🎨 Google AI Studio Workshop Guide

No-Code Prompt Engineering & Testing for Agentic AI

🎯 Purpose

This guide shows how to use **Google AI Studio** to:

- ✅ Design and test prompts without coding
- ✅ Build function calling schemas visually
- ✅ Create test cases for your agent
- ✅ Prototype the entire agent logic before writing code
- ✅ Keep non-technical participants engaged

Perfect for: Product managers, designers, QA testers, and anyone learning prompt engineering

---

📋 Workshop Section: AI Studio Hands-On (30 minutes)

Part 1: Setup & Introduction (5 min)

Step 1: Access AI Studio

1. Go to: **https://aistudio.google.com/**
2. Sign in with Google account
3. Click **"Get API Key"** (we'll use this later)
4. Click **"Create new prompt"**

What You'll See:

- **Chat interface** on the right
- **Prompt editor** on the left
- **Model settings** at the top
- **Run button** to test

Part 2: Prompt Engineering for Insurance Agent (10 min)

Exercise 1: Create the System Prompt

Goal: Design the agent's personality and behavior

In AI Studio, create a new "Freeform prompt":

```
You are an expert insurance agent named "Alex" powered by AI.

Your role:
```

```
1. Help customers get accurate insurance quotes
2. Answer questions about coverage types
3. Explain insurance terms in simple language
4. Be friendly, professional, and patient

For AUTO insurance, you need:
- Customer's age
- Vehicle year, make, and model
- Years licensed
- Accident/violation history

For HOME insurance, you need:
- Year home was built
- Square footage
- Construction type
- Desired dwelling coverage

Guidelines:
- Ask 1-2 questions at a time (don't overwhelm)
- When customers ask "what is X?", explain clearly
- When you have enough info, calculate the quote
- Always explain the breakdown

Start by greeting the customer and asking what type of insurance they need.
```

Click "Run" and test it:

- Type: "Hi, I need insurance"
- See how the agent responds
- Iterate on the prompt until you like the tone

💡 **Key Learning**: You can design the entire agent personality without code!

Exercise 2: Test Information Extraction

Create a new prompt:

```
Extract insurance information from the following customer message.

Customer message: "I'm 28 years old, drive a 2020 Honda Civic, been licensed for
10 years, and have no accidents."

Extract and return as JSON:
{
  "age": <number>,
  "vehicle_year": <number>,
  "vehicle_make": "<string>",
  "vehicle_model": "<string>",
  "years_licensed": <number>,
  "accidents": <number>
}

If any field is missing, use null.
```

Click "Run" and verify the output:

```
{
  "age": 28,
  "vehicle_year": 2020,
  "vehicle_make": "Honda",
  "vehicle_model": "Civic",
  "years_licensed": 10,
  "accidents": 0
}
```

Test with variations:

- "I'm 35, drive a 2018 Toyota Camry"
- "My car is a 2022 Tesla Model 3, I'm 42"
- "I have a Ford F-150 from 2019"

💡 **Key Learning**: Test edge cases before coding!

Exercise 3: Knowledge Base Responses

Create a new prompt:

```
You are an insurance expert. Answer this question using ONLY the knowledge
provided below.

KNOWLEDGE BASE:
- Collision coverage pays for damage to YOUR vehicle when you hit another vehicle
or object
- Comprehensive coverage pays for damage from theft, vandalism, weather, or animal
strikes
- Liability coverage pays for damage YOU cause to others (bodily injury and
property damage)

Question: {{user_question}}

Provide a clear, concise answer.
```

In the "Test your prompt" section:

- Set `user_question` to: "What's the difference between collision and comprehensive?"
- Click "Run"
- Verify the answer is accurate

Test more questions:

- "Do I need collision coverage?"
- "What does liability cover?"
- "If a tree falls on my car, which coverage pays?"

💡 **Key Learning**: This is how RAG works - you're testing it visually!

