World Digital Technology Academy (WDTA)

Generative AI Application Security Testing and Validation Standard

World Digital Technology Academy Standard

WDTA AI-STR-01

Edition: 2024-04

Discover more WDTA standard and related publications at https://wdtacademy.org/ .

Version History*

Standard ID	Version	Date	Changes
WDTA AI-STR-01	1.0	2024-04	Initial Release
WDTA AI-STR-01	1.01	2024-05	Adjusted author attribution

Foreword

World Digital Technology Academy (WDTA) is dedicated to becoming a trailblazer in global digital tech innovation, aligned with the United Nations framework as an NGO. Upholding its 3S principle—Speed, Safety, Sharing—WDTA strives to accelerate the creation of digital norms, spearhead research, encourage international cooperation, and maintain leadership in technological advancement.

Through collaborative efforts, WDTA is dedicated to advancing digital technology for the betterment of society. The AI STR (Safety, Trust, Responsibility) program, a core part of WDTA's international initiatives, addresses the complex challenges brought about by the proliferation of AI systems. Recognizing the rapid expansion and integration of AI technologies worldwide, AI STR stands at the forefront of global technological progression.

This standard document provides a framework for testing and validating the security of Generative AI applications. The framework covers key areas across the AI application lifecycle, including Base Model Selection, Embedding and Vector Database in the Retrieve Augment Generation design patterns, Prompt Execution/Inference, Agentic Behaviors, Fine-Tuning, Response Handling, and AI Application Runtime Security. The primary objective is to ensure AI applications behave securely and according to their intended design throughout their lifecycle. By providing a set of testing and validation standards and guidelines for each layer of the AI Application Stack, focusing on security and compliance, this document aims to assist developers and organizations in enhancing the security and reliability of their AI applications built using LLMs, mitigating potential security risks, improving overall quality, and promoting responsible development and deployment of AI technologies.

AI STR program represents a paradigm shift in how we approach the development and deployment of AI technologies. Championing safety, trust, and responsibility in AI systems, lays the groundwork for a more ethical, secure, and equitable digital future, where AI technologies serve as enablers of progress rather than as sources of uncertainty and harm. Generative AI Application Security Testing and Validation Standard is one of the AI STR standards.

Founding Chairman of WDTA

Executive Chairman of WDTA

Acknowledgments

Co-Chair of WDTA AI STR Working Group

Ken Huang *(CSA GCR)*

Nick Hamilton *(OpenAI)*

Josiah Burke *(Anthorphic)*

Lead Authors

Ken Huang *(CSA GCR)*

Heather Frase *(Georgetown University)*

Jerry Huang *(Kleiner Perkins)*

Leon Derczynski *(Nvidia)*

Krystal (A) Jackson *(University of California, Berkeley)*

Patricia Thaine *(Private AI)*

Govindaraj Palanisamy *(Global Payments Inc)*

Vishwas Manral *(Precize.ai)*

Qing Hu *(Meta)*

Ads Dawson *(OWASP® Foundation)*

Amit Elazari *(OpenPolicy)*

Apostol Vassilev *(National Institute of Standards and Technology)*

Bo Li *(University of Chicago)*

Reviewers

Cari Miller *(Center for Inclusive Change)*

Daniel Altman *(Google)*

Dawn Song *(University of California, Berkeley)*

Gene Shi *(Learning-Genie)*

Jianling GUO *(Baidu)*

Jing HUANG *(iFLYTEK)*

John Sotiropoulos *(Kainos)*

Josiah Burke *(Anthropic)*

Lars Ruddigkeit *(Microsoft)*

Guanchen LIN *(Ant Group)*

Melan XU *(World Digital Technology Academy)*

Nathan VanHoudnos *(Carnegie Mellon University)*

Rob van der Veer *(Software Improvement Group)*

Sandy Dunn *(BreachQuest, acquired by Resilience)*

Seyi Feyisetan *(Amazon)*

Yushi SHEN *(NovNet Computing System Tech Co., Ltd.)*

Song GUO *(The Hong Kong University of Science and Technology)*

Steve Wilson *(Exabeam)*

Swapnil Modal *(Meta)*

Tal Shapira *(Reco AI)*

Anyu WANG *(OPPO)*

Wicky WANG *(ISACA)*

Yongxia WANG *(Tencent)*

Table of Contents

Generative AI Application Security Testing and Validation Standard

1. Scope

The Generative AI Application Security Testing and Validation Standard document outlines a comprehensive framework to test or validate the security of downstream AI applications, particularly those built using Large Language Models (LLMs). It defines the scope of testing and validation across various layers of the AI Application Stack (Figure 1). Integrating a generative GenAI model into a larger AI-enabled system or downstream application can introduce security issues. Thus, all downstream AI applications need security testing and standards validation, even if the base GenAI model was thoroughly tested before integration into the downstream application.

