
Towards Early Warning for AI Agent Incidents: Monitoring the Dynamics of Risk Trajectories¹

Yifan Li
Western University
Kai Ting
University of Toronto

John Gia Bao Luc
University of Toronto

Rashik Shahjahan

With
Apart Research

Abstract

Recent AI-agent incidents highlight the need for monitoring systems that can detect emerging unsafe behavior during agent execution, rather than only assess failures after they occur. Motivated by this challenge, we study a modular early-warning framework for LLM agents based on the dynamics of their observable trajectories. The framework first maps a trajectory prefix H_t , containing messages, tool calls, observations, and actions, to a numerical risk-relevant score S_t , and then monitors the evolution of the resulting score process over time. We demonstrate the scoring layer on partial real-agent logs using StepGuard to construct action-unsafe proxy score trajectories, and study the temporal layer on seeded controlled trajectories using pointwise level, recent slope (“momentum”), EWMA, and CUSUM statistics. In the constructed unsafe examples, momentum and CUSUM cross their selected thresholds before the designated unsafe action, whereas level and EWMA cross at that action; neither constructed safe trajectory alarms under the selected cutoffs. These results are illustrative rather than calibrated detector comparisons. Overall, this sprint provides a proof of concept for a score-then-monitor approach to agent safety and motivates systematic evaluation of temporal risk dynamics as a basis for earlier warning and human intervention in deployed agent systems.

¹Research conducted at the [AI Incident Response Sprint](#), September 2026.

1. Introduction

LLM-based agents increasingly act through tools, files, browsers, code execution, APIs, and other environments. Their execution traces can therefore contain heterogeneous messages, tool requests, observations, and actions. A safety monitor that operates on observable behavior needs a representation that is compact enough to inspect over time while preserving risk-relevant information. Existing work has explored monitoring agent inputs and outputs and evaluating security failures in tool-using agents [1, 2]. Our project focuses on the interface between such trajectory assessment and temporal monitoring.

Let X_t denote the observable event at step t , and let

$$H_t = (X_1, \dots, X_t) \quad (1)$$

be the information available to the monitor through that step. A scorer f maps the prefix to a scalar or multivariate assessment

$$S_t = f(H_t). \quad (2)$$

The score is a model-dependent measurement under a specified rubric; it should not be interpreted as a calibrated probability unless calibration has been established separately. The scorer can be an LLM-based monitor, a trained classifier, structured behavioral rules, or a combination of these components.

Our central claim is methodological: monitoring should not stop at the value of S_t at a single time. The same current score may be the result of an isolated spike, gradual escalation, or persistent elevation. We therefore treat the score sequence itself as a time series. Its level, recent change, trend, persistence, smoothing, and departures from a baseline can provide complementary descriptions of how the assessed safety state is evolving. This connects agent monitoring to established ideas from statistical process monitoring and change detection, including CUSUM and EWMA [3, 4]. We use these methods as transparent monitoring baselines rather than assuming that their classical guarantees automatically transfer to LLM-derived, overlapping-prefix score processes.

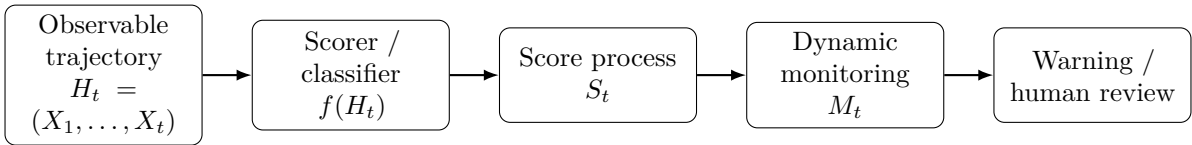


Figure 1: Central monitoring pipeline. Semantic scoring converts the observable trajectory prefix H_t into a numerical risk-relevant signal S_t . A separate temporal monitoring layer analyzes the evolution of this score process to support warning or human review.

