Okechukwu Okey Enelamah, Founder of African Capital Alliance by Chika Uwazie and Eche Emole

https://www.youtube.com/watch?v=q2jZp7o6CIw

"Water does not run uphill in Africa" - Okey Enelamah

Multi AI Agents Systems with crewAI

https://learn.deeplearning.ai/courses/multi-ai-agent-systems-with-crewai/lesson/wwou5/introduction

Think like a Manager
What is the goal and what is the process?
What kind of people would I need hire to get this done?

Example of Agent Roles etc:

Okay
- Researcher
- Writer
- Financial Analyst

Better
- HR Research Specialist
- Senior copywriter
- FINRA Approved Analyst

Agents "self improve" using memory
Guardrails prevents agents from going into rabbit holes(unproductive)

Agents always attempt to get to an answer(

Focus

- narrowly defined task
- Specific agent roles and objetives
- limited set of tools assigned to one agent

What Makes a Good Agent

1. Role Playing
2. Focus
3. Tools
4. Cooperation
5. Guardrails
6. Memory

Tools

- versatile
- fault tolerant
- caching

Examples Code

Agents

```
planner = Agent(
    role="Content Planner",
    goal="Plan engaging and factually accurate content on {topic}",
    backstory="You're working on planning a blog article "
              "about the topic: {topic}."
              "You collect information that helps the "
              "audience learn something "
              "and make informed decisions. "
              "Your work is the basis for "
              "the Content Writer to write an article on this topic.",
    allow_delegation=False,
verbose=True
```

```
)

writer = Agent(
    role="Content Writer",
    goal="Write insightful and factually accurate "
         "opinion piece about the topic: {topic}",
    backstory="You're working on a writing "
              "a new opinion piece about the topic: {topic}. "
              "You base your writing on the work of "
              "the Content Planner, who provides an outline "
              "and relevant context about the topic. "
              "You follow the main objectives and "
              "direction of the outline, "
              "as provide by the Content Planner. "
              "You also provide objective and impartial insights "
              "and back them up with information "
              "provide by the Content Planner. "
              "You acknowledge in your opinion piece "
              "when your statements are opinions "
              "as opposed to objective statements.",
    allow_delegation=False,
    verbose=True
)

editor = Agent(
    role="Editor",
    goal="Edit a given blog post to align with "
         "the writing style of the organization. ",
    backstory="You are an editor who receives a blog post "
              "from the Content Writer. "
              "Your goal is to review the blog post "
              "to ensure that it follows journalistic best practices,"
              "provides balanced viewpoints "
              "when providing opinions or assertions, "
              "and also avoids major controversial topics "
              "or opinions when possible.",
    allow_delegation=False,
    verbose=True
```

)

sales_rep_agent = Agent(role="Sales Representative", goal="Identify high-value leads that match " "our ideal customer profile", backstory=("As a part of the dynamic sales team at CrewAI, " "your mission is to scour " "the digital landscape for potential leads. " "Armed with cutting-edge tools " "and a strategic mindset, you analyze data, " "trends, and interactions to " "unearth opportunities that others might overlook. " "Your work is crucial in paving the way " "for meaningful engagements and driving the company's growth."), allow_delegation=False, verbose=True)

lead_sales_rep_agent = Agent(role="Lead Sales Representative", goal="Nurture leads with personalized, compelling communications", backstory=("Within the vibrant ecosystem of CrewAI's sales department, " "you stand out as the bridge between potential clients " "and the solutions they need." "By creating engaging, personalized messages, " "you not only inform leads about our offerings " "but also make them feel seen and heard." "Your role is pivotal in converting interest " "into action, guiding leads through the journey " "from curiosity to commitment."), allow_delegation=False, verbose=True)

Tasks

```
plan = Task(
    description=(
        "1. Prioritize the latest trends, key players, "
            "and noteworthy news on {topic}.\n"
        "2. Identify the target audience, considering "
            "their interests and pain points.\n"
        "3. Develop a detailed content outline including "
            "an introduction, key points, and a call to action.\n"
        "4. Include SEO keywords and relevant data or sources."
    ),
    expected_output="A comprehensive content plan document "
        "with an outline, audience analysis, "
        "SEO keywords, and resources.",
    agent=planner,
)


write = Task(
```

```
    description=(
        "1. Use the content plan to craft a compelling "
            "blog post on {topic}.\n"
        "2. Incorporate SEO keywords naturally.\n"
"3. Sections/Subtitles are properly named "
            "in an engaging manner.\n"
        "4. Ensure the post is structured with an "
            "engaging introduction, insightful body, "
            "and a summarizing conclusion.\n"
        "5. Proofread for grammatical errors and "
            "alignment with the brand's voice.\n"
    ),
    expected_output="A well-written blog post "
        "in markdown format, ready for publication, "
        "each section should have 2 or 3 paragraphs.",
    agent=writer,
)

edit = Task(
    description=("Proofread the given blog post for "
                 "grammatical errors and "
                 "alignment with the brand's voice."),
    expected_output="A well-written blog post in markdown format, "
                    "ready for publication, "
                    "each section should have 2 or 3 paragraphs.",
    agent=editor
)

lead_profiling_task = Task( description=( "Conduct an in-depth analysis of {lead_name}, " "a company in the {industry} sector " "that recently showed interest in our solutions. " "Utilize all available data sources " "to compile a detailed profile, " "focusing on key decision-makers, recent business " "developments, and potential needs " "that align with our offerings. " "This task is crucial for tailoring " "our engagement strategy effectively.\n" "Don't make assumptions and " "only use information you absolutely sure about." ), expected_output=( "A comprehensive report on {lead_name}, " "including company background, " "key personnel, recent milestones, and identified needs. " "Highlight potential areas where " "our solutions can provide value, " "and suggest personalized
```

