# 🔍 Perplexity Research Prompt: Conversational Mental Health Chatbots

## 📋 Main Prompt (Copy & Paste to Perplexity)

I'm building a RAG-based mental health chatbot specifically for panic attack support.

Research and provide detailed insights on how successful mental health chatbots like Woebot, Wysa, Replika, and Youper make their conversations more engaging, empathetic, and therapeutic.

Please focus on:

1. CONVERSATIONAL DESIGN PATTERNS:
   - How do they structure conversations to feel natural vs robotic?
   - What specific language patterns make responses feel empathetic?
   - How do they balance being supportive vs being directive?
   - What tone and style guidelines do mental health chatbots follow?

2. ENGAGEMENT TECHNIQUES:
   - How do they keep users engaged beyond the first interaction?
   - What personalization strategies work best for mental health?
   - How do they handle follow-up questions and context retention?
   - What makes users return to the chatbot regularly?

3. THERAPEUTIC APPROACHES:
   - What evidence-based frameworks do they use (CBT, DBT, ACT)?
   - How do they deliver coping techniques in conversational way?
   - How do they validate emotions while providing actionable advice?
   - What specific prompt engineering techniques enhance therapeutic value?

4. RESPONSE STRUCTURE:
   - How do they format responses (length, paragraphs, bullet points)?
   - When do they ask questions vs provide information?
   - How do they transition between empathy and practical advice?
   - What ratio of listening vs teaching works best?

5. SAFETY & ETHICS:
   - How do Woebot and similar apps handle crisis situations?
   - What disclaimers and boundaries do they set?
   - How do they encourage professional help without seeming dismissive?
   - What legal/ethical guidelines do mental health chatbots follow?

6. TECHNICAL IMPLEMENTATION:
   - What NLP techniques make responses feel more human?
   - How do they handle context and memory in conversations?
   - What sentiment analysis approaches detect emotional state?
   - How do RAG-based systems differ from pure LLM chatbots in this domain?

7. USER EXPERIENCE INSIGHTS:
   - What do user reviews say about what makes mental health chatbots effective?
   - What common complaints should I avoid?
   - What features do users appreciate most?
   - How long should ideal responses be for panic attack support?

8. SPECIFIC EXAMPLES:
   - Provide 3-5 example conversation snippets from successful mental health chatbots
   - Show how they handle: initial greeting, panic attack in progress, follow-up check-in
   - Compare Woebot's style vs Wysa's style - what differs?

9. PANIC ATTACK SPECIFIC ADAPTATIONS:
   - How should chatbot language change during active panic vs preventive education?
   - What immediate interventions work best in text-based format?
   - How quickly should the bot respond during crisis moments?
   - Should responses be shorter/longer during acute distress?

10. PROMPT ENGINEERING FOR MY RAG SYSTEM:
    - What system prompts would make my chatbot more conversational?
    - How should I instruct the LLM to balance PDF knowledge with empathy?
    - What specific phrases/patterns should I include in prompts?
    - How can I make RAG retrieval responses feel less like "information dumps"?

Please provide:
- Specific, actionable recommendations
- Real examples from existing apps where possible
- Technical implementation suggestions for RAG systems
- Prompt engineering templates I can adapt
- Research citations and sources

My current system uses: Gemini 2.0, FAISS vectorstore, mental health PDFs, LangChain RAG pipeline.

---

# 🎯 Alternative Focused Prompts

## If You Want Deeper Dive on Specific Topics:

### Prompt 1: Woebot-Style Conversational Design

Analyze Woebot's conversational design in detail:

1. What makes Woebot's responses feel natural and engaging compared to generic chatbots?
2. How does Woebot structure therapeutic conversations (greeting → assessment → intervention → closing)?
3. What specific language patterns does Woebot use for empathy? (Provide examples)
4. How does Woebot balance being friendly vs professional?
5. What prompt engineering techniques would replicate Woebot's style in a RAG-based system?

