```
import streamlit as st
from groq import Groq
import os
from datetime import datetime

client = Groq(api_key=os.environ.get("GROQ_API_KEY"))

SYSTEM_PROMPTS = {
    "input_refinement": """As a travel planning assistant, gather the following details:
    1. Dietary preferences
    2. Specific interests (e.g., historical sites, nightlife)
    3. Walking tolerance (1-5 scale)
    4. Accommodation preferences (luxury/budget)
    5. Transportation preferences""",

    "activity_suggestion": """Suggest activities considering:
    - User's budget: INR {budget}
    - Duration: {days} days
    - Number of travelers: {travelers}
    - Destination: {destination}
    - Preferences: {preferences}
    Include both popular attractions and lesser-known experiences.""",

    "itinerary_generation": """Create a {days}-day itinerary for {destination} with:
    - Morning, Afternoon, and Evening sections
    - Logical geographical grouping of activities
    - Estimated transportation times
    - Budget allocations for activities and meals
    - Considerations for {travelers} travelers
    - Travel options from {start_city} to {destination}"""
}
```

```
def get_groq_response(prompt, model="llama3-70b-8192"):
    response = client.chat.completions.create(
        messages=[{"role": "user", "content": prompt}],
        model=model,
        temperature=0.7
    )
    return response.choices[0].message.content

st.title("AI-Powered Travel Planner")

col1, col2 = st.columns(2)
with col1:
    start_city = st.text_input("Starting City", placeholder="e.g., Mumbai, India")
    destination = st.text_input("Destination", placeholder="e.g., Barcelona, Spain")
    start_date = st.date_input("Start Date")
with col2:
    budget = st.number_input("Budget (INR)", min_value=10000, step=5000)
    duration = st.text_input("Trip Duration (days)", placeholder="e.g., 7")
    travelers = st.number_input("Number of Travelers", min_value=1, step=1, value=1)

with st.expander("Additional Preferences"):
    preferences = st.text_area("Please specify any interests or requirements:")

if st.button("Generate Travel Plan"):
    if not destination or not duration or not start_city:
        st.warning("Please enter starting city, destination, and trip duration.")
    else:
        try:
            duration = int(duration)
            if duration <= 0:
```

```
            st.error("Trip duration must be a positive number.")

        # Check if the start date is valid
        if start_date < datetime.now().date():
            st.error("Please enter a valid future date.")



        # Determine travel options based on destination
        international_destinations = ["Barcelona", "New York", "Tokyo"]  # Example list
        if destination in international_destinations:
            travel_method = "air"
        else:
            travel_method = "air/road/train"

        refinement_prompt = f"{SYSTEM_PROMPTS['input_refinement']}\nUser  context:
{preferences}"
        refined_prefs = get_groq_response(refinement_prompt)

        activity_prompt = SYSTEM_PROMPTS['activity_suggestion'].format(
            budget=budget,
            days=duration,
            travelers=travelers,
            destination=destination,
            preferences=refined_prefs
        )
        activities = get_groq_response(activity_prompt)

        itinerary_prompt = SYSTEM_PROMPTS['itinerary_generation'].format(
            days=duration,
            destination=destination,
            travelers=travelers,
```

```
            start_city=start_city
        )
        final_prompt = f"{itinerary_prompt}\nSuggested activities: {activities}\nTravel method: {travel_method}"
        itinerary = get_groq_response(final_prompt)

        st.subheader(f"Your Personalized Travel Itinerary for {destination}")
        st.markdown(itinerary)
        st.download_button("Download Itinerary", itinerary, file_name="travel_itinerary.md")
    except ValueError:
        st.error("Please enter a valid number for trip duration.")
```