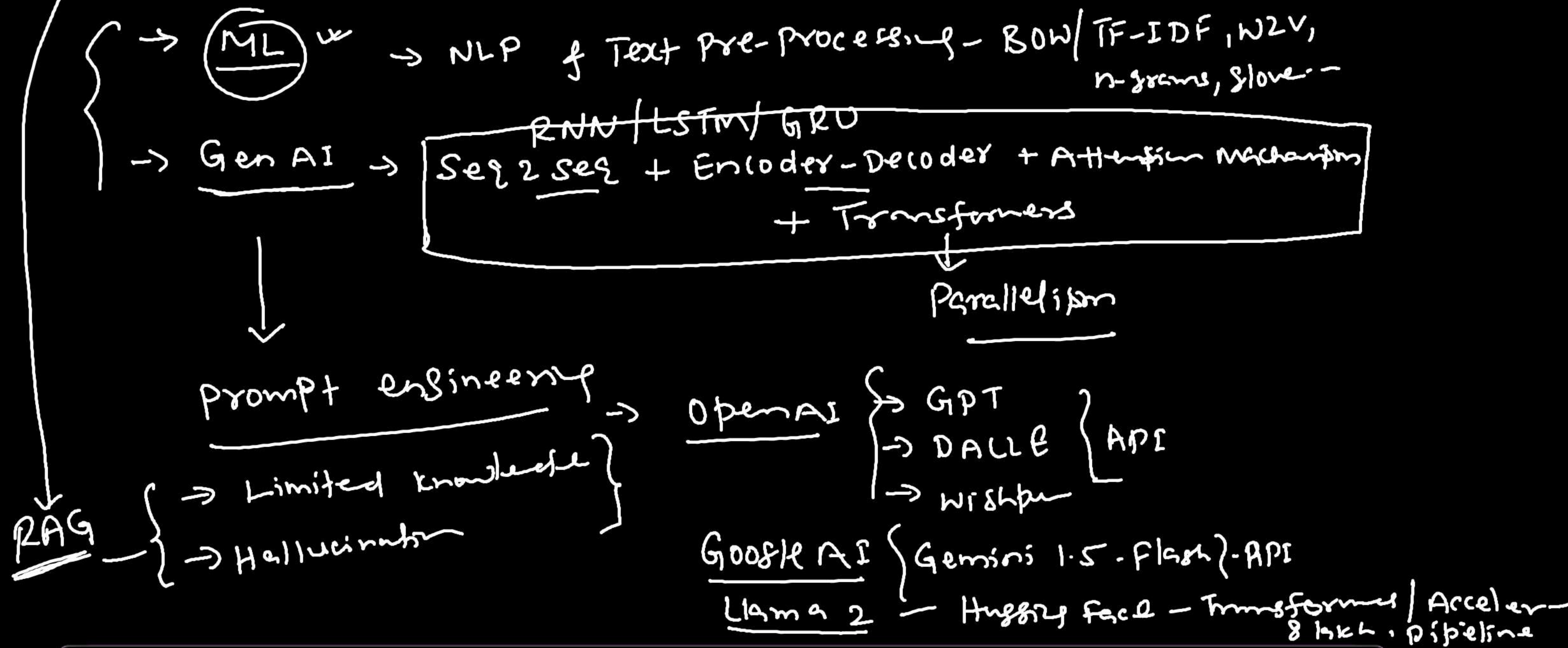
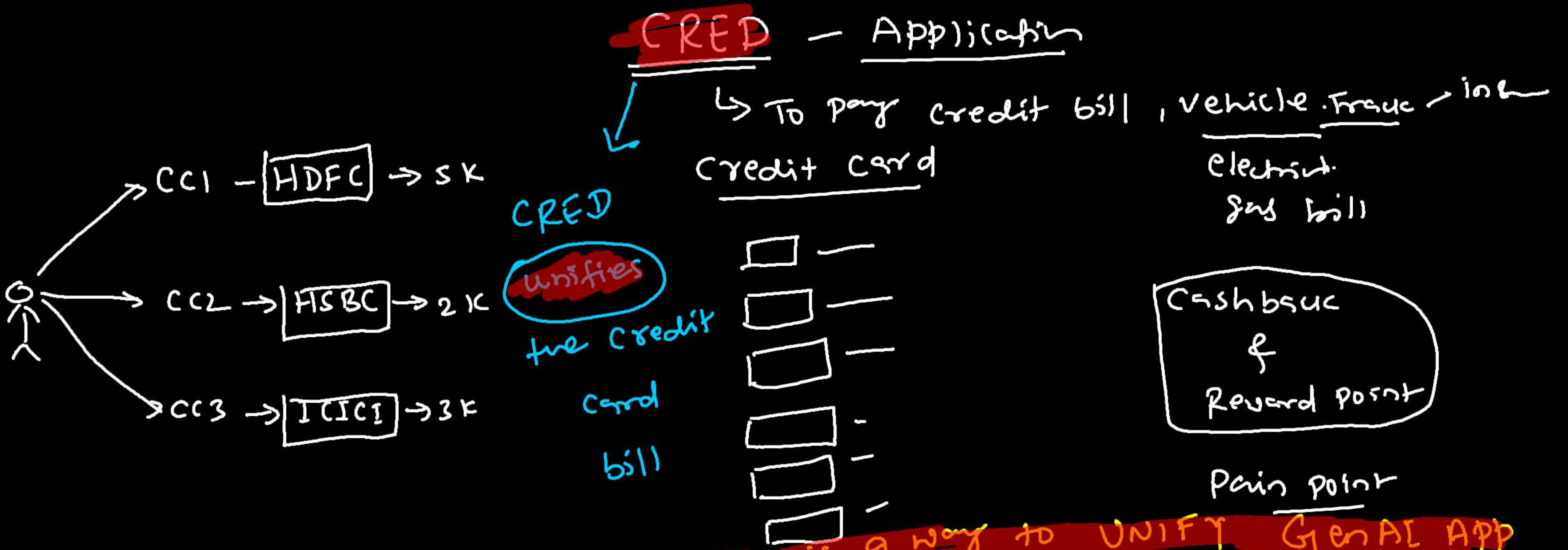


