Defense-in-Depth for Generative AI: An Architectural Evaluation and Comparison of LLM Guardrails

Caroline Miranda Pereira Lima[1], Jane Piantoni[1], Felipe Cajaiba Zambelli[1], Javier Martinez Silva[1], Alexandre Melo Braga[1]

[1]CPQD, Rua Dr. Ricardo Benetton Martins, 1.000, Campinas - SP

{carolinem, janepi, fzambelli, javiers, ambraga}@cpqd.com.br

Abstract. *This paper analyzes multi-layered guardrail architectures designed to secure LLMs at runtime. By systematically comparing five frameworks (Prompt Guard, Llama Guard, Llama Firewall, NeMo, and Amazon Bedrock), the study evaluates their structural designs and operational trade-offs through a qualitative architectural systematization. Ultimately, it concludes that selecting the appropriate architecture requires a systematic approach to balance security rigor and model utility.*

1. Introduction

Enterprise systems are increasingly incorporating Large Language Models (LLMs) into conversational services, Retrieval-Augmented Generation (RAG) pipelines, and autonomous AI agents. As Generative AI (GenAI) becomes embedded into production environments, these systems evolve into interconnected ecosystems involving orchestration services, external tools, vector databases, and multi-agent interactions, introducing significant security, privacy, and reliability challenges [Dong et al. 2025]. Unlike traditional software applications, LLM-based systems exhibit probabilistic and context-dependent behavior, making them susceptible to prompt injection, jailbreak attacks, hallucinations, insecure tool invocation, and data leakage [NIST 2025; OWASP 2026]. Studies show that model-centric alignment mechanisms alone are insufficient to guarantee safe behavior in adversarial and tool-integrated environments [Tripathi et al. 2026; Dong et al. 2024]. Consequently, runtime protection mechanisms have become increasingly important in enterprise AI infrastructures.

In this context, guardrails are evolving from isolated moderation filters into architectural security layers responsible for validating interactions, enforcing policies, and constraining unsafe generations in production systems. Recent frameworks combine semantic classifiers, orchestration pipelines, hallucination detectors, and Personally Identifiable Information (PII) validation mechanisms operating as defense-in-depth architectures for GenAI applications. However, current literature still lacks strong systematization regarding how these protection mechanisms are architecturally organized and deployed in enterprise environments. This paper analyzes guardrail architectures for enterprise LLM systems from a software engineering and cybersecurity perspective oriented by frameworks such as Prompt Guard, Llama Guard, Llama Firewall, NVIDIA NeMo Guardrails, and Amazon Bedrock Guardrails, The main contribution is a qualitative systematization of architectural patterns and runtime validation strategies to support deployment decisions for secure and scalable enterprise AI environments.

2. Related work

Studies increasingly treat guardrails as system-level security components for runtime protection in GenAI systems [Dong et al. 2025]. International initiatives such as the NIST adversarial machine learning taxonomy [NIST 2025], the NIST AI Risk Management Framework (AI RMF) [NIST 2023], the OWASP Top 10 for Large Language Model Applications [OWASP 2025], and the OWASP AI Security Overview [OWASP 2026] reinforce this direction by emphasizing layered protection and defense-in-depth strategies for LLM applications [NIST 2025; OWASP 2026].

Guardrails are commonly embedded into middleware layers or orchestration pipelines capable of intercepting inputs and outputs before interaction with the core model [Rebedea et al. 2023]. These protection layers combine semantic classifiers, moderation filters, and prompt inspection mechanisms for runtime protection [Malik et al. 2025]. Frameworks such as NVIDIA NeMo Guardrails [Rebedea et al. 2023], Llama Guard [Inan et al. 2023], and PromptGuard [Alzahrani 2026] illustrate this architectural approach, aligned with the MITRE ATLAS mitigation AML.M0020 [MITRE ATLAS 2025]. Enterprise-oriented platforms demonstrate the evolution of guardrails from isolated moderation filters into distributed runtime security ecosystems. Architectures such as LlamaFirewall, NVIDIA NeMo Guardrails, and Amazon Bedrock Guardrails [Amazon Web Services 2025] adopt multi-stage validation pipelines and distributed monitoring strategies to support scalability in production [Devino et al. 2025].

