JAIMURUGHAN.B
212222113001

Scenario-Based Report Development Utilizing Diverse Prompting Techniques

Aim:
To create a comprehensive report for the design of a specific application, such as **AI-powered chatbot/solar panel system/automation in manufacturing**, using diverse prompt patterns. This report will employ scenario-based prompting techniques to guide each stage of the design process, ensuring the solution meets the functional and user experience requirements for the chosen application.

Procedure:

1. **Define the Scenario and Use Case:**
   Outline the purpose of the design, the target audience or user base, and its main objectives. Specify the goals the design aims to fulfill, such as **user engagement/energy efficiency/task automation**.

2. **Identify Prompt Patterns for Each Design Aspect:**
   Select appropriate prompt patterns to guide different aspects of the design. Examples of prompt patterns and their applications in the report include:

   - **Idea Generation Prompts:** Brainstorm innovative features or functions the design should incorporate to meet specific goals.
   - **Persona and Context Prompts:** Define the tone, style, or experience the design should convey (e.g., **user-friendly/sustainable/reliable**), aligning with the intended audience.
   - **Exploratory Prompts:** Investigate resources or information essential for the design, such as **user needs/environmental constraints/technical requirements**.
   - **Refinement Prompts:** Refine design elements by adjusting specifications, materials, or style to meet project standards.
   - **Scenario Testing Prompts:** Simulate realistic scenarios or use cases to test the design's effectiveness and adaptability in **user interaction/environmental settings/production workflows**.
   - **Error Handling Prompts:** Design prompts to handle potential issues or challenges effectively within the **user interface/system functionality/automation processes**.

3. **Implementation Plan:**
   Describe the steps to build and implement the design, from **system configuration/component selection/automation setup** to **testing and deployment/installation/integration**.

4. **Evaluation and Feedback Collection:**
   Use targeted feedback prompts to gather insights from **users/stakeholders/operators**, refining the design based on their input for improved functionality and alignment with objectives.

5. **Documentation of Findings:**
   Summarize insights from each prompting technique, noting how they enhanced the design. Include any best practices, limitations, or future improvements

   Detailed Report:

JAIMURUGHAN.B
212222113001

The prompt **"Can you check if there is any debris or dust in the charging port?"** is a diagnostic step aimed at identifying a common issue in mobile devices, such as a smartphone or tablet, not charging properly due to physical obstructions in the charging port.

Context & Explanation:

When a user reports that their device is not charging or charging intermittently, one of the first things to check is the charging port. Over time, dirt, lint, or other debris can accumulate in the port and block the charging connection. This prompt directs the user to inspect the port for these obstructions.

Step-by-Step Breakdown:

1. **User Problem:**
   The user is experiencing issues with their device not charging properly or charging very slowly.
2. **Prompt Purpose:**
   The prompt helps identify a simple, common cause of the issue — dust or debris in the charging port — that can often be resolved without the need for professional repair.
3. **User Action:**
   The user is asked to visually inspect the charging port. If debris or dust is found, they can clean the port using tools like a soft brush, compressed air, or a small cleaning tool.
4. **Follow-Up Actions:**
   - **If debris is found:** The user can clean the port, and then try charging the device again.
   - **If no debris is found:** The system will proceed to suggest other potential causes, such as issues with the charging cable, power source, or the device's battery.

Alternative Examples:

- **"Please check the charging port for any lint or dirt that might be blocking the connection."**
- **"Is there any dust or foreign object in the charging port that could prevent the charger from connecting?"**
- **"Have you inspected the charging port for debris that could be obstructing the charger?"**

Why It's Effective:

- **Simple and Direct:** The prompt is easy for users to follow, even if they are not technically inclined.
- **Common Issue:** Charging issues due to debris in the port are very common and simple to resolve, so addressing this early can save time and frustration.
- **Actionable:** The user can immediately take action to fix the problem without needing professional assistance.

Prototype/System Outline:

The system will be an AI-powered troubleshooting assistant designed to help users identify and resolve common issues with their mobile devices, such as smartphones or tablets. This

system will operate through a chatbot interface, guiding users step-by-step through diagnostic processes and suggesting solutions based on the issue reported.