Gen AI

# Intro to Langchain Model (LLM & Chat Models)







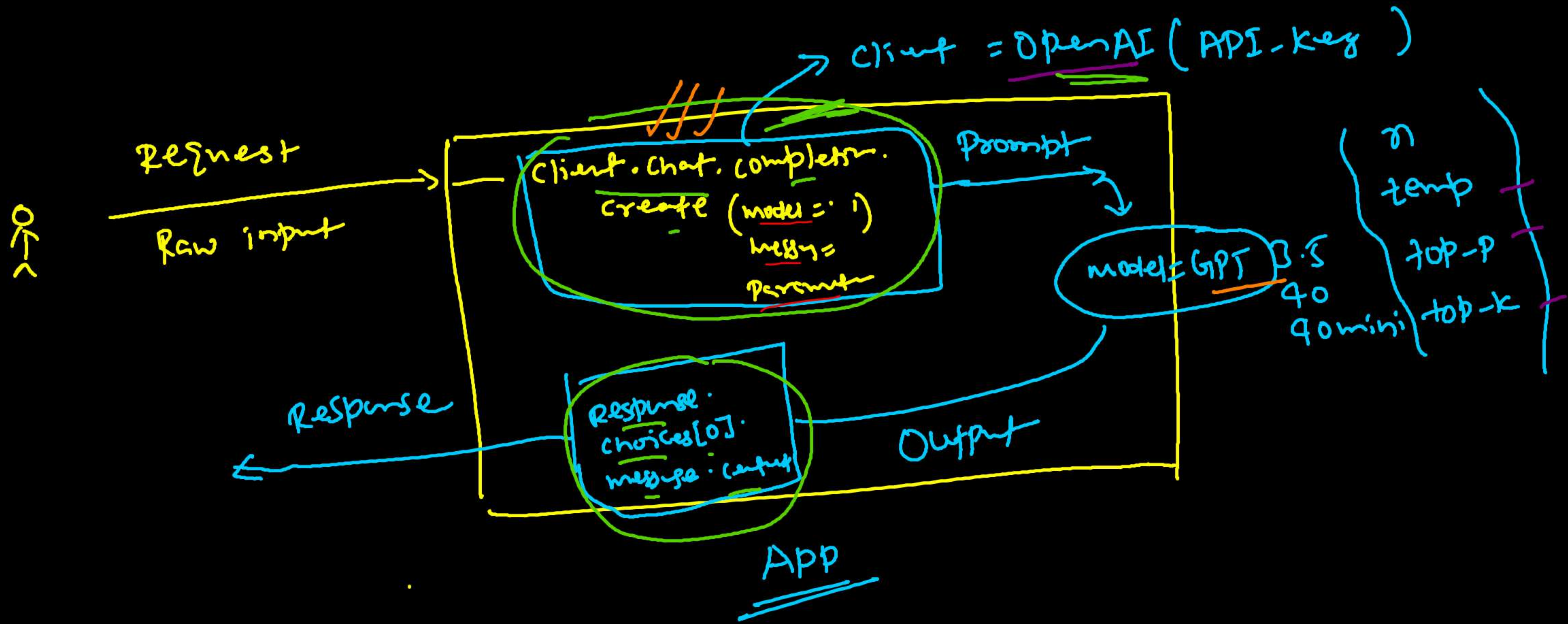
App powered by GenAI

- Gemini
- OpenAI GPT-4o mini
