```
"""
Enhanced Customer Support Chatbot for Electronic Gadgets Retail
"""

import json
from flask import Flask, request, jsonify, Response
import uuid
import re
from typing import Dict, List
import asyncio
from fuzzywuzzy import process
import random
from datetime import datetime

# Expanded FAQ database
FAQ_DATA = {
    "products": {
        "smartphone_x": {
            "name": "Smartphone X",
            "specs": "6.5-inch OLED, 8GB RAM, 128GB storage, 48MP camera",
            "warranty": "1 year manufacturer warranty",
            "price": "$699",
            "stock": "In stock (50+ units)"
        },
        "smartphone_y": {
            "name": "Smartphone Y",
            "specs": "6.7-inch AMOLED, 12GB RAM, 256GB storage, 64MP camera",
            "warranty": "1 year manufacturer warranty",
            "price": "$899",
            "stock": "In stock (20+ units)"
        },
        "laptop_pro": {
            "name": "Laptop Pro",
            "specs": "15-inch Retina, 16GB RAM, 512GB SSD, Intel i7",
            "warranty": "2 year extended warranty",
            "price": "$1499",
            "stock": "Low stock (5 units)"
        },
        "wireless_earbuds": {
            "name": "Wireless Earbuds",
            "specs": "True wireless, 20-hour battery life, noise cancellation",
            "warranty": "1 year warranty",
            "price": "$129",
            "stock": "In stock (100+ units)"
```

```
        }
    },
    "policies": {
        "return": "30-day return policy with original packaging and receipt. Refunds processed within 5-7 business days.",
        "payment_methods": "We accept Visa, Mastercard, PayPal, and bank transfers. No COD available.",
        "order_tracking": "Track your order using the tracking number in your confirmation email at our website's tracking page.",
        "shipping": "Free standard shipping on orders over $50. Express shipping available for $15 (2-3 business days).",
        "store_hours": "Online store: 24/7. Physical stores: 9 AM to 9 PM, Monday to Sunday.",
        "contact": "Call us at 1-800-555-TECH (8324) or email support@gadgetech.com"
    }
}

# Simulated conversation history storage
conversation_history: Dict[str, List[Dict]] = {}

# General message responses
GENERAL_RESPONSES = {
    "greeting": [
        "Hi there! How can I assist you with your electronic gadgets today?",
        "Hello! Ready to explore our latest tech products or need help with an order?",
        "Hey! What can I help you with - product info, orders, or something else?"
    ],
    "thanks": [
        "You're welcome! Let me know if you need anything else.",
        "Happy to help! What else can I assist you with today?",
        "No problem at all! Feel free to ask if you have more questions."
    ],
    "farewell": [
        "Goodbye! Have a great day!",
        "Thanks for chatting! Come back if you need more help.",
        "See you later! Happy gadget shopping!"
    ],
    "general": [
        "I'm not sure I understand. Could you rephrase that?",
        "Interesting question! Could you provide more details?",
        "I specialize in gadget-related questions. Could you ask about our products or policies?"
    ]
}

app = Flask(__name__)
```

```
def clean_input(text: str) -> str:
    """Clean user input by removing extra whitespace and special characters."""
    return re.sub(r'[^\w\s]', '', re.sub(r'\s+', ' ', text.strip()))

def fuzzy_product_search(query: str) -> str:
    """Search for products by partial name match using fuzzy matching."""
    product_names = {k: v["name"].lower() for k, v in FAQ_DATA["products"].items()}
    match = process.extractOne(query.lower(), product_names.values())
    if match and match[1] > 75:  # Confidence threshold
        for key, value in product_names.items():
            if value == match[0]:
                return key
    return None

def identify_intents(query: str) -> List[str]:
    """Identify multiple intents in the user's query based on keywords."""