While this document serves as an initial version, its primary emphasis in this iteration is on LLM. However, it's important to note that the scope extends to GenAI. In subsequent versions of this document, there will be opportunities to incorporate multi-modal and expansive GenAI models

AI security testing and validation work together to ensure AI Applications behave securely and as intended. Robust methodologies should be employed through development life cycles where feasible, using techniques like prompt injection, scanning, and red teaming exercises to identify issues proactively. However, testing alone has limitations, especially with third-party components where testing may not be possible or limited. In this situation, engaging external experts or organizations specializing in auditing AI governance, processes, and procedures is extremely important to validate the security of third-party components. Thoroughly auditing AI applications to check conformance to security standards across all lifecycle deployment environments is critical.

A thorough examination of the downstream AI application ensures adherence to security standards, even when model-level assessments are inadequate. An integrated assurance approach with strong testing practices plus ongoing validation of policies, processes, and performance provides assurance for responsible AI outcomes as systems continue autonomous learning. Together, they provide information about a system’s strengths and weaknesses, inform appropriate versus inappropriate end-use applications, and assist with risk mitigation.

This specification covers the security testing of downstream applications built on top of base LLM models but does not detail security test specifications for the base LLM models themselves. A separate document to be published in the future will cover security testing specifications specifically for base LLM models.

This specification addresses the below key areas:

Base Model Selection: Candidate models for a downstream AI application should be examined before selection. This section covers verifying the base model's compliance, appropriate data usage, and API security. The document provides guidance to ensure that the chosen model aligns with legal, ethical, and operational standards, a crucial step in ensuring the AI application's security. The scope includes both open-source and closed-source model selection.

Embedding and Vector Database: These are critical components in most downstream AI applications, storing and retrieving chunks of language data. This document outlines procedures for testing data integrity, quality, and anonymization processes to protect user privacy and comply with regulations. The specification provides guidelines for testing the confidentiality, integrity, and availability of the vector database.

Prompt and Knowledge Retrieval with Retrieval-Augmented Generation (RAG): RAG can significantly improve the factual accuracy and reliability of generative AI applications, such as large language models. It achieves this by dynamically incorporating relevant, domain-specific knowledge extracted from external sources in real-time during the text generation. This section guides the construction of effective prompts, the creation and use of prompt templates, and the integration of external APIs. It also covers testing the retrieval process from the vector database, ensuring that the AI Application can accurately access and utilize relevant information.

Prompt Execution/Inference: The document details testing procedures for LLM APIs in the Prompt Execution/Inference layer, including tests for caching mechanisms and validation processes to optimize performance and accuracy. This layer also includes tests for checking prompts and ensuring LLMs are not being used to perform unauthorized actions, which are not allowed for the use case.

Agentic Behaviors: These are advanced LLM application capabilities. The specification outlines tests for prompt interpretation, memory utilization, knowledge application, planning, and action initiation. This includes testing the tools integrated into the AI Application to enhance its capabilities securely.

Fine-Tuning: The GenAI model is often fine-tuned for a specific downstream AI application. This section encompasses tests for data privacy, re-evaluation of the base model selection, and model deployment, ensuring the AI's continual improvement and relevance.

Response Handling: This section involves testing for fact-checking of the AI's responses, relevance, toxicity, and ethical considerations to maintain the Trustworthy and security of the AI's interactions.

AI Application Runtime Security: Runtime security involves the continuous, real-time monitoring of the AI Application. It covers data protection, model security, infrastructure security, and compliance with audit trails. This ensures a comprehensive security approach, safeguarding the AI Application against various threats and vulnerabilities throughout its lifecycle.

Overall, the Generative AI Application Security Testing and Validation Standard document provides a detailed and structured approach to testing each layer of the AI Application Stack, ensuring that all aspects of an AI Application are rigorously evaluated for security and compliance.