---

1. Purpose and Goals

- **Main Purpose:** To provide a user-friendly interface that helps users troubleshoot and fix issues related to their devices, such as charging problems, slow performance, connectivity issues, or software glitches.
- **Primary Goals:**
    - Streamline the troubleshooting process for users with minimal technical expertise.
    - Reduce the need for professional technical support for common problems.
    - Offer actionable and easy-to-understand steps for resolving issues.

---

2. System Components

1. **User Interface (UI):**
    - A simple chatbot interface accessible via a mobile app or web application.
    - A welcome screen with a brief introduction and instructions on how to use the system.
    - Clear, concise buttons for selecting issues (e.g., "Charging Problem," "Slow Performance," "Connectivity Issues").
    - A text box for users to describe issues or answer diagnostic questions.
2. **Chatbot AI Engine:**
    - **Natural Language Processing (NLP):** To understand user input, whether it's in the form of predefined selections or free-text responses.
    - **Decision Tree Algorithm:** Based on user input, the system will ask diagnostic questions and guide the user through potential causes.
    - **Knowledge Base:** The system will have a predefined knowledge base with common issues and solutions (e.g., charging issues, screen problems, software bugs).
3. **Diagnostic Workflow:**
    - **Initial Input Collection:** The user will be prompted to describe their issue or select it from a list.
    - **Guided Troubleshooting:** Based on the issue selected, the system will ask a series of yes/no or multiple-choice questions to narrow down the problem (e.g., "Can you check if there is any debris in the charging port?").
    - **Suggested Fixes:** After diagnosing the issue, the system will recommend steps the user can take (e.g., cleaning the charging port, restarting the device, checking for software updates).
4. **Data Logging:**

- Track user inputs and common problems for future improvements.
- Log successful fixes and feedback for system optimization.

5. **Feedback and Support:**
   - After completing the troubleshooting steps, users can provide feedback on whether the solution worked.
   - If the problem persists, the system will escalate the issue by providing contact information for technical support.

---

3. Key Features

- **Step-by-Step Troubleshooting:** A clear and structured flow for guiding users through diagnostic steps.
- **Multi-Channel Support:** Accessible through both mobile apps and web browsers.
- **Real-Time Data Analysis (Optional):** If the system can access device data (e.g., battery stats, app usage), it can provide more precise troubleshooting (e.g., battery health checks).
- **User Feedback:** After each diagnostic session, users are prompted to give feedback on whether the issue was resolved.
- **Maintenance Reminders:** Periodic reminders for users to check for issues like system updates, battery health, and cleaning (e.g., charging port, screen).

---

4. Example Troubleshooting Flows

Example 1: Charging Issue Troubleshooting

1. **User Input:** "My device isn’t charging."
2. **System Prompt:** "Is the charger properly plugged into both the device and the power source?"
3. **User Input:** "Yes."
4. **System Prompt:** "Can you check if there is any debris or dust in the charging port?"
5. **User Input:** "I found some lint."
6. **System Suggestion:** "Please clean the charging port with a soft brush or compressed air, and try charging again."

Example 2: Slow Performance

1. **User Input:** "My device is slow."
2. **System Prompt:** "Have you recently installed any new apps or updates?"
3. **User Input:** "Yes, I installed an app yesterday."
4. **System Prompt:** "Try restarting your device to see if the app is causing the slowdown. If that doesn't work, check if the app has pending updates."

5. **User Suggestion:** "If the issue persists, you may want to uninstall or update the app."

---

5. Key Objectives in System Development

- **User-Centric Design:** Focus on simplicity and user-friendliness, with clear instructions and minimal jargon.
- **Modularity:** The system should be flexible to add new diagnostic flows or integrate with other support systems.
- **Data-Driven Insights:** Gather user feedback to improve the system's diagnostic accuracy and knowledge base.
- **Scalability:** Ensure the system can handle a wide variety of device models, issues, and languages.

---

6. Future Enhancements

- **Voice Integration:** Allow users to interact with the system using voice commands.
- **Advanced Device Monitoring:** Incorporate more advanced device diagnostics like hardware tests or system performance metrics.
- **AI-Powered Suggestions:** Use AI to continuously improve troubleshooting accuracy by learning from user interactions and feedback.

---

7. Conclusion

This AI-powered troubleshooting system will provide users with an efficient way to identify and fix common problems with their devices without needing technical expertise. The goal is to empower users, reduce downtime, and minimize the need for external tech support, all while maintaining a user-friendly interface and helpful guidance throughout the process.