    query = query.lower()
    intents = []

    # Product-related intents
    if any(word in query for word in ['spec', 'feature', 'configuration', 'details']):
        intents.append('product_specs')
    if 'warranty' in query:
        intents.append('warranty_info')
    if any(word in query for word in ['price', 'cost', 'how much']):
        intents.append('product_price')
    if any(word in query for word in ['stock', 'available', 'inventory']):
        intents.append('product_stock')

    # Order-related intents
    if any(word in query for word in ['track', 'order', 'delivery', 'shipment']):
        intents.append('order_tracking')
    if any(word in query for word in ['return', 'refund', 'exchange']):
        intents.append('return_policy')

    # General intents
    if any(word in query for word in ['payment', 'pay', 'card', 'method']):
        intents.append('payment_methods')
    if any(word in query for word in ['ship', 'delivery', 'arrive', 'time']):
        intents.append('shipping')
    if any(word in query for word in ['hours', 'open', 'close', 'time']):
        intents.append('store_hours')
    if any(word in query for word in ['contact', 'call', 'email', 'phone']):
```

```
        intents.append('contact_info')
    if any(word in query for word in ['hi', 'hello', 'hey', 'greetings']):
        intents.append('greeting')
    if any(word in query for word in ['thank', 'thanks', 'appreciate']):
        intents.append('thanks')
    if any(word in query for word in ['bye', 'goodbye', 'see you']):
        intents.append('farewell')

    return intents or ['general']

def get_faq_response(intent: str, context: Dict) -> str:
    """Retrieve FAQ response based on intent and context."""
    if intent == 'product_specs':
        product = context.get('product', 'smartphone_x')
        product_data = FAQ_DATA['products'].get(product, {})
        return f"{product_data.get('name', 'Product')} specifications: {product_data.get('specs', 'Not
available')}"

    elif intent == 'warranty_info':
        product = context.get('product', 'smartphone_x')
        product_data = FAQ_DATA['products'].get(product, {})
        return f"{product_data.get('name', 'Product')} warranty: {product_data.get('warranty', 'Not
available')}"

    elif intent == 'product_price':
        product = context.get('product', 'smartphone_x')
        product_data = FAQ_DATA['products'].get(product, {})
        return f"{product_data.get('name', 'Product')} price: {product_data.get('price', 'Not
available')}"

    elif intent == 'product_stock':
        product = context.get('product', 'smartphone_x')
        product_data = FAQ_DATA['products'].get(product, {})
        return f"Availability for {product_data.get('name', 'Product')}: {product_data.get('stock', 'Not
available')}"

    elif intent in FAQ_DATA['policies']:
        return FAQ_DATA['policies'][intent]

    elif intent in GENERAL_RESPONSES:
        return random.choice(GENERAL_RESPONSES[intent])

    return random.choice(GENERAL_RESPONSES['general'])
```

```
async def process_query(user_id: str, query: str) -> Dict:
    """Process user query, maintain context, and generate response."""
    query = clean_input(query)

    # Initialize conversation history if not exists
    if user_id not in conversation_history:
        conversation_history[user_id] = []

    # Extract context from previous interactions
    context = {}
    for entry in conversation_history[user_id][-3:]:
        context.update(entry.get('context', {}))

    # Update context with product search
    product_key = fuzzy_product_search(query)
    if product_key:
        context['product'] = product_key

    # Identify multiple intents
    intents = identify_intents(query)
    responses = []

    # Generate responses for each intent
    for intent in intents:
        response = get_faq_response(intent, context)
        responses.append(response)

    # Combine responses
    final_response = "\n".join(responses) if responses else
random.choice(GENERAL_RESPONSES['general'])

    # Store interaction
    conversation_history[user_id].append({
        'timestamp': datetime.now().isoformat(),
        'query': query,
        'response': final_response,
        'intents': intents,
        'context': context
    })

    return {
        'response': final_response,
        'intents': intents,
        'product': context.get('product'),
```

```
        'timestamp': datetime.now().isoformat()
    }

@app.route('/chat', methods=['POST'])
async def chat():
    """Handle chat requests via API."""
