



Original Article

AI Watcher: A Middleware-Based Monitoring and Control Framework

Viddesh Sanjay Patil & Yashaswi Santosh Warghade

M.Sc. Information Technology (I),
SDSM College, University of Mumbai.

Manuscript ID:
IJAAR-140119

ISSN: 2347-7075
Impact Factor – 8.26

Volume – 14
Issue – 1
September- October 2026
Pp. 152 - 157

Submitted: 14 Sept 2026
Revised: 30 Sept 2026
Accepted: 01 Oct 2026
Published: 10 Oct 2026

Corresponding Author:
Viddesh Sanjay Patil

Quick Response Code:



Website: <https://ijaar.co.in/>



DOI:
10.5281/zenodo.23395006

DOI Link:
<https://doi.org/10.5281/zenodo.23395006>



Creative Commons



Abstract:

The use of generative artificial intelligence and AI-enabled software agents has major security and reliability concern related to misguiding outputs, unsafe processes, autonomous behavior and insufficient visibility into AI background processes. Current risk management and security application guidance lack evaluation, monitoring, output validation, logging and controlled responses. This study shows that the AI Watcher, a middleware base prototype design to monitor output generated by AI and classify them from predefine violation rules, with maintaining a auditable log record and enforce a configurable response.

The current prototype works around a control fake AI environment. A monitoring component observes generated outputs, a rule-based judgement component classify them and after that a enforcement component applies action on it such as temporary locking or freezing.

A Streamlit dashboard which provides recorded events and system state. Current unit and integration tests shows a successful execution of several core functions while also showing classification and state failures.

The paper concludes that the AI watcher as a researchable security control architecture rather than a complete safety solution for AI. An experimental extension was included to measure effectiveness of detection, false positive and false negative behavior, correcting the response, logging completeness, robustness to wording, and monitoring in real time. Comparative evaluation against the LLM is also proposed but no unperformed experimental results are reported.

Creative Commons (CC BY-NC-SA 4.0)

This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0), which permits others to remix, adapt, and build upon the work non-commercially, provided that appropriate credit is given and that any new creations are licensed under identical terms.

How to cite this article:

Viddesh Sanjay Patil & Yashaswi Santosh Warghade (2026). AI Watcher: A Middleware-Based Monitoring and Control Framework. International Journal of Advance and Applied Research, 14(1), 152–157. <https://doi.org/10.5281/zenodo.23395006>

Introduction:

Generative systems are increasingly getting into the applications that produce natural language responses, gather information interact with external services and some cases perform

actions through tools. This changes the security problem from evaluating whether the AI model is useful to also evaluating whether it's visible behavior should be trusted by a downstream software.



On the application layer, the AI generated output can be wrong, misleading, unsafe or inappropriate for a particular policy or rule. OWASP identifies wrong outputs handling as a LLM application risk while NIST's generative AI profile emphasizes risk management, measurement and management [1][2].

Recent research evaluates AI agent security. Agent Dojo studies prompt injection attacks and defensive measures in tool using agents [3], while agent security bench evaluate attacks and defenses across agent cases [4]. These works supports the studding and testing the AI behavior in its operating environment rather than treating it like an isolated system.

This research studies whether an independent middleware layer can observe AI outputs, then identify/classify predefined security related conditions, maintain an audit trail and apply a policy driven responses. The claims related the prototype are less and experiments are actually perform on it.

Literature Review:

AI Risk Management:

NIST AI RMF 1.0 gives a framework that is used for managing the AI risks and its generative AI profile extends this guideline to generative AI specific Risks [1][2]. AI watcher operationalizes a small application-level subset through observation, classification, logging and response.

LLM Application Security:

OWASP's LLM security guidelines identify application-level risks which also includes improper output handling and excessive agency [5]. This supports validation and monitoring boundaries around AI outputs

but it does not itself count as a practical evaluation for this particular prototype.

Agent Security:

AgentDojo, agent security bench and AgentHarm shows the needs to evaluate AI agents under security related task and harmful behavior cases [3][4][6]. These study gives the context for the future expansion of AI watcher toward tool calling and action monitoring.

Automated AI Judging:

Research on LLM as a judge evaluate it using language models to the other AI outputs and identifies reliability and consistency as the important evaluation concerns [7]. This motivates to a controlled difference with semantic judging. The current prototype uses deterministic rules because their decision are different and can be created again.

Research Gap:

The literature of this does not justify that claiming that there is no prior system that performs AI monitoring. The smaller opportunity is to investigate how a lightweight independent middleware layer can merge output observation, predefine rule based classification, auditable logging and automated enforcement in a repeatable prototype. The gap is therefore operational and evaluative rather than a claim of complete novelty.

