ILLVZN Final Pitch

2025

HackRx 6.0

~ Avinash Anish & Nihaal SP (RVCE)

BAJAJ FINSERV

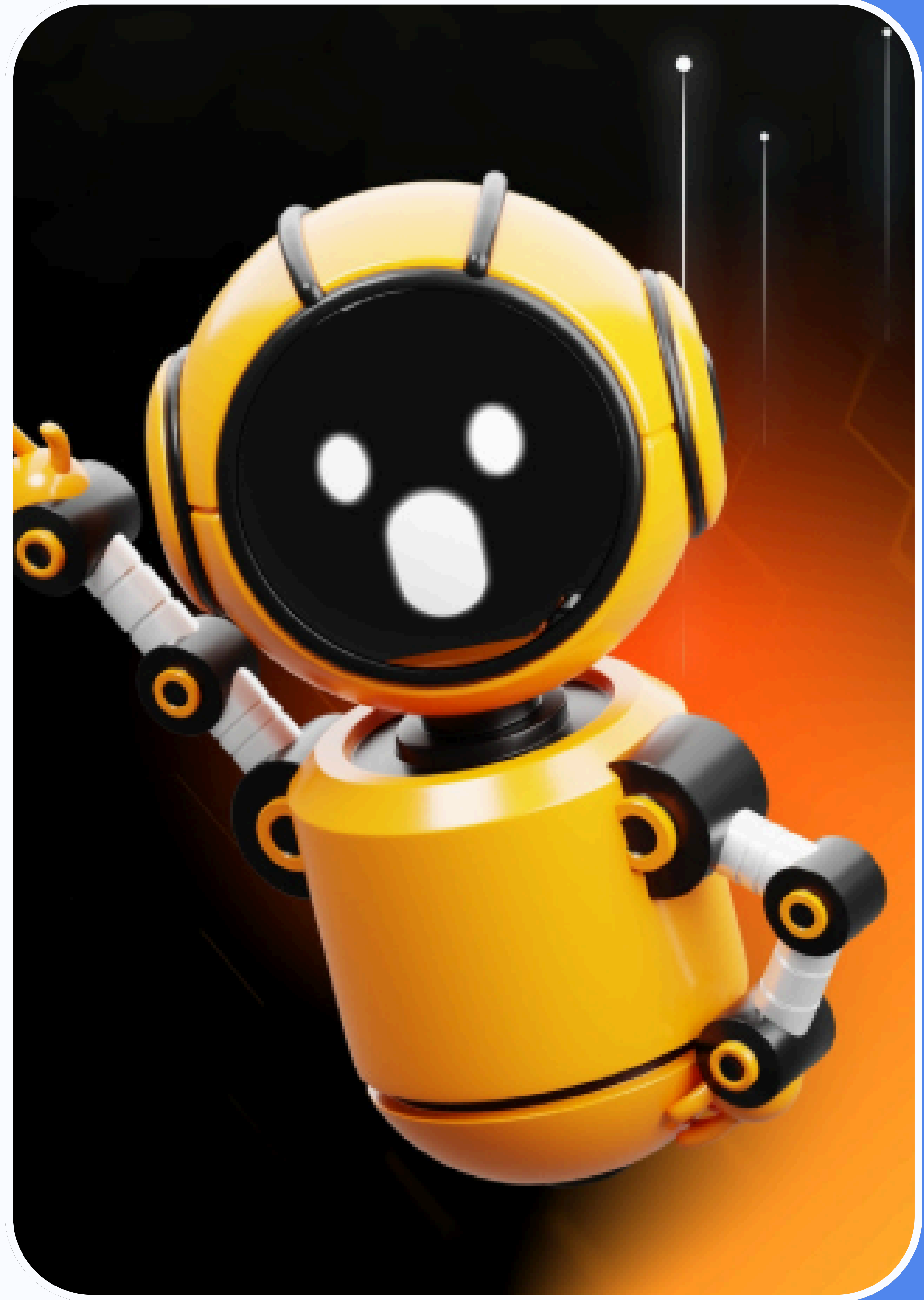
Problem Statement

The Real Challenge

Build a system that uses Large Language Models (LLMs) to process natural language queries and retrieve relevant information from large unstructured documents

The Hidden Crisis

- Information Asymmetry
- Context Collapse
- Human Bottlenecks
- Trust Deficit



Round 1, RAG?

- Entered hackathon with the idea that It would be a very fun problem statement to solve, and boy it was.
- Limited understanding of retrieval, chunking, and embeddings didn’t help initially.
- Used Pinecone for vector storage → faced latency & bottlenecks.
- Relied on a local embedding model with poor accuracy/quality.
- Outcome: Only 1 low scoring performance in Round 1 which almost buried us
- R1 accuracy: 4% and a rank of 323 :(

Round 2, Document Bombardment

- Now having a good idea about how things worked, both we and the number of documents entered into a higher gear.
- Switching to inmemory vector DB, Openai embedding models, and implementing async parallelisation really worked in our favour
- We topped the leader board in accuracy, and were 5th score wise and that's when we realised our standards and scope of HackRx
- At this point, we also started logging the questions and responses onto Supabase for debugging any issues and made testing scripts which can hit our endpoint in the same way.
- R2 accuracy: 66% and 1st rank(by accuracy), 5th(by score)