    try:
        data = request.get_json()
        if not data or 'query' not in data:
            return jsonify({'error': 'Invalid request format'}), 400

        user_id = data.get('user_id', str(uuid.uuid4()))
        query = data['query']

        if not query.strip():
            return jsonify({'error': 'Empty query'}), 400

        result = await process_query(user_id, query)
        return jsonify({
            'user_id': user_id,
            'response': result['response'],
            'intents': result['intents'],
            'product': result.get('product'),
            'timestamp': result['timestamp']
        })

    except Exception as e:
        return jsonify({'error': str(e)}), 500

@app.route('/history/<user_id>', methods=['GET'])
def get_history(user_id: str):
    """Get conversation history for a user."""
    if user_id in conversation_history:
        return jsonify({
            'user_id': user_id,
            'history': conversation_history[user_id]
        })
    return jsonify({'error': 'User not found'}), 404

@app.route('/reset', methods=['POST'])
async def reset():
    """Reset conversation history for a user."""
    try:
        data = request.get_json()
```

```
        user_id = data.get('user_id')

        if not user_id:
            return jsonify({'error': 'User ID required'}), 400

        if user_id in conversation_history:
            conversation_history[user_id] = []

        return jsonify({'message': 'Conversation reset', 'user_id': user_id})

    except Exception as e:
        return jsonify({'error': str(e)}), 500

@app.route('/')

def index():
    """Serve enhanced web interface."""
    return """<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>TechGadget Support Chat</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
</head>
<body class="bg-gray-100">
    <div class="container mx-auto max-w-4xl p-4">
        <div class="bg-white rounded-xl shadow-lg overflow-hidden">
            <div class="bg-blue-600 text-white p-4 flex items-center">
                <div class="w-10 h-10 rounded-full bg-blue-500 flex items-center justify-center mr-3">
                    <i class="fas fa-robot text-xl"></i>
                </div>
                <div>
                    <h1 class="text-xl font-bold">TechGadget Support</h1>
                    <p class="text-xs opacity-80">Virtual assistant for all your tech needs</p>
                </div>
                <button onclick="resetChat()" class="ml-auto bg-blue-700 hover:bg-blue-800 px-3
py-1 rounded-lg text-sm">
                    <i class="fas fa-sync-alt mr-1"></i> Reset
                </button>
            </div>
```

```
        <div id="chat" class="h-96 p-4 overflow-y-auto bg-gray-50">
            <div class="flex justify-start mb-4">
                <div class="bg-gray-200 text-gray-800 p-3 rounded-lg max-w-xs lg:max-w-md">
                    <div class="font-bold">TechGadget Bot</div>
                    <div>Hi there! I can help with product info, orders, returns, and more. What
would you like to know?</div>
                </div>
            </div>
        </div>

        <div class="p-4 border-t border-gray-200 bg-white">
            <div class="flex gap-2">
                <input id="query" type="text" placeholder="Type your question here..."
                    class="flex-1 p-3 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-blue-500">
                <button onclick="sendMessage()"
                    class="bg-blue-600 hover:bg-blue-700 text-white p-3 rounded-lg
focus:outline-none focus:ring-2 focus:ring-blue-500">
                    <i class="fas fa-paper-plane"></i>
                </button>
            </div>

            <div class="mt-3">
                <p class="text-sm text-gray-500 mb-2">Quick questions:</p>
                <div class="flex flex-wrap gap-2">
                    <button onclick="suggest('What are the specs of Smartphone Y?')"
                        class="bg-gray-200 hover:bg-gray-300 text-gray-800 px-3 py-1 rounded-full
text-sm">
                        Smartphone Y specs
                    </button>
                    <button onclick="suggest('How do I track my order?')"
                        class="bg-gray-200 hover:bg-gray-300 text-gray-800 px-3 py-1 rounded-full
text-sm">
                        Track order
                    </button>
                    <button onclick="suggest('What is the return policy?')"