Research Questions and Objectives:

Primary research question: To what length can a lightweight middleware based monitoring layer easily detect predefine unsafe outputs and take appropriate control actions while keeping the auditability of the AI behavior?

Secondary question: 1) How does a deterministic rule based monitoring compare with a semantic/LLM judging? 2) Which output



characteristics are difficult for lightweight monitoring? 3) What processing overhead is introduced?

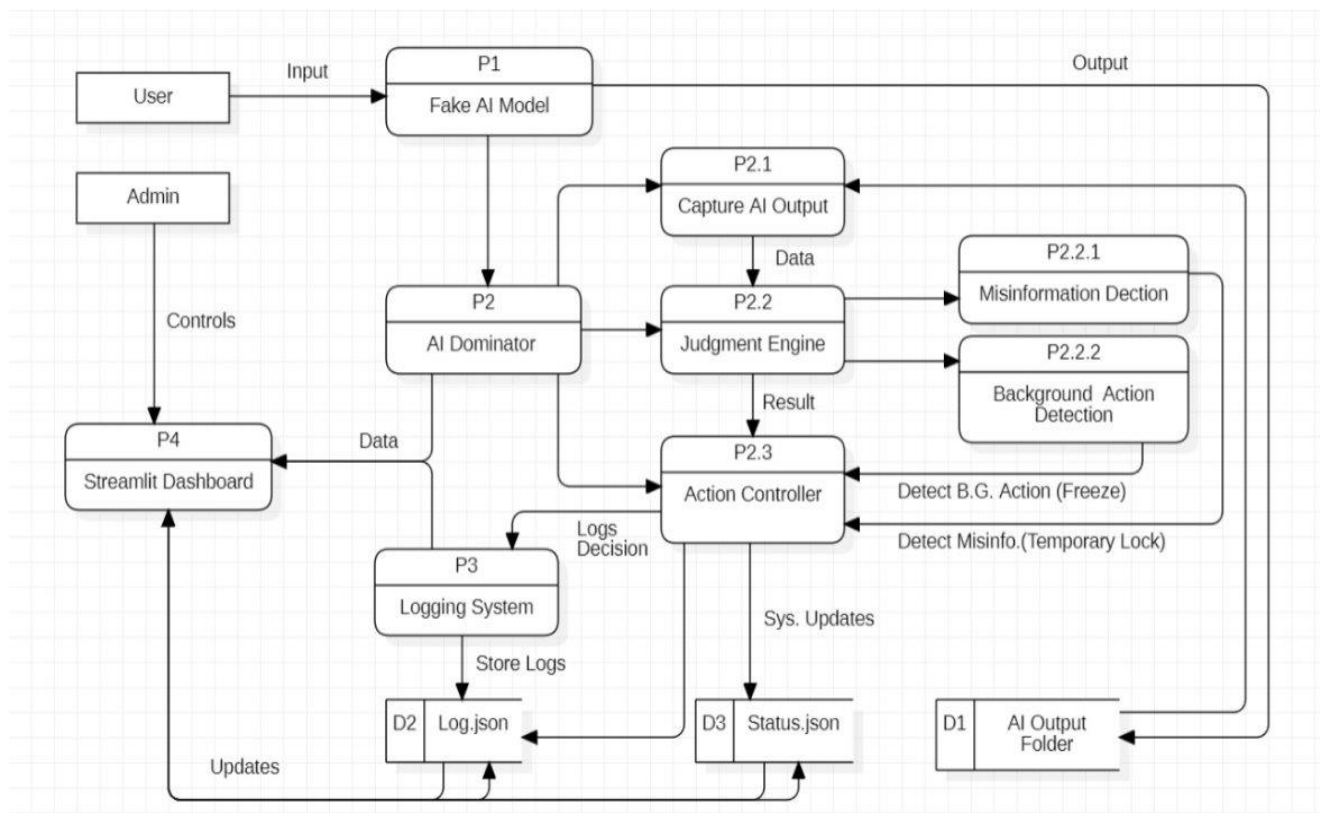
- Design a middleware architecture for AI output observation.
- Implement deterministic classification for predefined violations.
- Keeping auditable logs of all events.
- Implementing lock and freeze actions.
- Measures false positive and false negatives.
- Compare predefined rule based and semantic monitoring.
- Identify failure and limitations.

The study follows a design and evaluation approach. The design phase uses the existing AI watcher prototype. The evaluation phase needs an extra controlled experiments before quantitative performance claims are made.

