# Chatbot Use Case – Smart Support Assistant

This document provides a step-by-step practical tutorial on building and running a Chatbot use case using Python, NLTK, Transformers, and Flask.

## Objective

To build an AI-powered chatbot that understands user queries, responds using a trained conversational model, and runs locally as a web app.

## Step 1 – Install Required Tools

Ensure Python 3.8+ is installed, then run:

* pip install nltk transformers torch flask

## Step 2 – Create a New Project Folder

mkdir chatbot\_project  
cd chatbot\_project  
  
chatbot\_project/  
│  
├── app.py  
├── chatbot.py  
└── templates/  
 └── index.html

## Step 3 – Build the Chatbot Logic (chatbot.py)

from transformers import pipeline  
  
chatbot = pipeline("conversational", model="microsoft/DialoGPT-medium")  
  
def get\_chatbot\_response(user\_input):  
 from transformers import Conversation  
 conversation = Conversation(user\_input)  
 response = chatbot(conversation)  
 return str(response.generated\_responses[-1])

## Step 4 – Create Flask Web App (app.py)

from flask import Flask, render\_template, request  
from chatbot import get\_chatbot\_response  
  
app = Flask(\_\_name\_\_)  
  
@app.route("/")  
def home():  
 return render\_template("index.html")  
  
@app.route("/get", methods=["GET"])  
def get\_bot\_response():  
 user\_input = request.args.get('msg')  
 response = get\_chatbot\_response(user\_input)  
 return response  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run(debug=True)

## Step 5 – Build Web Interface (templates/index.html)

<!DOCTYPE html>  
<html>  
<head>  
 <title>Smart Support Chatbot</title>  
 <style>  
 body { font-family: Arial; text-align: center; background: #f5f5f5; }  
 #chatbox { width: 500px; margin: auto; border: 1px solid #ccc; background: white; border-radius: 10px; padding: 20px; }  
 .userText, .botText { margin: 10px; padding: 10px; border-radius: 10px; }  
 .userText { background-color: #007bff; color: white; text-align: right; }  
 .botText { background-color: #eee; text-align: left; }  
 input { width: 80%; padding: 10px; border: 1px solid #ccc; border-radius: 5px; }  
 button { padding: 10px 15px; border: none; background: #007bff; color: white; border-radius: 5px; }  
 </style>  
</head>  
<body>  
 <h1>🤖 Smart Support Chatbot</h1>  
 <div id="chatbox">  
 <div id="chatlog"></div>  
 <input id="userInput" type="text" placeholder="Type your message..." />  
 <button onclick="sendMessage()">Send</button>  
 </div>  
 <script>  
 async function sendMessage() {  
 let userText = document.getElementById("userInput").value;  
 if (!userText) return;  
 let chatlog = document.getElementById("chatlog");  
 chatlog.innerHTML += `<div class='userText'>You: ${userText}</div>`;  
 document.getElementById("userInput").value = "";  
 let response = await fetch(`/get?msg=${userText}`);  
 let botReply = await response.text();  
 chatlog.innerHTML += `<div class='botText'>Bot: ${botReply}</div>`;  
 chatlog.scrollTop = chatlog.scrollHeight;  
 }  
 </script>  
</body>  
</html>

## Step 6 – Run the Chatbot Locally

Run in terminal:

* python app.py

Then open: http://127.0.0.1:5000

## Architecture

User → Web Interface (HTML/JS)  
 → Flask Server (app.py)  
 → Chatbot Model (chatbot.py using Hugging Face)  
 → Response Sent Back to UI

## Enhancements

🧠 Context Memory: Store past messages for context  
💬 Multiple Intents: Add small-talk or FAQ  
🎙️ Voice Integration: Use SpeechRecognition and gTTS  
💾 Logging: Save chat history to a database  
🌐 Deploy: Host on Render / Railway / HuggingFace Spaces

## Expected Output Example

User: Hello, who are you?  
Bot: Hi there! I’m your AI support assistant.  
  
User: Can you tell me about Python?  
Bot: Python is a popular programming language known for its simplicity and versatility.