

Prompt Detail

Prompt Template

- A PromptTemplate in LangChain is a structured way to create prompts dynamically by inserting variables into a predefined template. Instead of hardcoding prompts, PromptTemplate allows you to define placeholders that can be filled in at runtime with different inputs.
- This makes it reusable, flexible, and easy to manage, especially when working with dynamic user inputs or automated workflows.

Why use PromptTemplate over f strings?

1. Default validation.
2. Reusable.
3. LangChain Ecosystem

Messages:

Console

- You: -----
- AI: -----
- You: -----
- AI: -----

Context Less.

- which one is greater 2 or 3
- Multiply bigger number with 5

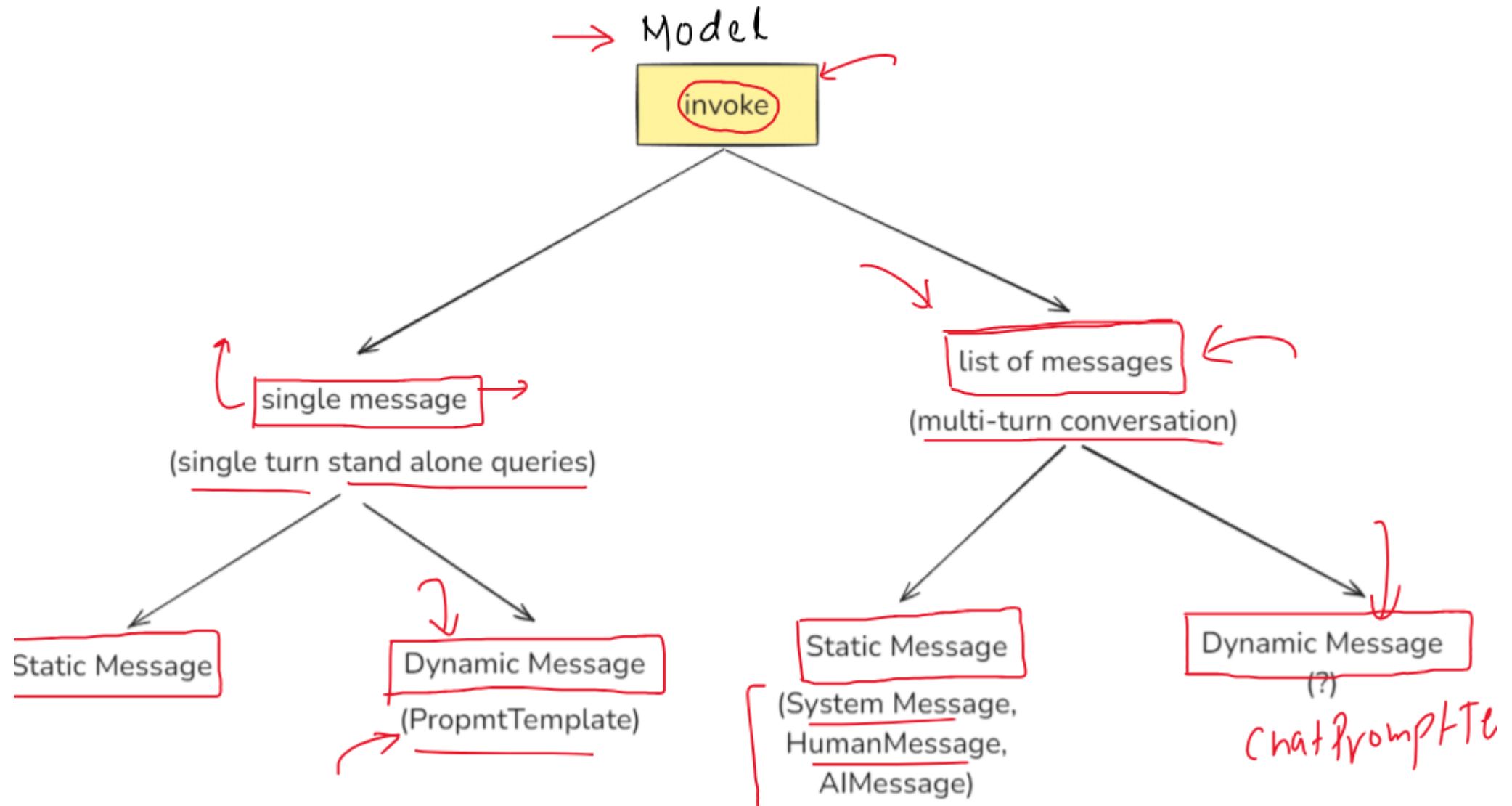
Need to maintain history of pervious messages