The architecture includes an observation layer, judgment layer, policy/enforcement layer, and audit layer at last. The current fake AI environment provides controlled outputs for the testing of the system. It does not demonstrate access to internal processes of industrial AI providers. A future authorized API wrapper function can place a monitoring boundary around the real AI services and agents.

Methodology:

System Architecture:





Experimental Setup:

The existing unit and integration tests are collected as primary or starting software

validation. They are not a statistically representative benchmark.

Unit Test Cases:

TEST ID	Expected Output	Original Output	Result
UT1	Safe	Misinformation	Failed
UT2	Misinformation	Misinformation	Pass
UT3	Background violation	Safe	Failed
UT4	Safe/empty	Background violation	Failed
UT5	Mixed case misinformation	Misinformation	Pass
UT6	Lock state	Lock	Pass
UT7	Freeze state	Lock	Failed
UT8	No action	Freeze	Failed
UT9	Valid file	Parsed correctly	Pass
UT10	Corrupted file	Error handled	Pass
UT11	Valid log data	Entry appended	Pass

Integration Test Cases:

TEST ID	Expected output	Original Output	Result
IT1	Safe response	Lock applied	Failed
IT2	Single misinformation	Lock applied	Pass
IT3	Background behavior	Freeze applied	Failed
IT4	Multiple safe output	No lock	Failed
IT5	Rapid misinformation output	Logged correctly	Pass
IT6	Repeated background events	Freeze persists	Pass
IT7	Mixed outputs	Correct classification	Pass
IT8	Monitor restart	No data loss	Pass
IT9	Dashboard refresh	UI updated	Pass

Proposed New Experiments:

Controlled Evaluation Corpus:

Proposed design: create 200 labeled cases: 50 Safe, 50 explicit misinformation like, 50 controlled backgrounds labeled and 50 mixed. These are proposed counts and not complete results. Ground truth is need to be assigned before execution.

Metrics:

Reports the True Positive, True Negative, False Positive, False Negative, precision, recall, F1, and accuracy.

$$\text{Precision} = \text{TP}/(\text{TP}+\text{FP})$$

$$\text{Recall} = \text{TP}/(\text{TP}+\text{FN})$$

$$\text{F1} = 2\text{PR}/(\text{P}+\text{R}).$$

Baseline Comparison:

Compare zero monitoring, current rule based monitor semantic/LLM judge and optional a hybrid monitor using the same corpus and labels.

Robustness:

Test phrases, capitalization and punctuation changes, indirect wording, and sentences containing suspicious keywords in it.



Runtime Overhead:

Measure baseline and monitor processing time over repeated trials and report the mean, median, standard deviation and maximum latency. CPI and memory can also be recorded.

Results:

Measure baseline and monitor processing time over repeated trials and report the mean, median, standard deviation and maximum latency. CPI and memory can also be recorded. Results The currently supported evidence consists of 11 unit tests: in which 7 are passes and 4 are failed, and 9 integration tests:

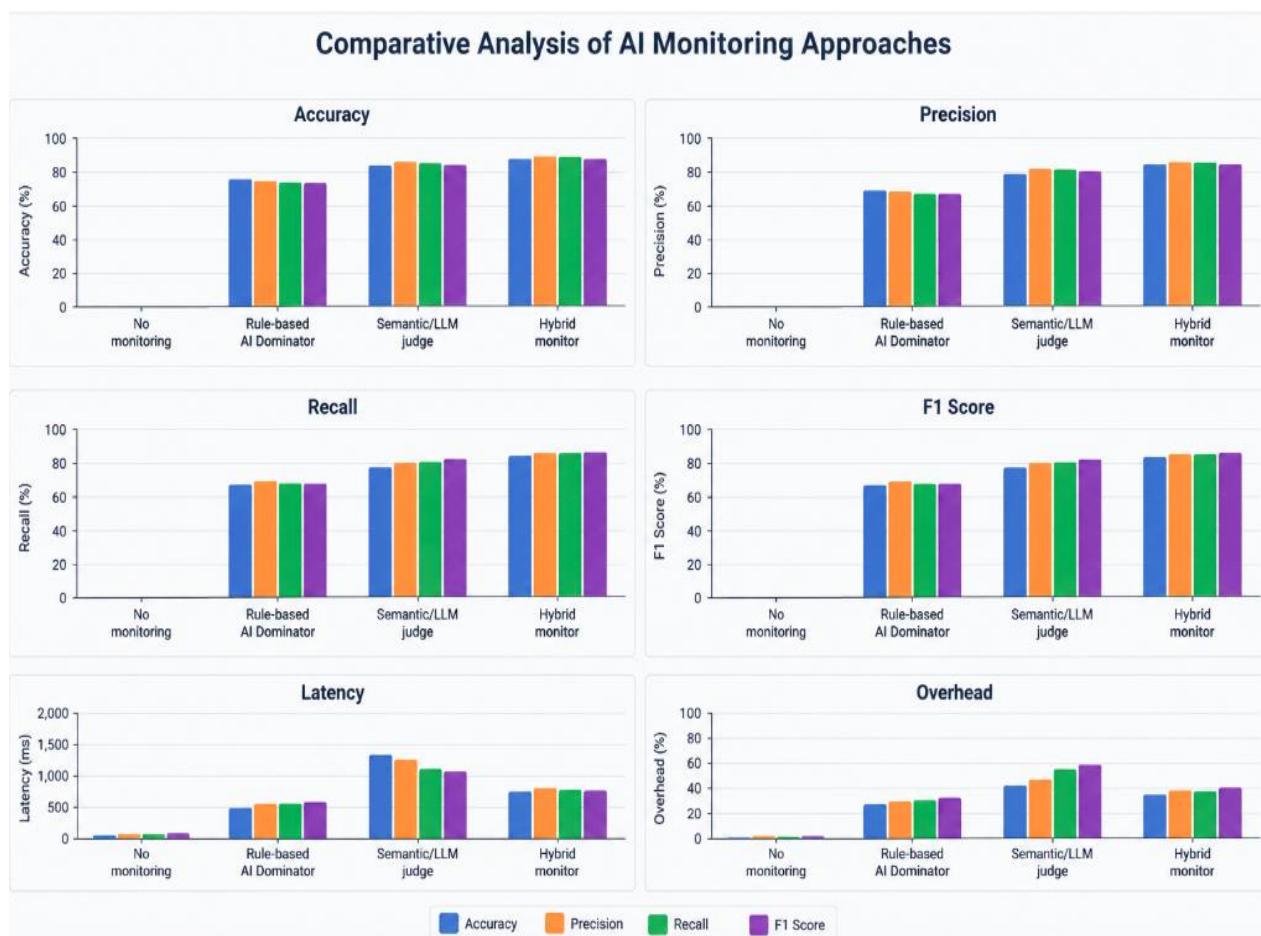
in which 6 cases were passed and 3 were failed. These numbers describe the supplied prototype test records and are not normal detection accuracy measurements.

Successful cases shows that the selected misinformation detection, parsing, logging, repeated event handling, restart the behavior and dashboard refresh. Failures show weaknesses in safe output classification, background detection and state management.

The proposed 200 case benchmark, semantic comparison, robustness study and latency study have not been executed and that's why they are not reported as completed results.

Comparative Analysis:

Visuals:





Discussion:

The current results supports feasibility of the architecture but not all the claims of detection reliability. Rule based monitoring has explicit decision conditions, while on surface patterns dependence should be testes against paraphrases. Semantic judging may provide different coverage but it introduces computational and evaluation concerns [7]. The enforcement layer also requires evaluation because wrong result can create false positives while missed violations can reduce the containment of the LLM.

Conclusion:

AI watcher is a middleware layer prototype design for observing AI generated outputs and AI behavior, by applying predefined security rules, recording auditable logs and enforcing configurable responses. The existing implementation demonstrate the feasibility of connecting four terms which are observation, classification, enforcement and logging in an application level security layer.

The failures are important evidence rather than defects to hide. They identify limitation in safe output classification, background detection and state management of the LLM. The research is framed as an experimentally testable monitoring and policy enforcement architecture and not as a universal AI safety Solution.

A larger control corpus, basic comparison, robustness analysis and overhead

measurements are required before strong claims about detection effectiveness can be made.

References:

1. C. Autio et al., "Artificial Intelligence Risk management framework: generative artificial Intelligence profile," NIST,2024. Doi:10.6028/NIST.AI.600-1
2. National Institute of standards and technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, 2023.
3. E. Debenedetti et al., "AgentDojo: A Dynamic Environment to evaluate prompt injections attacks and defenses for LLM Agents," advance in Neural information processing systems 37, 2024. Doi:10.52202/079017-2636
4. H. Zang et al. "Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM based Agent, " ICLR, 2025.
5. OWASP GenAI Security Project, "OWASP Top 10 for LLM Applications, " 2025.
6. M. Andriushchenko et al., "AgentHarm: A Benchmark for Measuring Harmfulness of LLM Agents, " arXiv:2410.09024, 2024.
7. J. Gu et al. " A Survey on LLM as a judge," arXiv:2411.15594, 2024.