engagement strategies."), tools=[directory_read_tool, file_read_tool, search_tool], agent=sales_rep_agent,)

personalized_outreach_task = Task(description=("Using the insights gathered from " "the lead profiling report on {lead_name}, " "craft a personalized outreach campaign " "aimed at {key_decision_maker}, " "the {position} of {lead_name}. " "The campaign should address their recent {milestone} " "and how our solutions can support their goals. " "Your communication must resonate " "with {lead_name}'s company culture and values, " "demonstrating a deep understanding of " "their business and needs.\n" "Don't make assumptions and only " "use information you absolutely sure about."), expected_output=("A series of personalized email drafts " "tailored to {lead_name}, " "specifically targeting {key_decision_maker}." "Each draft should include " "a compelling narrative that connects our solutions " "with their recent achievements and future goals. " "Ensure the tone is engaging, professional, " "and aligned with {lead_name}'s corporate identity."), tools=[sentiment_analysis_tool, search_tool], agent=lead_sales_rep_agent,)

Crew

```
crew = Crew(
   agents=[planner, writer, editor],
   tasks=[plan, write, edit],
   verbose=2
)
```

```
crew = Crew( agents=[sales_rep_agent, lead_sales_rep_agent],
 tasks=[lead_profiling_task,       personalized_outreach_task], verbose=2,
memory=True
)
```

Running the Crew

```
result = crew.kickoff(inputs={"topic": "Artificial Intelligence"})
```

```
from IPython.display import Markdown
Markdown(result)
```

inputs = { "lead_name": "DeepLearningAI", "industry": "Online Learning Platform", "key_decision_maker": "Andrew Ng", "position": "CEO", "milestone": "product launch" }
result = crew.kickoff(inputs=inputs)

Additional Resources: from crewai_tools import DirectoryReadTool, \
FileReadTool, \
SerperDevTool

directory_read_tool = DirectoryReadTool(directory='./instructions') file_read_tool = FileReadTool() search_tool = SerperDevTool()

from crewai_tools import BaseTool

- Every Tool needs to have a `name` and a `description`.
- For simplicity and classroom purposes, `SentimentAnalysisTool` will return `positive` for every text.

e.g.

```
class SentimentAnalysisTool(BaseTool):
    name: str ="Sentiment Analysis Tool"
    description: str = ("Analyzes the sentiment of text "
        "to ensure positive and engaging communication.")

    def _run(self, text: str) -> str:
        # Your custom code tool goes here
        return "positive"

sentiment_analysis_tool = SentimentAnalysisTool()
```

Other Popular Models as LLM for your Agents

Hugging Face (HuggingFaceHub endpoint)

```
from langchain_community.llms import HuggingFaceHub
```

```
llm = HuggingFaceHub(

repo_id="HuggingFaceH4/zephyr-7b-beta",

huggingfacehub_api_token="<HF_TOKEN_HERE>",

task="text-generation",

)




### you will pass "llm" to your agent function
```

Mistral API

```
OPENAI_API_KEY=your-mistral-api-key

OPENAI_API_BASE=https://api.mistral.ai/v1

OPENAI_MODEL_NAME="mistral-small"
```

Cohere

```
from langchain_community.chat_models import ChatCohere

# Initialize language model
```

```
os.environ["COHERE_API_KEY"] = "your-cohere-api-key"


llm = ChatCohere()




### you will pass "llm" to your agent function
```

For using Llama locally with Ollama and more, checkout the crewAI documentation on Connecting to any LLM.Other Popular Models as LLM for your Agents Hugging Face (HuggingFaceHub endpoint) from langchain_community.llms import HuggingFaceHub

llm = HuggingFaceHub(repo_id="HuggingFaceH4/zephyr-7b-beta", huggingfacehub_api_token="<HF_TOKEN_HERE>", task="text-generation",)

you will pass "llm" to your agent function

Mistral API OPENAI_API_KEY=your-mistral-api-key OPENAI_API_BASE=https://api.mistral.ai/v1 OPENAI_MODEL_NAME="mistral-small" Cohere from langchain_community.chat_models import ChatCohere

Initialize language model

os.environ["COHERE_API_KEY"] = "your-cohere-api-key" llm = ChatCohere()

you will pass "llm" to your agent function

For using Llama locally with Ollama and more, checkout the crewAI documentation on Connecting to any LLM.

Use Case 1: Event Planning

Agents

Agent 1: Venue Coordinator

venue_coordinator = Agent(role="Venue Coordinator", goal="Identify and book an appropriate venue " "based on event requirements", tools=[search_tool, scrape_tool], verbose=True, backstory=("With a keen sense of space and " "understanding of event logistics, " "you excel at finding and securing " "the perfect venue that fits the event's theme, " "size, and budget constraints."))

Agent 2: Logistics Manager

logistics_manager = Agent(role='Logistics Manager', goal=("Manage all logistics for the event " "including catering and equipmen"), tools=[search_tool, scrape_tool], verbose=True, backstory=("Organized and detail-oriented, " "you ensure that every logistical aspect of the event " "from catering to equipment setup " "is flawlessly executed to create a seamless experience."))

Agent 3: Marketing and Communications Agent

marketing_communications_agent = Agent(role="Marketing and Communications Agent", goal="Effectively market the event and " "communicate with participants", tools= [search_tool, scrape_tool], verbose=True, backstory=("Creative and communicative, " "you craft compelling messages and " "engage with potential attendees " "to maximize event exposure and participation."))