- which one is greater 2 or 3
- Multiply bigger number with 5

Types of Messages

- As the history becomes larger, model does not know the message is whom.
- Message should be saved with Identity. E.g.
 - user-----
 - AI:--
- Types of Messages:
 - System Messages
 - Human Messages
 - AI Messages

Chat Prompt Template



dynamic

- system message ->
- [You are a helpful {domain} expert]
- Human:
- explain about: {topic}

Message Placeholder

- A MessagesPlaceholder in LangChain is a special placeholder used inside a ChatPromptTemplate to dynamically insert chat history or a list of messages at runtime.

- Amazon Assistant:

User: Request about refund for a particular order

Assistant: you will receive refund with 2 to 3 days

User: what is the status of my refund? (here required the previous chat)

Store the chat in db

And used next time chat → message place holder

Structured output

- Wish LLM should talk/interact with my databases or API
- Structured output is very relevant to Agents
- Prompt → input to LLM → output is text(unstructured)
- Should be in Structured format.

Structured Output

- In LangChain, structured output refers to the practice of having language models return responses in a well-defined data format (for example, JSON), rather than free-form text. This makes the model output easier to parse and work with programmatically.

[Prompt] - *Can you create a one-day travel itinerary for Paris?*

[LLM's Unstructured Response]

[LLM's Unstructured Response]

Here's a suggested itinerary: Morning: Visit the Eiffel Tower.

Afternoon: Walk through the Louvre Museum.

Evening: Enjoy dinner at a Seine riverside café.