- Hugging face → meta AI - LLaMA 3.1

OpenAPI — code implementation  
or  
Google API — is very different

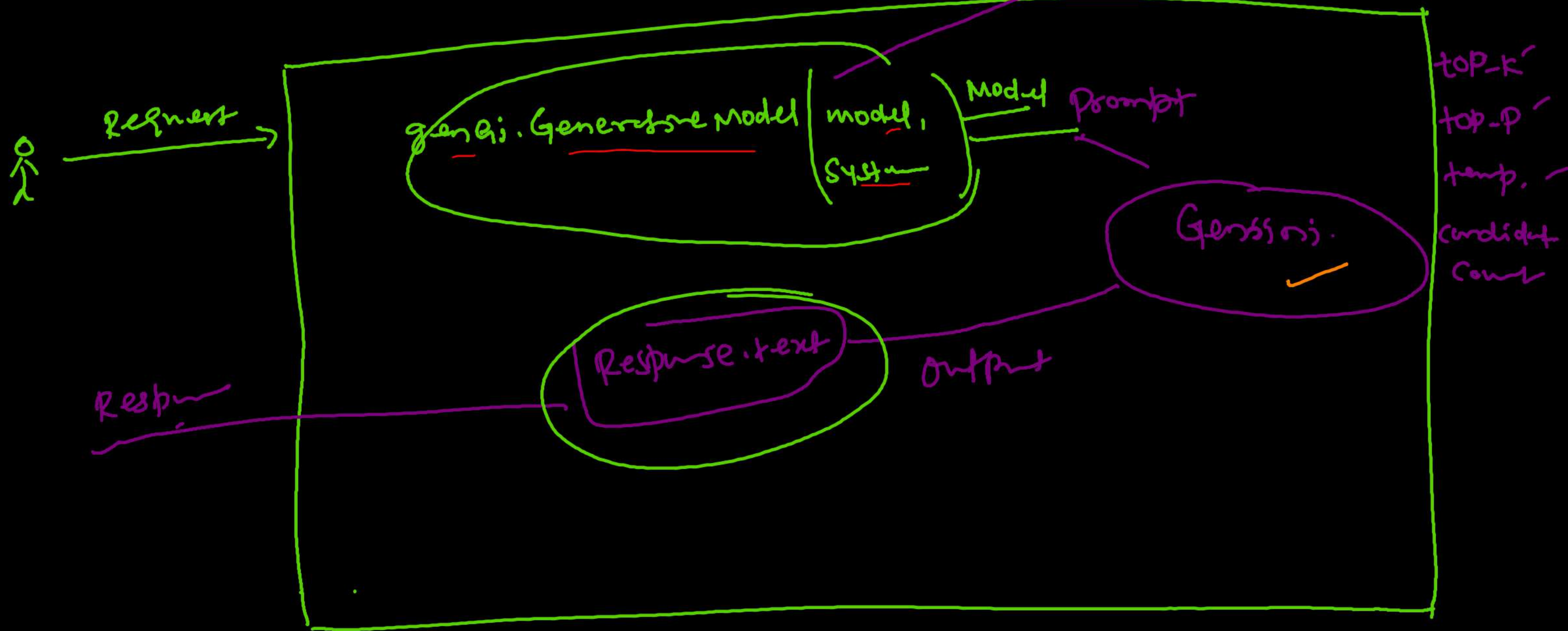