Create Pydantic Object to convert and structure data into JSON

from pydantic import BaseModel

Define a Pydantic model for venue details (demonstrating Output as Pydantic)

class VenueDetails(BaseModel): name: str address: str capacity: int booking_status: str

Tasks

venue_task = Task(description="Find a venue in {event_city} " "that meets criteria for {event_topic}.", expected_output="All the details of a specifically chosen" "venue you found to accommodate the event.", human_input=True, output_json=VenueDetails, output_file="venue_details.json", # Outputs the venue details as a JSON file agent=venue_coordinator)

```
logistics_task = Task(
    description="Coordinate catering and "
                "equipment for an event "
                "with {expected_participants} participants "
                "on {tentative_date}.",
    expected_output="Confirmation of all logistics arrangements "
                    "including catering and equipment setup.",
    human_input=True,
    async_execution=True,
    agent=logistics_manager
)

marketing_task = Task( description="Promote the {event_topic} " "aiming to engage at least" "{expected_participants} potential attendees.", expected_output="Report on marketing activities " "and attendee engagement formatted as markdown.", async_execution=True, output_file="marketing_report.md",  # Outputs the report as a text file agent=marketing_communications_agent )

Crew

# Define the crew with agents and tasks

event_management_crew = Crew( agents=[venue_coordinator, logistics_manager, marketing_communications_agent],
 tasks=[venue_task,        logistics_task,        marketing_task],
verbose=True
)

event_details = {
    'event_topic': "Tech Innovation Conference",
    'event_description': "A gathering of tech innovators "
                         "and industry leaders "
                         "to explore future technologies.",
    'event_city': "San Francisco",
    'tentative_date': "2024-09-15",
    'expected_participants': 500,
    'budget': 20000,
```

```
    'venue_type': "Conference Hall"
}

#Run the result


result = event_management_crew.kickoff(inputs=event_details)


import json
from pprint import pprint

with open('venue_details.json') as f:
  data = json.load(f)

pprint(data)


from IPython.display import Markdown
Markdown("marketing_report.md")
```

Use Case 2: Financial Analysis

```
 !pip install crewai==0.28.8 crewai_tools==0.1.6
langchain_community==0.0.29
from crewai import Agent, Task, Crew

import os from utils import get_openai_api_key, get_serper_api_key
openai_api_key = get_openai_api_key() os.environ["OPENAI_MODEL_NAME"] = 'gpt-3.5-
turbo' os.environ["SERPER_API_KEY"] = get_serper_api_key()

from crewai_tools import ScrapeWebsiteTool, SerperDevTool

search_tool = SerperDevTool()
scrape_tool = ScrapeWebsiteTool()
```

Agents

data_analyst_agent = Agent(role="Data Analyst", goal="Monitor and analyze market data in real-time " "to identify trends and predict market movements.", backstory="Specializing in financial markets, this agent " "uses statistical modeling and machine learning " "to provide crucial insights. With a knack for data, " "the Data Analyst Agent is the cornerstone for " "informing trading decisions.", verbose=True, allow_delegation=True, tools = [scrape_tool, search_tool])

trading_strategy_agent = Agent(role="Trading Strategy Developer", goal="Develop and test various trading strategies based " "on insights from the Data Analyst Agent.", backstory="Equipped with a deep understanding of financial " "markets and quantitative analysis, this agent " "devises and refines trading strategies. It evaluates " "the performance of different approaches to determine " "the most profitable and risk-averse options.", verbose=True, allow_delegation=True, tools = [scrape_tool, search_tool])

execution_agent = Agent(role="Trade Advisor", goal="Suggest optimal trade execution strategies " "based on approved trading strategies.", backstory="This agent specializes in analyzing the timing, price, " "and logistical details of potential trades. By evaluating " "these factors, it provides well-founded suggestions for " "when and how trades should be executed to maximize " "efficiency and adherence to strategy.", verbose=True, allow_delegation=True, tools = [scrape_tool, search_tool])

risk_management_agent = Agent(role="Risk Advisor", goal="Evaluate and provide insights on the risks " "associated with potential trading activities.", backstory="Armed with a deep understanding of risk assessment models " "and market dynamics, this agent scrutinizes the potential " "risks of proposed trades. It offers a detailed analysis of " "risk exposure and suggests safeguards to ensure that " "trading activities align with the firm’s risk tolerance.", verbose=True, allow_delegation=True, tools = [scrape_tool, search_tool])

Task

Task for Data Analyst Agent: Analyze Market Data

data_analysis_task = Task(description=("Continuously monitor and analyze market data for " "the selected stock ({stock_selection}). " "Use statistical modeling and machine learning to " "identify trends and predict market movements."), expected_output=("Insights and alerts about significant market " "opportunities or threats for {stock_selection}."), agent=data_analyst_agent,)

Task for Trading Strategy Agent: Develop Trading Strategies

strategy_development_task = Task(description=("Develop and refine trading strategies based on " "the insights from the Data Analyst and " "user-defined risk tolerance ({risk_tolerance}). " "Consider trading preferences ({trading_strategy_preference})."), expected_output=("A set of potential trading strategies for {stock_selection} " "that align with the user's risk tolerance."), agent=trading_strategy_agent,)

Task for Trade Advisor Agent: Plan Trade Execution

execution_planning_task = Task(description=("Analyze approved trading strategies to determine the " "best execution methods for {stock_selection}, " "considering current market conditions and optimal pricing."), expected_output=("Detailed execution plans suggesting how and when to " "execute trades for {stock_selection}."), agent=execution_agent,)