Figure 1: AI Application Stack

2. Intended Audience

The intended audience for this document is professionals and stakeholders involved in ensuring the security and integrity of Generative AI Applications. This document is particularly relevant to:

AI Security Engineers and Analysts: These individuals are primarily responsible for implementing and maintaining the security measures outlined in the specification. They assess the AI application for threats, design security architectures, and monitor systems to prevent, detect, and respond to security incidents. These engineers also look at bias and threats.

AI Developers, MLOps and AI Engineers: These are the people who build, maintain, and automate the workflow of AI applications. They use the security specification to understand and integrate security best practices into the application development lifecycle.

Compliance Officers and Regulatory Experts: Professionals responsible for ensuring that AI applications comply with the constantly evolving legal and regulatory standards use the specification to guide compliance efforts, particularly in industries with strict data protection and privacy regulations.

Data Protection Officers: These officers ensure that AI applications handle data securely and in compliance with data protection laws and policies. The security specification provides them with guidelines for proper data management and protection strategies.

IT and Network Administrators: These administrators are responsible for the underlying infrastructure of AI Applications. These professionals will use the security specification to secure networks, servers, and other components against vulnerabilities that bad-actors could exploit in AI-related processes.

Risk Management Professionals: These individuals assess and manage risks associated with AI applications. The security specification aids them in identifying potential security risks and implementing measures to mitigate them.

Ethics Review Boards: Boards tasked with overseeing the ethical use of AI rely on security specifications to ensure that AI applications are ethically sound and secure against misuse or harmful manipulation.

Project Managers and Product Owners in AI Projects: These stakeholders ensure that the AI projects are delivered securely and efficiently. The security specification guides them in setting security-related project goals and benchmarks.

Third-Party or External Security Auditors and Consultants: These experts provide an external review of the AI application's security posture. They use the specification as a benchmark to evaluate the application's adherence to security best practices.

End-Users or Business Stakeholders: While not directly involved in implementing security, end-users or business stakeholders of AI applications have a vested interest in the security of these systems. Understanding the security specifications can help them gauge the reliability and Trustworthy of AI applications.

Each of these groups plays a pivotal role in ensuring the security of AI applications, from development through deployment and operation, using the Generative AI Application Security Testing and Validation Standard as a guiding framework.

3. Normative References

The references listed below are essential for applying and understanding this document. They provide foundational theories, practices, legal frameworks, and guidelines critical to the secure and responsible development and deployment of AI applications:

- WDTA Declaration on Global AI Governance
- Generative AI security: Theories and Practices
- Applying the AIS Domain of the CCM to Generative AI
- EU AI Act
- Biden Executive Order on Safe, Secure, and Trustworthy Artificial Intelligence
- NIST Trustworthy and Responsible AI NIST AI 100-2e2023
- NIST Artificial Intelligence Risk Management Framework (AI RMF 1.0)
- Chinese Generative AI Regulation
- Chinese Approach to AI Regulations
- Confidential Computing Consortium
- OWASP Top 10 for LLM Applications
- CSA Cloud Controls Matrix (CCM v4)
- MITRE ATLAS™ (Adversarial Threat Landscape for Artificial-Intelligence Systems)
- NIST Secure Software Development Framework(SSDF)
- OWASP Top 10 API Security Risks
- Mitigating Security Risks in Retrieval Augmented Generation (RAG) LLM Applications
- OWASP AI Exchange

4. Terms and Definitions

For this document, the following terms and definitions apply:

Agentic Behaviors: The capacity of LLM Applications to exhibit agency through actions like memory utilization, knowledge application, planning, and executing actions based on prompts.

AI Application Runtime Security: The comprehensive security measures implemented to protect the AI Application during operations. It encompasses data protection, model security, and infrastructure security.

AI Governance: The framework, requirements, oversight, and accountability towards AI risk. These structures enable mapping risk to organizational context (like teams, identities, priorities, and rules) while providing attestations, traceability, metrics, and oversight.

AI Response Handling: Involves processing and evaluating the AI's responses for accuracy, relevance, non-toxicity, privacy, confidentiality, and ethical alignment.

API Security Check: Verifying the security measures of the APIs interfacing with the model, such as authentication, authorization, and data encryption to prevent unauthorized access and data breaches.