# Building Apps powered by GenAI using OpenAI



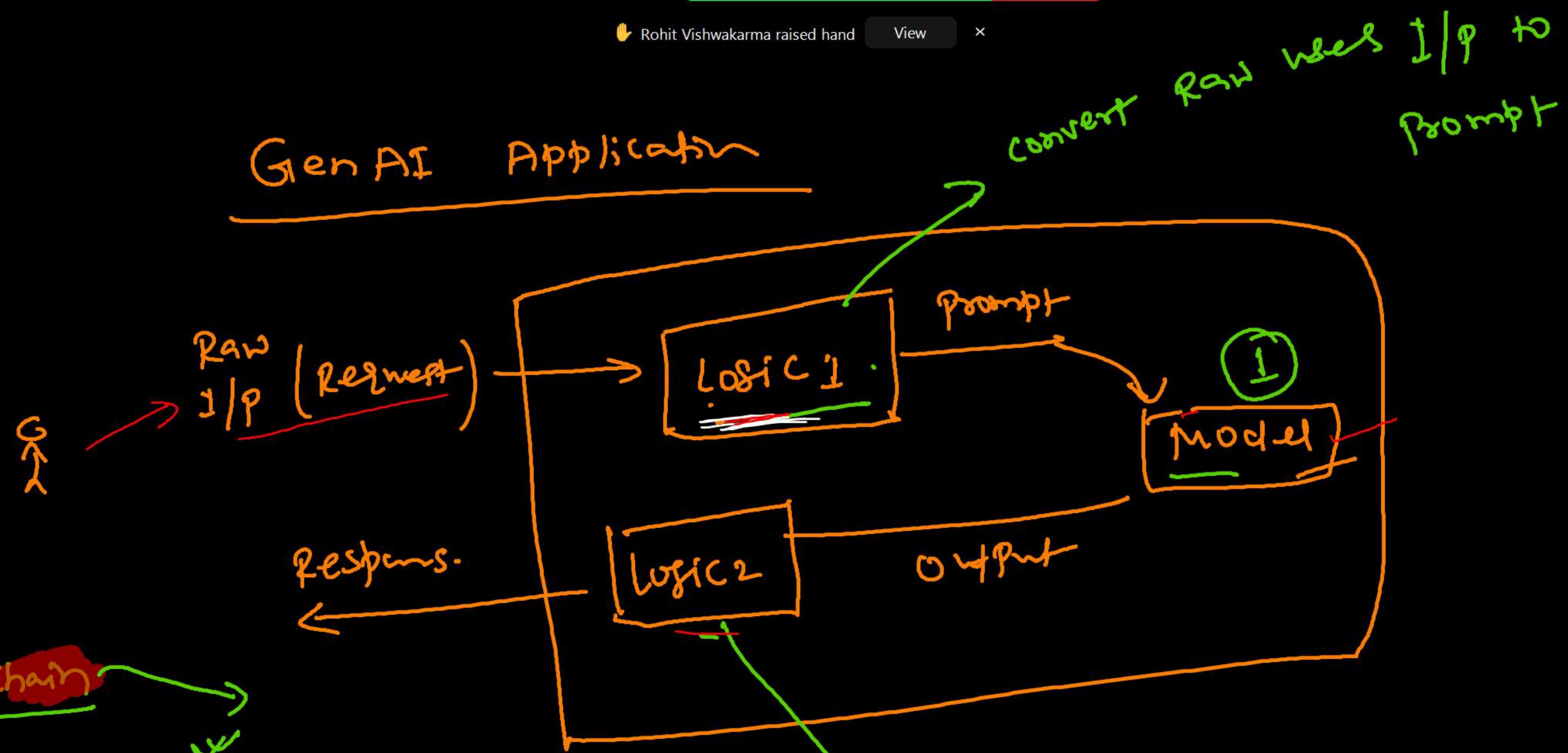


# Building Apps powered by GenAI using Google AI genai.configure(api\_key=)





# Gen AI