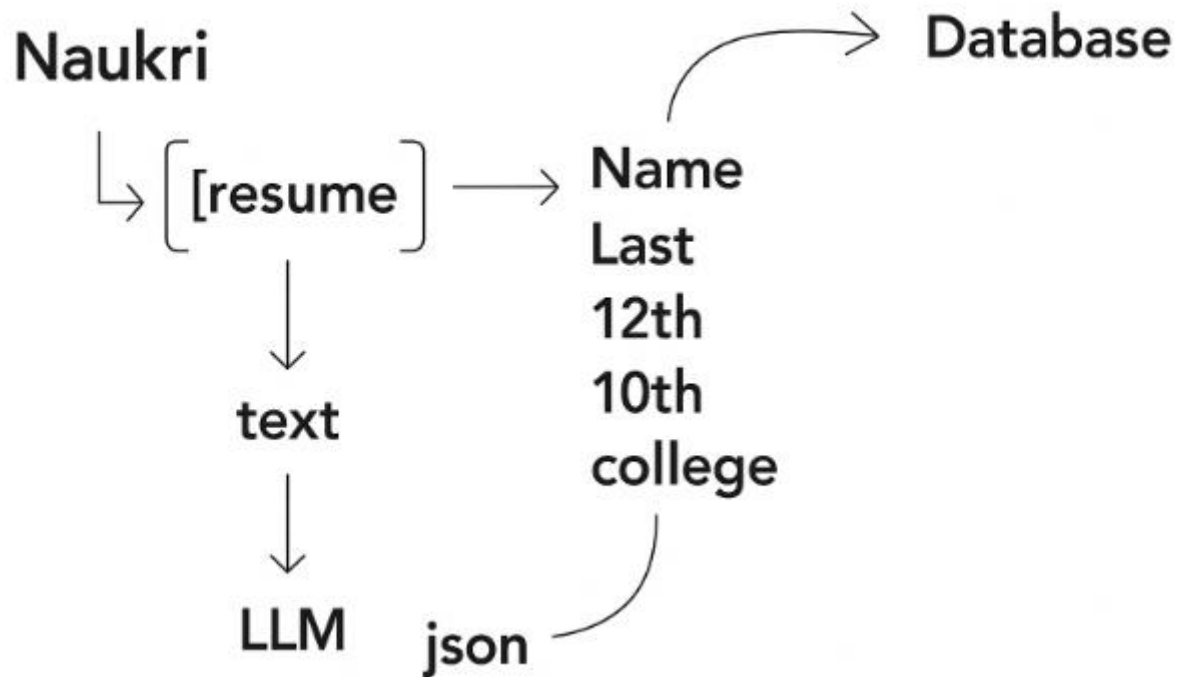


[JSON enforced output]

```
[  
  {"time": "Morning", "activity": "Visit the Eiffel Tower"},  
  {"time": "Afternoon", "activity": "Walk through the Louvre Museum"},  
  {"time": "Evening", "activity": "Enjoy dinner at a Seine riverside café"}  
]
```

