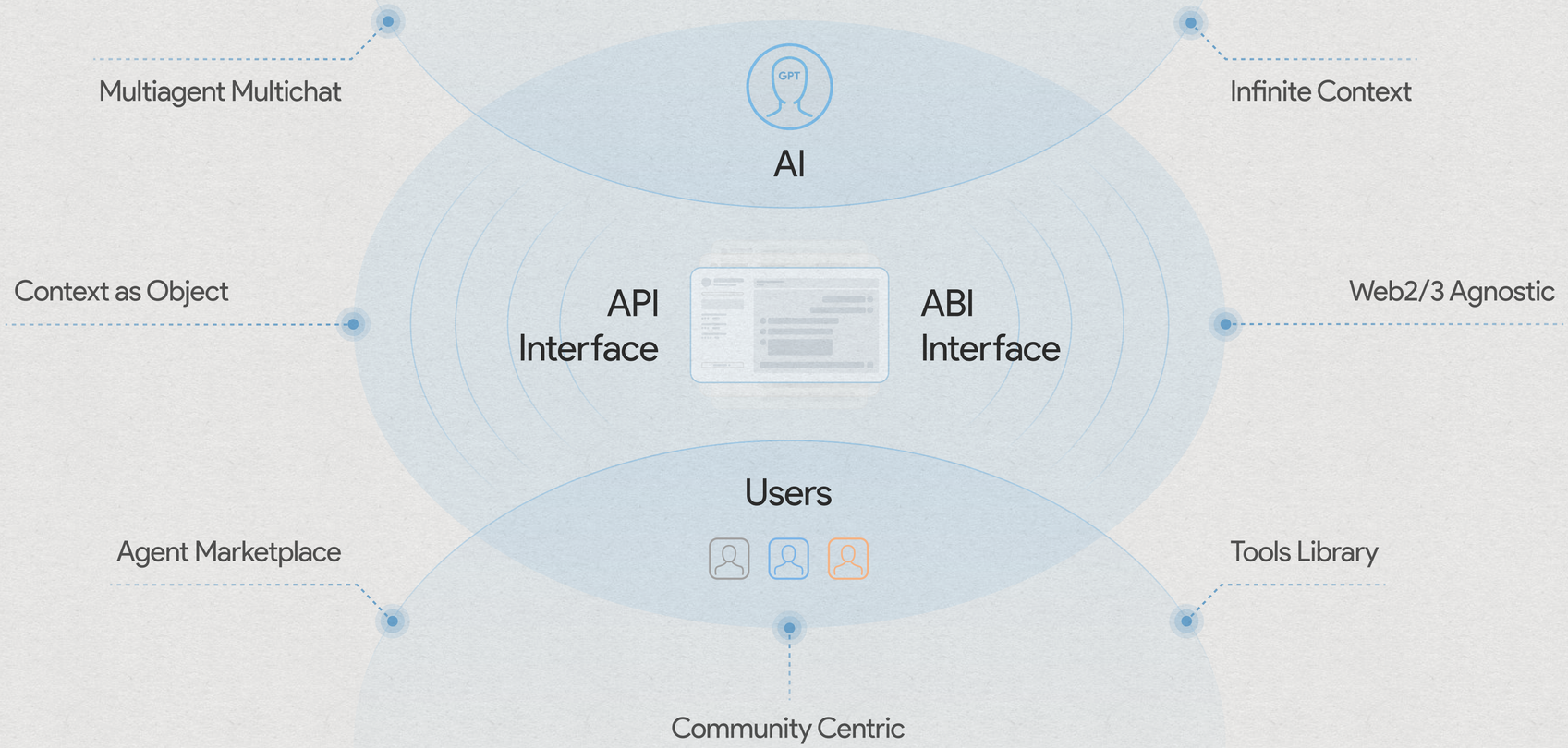
