AI Prompt

- Thiết kế Prompt cho AI Agents để phục vụ nghiệp vụ một cách hiệu quả với FPL (Flow Prompt Language)

Thiết kế Prompt cho AI Agents để phục vụ nghiệp vụ một cách hiệu quả với FPL (Flow Prompt Language)

Flow-Prompt Language (FPL) giải quyết bài toán gì – và giải quyết như thế nào?

Khi AI Agents bắt đầu được đưa vào các hệ thống nghiệp vụ thật – callbot xác minh hợp đồng, trợ lý CSKH, agent bán hàng, agent gọi API – prompt không còn là vài dòng mô tả "hãy làm tốt nhất có thể". Prompt trở thành **thứ quyết định Agent sẽ hành xử ra sao trong từng tình huống cụ thể**.

Và chính ở thời điểm đó, rất nhiều đội ngũ gặp cùng một vấn đề:
prompt càng chi tiết thì Agent càng khó kiểm soát.

Khi prompt viết đúng ý người, nhưng sai cách cho máy

Một prompt nghiệp vụ điển hình thường cố gắng bao trùm mọi thứ: vai trò Agent, mục tiêu, các bước xử lý, ngoại lệ, quy tắc an toàn. Về mặt ngôn ngữ, prompt hoàn toàn hợp lý. Nhưng khi chạy

thực tế, Agent vẫn bỏ bước, rẽ sai nhánh, hoặc gọi tool ở những thời điểm không mong muốn.

Điểm mấu chốt nằm ở đây:
prompt đang được viết như một đoạn mô tả mong muốn, trong khi AI Agent cần một bản thiết kế thực thi.

Một Agent nghiệp vụ không chỉ "trả lời", nó đi qua nhiều trạng thái, phải tuân thủ thứ tự, phải dừng đúng lúc, và phải gọi đúng hành động khi điều kiện thỏa mãn. Nếu những điều này chỉ được viết bằng câu chữ, Agent buộc phải tự suy luận logic – và đó chính là nguồn gốc của sai lệch.

Tư duy nền tảng của Flow-Prompt Language

> **Prompt cho AI Agents không nên là văn bản tự do, mà là đặc tả luồng nghiệp vụ.**

Thay vì hỏi "viết prompt sao cho model hiểu", FPL đặt câu hỏi: luồng nghiệp vụ này gồm những bước nào, bước nào bắt buộc phải đi trước, điều kiện nào thì rẽ nhánh, khi nào thì kết thúc, và ở đâu thì được phép gọi tool.

FPL không cố làm prompt "thông minh" hơn, mà làm nó **rõ ràng hơn**.

FPL là gì, nói một cách chính xác?

Về bản chất, **FPL là một DSL (domain-specific language) cho prompt**, được biểu diễn hoàn toàn bằng Markdown. Nó cho phép mô tả luồng nghiệp vụ, nhánh xử lý, trigger flow, các khối logic tái sử dụng, thứ tự thực thi, điều kiện, vòng lặp và hành động cụ thể.

Điều quan trọng là: **LLM đọc được cấu trúc này và tuân theo nó tốt hơn rất nhiều so với văn bản tự do**.

Cấu trúc cốt lõi của FPL: Agent nhìn thế nào?

FPL sử dụng chính cấu trúc heading của Markdown để biểu diễn quan hệ logic. Đây không phải là

format trình bày cho đẹp, mà là **cách để Agent hiểu phạm vi và thứ bậc của logic**.

Một Flow là một tuyến nghiệp vụ hoàn chỉnh. Khi Flow kết thúc, Agent phải dừng hoặc chuyển sang Flow khác theo chỉ dẫn. Sub-Flow là nhánh xử lý con, chỉ được đi vào khi điều kiện thỏa mãn. Trigger Flow thì khác: nó có thể được kích hoạt ở bất kỳ thời điểm nào trong hội thoại, miễn là trigger xuất hiện.

Một cấu trúc FPL tối giản có thể trông như sau:

```
## MAIN FLOW: APPOINTMENT BOOKING FLOW
  1. Greeting User
    - Reply "Hello, how can I assist you with booking an appointment?"

### SUB_FLOW: RESCHEDULE_SUB_FLOW
  1. Offer alternative slots

## TRIGGER FLOW: BUSY_TRIGGER_FLOW
*Trigger*: user says any phrase matching "busy|call back"
  1. Ask for callback time
```

Ngay ở mức heading, Agent đã phân biệt được đâu là tuyến chính, đâu là nhánh phụ, và đâu là luồng có thể chen ngang.