Base Language Model: A base model (sometimes called a foundation model) is a large language model that has already been trained and fine-tuned for general capabilities by its original model builders using techniques like Reinforcement Learning from Human Feedback (RLHF). These base models (e.g, OpenAI’s GPT-4, Anthropic’s Claude 3, Google’s Gemini 1.5, Cohere Command, Amazon Titan, or Meta's open-sourced LLaMA2) are a strong foundation for further task-specific customization. Typically, developers tune the base model outputs to display broad linguistic proficiency and adaptability to specialized use cases in a downstream application. Engineers and companies then take these base models as a starting point for efficiently developing and deploying customized AI solutions tailored to their precise needs and applications. The base model eliminates the need to train a full model from scratch, providing closed and open-sourced AI application launching pads.

Base Model Selection: Choosing an appropriate base model with consideration of AI security. Selection involves evaluating factors like performance benchmarks, training data quality, potential biases, security procedures, potential harmful outputs, intended use cases, and regulatory compliance requirements. Sound model provenance, transparency, auditing of data/training approaches, cross-functional review processes, and adherence to codes of conduct are vital considerations to uphold security, compliance, and ethics standards in deploying LLMs responsibly.

Caching: Techniques used to store inferred outputs from AI models to avoid repeated computations during inference. Since deep neural network models can be computationally expensive to run, caching their outputs allows for quicker response times during live requests. Typical solutions include caching question-answer pairs for chatbots, classifications for computer vision models, or generated text for large language models.

Caching Validation: Checking the cached outputs from AI Applications for accuracy, relevance, and security before returning them to users. This can involve confidence checks, semantic analysis, input blocking for sensitive topics, or human confirmation. Validation works with caching to ensure reliable and secure real-time AI while benefiting from faster performance. Applying thoughtful validation methods is essential when leveraging caching.

Closed-Source Model: A model whose weights, inference code, and training data inventory are not publicly available.

Data Cleaning and Anonymization: The removal of inaccuracies and inconsistencies from data and anonymizing personal or sensitive information to maintain privacy and compliance.

Data Usage Check: Ensuring the data used for training and operating the model is appropriate, ethically sourced, and compliant with data protection regulations.

External API Integration: Incorporating external APIs into the LLM application to enhance functionality, such as accessing additional data sources or performing specialized computations.

Fine Tuning: The process of adjusting a based model for specific tasks or datasets to improve performance, relevance, and compliance with data privacy (with futuristic techniques like fine-tuning for unlearning sensitive data).

LLM (Large Language Model): A large language model (LLM) is a neural network trained on a huge text corpus to generate intelligent text by predicting the next word or token, allowing open-ended text generation applications like conversational AI chatbots.

Model Compliance Check: Assessing whether the chosen model adheres to legal, regulatory, and ethical standards. This includes considerations like data privacy laws and bias minimization. Keep in mind that compliance will change over time and do not assume that it is always a given. Also do not infer compliance of one vendor to another. Models itself are rarely certified but their hosted solutions.

Model Registry: A database, repository, or system for storing, versioning, and cataloging Machine Learning/AI models and associated metadata (e.g., model cards). A Model Garden is a curated version of the Registry with the provider's curated models. It normally requires a meta-data lineage between the model and the used training data and inference data points.

Prompt Construction and Templates: The creation of effective and secure prompts for the LLM and developing templates to standardize and streamline prompt generation.

Prompt Processing: The process where the LLM interprets and processes a prompt to generate a response. This process involves understanding the prompt, accessing relevant knowledge, and generating output based on prompts and contextual knowledge.

RAG, or Retrieval-Augmented Generation: Retrieval-augmented generation improves the factual accuracy of generative AI applications, like large language models, by augmenting them with relevant knowledge extracted in real-time from a vector database. During inference, a retriever module first uses the generator's internal state vector to query a vector database storing external knowledge (texts, images, etc). The retrieved vectors most relevant to the generation context are then crossed with the internal state to produce the next generated outputs. This process dynamically grounds the model's generation closer to reality, correcting false assumptions and reducing hallucinated content. This scalable retrieval infrastructure provides the generator with a continuous supply of relevant external data during open-ended inference. This retrieval-augmented generation approach counteracts the generator's knowledge limitations and tendency to fabricate information, thereby improving factual consistency, security, and trust in open-domain generative AI applications.