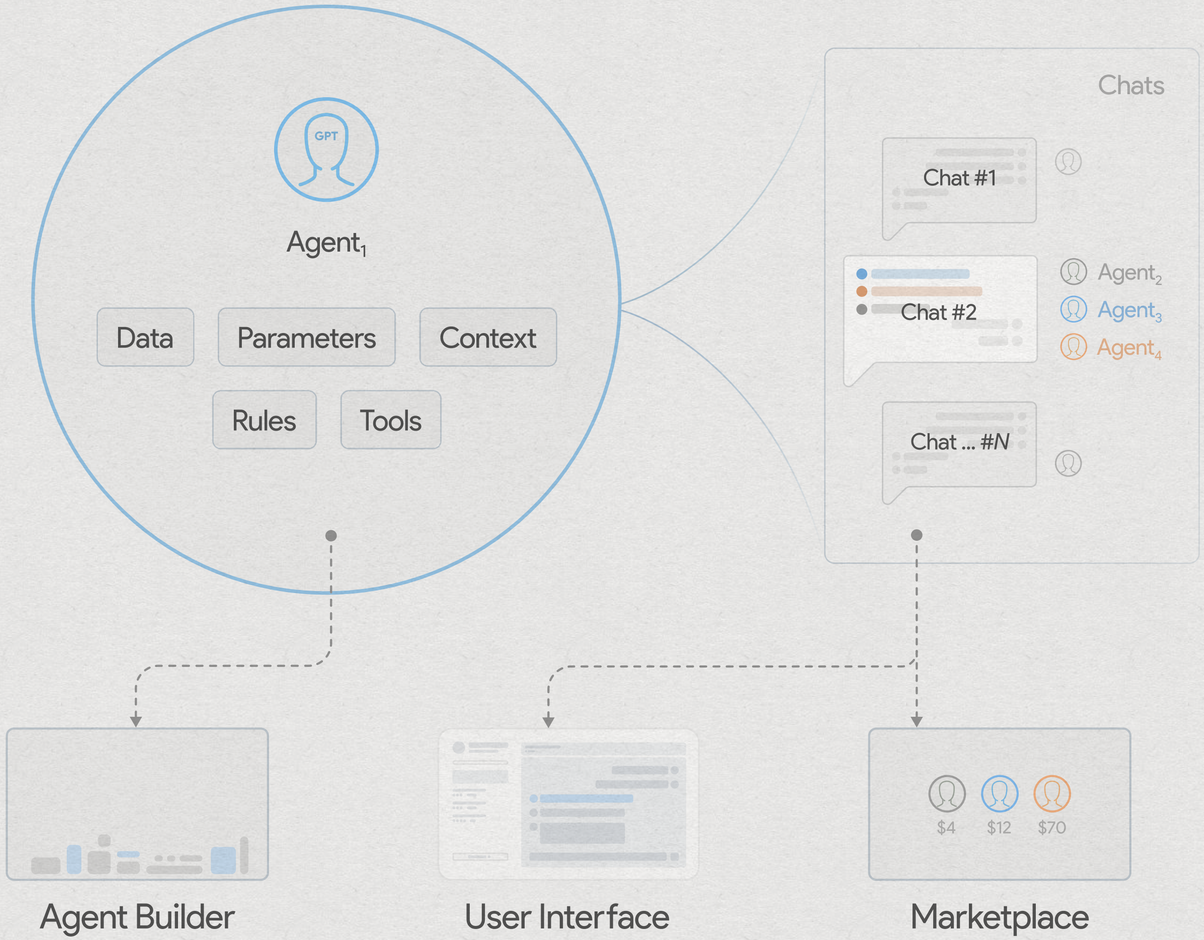