Conversation Routine: khối logic mà prompt truyền thống luôn thiếu

Một điểm rất quan trọng trong FPL là khái niệm **Conversation Routine**. Routine là một khối logic có tên, bao gồm nhiều Step liên tiếp để hoàn thành một nhiệm vụ nhỏ nhưng nhất quán, ví dụ xác minh danh tính hoặc xác nhận booking.

Trong prompt truyền thống, những logic này thường bị mô tả lặp lại hoặc viết mơ hồ. FPL buộc bạn phải đóng gói chúng thành một module rõ ràng:

```
#### AUTHENTICATION ROUTINE
1. Ask for policy number
  - Reply "May I have your policy number?"
```

```
2. Verify policy number
  - call `verify_policy(policy_number=user input)`
  - If **verification_failed**, then call `end_call_outcome("POLICY_NOT_FOUND")` and **End**
```

Routine có tên, có thứ tự, có điểm dừng. Khi được gọi, Agent thực thi toàn bộ khối này như một đơn vị logic hoàn chỉnh, thay vì phải "hiểu ngầm" ý đồ của người viết prompt.

Step và Action: tách "quyết định" khỏi "thực thi"

Ở mức thấp nhất, FPL làm rõ hai khái niệm mà prompt tự do thường trộn lẫn: **Step** và **Action**.

Một Step là một đơn vị logic, nơi Agent đưa ra quyết định. Một Action thì chỉ làm đúng một việc: gọi tool, trả lời người dùng, ghi trạng thái, hoặc kết thúc Flow. Action không chứa logic, nó chỉ được kích hoạt khi Step cho phép.

```
2. Availability Confirmation
  - call `get_available_slots(date=preferred_date)`
  - If **slot_unavailable**, then reply "Sorry, that slot is taken" and **Proceed**
RESCHEDULE_SUB_FLOW
  - Otherwise, call `create_appointment(date=preferred_date)` and **End**
```

Nhờ tách bạch này, Agent không còn nhầm lẫn giữa việc "suy nghĩ nên làm gì" và "thực hiện hành động".

Điều kiện và rẽ nhánh trong FPL hoạt động ra sao?

FPL yêu cầu điều kiện được viết theo dạng **If – Then – Otherwise**, với thứ tự ưu tiên rõ ràng. Agent đọc từ trên xuống và thực thi nhánh đúng đầu tiên, sau đó dừng việc đánh giá các nhánh còn lại.

```
```

```
- If **wrong_number**, then call `end_call_outcome("WRONG_NUMBER")` and **End**
- Otherwise, proceed IDENTITY_VERIFICATION_ROUTINE
```

Cách làm này khiến Agent hành xử giống một hệ thống nghiệp vụ thực thụ, thay vì cố gắng cân nhắc mọi khả năng cùng lúc.

Loop trong FPL: retry có kiểm soát, không lặp vô hạn

FPL cho phép dùng vòng lặp, nhưng mỗi loop đều phải được thiết kế có chủ đích: lặp cái gì, điều kiện thoát là gì, và giới hạn bao nhiêu lần thử.

```
**DO: **
  - Ask the user for district and city
  - Validate input
**LOOP UNTIL: **
  - Input is valid
  - Invalid input **3 times**, then call `end_call_outcome("INPUT_INVALID")` and **End**
```

Retry được cho phép, nhưng retry không kiểm soát thì không.

Tool call: từ “model tự đoán” sang “thiết kế chủ động”

Trong FPL, tool call chỉ xuất hiện tại những Step mà nghiệp vụ cho phép. Tool không còn là “ý định ngôn ngữ”, mà là hệ quả tất yếu của một điều kiện đã được thiết kế trước.

```
- call `retrieve_booking_info(booking_code=user input)`
- If **booking_not_found**, then **Proceed** ERROR_HANDLING_FLOW
```

Cách tiếp cận này giúp giảm hallucination, tăng độ chính xác của function call, và rất dễ gắn logging, metric hoặc audit.

FPL không thay thế model – nó thay thế sự mơ hồ

Một hiểu lầm phổ biến là nghĩ rằng FPL làm Agent thông minh hơn. Thực tế, FPL làm một việc quan trọng hơn nhiều: **loại bỏ sự mơ hồ khỏi prompt**.

Model vẫn chịu trách nhiệm hiểu ngôn ngữ tự nhiên. Nhưng cách Agent hành động, rẽ nhánh, gọi tool và kết thúc – tất cả đều nằm trong phạm vi đã được thiết kế.

Ví dụ tổng hợp: một Flow hoàn chỉnh trong FPL

Để thấy rõ cách các khái niệm Flow, Sub-Flow, Trigger Flow, Step, Action và Loop phối hợp với nhau, hãy xem ví dụ sau về một Agent đặt lịch hẹn.