Vector Database: Vector databases serve as ground truth, help extend knowledge beyond training time to run time, and reduce hallucination in generative AI models. They allow storing large volumes of real-world data (images, texts, molecular structures, etc.) as vector representations that capture semantic concepts and features. These vector datasets then act as a reference for generative models during inference to align their outputs closer to reality and avoid fabricating false details (hallucination). Retrieving the nearest vector matches from the database for generative model outputs provides an automated way to detect and filter hallucinated content. This dataset conditioning is crucial for security-critical applications of generative AI like drug discovery and content creation. Optimized vector search and scalability make databases like PG Vector, Milvus, Weaviate, and Pinecone well-suited to enable such large-scale hallucination detection for deployed generative AI applications in the real world.

5. AI Applications Security and Validation Standards

Ensuring the security and integrity of AI applications requires structured and rigorous testing across all components in the AI application stack. A comprehensive testing regime verifies that every aspect (from foundational model selection through to runtime security) of a downstream AI application functions securely, as intended, and without vulnerabilities. Meticulous testing specifications set clear requirements, methods, and expected results, enabling transparent evaluations. This section provides detailed testing standards for each layer of the AI application architecture, as referenced in Figure 1: AI Application Stack.

5.1 Base Model Selection Testing Standards

The Base Model Selection is a crucial aspect of ensuring the security and compliance of an AI application. Selection involves distinct considerations for both open-source and closed-source models, recognizing that while closed-source models may have more readily available compliance documentation, open-source models might lack established compliance documents. Both, however, require thorough testing and validation.

It's important to note that testing and validation of a base model is a continuous process, especially if the upstream base model changes. As the base model evolves and is updated, it is essential to re-validate the model to ensure that it still meets the required security and compliance standards. This ongoing validation process helps maintain the integrity and reliability of the AI application, even as the underlying base model undergoes modifications or improvements.

5.1.1 Model Compliance and Context Testing

Checking model compliance involves different methodologies for each model type, considering their unique characteristics and the availability of information.

Requirement: Ensure the AI-based model, whether open-source or closed-source, complies with legal, regulatory, security and ethical standards.

Method: For closed-source models, conduct a detailed review of available compliance documentation provided by the vendor against relevant laws, industry regulations, and ethical guidelines. To ensure compliance, the model's quality of the training data (fit for purpose), output behavior, operational parameters, and community feedback should be reviewed and evaluated. Permission and access to closed models may limit this evaluation. For all models, tools such as Model Cards [1] and Data Statements[2] provide baselines for model and data documentation. This may include consultations with legal and industry experts to interpret compliance in areas lacking formal documentation. Test if the model's accuracy, relevance, consistency, and performance on a specific task meets predetermined requirements. Benchmark the model using preset scenarios and datasets to measure its performance and the quality of its output.

There are resources for testing, but their utility may vary and degrade over time. There are many publicly available prompt security testing results for both closed- and open-sourced models. Because they use different test datasets, these security assessment efforts can produce different results for the same model. Still, consulting multiple of publically available results can point out which types of harmful behavior an LLM is more likely to show. Benchmarks will also change over time as new failure models and attacks are discovered; consider and report the age/version of benchmarks used to assess the model.

Additionally, identify and list known vulnerabilities from sources such as the MITRE Atlas(™)[3], AVID[4], the AI Risk Database[5] and the AI Incident Database[6]

Expected Results: The base model fully meets all legal, regulatory, security, and ethical requirements, regardless of its source. Any areas of non-compliance are clearly identified for closed-source models and rigorously inferred and documented for open-source models.

2. Checking the context metadata, lineage of training and fine-tuning of each model exists in the model card. Model cards provide details of models used by each application.

[1] https://dl.acm.org/doi/10.1145/3287560.3287596

[2] https://techpolicylab.uw.edu/data-statements/

[3] https://atlas.mitre.org/

[4]https://avidml .org/

[5] https://airisk.io/

[6] https://incidentdatabase.ai/

Requirement: Ensure the AI model, whether open-source or closed-source, has the model card detailing the source of the model, data sensitivity, training datasets, and model custodian. This information should be accessible directly or indirectly through the model card.

Method: Conduct a detailed review of available model cards for all models, including hosted models. Make sure the model card has details of the model lineage and ownership. The model card should be able to provide the data sets for both training and fine-tuning (if applicable). For closed-source models, get details of the model card from the provider if known.

Maintain and manage model cards as model stewards, ownership, datasets as these change with time.

Expected Results: The base model, irrespective of its source, has complete metadata of the model applications.

