```python
import tkinter as tk
from tkinter import scrolledtext, messagebox
import re
from datetime import datetime

class AdvancedMedicalAssistant:
    def __init__(self):
        self.name = "MediAssist"

        # Comprehensive medical keywords
        self.medical_keywords = [
            'pain', 'ache', 'hurt', 'fever', 'cough', 'cold', 'headache',
            'stomach', 'nausea', 'vomit', 'diarrhea', 'constipation', 'dizzy',
            'fatigue', 'tired', 'weak', 'sore', 'throat', 'chest', 'back',
            'muscle', 'joint', 'rash', 'itch', 'burn', 'bleed', 'swelling',
            'symptom', 'sick', 'ill', 'disease', 'infection', 'allergy',
            'breathing', 'breath', 'asthma', 'diabetes', 'pressure', 'blood',
            'heart', 'kidney', 'liver', 'lung', 'skin', 'eye', 'ear',
            'nose', 'teeth', 'tooth', 'gum', 'anxiety', 'stress', 'depression',
            'sleep', 'insomnia', 'migraine', 'injury', 'sprain', 'fracture',
            'bruise', 'cut', 'wound', 'medication', 'medicine', 'drug', 'pill',
            'doctor', 'hospital', 'clinic', 'diagnose', 'treatment', 'remedy',
            'chills', 'sweating', 'runny nose', 'congestion', 'sneezing',
            'body ache', 'cramps', 'tender', 'numbness', 'tingling'
        ]

        # Disease pattern database with symptom combinations
        self.disease_patterns = {
            'common_cold': {
                'symptoms': ['runny nose', 'sneezing', 'sore throat', 'mild
fever', 'cough', 'congestion'],
                'match_threshold': 3,
                'likelihood': 'HIGH',
                'description': 'Viral infection of the upper respiratory tract'
            },
            'flu': {
                'symptoms': ['fever', 'body ache', 'fatigue', 'headache',
'cough', 'chills', 'sore throat'],
                'match_threshold': 4,
                'likelihood': 'HIGH',
                'description': 'Influenza viral infection'
            },
            'migraine': {
                'symptoms': ['headache', 'nausea', 'sensitivity to light',
'dizziness', 'vomit'],
                'match_threshold': 2,
                'likelihood': 'MODERATE',
                'description': 'Severe recurring headache disorder'
            },
            'gastroenteritis': {
                'symptoms': ['stomach pain', 'diarrhea', 'nausea', 'vomit', 
'fever', 'cramps'],
                'match_threshold': 3,
                'likelihood': 'HIGH',
                'description': 'Stomach and intestinal inflammation'
            },
            'dehydration': {
                'symptoms': ['dizzy', 'fatigue', 'headache', 'dry mouth', 'dark 
urine', 'weak'],
                'match_threshold': 3,
                'likelihood': 'MODERATE',
                'description': 'Excessive fluid loss from the body'
            },
            'tension_headache': {
                'symptoms': ['headache', 'muscle tension', 'stress', 'fatigue', 
'neck pain'],
                'match_threshold': 2,
                'likelihood': 'MODERATE',
                'description': 'Stress or tension-related headache'
            },
            'food_poisoning': {
                'symptoms': ['nausea', 'vomit', 'diarrhea', 'stomach pain', 
'cramps', 'fever'],
                'match_threshold': 3,
                'likelihood': 'MODERATE',
                'description': 'Illness from contaminated food'
            },
            'sinusitis': {
                'symptoms': ['headache', 'facial pain', 'congestion', 'runny 
nose', 'cough', 'fever'],
                'match_threshold': 3,
                'likelihood': 'MODERATE',
                'description': 'Inflammation of the sinuses'
            },
            'viral_infection': {
                'symptoms': ['fever', 'fatigue', 'body ache', 'headache', 
'chills'],
                'match_threshold': 3,
                'likelihood': 'HIGH',
                'description': 'General viral infection'
```

```python
            },
            'allergic_reaction': {
                'symptoms': ['rash', 'itch', 'swelling', 'runny nose',
'sneezing', 'watery eyes'],
                'match_threshold': 2,
                'likelihood': 'MODERATE',
                'description': 'Immune response to allergen'
            }
        }

        # Home care remedies database
        self.home_care_db = {
            'fever': [