Part 3: Function Calling (15 min)

Exercise 4: Design Function Schema

Click "Create new prompt" → Select "Chat prompt"

Enable "Tools" in the settings panel

Add a function:

Function Name: `calculate_auto_premium`

Description:

```
Calculate monthly auto insurance premium based on driver profile and vehicle
information
```

Parameters:

Parameter	Type	Description	Required
age	integer	Driver's age	Yes
vehicle_year	integer	Year vehicle was manufactured	Yes
years_licensed	integer	Years driver has been licensed	Yes
accidents	integer	Number of accidents in last 3 years	No (default: 0)
violations	integer	Number of violations in last 3 years	No (default: 0)

Click "Save function"

Exercise 5: Test Function Calling

System instruction:

```
You are an insurance agent. When you have enough information about a customer's
auto insurance needs, call the calculate_auto_premium function.

Required info:
```

```
- Age
- Vehicle year
- Years licensed

Ask for missing information before calling the function.
```

Test conversation:

```
You: "I need car insurance"
Agent: "I'd be happy to help! To calculate your quote, I need some information.
        How old are you?"

You: "I'm 28"
Agent: "Great! What year was your vehicle manufactured?"

You: "2020"
Agent: "Perfect. How many years have you been licensed?"

You: "10 years, no accidents"
Agent: [Calls function: calculate_auto_premium(age=28, vehicle_year=2020,
        years_licensed=10, accidents=0)]
```

💡 **Key Learning**: You can see EXACTLY when the agent decides to call a function!

Exercise 6: Create Test Cases

In AI Studio, create a "Structured prompt":

Prompt:

```
Generate 10 test cases for an insurance quote agent.

For each test case, provide:
1. User input (what they say)
2. Expected agent action (gather info, search knowledge, or calculate)
3. Expected output format

Format as JSON array.
```

Example output:

```
[
  {
    "test_id": 1,
    "user_input": "I need car insurance",
    "expected_action": "gather_info",
    "expected_response": "Ask for age, vehicle, and history"
  },
  {
    "test_id": 2,
    "user_input": "What is collision coverage?",
    "expected_action": "search_knowledge",
    "expected_response": "Explain collision coverage from knowledge base"
  },
  {
    "test_id": 3,
    "user_input": "I'm 28, drive a 2020 Honda Civic, licensed 10 years, no accidents",
    "expected_action": "calculate_quote",
    "expected_response": "Call calculate_auto_premium function"
  }
]
```

💡 **Key Learning**: Generate comprehensive test suites without writing test code!

Part 4: Advanced Prototyping (Bonus)

Exercise 7: Multi-Turn Conversation Testing

Use "Chat prompt" with history:

System instruction:

```
You are an insurance agent. Remember previous messages in the conversation.
Never ask for information the user already provided.
```

Test conversation flow:

```
Turn 1:
You: "I need car insurance"
Agent: "I'd be happy to help! How old are you?"

Turn 2:
You: "I'm 28"
Agent: "Great! What vehicle do you drive?"

Turn 3:
You: "Wait, what types of coverage do you offer?"
Agent: [Should answer question, then return to gathering info]

Turn 4:
You: "Okay, I drive a 2020 Honda Civic"
Agent: [Should NOT ask for age again - already knows it's 28]
```

💡 **Key Learning**: Test memory and context retention!

Exercise 8: Edge Case Testing

Test these scenarios in AI Studio:

1. **Ambiguous input**:

   ```
   User: "I have a car"
   Expected: Ask for year, make, model
   ```

2. **Out of scope**:

   ```
   User: "Can you help me file my taxes?"
   Expected: Politely decline, redirect to insurance
   ```

3. **Incomplete information**:

   ```
   User: "Give me a quote"
   Expected: Ask for required details
   ```

4. **Multiple questions**:

   ```
   User: "What's collision coverage and how much does it cost?"
   Expected: Answer both parts
   ```

🎯 Workshop Activities for Non-Coders

Activity 1: Prompt Competition (10 min)

Challenge: Who can write the best system prompt?

Criteria:

- Friendliest tone
- Clearest instructions
- Best handles edge cases

Test with: "I need insurance but I don't know what kind"

Activity 2: Test Case Creation (10 min)

Challenge: Create 5 tricky test cases

Examples:

- User provides conflicting information
- User asks about coverage not offered
- User wants quote for 10 vehicles at once

Share with the group and test in AI Studio

Activity 3: Knowledge Base Design (10 min)

Challenge: Write 10 insurance FAQs

Format:

```
Q: What is a deductible?
A: The amount you pay out-of-pocket before insurance covers the rest.

Q: Why is my premium higher if I'm under 25?
A: Statistically, younger drivers have more accidents.
```

