stuffy nose": ["stuffy nose", "nasal congestion", "blocked nose"],

"nasal congestion": ["nasal congestion", "blocked nose"]

};

// Basic dataset for demo - since no server-side reading is possible,

// emulating the medical data as a JS array of objects here for demo purposes.

const medicalData = [

{ Disease: "flu", Medicine: "Oseltamivir" },

{ Disease: "cold", Medicine: "Antihistamines or decongestants" },

{ Disease: "headache", Medicine: "Paracetamol or Ibuprofen" },

{ Disease: "strep throat", Medicine: "Antibiotics prescribed by a doctor" },

{ Disease: "flu", Medicine: "Oseltamivir, Paracetamol for fever" },

{ Disease: "cold", Medicine: "Antihistamines like Cetirizine, Decongestants like Pseudoephedrine" },

{ Disease: "headache", Medicine: "Paracetamol or Ibuprofen" },

{ Disease: "strep throat", Medicine: "Penicillin or Amoxicillin (as prescribed)" },

{ Disease: "diarrhea", Medicine: "ORS solution, Loperamide for symptom relief" },

{ Disease: "constipation", Medicine: "Laxatives like Bisacodyl or fiber supplements" },

{ Disease: "acidity", Medicine: "Antacids like Ranitidine or Omeprazole" },

{ Disease: "indigestion", Medicine: "Antacids, Digestive enzymes" },

{ Disease: "asthma", Medicine: "Inhalers with Salbutamol or corticosteroids" },

{ Disease: "hypertension", Medicine: "Amlodipine, Lisinopril, or Losartan" },

{ Disease: "diabetes", Medicine: "Metformin or insulin (depending on type)" },

{ Disease: "migraine", Medicine: "Sumatriptan, Naproxen" },

{ Disease: "allergy", Medicine: "Loratadine, Cetirizine" },

{ Disease: "back pain", Medicine: "Ibuprofen, Muscle relaxants like Cyclobenzaprine" },

{ Disease: "urinary tract infection", Medicine: "Nitrofurantoin, Ciprofloxacin (as prescribed)" },

{ Disease: "skin rash", Medicine: "Topical corticosteroids or antihistamines" },

{ Disease: "eczema", Medicine: "Moisturizers, Hydrocortisone cream" },

{ Disease: "acne", Medicine: "Benzoyl peroxide, Salicylic acid, Clindamycin gel" },

{ Disease: "bronchitis", Medicine: "Cough syrups, Amoxicillin if bacterial" },

{ Disease: "tonsillitis", Medicine: "Paracetamol, Antibiotics if bacterial" },

{ Disease: "sinusitis", Medicine: "Decongestants, Saline nasal spray, Amoxicillin if bacterial" },

{ Disease: "dengue", Medicine: "Paracetamol for fever, Hydration, No NSAIDs" },

{ Disease: "malaria", Medicine: "Chloroquine, Artemisinin-based combinations (ACT)" },

{ Disease: "typhoid", Medicine: "Cefixime, Azithromycin (as prescribed)" },

{ Disease: "chickenpox", Medicine: "Calamine lotion, Acyclovir in severe cases" },

{ Disease: "measles", Medicine: "Paracetamol, Vitamin A supplements" },

{ Disease: "hepatitis A", Medicine: "Supportive care, Rest, Hydration" },

{ Disease: "covid-19", Medicine: "Paracetamol for fever, supportive care, consult doctor" },

{ Disease: "pink eye", Medicine: "Lubricant eye drops, Antibiotic drops for bacterial conjunctivitis" }

];

// Extract symptoms from user input

function extractSymptoms(text) {

const lowerText = text.toLowerCase();

return Object.keys(symptomKeywords).filter(symptom =>

symptomKeywords[symptom].some(keyword => lowerText.includes(keyword))

);

}

// Recommend OTC medicines

function recommendOTC(symptoms) {

const rec = [];

if (symptoms.some(sym => ["headache", "fever", "body ache"].includes(sym))) {

rec.push("Consider Paracetamol or Ibuprofen for pain and fever.");

}

if (symptoms.includes("cough")) {

if (symptoms.includes("dry cough")) {

rec.push("Try a suppressant like Dextromethorphan.");

} else if (symptoms.includes("productive cough")) {

rec.push("Try an expectorant like Guaifenesin.");

} else {

rec.push("For cough relief, consider a cough syrup.");

}

}

if (symptoms.includes("sore throat")) {

rec.push("Lozenges or throat sprays may help.");

}

if (symptoms.some(sym => ["stuffy nose", "nasal congestion"].includes(sym))) {

rec.push("Use nasal decongestant sprays or Pseudoephedrine tablets.");

}

return rec.length > 0 ? rec.join("\n") : null;

}

// Check if user input matches a disease in the dataset

function checkDatasetDisease(text) {

const input = text.toLowerCase();

for (const entry of medicalData) {

if (entry.Disease && entry.Medicine) {

if (input.includes(entry.Disease.toLowerCase())) {

return `For ${entry.Disease}, consider: ${entry.Medicine}`;

}

}

}

return null;

}

// Chat history - array of {speaker: "user" | "bot", text: ""}

let chatHistory = [];

// Function to add message to chat UI

function addMessage(speaker, text) {

chatHistory.push({speaker, text});

const chat = document.getElementById("chat");

const messageDiv = document.createElement("div");

messageDiv.classList.add("message", speaker);

messageDiv.textContent = text;

chat.appendChild(messageDiv);

chat.scrollTop = chat.scrollHeight;

}

// Function to sanitize user input

function sanitizeInput(text) {

return text.trim();

}

// Function for generating bot response synchronously

// Since we can't call transformers locally here,

// We'll simulate conversational responses with some predefined fallback

function generateBotResponse(userText) {

// First check dataset

const datasetRec = checkDatasetDisease(userText);

if (datasetRec) {

return `Bot (Dataset): ${datasetRec}`;

}

// Extract symptoms and recommend OTC

const symptoms = extractSymptoms(userText);

const otcRec = recommendOTC(symptoms);

if (otcRec) {

return `Bot (OTC): ${otcRec}`;

}

// Handle thanks / thank you

if (/^thanks?$|thank you/i.test(userText.trim())) {

return "Bot: You're welcome! Stay safe. 😊";

}

// Fallback "chatty" response

// Some simple canned responses for demo:

const lower = userText.toLowerCase();

if (lower.includes("hello") || lower.includes("hi")) {

return "Bot: Hello! How can I assist you today?";

}

if (lower.includes("help") || lower.includes("support")) {

return "Bot: Sure, ask me about common symptoms or diseases!";

}

// Default fallback

return "Bot: Sorry, I couldn't find specific recommendations. Please consult a healthcare professional.";

}

// Event handler for form submit

document.getElementById("inputForm").addEventListener("submit", function(event) {

event.preventDefault();

const inputField = document.getElementById("userInput");

let userText = sanitizeInput(inputField.value);

if (!userText) return;

addMessage("user", userText);

// Generate bot reply (simulated async for better UX)

setTimeout(() => {

const botResponse = generateBotResponse(userText);

addMessage("bot", botResponse);

}, 300);

inputField.value = "";

inputField.focus();

});

</script>

</body>

</html>

