**CAPSTONEPROJECT**

**"Medical Recommendation Chatbot"**

**Project Report**

Submitted to: Submitted by:

Dr. Amita Sharma Shraddha Laddha

Enrollment no:

IISU/2024/ADM/

Course:

B.Sc.(H) Data Analytics

And AI



**IIS (Deemed to be University),**

**JAIPUR**

**(2024-2025)**

# Certificate

This is to certify that the capstone project titled **"Medical Recommendation Chatbot"** has been successfully completed and submitted by **Shraddha Laddha**, a student of Semester 4, B.Sc. (Hons.) Data Analytics and AI, under the guidance of **Dr. Amita Sharma**, at IIS (Deemed to be) University, Jaipur.  
  
Date of Submission: May 2025

# Index

1. Abstract

2. Introduction

3. Literature Review

4. Methodology

5. Design

6. Backend Integration

7. Data Privacy and Ethical Considerations

8. Testing and User Feedback

9. Code

10. Results and Observations

11. References

## 1. Abstract

This capstone project introduces a web-based Medical Recommendation Chatbot that provides Over-the-Counter (OTC) medicine suggestions based on user-described symptoms. The chatbot identifies symptoms using keyword detection and matches them against a predefined dataset to suggest appropriate medications. It is designed for accessible and immediate health assistance and is implemented using HTML, CSS, and JavaScript.

## 2. Introduction

Access to basic medical guidance remains a significant challenge, especially in underserved regions. Chatbots offer scalable solutions by automating user interactions and delivering health-related recommendations. This project aims to develop a symptom-based chatbot that offers OTC medicine suggestions for common ailments, empowering users to make informed decisions before consulting a healthcare professional.

## 3. Literature Review

With advancements in Natural Language Processing (NLP), research on medical chatbots has significantly expanded. Platforms such as Microsoft’s Health Bot and Babylon Health employ advanced AI and cloud computing. However, simpler rule-based systems remain effective for addressing minor health concerns. This project leverages various Kaggle medical datasets to analyze symptom-medicine relationships.

## 4. Methodology

This project employs a rule-based approach:  
- Keywords are extracted from user input.  
- Keywords are matched to predefined symptoms.  
- Relevant medicines are suggested based on the dataset.  
- If no match is found, a fallback response is returned.

## 5. Design

The user interface is developed using HTML and CSS, ensuring a clean and responsive layout. Key features include:  
- An interactive chat window.  
- A text input field for symptom entry.  
- A layout adaptable to both desktop and mobile devices.

## 6. Backend Integration

Due to browser-based constraints, the chatbot operates without server-side integration. All backend logic is handled using JavaScript within the browser environment. A dummy dataset embedded in the script simulates backend behavior.

## 7. Data Privacy and Ethical Considerations

The chatbot does not collect, store, or transmit personal data. Users are advised not to enter sensitive or private health information. The tool is intended solely for educational and demonstrative purposes and is not a substitute for professional medical advice.

## 8. Testing and User Feedback

The chatbot was tested with queries involving symptoms such as fever, sore throat, and cough. Feedback indicated that the system’s responses were generally accurate and useful. However, users expressed a desire for more personalized interactions and support for diverse linguistic inputs.

## 9. Code

The chatbot is implemented entirely using front-end technologies:  
- HTML for page structure  
- CSS for visual design  
- JavaScript for logic, symptom recognition, and response generation  
  
<!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" }

        ];

        // 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>

## 10. Results and Observations

- Accurately identifies and responds to symptom-based queries  
- Interface is responsive and user-friendly  
- Limitations include reliance on a static dataset and the absence of NLP for complex sentence interpretation

## 11. References

- Kaggle Symptom-Disease Dataset: https://www.kaggle.com/datasets/  
- Mozilla MDN Web Docs  
- W3Schools JavaScript Tutorials  
- Stack Overflow discussions on symptom-matching logic  
- Sample GitHub repositories related to health chatbots

Submitted by:  
Shraddha Laddha  
Semester 4, B.Sc. (Hons.) Data Analytics and AI  
IIS (Deemed to be) University, Jaipur