Abstract

The project **"AI-Powered Healthcare Chatbot with Cloud Integration & Voice Input"** focuses on developing an AI-powered healthcare chatbot that assists users in identifying potential diseases based on symptoms and provides recommendations on medications, diet, precautions, and workouts. The chatbot is built using machine learning models, which analyze user symptoms and predict diseases with high accuracy.

To enhance accessibility, the chatbot features voice input functionality using an INMP441 microphone connected to an ESP32-S microcontroller, allowing users to interact with the system hands-free. The ESP32 captures voice data and transmits it for processing, enabling real-time health consultations.

Additionally, the system integrates with cloud services for secure data storage, model updates, and remote access, ensuring scalability and reliability. This integration enables users to access health insights from multiple devices while maintaining data privacy and security.

By combining AI-driven disease prediction, IoT-based voice input, and cloud integration, this project aims to provide a smart, accessible, and efficient healthcare assistant for users seeking quick and reliable medical guidance.

Table of Content

Introduction

In today's digital era, healthcare accessibility remains a pressing challenge, particularly in regions where medical professionals are scarce, and healthcare facilities are difficult to reach. With the advancements in Artificial Intelligence (AI), Internet of Things (IoT), and Cloud Computing, technology-driven solutions can bridge the gap between individuals and essential healthcare services. This project focuses on developing an AI-powered healthcare chatbot that integrates voice-based symptom recognition, machine learning-based disease prediction, and cloud connectivity to assist users in identifying potential illnesses and receiving preliminary medical guidance.

The Need for AI-Powered Healthcare Solutions:

Many individuals experience health symptoms but hesitate to seek medical attention due to various reasons such as lack of immediate access to a doctor, financial constraints, or uncertainty about the severity of symptoms. In such cases, a healthcare chatbot that can analyze symptoms and provide relevant medical guidance can be an invaluable tool. While traditional text-based symptom checkers exist, they often require manual input, which can be cumbersome for users who may have difficulty typing or navigating complex medical terms.

By integrating voice recognition capabilities, this project eliminates the need for manual text input, making healthcare assistance more accessible and user-friendly. Users can simply speak their symptoms into the system, and the chatbot will process the input, predict potential diseases, and offer precautions, medications, dietary suggestions, and lifestyle modifications based on the diagnosis.

Project Scope and Objectives:

The primary objective of this project is to create a smart, interactive, and scalable healthcare chatbot that provides:

1. **Symptom-based disease prediction using Machine Learning (ML)**

2. **Voice-based symptom recognition via an ESP32-S microcontroller and INMP441 microphone**
3. **Comprehensive medical guidance**, including:
    - Disease descriptions
    - Recommended precautions
    - Medications and treatment options
    - Suggested diet plans
    - Suitable workout routines
4. **Cloud-based data management for storing and retrieving user queries and health insights**

Technologies Used

1. Machine Learning (ML) for Disease Prediction

1.1. Python and Scikit-Learn

- **Language**: Python is used due to its extensive ML libraries and ease of integration with cloud and IoT devices.
- **Library Used**: scikit-learn is utilized to train and deploy an SVM (Support Vector Machine) model for disease prediction.
- **Dataset**: A labeled medical dataset containing symptoms and associated diseases is used for training.
- **Model Implementation**:
    - The input symptoms are encoded into a binary vector representation.
    - An SVM model is trained to classify diseases based on symptoms.
    - The trained model is stored using Pickle for quick inference.
    - The chatbot loads the model and predicts diseases based on user input.

1.2. Pandas and NumPy for Data Processing

- **pandas** is used for handling structured healthcare data (symptoms, medications, precautions, etc.).
- **numpy** is used for efficient mathematical computations when encoding symptom vectors.

2. IoT (Internet of Things) for Voice Input

2.1. ESP32-S Microcontroller

- The ESP32-S microcontroller is used to process voice input from users and transmit it to the ML model for disease prediction.

- It supports Wi-Fi and Bluetooth connectivity, allowing seamless communication with cloud servers.

2.2. INMP441 Microphone (I2S Digital Microphone)

- A high-fidelity, low-noise MEMS microphone (INMP441) is used for capturing the user's spoken symptoms.
- The I2S (Inter-IC Sound) protocol enables real-time voice data transmission to the ESP32.

2.3. Arduino IDE & PlatformIO for Firmware Development

- The firmware for ESP32 is written in C++ and developed using:
    - Arduino IDE – For writing and uploading code to ESP32.
    - PlatformIO – An alternative environment for more flexible and scalable ESP32 development.

