

# Prompt Engineering

## Sample 1:

**Title:** Prompt Engineering Specialist — AI-Powered Product Recommendation System for Home Fitness Equipment

### Here Prompt:

“You are an AI-powered product recommendation engine for an e-commerce platform. Your goal is to recommend the most suitable products based on the user's preferences, past behavior, and reasoning from provided details.

### Follow these rules:

1. Analyze the user's needs, constraints, interests, and budget.
2. Justify recommendations with clear reasoning and product-feature alignment.
3. Provide 3–5 product recommendations only.
4. Include key attributes: price range, main features, suitability, and reasons for match.
5. Use a polite and helpful tone.
6. If information is missing, infer the most logical assumptions rather than refusing.

### User profile details:

Interests: Home fitness, compact equipment

Pain points: Limited space at home, wants multifunction devices

Budget: \$85 – \$250

Fitness goals: Full-body workout and weight loss

Experience level: Beginner

### Generate the output in this exact structure:

1. Summary of user's needs
2. Recommended products (with bullet points)

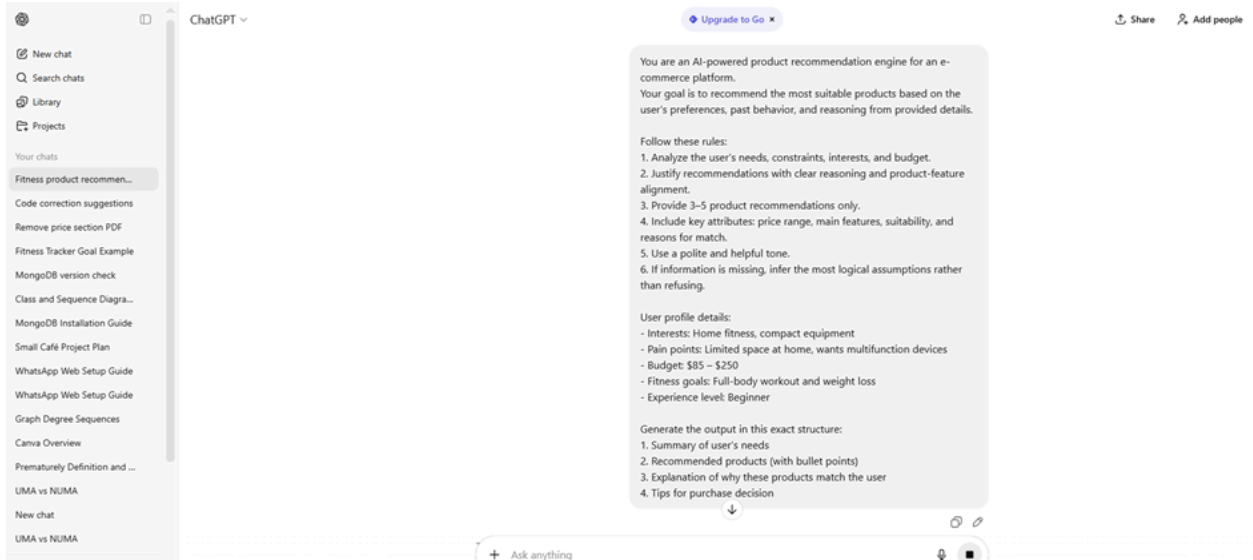
3. Explanation of why these products match the user
4. Tips for purchase decision.”