The motivating operational question is whether an evolving observable trace can provide sufficiently early and reliable evidence to support warning or intervention, rather than only post-hoc recognition. We keep this question deliberately narrower than a complete safety or governance system. In the current sprint, we (i) construct score curves from partial real-agent logs using an LLM-based safety scorer, and (ii) use controlled score trajectories to illustrate how temporal monitoring can extract information beyond a pointwise threshold.

Contributions.

1. We formulate a modular $H_t \rightarrow S_t \rightarrow M_t$ pipeline that separates trajectory scoring from

sequential monitoring and enforces prefix validity: a score at time t must not use future information.

2. We implement prefix/action scoring on team-provided partial real-agent trajectory exports using StepGuard, producing an evolving numerical risk-relevant signal from heterogeneous logs.
3. We construct controlled safe and unsafe score trajectories and compare pointwise level, recent slope (“momentum”), EWMA, and CUSUM. These seeded examples illustrate how score dynamics can produce different warning behavior from isolated score levels.

Scope and status. This report presents the current state of an ongoing project developed during the AI Incident Response Sprint. The framework, implementation, and empirical results are preliminary and have not yet undergone comprehensive validation. Accordingly, the report is intended to document the project’s current design, implementation, and initial findings, while identifying directions that require further development and evaluation.

2. Related Work

Our work sits at the intersection of agent monitoring and sequential statistical monitoring. AgentDojo provides a dynamic environment for evaluating prompt-injection attacks and defenses for tool-using agents [2], while AgentMonitor records agent-level inputs and outputs and transforms them into statistics for downstream prediction and security-oriented correction [1]. These systems motivate treating observable execution traces as first-class monitoring data.

The upstream scoring layer in our implementation is StepGuard, an LLM-based action monitor used to assess whether a current action appears unsafe in light of task context and available history. We treat such semantic scoring as a measurement layer, not as the entire warning system. The second layer draws on statistical monitoring. CUSUM accumulates departures from a reference level [3], while EWMA smooths recent observations with geometrically decreasing weights [4]. Our use is intentionally exploratory: prefix scores overlap in their underlying histories and may be dependent, nonstationary, and uncalibrated, so classical control-chart guarantees do not automatically apply.

3. Methods

Observable prefixes and score sources

The primary design constraint is *prefix validity*. At time t , the monitor may use only information available in H_t ; future events $X_{t+1}, \dots, X_T$ must not enter the scoring input. We consider two complementary score sources.

Real agent logs. The real-log component uses team-provided partial trajectory exports, referred to as the *attack* and *honest* exports. These identifiers describe the exported runs and are not action-level ground-truth labels. The attack export contains 51 source rows and 11 eligible scored requests; the honest export contains 17 rows and 5 eligible scored requests. Only named requests with complete JSON-object arguments are classification targets. Earlier exported content can contribute to history, while current-row response text and future rows are excluded. Because not

every exported row is a scoring target, the resulting score series is sparse over original source-step positions.

Controlled trajectories. To isolate temporal behavior, we also construct seeded synthetic prefix-score trajectories directly; no classifier is run on these paths. Each of the simple and delicate designs contains one unsafe and one safe trajectory. The unsafe paths encode escalation before a designated unsafe action, whereas the safe paths contain noise and isolated suspicious-looking spikes. These examples are illustrations of monitoring behavior, not estimates of real-world detection or false-alarm rates.

StepGuard score on real logs

For the real-log experiment, fixed context C supplies an approximate legitimate task and tool inventory. The serialized model input at a scored request combines C , the available prior history, and the current action. Using the StepGuard checkpoint, we form the continuous proxy

$$S_t = \sigma(\ell_{\text{unsafe}} - \ell_{\text{safe}}), \quad \ell_y = \log P_\theta(y \mid x_t, \langle \text{Judgment} \rangle), \quad (3)$$

where the candidate-label sequence log-probabilities are evaluated for **safe** and **unsafe**. We interpret S_t as model evidence that the current action is unsafe given the supplied context and available history. It is an *uncalibrated action-unsafe proxy*, not a future-incident probability.