Application



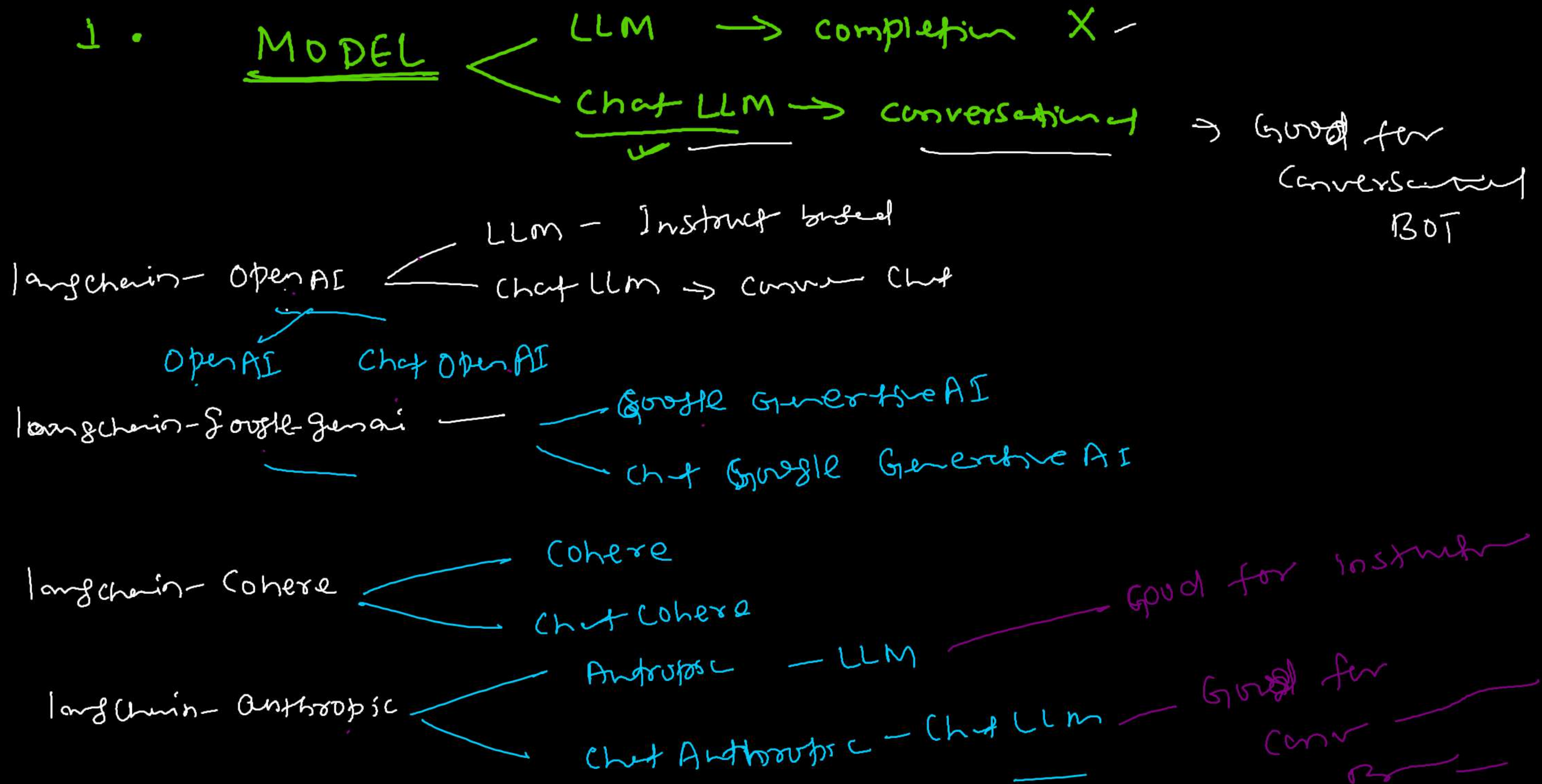
Langchain

- ① Model
- ② Template - Logic 1 - take the user input & convert into a proper prompt
- ④ Chain