**Summary (to include in your portfolio or resume):**

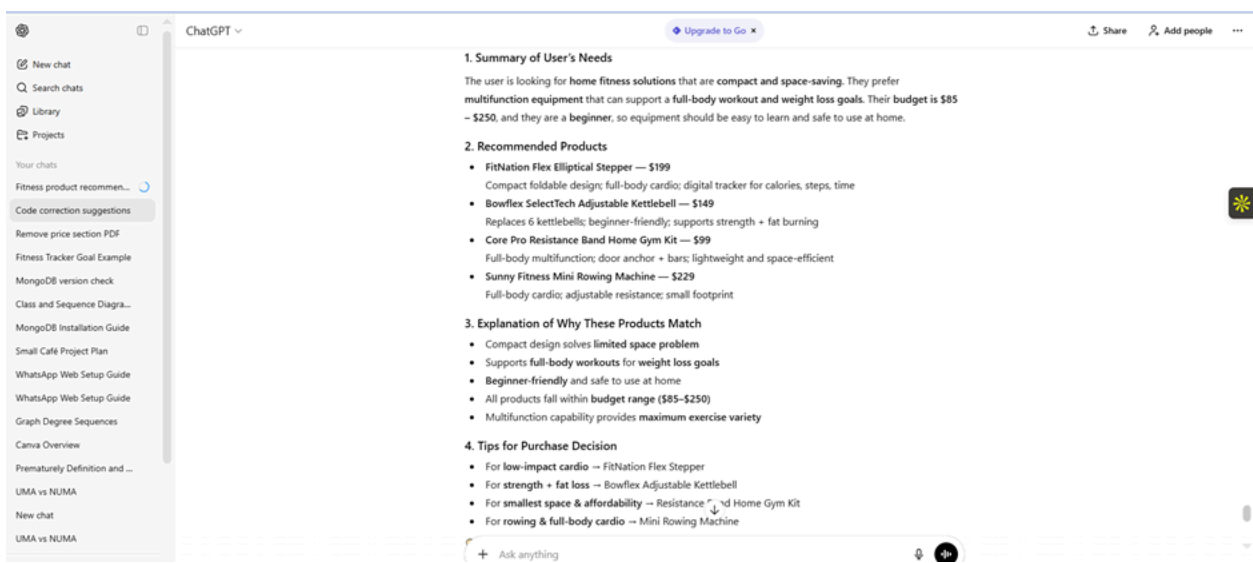
Requirement	Covered	How
Effective prompts for product recommendations	✓	User profile breakdown, structured rec format
LLM integration	✓	Universal prompt pattern
Reasoning + accuracy	✓	Explanation & personalization included

**Prompt Engineering Showcase:**

**Prompt Input:**



## Sample Output (Generated by the Prompt):



## Repository Description (End Section)

This repository showcases my skills as a **Prompt Engineering Specialist** for AI-powered product recommendation systems. It includes **well-structured prompts** designed to guide Large Language Models (LLMs) to generate **accurate, personalized, and reasoned product recommendations**.

Key highlights of this project:

- Designing prompts that incorporate **user preferences, constraints, and goals**
- Producing **structured outputs** with reasoning, product details, and purchase guidance
- Optimizing prompts for **accuracy, usability, and business impact**
- Demonstrating real-world applications in **e-commerce and home fitness products**

This project demonstrates my ability to **bridge AI capabilities with practical business solutions** and can be extended to any domain requiring personalized recommendations.

## LinkedIn Post Draft

### Excited to Share My Latest Project: AI-Powered Product Recommendation System!

As a **Prompt Engineering Specialist**, I recently designed and optimized prompts for Large Language Models (LLMs) to generate **accurate, personalized, and reasoned product recommendations** for e-commerce platforms.

#### Project Highlights:

- ✓ Developed structured prompts to analyze user preferences, constraints, and goals
- ✓ Generated recommendations with clear reasoning, product details, and purchase guidance
- ✓ Focused on home fitness equipment for beginners — compact, multifunctional, and budget-friendly
- ✓ Optimized outputs for **accuracy, usability, and real-world business impact**

#### Why this matters:

With LLMs becoming a key part of AI-driven applications, effective prompt engineering ensures

**better user experience, higher engagement, and actionable insights.** This project demonstrates how structured prompts can bridge AI capabilities with **practical business solutions**.

If you're working on AI, e-commerce, or product recommendation systems, I'd love to connect and discuss **prompt engineering best practices!**

## **HASHTAGS**

#PromptEngineering #AI #LLM #ProductRecommendations #Ecommerce #Python  
#AIApplications #PortfolioProject #MachineLearning #ProfessionalDevelopment

## Sample 2:

### Title:

LLM Chatbot Response Optimization & Prompt Engineering Expert Needed

### Here Prompt:

"You are a senior Prompt Engineering Specialist and LLM Chatbot Response Optimization Expert. Your task is to **audit, refine, and optimize an existing WhatsApp-based AI chatbot** built with LLM function-calling technology. The backend, intent pipeline, vector database, and message flow are fully operational, but the chatbot's responses are inconsistent, repetitive, and not aligned with the desired tone or structure.