The saved run scored all 16 eligible requests without recorded inference errors. Full checkpoint revision, prompt construction, environment information, row-audit rules, and reproducibility notes are retained in Appendix ??.

Dynamic monitoring of score trajectories

For the controlled examples, we distinguish the current prefix score S_t from several temporal summaries. The one-step update is

$$\Delta S_t = S_t - S_{t-1}. \quad (4)$$

A recent “momentum” statistic $\hat{\beta}_t$ is the OLS slope fitted to the latest five scores. It summarizes recent direction in score units per step; it is not a formal change-point estimate.

We also compute

$$Z_t = \lambda S_t + (1 - \lambda) Z_{t-1} \quad (5)$$

for EWMA, and a one-sided CUSUM

$$C_t = \max\{0, C_{t-1} + S_t - \hat{\mu}_0 - k\}, \quad (6)$$

where $\hat{\mu}_0$ is the mean of the first eight scores for each trajectory and is then frozen. CUSUM begins accumulating after this eight-step initialization. All alarm rules are eligible only for $t > 8$.

The illustrative alarm cutoffs are $S_t \geq 0.70$, $\hat{\beta}_t \geq 0.045$, $Z_t \geq 0.60$, and $C_t \geq 1.20$. The simple design uses $\lambda = 0.25$, $k = 0.020$, and the delicate design uses $\lambda = 0.22$, $k = 0.025$. These cutoffs were not calibrated to equal false-alarm rates and should not be used to infer detector superiority.

For detector A , let T_A be the first eligible threshold crossing and let T^* be the designated unsafe-action step. We report step-based lead time