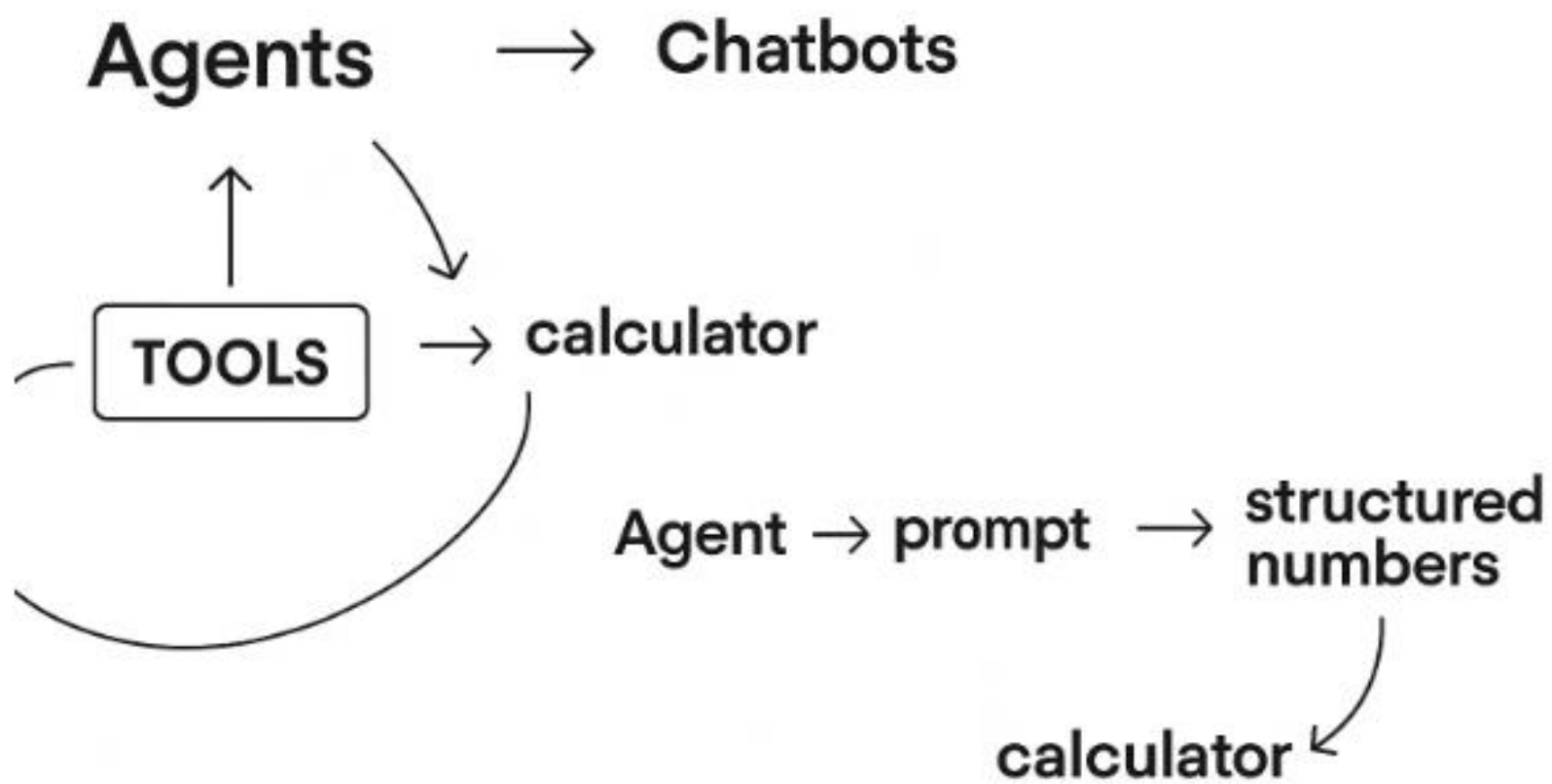
Why do we need Structured Output

- Data Extraction
- API building
- Agents



Amazon Product Reviews:

```
[ ] Topic → [ ]
[ ] pros → [ ]
[ ] cons → [ ]
sentiments = 0.5
```



Ways to get Structured Output

Till Now

LLM \leftrightarrow Human

But not with other system/API

LLM Types:

- Can \rightarrow (e.g gpt models)
 `with_structure_output ()`
- Can't \rightarrow `output parser`

with_structured_output

- you mention only Data is format

You are an AI assistant that extracts structured insights from text.
Given a product review,
extract: - Summary: A brief overview of the main points. - Sentiment: Overall tone of the review (positive, neutral, negative). Return the response in JSON format.

- Ways to tell about data format
- We invoke model with:
 1. TypedDict
 2. Pydantic
 3. Json_schema

TypedDict

- **TypedDict** is a way to define a dictionary in Python where you specify what keys and values should exist. It helps ensure that your dictionary follows a specific structure.
- `Person{ name : Ahmed, age: 32}`

Why use TypedDict?

- It tells Python what keys are required and what types of values they should have.
- It does not validate data at runtime (it just helps with type hints for better coding).

```
Person={  
    Name: str  
    Age: int  
}
```

1,2,3,4

You are an AI assistant that extracts structured insights from text.

Given a product review,

extract: - Summary: A brief overview of the main points. - Sentiment: Overall tone of the review (positive, neutral, negative). Return the response in JSON format.

-> simple TypedDict

-> Annotated TypedDict

-> Literal

-> More complex -> with pros and cons

Data is not Validated . it representation

Pydantic

- Pydantic is a data validation and data parsing library for Python. It ensures that the data you work with is **correct, structured, and type-safe**.
- **Pip install pydantic**

Basic example

Default values

Optional fields

Coerce

Builtin validation

Field Function -> default values, constraints, description, regex expressions

Returns pydantic object -> convert to json/dict

Json

Three ways for structure output parser(Typedic, Pydantic, json)

Json Attributes:

- ✓ title
- ✓ Description
- ✓ Type
- ✓ Properties
- ✓ required

```
{  
  "title": "student",  
  "description": "schema about students",  
  "type": "object",  
  "properties": {  
    "name": "string",  
    "age": "integer"  
  },  
  "required": ["name"]  
}
```

When to use what?

✓ Use `TypedDict` if: ✓

- You only need type hints (basic structure enforcement).
- You don't need validation (e.g., checking numbers are positive).
- You trust the LLM to return correct data.























✓ Use `Pydantic` if:

- You need data validation (e.g., sentiment must be `"positive"`, `"neutral"`, or `"negative"`).
- You need default values if the LLM misses fields.
- You want automatic type conversion (e.g., `"100"` → `100`).

✓ Use `JSON Schema` if:

- You don't want to import extra Python libraries (Pydantic).
- You need validation but don't need Python objects.
- You want to define structure in a standard JSON format.

When to Use What?

Feature	TypedDict 	Pydantic 	JSON Schema 
Basic structure 	 —	 —	 —
Type enforcement —	 —	 —	 —
Data validation			
Default values			 →
Automatic conversion			
Cross-language compatibility			

A few things to remember!

- With_structured_output, you must tell the method
- Method
 - Json Mode(json format) [claude, gimini]
 - Function/tool calling(OpenAI)→ out put is used by another function e.g. Agents

Output Parsers(Can't) → Open source

- Output Parsers in LangChain help convert raw LLM responses(text) into structured formats like JSON, CSV, Pydantic models, and more. They ensure consistency, validation, and ease of use in applications.

