Exno.7-Prompt-Engineering

Date:

Register no.

Aim: To Develop a prompt-based application tailored to their personal needs, fostering creativity and practical problem-solving skills while leveraging the capabilities of large language models.

Algorithm: Develop a prompt-based application using ChatGPT - To demonstrate how to create a prompt-based application to organize daily tasks, showing the progression from simple to

more advanced prompt designs and their corresponding outputs.

🧠 **OVERVIEW** You will build a prompt-based productivity assistant using ChatGPT. It evolves from simple prompt interaction to a smarter, contextual assistant that personalizes your task schedule based on your input and preferences.

📚 **TABLE OF CONTENTS** 🎯 Objective

🎯 OBJECTIVE Build an AI app that:

Accepts natural language tasks.

Understands user energy patterns and preferences.

Uses prompt engineering to improve output quality.

Generates a time-blocked, prioritized, and rationale-based schedule.

🪄 PROMPT DESIGN PROGRESSION Level Prompt Complexity Features Basic List to schedule Just organizes task order Intermediate Time + priority Adds time slots and priorities Advanced Context-aware assistant Personalizes based on energy, adds rationales

🧪 EXAMPLE PROMPTS AND RESPONSES ⚪ Basic Prompt text Copy Edit Organize these tasks for today: buy groceries, call John, finish the report, exercise. Output

markdown Copy Edit

1. Finish the report
2. Call John
3. Buy groceries
4. Exercise ⚪ Intermediate Prompt text Copy Edit Organize these tasks: buy groceries, call John, finish the project report, exercise. Prioritize them and assign time slots from 9am to 6pm. Output

makefile Copy Edit 9:00 AM – 11:00 AM: Finish the project report 11:30 AM – 12:00 PM: Call John 2:00 PM – 3:00 PM: Buy groceries 4:00 PM – 5:00 PM: Exercise ● Advanced Prompt text Copy Edit You're my AI assistant. My energy is high in the morning and low after lunch. Organize these tasks: buy groceries, call John, finish the report, exercise. Create a schedule from 9 AM to 6 PM with rationale and priorities. Output

makefile Copy Edit 9:00 – 11:00: Finish the report (High priority)

- Requires deep focus. Done during high energy window. 11:15 – 11:45: Call John (Medium priority)
- Light mental load before lunch. 2:00 – 3:00: Buy groceries (Low priority)
- Simple task for low energy. 4:00 – 5:00: Exercise (Medium priority)
- Refreshes mind and body for evening. 🛠️ ARCHITECTURE & TOOLS Component Technology Prompt Processing OpenAI GPT-4 Backend Logic Python CLI Interface input() + print() (Optional) Web UI Streamlit

💻 **FULL PYTHON CODE** (CLI APP) python Copy Edit import openai

Insert your actual OpenAI API key here

openai.api_key = "your-openai-api-key"

def build_prompt(tasks, style): task_str = ', '.join(tasks) if style == 'basic': return f"Organize the following tasks for today: {task_str}." elif style == 'intermediate': return f"Organize these tasks: {task_str}. Prioritize them and assign a time slot between 9am and 6pm." elif style == 'advanced': return (f"You are my AI productivity assistant. I have high energy in the morning and low energy after lunch. " f"My working hours are 9am to 6pm. Organize the following tasks: {task_str}. " f"Include time slots, priorities, and a short rationale for each task's placement.") else: raise ValueError("Unknown style: choose from 'basic', 'intermediate', or 'advanced'.")

def get_ai_response(prompt): response = openai.ChatCompletion.create(model="gpt-4", messages=[{"role": "user", "content": prompt}], temperature=0.7) return response.choices[0].message.content.strip()

def main(): print("\n🧠 Welcome to the AI Task Organizer") tasks_input = input("Enter your tasks (comma-separated): ") style = input("Choose prompt style - basic / intermediate / advanced: ").strip().lower()

```
tasks = [task.strip() for task in tasks_input.split(',')]
prompt = build_prompt(tasks, style)

print("\n📨 Prompt sent to ChatGPT:\n" + prompt)
output = get_ai_response(prompt)
```

```
print("\n✅ AI-Generated Schedule:\n" + output)
```

if **name** == "**main**": main() 🌐 OPTIONAL WEB APP USING STREAMLIT You can turn this into a web app with streamlit:

bash Copy Edit pip install streamlit task_organizer_app.py python Copy Edit import streamlit as st import openai

openai.api_key = "your-openai-api-key"

def build_prompt(tasks, style): task_str = ', '.join(tasks) if style == 'Basic': return f"Organize the following tasks: {task_str}." elif style == 'Intermediate': return f"Organize and assign time slots (9 AM – 6 PM): {task_str}." elif style == 'Advanced': return (f"You are my AI productivity assistant. Energy is high in the morning. " f"Organize tasks: {task_str}. Include time, priority, and rationale.")

st.title("🧠 AI Task Organizer")

tasks_input = st.text_input("Enter your tasks (comma-separated):", "finish report, call John, buy groceries, exercise") style = st.selectbox("Prompt Complexity", ["Basic", "Intermediate", "Advanced"])

if st.button("Generate Schedule"): tasks = [t.strip() for t in tasks_input.split(',')] prompt = build_prompt(tasks, style)

```
with st.spinner("Thinking..."):
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[{"role": "user", "content": prompt}],
        temperature=0.7
    )
    result = response.choices[0].message.content.strip()

st.subheader("📋 AI-Generated Schedule")
st.text(result)
```

Run it with:

bash Copy Edit streamlit run task_organizer_app.py 🧩 ENHANCEMENTS & FUTURE IDEAS 🔔 Task Notifications via email/SMS.

📅 Calendar Integration (Google Calendar API).

📊 Productivity Reports from historical data.

🧠 Memory-aware Assistant: track recurring tasks over time.

Result: The Prompt is executed successfully