Despite advances, guardrails remain vulnerable to adversarial strategies capable of bypassing superficial filtering layers [Paim et al. 2025]. To ensure robustness, research advocates for systematic architectural design involving validation, orchestration, and continuous monitoring [Dong et al. 2024]. Recent literature highlights the need for red-teaming and empirical evaluation using tools such as Garak [Leon et al. 2024], combined with benchmarks like HarmBench [Mazeika et al. 2024] and JailbreakBench [Chao et al. 2024]. These tools validate guardrail efficacy and establish system-level metrics for adversarial environments [Azambuja et al. 2026].

3. Methodology: architectural design and implementation for guardrails

Based upon an *Input/Output (I/O) Engineering* perspective, the methodology adopted a multi-layered defense approach that implements neuro-symbolic systems to dynamically intercept data flows. Open-source architectures were evaluated in Google Colab (T4 GPUs): NVIDIA NeMo Guardrails deterministically guided the AI using Colang scripting, while Meta's Llama Firewall orchestrated Prompt Guard for adversarial defense and Llama Guard for semantic auditing, optimized through fine-tuning. In contrast, Amazon Bedrock Guardrails was tested on a managed AWS infrastructure, parameterizing content filters and denied topics to assess real-time prompt attack detection, PII masking, and contextual grounding. Ultimately, this approach balances structural efficiency with architectural flexibility, ensuring robust security without degrading the model's fluency and problem-solving capabilities.

4. Results and discussion: architectural analysis of guardrail frameworks

This section presents the results obtained from the structural and functional analysis of diverse guardrail architectures designed to secure LLMs, exploring architecture's characteristics, respective strengths, limitations, and operational trade-offs through a qualitative architectural systematization. The detailed structural diagrams of the evaluated architectures (Figures 1 to 5) are provided in the Appendix.

4.1. Prompt Guard

Designed as a specialized first-line input defense, Prompt Guard architecture (Figure 1) intercepts user messages through a lightweight *BERT-style* classifier before they reach the Core LLM. This structural choice brings a significant practical advantage: being a small encoder-only classification model allows it to intercept inputs with near-zero latency overhead, unlike slow and compute-heavy LLMs. It detects prompt injections or jailbreak attempts and immediately halts malicious flows with a predefined response [Chennabasappa et al. 2025]. If a malicious intent is detected, the flow is immediately interrupted with a predefined response. Although highly computationally efficient with low latency, this architecture is constrained by a 512-token context window and a significant drop in multilingual effectiveness in its smallest versions (e.g., 22M parameters).

4.2. Llama Guard

The Llama Guard architecture functions as a comprehensive semantic evaluator, acting on both ends of the LLM interaction. According to its architecture (Figure 2), it operates as an assessment model that communicates directly with the Input Guardrails (Check Input) and Output Guardrails (Check Output). It evaluates whether user's prompts or Core LLM's generated responses violate specific ethical and safety taxonomies. If an interaction is flagged as "UNSAFE," the system triggers a standard flow interruption [Inan et al. 2023]. Llama Guard demonstrates a deep semantic understanding and adaptability via zero-shot or few-shot prompting, with newer versions introducing native multimodal processing (early fusion). A limitation of Llama Guard is the high computational cost for continuously running large models in production, while smaller quantized versions may lose contextual accuracy.