③ Output parser (Logic 2) → how to see your output

Chain	Template	Model GPT	Output = Res
G			







(before 2024 mscan) old model  
 LLM → Instruction based model

Open AI → from LangChain = OpenAI import

~~chatmodel~~ = ChatOpenAI(

~~Open AI~~

ChatOpenAI ✓ Chat Based model

Google AI — " — google-gemini

~~chatmodel~~ = ChatGoogle Gemini AI(

Chat Google Gemini AI

Anthropic = " — anthropic

~~chatmodel~~ = ChatAnthropic(

Chat Anthropic

Cohere = " — cohere

~~chatmodel~~ = ChatCohere(

API Key  
 Name of Chat LLM  
 Parameter

chatmodel.invoke(user\_prompt)



## LangChain

① Model — LLM & ChatLLM

② prompt template

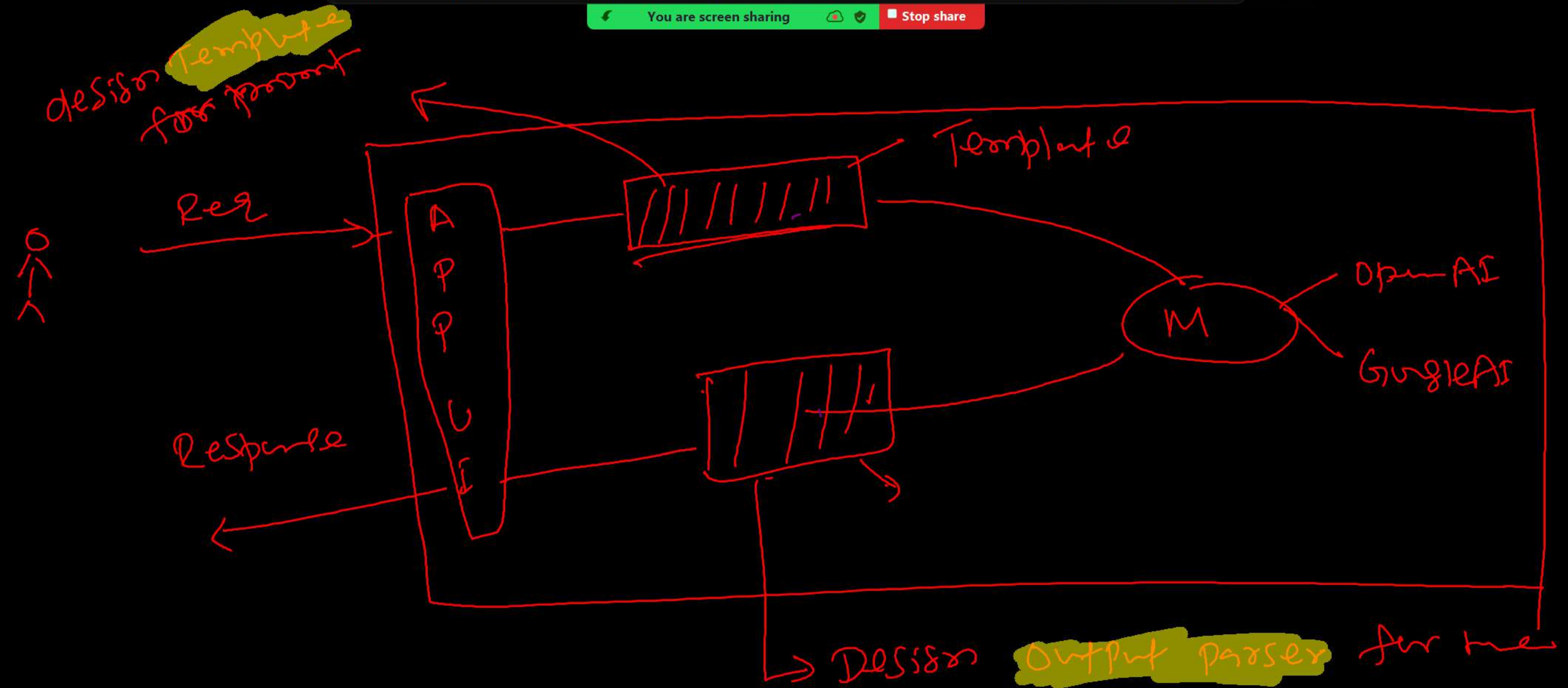
③ output structure

④ chain