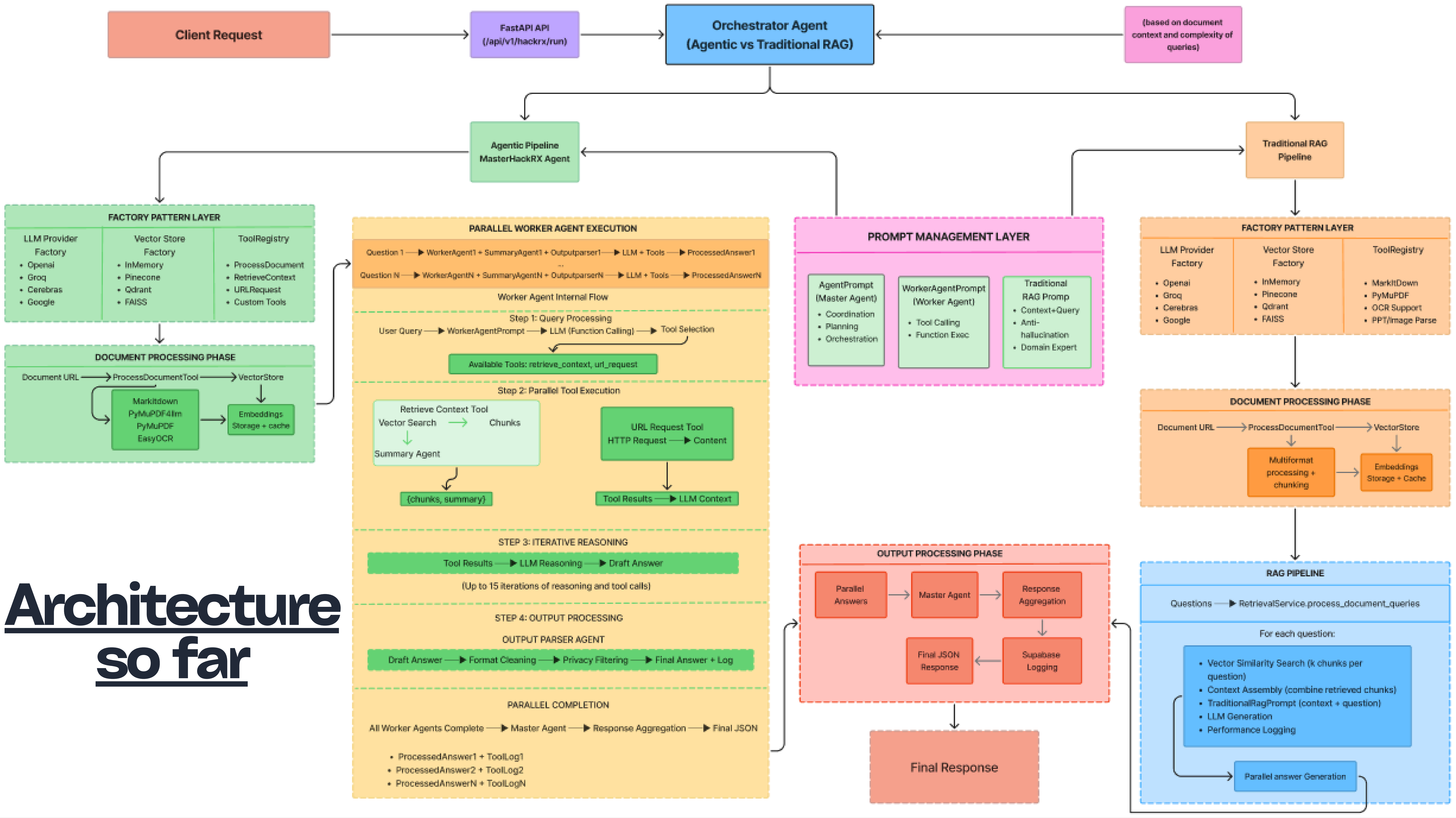
Round 3, the Rain continues

- Round 2 had some huge documents which we were just able to abide by, but then Round 3 added onto that with all kinds of documents like PPTs, Images, binary dumps, etc
- Added factory patterns for LLM providers, vector DBs, better document processing libraries and any small optimisations we could think of
- All our scores were close, and we probably should have thought about using better opensource embedding models which we now realise perform much better than the Openai embeddings.
- R3 accuracy: 75%

Round 4, the Enigma

- Until round 3 all queries were based off of documents that our Traditional RAG system could solve, but the queries of Round 4 were on a different level.
- We then realised our naivety and the small reach of our architecture and decided we needed to improve as R4 needed agentic tool calls.
- Then things fell in correctly, and we added an agentic pipeline along with the traditional RAG system with a master orchestrator to make the choice for the pipeline to be used based on the query and its complexity
- A common prompt layer provided prompts for both the pipelines to encourage Generalism, and the agentic pipeline had worker agents performing the tool calls
- With agentic comes latency which was our bane this round and we almost didn't make it to the final showdown.
- R4 accuracy: 84.31%