                'Rest in a cool, comfortable room',
                'Drink plenty of fluids (water, herbal tea, broth)',
                'Take lukewarm sponge bath',
                'Wear light, breathable clothing',
                'Monitor temperature regularly'
            ],
            'headache': [
                'Rest in a quiet, dark room',
                'Apply cold compress to forehead',
                'Stay well hydrated',
                'Practice relaxation techniques',
                'Gentle neck and shoulder massage'
            ],
            'nausea': [
                'Sip clear fluids slowly',
                'Try ginger tea or peppermint tea',
                'Eat bland foods (crackers, toast)',
                'Avoid strong smells',
                'Get fresh air and rest'
            ],
            'body ache': [
                'Rest and avoid strenuous activity',
                'Apply warm compress to affected areas',
                'Take warm bath or shower',
                'Stay hydrated',
                'Gentle stretching exercises'
            ],
            'sore throat': [
                'Gargle with warm salt water',
                'Drink warm liquids with honey',
                'Use throat lozenges',
                'Stay hydrated',
                'Use humidifier'
            ],
            'cough': [
                'Stay hydrated with warm fluids',
                'Use honey (for adults)',
                'Use humidifier in room',
                'Avoid irritants and smoke',
                'Elevate head while sleeping'
            ],
            'stomach pain': [
                'Avoid solid foods initially',
                'Sip clear liquids',
                'Try ginger or peppermint tea',
                'Apply warm compress to abdomen',
                'Eat bland foods when ready (BRAT diet)'
            ],
            'diarrhea': [
                'Stay well hydrated',
                'Drink oral rehydration solution',
                'Eat bland, binding foods',
                'Avoid dairy and fatty foods',
                'Rest and avoid stress'
            ],
            'congestion': [
                'Use saline nasal spray',
                'Steam inhalation',
                'Use humidifier',
                'Stay hydrated',
                'Elevate head while sleeping'
            ]
        }

    def is_medical_query(self, text):
        """Check if query is medical-related"""
        text_lower = text.lower()

        for keyword in self.medical_keywords:
            if re.search(r'\b' + keyword + r'\b', text_lower):
                return True

        medical_patterns = [
            r'i (have|feel|am having|am feeling|got|experiencing)',
            r'my (head|stomach|throat|chest|back|body|neck)',
            r'feeling (sick|unwell|bad|terrible|ill)',
            r'what (could|might|should)',
            r'(symptoms?|condition|disease|illness)'
        ]

        for pattern in medical_patterns:
            if re.search(pattern, text_lower):
                return True

        return False

    def extract_symptoms(self, text):
        """Extract all symptoms from user input"""
        text_lower = text.lower()
        found_symptoms = []

        # Common symptom mappings
        symptom_map = {
            'head': 'headache',
            'headache': 'headache',
            'head pain': 'headache',
            'head ache': 'headache',

            'fever': 'fever',
            'temperature': 'fever',
            'high temp': 'fever',

            'nausea': 'nausea',
            'nauseated': 'nausea',
            'feel sick': 'nausea',
            'queasy': 'nausea',

            'vomit': 'vomit',
            'vomiting': 'vomit',
            'throw up': 'vomit',
            'throwing up': 'vomit',

            'body ache': 'body ache',
            'body pain': 'body ache',
            'aching': 'body ache',
            'muscle pain': 'body ache',

            'stomach': 'stomach pain',
            'stomach pain': 'stomach pain',
            'stomach ache': 'stomach pain',
            'belly': 'stomach pain',
            'abdominal': 'stomach pain',
            'sore throat': 'sore throat',
            'throat pain': 'sore throat',
            'throat': 'sore throat',

            'cough': 'cough',
