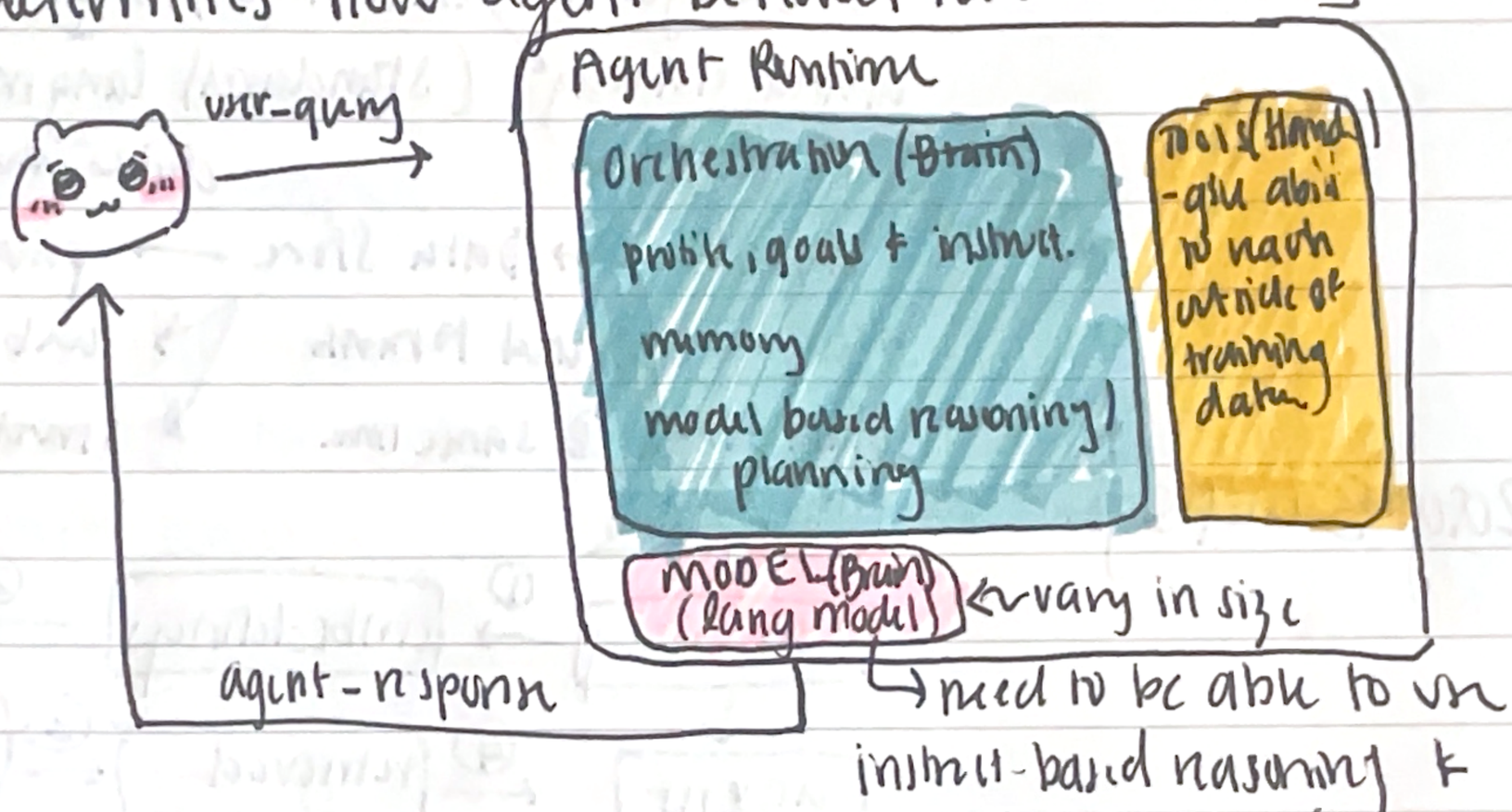
[Day 3] Unit 3a: "Generative AI Agents"

Reading / Pod.