Architecture so far


Client Request
FastAPI API (/api/v1/hackrx/run)
Orchestrator Agent (Agentic vs Traditional RAG)
(based on document context and complexity of queries)
Agentic Pipeline MasterHackRX Agent
Traditional RAG Pipeline
FACTORY PATTERN LAYER
LLM Provider Factory
• Openai • Groq • Cerebras • Google
Vector Store Factory
• InMemory • Pinecone • Qdrant • FAISS
ToolRegistry
• ProcessDocument • RetrieveContext • URLRequest • Custom Tools
DOCUMENT PROCESSING PHASE
Document URL → ProcessDocumentTool → VectorStore
Markitdown PyMuPDF4llm PyMuPDF EasyOCR
Embeddings Storage + cache
PARALLEL WORKER AGENT EXECUTION
Question 1 → WorkerAgent1 + SummaryAgent1 + Outputparser1 → LLM + Tools → ProcessedAnswer1
...
Question N → WorkerAgentN + SummaryAgentN + OutputparserN → LLM + Tools → ProcessedAnswerN
Worker Agent Internal Flow
Step 1: Query Processing
User Query → WorkerAgentPrompt → LLM (Function Calling) → Tool Selection
Available Tools: retrieve_context, url_request
Step 2: Parallel Tool Execution
Retrieve Context Tool
Vector Search → Chunks
Summary Agent
{chunks, summary}
URL Request Tool
HTTP Request → Content
Tool Results → LLM Context
STEP 3: ITERATIVE REASONING
Tool Results → LLM Reasoning → Draft Answer
(Up to 15 iterations of reasoning and tool calls)
STEP 4: OUTPUT PROCESSING
OUTPUT PARSER AGENT
Draft Answer → Format Cleaning → Privacy Filtering → Final Answer + Log
PARALLEL COMPLETION
All Worker Agents Complete → Master Agent → Response Aggregation → Final JSON
• ProcessedAnswer1 + ToolLog1
• ProcessedAnswer2 + ToolLog2
• ProcessedAnswerN + ToolLogN
PROMPT MANAGEMENT LAYER
AgentPrompt (Master Agent)
• Coordination • Planning • Orchestration
WorkerAgentPrompt (Worker Agent)
• Tool Calling • Function Exec
Traditional RAG Promp
• Context+Query • Anti-hallucination • Domain Expert
FACTORY PATTERN LAYER
LLM Provider Factory
• Openai • Groq • Cerebras • Google
Vector Store Factory
• InMemory • Pinecone • Qdrant • FAISS
ToolRegistry
• MarkItDown • PyMuPDF • OCR Support • PPT/Image Parse
DOCUMENT PROCESSING PHASE
Document URL → ProcessDocumentTool → VectorStore
Multiformat processing + chunking
Embeddings Storage + Cache
OUTPUT PROCESSING PHASE
Parallel Answers → Master Agent → Response Aggregation
Supabase Logging → Final JSON Response
Final Response
RAG PIPELINE
Questions → RetrievalService.process_document_queries
For each question:
• Vector Similarity Search (k chunks per question)
• Context Assembly (combine retrieved chunks)
• TraditionalRagPrompt (context + question)
• LLM Generation
• Performance Logging
Parallel answer Generation

Tech Stack

JS TS Tailwind CSS React

NEXT.JS FastAPI supabase

Gemini OpenAI LangChain

Round 5, Waste of potential

- Round 5 kicked off with a token based game where we had to find the hidden text in the source code.
- Since the code was client side, we generated valid JWTs which opened 3 more games to us which were actually meant to be R7 questions lol
- Instead of submitting, we kept testing to no avail as the server was decoding our token, due to which the game couldn't be played and we wasted hours
- When we made our first submission, it immediately succeeded which left us in mixed emotions of regret for wasting time but also to strive even harder and not make such rookie mistakes again
- Initialised Playwright MCP for all browser based actions
- Achieved blazing fast speeds but were moved onto the penultimate round by a margin of 0.42
- R5 accuracy: 85%