            'coughing': 'cough',

            'dizzy': 'dizzy',
            'dizziness': 'dizzy',
            'lightheaded': 'dizzy',

            'fatigue': 'fatigue',
            'tired': 'fatigue',
            'exhausted': 'fatigue',
            'weak': 'weak',
            'weakness': 'weak',

            'runny nose': 'runny nose',
            'running nose': 'runny nose',
            'nasal': 'runny nose',

            'congestion': 'congestion',
            'congested': 'congestion',
            'stuffy': 'congestion',

            'sneezing': 'sneezing',
            'sneeze': 'sneezing',

            'chills': 'chills',
            'shivering': 'chills',

            'diarrhea': 'diarrhea',
            'loose stool': 'diarrhea',

            'rash': 'rash',
            'skin rash': 'rash',

            'back pain': 'back pain',
            'back ache': 'back pain',

            'chest pain': 'chest pain',
            'chest': 'chest pain',

            'cramps': 'cramps',
```

```python
            'cramping': 'cramps'
        }

        for keyword, symptom in symptom_map.items():
            if keyword in text_lower and symptom not in found_symptoms:
                found_symptoms.append(symptom)

        return found_symptoms

    def analyze_disease_patterns(self, symptoms):
        """Analyze symptoms to suggest possible conditions"""
        matches = []

        for disease, data in self.disease_patterns.items():
            match_count = 0
            disease_symptoms = data['symptoms']

            for symptom in symptoms:
                for disease_symptom in disease_symptoms:
                    if symptom.lower() in disease_symptom.lower() or
disease_symptom.lower() in symptom.lower():
                        match_count += 1
                        break

            if match_count >= data['match_threshold']:
                matches.append({
                    'disease': disease,
                    'match_count': match_count,
                    'likelihood': data['likelihood'],
                    'description': data['description']
                })

        # Sort by match count
        matches.sort(key=lambda x: x['match_count'], reverse=True)
        return matches

    def get_home_care_tips(self, symptoms):
        """Get relevant home care tips for symptoms"""
        all_tips = []
        covered_categories = set()

        for symptom in symptoms:
            for care_category, tips in self.home_care_db.items():
                if care_category in symptom.lower() and care_category not in
covered_categories:
```

```python
                    all_tips.extend(tips)
                    covered_categories.add(care_category)

        # Remove duplicates while preserving order
        seen = set()
        unique_tips = []
        for tip in all_tips:
            if tip not in seen:
                seen.add(tip)
                unique_tips.append(tip)

        return unique_tips[:8]  # Limit to 8 tips

    def generate_comprehensive_response(self, symptoms):
        """Generate detailed medical response following required structure"""
        if not symptoms:
            return self.general_prompt()

        response = "=" * 60 + "\n"
        response += "🛡️ MEDICAL ASSESSMENT REPORT\n"
        response += "=" * 60 + "\n\n"

        # 1. UNDERSTANDING THE SYMPTOMS
        response += "📋 UNDERSTANDING YOUR SYMPTOMS:\n"
        response += "-" * 60 + "\n"
        response += f"You are experiencing: {', '.join(symptoms)}\n\n"

        # 2. POSSIBLE CAUSES
        response += "🔍 POSSIBLE CONDITIONS (NOT A DIAGNOSIS):\n"
        response += "-" * 60 + "\n"

        possible_diseases = self.analyze_disease_patterns(symptoms)

        if possible_diseases:
            for idx, disease in enumerate(possible_diseases[:4], 1):
                disease_name = disease['disease'].replace('_', ' ').title()
                response += f"\n{idx}. {disease_name} [{disease['likelihood']}
likelihood]\n"
                response += f"   • {disease['description']}\n"
                response += f"   • Matching symptoms: {disease['match_count']}\n"
        else:
            response += "\nBased on your symptoms, you may be experiencing:\n"
            response += "• General illness or infection\n"
            response += "• Temporary discomfort\n"
            response += "• Physical stress or overexertion\n"