4.3. Llama Firewall

The Llama Firewall architecture represents a robust, open-source orchestration framework that integrates multiple independent scanners into a unified defense pipeline. In its default architecture (Figure 3), the system intercepts inputs for adversarial attack and PII detection, while applying code checks and PII detection on the results. Uniquely, it incorporates a Reasoning Auditor (AlignmentCheck) that monitors the Core LLM's chain-of-thought in real-time, especially when interacting with external tools like web search (Tavily) [Chennabasappa et al. 2025]. A customized architecture can significantly expand this by enabling explicit Output Guardrails (Figure 3, detailed in the cross-hatched pink blocks), which route the Core LLM's responses through customized flows combining Llama Guard (Content Safety), CodeShield, and regex-based PII detection. Llama Firewall is a highly modular defense-in-depth

approach that effectively neutralizes complex agentic threats. The downside is the increased operational overhead and latency; running multiple sophisticated scanners concurrently (especially the Reasoning Auditor) requires substantial computing power and processing time.

4.4. NVIDIA NeMo Guardrails

The NVIDIA NeMo Guardrails architecture (Figure 4) introduces a programmable dialogue management approach. It first screens inputs through structural and semantic barriers (PII, Content, and Topic Safety). Then, using a vector database and the Colang language, it dynamically matches user intents to pre-defined conversational flows [Rebedea et al. 2023], while similarly guarding the final output with hallucination and PII detectors. NeMo Guardrails effectively reduces factual hallucinations by deterministically steering conversations and anchoring responses in trusted data (RAG). However, its Colang syntax imposes a steep learning curve, and binary topic classifications can restrict the fluidity of complex interactions [Banwasi, Friedman and Khanzadeh 2024].

4.5. Amazon Bedrock Guardrails: managed cloud infrastructure

The Amazon Bedrock Guardrails architecture provides a fully managed, declarative cloud infrastructure. In Figure 5, it establishes a sequential pipeline, where the Input Guardrails process prompt attacks, content filters, denied topics, profanities, custom words, PII, and regex validations and the Output Guardrails mirror this exhaustive pipeline, adding relevance and contextual grounding checks to the generated responses. The primary positives of this architecture are its usability, centralized administration, and native integration into the AWS ecosystem without requiring custom code. However, its drawbacks include vendor lock-in, limited transparency of its moderation models, and reduced control over excessive token generation.

4.6. Comparative analysis among architectures

The comparison (Table 1) demonstrates the trade-off between security, latency, and utility in guardrail selection. While open-source frameworks (Llama Firewall, NeMo) offer granular defense-in-depth and managed services (Amazon Bedrock) favor rapid compliance, effective enterprise protection ultimately demands a socio-technical approach, pairing lightweight filters with robust semantic auditors.

Table 1. Comparative Framework of Guardrail Architectures.

Architecture / Criterion	Prompt Guard	Llama Guard	Llama Firewall	NVIDIA NeMo Guardrails	Amazon Bedrock Guardrails
Interventionstage and approach	Input-layer structural filtering.	Semantic validation of prompts and outputs.	Monitors input, output, and tool interactions.	Rule-based orchestration with declarative flows.	Managed input/output filtering and policy enforcement.
Core mechanisms	BERT/DeBERTa classifier for prompt injection and jailbreak detection.	Risk-based semantic evaluator (MLCommons taxonomy); multimodal support.	Combines Prompt Guard, Llama Guard, regex, and code validation for agent monitoring.	Uses Colang, RAG grounding, and intent classification for controlled LLM behavior.	Sequential checks for prompts, PII, toxicity, grounding, and contextual safety.
Key Features	Fast, lightweight, resistant to token obfuscation.	Strong semantic understanding and multimodal moderation.	Highly customizable; real-time agent security monitoring.	Reduces hallucinations with deterministic workflows and trusted data grounding.	Easy deployment, centralized management, seamless AWS integration.
Limitations	Limited context window and multilingual support.	Higher computational cost; vulnerable to adversarial attacks.	Added latency and dependency on external integrations.	Steeper learning curve and reduced conversational flexibility.	Limited customization and lower transparency in moderation logic.
OWASP Threats potentially mitigated	LLM01 Prompt Injection, Jailbreaks.	LLM01 Prompt Injection, LLM05 Improper Output Handling.	LLM01 Prompt Injection, LLM05 Improper Output Handling, LLM06 Excessive Agency	LLM06 Excessive Agency, LLM09 Misinformation.	LLM01 Prompt Injection, LLM02 Sensitive Information Disclosure, LLM09 Misinformation.