BAJAJ FINSERV

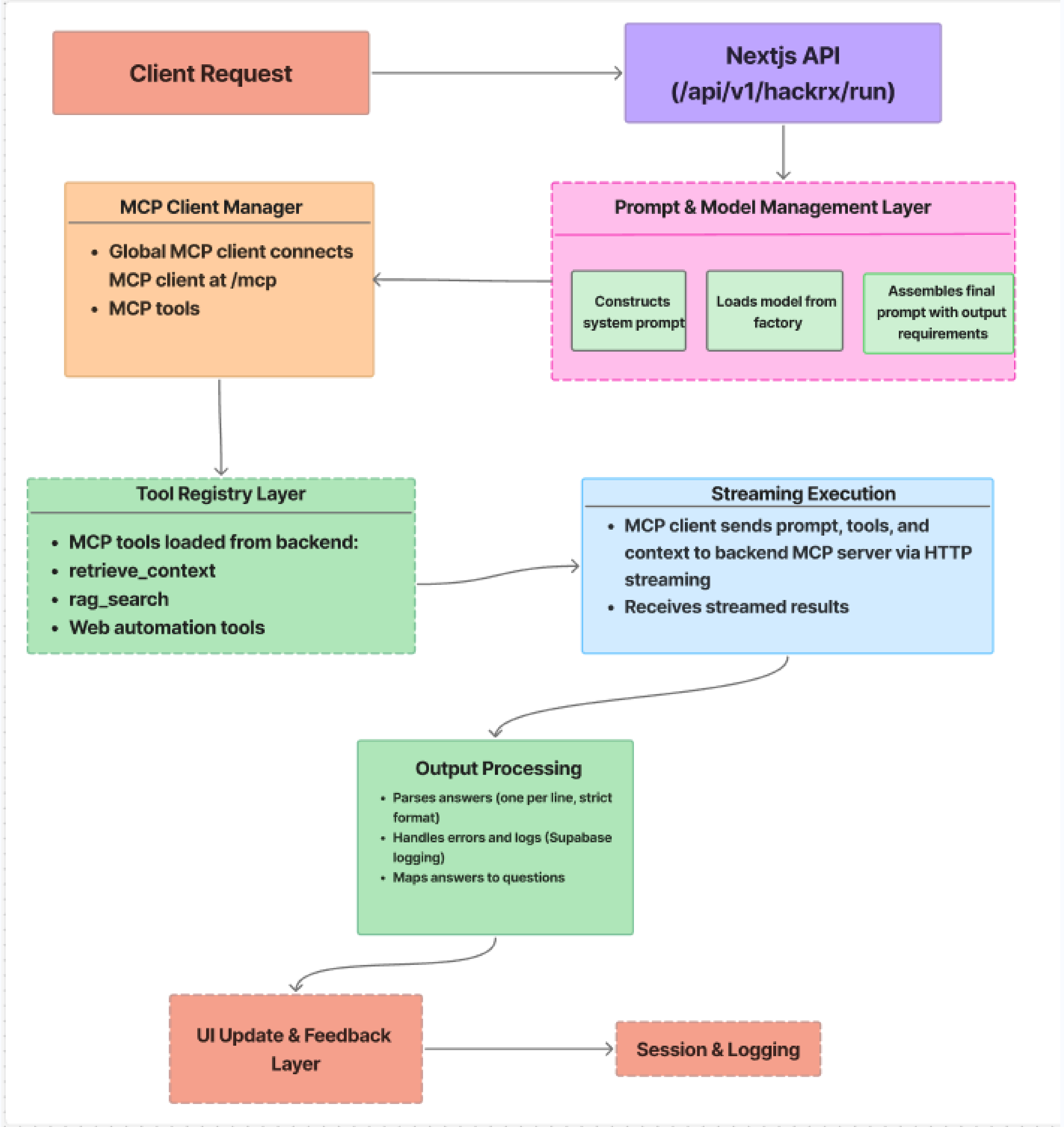
Round 6, Agentic Dev

- Wrote more MCP tools, like a code execution and a Github commit and push tools.
- The agent had to solve a question, run it with test cases, return the answer and push it to our private HackRx repos.
- As usual everything can't go as planned, and we forget to verify the response format, and hadn't catered for the URL field(it was an empty string) and were hit with a score of 0 after 0 due to the Azure content policy warnings.
- After several failed attempts, it clicked to us at 3am and we struck gold with a successful submission and off to round 7 we went.
- R6 accuracy: 95%