Persistent AI

Table of Contents

The future of interfaces are chats. Persistent AI is holding many contexts in 1 app, 1 interface.



The future of UX is chat and persistent AI agents



1. Agents interact in chats. Chats are user defined.

2. Agents are aggregated and embedded context: various content, datasets, rules, tools and parameters. Agents' contexts are persistent.

3. Agents interact in chats. Chats' history augments contexts of agents participating in them.

4. Agents live in a persistent database. Agents can be built, RLHF'ed, augmented with tools and rules as well as rented out on a market place.

Overview of the Architecture

User Interface

Chatrooms and agents marketplace — User purchases Agents and adds them to created chats to collaborate.

An **Admin User** can invite a friend to the chat and they can together communicate with Agents

Maintainer Users are accessing their Agents through chat rooms as well. All the changes maintainer does to the Agent's Context are saved to the persistent database so that maintainers can learn and upgrade their Agents. Also, **maintainers** can view and edit the persistent database directly.

When an **Admin User** changes Agent's parameters, they are being saved only for that chatroom. Any **User** can import PDFs, CSVs and other documents. These documents form the context DB of the chatroom that is available to all added Agents.

Backend

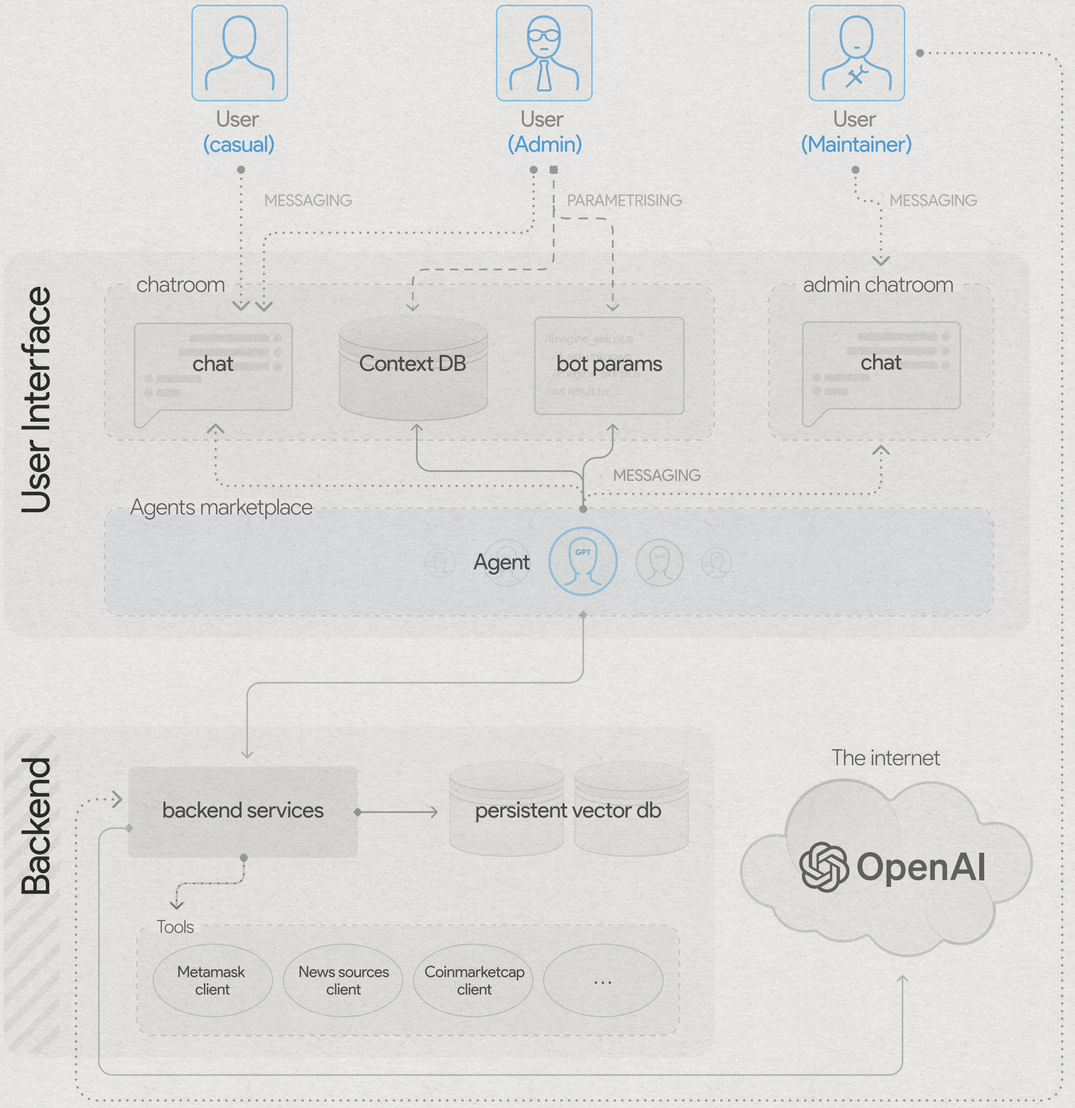
Persistent vector DB — storing Agents' datasets.