This analysis distinguishes between architectural statefulness and semantic understanding. While guardrails may demonstrate deep semantic awareness, stateless operation leaves systems vulnerable to multi-turn exploits like the 'Crescendo' attack [Russinovich, Salem and Ronen 2024]. Consequently, addressing both semantic depth and stateful, multi-turn validation is vital for a comprehensive view of runtime security.

5. Concluding remarks

When comparing these architectures, a clear distinction emerges between flexible, open-source orchestrators (like Llama Firewall and NeMo Guardrails) and managed, proprietary services (like Amazon Bedrock). While modular frameworks provide the defense-in-depth and granular control required for complex or highly regulated environments, managed services offer rapid deployment and standardized compliance at the cost of architectural rigidity. Since chaining multiple verifications can degrade the user experience, an effective safety architecture must strategically balance lightweight filters (e.g., Prompt Guard, Regex) for early-stage rejection with heavier semantic and reasoning auditors (e.g., Llama Guard, AlignmentCheck) for complex, high-risk interactions. Selecting the appropriate guardrail architecture depends heavily on the organization's specific operational requirements, balancing the flexibility and granular control of open-source solutions against the rapid deployment and centralized governance of proprietary ecosystems. Although this study provides a foundational qualitative systematization, it is currently limited by the absence of a quantitative empirical protocol. Future research and practical implementations must prioritize adversarial testing using structured datasets (e.g., direct and indirect prompt injections), measuring specific evaluation metrics such as attack detection rate, latency overhead, false positive/negative rates, and monetary cost. Structured benchmarking and the continuous refinement of evaluation metrics will be essential to assess the true resilience of these barriers against sophisticated attacks.

Acknowledgements

This work is supported by the Ministry of Science, Technology and Innovation (MCTI), with financial resources from the National Fund for Scientific and Technological Development (FNDCT) administered by the Brazilian Funding Authority for Studies and Projects (FINEP), specifically within the scope of the INSPIRE projects - National Data Infrastructure, Core, and AI Solutions for Public Service (Contract 01.25.0728.00, Reference 2315/25), executed in partnership with the Ministry of Management and Innovation (MGI). This work exclusively reflects the views of the authors; the funding agencies are not responsible for the use of the information presented herein. GenAI tools, NotebookLM and Gemini (Google), were used to support translation, text revision, and bibliography formatting in this work.

References

Alzahrani, A. (2026) “PromptGuard a structured framework for injection resilient language models”, Scientific Reports, 16, 1277, doi:10.1038/s41598-025-31086-y.

Amazon Web Services (2025) “Guardrails for Amazon Bedrock”, available at: https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails.html.

Azambuja, A. J. et al. (2026) “Evaluation of NeMo Guardrails as a Firewall for User–LLM Interaction”, Future Internet, 18(5), article number 252, doi:10.3390/fi18050252.

Banwasi, A., Friedman, S. M. and Khanzadeh, M. (2024) “A Comparative Analysis of Guardrail Frameworks for Large Language Models and Enhancement with Ensemble Techniques”, In: New York: Columbia Univ. School of Eng. and Applied Science.

Chao, P. et al. (2024) "JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models", In: Proceedings of the 38th Conference on NeurIPS 2024, doi: https://dl.acm.org/doi/10.5555/3737916.3739661

Chennabasappa, S. et al. (2025) “LlamaFirewall: an open source guardrail system for building secure AI agents”, Meta AI, available at: https://ai.meta.com/research/publications/llamafirewall-an-open-source-guardrail-system-for-building-secure-ai-agents/.

Devino, M., Ju, E. and Caldeira Junior, P. M. (2025) “Designing and implementing LLM guardrails components in production environments”, In: Proceedings of the 2025 IEEE/ACM International Conference on AI Engineering – Software Engineering for AI, pages 12–17, doi:10.1109/CAIN66642.2025.00010.