        response += "\n⚠️  NOTE: These are POSSIBLE causes only, NOT confirmed
diagnosis.\n\n"

        # 3. HOME CARE & PAIN RELIEF TIPS
        response += "💊 HOME CARE & PAIN RELIEF TIPS:\n"
        response += "-" * 60 + "\n"

        home_tips = self.get_home_care_tips(symptoms)
        if home_tips:
            for idx, tip in enumerate(home_tips, 1):
                response += f"{idx}. {tip}\n"
        else:
            response += "• Rest and get adequate sleep\n"
            response += "• Stay well hydrated\n"
            response += "• Eat nutritious, balanced meals\n"
            response += "• Avoid stress and overexertion\n"
            response += "• Monitor your symptoms\n"

        response += "\n"

        # 4. WARNING SIGNS (RED FLAGS)
        response += "🚨 SEEK IMMEDIATE MEDICAL ATTENTION IF:\n"
        response += "-" * 60 + "\n"
        response += "• Symptoms suddenly worsen or become severe\n"
        response += "• High fever (above 103°F/39.4°C) that won't go down\n"
        response += "• Difficulty breathing or chest pain\n"
        response += "• Severe dehydration (dark urine, extreme thirst)\n"
        response += "• Persistent vomiting or inability to keep fluids down\n"
        response += "• Signs of confusion or loss of consciousness\n"
        response += "• Symptoms persist beyond 3-5 days without improvement\n"
        response += "• Any symptom that concerns you significantly\n\n"

        # 5. DOCTOR CONSULTATION REMINDER
        response += "👨‍⚕️ IMPORTANT DOCTOR CONSULTATION REMINDER:\n"
        response += "-" * 60 + "\n"
        response += "✓ Please consult a qualified healthcare professional\n"
        response += "✓ A doctor can provide accurate diagnosis through
examination\n"
        response += "✓ Only a medical professional can prescribe medication\n"
        response += "✓ Don't delay seeking professional help if needed\n"
        response += "✓ This assessment is for informational purposes only\n\n"

        # 6. MEDICAL DISCLAIMER
        response += "⚖️  MEDICAL DISCLAIMER:\n"
        response += "-" * 60 + "\n"
        response += "This information is for educational purposes only and does
NOT\n"
        response += "constitute medical advice, diagnosis, or treatment. Always
seek\n"
        response += "the advice of your physician or qualified healthcare
provider\n"
        response += "with any questions regarding a medical condition. Never
disregard\n"
        response += "professional medical advice or delay seeking it because
of\n"
        response += "information provided by this chatbot.\n\n"

        response += "=" * 60 + "\n"
        response += "💬 Feel free to describe more symptoms or ask follow-up
questions!\n"
        response += "=" * 60 + "\n"

        return response

    def general_prompt(self):
        """Initial prompt for user"""
        return """══════════════════════════════════════════════════════════
🛡️ WELCOME TO MEDIASSIST - YOUR MEDICAL ASSISTANT
══════════════════════════════════════════════════════════

I can help analyze your symptoms and provide health guidance!

To get started, please describe your symptoms. For example:
• "I have headache, fever, and body pain"
• "I'm experiencing nausea and stomach pain"
• "I have a sore throat and cough"

📝 For better assessment, you can also mention:
• Duration: How long have you had these symptoms?
• Severity: Mild, moderate, or severe?
• Age group: Child, adult, or senior?

I will provide:
✓ Analysis of your symptoms
✓ Possible conditions (not diagnosis)
✓ Safe home care tips
✓ Warning signs to watch for
✓ Doctor consultation advice
```

```
⚠️  IMPORTANT REMINDERS:
• I provide information only, NOT medical diagnosis
• I cannot prescribe medications or dosages
• Always consult a healthcare professional
• This is NOT a replacement for professional medical care

Let me know your symptoms and I'll help guide you! 💝
═════════════════════════════════════════════════════════
"""

    def non_medical_response(self):
        """Response for non-medical queries"""
        return "Sorry, this is not my field. I can only help with medical-related 
questions."

    def chat(self, user_input):
        """Main chat processing"""
        if not user_input.strip():
            return "Please describe your symptoms or health concerns."