Routing backend — routing all the requests between our DB, OpenAI API, and the Tools.

Tools — can be used by Agents (incl. clients for different services).

Orchestrator Agent

— organising the work of other Agents in cases where many Agents are involved in solving a single task, or when the a task requires multiple (many) iterations of of a single Agent.



Long Context

1. Log in with a wallet to initiate a session (i.e. private key is the trader's ID).

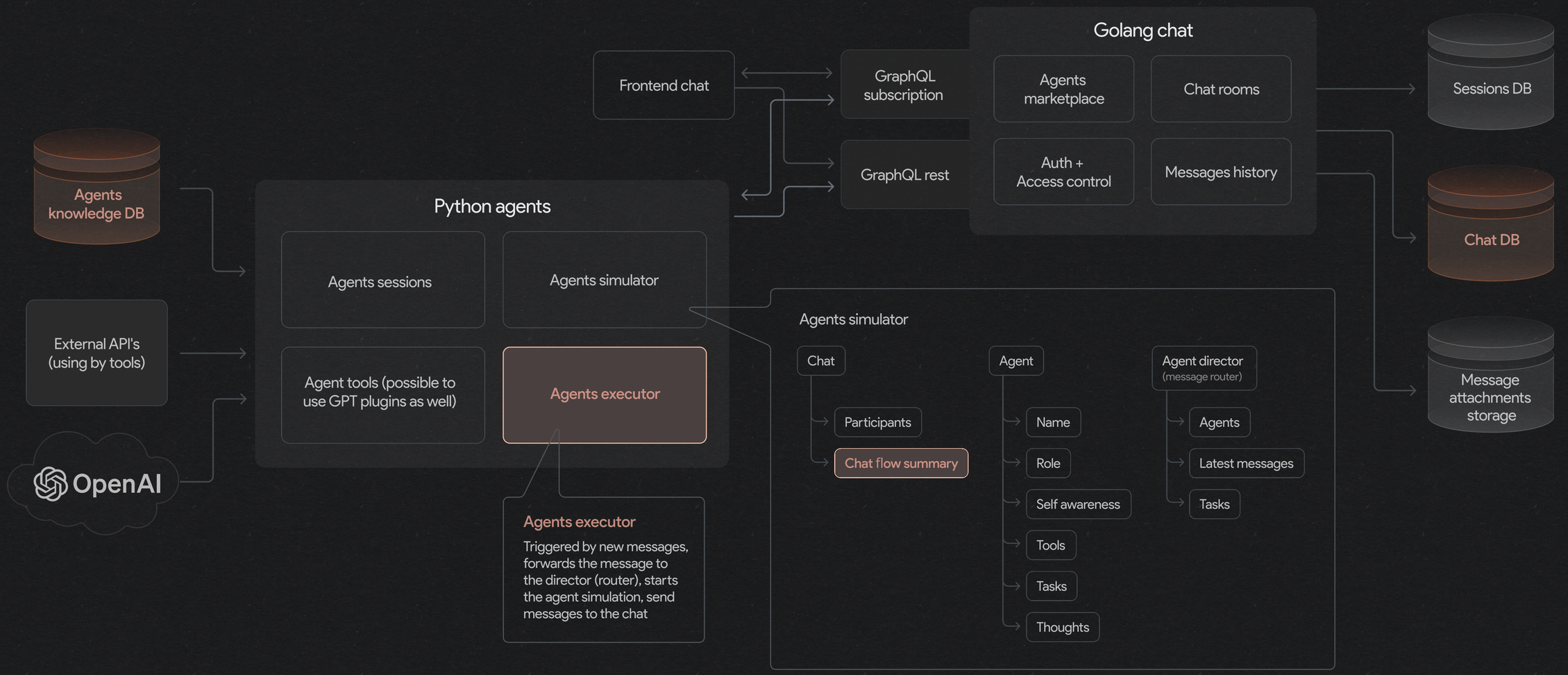
2. Trader's ID aggregates Context both, interchat and intrachat (i.e. between Agents for all Chats and between Chats for all Agents).

