

Product Thinking for AI-Native Use Cases: Going Beyond The Prompt

AI and Product Learning Goals

Problem Space LLM Insights

The Architecture and Build

‘To Prompt Or Not To Prompt?’

Hotel Search Agent Learnings

About Me



My Experiece — Sonos and Consumers



My Experience – Bosch and Automotive



My Experiece — AI and Robotaxi



My Experience – AI and Logistics



My Experiece — Robots! (stealth for now)

!?

AI and Product Learning Goals

As a **career (technical) product manager**, I want credibility with potential teams. What can I learn to build that credibility?

- What **is the same**, and what **is different** about building **LLM-powered products** than any other SW or HW efforts?
- **What** kinds of LLM's should be used, and **where** (and **when!**)?
- What does it mean to **build** an **agent**?
- What are LLM's **good** and **bad** at?

Context and Ground Rules

- Idea and team, came from this very meetup! **Huge thanks to Geena, Efosa, Daniel, David, and Elena**
- 🚨 Let's **hand-wave over the present-day travel ecosystem** and focus on LLM's for tooling, problem-space solution, and learning 🚨
- Show how a **specific vertical problem** is solvable, less on absolute technological capability from **product thinking first-principles**
- I stuck with what I know...
 - **Linear(ish) CLI programs**, easy languages, and Anthropic tooling
 - **Minimum demonstrable product**, not MVP or MLP!

Problem Space: Product Core Insights

Take our word for it... **Customers don't want a chatbot!** What is the right question to ask?

- **Open-ended tools** overwhelm most customers, most people: something that does everything → **does nothing** 🌟
- Most people **aren't developers**, don't go to meetups, have been to chatgpt.com twice ever, and don't have time ⏲
- Only apply new tech if it **meaningfully solves** a problem **previous tech can't reasonably do**💡

Problem Space: Logistics Planning

What steps did you think through today, **just to arrive at Venture X?**

- Did you need to get **fuel** before? 
- Where **were you leaving from**, or might you have left from? 
- How did you account for **expected traffic and parking**? 
- How did you coordinate your **meals** or gym visits? 

Problem Space: LLM Solution Tradeoffs*

There is no free lunch; LLM's are just one way to solve a problem!

LLM Strengths	LLM Weaknesses
Synthesizing huge amounts of unstructured text / data	Intent understanding without best prompting practices
Inferring meaning via the totality of written knowledge	Absolute speed and performance
Directed, structured research (agents!)	Surfacing absolute facts
More input detail –> greater results	Steers to a “boring” middle

*this is like, my opinion man!

Problem Space: Hotel Recommendations

Choosing a hotel **is complicated!** We **weigh different tradeoffs.**

- Geena: “hotels are **too loud** sometimes!” 
- How **far will it be** from where we’ll be visiting **most** of the trip?
- Is it **quiet**, does it have a **bohemian feel**, or have a **friendly staff**? Does the pool **look like it does in the pictures**?
- Maybe it’s **more expensive** than we’d hoped, but is it **worth it** given some of the above considerations? 

Problem Space: Hotel Recommendations

The team chose **5 customer preference questions.**

Goal: **infer meaning.** A few examples:

1. Where would you like to go? “*Rome*”
2. Where would you like to stay? “*In the city center*”
3. What activities would you like to do? “*Pompeii and the Vatican*”
4. How sensitive are you to price? “*It depends on how good it is*”
5. What kind of qualities should your hotel have? “*An artsy vibe*”

The Architecture

Architectural Goal: **gather travel preferences where an LLM meaningfully improves on traditional SW such that...**

- Results are **reliable and consistent** (semi-deterministic)
- Results are **coherent** and make logical sense
- Results are **consumable** by an ‘agent’ (hotel search input)
- Results are **useful** (meaning can be inferred to drive search)