Task for Risk Advisor Agent: Assess Trading Risks

risk_assessment_task = Task(description=("Evaluate the risks associated with the proposed trading " "strategies and execution plans for {stock_selection}. " "Provide a detailed analysis of potential risks " "and suggest mitigation strategies."), expected_output=("A comprehensive risk analysis report detailing potential " "risks and mitigation recommendations for {stock_selection}."), agent=risk_management_agent,)

Crew

from crewai import Crew, Process from langchain_openai import ChatOpenAI

Define the crew with agents and tasks

financial_trading_crew = Crew(agents=[data_analyst_agent, trading_strategy_agent, execution_agent, risk_management_agent],
tasks=[data_analysis_task, strategy_development_task, execution_planning_task, risk_assessment_task],
manager_llm=ChatOpenAI(model="gpt-3.5-turbo", temperature=0.7), process=Process.hierarchical, verbose=True
)

Example data for kicking off the process

financial_trading_inputs = { 'stock_selection': 'AAPL', 'initial_capital': '100000', 'risk_tolerance': 'Medium', 'trading_strategy_preference': 'Day Trading', 'news_impact_consideration': True }

this execution will take some time to run

result = financial_trading_crew.kickoff(inputs=financial_trading_inputs)

from IPython.display import Markdown
Markdown(result)

Embedding Models-To Do

https://learn.deeplearning.ai/courses/embedding-models-from-architecture-to-implementation/lesson/vu3si/introduction

First App Launched

- What you built.

I built a stock market dashboard that I hope to build on for more complex functionality. The mvp is just a dashboard that shows stocks based on the user entry. Plan to eventually build

the app to enable a auto trade application. I used Streamlit, Yfinance, Matplotlib, Plotly to build the app,
After several attempts that included deprecated tool versions and exploring different ideas I landed on this as it enabled me to reach the goal of creating my first app without too much complicated functionality limiting my ability to complete this first app.
Cursor was very helpful as I continued to struggle with launching this first app.

Github: https://github.com/toyeade/Stock-Market-Dashboard

Why ChatGPT will become new Distribution(10x)

🔗 https://www.youtube.com/watch?v=cX4cL6B-_aU&t=329s

Distribution Shift

Technology Platform Shifts Cycle

1. Conditions of the market have been met
2. Creating a moat around platform
3. Platform opening
4. Platform closing for monetization

What phase is ChatGPT, Claude, Gemini in this platform shift?

What would they do if they are focusing on monopoly?

What would they do if they are focusing on a specific customer segment e.g. consumer, enterprise, developer communities?

Moat is context and memory; differentiator is which platform has the best context for the application use cases?

Retention is increasing in AI Platforms "smile effect" curve

- Retention and user depth vs MAU+ Signups
- what is the value exchange i.e. what are they giving you in order to incentivize you on the platform
- Scale + Scalability
- Consider exit strategy; how are you going to own end user experience? how are you going to leverage specialized data and context? How will you create and leverage network effects?

ChatGPT currently has 10x in users than Claude and other platforms.

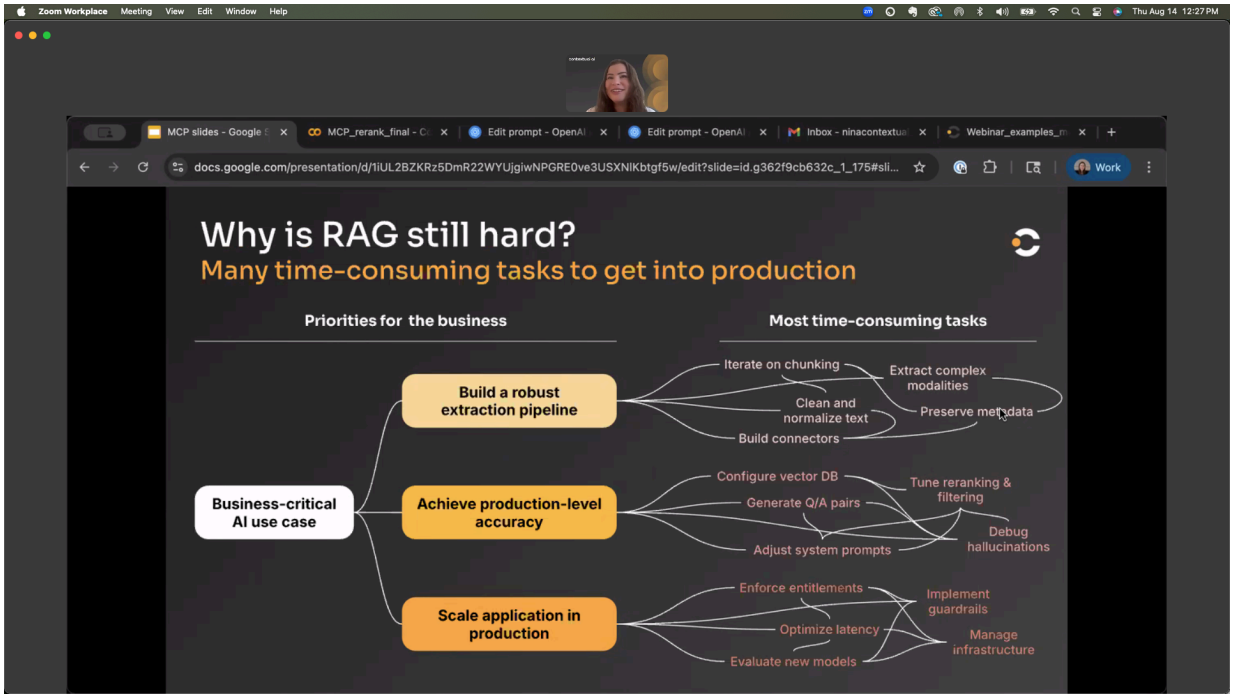
'AI Agent' pricing is attributable and autonomous