Agent này có thể:

- Thu thập ngày mong muốn
- Retry có kiểm soát nếu người dùng nhập sai
- Rẽ nhánh sang Sub-Flow khi lịch không khả dụng
- Bị ngắt ngang bởi Trigger Flow khi người dùng bận

```
## MAIN FLOW: APPOINTMENT BOOKING FLOW
  1. Greeting User
    - Reply "Hello, how can I assist you with booking an appointment?"

  2. Preferred Date Collection
    **DO: **
      - Ask user for preferred date
      - Validate date format and availability
```

```
    **LOOP UNTIL:**
      - Date is valid
      - Invalid input **3 times** → end_call_outcome("DATE NOT PROVIDED") and **End**

  3. Availability Confirmation
    - call get_available_slots(date=preferred_date)
    - If **slot_unavailable**, then reply "Sorry, that slot is taken" and **Proceed**
RESCHEDULE_SUB_FLOW
    - Otherwise, call create_appointment(date=preferred_date)
    - Reply "Your appointment on preferred date is confirmed. Have a great day!"

### SUB_FLOW: RESCHEDULE_SUB_FLOW
  1. Offering Alternative Slots
    - Offer alternative slots
    - If user declines all, then call end_call_outcome("USER_DECLINED") and **End**
    - Otherwise, call create_appointment(date=selected_slot) and **End**

## TRIGGER FLOW: BUSY_TRIGGER_FLOW
*Trigger*: user says any phrase matching "busy|call back"
  1. Ask for Callback Time
    - If user provides time, call schedule_callback(time=provided_time) and **End**
    - Otherwise, call end_call_outcome("CALLBACK_DECLINED") and **End**
```

Điểm quan trọng không nằm ở nghiệp vụ đặt lịch, mà ở cách Flow được thiết kế: Agent **không phải suy đoán** nên làm gì tiếp theo – mọi hành vi đều đã được định nghĩa.

Quy trình authoring FPL trong thực tế

Một workflow phổ biến khi viết FPL cho hệ thống nghiệp vụ:

1. Phác thảo quy trình nghiệp vụ → mỗi quy trình chính là một `Flow` hoặc `Trigger Flow`.
2. Xác định trigger ngôn ngữ tự nhiên cho từng Trigger Flow.
3. Tách các nhánh phức tạp thành `Sub-Flow`.
4. Đóng gói logic lặp lại thành `Routine`.
5. Đánh số Step rõ ràng; đặt Action đúng vị trí thực thi.
6. Thêm điều kiện rẽ nhánh và guard cho loop.

7. Review chéo với checklist:
    - Đúng hierarchy & numbering
    - Tên tool và tham số hợp lệ
    - Loop có điều kiện thoát
    - Biến placeholder được định nghĩa rõ

Template tham chiếu cho FPL

```
## <FLOW_NAME>
1. Purpose sentence
  - If <trigger_condition>, then **Proceed** <TRIGGER_FLOW_NAME>

### <SUB_FLOW_NAME>
1. Step Title
  - Prompt: "..."
  - If &lt explained_condition>, then call <function>(...)

---
## <TRIGGER_FLOW_NAME>
*Trigger*: <condition>
  1. Step Title
    - ...
```

Kết luận: từ prompt cảm tính sang prompt có kiến trúc

Khi AI Agents trở thành một phần của hệ thống nghiệp vụ, prompt không còn là một artefact phụ. Nó là một lớp kiến trúc.

Flow-Prompt Language không phải là một “style viết prompt”, mà là một **phương pháp thiết kế**. Nó buộc người viết phải suy nghĩ như một kiến trúc sư nghiệp vụ, thay vì chỉ là người “viết cho model hiểu”.

Câu hỏi quan trọng không còn là *model nào thông minh hơn*, mà là:

> **“ Prompt của bạn đang mô tả mong muốn, hay đang điều khiển hành vi?**

Glossary

Thuật ngữ	Định nghĩa
Agent	Trợ lý dựa trên LLM, thực thi Flow theo đặc tả FPL.
Flow / Trigger Flow	Các module prompt cấp cao, định nghĩa bằng heading `##`.
Sub-Flow	Nhánh xử lý con, chỉ được đi vào khi điều kiện thỏa mãn.
Routine	Khối logic tái sử dụng gồm nhiều Step liên tiếp.
Step	Đơn vị hội thoại nguyên tử, nơi Agent ra quyết định.
Action	Hành động cụ thể: gọi tool, reply, ghi trạng thái, hoặc kết thúc Flow.
Placeholder	Biến được thay thế runtime (ví dụ `{{user_input}}`) bởi prompt loader như Jinja.

Source: https://aiourlife.blogspot.com/2026/01/ke-prompt-cho-ai-agents-theo-luong.html