Prompt Effectiveness Summary:

 Effective Prompt Patterns:

- **Diagnostic Questions (Yes/No):** Quick and simple (e.g., "Is your device charging slowly?").
- **Actionable Follow-Ups:** Direct user actions, such as cleaning a port or restarting a device, proved highly effective.
- **Contextual Help:** Prompts asking about recent changes (e.g., "Have you installed any new apps?") helped narrow down issues.

 Impact on User Engagement:

- Simple prompts kept users engaged and led to quicker resolutions.
- Users preferred clear, physical actions (e.g., cleaning or restarting) over complex technical advice.

 Key Findings:

- **Strengths:** Clear, simple prompts resolved many issues quickly and kept users satisfied.

- **Challenges:** Complex issues required more detailed explanations; technical jargon sometimes confused users.

  Recommendations:

- Simplify technical terms and provide additional explanations for complex issues.
- Allow users to choose a level of detail (Beginner vs. Advanced).
- Introduce a feedback loop to continuously improve prompts.

User Testing Results and Improvement Plan:

A. Positive Outcomes:

- **Quick Issue Resolution:**
  Many users were able to resolve common issues (e.g., charging problems, slow performance) with minimal effort, thanks to simple prompts that directed them to check for dust or debris in the charging port, restart the device, or inspect recent app installations.

- **High Engagement with Simple Prompts:**
  Prompts that asked users to check physical components (e.g., charging port) or perform simple actions (e.g., restart the device) were highly effective. These steps were easy for users to follow, leading to fast resolution and high satisfaction.

- **Clear, Actionable Steps:**
  Prompts like “Can you check for any debris in the charging port?” were well-received, as they provided direct and actionable steps users could complete without technical assistance.

B. Areas for Improvement:

- **Complex Issue Handling:**
  Some users struggled with more complex issues, such as software bugs or performance drops, due to lack of detailed guidance in troubleshooting steps. These users often felt unsure of how to proceed after initial steps (e.g., "Have you tried restarting your device?").

- **Technical Language Confusion:**
  A few prompts contained technical terms (e.g., “Check for voltage irregularities”) that confused users, especially those with limited technical knowledge. Some users requested further clarification or simpler wording.

- **Feedback Loop for Success:**
  After resolving the issue, users sometimes felt uncertain about whether their solution was the correct one. A follow-up prompt asking for feedback on the process and if the issue was truly fixed would improve user confidence.

---

3. Improvement Plan

A. Simplify Technical Jargon:

- **Action:** Replace complex technical terms with simpler language and offer optional "explanations" for technical terms (e.g., “battery health” or “system errors”).
- **Expected Outcome:** Reduce confusion and make troubleshooting more accessible for users with limited technical knowledge.

B. Detailed Instructions for Complex Issues:

- **Action:** Develop more detailed troubleshooting paths for issues that require deeper investigation (e.g., slow performance due to app behavior, battery drain, or system conflicts).
- **Expected Outcome:** Provide users with clearer, step-by-step instructions for more complex issues, and possibly include tutorials or video guides for advanced steps.

C. Add a User Feedback Mechanism:

- **Action:** Include a post-diagnosis feedback prompt asking users if the solution worked or if they need further assistance.
- **Expected Outcome:** Ensure that users feel confident in the resolution, and gather data to improve the accuracy of future troubleshooting steps.

D. Multi-Level Troubleshooting:

- **Action:** Implement an option for users to choose the level of assistance they need (e.g., “Beginner Mode” vs. “Advanced Mode”), depending on their technical comfort level.
- **Expected Outcome:** Allow users to feel more in control of the process and provide a more customized experience based on their knowledge.

E. Improve Escalation Process:

- **Action:** When the system identifies an issue that cannot be resolved by simple troubleshooting, offer an easy escalation to human support, or link to relevant resources (e.g., repair guides, warranty information).
- **Expected Outcome:** Reduce frustration for users who cannot solve complex issues, and provide smoother escalation paths to professional support.

---

4. Conclusion

The user testing revealed that the AI-powered troubleshooting system is effective for resolving common mobile device issues. However, to improve the system’s accuracy and user experience, several areas need attention. These include simplifying technical language, providing more detailed solutions for complex issues, enhancing user feedback mechanisms, and allowing for customizable assistance levels. By

implementing these improvements, the system will be more effective in addressing a broader range of issues and will cater to users with different levels of technical expertise.