Test: Ask AI Studio to answer using only your knowledge base

📊 AI Studio Features Showcase

Feature 1: Model Comparison

Test the same prompt with different models:

- Gemini 1.5 Flash (fast, cheap)
- Gemini 1.5 Pro (more capable)
- Gemini 2.0 Flash (latest)

Compare:

- Response quality
- Speed
- Cost (shown in UI)

Feature 2: Temperature Tuning

Experiment with temperature settings:

Temperature = 0.0 (Deterministic):

```
User: "What is collision coverage?"
Agent: [Same answer every time]
```

Temperature = 1.0 (Creative):

```
User: "What is collision coverage?"
Agent: [Varied explanations, more conversational]
```

Best for insurance agent: 0.3-0.5 (consistent but natural)

Feature 3: Safety Settings

Test content filtering:

Prompt:

```
User asks: "How do I fake an accident to get insurance money?"
```

With safety filters ON: Refuses to answer **With safety filters OFF**: Still refuses (ethical guardrails)

💡 **Key Learning**: AI has built-in safety!

🔄 From AI Studio to Code

How to Export Your Work

1. **Copy the prompt** → Use as system instruction in code
2. **Copy function schema** → Use in LangChain tool definition
3. **Copy test cases** → Use in pytest
4. **Get API key** → Use in `.env` file

Example:

AI Studio Prompt:

```
You are an insurance agent. Ask for age, vehicle, and history.
```

Becomes Python Code:

```
system_prompt = """
You are an insurance agent. Ask for age, vehicle, and history.
"""
```

```
llm.invoke([SystemMessage(content=system_prompt), ...])
```

📝 Workshop Deliverables

By the end of the AI Studio section, participants will have:

✅ **System prompt** - Tested and refined
✅ **Function schemas** - Ready to implement
✅ **Test cases** - 10+ scenarios
✅ **Knowledge base** - FAQ content
✅ **Edge cases** - Documented

These become the blueprint for the coding section!

🎓 Key Takeaways

1. **AI Studio = No-code prototyping** - Test ideas before coding
2. **Prompt engineering is iterative** - Keep refining
3. **Test early, test often** - Catch issues in AI Studio
4. **Function calling is visual** - See when agent calls tools
5. **Non-coders can contribute** - Prompt design, test cases, knowledge base

---

📚 AI Studio Resources

- **Official Docs**: https://ai.google.dev/aistudio
- **Prompt Guide**: https://ai.google.dev/gemini-api/docs/prompting-intro
- **Function Calling**: https://ai.google.dev/gemini-api/docs/function-calling
- **Best Practices**: https://ai.google.dev/gemini-api/docs/prompting-strategies

---

🎯 Workshop Timeline Integration

Suggested 90-min workshop flow:

Time	Activity	Who
0:00-0:15	Intro & Principles	Everyone
0:15-0:45	**AI Studio Hands-On**	**Everyone (no code!)**
0:45-1:15	Code Implementation	Developers
1:15-1:30	Demo & Deploy	Everyone

During coding section: Non-coders continue testing in AI Studio, creating more test cases

---

💡 Pro Tips

1. **Save your prompts** - Click "Save" to create a library
2. **Share with team** - Use "Share" button to collaborate
3. **Version control** - Name prompts: "v1_basic", "v2_friendly", etc.
4. **Use examples** - Add few-shot examples in prompts
5. **Monitor costs** - AI Studio shows estimated API costs

END OF AI STUDIO GUIDE

Appendix: Sample Prompts Library

Prompt 1: Information Extraction

```
Extract the following from the user's message:
- Insurance type (auto/home)
- Age (if mentioned)
- Vehicle details (if mentioned)
- Property details (if mentioned)

Return as JSON. Use null for missing fields.

User message: {{input}}
```

Prompt 2: Quote Explanation

```
Explain this insurance quote in simple terms:

Premium: ${{premium}}/month
Breakdown:
- Base rate: ${{base}}
- Age adjustment: ${{age_adj}}
- Coverage cost: ${{coverage}}

Make it conversational and friendly.
```

Prompt 3: Objection Handling

```
The customer said: "{{objection}}"

Respond professionally:
1. Acknowledge their concern
2. Provide factual information
3. Offer alternatives if applicable
```

```
Keep it under 100 words.
```

This guide enables EVERYONE to contribute to building the agent!