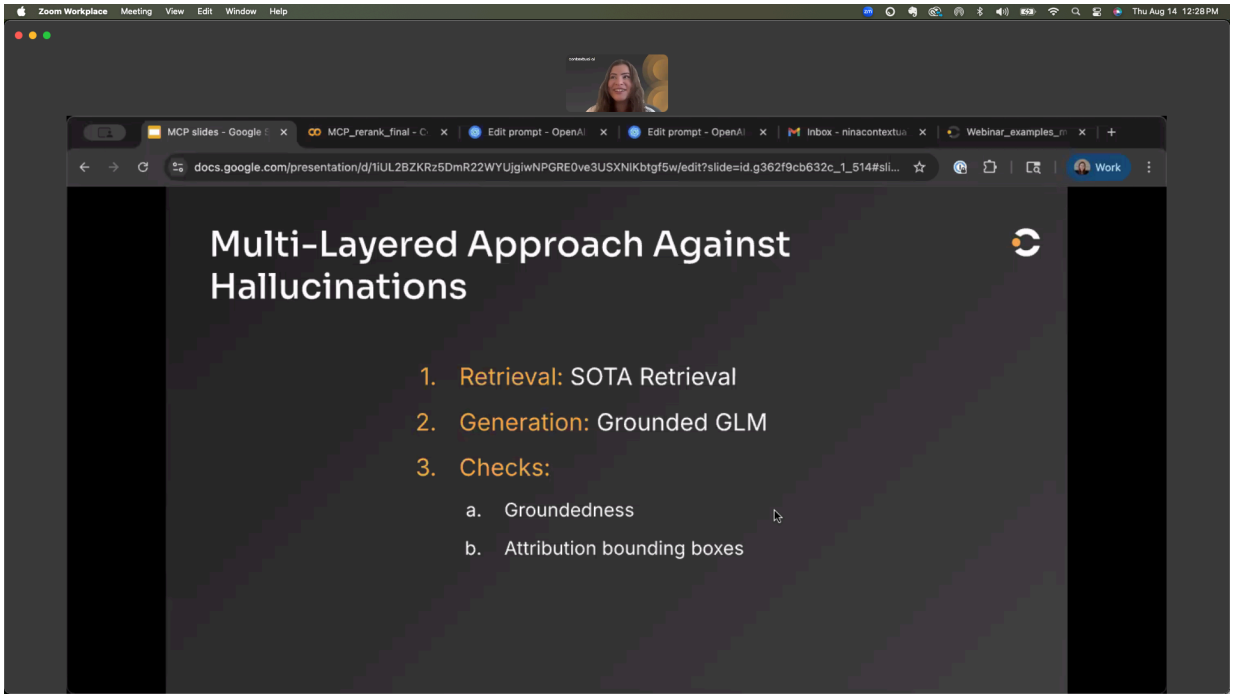
How are platforms incentivizing to get users and developers on the platform?

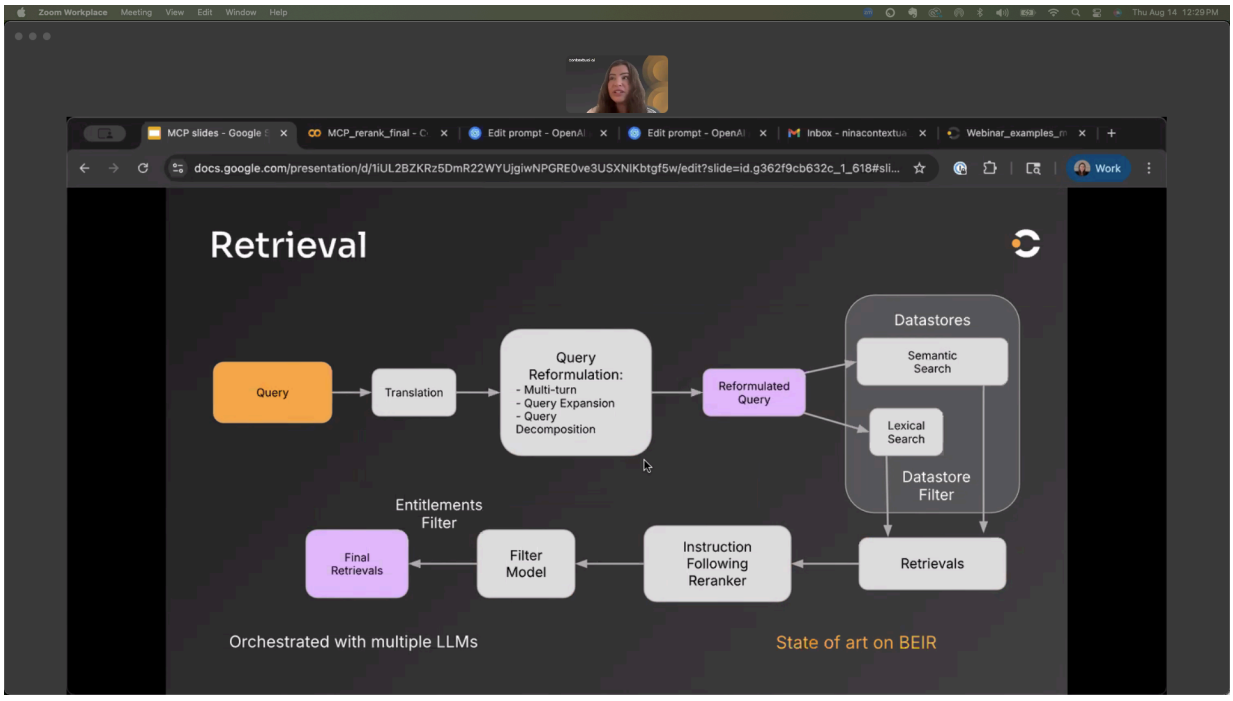
Pricing: subscriptions, outcome based, usage based
AI Agents pricing: subscriptions, outcome based, usage based

