Daily Prompt Generator Blogger AI Agent

🗒️ SECTION 1: BASIC DETAILS

Name: Abhishek Rai

AI Agent Title / Use Case: AI Agent to suggest daily writing prompts for bloggers

🧠 SECTION 2: PROBLEM FRAMING

1.1. What problem does your AI Agent solve?

Many bloggers struggle with coming up with fresh topics, especially when they're short on time or inspiration. This agent takes care of that by understanding the user's niche, preferred tone, and number of ideas requested — and then instantly provides engaging blog prompts they can start writing on. It simplifies the ideation phase, saves time, and makes the content planning process more consistent and creative.

1.2. Why is this agent useful?

It helps bloggers who are stuck or looking for inspiration by suggesting blog post ideas tailored to their niche and tone. It reduces writer's block by generating prompts that are relevant and fresh — one idea at a time.

1.3. Who is the target user?

Freelance bloggers, content marketers, solo creators, or even students who need help coming up with article ideas in their specific domain.

1.4. What *not* to include?

I avoided building a full writing assistant that drafts blog posts. That felt out of scope for now. I also didn't add memory across sessions or voice input, to keep the MVP lightweight and focused on idea generation.

🧱 SECTION 3: 4-LAYER PROMPT DESIGN

Create a subsection for each of the 4 components of the agent architecture:

🔹 3.1 INPUT UNDERSTANDING

Prompt: What topic do you want blog ideas for? Please specify tone and number of suggestions.

What is this prompt responsible for?: It extracts the core intent — the blog topic, desired tone (like casual or professional), and how many prompts the user wants.

Example Input + Output: "Can I get 5 casual travel blog ideas?"

🔹 3.2 STATE TRACKER

Prompt: (Not a literal prompt — used st.session_state in Streamlit)

How does this help the agent "remember"?: It stores past messages from both the user and assistant so the chat doesn't reset with each rerun. I used st.session_state.messages to simulate memory in a

stateless Streamlit app.
Did you simulate memory with variables / system messages? If yes, how?: Yes — variables store past conversation data to persist context.

🔹 3.3 TASK PLANNER

Prompt: Generate {count} blog prompt ideas in a {tone} tone for the topic: {topic}.
What steps does your agent take internally to solve the problem?:

- Validate the user input (topic, tone, number).
- Understand the intent and extract keywords.
- Generate prompts using OpenAI's GPT model.
- Return a numbered list in a user-friendly tone.

Did you use chaining? Branching? How did you manage complexity?: Basic chaining + branching. For example, if topic isn't found, I fallback to session history or a default topic like "lifestyle".

3.4 OUTPUT GENERATOR

Prompt: You are a helpful assistant giving blog prompt ideas in a {tone} tone for the topic {topic}. Give {count} ideas.

What kind of output formatting or phrasing did you aim for?:

- Outputs are structured as numbered ideas (1 to n).
- Customizes tone as requested (e.g. friendly, formal).
- If the topic wasn't detected from input, it displays a fallback message but still delivers usable prompts.

Any special behavior? (e.g., examples, markdown formatting, tone control, etc.): markdown formatting used

🔍 SECTION 4: CHATGPT EXPLORATION LOG

You may structure this as a table or bullet list.

Attempt #	Prompt Variant	What Happened	What You Changed	Why You Changed It

1		Used stream=True directly in session	Stored object instead of text	Converted stream to plain string before storing	To avoid storing broken responses
2		chat_input reran full script	Lost chat memory	Used st.session_state to track messages	To simulate memory and fix state
3		Git error leaked secrets	.gitignore missed full path	Added .streamlit/secrets.toml to .gitignore	To protect API keys and keep repo clean

🧪 SECTION 5: OUTPUT TESTS (Optional but Recommended)

- **Test 1:** Normal input
  Input: “Give me 3 professional finance blog ideas.”
  Output:

  Here are 3 finance blog post ideas written in a professional tone:

  - "The Future of Money: How Digital Currencies are Reshaping Global Finance"
  - "Building Wealth: A Step-by-Step Guide to Creating a Diversified Investment Portfolio"
  - "Financial Literacy in the Digital Age: Essential Skills for Navigating Online Banking and Investments"

- **Test 2:** Vague input
  Input: "Give me something to think about"
  Output: Hi fellow-blogger. I can generate between 1 to 10 blog ideas for you at a time. Please enter which topic do you want me to suggest ideas for.

- **Test 3:** Invalid input or challenge
  Input: ""
  Output: Hi fellow-blogger. Please enter a valid input upto 200 words so that i can help you with your blog ideas.

🔄 SECTION 6: REFLECTION

Respond to each question briefly (3–5 sentences each):

- **6.1. What was the hardest part of this assignment?:** Figuring out Streamlit's auto-rerun behavior and how to maintain conversation state across reruns. I had used agents before but not with real-time UI like this.
- **6.2. What part did you enjoy the most?:** When everything clicked together — real-time input, prompt parsing, and OpenAI's response — and it felt like a real assistant. That "aha" moment gave me confidence that I could build useful AI tools.
- **6.3. If given more time, what would you improve or add?:** I'd add session-based memory so the assistant remembers previous conversations
- **6.4. What did you learn about ChatGPT or prompt design?:** The structure of the prompt really matters. Even changing one word affects how creative or relevant the ideas are. I also learned that validation and clarity at each layer saves a lot of debugging later.
- **6.5. Did you ever feel stuck? How did you handle it?:** Yes — especially during the state management and Git mistake. I paused, used ChatGPT to research, and made notes so I wouldn't repeat those mistakes. Eventually, I fixed the issue and made the system more robust.

🧠 SECTION 7: HACK VALUE (Optional)

Did you go beyond the brief in any way?

- SI used a layered prompt design that breaks down input into topic, tone, and count.
- Simulated short-term memory using st.session_state.
- Reused regex-based logic for intelligent parsing of vague user inputs.
- Cleaned up Git with .gitignore corrections and secret key protection SOP.