1. **String output parser**
2. **Json output parser**
3. **Structure output parser**
4. **Json output parser**

1.StrOutputParser

- The StrOutputParser is the simplest output parser in LangChain. It is used to parse the output of a Language Model (LLM) and return it as a plain string.

```
content='A black hole is a region in space where gravity is so strong that nothing, not even light, can escape its pull. It is formed when a massive star collapses upon itself.'  
additional_kwargs={'refusal': None} response_metadata={'token_usage': {'completion_tokens': 37, 'prompt_tokens': 15, 'total_tokens': 52, 'completion_tokens_details': {'accepted_prediction_tokens': 0, 'audio_tokens': 0, 'reasoning_tokens': 0, 'rejected_prediction_tokens': 0}, 'prompt_tokens_details': {'audio_tokens': 0, 'cached_tokens': 0}}, 'model_name': 'gpt-3.5-turbo-0125', 'system_fingerprint': None, 'finish_reason': 'stop', 'logprobs': None} id='run-a7b90203-58f8-47c5-a01b-01184b6aec14-0' usage_metadata={'input_tokens': 15, 'output_tokens': 37, 'total_tokens': 52, 'input_token_details': {'audio': 0, 'cache_read': 0}, 'output_token_details': {'audio': 0, 'reasoning': 0}}
```

result.content is not required.

e.g.

Topic → LLM → [Detailed Report] → LLM → 5 Lines summary

JSONOutputPaser

LLM output → Json → Schema Enforcement

Fact-1 -----value

Fact-2 -----

Fact-3 -----

Fact-4 -----

StructuredOutputParser(schema enforced)

- **StructuredOutputParser** is an output parser in LangChain that helps extract structured JSON data from LLM responses based on predefined field schemas.
- It works by defining a list of fields (ResponseSchema) that the model should return, ensuring the output follows a structured format.

Data Validation:

PydanticOutputParser

♦ What is `PydanticOutputParser` in LangChain?

`PydanticOutputParser` is a structured output parser in LangChain that uses Pydantic models to enforce schema validation when processing LLM responses.

🚀 Why Use `PydanticOutputParser`?

- ✓ Strict Schema Enforcement → Ensures that LLM responses follow a well-defined structure.
- ✓ Type Safety → Automatically converts LLM outputs into Python objects.
- ✓ Easy Validation → Uses Pydantic's built-in validation to catch incorrect or missing data.
- ✓ Seamless Integration → Works well with other LangChain components.

str → string

JSON → json → X schema

schema → str json → X validation

pydantic → str json + validation

What and Why

PipeLine:

Input → Prompt → LLM → output →

Prompt can be:

