The Challenge of Using an LLM for Task Scheduling:

When we think of a powerful AI like GPT-4o, it's easy to assume it can handle every task perfectly. However, for a service like setting and sending reminders, a solution that relies only on the LLM presents significant technical problems.

LLM's are based on the architecture of request-response cycle. So they cannot set a reminder by themselves. Most of the AI Conversation Applications like "ChatGPT" and "Gemini" are based on the Internet**.** They are integrated with Google services like **"Google Calendar"** and **"Microsoft Tasks",** so when asked to schedule a task , the remainder is set in those apps. But since the core functionality of the Nudge App is working without the internet , it is not possible for our LLM to set the remainders and Tasks by integrating with Google or Microsoft services.

So there are two proposed solutions for the functionality of 'setting remainders and task scheduling'

1) The Hybrid Approach:

The Hybrid Approach combines the flexibility of an AI model with the reliability of a dedicated scheduling system. This method uses the LLM's natural language understanding (NLU) capabilities only when they are most valuable, and relies on a simpler, more cost-effective system for all other tasks."**Remind**" will be the keyword which the user compulsorily needs to use in the starting of the prompts ,when setting a reminder or scheduling a task.

How It Works

- **User Texts a Reminder**: A user sends an SMS with the keyword, like "Remind me to call Mom tomorrow at 5 PM."
- **Twilio Receives It**: Twilio sends the message to your application's API.
- **Check for Keywords**: Your application detects the "Remind" keyword. This immediately tells the system to use a different, more specific process.
- **Send to the LLM for Extraction**: Instead of asking for a conversational response, your app sends a very specific request to the LLM, asking it to **extract the task, date, and time** from the user's message.
- **LLM Returns Structured Data**: The LLM returns a clean, structured piece of data, like a JSON object: `{"task": "call Mom", "date": "tomorrow", "time": "5 PM"}`.
- **Store in Scheduler**: Your app takes this structured data and saves it in a database table for scheduled tasks.
- **Send Confirmation SMS**: Your app sends a quick confirmation to the user, like "Got it! I'll remind you to call Mom tomorrow at 5 PM." This confirms the reminder was set successfully.

- **Scheduler's Job**: A separate system (a scheduler like a cron job or cloud service) periodically checks the database. When the specified time arrives, it triggers and sends an SMS to the user with the task: "Reminder: Call Mom."

Pros and Cons

Pros

- **Superior User Experience**: Users can type in natural language, just as they would when talking to a friend. They don't have to remember a specific format. This is the most intuitive and user-friendly option.

Cons

- **Initial Development Complexity**: This approach requires slightly more initial development effort to set up the smart routing logic and the data extraction prompts for the LLM.
- **Complexity increased by NLP concept :** To make a json object suitable for our scheduler ,our LLM needs to extract the data from the user prompts , which requires **natural language understanding** (a concept of natural language processing).
- **Increased Cost:** Need a cloud service to use a scheduler which will schedule the tasks.
- **Uncertainty in Project Success:** The Hybrid Approach, while elegant in theory, has a higher degree of uncertainty regarding its final outcome, since LLM might make a Json object not very reliable for scheduler.

Example prompts of Hybrid Approach: “User will always start with the keyword Remind**”**

Prompt 1: "**Remind** me to call Mom tomorrow at 5 PM."

Prompt 2: “**Remind** me on 10th july to take my clothes for dry cleaning at 9 PM"

Prompt 3: "**Remind** every year, on the day before Christmas, to buy presents "

The Fixed-Format Approach: Simple and Direct

The Fixed-Format Approach relies on the user to provide all the necessary information in a predefined structure. This method avoids the need for a complex natural language understanding system by simplifying the user's input.

How It Works

1. **User Sends a Formatted Message**: The user is instructed to send a message in a specific format, such as:

   Remind:
   Task: [task]
   Date : [date]
   Time: [time]

2. **Simple Parsing**: Your application receives the message and uses a simple, rule-based parser to extract the **`Task`**, **`Date`**, and **`Time`** from the message. This parser is a simple piece of code that looks for the keywords and the corresponding data.
3. **Confirmation & Scheduling**: The application sends a confirmation message to the user and saves the data in the database. A scheduler then uses this information to trigger the reminder at the correct time.

Pros and Cons

Pros

- **Lower Development Cost**: This is the simplest and fastest solution to build. It doesn't require any advanced AI models or complex logic, relying on basic keyword and text parsing.
- **Predictable and Reliable**: Because the input is always in a fixed format, the system will always understand it correctly. There is no risk of misinterpreting the user's intent.
- **No LLM Costs**: This method completely avoids the use of an LLM for reminder creation, which eliminates the associated API costs for this specific function.

Cons

- **Poor User Experience**: This approach is rigid and unintuitive. The user must remember the exact format and keywords, which can lead to frustration and a high rate of errors.
- **No Flexibility**: It cannot handle complex or conversational requests like "Remind me to call Mom the day before her birthday" or "Remind me every year on March 3rd to get my car serviced." Any request outside of the exact format will fail.

- **Increased Cost:** Need a cloud service to use a scheduler which will schedule the tasks.

Fixed-Format Approach Examples

The user must adhere to a strict, predefined format.

Prompt 1:
```
Remind:
        Task: call Mom,
        Date: 23/05/2025,
        Time: 5 PM
```

Prompt 2:
```
Remind:
        Task: Take my clothes for dry cleaning,
        Date: 22/06/2025,
        Time: 9 PM
```

Conclusion: The feasibility seems to be uncertain at the point , but might be achieved by trying two solutions.