3. Agents can access the internet, customized news feeds, custom charting Tools, trader's portfolio data, import function for custom sources (docs, PDFs, etc.)

4. Practically unlimited Context (constantly embedding and appending) can be made available for trades/news/assets/ charting requests/topics.

5. Knows you better than your friends - the Agent can be augmented with Tools which it uses to query (and analyze once taught) the data from point 4.

6. The core idea is to create a positive feedback loop between the time one uses the product and the quality of UX (think Telegram).


Golang chat
Frontend chat
GraphQL subscription
GraphQL rest
Agents marketplace
Chat rooms
Auth + Access control
Messages history
Sessions DB
Chat DB
Message attachments storage
Agents knowledge DB
External API's (using by tools)
OpenAI
Python agents
Agents sessions
Agents simulator
Agent tools (possible to use GPT plugins as well)
Agents executor
Agents executor
Triggered by new messages, forwards the message to the director (router), starts the agent simulation, send messages to the chat
Agents simulator
Chat
Participants
Chat flow summary
Agent
Name
Role
Self awareness
Tools
Tasks
Thoughts
Agent director (message router)
Agents
Latest messages
Tasks

Wallet Login & Interaction

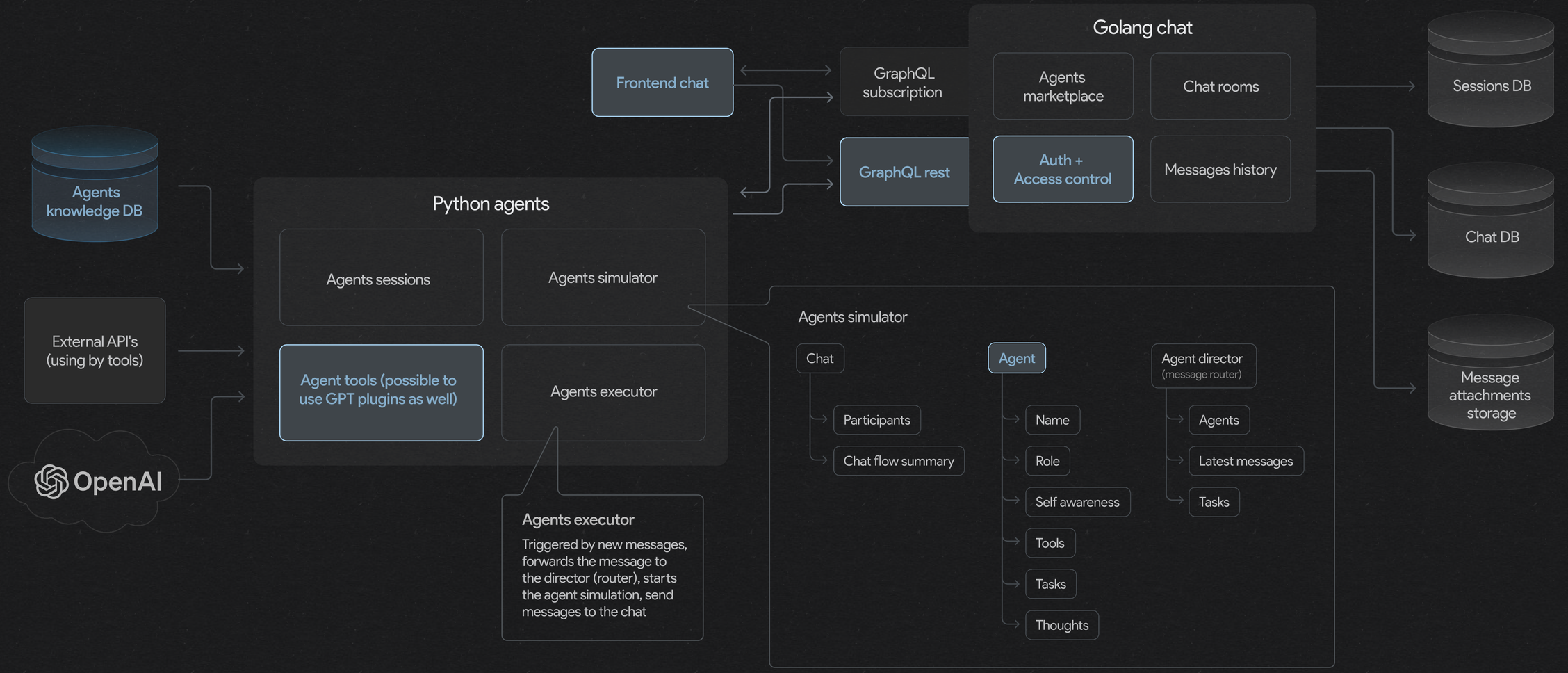
Can initiate a session with Metamask and practically anything else that WalletConnect supports (can be extended to any blockchain and wallet).