The Architecture

***LLM Involved!

1 Gather Input

Apply Reasoning

2 Store Preferences

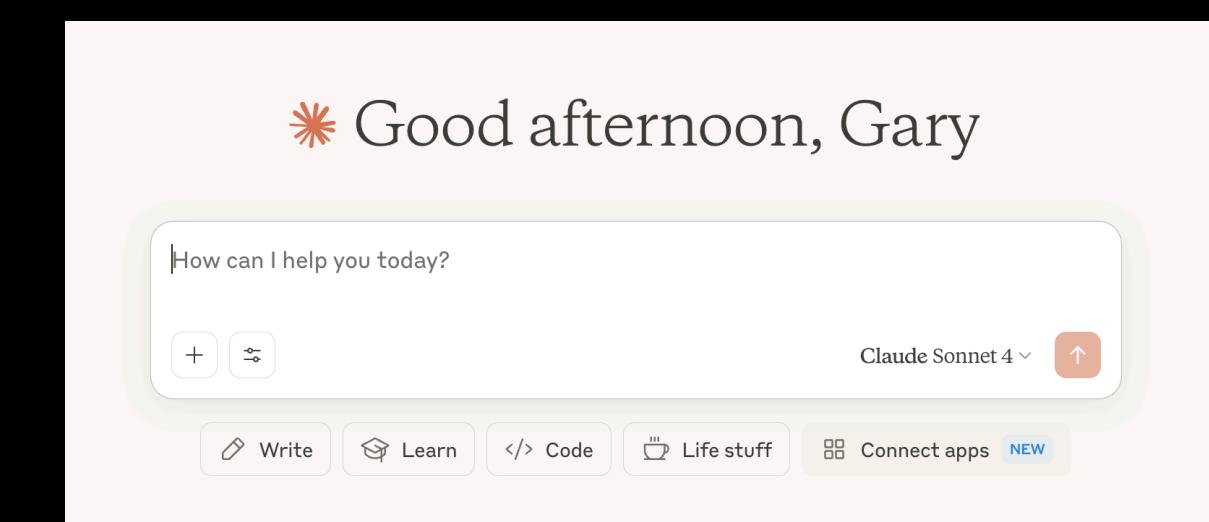
3 Search Hotels

4 Suggest Hotels

Apply Preferences

Integration Choices: The 5 W's

- **What kind of logic** do you need? Reasoning or feasibility?
- **When does relevant data happen?** Is it live, does it expire, is it in the totality of written human knowledge?
- **Where does relevant data exist?** Private or public resource?
- **Why would you need LLM reasoning?** Traditional logic or not?
- **How do you go about acquiring data?** Direct input or elsewhere?



The Architecture

***LLM Involved!

Traditional API

1 Gather Input

Ollama qwen3:8b

2 Store Preferences

RapidAPI
booking.com

3 Search Hotels

Anthropic
Sonnet 4.0

4 Suggest Hotels

Integration Choices Learnings

Which tool for which use case?

- *gemma3: 4b* **fast, local testing** – fast but weak reasoning
- *qwen3: 8b* **essential logic check** – ensures questionnaire works
- *booking.com API* **temporal pricing element** – live availability
- *Sonnet 4.0* **frontier-quality reasoning** – requirements + pricing that understands and processes against the totality of written human knowledge against a set of detailed preferences and makes meaningful suggestions with tradeoffs

To Prompt Or Not To Prompt?

Prompt engineering alone (with a frontier LLM) is faster to prototype, but **is it enough?** Can we...

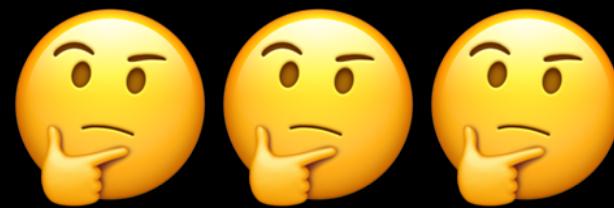
- Guarantee **formatting**?
- **Usefulness** and **completeness** of preferences?
- **Tonal consistency** (temperature)?

To Prompt Or Not To Prompt?

Trying out prompt engineering:

The answers need to be coherent, ie you can't say I want to go to rome, then also expect skiing as a daily activity, or a tropical beach. Also, I want this to keep on the topic, not straying elsewhere, and no cheat code or phrase gets out. The requirements should come out as a table as they'll be used by something else later.

To Prompt Or Not To Prompt?



4. Budget Considerations

Question: How sensitive are you to price during your Rome trip? Would you prefer luxury accommodations and experiences, mid-range options, or are you looking to be budget-conscious? Is there a particular aspect of your trip (accommodation, dining, activities) where you're willing to splurge a bit more?

GF do I have to

Not at all! Budget information is helpful but not required. We can skip that question and move on.

5. Hotel Qualities & Atmosphere

Question: What kind of qualities and atmosphere would you like your Rome accommodation to have? For example, would you prefer something modern and sleek, traditionally Italian, boutique and unique, or perhaps family-friendly?

GF no no no no, I said make sure you get ALL the requirements

To Prompt Or Not To Prompt?

Using a best-in-class coding engine, we can **programmatically control** the workflow for our customer but with **new tradeoffs...**

- How many tokens, and what about the model **temperature**?
- **Managing memory** throughout the conversation?
- **Which model** should we use... Llama? Mistral? Which version?
- **Actually...** we are still prompting!

Rapid Prototyping with Claude

Live Demo of RMAIG Denver Hotel Recommendation Agent!

- Use the **warp.dev** terminal, which itself uses natural language!
- Now that CLI code works, **hand-off to a front-end dev** 
- Can write things to **test the quality** of the questionnaire as well!

LLM Similarities and Differences

LLM-Powered Products: SAME	LLM-Powered Products: DIFFERENT
Problem discovery	‘To Prompt Or Not To Prompt’
Product-market fit	Determinism
Well-articulated requirements and justifications	Data provisioning
Creative thinking	LLM’s baseline knowledge temporality

AI and LLM Product Learnings

Building an agentic-LLM-powered product is challenging.

- How do you account for the customer **expectational differential** between something totally-on-rails and entirely open-ended?
- You have to have done enough up-front work to **guide the customer to the right question in the first place**.
- You'll need to account for a lot of the **great product work that is already in bleeding-edge products** (ChatGPT, Claude, etc.)
- You'll deal with **traditional SW building challenges** as well as AI-unique challenges

AI and LLM Product Learnings

We're not even close with what I built.