- Gen AI agents: agent = program that goes beyond scan alum gen ai model can do → adds in reasoning / logic
  * add info outside of internal knowledge
  * observes & takes action based on observations
  * given a goal & take steps.
- Agents → built by gen AI mod. (lang model = brain)
  - cognitive architecth. ("internal OS"): set of components, determines how agent behaves & decision making



- flexibility in model, don't pre-trained
  → give tools to help learn
- Tools: allow agents to connect internal reasoning w/ "outside world"
- Orchestration (control center): process that manages how agent takes in info., reasons info, how it uses reasoning to decide. *cycles until goal reached*

Agents vs. Mod.

- Knowledge: mod. limited, agent has access to tools can keep learning
- Handling into overtime
  - Model: makes single pred. based on input (doesn't remember)
  - agent: keeps track of history of interaction
- Models don't have built in support for tools.
- Models don't have "logic" layer.

Cognitive Architectures

- Allow process of: taking in info., reasoning, making decisions, taking actions & learning
  * Heart = orchestration.

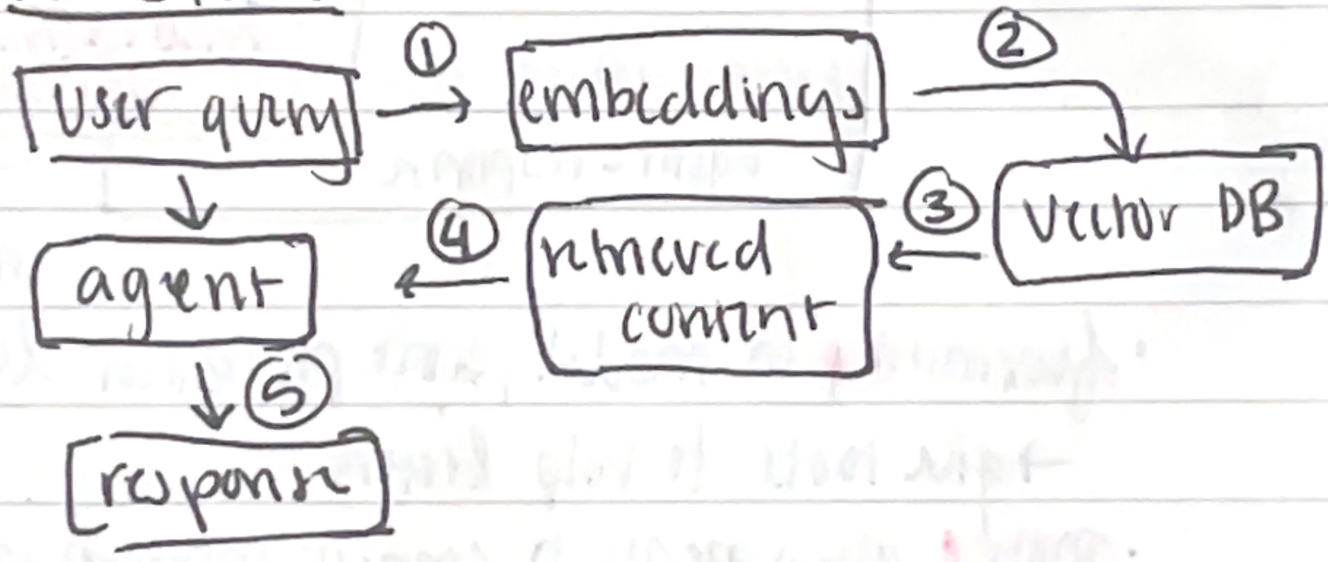
Reason & Act
- Prompt eng. plays ↑ role.
- Framework that encourages lang. model to think step by step, reason & interact w/ environ. w/ tools (ACT)

Chain of Thought
- lets model to think through problem in structured way
  → explain process step by step.
- types: self-consist, active prompt, tree of thought ← explores multi paths.

Tools
- Bridge gap between model & external sources
- 3 categories: extensions, functions & data stores
  - extensions: giving std way to use APIs
  - functions: (standard) lang model decide when to use
  - data stores: ↳ give agents data that dynamic & up to date.

agent → Data Store → private docs
*can be connected to multi data stores @ same time
→ websites
→ struct. data.

Rag Based Sys.
- Workflow



Tools Recap

Exec.	Extensions	Functions	data stores
	Agent-side	Client-side	Agent-side
Use	• Dev. control int. w/ API • Agent int. w/ API	• security/auth. • more control of API calls	• RAG

Enhancing Perf. w/ Targeted Learning
- In-context: learns from info given in prompt (fast, ↓ effect)
- Retrieval based in-context: wider range of knowledge (RAG)
  - augments model knowledge w/ external DBs or info/exs
- Fine-tuning: in depth training in specific area
  - ↑ time & resources

* can be combined.