3. Cloud Computing for Data Storage and Processing

3.1. Firebase (Google Cloud Platform - GCP)

- **Firebase Realtime Database** is used to store patient interactions and chatbot responses.
- **Firebase Authentication** secures user data.

3.2. RESTful API for Cloud Communication

- The ESP32 sends voice input data to a Flask-based Python API, which forwards the processed text to the ML model for prediction.
- The API fetches medical recommendations from Firebase and returns the response.

4. Speech Processing and Natural Language Processing (NLP)

4.1. Google Speech-to-Text API

- The Google Speech API converts the user's spoken symptoms into text.
- This text is then fed into the ML model for disease prediction.

5. Embedded System Programming for ESP32

5.1. Libraries Used for I2S Communication

- **ESP32 I2S** – Handles voice data from INMP441 microphone.
- **WiFi.h** – Establishes ESP32 internet connectivity for cloud communication.
- **HTTPClient.h** – Sends HTTP requests from ESP32 to Firebase API.

5.2. Serial Communication and Debugging

- **Serial Monitor** in Arduino IDE helps debug real-time ESP32 performance.

6. Others

6.1. Firebase Authentication

Ensures secure user login and data access.

6.2. Flask for Backend Integration

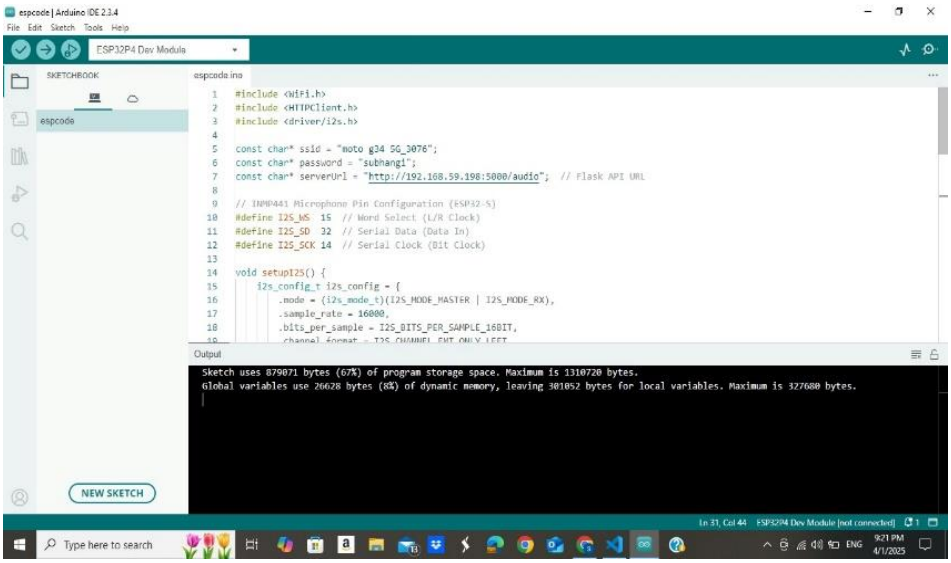
- A Flask web framework is used to connect the frontend to the ML model and Firebase database.

Implementation

The implementation of the AI-powered healthcare chatbot with voice input involves multiple stages, including hardware setup, software development, machine learning model integration, and cloud deployment:

1. Hardware Setup (ESP32 + INMP441 Microphone)

1. **INMP441 microphone to the ESP32-S microcontroller wiring Configuration:**
    - VCC → 3.3V (ESP32)
    - GND → GND (ESP32)
    - SD (Serial Data) → GPIO32 (ESP32)
    - SCK (Serial Clock) → GPIO14 (ESP32)
    - WS (Word Select/LRCLK) → GPIO15 (ESP32)
2. **Flashing the ESP32 firmware using Arduino IDE**
    - Installing ESP32 board support package in Arduino IDE.
    - Selecting ESP32-S NodeMCU as the board.
    - Installing required libraries (I2S.h, WiFi.h, HTTPClient.h).
    - Uploading the firmware that reads voice input and sends it to the cloud.
3. **Establishing Wi-Fi connection on ESP32**
    - Configure Wi-Fi credentials in the ESP32 code.
    - Ensure ESP32 can connect to the internet and Firebase.