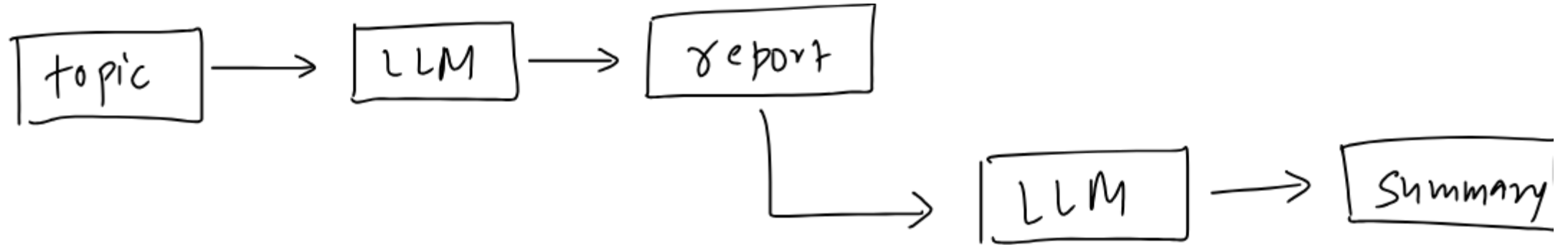
- Sequential Chain

- Parallel Chain

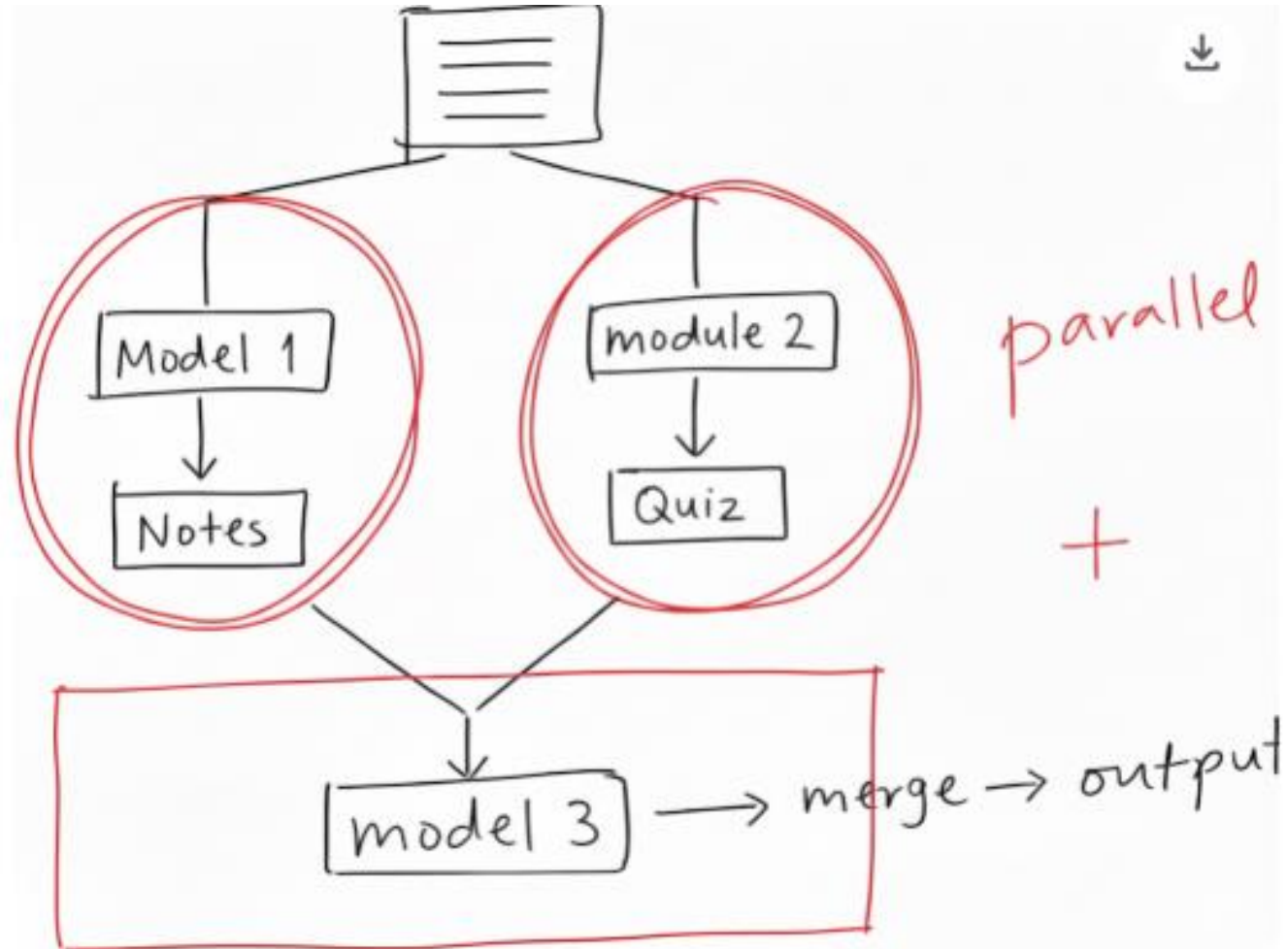
- Conditional Chain

Runnables

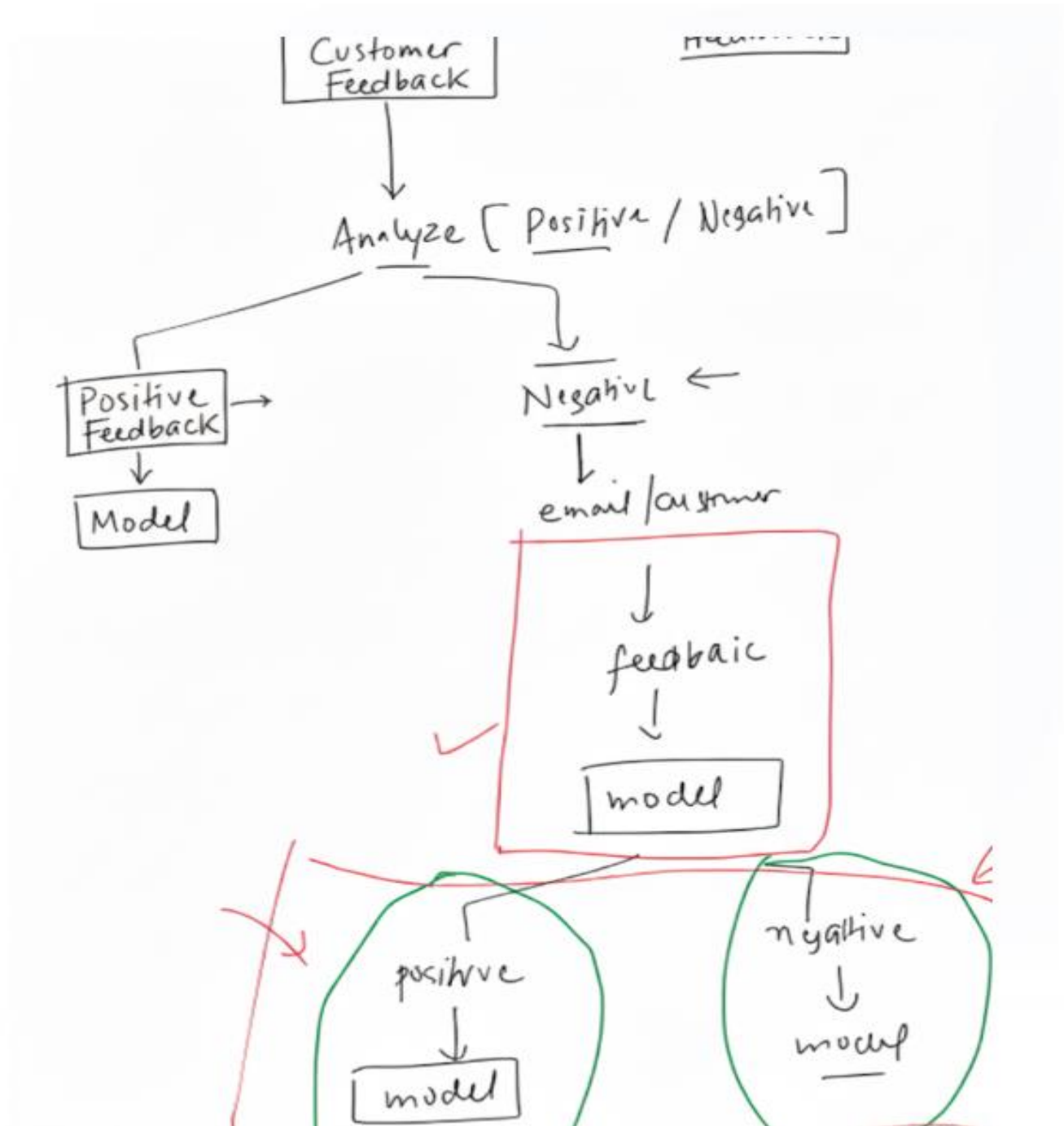
Sequential Chain



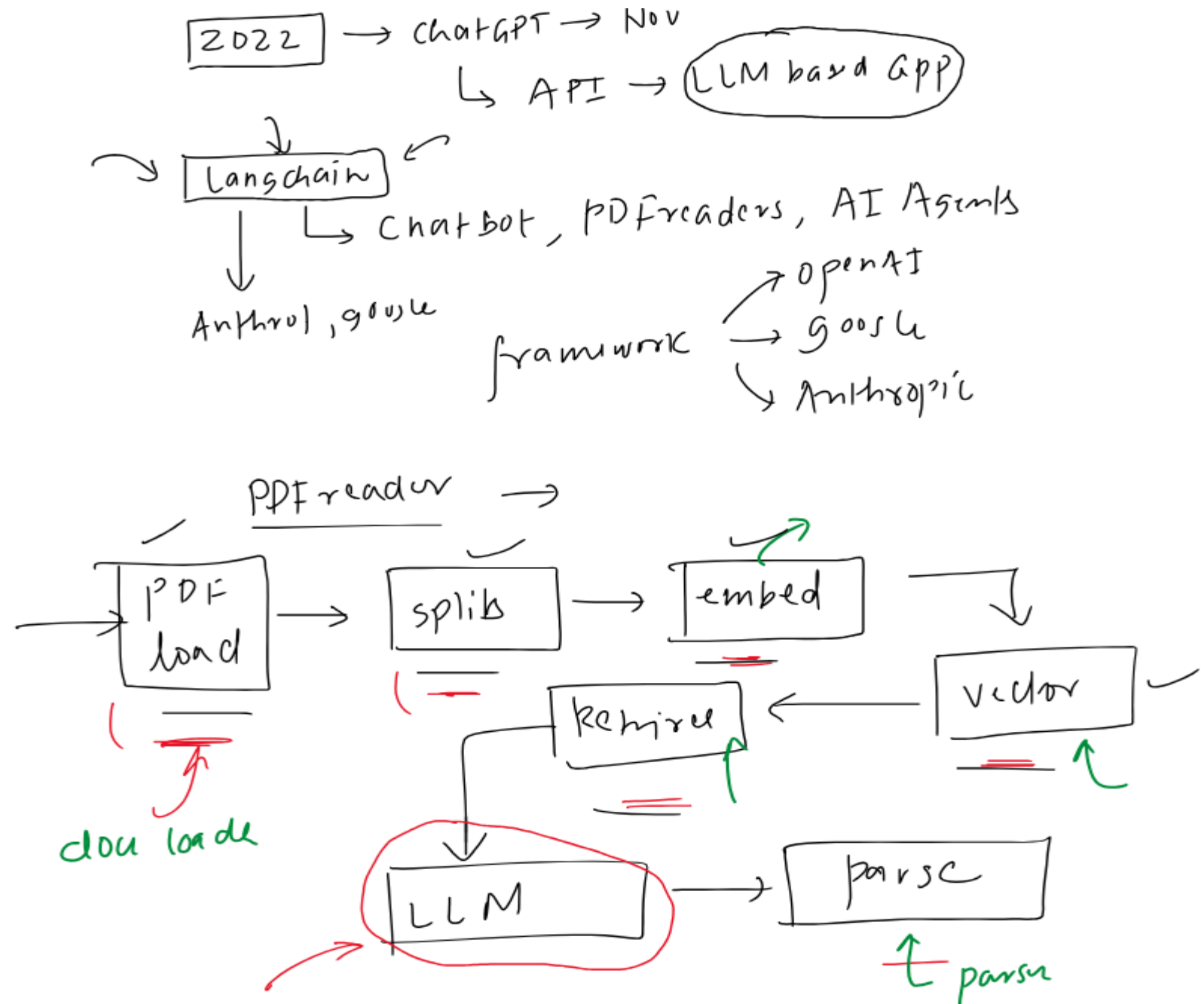
Parallel Chain



Conditional Chain



The Why



Chain Name	Description
LLMChain	Basic chain that calls an LLM with a prompt template.
SequentialChain	Chains multiple LLM calls in a specific sequence.
SimpleSequentialChain	A simplified version of SequentialChain for easier use.
ConversationalRetrievalChain	Handles conversational Q&A with memory and retrieval.
RetrievalQA	Fetches relevant documents and uses an LLM for question-answering.
RouterChain	Directs user queries to different chains based on intent.
MultiPromptChain	Uses different prompts for different user intents dynamically.
HydeChain (Hypothetical Document Embeddings)	Generates hypothetical answers to improve document retrieval.
AgentExecutorChain	Orchestrates different tools and actions dynamically using an agent.

The What