                        class="bg-gray-200 hover:bg-gray-300 text-gray-800 px-3 py-1 rounded-full
text-sm">
                        Return policy
                    </button>
                    <button onclick="suggest('Contact information')"
                        class="bg-gray-200 hover:bg-gray-300 text-gray-800 px-3 py-1 rounded-full
text-sm">
                        Contact us
```

```
                </button>
            </div>
        </div>
    </div>
</div>

<script>
    let userId = localStorage.getItem('userId') || null;

    // Display loading indicator
    function showLoading() {
        const chat = document.getElementById('chat');
        const loadingId = 'loading-' + Date.now();
        chat.innerHTML += `
            <div id="${loadingId}" class="flex justify-start mb-4">
                <div class="bg-gray-200 text-gray-800 p-3 rounded-lg max-w-xs">
                    <div class="flex space-x-2">
                        <div class="w-2 h-2 rounded-full bg-gray-400 animate-bounce"></div>
                        <div class="w-2 h-2 rounded-full bg-gray-400 animate-bounce"
style="animation-delay: 0.2s"></div>
                        <div class="w-2 h-2 rounded-full bg-gray-400 animate-bounce"
style="animation-delay: 0.4s"></div>
                    </div>
                </div>
            </div>`;
        chat.scrollTop = chat.scrollHeight;
        return loadingId;
    }

    // Remove loading indicator
    function hideLoading(loadingId) {
        const element = document.getElementById(loadingId);
        if (element) element.remove();
    }

    async function sendMessage() {
        const queryInput = document.getElementById('query');
        const query = queryInput.value.trim();
        if (!query) return;

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

        // Add user message
```

```
chat.innerHTML += `
    <div class="flex justify-end mb-4">
        <div class="bg-blue-100 text-blue-800 p-3 rounded-lg max-w-xs lg:max-w-md">
            <div class="font-bold">You</div>
            <div>${query}</div>
        </div>
    </div>`;

queryInput.value = '';
chat.scrollTop = chat.scrollHeight;

const loadingId = showLoading();

try {
    const response = await fetch('/chat', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            user_id: userId,
            query: query
        })
    });

    if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
    }

    const data = await response.json();
    userId = data.user_id;
    localStorage.setItem('userId', userId);

    // Add bot response
    chat.innerHTML += `
        <div class="flex justify-start mb-4">
            <div class="bg-gray-200 text-gray-800 p-3 rounded-lg max-w-xs lg:max-w-md">
                <div class="font-bold">TechGadget Bot</div>
                <div>${data.response.replace(/\n/g, '<br>')}</div>
            </div>
        </div>`;

} catch (error) {
    console.error('Error:', error);
```

```
            chat.innerHTML += `
                <div class="flex justify-start mb-4">
                    <div class="bg-red-100 text-red-800 p-3 rounded-lg max-w-xs lg:max-w-md">
                        <div class="font-bold">Error</div>
                        <div>Sorry, there was a problem processing your request. Please try
again.</div>
                    </div>
                </div>`;
        } finally {
            hideLoading(loadingId);
            chat.scrollTop = chat.scrollHeight;
        }
    }

    async function resetChat() {
        if (!userId) return;

        try {
            const response = await fetch('/reset', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({
                    user_id: userId
                })
            });

            if (!response.ok) {
                throw new Error(`HTTP error! status: ${response.status}`);
            }

            document.getElementById('chat').innerHTML = `
                <div class="flex justify-start mb-4">
                    <div class="bg-gray-200 text-gray-800 p-3 rounded-lg max-w-xs lg:max-w-md">
                        <div class="font-bold">TechGadget Bot</div>
                        <div>Hi there! I can help with product info, orders, returns, and more. What
would you like to know?</div>
                    </div>
                </div>`;

        } catch (error) {
            console.error('Error resetting chat:', error);
        }
```

```
        }

        function suggest(query) {
            document.getElementById('query').value = query;
            sendMessage();
        }

        // Add event listener for Enter key in the input field
        document.getElementById('query').addEventListener('keypress', function(event) {
            if (event.key === 'Enter') {
                sendMessage();
            }
        });
    </script>
</body>
</html>
"""
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5003)
```

Github : https://github.com/venkatreddy13052005/chatbot