Vertex AI Agents Takeaways
- Build, deploy & Manage. (Lang Graph)
- Lang models → orchestration → tools.

[LiuStream]

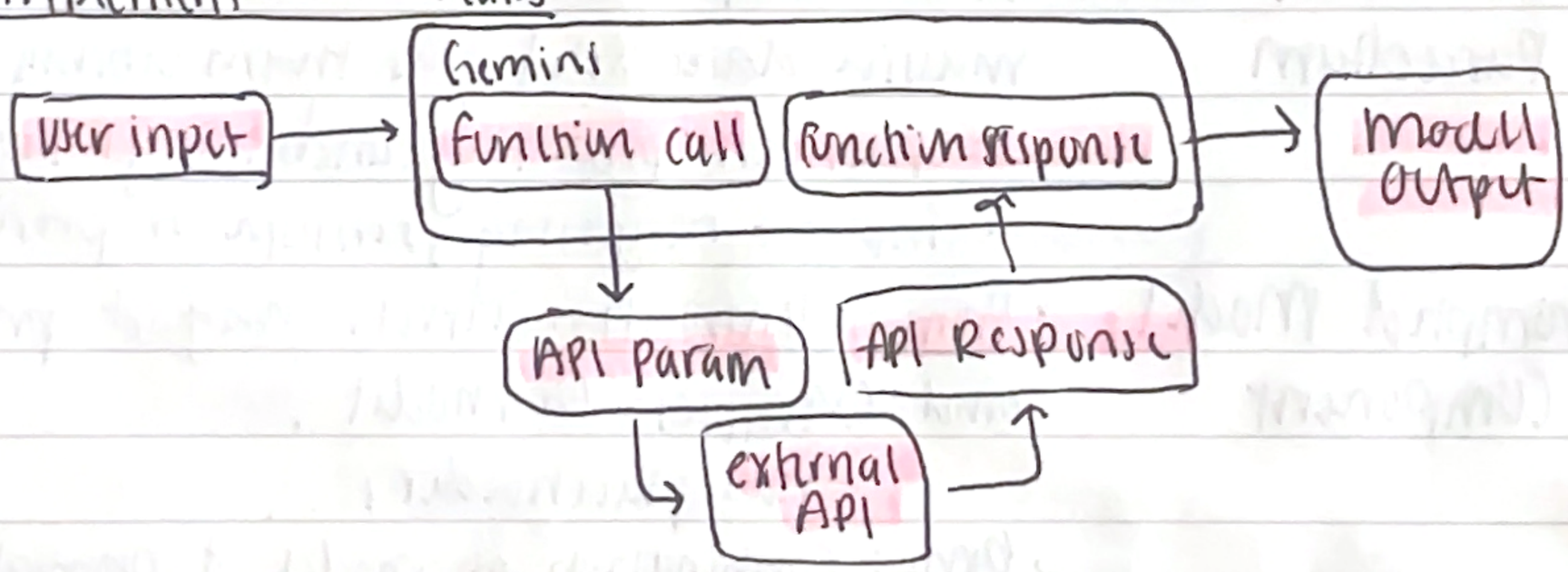
- Notebook LM personalized AI w/ citations.
  - mindmaps: get visual rep of categorized data.
- Project Mariner: research & human & agent interaction
  - chrome ext.
- Design Principles
  - Balance, model always grounded on sources.
- Fund. concepts for scenarios: gets robustness from focus on outcomes & evaluate.
- Mixture of agent experts approach: combining specialized agents to compute a task

[Codelabs]

Function Calling

ex) w/ a database
→ define DB functions: function calling can be impl. w/ OpenAI schema.

Implement function calls:



Compositional function calling: model's ability to compose user-prov. functions w/ gen. code.
→ model looks @ tools, makes functions, executes.

Building an Agent w/ LangGraph
A chat bot pretty cool :)

LangGraph: built around graph structure
→ as dev you define an application graph that models state transitions for ur applic. (+def state schema)
→ nodes: rep. an action that can be taken & makes changes to the state.
→ edge: rep. a transition between states, defining flow of program. (can be conditional to allow for branching)