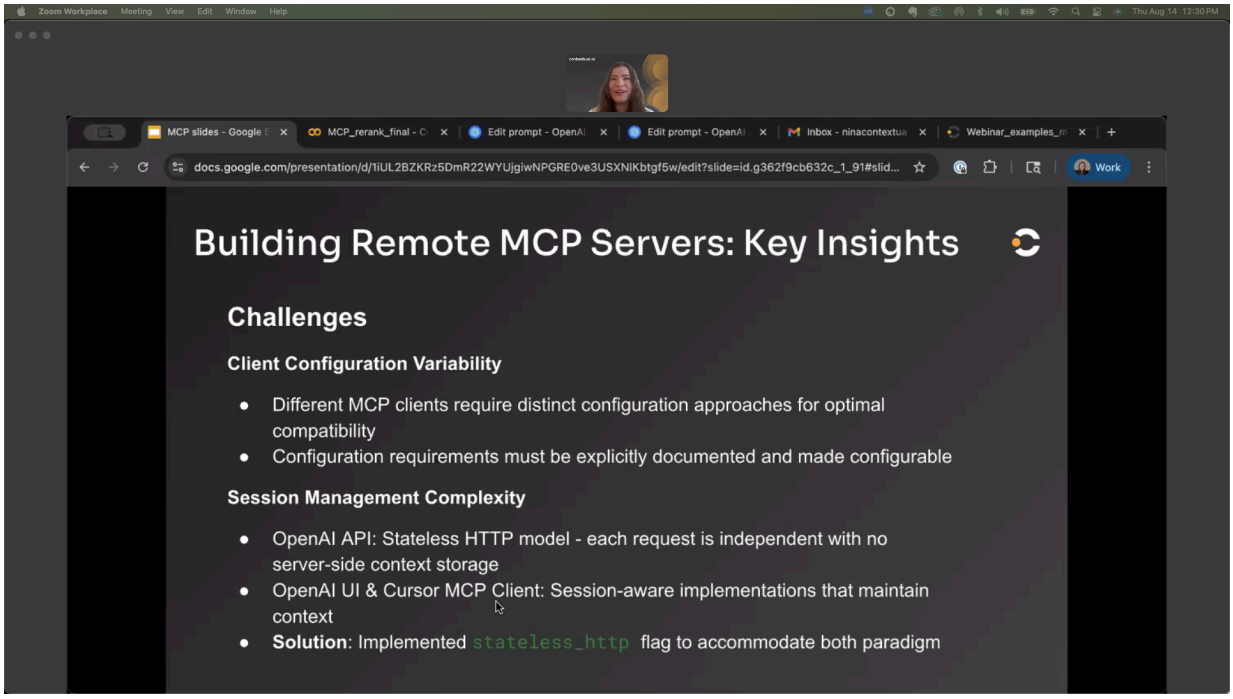
Smart Tools for Agentic AI

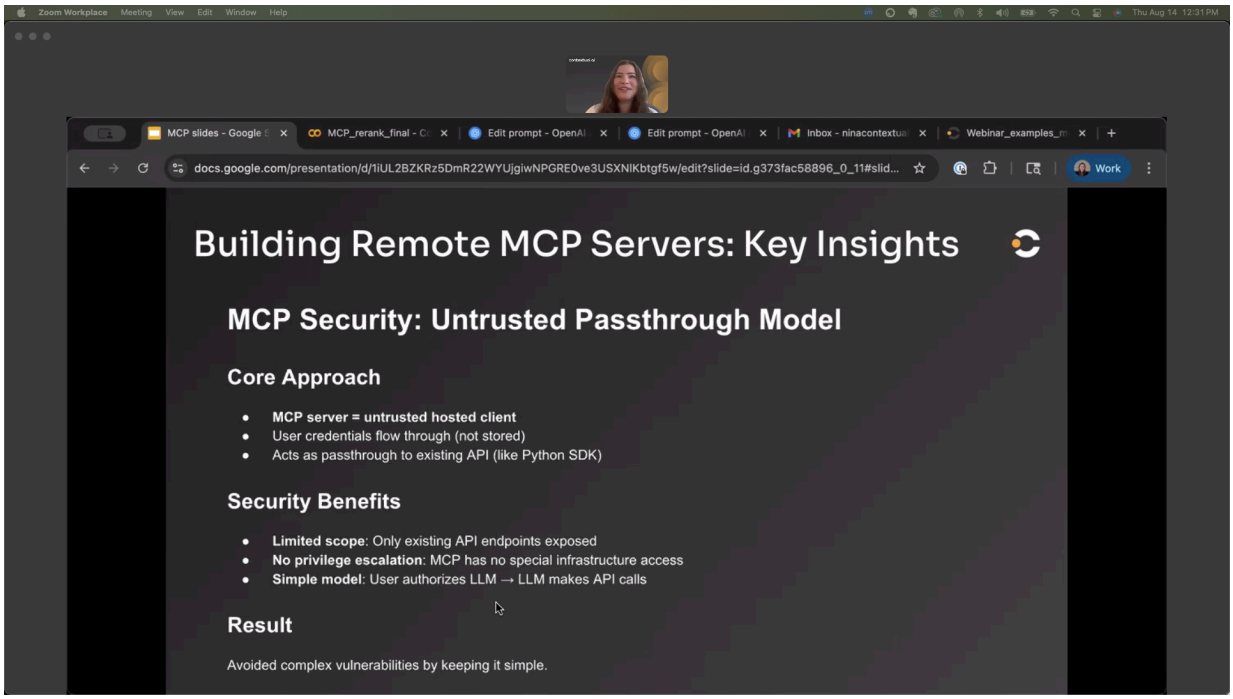
Held by Contextual.ai

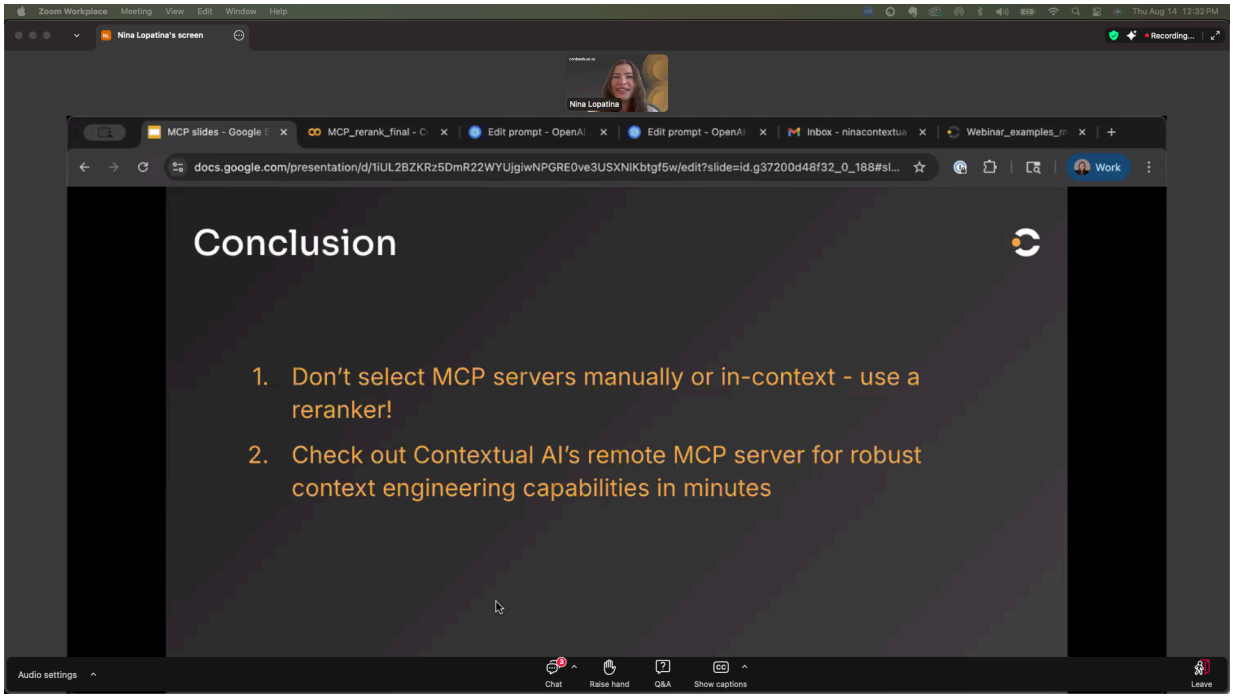

Zoom Workplace Meeting View Edit Window Help
Thu Aug 14 12:27 PM
MCP slides - Google
MCP_rerank_final -
Edit prompt - OpenAI
Edit prompt - OpenAI
Inbox - ninacontextual
Webinar_examples_
docs.google.com/presentation/d/1iUL2BZKRz5DmR22WYUjgiwNPGRE0ve3USXNlKbtgf5w/edit?slide=id.g362f9cb632c_1_175#sli...
Work
Why is RAG still hard?
Many time-consuming tasks to get into production
Priorities for the business
Most time-consuming tasks
Business-critical AI use case
Build a robust extraction pipeline
Achieve production-level accuracy
Scale application in production
Iterate on chunking
Extract complex modalities
Clean and normalize text
Preserve metadata
Build connectors
Configure vector DB
Tune reranking & filtering
Generate Q/A pairs
Debug hallucinations
Adjust system prompts
Enforce entitlements
Implement guardrails
Optimize latency
Manage infrastructure
Evaluate new models