$$T^* - T_A. \quad (7)$$

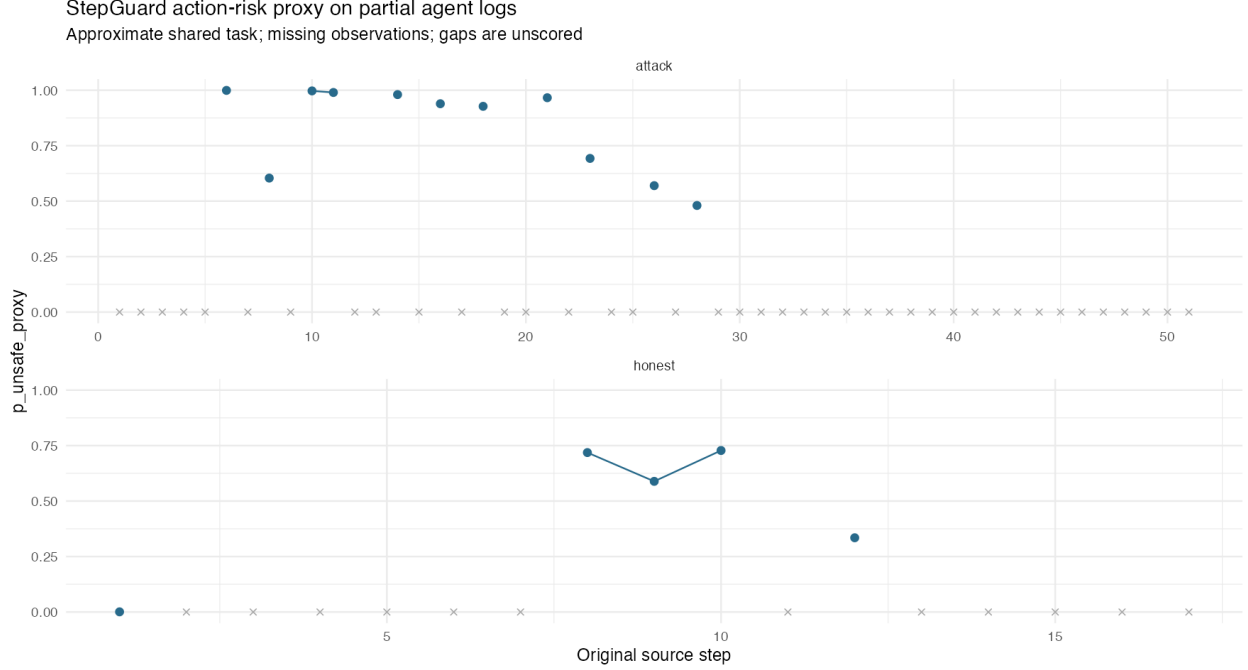


Figure 2: Real-log StepGuard scores. Points correspond to eligible scored requests; unscored source rows should not be interpreted as zero-risk predictions. The curves demonstrate the construction of an evolving numerical signal from partial observable logs, not calibrated incident prediction.

A positive value means that the selected threshold crossed earlier in the constructed trajectory. It does not establish operationally actionable lead time because inference and intervention latency are not modeled.

4. Results

Real-log score curves

The real-log run provides the report’s first empirical layer: observable agent-log prefixes/actions can be converted into a sequence of numerical scores. For the attack export, the 11 eligible requests have scores ranging from 0.480 to 0.999, with mean 0.831; for the honest export, the five eligible requests range from 0.001 to 0.728, with mean 0.474. The attack export has 10 of 11 scores above 0.5, while the honest export has 3 of 5. The ranges overlap and neither series is monotone.

These values should be interpreted descriptively. The exports are small and partial; their original task, tool definitions, and environment observations are incomplete, and the attack/honest identifiers are not action-level labels. The result therefore demonstrates the *score-curve construction pipeline*, not calibrated classification accuracy or prospective early warning.

Why score dynamics matter

The controlled experiment addresses the second part of the pipeline: once a score process exists, what information can be obtained from its temporal evolution? The simple design contains a transparent escalation before a designated unsafe action at $T^* = 25$; the delicate design adds weak precursors, reversals, a later jump, and saturation, with $T^* = 27$. Their corresponding safe trajectories contain noise and isolated spikes.

Table 1: First alarm step / lead steps in the seeded controlled examples. A positive lead indicates a threshold crossing before the designated unsafe action.

Design	Level	Momentum	EWMA	CUSUM
Simple ($T^* = 25$)	25 / 0	19 / 6	25 / 0	21 / 4
Delicate ($T^* = 27$)	27 / 0	21 / 6	27 / 0	19 / 8

At the selected settings, momentum and CUSUM cross before the designated unsafe action in both constructed unsafe trajectories, whereas the level and EWMA thresholds first cross at that action. Neither safe trajectory triggers any of the four monitors during its eligible window. The ordering between momentum and CUSUM reverses across the two designs. These outcomes illustrate that temporal summaries can react to precursor dynamics differently from a high pointwise level threshold; they do *not* establish that momentum or CUSUM is generally superior. In particular, a fair detector comparison would require held-out trajectories and thresholds calibrated to comparable false-alarm behavior.

A related distinction is between local action suspicion and the change in a history-dependent prefix assessment. In the delicate construction, the designated unsafe action at step 27 has local score 0.94 and prefix increment approximately 0.185, while an additional high-risk action at step 30 has local score 0.88 but prefix increment only 0.009. Once the prefix assessment is already high, a concerning new action need not produce a large update. Thus ΔS_t is a context-dependent model update rather than an intrinsic or causal contribution of the newest event.