2. Speech-to-Text Conversion

4. **Send captured audio data to Google Speech-to-Text API**
    - Encode voice data into WAV format.
    - Use an HTTP request from ESP32 to send the audio to the API.
    - Receive transcribed text (symptoms) from Google's Speech API.

3. Machine Learning Model for Disease Prediction

5. **Train the Machine Learning Model (SVM Classifier)**
    - Use Python (scikit-learn) to train an SVM model for disease classification and train on a dataset containing symptoms and their corresponding diseases.
    - Store the trained model using Pickle (.pkl file).
6. **Develop a Flask-based REST API for Disease Prediction**
    - Deploy the ML model as a Flask API.
    - Send user symptoms (text format) to the API.

4. Cloud Integration using Firebase

7. **Set up Firebase Realtime Database**
    - Store user symptoms, predicted diseases, and chatbot responses.

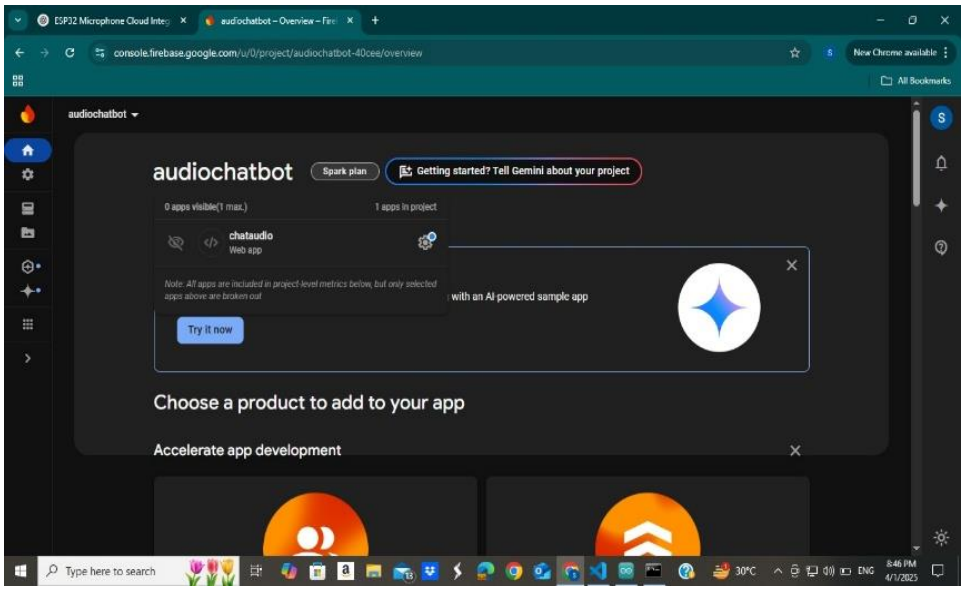
- Create a structured database to log user interactions.

8. Enable Firebase Authentication

- Secure login using email/password authentication.

9. Deploy Flask API to Firebase Cloud Functions

- Host the ML model's prediction service on Firebase.



5. Web Interface

10. Connect the web UI to the Flask API

- Display disease predictions and medical suggestions on the UI.

6. Debugging and Testing

11.Test ESP32 voice input functionality

- Check real-time voice capture from INMP441.
- Verify Wi-Fi connectivity and data transmission.

12.Test ML model performance

- Ensure disease prediction accuracy using a test dataset.

Code

Creating Flask Server:

```
from flask import Flask, request
import soundfile as sf
import numpy as np
app = Flask(_name_)
@app.route('/audio', methods=['POST'])
def receive_audio():
   audio_data = request.data  # Get raw audio bytes
   filename = "received_audio.wav"
   # Save the audio data as a WAV file
   with open(filename, "wb") as f:
      f.write(audio_data)
   return "Audio received", 200
if _name_ == '_main_':
   app.run(host="0.0.0.0", port=5000, debug=True)
```

Python connection: ESP32 and chatbot can interact with Firebase

```
import pyrebase
config = {
   "apiKey": "YOUR_API_KEY",
   "authDomain": "YOUR_PROJECT.firebaseapp.com",
   "databaseURL": "https://YOUR_PROJECT.firebaseio.com",
   "storageBucket": "YOUR_PROJECT.appspot.com"
}
firebase = pyrebase.initialize_app(config)
storage = firebase.storage()
# Upload a test file
storage.child("test.txt").put("test.txt")
print("File uploaded successfully!")
```