⑤ Memory → conversational Bot

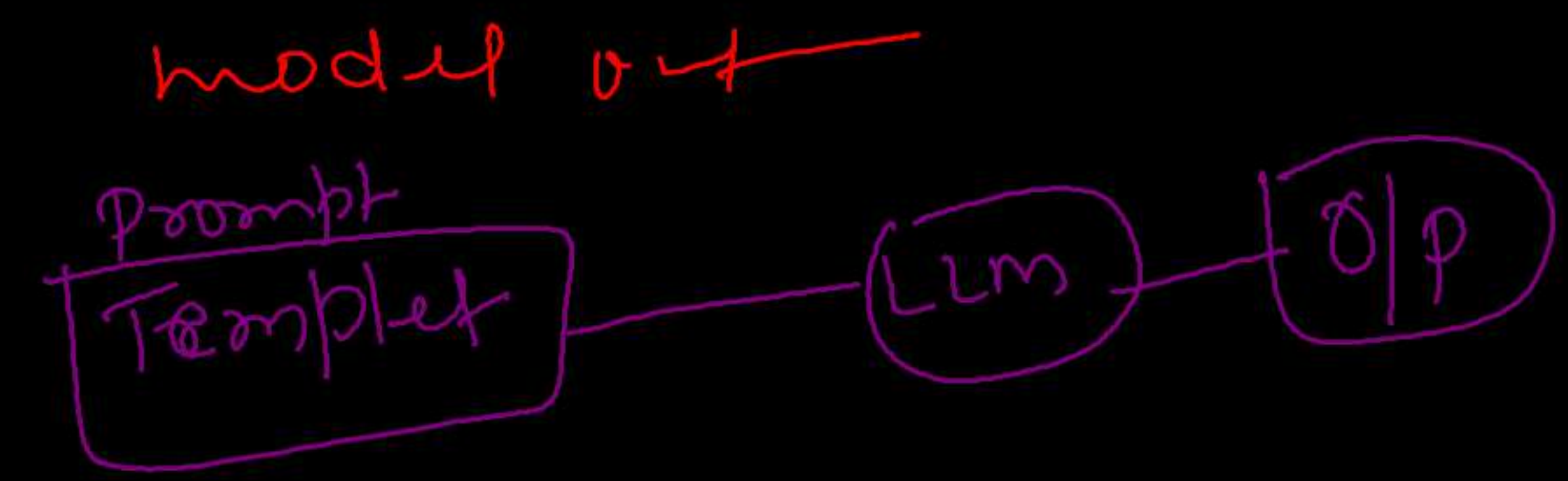
⑥ vector DB — ChromaDB / BERT / GPT





Switch kernel

→ my need here is  
 model Agnostic (independent)  
 Logic 1 - Template  
 Logic 2 = Output parser





# Templates

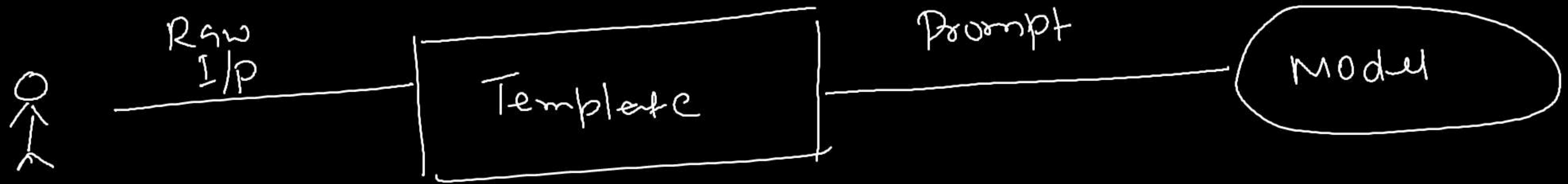
eg ② → Explain topic in defense -  
 ↑  
outlier

Prompt Template

Chat Prompt Template

String - LLM

list of chat mss - chat model



① Tell me a funny Joke about Data Science -



PromptTemplate.from\_template ( Tell me a <sup>add</sup> joke about <sup>Cont</sup> \_\_\_\_\_ )



Tell me a funny  
Joke about Data  
Science.

HumanMessage - - -  
Tell me a funny Joke  
about data science

Human  
System  
ai } → c.