Private key is used to sign transactions (note that the Agent does not sign transactions).

On-chain data from user's account is used to continuously augment Agent's Context.

Therefore the following functionality is enabled:

1. Execution of trades and other types of supported transactions.
2. Setting push notifications for executing transactions.
3. Upload portfolio and transaction history details into Context.
4. Integrations with on-chain data analytics providers like Nansen.



Agent Modeling & Multi-Agent Interactions

Agents can import interchat/ interbot Contexts

(i.e. treat trader's Context as an Agent that can be imported/exported, augmented, etc.).

Multiple agents in a chat:

Chat interface — different Agents can be added to interact with each other. Chats can be programed to have Agents reach a particular goal.

Every Agent has its own "personality", set of Tools, and Context. Contexts are additive interchat.

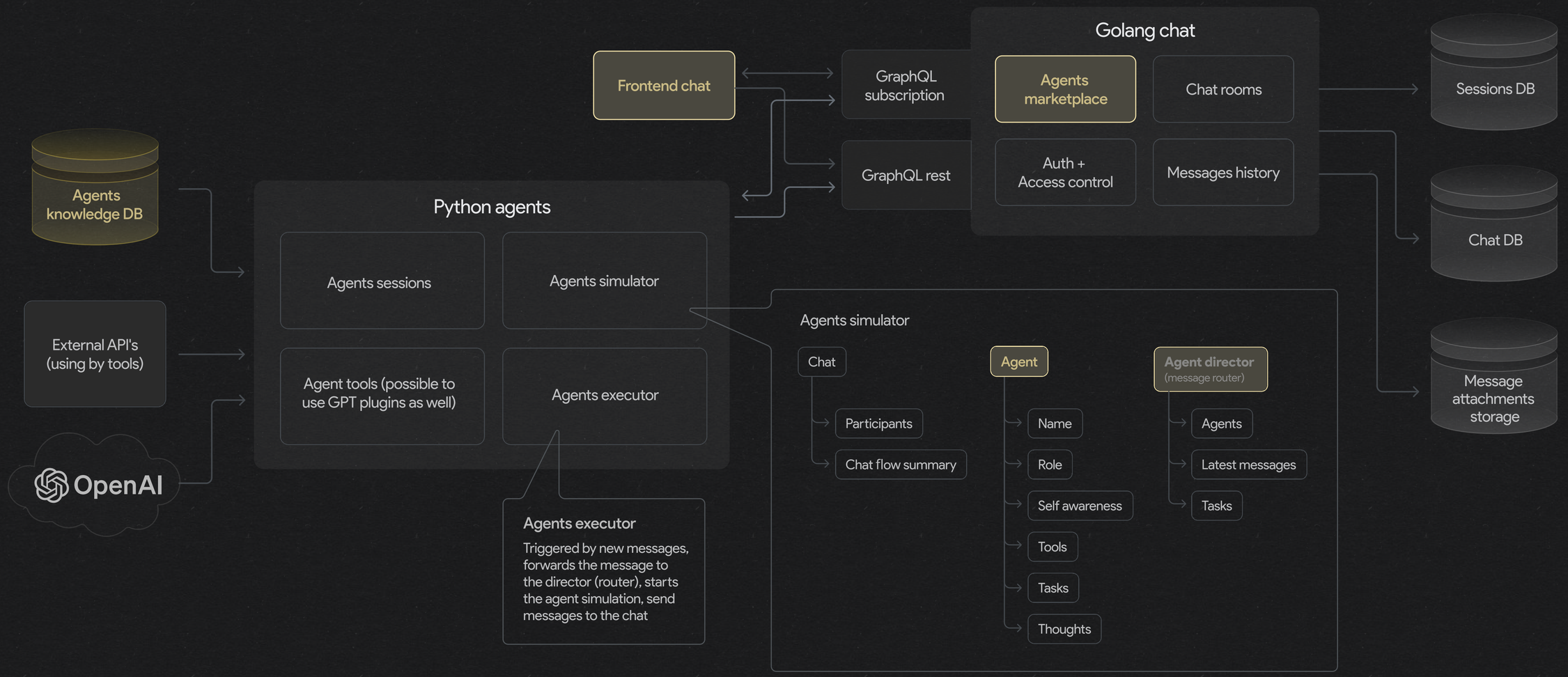
Agents can be traded on a Marketplace. A purchased Agent can be modifiable or not.

Agent Modeling:

Chat interface — upload content and tune Agent' Context, personality, Tools, Rules of interaction. Context is practically unlimited.