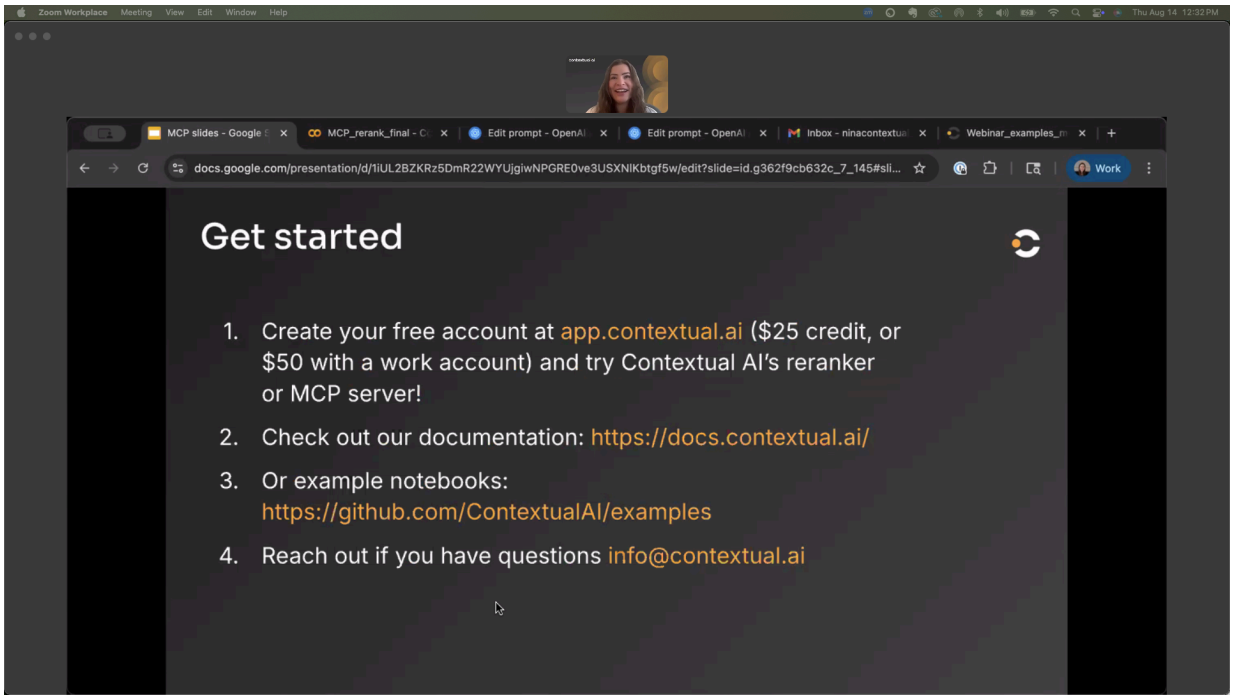

Multi-Layered Approach Against Hallucinations
1. Retrieval: SOTA Retrieval
2. Generation: Grounded GLM
3. Checks:
a. Groundedness
b. Attribution bounding boxes


Retrieval
Query
Translation
Query Reformulation:
- Multi-turn
- Query Expansion
- Query Decomposition
Reformulated Query
Datastores
Semantic Search
Lexical Search
Datastore Filter
Retrievals
Instruction Following Reranker
Filter Model
Entitlements Filter
Final Retrievals
Orchestrated with multiple LLMs
State of art on BEIR


Zoom Workplace Meeting View Edit Window Help
Thu Aug 14 12:30 PM
MCP slides - Google
MCP_rerank_final -
Edit prompt - OpenAI
Edit prompt - OpenAI
Inbox - ninacontextua
Webinar_examples_m
docs.google.com/presentation/d/1iUL2BZKRz5DmR22WYUjgiwNPGRE0ve3USXNlKbtgf5w/edit?slide=id.g362f9cb632c_1_91#slid...
Work
Building Remote MCP Servers: Key Insights
Challenges
Client Configuration Variability
Different MCP clients require distinct configuration approaches for optimal compatibility
Configuration requirements must be explicitly documented and made configurable
Session Management Complexity
OpenAI API: Stateless HTTP model - each request is independent with no server-side context storage
OpenAI UI & Cursor MCP Client: Session-aware implementations that maintain context
Solution: Implemented stateless_http flag to accommodate both paradigm


Zoom Workplace Meeting View Edit Window Help
Thu Aug 14 12:31 PM
MCP slides - Google
MCP_rerank_final -
Edit prompt - OpenAI
Edit prompt - OpenAI
Inbox - ninacontextua
Webinar_examples_m
docs.google.com/presentation/d/1iUL2BZKRz5DmR22WYUjgiwNPGRE0ve3USXNlKbtgf5w/edit?slide=id.g373fac58896_0_11#slid...
Work
Building Remote MCP Servers: Key Insights
MCP Security: Untrusted Passthrough Model
Core Approach
MCP server = untrusted hosted client
User credentials flow through (not stored)
Acts as passthrough to existing API (like Python SDK)
Security Benefits
Limited scope: Only existing API endpoints exposed
No privilege escalation: MCP has no special infrastructure access
Simple model: User authorizes LLM → LLM makes API calls
Result
Avoided complex vulnerabilities by keeping it simple.


Nina Lopatina's screen
Recording...
Nina Lopatina
MCP slides - Google
MCP_rerank_final - C
Edit prompt - OpenAI
Edit prompt - OpenAI
Inbox - ninacontextua
Webinar_examples_m
docs.google.com/presentation/d/1iUL2BZKRz5DmR22WYUjgiwNPGRE0ve3USXNlKbtgf5w/edit?slide=id.g37200d48f32_0_188#sl...
Work
Conclusion
1. Don't select MCP servers manually or in-context - use a reranker!
2. Check out Contextual AI's remote MCP server for robust context engineering capabilities in minutes
Audio settings
Chat
Raise hand
Q&A
Show captions
Leave


MCP slides - Google
MCP_rerank_final - C
Edit prompt - OpenAI
Edit prompt - OpenAI
Inbox - ninacontextua
Webinar_examples_m
docs.google.com/presentation/d/1iUL2BZKRz5DmR22WYUjgiwNPGRE0ve3USXNlKbtgf5w/edit?slide=id.g362f9cb632c_7_145#sli...
Work
Get started
1. Create your free account at app.contextual.ai ($25 credit, or $50 with a work account) and try Contextual AI's reranker or MCP server!
2. Check out our documentation: https://docs.contextual.ai/
3. Or example notebooks: https://github.com/ContextualAI/examples
4. Reach out if you have questions info@contextual.ai