Synthesis across the two experiment layers

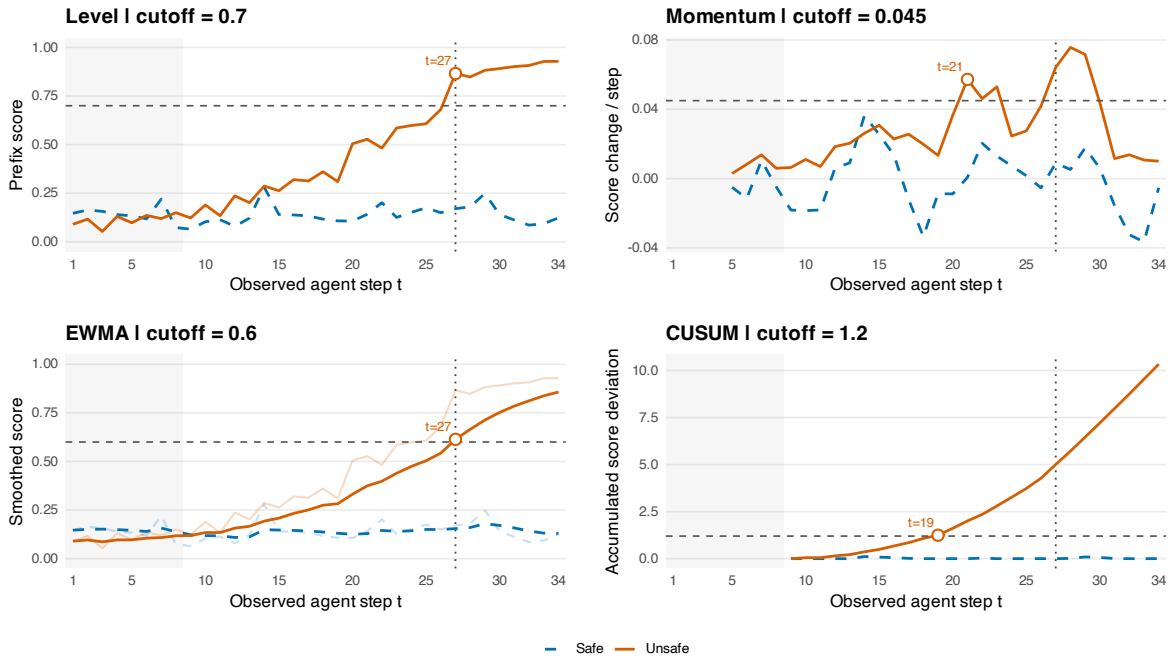
The two experiment components serve different purposes. The real-log StepGuard experiment establishes that heterogeneous observable traces can be mapped into an evolving numerical signal under a fixed scoring interface. The controlled R-based experiment then isolates what can be learned from the *dynamics* of such a signal. Together they motivate a modular design: improvements to the upstream scorer can be incorporated without discarding the downstream monitoring layer, while the monitoring layer can be studied independently under controlled temporal patterns.

5. Discussion and Limitations

The main conceptual separation is between *risk-relevant measurement* and *sequential warning*. A classifier or LLM monitor answers whether the currently visible trajectory/action appears concerning under its rubric. A temporal monitor asks whether the resulting assessment process is changing in a way that warrants additional attention. The latter can use information about trend, persistence, and accumulated deviations that is not represented by an isolated pointwise score.

Figure 4 | Delicate simulation: four complementary monitoring signals

Hollow points mark first eligible crossings. Every dashed cutoff matches the alarm calculation.