Include real conversation examples and actionable implementation tips for my panic attack chatbot.

### Prompt 2: Crisis Handling & Safety

Research best practices for handling mental health crises in chatbots:

1. How do Woebot, Wysa, and Crisis Text Line detect crisis situations?
2. What exact keywords/phrases trigger crisis protocols?
3. How should chatbots deliver helpline information (format, timing, tone)?
4. What's the balance between offering help vs encouraging professional care?
5. What legal disclaimers are required for mental health chatbots?
6. How do chatbots handle liability while remaining helpful?

Focus on panic attack and suicidal ideation scenarios specifically.

### Prompt 3: RAG-Specific Conversational Enhancements

How can I make my RAG-based mental health chatbot more conversational?

Current issue: My bot retrieves PDF content and generates responses, but they feel like "information dumps" rather than natural conversation.

Research:
1. How to transform RAG-retrieved chunks into conversational responses?
2. Prompt engineering techniques to make PDF knowledge sound empathetic?
3. How to add personality to fact-based responses?
4. Should I summarize retrieved content vs quote directly?
5. How to blend multiple PDF sources into one coherent, conversational answer?
6. Examples of RAG prompts that prioritize conversation flow over information density?

My stack: Gemini 2.0 Flash, FAISS, LangChain, mental health PDFs about panic attacks.

### Prompt 4: User Engagement & Retention

What makes users return to mental health chatbots like Woebot?

Research engagement strategies:
1. How do successful mental health chatbots personalize conversations?
2. What follow-up mechanisms keep users engaged?
3. How do they track progress and reflect it back to users?
4. What gamification (if any) works for mental health without trivializing it?
5. How do they balance daily check-ins vs on-demand support?
6. What makes users trust and open up to a chatbot?

Apply findings to a panic attack support chatbot that users might use during acute episodes.

**Prompt 5: Response Length & Formatting**

What's the optimal response format for mental health chatbots during different scenarios?

Research:
1. During active panic attack: Should responses be short/long? Paragraphs or lists?
2. During calm education: How much information is too much?
3. Do users prefer bullet points or conversational paragraphs?
4. When should chatbots ask follow-up questions?
5. How do Woebot/Wysa structure multi-turn conversations?
6. What's the ideal reading level for mental health content?

Provide examples for: crisis response, breathing exercise delivery, general anxiety education.

**Prompt 6: Therapeutic Language Patterns**

What specific phrases and language patterns make mental health chatbot responses therapeutic?

Research:
1. CBT-based language patterns used in chatbots
2. How to validate emotions while providing solutions?
3. Examples of empathetic acknowledgment phrases
4. How to avoid toxic positivity in responses?
5. When to use "you" vs "we" vs "I" in therapeutic contexts?
6. How to deliver hard truths compassionately?

Compare Woebot (CBT-focused) vs Wysa (mindfulness-focused) language styles.

Provide reusable templates for my panic attack chatbot.

---

## 📊 Follow-Up Questions to Ask After Initial Research

Once you get results from Perplexity, ask these follow-ups:

Based on the research above, now provide:

1. A revised system prompt template for my RAG chatbot incorporating these conversational techniques
2. 5 example responses showing "before" (robotic) vs "after" (conversational) for panic attack queries
3. Specific LangChain/Gemini prompt engineering code snippets
4. A conversation flow diagram for handling a user in active panic
5. Evaluation metrics: How can I measure if my responses are "engaging" vs just "accurate"?