Dong, Y. et al. (2024) “Position: building guardrails for large language models requires systematic design”, In: Proc. of the 41st ICML'24, article number 451, 20 pages, JMLR.org, doi:10.5555/3692070.3692521.

Dong, Y. et al. (2025) “Safeguarding large language models: a survey”, Artificial Intelligence Review, 58(12), article number 382, doi:10.1007/s10462-025-11389-2.

Inan, H. et al. (2023) “Llama Guard: LLM-based input-output safeguard for human-AI conversations”, Meta AI, available at: https://ai.meta.com/research/publications/llama-guard-llm-based-input-output-safeguard-for-human-ai-conversations/.

Leon, J. et al. (2024) "Garak: A Framework for Security Probing Large Language Models", doi: https://doi.org/10.48550/arXiv.2406.11036

Malik, V. et al. (2025) “AI-Native LLM Security: Threats, Defenses, and Best Practices for Building Safe and Trustworthy AI”, Packt Publishing, Birmingham.

Mazeika, M. et al. (2024) "HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Alignment", In: Proc. of the 41st ICML'24, doi: https://dl.acm.org/doi/abs/10.5555/3692070.3693501.

MITRE ATLAS (2025) “Generative AI Guardrails - Mitigation AML.M0020”, available at: https://atlas.mitre.org/mitigations/AML.M0020.

NIST (2023) "Artificial Intelligence Risk Management Framework (AI RMF 1.0)", NIST Trustworthy and Responsible AI, NIST AI 100-1.

NIST (2025) “Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations”, NIST Trustworthy and Responsible AI Report, NIST AI 100-2, doi:10.6028/NIST.AI.100-2.

OWASP (2025) “OWASP Top 10 for Large Language Model Applications”, OWASP Foundation.

OWASP (2026) “AI Security Overview - The AI Exchange Framework”, OWASP Foundation..

Paim, K. O. et al. (2025) “Exploiting latent space discontinuities for building universal LLM jailbreaks and data extraction attacks”, In: Anais do XXV SBSeg, pages 417–431, doi:10.5753/sbseg.2025.11448.

Rebedea, T. et al. (2023) “NeMo Guardrails: a toolkit for controllable and safe LLM applications with programmable rails”, In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Singapore, pages 431–445, doi:10.18653/v1/2023.emnlp-demo.40.

Russinovich, M., Salem, A. and Ronen, R. (2024) "Great, Now Write an Article About That: The Crescendo Multi-Turn LLM Jailbreak Attack", Microsoft, doi: https://doi.org/10.48550/arXiv.2404.01833.

Tripathi, P. et al. (2026) “Guardrail-based approaches for enhancing safety, security, and privacy in large language models”, In: Proc. of the 2026 IC3ECSBHI, pages 1619–1624, doi:10.1109/IC3ECSBHI67834.2026.11469093.

Appendix

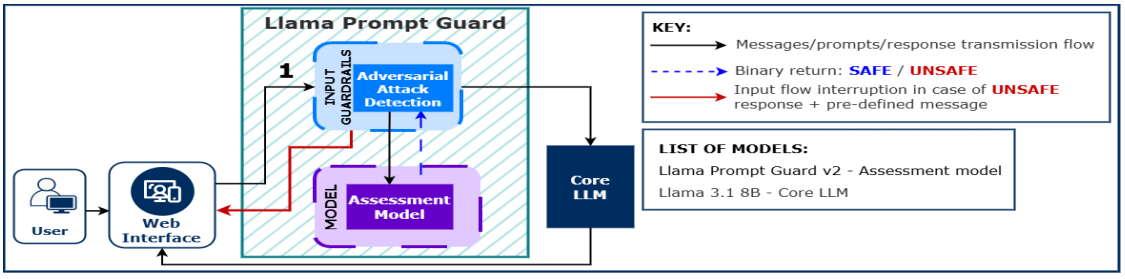

Figure 1. Llama Prompt Guard Deployment Architecture.