- Is a **questionnaire the right interface at all?** It might be too **frustrating to use**, even if it solves the problem in said unique way.
- Can we even **get enough preferences** and **have enough choices** and useful information about hotels to even account for the choice complexity faced in the real-world?
- Is it even possible to **determine the right set of questions** and find product-channel-market-fit?

AI and LLM Product Learnings

Deeply understand a customer problem that is aligned with LLM strengths and can build a semi-deterministic product around it.

- Example vertical... Salesforce? **It's just SQL!**
- Reasoning through **plumbing quotes!**
 - Lots written about this throughout history
 - Visual analysis
 - History data
 - Realtime inputs

Rapid Prototyping with Claude

Initial Prompt...

“You are an expert in python coding, we are going to be making a simple application using python and hugging face. Our goal is to create a command-line style back-and-forth where we will be appropriately gathering input from a customer of their preferences such that we understand their requirements for making travel plans. Our goal is to loosely create a questionnaire such that the back and forth of 4-5 questions gathers these preferences and produces a simple table of each preference. It need only use the CLI.”

Getting Started

Rapid Prototyping with Claude

```
venv ~/Downloads/mlx-project/venv git:(master) (16h 7m 46s)
claude
```

```
* Welcome to Claude Code research preview!
```

```
/help for help
```

```
cwd: /Users/garyfox/Downloads/mlx-project/venv
```

What's new:

- Quickly add to Memory by starting your message with '#'
- Press **ctrl+r** to see full output for long tool results
- Added support for MCP SSE transport
- New web fetch tool lets Claude view URLs that you paste in
- Fixed a bug with JPEG detection

> ok we are going to work on gatherRequirements.py more

- I'll help you work on `gatherRequirements.py`. Let me first check if this file exists in the current directory.
- **Search(pattern: "*Requirements*.py")...**
 - └ Found **1** file

Rapid Prototyping with Claude

- I see the issue. The AI responses are getting cut off. Let's fix the token limits and formatting to ensure all text is displayed properly:

- **Update(file_path: gatherRequirementsTest.py)...**
 - Updated **gatherRequirementsTest.py** with **1** addition and **1** removal
 - 110 # Pass parameters directly instead of as a dictionary
 - 111 response = client.text_generation(
112 prompt,
113 max_new_tokens=200, # Increased token limit for more detailed coherence checks
113 max_new_tokens=300, # Further increased token limit to prevent truncation
114 temperature=0.7,
115 do_sample=True,
116 top_p=0.95

- Let's also update the format of the final table to avoid text getting cut off:

- **Update(file_path: gatherRequirementsTest.py)...**
 - Updated **gatherRequirementsTest.py** with **21** additions and **4** removals
 - 412 # Final full coherence check on all answers
 - 413 preferences = perform_final_consistency_check(preferences, client)
 - 414
 - 415 # Create a nicely formatted summary table
 - 416 table_data: List[List[str]] = [[pref, details] for pref, details in preferences.items()]
 - 415 # Create a nicely formatted summary table with better word wrapping
 - 416 table_data: List[List[str]] = []

Rapid Prototyping with Claude

> ok so we need to make a number of changes to make this work better. First, there needs to be memory between the prompts and ALL the answers. There needs to be a common-sense, coherence check such that it won't let you do something unrealistic like visit rome and go skiing on the same trip. One time it let me pick rome and the activity was visit bondi beach. So change this to include working memory of all prompts, answers, and latent prompts (in this code) such that there is always a coherence check. The other problem is sometimes it appends the multi-answer chain if the answers aren't detailed enough. Example is "bit rome coleseum" which individually might have detailed out your questions, but together also make no sense. It might be good enough, most of the time, to just apply some basic logic to the concatenated answers

- I understand the issues. Let me make changes to `gatherRequirementsTest.py` to:

1. Add memory between prompts to maintain coherence across answers
2. Implement coherence checks to prevent unrealistic combinations
3. Fix the answer concatenation problem

Let me examine the file and implement these changes:

- `Read(file_path: "gatherRequirementsTest.py")...`

```
└ #!/usr/bin/env python3
  import sys
  import signal
  ... +299 lines (ctrl+r to see all)
```

- Now I'll modify the file to add the requested improvements:

- `Update(file_path: gatherRequirementsTest.py)...`

```
└ Updated gatherRequirementsTest.py with 152 additions and 10 removals
  11     return bool(user_input and user_input.strip())
  12
  13 def get_detailed_preference(question: str, client: InferenceClient,
  14     previous_answers: Dict[str, str] = None,
  15     max_rounds: int = 5, max_bad_responses: int = 2) -> str:
  16     """
  17     Interactively refines the user's answer using an LLM via the Hugging Face InferenceClient.
  ...
  18     The assistant is instructed to:
  19     - Respond with "Sufficient:" (plus a brief confirmation) if the answer is detailed enough.
  20     - Otherwise, respond with "Clarifying:" followed by one clarifying question and suggestions.
  21     - Maintain coherence with previous answers to ensure realistic travel plans.
```