Agentic AI Hackathon

Problem Statements — Participant Edition

Four tracks. One goal: build agents that think.

General Rules & Expectations

- All problem statements are self-contained. You do not need any files from the organizers to get started.
- You define your own inputs, generate your own test data, and bring your own creativity. The core requirements are intentionally minimal — extend them, remix them, and make them yours.
- You are free to use any framework, language, or tooling you choose. What matters is that your system genuinely thinks — not just retrieves and reformats.
- A few expectations across all tracks:
- **Build your own architecture.** Do not wire together pre-built agents and call it done. Your orchestration logic — how agents are created, how they hand off, how they resolve conflict — should reflect deliberate design decisions.
- **Document your thinking.** Every submission must include an architecture document covering your system design, agent roles, tech stack, and the key decisions you made and why. This is not optional.
- **Full stack effort counts.** Judges will evaluate your frontend and backend as part of the submission. A working, usable interface is expected — not a demo script.
- **Creativity is rewarded.** Teams that go beyond the brief — in how they interpret the problem, what they build, or how they present it — will be recognised. Following the problem statement to the letter is the floor, not the ceiling.
- **Your inputs are your responsibility.** Each track requires you to define and provide your own test inputs. Include them in your submission. Judges will use them to reproduce your results.

Track 1 — Secure Code Review

Theme: Building Autonomous Security Researchers

1. The Challenge

In modern CI/CD pipelines, security is often the bottleneck. Most automated tools can catch obvious mistakes, but they fail at context-aware reasoning — understanding how a small logic flaw in one file can be exploited through a completely different part of the codebase.

Your goal is to build an autonomous multi-agent system that acts as a Senior Security Engineer. Starting from a codebase of your choice, it must discover, verify, and patch vulnerabilities — without any human guidance.

2. Input

You choose your own codebase. Some good options:

- A deliberately vulnerable app
- A public GitHub repo that you suspect has real vulnerabilities

3. Output

A Vulnerability report, a PoE script that proves the vulnerability, the proposed fixes and a log showing the agents' internal reasoning and tool call abilities.

Track 2 — Enterprise Intelligence Engine

Theme: Beyond the Search Bar — Building Autonomous Strategic SDRs

1. The Challenge

In B2B sales, finding a high-value client is not about search volume — it is about strategic context: understanding a company's recent pivot, their technical debt, and their fiscal pressures.

Your goal is to build an autonomous multi-agent business intelligence system. It takes a raw company domain, researches the company without any human guidance, and produces a verdict on whether that company is a strong lead — along with a personalized outreach strategy.

2. Input

You define both inputs yourself:

- A Company Domain — pick any real company (e.g., acme-robotics.com)
- A DataVex Service Catalog — you define what DataVex offers, or check our website https://datavex.ai for inspiration

3. Output

A company dossier, a justified verdict on whether to pursue the lead and why now, a draft outreach strategy for a specific decision-maker, and a trace of the agent's research journey.

Track 3 — Technical Interview Evaluator

Theme: Building Autonomous Hiring Panels

1. The Challenge

Current AI interview tools are answer checkers. They grade based on keyword matching and rigid rubrics, failing to distinguish between a candidate who memorized LeetCode answers and one who genuinely understands system architecture.

Your goal is to build an Autonomous Technical Interview Panel — a multi-agent system that evaluates a candidate's resume and interview transcript, identifies gaps or contradictions, and reaches a hiring consensus the way a real committee would.

2. Input

You create all inputs yourself as part of the challenge. This is by design — it lets you stress-test your own system:

- Resume.
- Interview Transcript
- Job Description

3. Output

A candidate assessment, a log of discrepancies between claims and demonstrated knowledge, a final hire or no-hire verdict with reasoning, and a trace of how the agents challenged each other to get there.

Track 4 — Growth Intelligence Engine

Theme: Building the Growth Loop

1. The Challenge

Growth is not about writing blogs — it is about information arbitrage. Finding a technical insight in a research paper or a competitor's failure and converting it into a high-authority piece of content before anyone else does.

Your goal is to build an autonomous engine that listens to the internet for DataVex and converts signals into publish-ready growth assets — across multiple platforms, in the right voice, without human editing.

2. Input

You define both inputs yourself: a brand voice guide for DataVex, and a real external signal for your agent to work from. Both must be included in your submission.

3. Output

A strategy brief explaining the angle chosen and what was discarded, three publish-ready content assets, and a trace showing how the drafts evolved through the critique loop.

Good luck. Build something that surprises you.