Various content formats — audio, plain text, PDF, CSV, Word, Excel, etc.

Tools management — create Tools with tailored functionality, such as graphs building, voice commands, on-chain data analytics, etc.


Golang chat
Frontend chat
GraphQL subscription
GraphQL rest
Agents marketplace
Chat rooms
Auth + Access control
Messages history
Sessions DB
Chat DB
Message attachments storage
Agents knowledge DB
External API's (using by tools)
OpenAI
Python agents
Agents sessions
Agents simulator
Agent tools (possible to use GPT plugins as well)
Agents executor
Agents executor
Triggered by new messages, forwards the message to the director (router), starts the agent simulation, send messages to the chat
Agents simulator
Chat
Participants
Chat flow summary
Agent
Name
Role
Self awareness
Tools
Tasks
Thoughts
Agent director
(message router)
Agents
Latest messages
Tasks

Tools

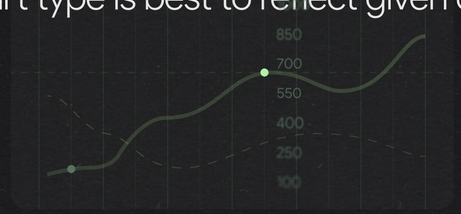
(Charts, News, & Alerts)

Agents have access to selected news sources.

The Context can be set to automatically expand based on some preset criteria

(e.g. load all news by certain hashtag everyday at 9am).