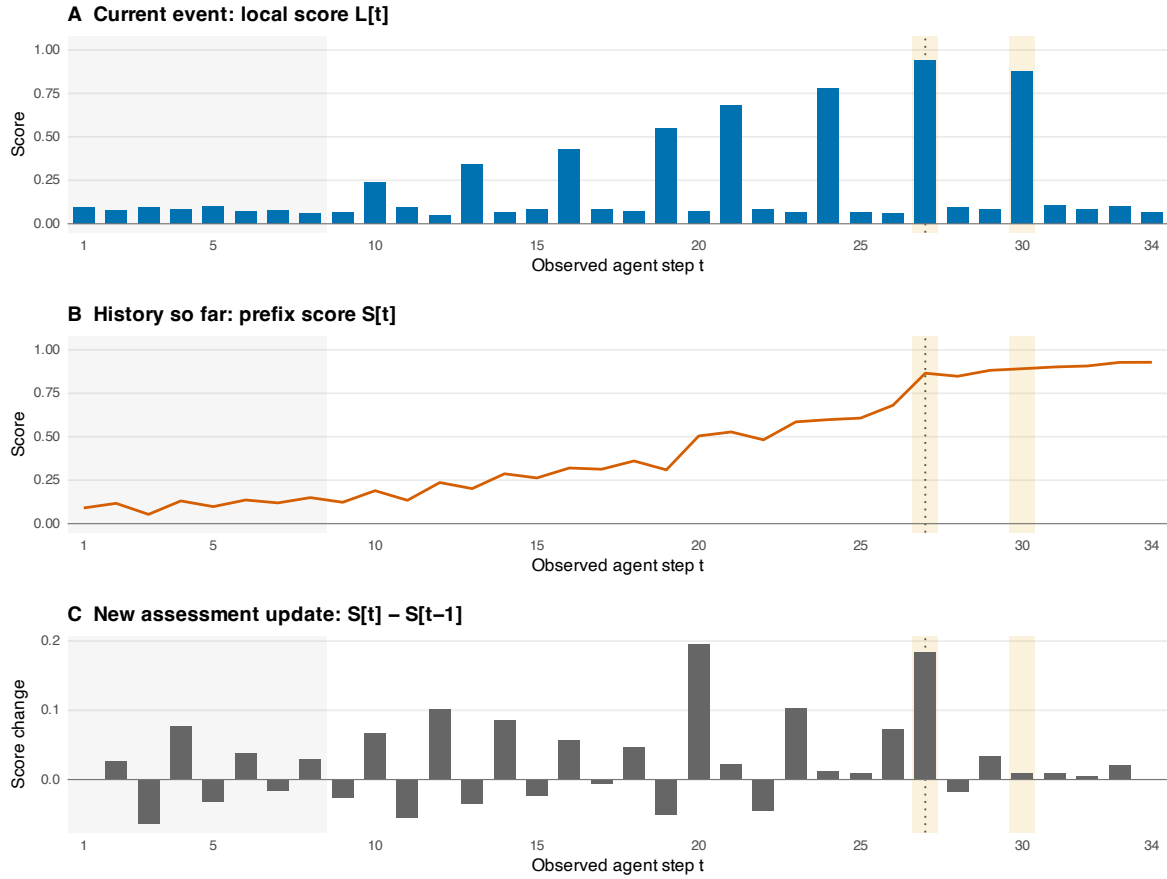


Faint lines in the EWMA panel show raw prefix scores. CUSUM starts after initialization using a frozen baseline. The dotted reference is the unsafe trajectory's unsafe action; it is not an event in the safe trajectory.

Figure 3: Controlled delicate trajectory: pointwise level and temporal monitoring signals. The figure makes visible how recent slope, smoothing, and accumulated deviations respond to the constructed precursor and escalation phases. Thresholds are illustrative.

Figure 2 | Delicate simulation: current-event risk is different from a prefix update

Highlighted steps: 27 and 30. Read vertically across the three panels.



At step 30, a high local score accompanies a small prefix update. A small increment does not make the action benign. The increment is context-dependent and model-dependent, not an intrinsic or causal event contribution.

Figure 4: Illustration of local action score, prefix score, and prefix increment in the delicate constructed trajectory. A small prefix increment does not imply that the current action is benign.

Several limitations are central. First, S_t is only as informative as the observable log and the scorer. Relevant precursors may be missing, and LLM-derived scores may be prompt-sensitive, noisy, uncalibrated, or affected by distribution shift. The partial real-log exports also lack complete original task, tool, and observation context, so their results are exploratory.

