|  |  |  |
| --- | --- | --- |
| **Author** | **Reviewer** | **Approver** |
| **DongNT17** |  |  |

**Exercise: Exploring ChatGPT**

**1. Objectives**

* Understand the core functionalities and capabilities of ChatGPT.
* Identify limitations and common constraints of the model.
* Develop and test a variety of prompts to evaluate different use cases.
* Document observations, outcomes, and reflections in a structured report.

**2. Problem Statement**

While ChatGPT is powerful, its abilities and limits are not always clear to users. In real practice, vague prompts, complex tasks, or reliance on up-to-date knowledge can lead to surprising errors or failures. This assignment challenges you to actively explore what ChatGPT does well—and where it falls short—through hands-on experiments, so you can learn how to use the tool effectively and recognize when its answers need to be verified or improved.

**3. Inputs / Shared Artifacts**

* There are no predefined inputs or documents provided.
* You are expected to:
  + Access ChatGPT (via chat.openai.com or your organization's platform).
  + Self-direct your exploration using your own questions and ideas.
  + Reference the public OpenAI documentation as needed.

**4. Expected Outcomes**

Submit a short **Exploration Report** (PDF or Google Doc) including the following:

* **Summary** of what ChatGPT can do effectively (e.g., creative writing, rephrasing, coding help, summarization).
* **Examples of limitations or failure cases** (e.g., hallucinations, outdated knowledge, struggles with math).
* **A collection of prompts and responses** that demonstrate both capabilities and shortcomings.
* **Reflection section** with your thoughts on:
  + What surprised you?
  + What types of tasks would you or wouldn’t you trust ChatGPT to handle?
  + How would you improve prompt quality after trial and error?

**5. Example**

**1. What ChatGPT Can Do Effectively**

**Code Generation and Debugging**  
 ChatGPT excels at writing clean, functional code across multiple programming languages and explaining complex algorithms step-by-step.

* **Example Prompt:**  
   "Write a Python function to find the longest palindromic substring in a given string, including comments explaining the logic."
* **Outcome:**  
   Generated efficient code with clear explanations of the dynamic programming approach, proper edge case handling, and comprehensive comments.

**Technical Documentation and Explanation**  
 The model demonstrates strong ability to break down complex technical concepts into digestible explanations.

* **Example Prompt:**  
   "Explain how SSL/TLS handshake works, targeting someone with basic networking knowledge."
* **Outcome:**  
   Provided a clear, step-by-step breakdown with appropriate technical depth and helpful analogies.

**Creative Problem-Solving and Brainstorming**  
 ChatGPT generates diverse approaches to engineering challenges and helps explore solution spaces.

* **Example Prompt:**  
   "I need to design a system to handle 1 million concurrent users for a real-time chat application. What are different architectural approaches?"
* **Outcome:**  
   Suggested multiple valid architectures (microservices, event-driven, WebSocket scaling) with trade-offs clearly explained.

**2. Examples of Limitations and Failure Cases**

**Mathematical Computation Errors**

* **Failed Prompt:**  
   "Calculate the compound interest on $50,000 invested at 7.25% annually for 23 years, compounded quarterly."
* **Issue:**  
   Provided incorrect final calculation despite showing correct formula. Manual verification revealed a ~15% error in the result.

**Outdated Technical Information**

* **Failed Prompt:**  
   "What are the latest features in Python 3.12 released in 2023?"
* **Issue:**  
   Could not provide accurate information about recent language updates, suggesting knowledge cutoff limitations.

**Context Loss in Complex Multi-Step Problems**

* **Failed Prompt:**  
   Attempted to build a complex distributed system design through a lengthy conversation.
* **Issue:**  
   Lost track of earlier architectural decisions, leading to contradictory recommendations later in the conversation.

**Hallucinated Technical Details**

* **Failed Prompt:**  
   "Explain the specific implementation details of Google's Spanner database consistency model."
* **Issue:**  
   Provided confident sounding but partially inaccurate technical details about internal Google systems.

**3. Engineering Recommendations**

**Integration Strategy**

* Use as a "pair programming" partner for initial code generation and debugging.
* Leverage for documentation and knowledge transfer tasks.
* Employ for exploring multiple solution approaches before detailed design.

**Risk Mitigation**

* Always verify mathematical calculations independently.
* Cross-reference technical facts with authoritative sources.
* Test all generated code thoroughly before production use.
* Maintain awareness of knowledge cutoff dates for rapidly evolving technologies.

**Workflow Enhancement**

* Create prompt templates for common engineering tasks.
* Use iterative prompt refinement for complex problems.
* Combine ChatGPT outputs with traditional engineering tools and practices.

**6. Final Submission Checklist**

* **Exploration Report (PDF or Google Doc) includes:**
  + **Summary of ChatGPT capabilities** (what it does well, with examples)
  + **Examples of limitations and failure cases** (with failed prompts and issues)
  + **A collection of prompts and responses** showing both strengths and weaknesses
  + **Reflection section** (what surprised you, trust boundaries, how you improved prompts)
  + Features discovered that are not in the course (reasoning, searching)
* **Demonstration of experimentation:**
  + Evidence of prompt iterations and testing different use cases
* **Clear structure:**
  + Well-organized headings and sections for readability
* **(Optional) Screenshots or tables** summarizing findings
* **Timely submission** as per your course or instructor guidelines