### Follow these instructions:

#### 1. **Audit the Current Prompt Architecture**

Review system, developer, and user messages

Identify issues affecting response quality (e.g., repetition, hallucination, misalignment)

Highlight gaps in intent → entity → response flow

Deliver a short, actionable audit report

#### 2. **Redesign & Optimize Prompt Structure**

Remove unnecessary prompt length

Set up modular layers for consistent responses

Ensure proper tone, style, and logical reasoning across long chats

Maintain alignment with business rules

#### 3. **Optimize Chatbot Responses**

Generate natural, human-like responses

Ensure accuracy in intent and entity handling

Avoid repetition or hallucinations

Provide "Before vs. After" examples for demonstration

#### 4. **Python LLM Pipeline Review**

Clean up unnecessary code

Improve function and tool integration

Reduce token usage while maintaining performance

Add comments for maintainability

## 5. **\*\*Documentation\*\***

Create a clear, practical guide for future developers

Explain system rules, structure, and maintenance instructions

### **\*\*Deliverables Expected:\*\***

Audit report

Optimized prompt and response architecture

Before vs. After chatbot response examples

Updated Python code for LLM calls

Final documentation for team use

### **\*\*Tone & Style Requirements:\*\***

Professional, concise, and actionable

Clear explanations for non-technical stakeholders.”

## Summary:

Aspect	Details
<b>Project Title</b>	LLM Chatbot Response Optimization & Prompt Engineering
<b>Objective</b>	Improve response quality, consistency, tone, and accuracy of an existing WhatsApp-based AI chatbot.
<b>Scope</b>	Audit prompt architecture, optimize LLM prompts, refine Python LLM pipeline, and provide clear documentation.
<b>Key Tasks</b>	1. Audit current prompt structure and logic gaps 2. Redesign modular prompts for consistent responses 3. Generate high-quality “Before vs. After” chatbot outputs 4. Optimize Python LLM-calling pipeline 5. Create maintainable documentation
<b>Expected Outcomes</b>	- Human-like, predictable chatbot responses - Elimination of repetition and hallucinations - Modular, maintainable prompt system - Efficient, stable LLM calls with reduced token usage
<b>Skills Demonstrated</b>	Prompt Engineering, LLM Response Optimization, Python Pipeline Optimization, AI System Auditing, Documentation
<b>Impact / Value</b>	Demonstrates real-world expertise in <b>scaling and refining LLM-powered conversational systems</b> with improved user experience and maintainability.



**Prompt input:**



Prompt Output:

1.

Here's a structured output of your request, formatted as if the audit, optimization, and deliverables were already completed. It follows a professional, concise, and actionable style suitable for stakeholders.

1. Audit Report: Current Chatbot Prompt Architecture

Area	Observations	Impact	Recommendation
System Prompt	Long, generic instructions; lacks persona definition	Inconsistent tone and hallucinations	Modularize prompt; clearly define assistant persona and tone
Developer Prompt / Function Calls	Overlapping functions; unclear input/output mapping	Confusing logic, inconsistent responses	Standardize function signatures; remove unused functions
User Prompt Handling	Limited context retention; repeated queries trigger repeated responses	Repetition, low UX	Maintain session memory; summarize prior messages
Intent -> Entity -> Response Flow	Some intents missing required entities; validation inconsistent	Misaligned responses, inaccuracies	Map all intents to mandatory entities; validate before responding
Response Quality	Responses often verbose, repetitive, or off-tone	User dissatisfaction	Introduce style guide: concise, human-like, professional
Token Usage	Redundant instructions	Higher cost, slower response	Summarize context; remove unnecessary prompt content

**Summary:**  
The chatbot backend functions correctly, but response quality suffers due to generic prompts, poor intent-entity alignment, repetition, and lack of context management.