5.1.2 Data Usage Check Testing

1. Protect user privacy in interactions with AI Applications.

Requirement: Safeguarding User Privacy in Prompts

Method: Implement data anonymization or pseudo-anonymization techniques on user prompts for sensitive and personal data. Conduct regular audits to ensure that personal identifiers are effectively obscured. Also, make sure prompts and output are not stored more than what is specified by policy. This may limit both the content stored and the length of storage time. Conduct adversarial testing to check for data leaks, using methods (e.g., continuation testing, cloze task testing, etc.) to fill in gaps.

Expected Results: User prompts are processed without revealing personal identities, ensuring privacy and compliance with data protection laws. If needed, there are controls to prevent sensitive information from being collected or sent to an API.

2. Maintain ethical integrity and legal compliance in the handling and use of data.

Requirement: Ethical and Legal Use of Data

Method: Establish guidelines for data usage that align with ethical standards and legal requirements. Perform routine compliance checks and audits to monitor adherence to these guidelines. Continuously ensure that model documentation is sufficient to assess compliance.

Expected Results: Data from user prompts, fine tuning training data, and the vector database is used ethically and legally, with no instances of misuse or mishandling.

3. Ensure adherence to international and local data protection regulations.

Requirement: Adherence to Data Protection Regulations

Method: Implement procedures for obtaining user consent, ensuring data transparency, and providing users control over their data. Data minimization collects only necessary personal data and avoids excessive collection. Minimize the use and storage period of personal data through technical means. Take differentiated privacy protection measures based on the sensitivity of the data. Regularly train staff on data protection laws and conduct compliance audits. Implement the process and procedures of responding to access to information requests and requests to be forgotten.

Expected Results: Full compliance with data protection laws like GDPR or CCPA, demonstrated through audit results and user feedback.

4. Data provenance process using data lineage and metadata.

Requirement: Ensure that datasets used to train an ML model, whether open-source or closed-source, have the data card and that the source of the data, data sensitivity, compliance regime, and data custodian of the dataset are accessible directly or indirectly through the data card.

Method: Conduct a review of data cards and datasets. Verify that each datasets, especially those that have been used to train or fine-tune models, has a data card. Make sure the data card has the details of the dataset lineage and ownership.

Maintain and manage data cards. The collection and content in data cards will change as data stewards, ownership, and datasets change with time.

Expected Results: The base model, irrespective of its source, has complete metadata of the model applications.

5. Obtain and maintain Data Usage Agreement between AI application developers and base model providers

Requirement: Data Usage Agreement between AI application developers and base model providers. This includes the following sub-test or validation

Sub Requirement: Identification of Parties and Scope

Method: Review the agreement to verify that all parties are correctly identified and the scope of the agreement, including the specific base models or datasets, is clearly defined.

Expected Result: The agreement accurately identifies all parties and outlines the scope, with less ambiguity about the models or datasets involved.

Sub Requirement: Usage Rights and Limitations

Method: Conduct a detailed analysis of the agreement to ensure that the usage rights and limitations, including any restrictions on modifications and redistribution, are explicitly stated and understood.

Expected Result: Clear understanding and documentation of the usage rights, ensuring that the license type (exclusive or non-exclusive) and any limitations are well-defined and adhered to.

Sub Requirement: Data Handling and Compliance

Method: Review processes for handling user prompts, fine turing training data and vector database content. Check for compliance with data protection laws and procedures for data anonymization, data residency, and security.

Expected Result: Data handling methods comply fully with the agreement and legal standards, with secure and compliant data management practices in place.

Sub Requirement: Intellectual Property Rights

Method: Verify that the agreement clearly outlines the intellectual property rights concerning the base models, the fine-tuned models, the input data, fine-tuning data, and output data. Check for compliance in practice. Also, look at any indemnification clauses that the model provider entitles its user. Establish legal risks by reviewing the provider's indemnification clauses.

Expected Result: Intellectual property rights are respected, and clear guidelines are provided for using, modifying, and redistributing model outputs. Any indemnification clauses and conditions are understood based on the company's needs.

Sub Requirement: Confidentiality and Non-Disclosure

Method: Evaluate the enforcement of confidentiality and non-disclosure provisions, particularly regarding sensitive data.

Expected Result: Strong adherence to confidentiality obligations, with all sensitive information protected according to the organization's policies on the treatment of confidential company information and material nonpublic information.

Sub Requirement: Liability and Warranties

Method: Review the organization's approach to addressing liabilities and warranties related to AI systems. Evaluate the extent to which the organization makes good faith efforts to understand and implement relevant terms, respond to potential model failures or data breaches, and adhere to applicable standards and regulations in this area.

