شرح مسئله

با پیشرفت مدل‌های زبانی بزرگ و ابزارهای Coding Agent، شاهد ایجاد روندهایی مانند Vibe Coding هستیم که از طرفی جامعه توسعه‌دهندگان، و از طرف دیگر حفره‌های امنیتی و ضعف ساختاری نرم‌افزارها را افزایش می‌دهد.

یکی از حوزه‌هایی که Agentها می‌توانند در آن بسیار مؤثر باشند، **Chaos Engineering یا مهندسی آشوب** است. مهندسی آشوب به زبان ساده، مانند یک "واکسن" برای سیستم‌های نرم‌افزاری عمل می‌کند. به جای اینکه منتظر بمانیم تا یک مشکل یا قطعی غیرمنتظره در سیستم رخ دهد و کاربران را تحت تأثیر قرار دهد، به طور **عمدی و کنترل‌شده**، خرابی‌ها و شرایط بحرانی کوچک (مانند از دسترس خارج شدن یک سرویس، افزایش ناگهانی تأخیر در شبکه، یا پر شدن حافظه) را در محیط تست (یا حتی با احتیاط در محیط عملیاتی) شبیه‌سازی می‌کنیم. هدف این است که ببینیم سیستم چگونه به این "آشوب" واکنش نشان می‌دهد و نقاط ضعف پنهان آن را **قبل از اینکه منجر به مشکلات بزرگ شوند**، شناسایی و برطرف کنیم. این کار به ما کمک می‌کند تا سیستم‌هایی بسازیم که در برابر شرایط غیرمنتظره مقاوم‌تر و پایدارتر باشند.

در این چالش، شرکت‌کنندگان باید یک **Chaos Engineering Agent** طراحی کنند که این فرایند را تسهیل و خودکار می‌کند.

اهداف مسئله

به طور مشخص، عامل باید بتواند:

- با تحلیل اولیه سیستم (مثلاً از طریق مستندات یا بررسی کد)، نقاط بالقوه برای آزمایش را شناسایی کند (به طور مثال اتصالات بین سرویس‌های مختلف).
- سناریوهای آشوب مناسب را طراحی و پیشنهاد دهد.
- این آزمایش‌ها را اجرا کرده و نحوه واکنش سیستم را مشاهده و داده‌های مربوطه را جمع‌آوری کند.
- نتایج را تحلیل کرده، نقص‌های سیستمی و آسیب‌پذیری‌ها را شناسایی کند.
- پیشنهاداتی برای اصلاح و بهبود نرم‌افزار (مانند تغییرات کد، پیکربندی، یا معماری) ارائه دهد.
- (به عنوان یک قابلیت پیشرفته) حتی به طور خودکار (با نظارت یا تایید کاربر) برای ارتقای پایداری و کارایی نرم‌افزار اقدام کند.

هدف اصلی، افزایش کیفیت نرم‌افزار و کاهش زمان لازم برای شناسایی و رفع مشکلات پیچیده سیستمی است، و همچنین توانمندسازی تیم‌های توسعه برای ساخت سیستم‌های مقاوم‌تر، به خصوص اگر برخی اعضای تیم درک عمیقی از تمام پیچیدگی‌های داخلی نرم‌افزار نداشته باشند.

خروجی مورد انتظار

1. نمونه اولیه قابل اجرا از عامل مهندسی آشوب (در قالب Web App، CLI، یا API).
2. ویدئوی دمو که به وضوح عملکرد عامل در اجرای سناریوهای آشوب، شناسایی مشکلات و پیشنهاد راهکار را نشان دهد.
3. شرح کامل معماری و جریان کاری عامل شامل نحوه تعامل LLM با ابزارهای جانبی و سیستم هدف.

ابزارها و منابع پیشنهادی

- LLM APIs: GPT-4، Gemini, Claude، LLaMA و مدل‌های مشابه.
- چارچوب‌های عامل هوشمند (Agent Frameworks): LangChain، AutoGen، Semantic Kernel.
- ابزارهای ساخت واسط کاربری: Flask، Gradio، Streamlit.
- APIها و سیستم‌های نمونه برای تست عملیاتی.

نکات کلیدی داوری

به دنبال راه‌حل‌هایی هستیم که:

- تا حد امکان نیاز به دانش عمیق فنی در سمت کاربر نداشته باشد و نحوه استفاده از آن ساده باشد
- تجربه کاربری خلاقانه‌ای داشته باشد
- دقت و قابلیت اطمینان بالاتر از راهکارهای فعلی داشته باشد
- فرآیندها را به صورت خودکار یا با حداقل دخالت انسانی انجام دهد