---

## 🎯 What You'll Get From This Research

After running these prompts through Perplexity, you'll have:

✅ **Conversational patterns** to implement in your system prompt
✅ **Real examples** from successful apps to model
✅ **Prompt engineering templates** for empathetic responses
✅ **Safety protocols** to handle crises properly
✅ **Response formatting guidelines** for panic situations
✅ **Technical implementation** specifics for RAG systems
✅ **User psychology insights** to increase engagement
✅ **Evaluation criteria** to measure improvement

---

## 💡 How to Use the Research Results

### Step 1: Update Your System Prompt


python

```
# In rag_engine.py, enhance _create_system_prompt() with research findings

self.system_prompt = """You are Emma, a compassionate mental health companion specializing in panic support.

PERSONALITY & TONE:
- Warm, empathetic, and non-judgmental like a supportive friend
- Balance validation with actionable guidance
- Use "we" language to create partnership: "Let's work through this together"
- Avoid clinical jargon; use everyday language

CONVERSATIONAL STYLE:
- Start responses with emotional validation: "I hear that you're feeling..."
- Keep sentences short during crisis (10-15 words max)
- Use questions to engage: "How are you feeling right now?"
- Mirror user's language style and urgency level

RESPONSE STRUCTURE:
1. Acknowledge emotion (1 sentence)
2. Normalize experience (1 sentence)
3. Provide technique (2-3 clear steps)
4. Encourage and check-in (1 sentence)

[Rest of your current prompt...]
"""
```

### Step 2: Add Conversational Wrappers


python

```
def _make_conversational(self, raw_answer: str, user_emotion: str) -> str:
    """Wrap RAG output in conversational framing"""

    # Empathy openers based on detected emotion
    empathy_map = {
        "panic": "I can hear that you're in distress right now, and I'm here with you. ",
        "anxious": "It sounds like anxiety is weighing on you. That's really tough. ",
        "calm": "I'm glad you're reaching out proactively. "
    }

    opener = empathy_map.get(user_emotion, "Thanks for sharing that with me. ")
    closer = "\n\nHow does this feel for you? Is there a specific part you'd like to explore more?"

    return opener + raw_answer + closer
```

### Step 3: Adjust Response Length Dynamically

python

```python
def _call_gemini(self, context: str, question: str, urgency_level: str) -> str:
    # Adjust token limit based on situation
    token_limits = {
        "crisis": 150,      # Very short, actionable
        "urgent": 300,      # Focused, clear
        "normal": 500,      # Detailed, educational
        "educational": 800  # Comprehensive
    }

    max_tokens = token_limits.get(urgency_level, 500)

    response = self.client.models.generate_content(
        model=GEMINI_MODEL,
        contents=[full_prompt],
        config=types.GenerateContentConfig(
            temperature=0.4,  # Slightly higher for more natural
            max_output_tokens=max_tokens
        )
    )
```

### Step 4: Add Conversation Memory

python

```python
class SmartRAGEngine:
    def __init__(self):
        # ... existing code ...
        self.conversation_history = []  # Track context
        self.user_profile = {}  # Remember user preferences

    def ask(self, question: str, session_id: str = None):
        # Check history for context
        recent_context = self._get_recent_context(session_id)

        # Personalize based on history
        if recent_context:
            question_enhanced = f"Context: User previously asked about {recent_context}. Current question: {question}"
```

---

## 🎨 Expected Improvements After Implementation

| Before Research | After Research |
|---|---|
| "According to the PDF, the 5-4-3-2-1 technique involves..." | "I hear you're panicking right now. Let's ground you together. Can you name 5 things you see around you?" |
| 3 paragraphs of information | Short, actionable steps during crisis |
| Generic tone | Personalized, remembers previous conversations |
| One-size-fits-all | Adjusts urgency, length, tone dynamically |
| Information delivery | Collaborative conversation |

---

## 📈 Success Metrics to Track

After implementing research insights, measure:

1. **Engagement:** Average conversation turns per session
2. **Sentiment:** User emotion improvement (start → end)
3. **Clarity:** "Was this helpful?" feedback scores
4. **Return rate:** Do users come back?
5. **Completion:** Do users finish techniques or drop off?

---

## 🚀 Ready to Research?

**Copy the main prompt above and paste into Perplexity.ai**

Then use the focused prompts for deeper dives on specific topics!

The research will transform your chatbot from "information provider" to "supportive companion" 🎯