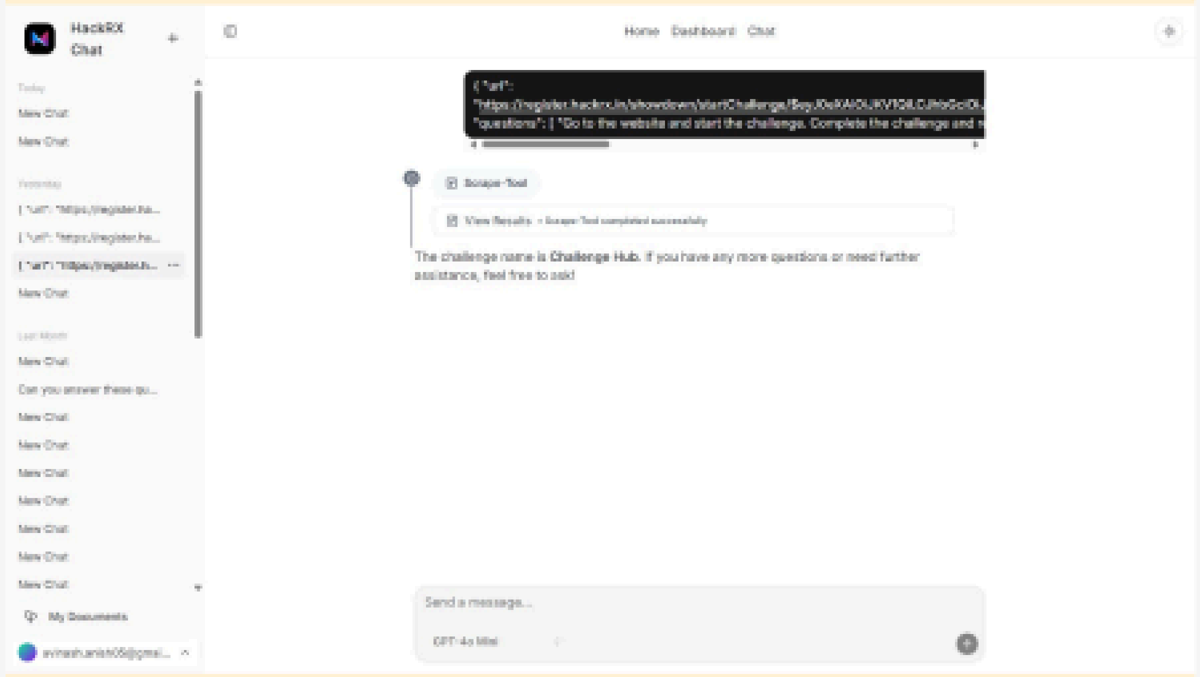
Round 7, Final Push

- Despite knowing the questions from the start, we simply couldn't solve the Simon says pattern matching question, but solved the Github repository analysis question.
- Probably could've written a tool to check the CSS mutations to solve it but felt it would be too specific and would go against the rules of the hackathon to make everything generalised. Thus, we finished 10th on the leader board.
- The mentorship was phenomenal, and no matter the result ultimately at the end of the day we learnt a lot, and will definitely build some agentic pipelines with MCP integrations in the future.
- R7 accuracy: 79.17%

Architecture at the end of Round7


Client Request
Nextjs API (/api/v1/hackrx/run)
Prompt & Model Management Layer
Constructs system prompt
Loads model from factory
Assembles final prompt with output requirements
MCP Client Manager
• Global MCP client connects MCP client at /mcp
• MCP tools
Tool Registry Layer
• MCP tools loaded from backend:
• retrieve_context
• rag_search
• Web automation tools
Streaming Execution
• MCP client sends prompt, tools, and context to backend MCP server via HTTP streaming
• Receives streamed results
Output Processing
• Parses answers (one per line, strict format)
• Handles errors and logs (Supabase logging)
• Maps answers to questions
UI Update & Feedback Layer
Session & Logging

Tech Stack- Frontend


HackRX Chat

Home Dashboard Chat

Today
New Chat
New Chat

Yesterday
{ "url": "https://register.ha...
{ "url": "https://register.ha...
{ "url": "https://register.h... ···
New Chat

Last Month
New Chat
Can you answer these qu...
New Chat
New Chat
New Chat
New Chat
New Chat
New Chat
New Chat

My Documents
avinash.anish05@gmai...

```
{ "url":
"https://register.hackrx.in/showdown/startChallenge/[illegible]
"questions": [ "Go to the website and start the challenge. Complete the challenge and
```

Scrape-Tool

View Results · Scrape-Tool completed successfully

The challenge name is **Challenge Hub.** If you have any more questions or need further assistance, feel free to ask!

Send a message...