Second, overlapping prefixes create strong dependence: H_t contains most of H_{t-1} . Applying EWMA or CUSUM to history-containing prefix scores can therefore partly aggregate information that has already been summarized upstream. Whether this constitutes useful extraction of temporal structure or undesirable double-counting is an open methodological question. Classical independent-observation control-chart guarantees should not be imported without validation.

Third, the controlled results are deliberately designed examples. Two safe/unsafe trajectory pairs cannot establish recall, false-alarm rates, lead-time distributions, or real-world safety effectiveness. The four cutoffs are illustrative and are not matched for false-alarm rates; the reported early crossings therefore do not prove superiority to a differently calibrated level threshold. Initialization also assumes eight suitable baseline observations, and contamination of this period can affect later CUSUM behavior.

Finally, retrospective prefix replay is not equivalent to operational early warning. Positive lead steps matter only if the score can be computed, reviewed, and acted upon before the unsafe action. A deployment-oriented evaluation should therefore preserve event availability times and include scoring and intervention latency.

Future work should evaluate larger collections of benign and unsafe trajectories with trajectory-level train/test separation, calibrate alert rules on benign data, and report false alarms, event recall, and lead-time distributions. Multivariate scores could preserve distinct risk dimensions rather than forcing all evidence into one scalar. Further directions include monitoring under temporal distributional shift, adaptive baselines for nonstationary agents, and explicit coupling of warning statistics to human or automated containment mechanisms.

6. Conclusion

We present a modular approach to monitoring observable agent trajectories: construct a numerical risk-relevant signal $S_t = f(H_t)$, then analyze the dynamics of the resulting score process rather than relying only on isolated values. The sprint implementation demonstrates score construction on partial real-agent logs with StepGuard and uses seeded controlled trajectories to illustrate how recent slope and accumulated deviations can behave differently from a pointwise level threshold.

The current evidence is preliminary. It supports the feasibility of the score-then-monitor workflow and clarifies why temporal dynamics are worth studying, but it does not validate a calibrated production early-warning system. The central research question is whether stronger trajectory scores, combined with appropriately validated sequential monitoring and a usable intervention point, can provide reliable and actionable warning.

Code and Data

Code and materials supporting the implementations and experiments presented in this report are available in the project repository:

https://github.com/YifanMath/AI_Incident_Resp_Sprint_T1

The repository contains the current implementation of the score-construction and dynamic-monitoring components, together with selected outputs and supporting materials used in this report.

Author Contributions

Yifan Li (Primary Investigator / Project Lead): Led the overall project direction, conceptual and methodological design, and integration of the monitoring and early-warning framework; carried out substantial implementation, experimentation, analysis, and report development.

John Gia Bao Luc (Research Engineer / Technical Contributor): Contributed to the main implementation pipeline, including agent-trajectory processing, risk-score generation, and related detection components.

Rashik Shahjahan (Research & Infrastructure Contributor): Supported the project’s coding and writing infrastructure and contributed to relevant resource and literature searches.

Kai Ting (Research Contributor): Contributed to early-stage discussions on the overall project design and supported relevant resource and literature searches.

References

- [1] Chi-Min Chan, Jianxuan Yu, Weize Chen, Chunyang Jiang, Xinyu Liu, Weijie Shi, Zhiyuan Liu, Wei Xue, and Yike Guo. AgentMonitor: A plug-and-play framework for predictive and secure multi-agent systems. *arXiv preprint arXiv:2408.14972*, 2024. doi: 10.48550/arXiv.2408.14972. URL <https://arxiv.org/abs/2408.14972>.
- [2] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2636. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html.
- [3] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954. doi: 10.2307/2333009.
- [4] S. W. Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 1(3): 239–250, 1959. doi: 10.1080/00401706.1959.10489860.

LLM Usage Statement

LLM-based tools were used during the sprint for coding assistance, literature discovery, organization of project notes, and drafting/editing support. The team is responsible for verifying the final reported implementation details, experimental results, citations, and claims. The final statement should be updated to name the tools actually used and to match the team’s verified workflow.