        if not self.is_medical_query(user_input):
            return self.non_medical_response()

        symptoms = self.extract_symptoms(user_input)
        return self.generate_comprehensive_response(symptoms)


class MedicalAssistantGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("⛨ MediAssist - Advanced Medical Assistant")
        self.root.geometry("900x700")
        self.root.configure(bg="#e8f4f8")

        self.bot = AdvancedMedicalAssistant()

        # Header
        header_frame = tk.Frame(root, bg="#2c3e50", height=100)
        header_frame.pack(fill=tk.X, side=tk.TOP)

        title = tk.Label(
            header_frame,
            text="⛨ MediAssist",
            font=("Arial", 22, "bold"),
            bg="#2c3e50",
            fg="white",
            pady=10
        )
        title.pack()

        subtitle = tk.Label(
            header_frame,
            text="Advanced Medical Symptom Analysis & Health Guidance",
            font=("Arial", 11),
            bg="#2c3e50",
            fg="#ecf0f1"
        )
        subtitle.pack()

        # Chat Area
        chat_frame = tk.Frame(root, bg="#e8f4f8")
        chat_frame.pack(fill=tk.BOTH, expand=True, padx=15, pady=15)

        self.chat_display = scrolledtext.ScrolledText(
            chat_frame,
            wrap=tk.WORD,
            font=("Consolas", 10),
            bg="#ffffff",
            fg="#2c3e50",
            padx=15,
            pady=15,
            state=tk.DISABLED,
            relief=tk.FLAT,
            borderwidth=2
        )
        self.chat_display.pack(fill=tk.BOTH, expand=True)

        # Configure tags
        self.chat_display.tag_config("user", foreground="#e74c3c", font=("Arial",
11, "bold"))
        self.chat_display.tag_config("bot", foreground="#27ae60",
font=("Consolas", 10))
        self.chat_display.tag_config("time", foreground="#95a5a6", font=("Arial",
8))

        # Input Area
        input_frame = tk.Frame(root, bg="#e8f4f8")
        input_frame.pack(fill=tk.X, padx=15, pady=(0, 15))

        input_label = tk.Label(
```

```python
            input_frame,
            text="Describe your symptoms:",
            font=("Arial", 10, "bold"),
            bg="#e8f4f8",
            fg="#34495e"
        )
        input_label.pack(anchor=tk.W, pady=(0, 5))

        input_container = tk.Frame(input_frame, bg="#e8f4f8")
        input_container.pack(fill=tk.X)

        self.user_input = tk.Entry(
            input_container,
            font=("Arial", 12),
            bg="white",
            fg="#2c3e50",
            relief=tk.SOLID,
            borderwidth=1
        )
        self.user_input.pack(side=tk.LEFT, fill=tk.X, expand=True, padx=(0, 10))
        self.user_input.bind("<Return>", lambda e: self.send_message())

        send_btn = tk.Button(
            input_container,
            text="📤 Send",
            command=self.send_message,
            font=("Arial", 11, "bold"),
            bg="#3498db",
            fg="white",
            relief=tk.FLAT,
            padx=25,
            pady=8,
            cursor="hand2"
        )
        send_btn.pack(side=tk.LEFT, padx=(0, 5))

        clear_btn = tk.Button(
            input_container,
            text="🗑️ Clear",
            command=self.clear_chat,
            font=("Arial", 11, "bold"),
            bg="#e74c3c",
            fg="white",
            relief=tk.FLAT,
            padx=25,
            pady=8,
            cursor="hand2"
        )
        clear_btn.pack(side=tk.LEFT)

        # Footer
        footer = tk.Label(
            root,
            text="⚠️ DISCLAIMER: For informational purposes only. Not a
substitute for professional medical advice. Always consult a healthcare
provider.",
            font=("Arial", 9),
            bg="#e8f4f8",
            fg="#7f8c8d",
            wraplength=850,
            justify=tk.CENTER
        )
        footer.pack(pady=(0, 10))

        # Show welcome message
        self.display_message("bot", self.bot.general_prompt())
        self.user_input.focus()

    def send_message(self):
        user_text = self.user_input.get().strip()

        if not user_text:
            return

        self.display_message("user", user_text)
        self.user_input.delete(0, tk.END)

        bot_response = self.bot.chat(user_text)
        self.display_message("bot", bot_response)

    def display_message(self, sender, message):
        self.chat_display.config(state=tk.NORMAL)

        timestamp = datetime.now().strftime("%I:%M %p")

        if sender == "user":
            self.chat_display.insert(tk.END, f"\n[{timestamp}] ", "time")
            self.chat_display.insert(tk.END, "YOU: ", "user")
            self.chat_display.insert(tk.END, f"{message}\n")
        else:
```

```python
            self.chat_display.insert(tk.END, f"\n[{timestamp}] ", "time")
            self.chat_display.insert(tk.END, "MEDIASSIST:\n", "bot")
            self.chat_display.insert(tk.END, f"{message}\n", "bot")

        self.chat_display.see(tk.END)
        self.chat_display.config(state=tk.DISABLED)

    def clear_chat(self):
        if messagebox.askyesno("Clear Chat", "Clear all conversation history?"):
            self.chat_display.config(state=tk.NORMAL)
            self.chat_display.delete(1.0, tk.END)
            self.chat_display.config(state=tk.DISABLED)
            self.display_message("bot", self.bot.general_prompt())


def main():
    root = tk.Tk()
    app = MedicalAssistantGUI(root)
    root.mainloop()


if __name__ == "__main__":
    main()
```