GPT-4o Mini

TypeScript
Tailwind CSS
React
NEXT.JS

Database

supabase

Database

Tech Stack- Backend

FastAPI
PyTorch
LangChain
OpenAI

Deployment

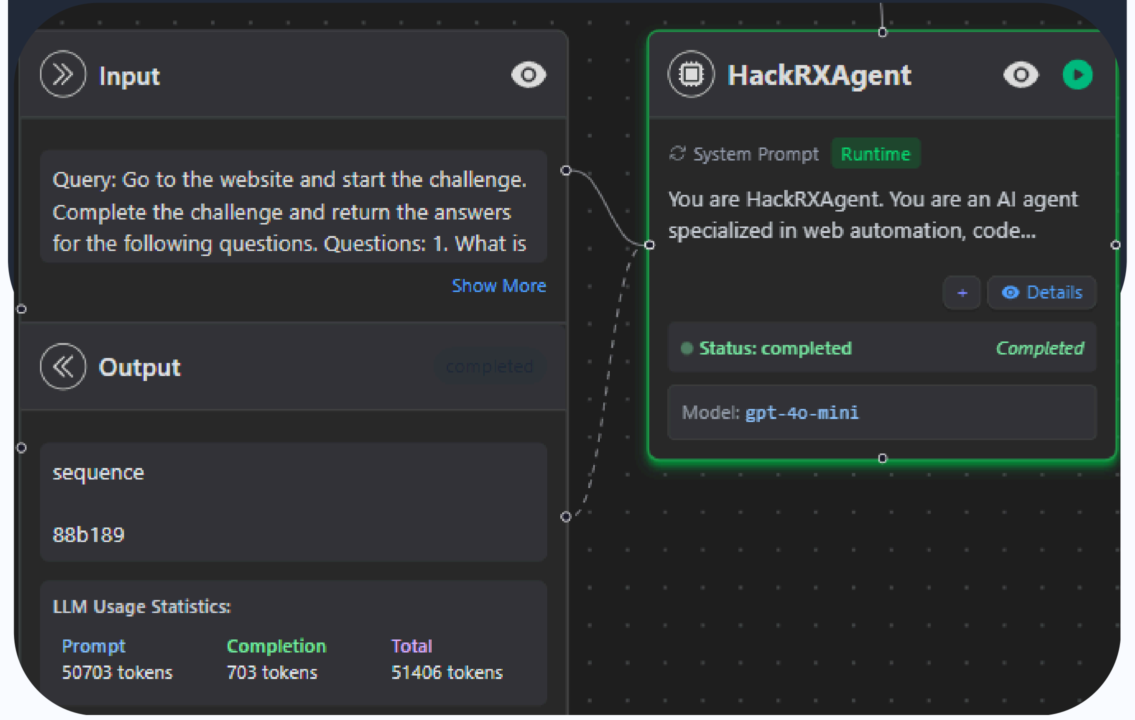
ngrok
Vertex AI
Render

Technology Stack

Some Metrics

Average tokens per Agentic Task - GPT 4.1-mini/4o-mini

Round 5 - 50k in, 703 out
Average Time: 28 s


Input

Query: Go to the website and start the challenge. Complete the challenge and return the answers for the following questions. Questions: 1. What is

Show More

Output completed

sequence

88b189

LLM Usage Statistics:

Prompt 50703 tokens · Completion 703 tokens · Total 51406 tokens

HackRXAgent

System Prompt · Runtime

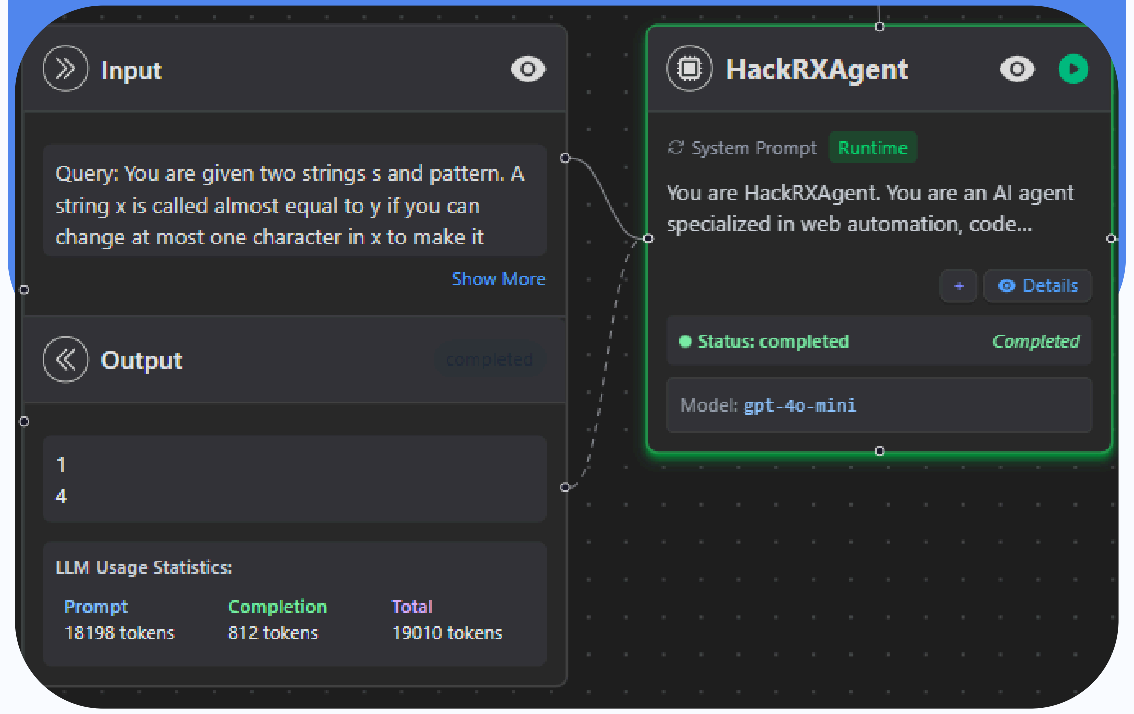
You are HackRXAgent. You are an AI agent specialized in web automation, code...

+ · Details

Status: completed · Completed

Model: gpt-4o-mini

Round 6 - 18k in, 812 out
Average Time: 8 s


Input

Query: You are given two strings s and pattern. A string x is called almost equal to y if you can change at most one character in x to make it

Show More

Output completed

1
4

LLM Usage Statistics:

Prompt 18198 tokens · Completion 812 tokens · Total 19010 tokens

HackRXAgent

System Prompt · Runtime

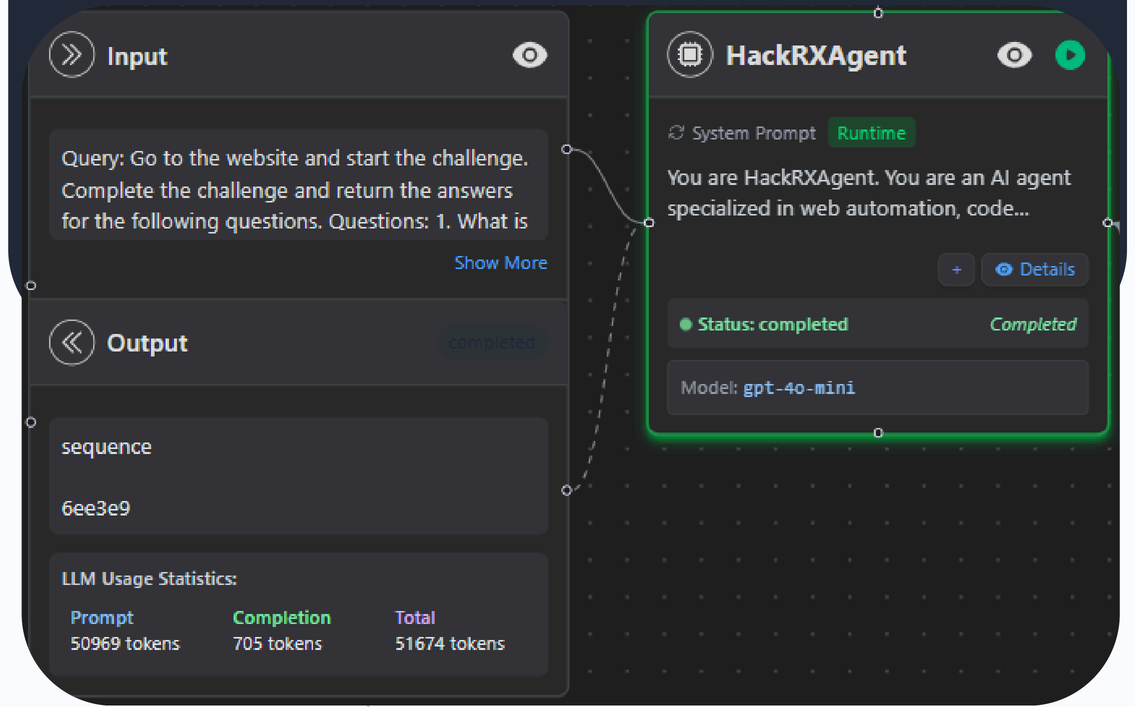
You are HackRXAgent. You are an AI agent specialized in web automation, code...

+ · Details

Status: completed · Completed

Model: gpt-4o-mini

Round 7 - 50k in, 705 out
Average Time: 30s


Input

Query: Go to the website and start the challenge. Complete the challenge and return the answers for the following questions. Questions: 1. What is

Show More

Output completed

sequence

6ee3e9

LLM Usage Statistics:

Prompt 50969 tokens · Completion 705 tokens · Total 51674 tokens

HackRXAgent

System Prompt · Runtime

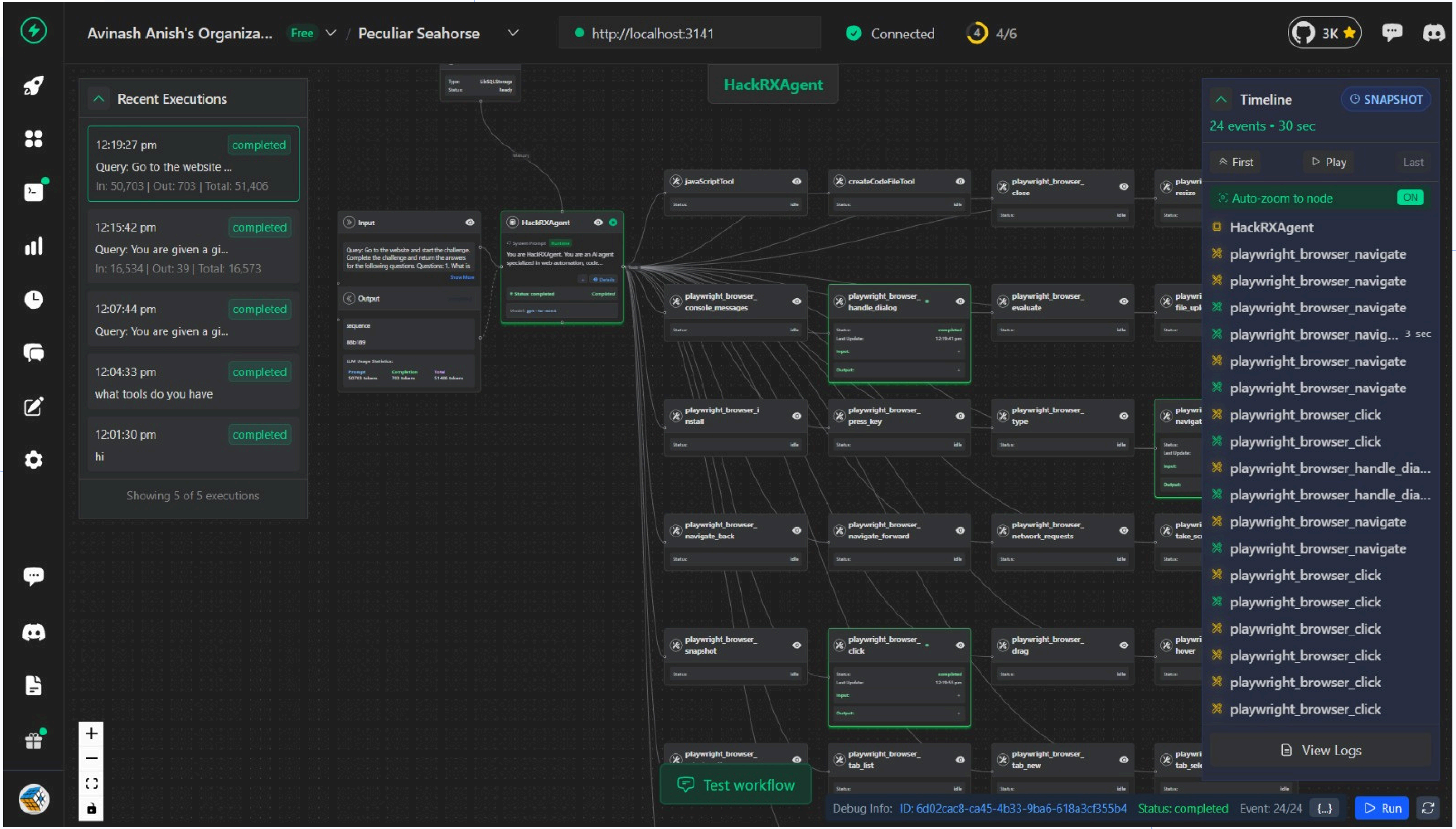
You are HackRXAgent. You are an AI agent specialized in web automation, code...