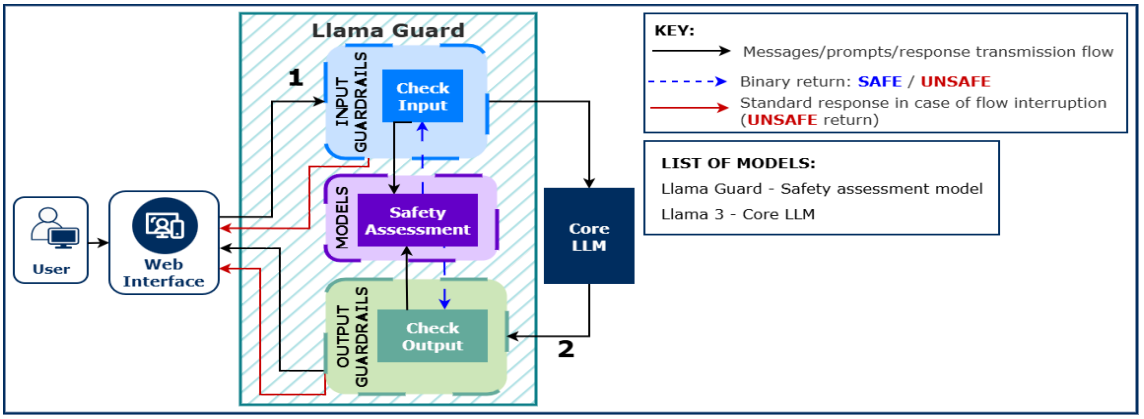

Figure 2. Llama Guard Deployment Architecture.

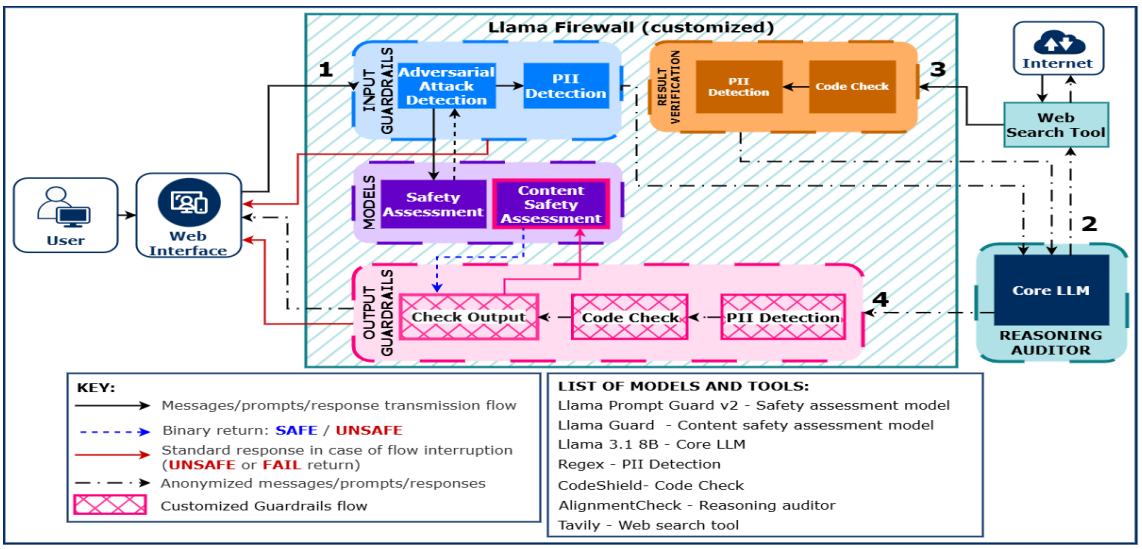

Figure 3. Llama Firewall (customized) Deployment Architecture.