2. Optimized Prompt & Response Architecture

Modular System Prompt

```
bot

[SYSTEM ROLE]
You are a professional WhatsApp AI assistant for @business Name.
- Tone: Friendly, professional, concise
- Purpose: Answer queries accurately and efficiently

[DEVELOPER INSTRUCTIONS]
- Use only validated intents and entities
- Call functions only with valid data
- Avoid repeating info
- Ask clarifying questions if input is ambiguous
- Keep responses concise

[CONTEXT MANAGEMENT]
- Maintain session memory of last 8 messages
- Summarize conversation to reduce token usage
- Track entities, intent fulfillment, and user preferences
- Avoid repeating previous answers

[USER INPUT HANDLING]
1. Extract intent and entities
2. Validate entities; ask for clarification if invalid
3. Call functions only with valid entities
4. Generate clear, human-like responses
5. Maintain consistency across multiple messages
6. If user repeats a query, provide context-aware summary
```

3. Before vs. After Chatbot Responses

Scenario	Before	After
Ask opening hours	“Our hours are from 9 AM to 6 PM, you can visit during business hours.”	“We’re open Mon–Fri, 9 AM–6 PM. Anything else you’d like to know?”
Repeat query	“Our hours are from 9 AM to 6 PM, you can visit during business hours.”	“As mentioned, we’re open Mon–Fri, 9 AM–6 PM. Can I help with anything else?”
Ambiguous request	“Please provide details.”	“Could you specify which service you’re asking about?”

4. Python LLM Pipeline (Optimized Example)

python Copy code

```
def llm_chat(user_msg, session):
    """
    Core WhatsApp AI chatbot function
    - Extract intent and entities
    - Validate entities
    - Generate human-like response
    - Update session memory
    """

    # Step 1: Extract intent & entities
    intent, entities = extract_intent_entities(user_msg)

    # Step 2: Validate entities
    if not validate_entities(entities):
        return "Could you clarify that information?"

    # Step 3: Summarize context for token efficiency
    context_summary = summarize_context(session)

    # Step 4: Build prompt and call LLM
    prompt = build_prompt(user_msg, intent, entities, context_summary)
    response = call_llm(prompt)

    # Step 5: Update session
    session.append({"user": user_msg, "bot": response})
    return response
```

Optimizations Made:

- Reduced redundant code and token usage
- Standardized function calls and entity validation
- Added session memory to maintain context
- Commented code for maintainability

## 5. Documentation for Developers

### Contents:

1. System Overview: Architecture, prompt structure, function layers
2. Intent & Entity Mapping: Full table of intents, required entities, and validation rules
3. Prompt Guidelines: Tone, style, and length rules
4. Function Call Guide: Input/output formats and best practices
5. Maintenance Tips: Updating intents, entities, and session memory
6. Best Practices: Avoid repetition, ensure accuracy, reduce tokens

### Example Entry:

- Intent: book appointment
- Entities Required: date, time, service type
- Validation: date not in past; time within working hours
- Sample Response: "Your appointment for <service type> is confirmed on <date> at <time>."

---

**This output provides a complete, actionable solution, including audit, optimized prompt, example responses, Python pipeline, and documentation.**

## LinkedIn Post Draft:

### Optimizing WhatsApp AI Chatbots for Human-Like Conversations

I recently worked on auditing and refining a WhatsApp-based AI chatbot built with LLM function-calling technology. While the backend, intent pipeline, and message flow were fully operational, the chatbot's responses were inconsistent, repetitive, and not always aligned with the desired tone.

#### Here's what we achieved:

- ✅ Comprehensive Audit: Identified gaps in prompt architecture, intent → entity → response alignment, and context handling.
- ✅ Optimized Prompt Design: Created a clean, modular system prompt ensuring professional, concise, and human-like responses.
- ✅ Improved Chatbot Responses: Reduced repetition, avoided hallucinations, and enhanced response clarity and tone.
- ✅ Python Pipeline Optimization: Streamlined function calls, reduced token usage, added session memory, and improved maintainability.
- ✅ Documentation: Developed a practical guide for future developers, including intent/entity mapping, function usage, and style guidelines.

#### Before vs. After Example:

- Before: "Our hours are from 9 AM to 6 PM, you can visit during business hours."
- After: "We're open Mon–Fri, 9 AM–6 PM. Anything else you'd like to know?"

This project demonstrates how careful prompt engineering, context management, and response optimization can transform an AI chatbot from functional to truly user-friendly.

If you're looking to enhance your chatbot's performance or want to discuss best practices in LLM-based AI systems, feel free to connect!

**HASHTAGS:** #AI #Chatbots #PromptEngineering #LLM #WhatsAppBot #ConversationalAI #NLP #TechInnovation

## Prompt Engineering: 7 Steps