+ · Details

Status: completed · Completed

Model: gpt-4o-mini

Observability - VoltAgent



Avinash Anish's Organiza... Free / Peculiar Seahorse
http://localhost:3141
Connected
4/6
3K
HackRXAgent
Input
HackRXAgent
Output
javaScriptTool
createCodeFileTool
playwright_browser_close
playwright_browser_console_messages
playwright_browser_handle_dialog
playwright_browser_evaluate
playwright_browser_install
playwright_browser_press_key
playwright_browser_type
playwright_browser_navigate_back
playwright_browser_navigate_forward
playwright_browser_network_requests
playwright_browser_snapshot
playwright_browser_click
playwright_browser_drag
playwright_browser_tab_list
playwright_browser_tab_new
Test workflow
SNAPSHOT
First
Play
Last
Auto-zoom to node
ON
View Logs
Run

Recent Executions

12:19:27 pm	completed
Query: Go to the website ...	
In: 50,703 | Out: 703 | Total: 51,406	
12:15:42 pm	completed
Query: You are given a gi...	
In: 16,534 | Out: 39 | Total: 16,573	
12:07:44 pm	completed
Query: You are given a gi...	
12:04:33 pm	completed
what tools do you have	
12:01:30 pm	completed
hi	

Showing 5 of 5 executions

Query: Go to the website and start the challenge. Complete the challenge and return the answers for the following questions. Questions: 1. What is

You are HackRXAgent. You are an AI agent specialized in web automation, code...

sequence 88b189

Timeline

24 events • 30 sec

- HackRXAgent
- playwright_browser_navigate
- playwright_browser_navigate
- playwright_browser_navigate
- playwright_browser_navig... 3 sec
- playwright_browser_navigate
- playwright_browser_navigate
- playwright_browser_click
- playwright_browser_click
- playwright_browser_handle_dia...
- playwright_browser_handle_dia...
- playwright_browser_navigate
- playwright_browser_navigate
- playwright_browser_click
- playwright_browser_click
- playwright_browser_click
- playwright_browser_click
- playwright_browser_click
- playwright_browser_click

Debug Info: ID: 6d02cac8-ca45-4b33-9ba6-618a3cf355b4 Status: completed Event: 24/24

Our USP

How we stand out

Multi Agent System

The combination of orchestrator, query enhancers, output parsers and worker agents makes out system fully autonomous due to the

MCP Server Integration

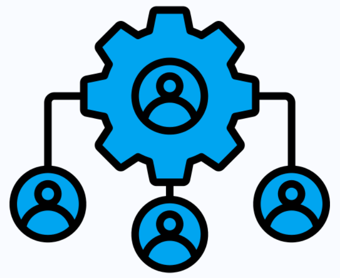
We integrate MCP Servers from various providers along with custom tools, enabling our agentic system to perform complex tasks with tool calling

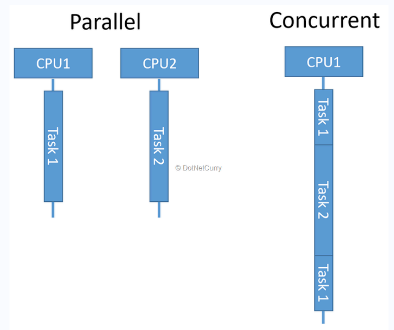
Observability

Integration of theTypeScript Multi Agent framework - Volt Agent, allows us to have unmatched observability, with request level logging and orechestration visualisations

Parallel Execution

Parallel execution of user queries ensures that our system provides answers to the user's queries in less than 20 seconds on average without streaming





Scalability Aspects

1. Horizontal Architecture Scalability
   - Container-Ready Deployment
   - Stateless API Design
2. Vector Store & Data Scalability
   - Multi-Factory VectorDB Support
   - Intelligent Caching Strategy
3. AI Model & Provider Scalability
   - Multi-Provider Architecture
   - Provider Factory Pattern
4. Auto-Scaling & Load Management
   - Kubernetes Ready
   - Health Check Endpoints
   - Graceful shutdown

Thank You