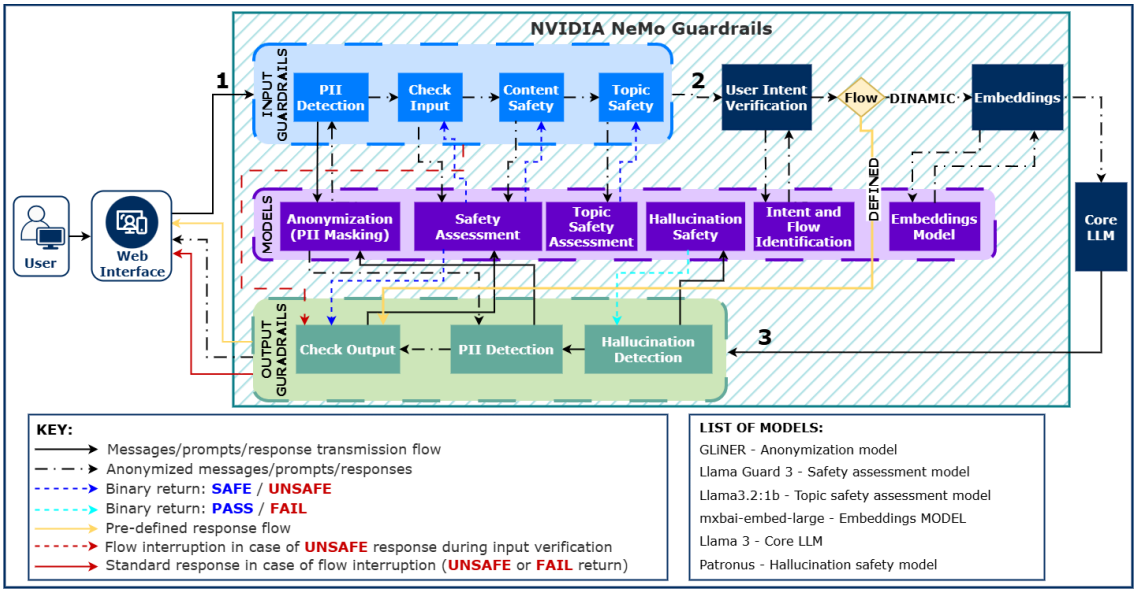

Figure 4. NVIDIA NeMo Guardrails Deployment Architecture.

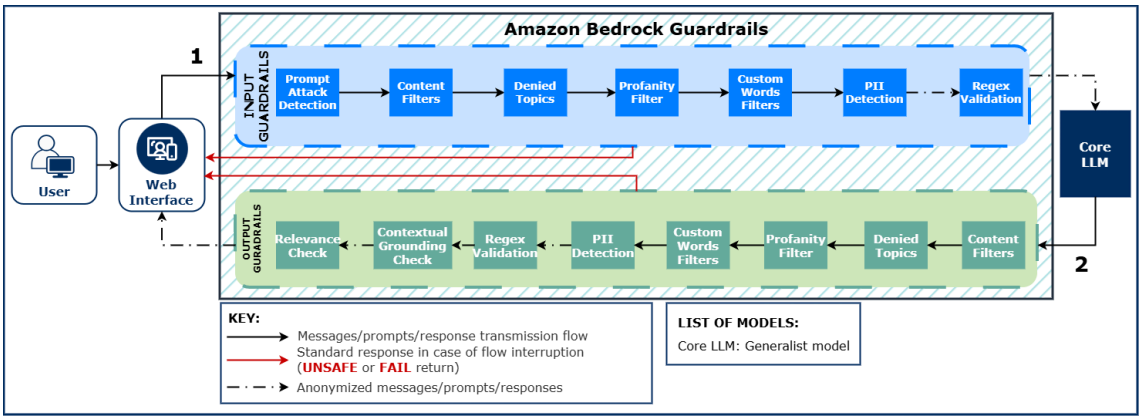

Figure 5. Amazon Bedrock Guardrails Deployment Architecture.