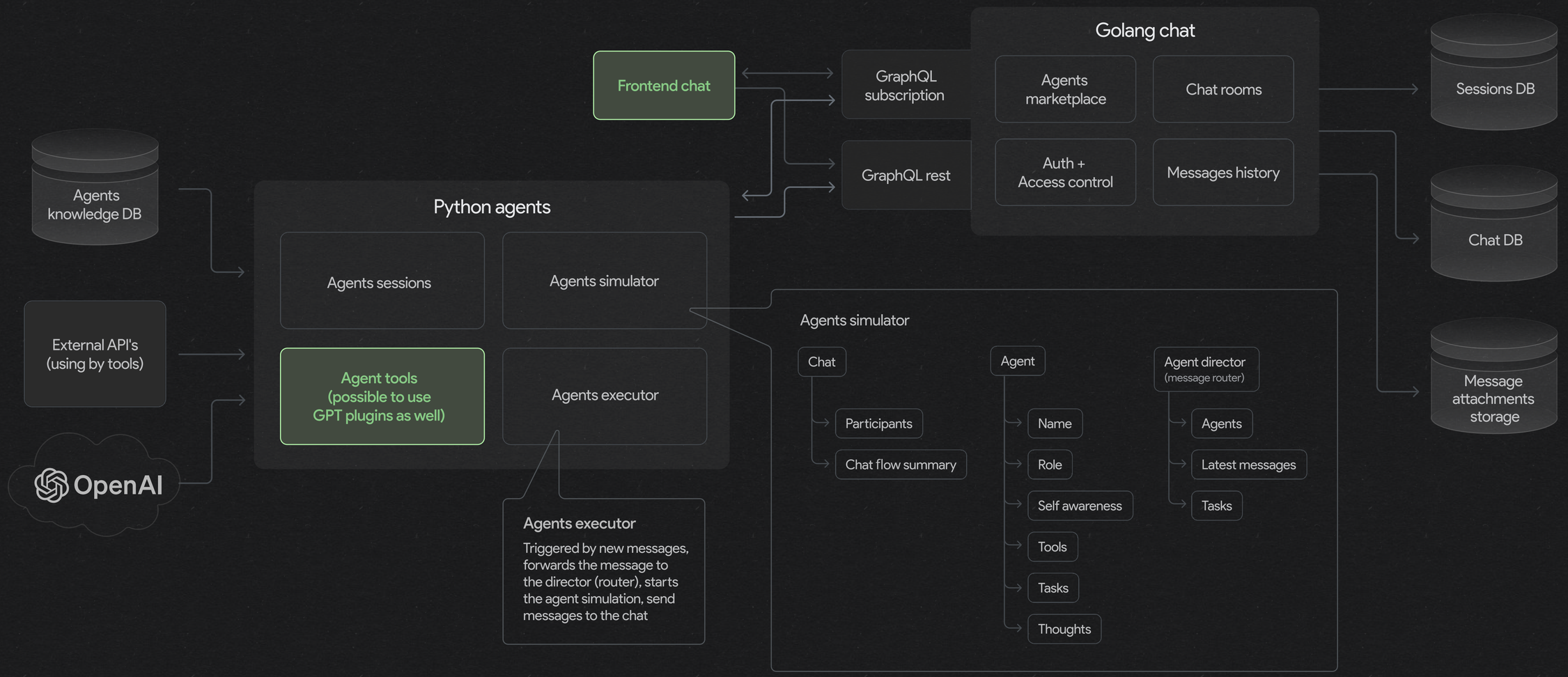
Custom charts can be created based on **voice input**. Agents decide (once trained) themselves which chart type is best to reflect given data.



The Agents are able to **summarise the news** (e.g. of the day and by topic); and (once trained) provide analysis based on these and past news.

A user can instruct Agents to create **custom alerts** with arbitrary triggers.

! *Note that these are simply Tools rather than trained Agents.*


Golang chat
Frontend chat
GraphQL subscription
GraphQL rest
Agents marketplace
Chat rooms
Auth + Access control
Messages history
Sessions DB
Chat DB
Message attachments storage
Agents knowledge DB
External API's (using by tools)
OpenAI
Python agents
Agents sessions
Agents simulator
Agent tools (possible to use GPT plugins as well)
Agents executor
Agents executor
Triggered by new messages, forwards the message to the director (router), starts the agent simulation, send messages to the chat
Agents simulator
Chat
Participants
Chat flow summary
Agent
Name
Role
Self awareness
Tools
Tasks
Thoughts
Agent director (message router)
Agents
Latest messages
Tasks

Persistent AI

Contact Us
contact@persistentai.org