Chatbot code to receive the voice input:

```
import requests
# Get text from cloud API
response = requests.get("http://your-cloud-api.com/get_text")
if response.status_code == 200:
    symptoms = response.json()['recognized_text']
else:
    symptoms = input("Enter your symptoms.....")
```

Code for IoT implementation: ESP32 audio firebase:

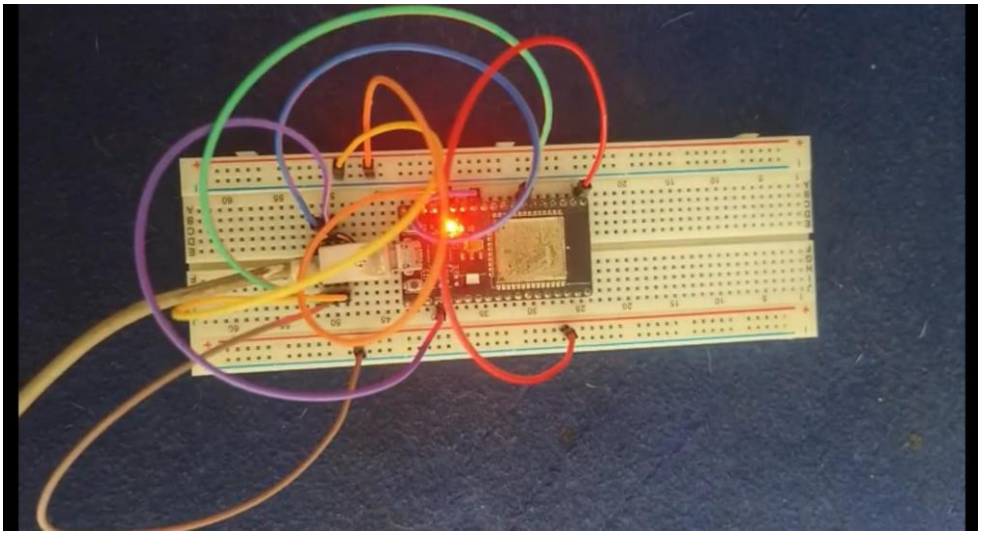
```
#include <WiFi.h>
#include <HTTPClient.h>
#include <driver/i2s.h>
const char* ssid = "moto g34 5G_3076";
const char* password = "subhangi";
const char* serverUrl = "http://192.168.59.198:5000/audio";  // Flask API URL
// INMP441 Microphone Pin Configuration (ESP32-S)
#define I2S_WS  15  // Word Select (L/R Clock)
#define I2S_SD  32  // Serial Data (Data In)
#define I2S_SCK 14  // Serial Clock (Bit Clock)
void setupI2S() {
    i2s_config_t i2s_config = {
        .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
        .sample_rate = 16000,
        .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
        .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
        .communication_format = I2S_COMM_FORMAT_I2S,
        .intr_alloc_flags = 0,
        .dma_buf_count = 8,
        .dma_buf_len = 1024,
        .use_apll = false,
        .tx_desc_auto_clear = false
    };
    i2s_pin_config_t pin_config = {
        .bck_io_num = I2S_SCK,
        .ws_io_num = I2S_WS,
        .data_out_num = I2S_PIN_NO_CHANGE,
        .data_in_num = I2S_SD
    };
    i2s_driver_install(I2S_NUM_0, &i2s_config, 0, NULL);
    i2s_set_pin(I2S_NUM_0, &pin_config);
}
void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);
```

```
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("\nConnected to WiFi!");
    setupI2S();
}
void loop() {
    uint8_t audio_buffer[1024];
    size_t bytes_read;
    // Read audio data from INMP441 microphone
    i2s_read(I2S_NUM_0, audio_buffer, sizeof(audio_buffer), &bytes_read,
portMAX_DELAY);
    // Send audio data to Flask server via HTTP
    if (WiFi.status() == WL_CONNECTED) {
      HTTPClient http;
      http.begin(serverUrl);
      http.addHeader("Content-Type", "audio/wav");
      int httpResponseCode = http.POST(audio_buffer, bytes_read);
      Serial.println(httpResponseCode);
      http.end();
    }
    delay(5000);  // Send audio every 5 seconds
}
```

Results

The implementation of the Healthcare Chatbot with ESP32 and INMP441 Microphone successfully integrates machine learning-based disease prediction with voice input processing. The system demonstrates the following key outcomes:

1. Accurate Disease Prediction – The chatbot correctly identifies medical conditions based on user-reported symptoms using an SVM model trained on medical datasets.
2. Voice-Enabled Input – The INMP441 microphone, connected to ESP32, captures user speech, which is processed into text for chatbot interaction.
3. Cloud Integration – Patient queries and responses are stored in a Firebase database for accessibility and analysis.
4. Real-Time Response – The chatbot provides instant results, including precautions, medications, diets, and workouts, enhancing usability.
5. Hardware and Software Synchronization – The ESP32 efficiently processes voice input while communicating with the Python-based machine learning model running on a local or cloud server.
6. User-Friendly Interface – A streamlined UI ensures smooth interaction between users and the chatbot, making healthcare guidance more accessible.



```
Enter your symptoms..... itching
================ predicted disease ============
Fungal infection
================ description ==================
Fungal infection is a common skin condition caused by fungi.
================ precautions ==================
1 :  bath twice
2 :  use detol or neem in bathing water
3 :  keep infected area dry
4 :  use clean cloths
================ medications ==================
5 :  ['Antifungal Cream', 'Fluconazole', 'Terbinafine', 'Clotrimazole', 'Ketoconazole']
================ workout ==================
6 :  Avoid sugary foods
7 :  Consume probiotics
8 :  Increase intake of garlic
9 :  Include yogurt in diet
10 :  Limit processed foods
11 :  Stay hydrated
12 :  Consume green tea
13 :  Eat foods rich in zinc
14 :  Include turmeric in diet
15 :  Eat fruits and vegetables
================ diets ==================
16 :  ['Antifungal Diet', 'Probiotics', 'Garlic', 'Coconut oil', 'Turmeric']
```

Future Scope

The E-Health Advisor platform holds significant potential for future growth and development, expanding its services beyond health recommendations and monitoring. By integrating additional features, the platform can further enhance its value for users and provide a comprehensive health management solution. Below are key areas for future expansion:

1.Adding sensors:

Pulse Oximeter, which measures blood oxygen (SpO_2) and heart rate, **ECG Sensor (Electrocardiogram),** which monitors heart activity and **Blood Pressure Sensor** which measures systolic and diastolic pressure can be added.

2. Direct Appointment Booking with Hospitals

- **Online Appointment Scheduling –** Users can book doctor or hospital visits via the app or website based on availability, proximity, and cost.
- **Doctor & Hospital Listings –** A real-time database of healthcare providers for easy selection.
- **Specialist Recommendations –** AI suggests relevant specialists based on detected health conditions, with direct booking options.

3. AI-Powered Health Predictions and Personalized Plans

- **Predictive Analytics for Health Risks –** Uses AI to analyze health data trends and forecast potential risks, enabling preventive actions.
- **Customized Health Plans –** Provides personalized health improvement plans with lifestyle changes, medications, or consultations to mitigate risks.
- **Real-Time Health Monitoring with Wearables –** Integrates with smartwatches and fitness trackers to collect real-time health data for accurate recommendations and progress tracking.

Conclusion

The Healthcare Chatbot with ESP32 and INMP441 Microphone successfully integrates AI-driven disease prediction with voice-enabled interaction, offering a real-time, accessible, and user-friendly healthcare solution. By leveraging machine learning, cloud storage, and embedded systems, the project enables users to input symptoms through text or voice, process them using a trained SVM model, and receive medical insights, including precautions, medications, diets, and workouts.

The project demonstrates the potential of AI and IoT in remote healthcare assistance, allowing users to obtain quick medical guidance without needing an immediate doctor consultation. The ESP32-based voice processing, combined with cloud integration, enhances the system's efficiency and accessibility.

Future improvements could include support for additional sensors, improved speech-to-text accuracy, and integration with medical databases for a more robust personalized healthcare assistant. Overall, this project serves as a stepping stone toward AI-powered, IoT-based smart healthcare solutions.

References

- **Material Design Guidelines by Google**: A comprehensive guide to building clean, user-friendly interfaces for web and mobile applications.
- **Deep Learning for Healthcare"**: This paper provides insights into applying machine learning techniques to medical datasets for disease prediction and health monitoring.
- **"Artificial Intelligence in Healthcare"** by Eric Topol: Discusses the use of AI and machine learning in modern healthcare applications, including personalized medicine and patient care systems.
- **"Healthcare AI: Applications of Artificial Intelligence for Medical Practices"**: A resource on the practical applications of AI in medical care and diagnostic systems.