Expected Result: The organization demonstrates a commitment to properly managing liabilities and honoring warranties to the best of its abilities, with policies and processes in place that aim to handle any issues that may arise. Efforts are made to comply with relevant industry standards, best practices, and legal requirements regarding the allocation of responsibility and remediation of failures or breaches, recognizing that perfect execution may not always be possible.

Sub Requirement: Termination and Renewal Terms

Method: Review the agreement for clarity on termination and renewal conditions, including notice periods and termination procedures.

Expected Result: Well-defined terms for termination and renewal, with a straightforward process for either party to follow.

Sub Requirement: Dispute Resolution

Method: Examine the dispute resolution clause and assess the preparedness to engage in the outlined process in case of disagreements or breaches.

Expected Result: Effective mechanisms in place for dispute resolution, aligning with the agreement's terms.

Sub Requirement: Governing Law

Method: Ensure that the governing law and jurisdiction are clearly stated and understood, and check for any potential conflicts with local laws where the AI applications are developed or used.

Expected Result: Clear understanding and compliance with the specified governing law, with no legal conflicts.

Sub Requirement: Signatures

Method: Verify that authorized representatives of both parties sign the agreement.

Expected Result: Legally binding agreement with signatures from all relevant parties, ensuring its enforceability.

5.1.3 Base Model Inference API Security Testing

This section outlines specific testing specifications for comprehensively evaluating how the client application integrates with a third-party model inference API. These tests are critical when the application interacts with external APIs, requiring a different approach than traditional API testing. This section focuses on testing from the client application's perspective, which is distinct from the testing done by the API provider, as outlined in Section 5.4.1. This distinction is crucial because we are dealing with a client application that will consume third-party inference APIs. A unique testing methodology tailored for this use case is necessary to ensure secure integration.

To ensure a comprehensive and structured approach to testing the security of a client application integrating with a third-party model inference API, we listed the following testing specifications.

1. Authentication and Authorization:

Requirement: All requests to the API must be authenticated and authorized to ensure that the client has both permission and appropriate access to the requested resource.

Method: Simulate various authentication scenarios to test protocol implementation and key/token management

Expected Results: Clients must include a valid authentication token in the request headers, typically as a Bearer token. The API should respond with appropriate status codes:

- 200 OK if the request is successful.

- 401 Unauthorized if the token is missing, invalid, or expired.
- 403 Forbidden if the authenticated client lacks permissions for the requested action.

Provide clear and concise error messages to indicate the specific authorization issue, such as lack of access or an expired token.

2. Data Encryption:

Requirements: Encryption must be applied across all states: in-transit, at rest, and in-use.

Data in-Transit Sub-Requirements: Use strong encryption protocols such as TLS 1.2 or later for data transmitted over networks to ensure secure confidentiality and integrity. Implement perfect forward secrecy to safeguard past encrypted communications from decryption, even if long-term keys are compromised. Prior to transmission, sensitive data must be encrypted, ensuring authorized decryption upon arrival at the destination.

Methods: Conduct thorough vulnerability assessments and penetration tests on the Base model inference API endpoint to gauge its resilience against potential attacks. Ensure that its cryptographic configurations and encryption protocols are robust and impenetrable. Continuously monitor network traffic to and from the endpoint, verifying the enforcement of stringent encryption standards and the adoption of secure protocols, such as the latest TLS version, with priority given to maintaining perfect forward secrecy.

Expected Results: All transmitted data are securely encrypted using up-to-date and secure protocols adhering to current cryptographic standards. Encryption keys should dynamically change per session to prevent decryption of past sessions if future keys are compromised.

Data at-Rest Sub Requirements: Implement a multi-layered security approach to de-identify sensitive data, including tokenization, anonymization, and pseudonymization. Strong encryption standards like AES-256 or equivalent should be employed to securely store sensitive data, where de-identifying sensitive data or personal information is not feasible. Encryption keys should be stored separately from encrypted data to enhance security. Stringent access controls and effective key management policies must be implemented.

Methods: Validate the compliance requirements based on defined testing procedures.

For tokenization, assess token generation security and randomness. Assess token dictionary access security. Verify logging/auditing of all tokenization operations. Evaluate encryption and access controls for token-data mapping. Test for data leakage risks in tokenized data usage, especially related to frequency attacks.

For anonymization, validate irreversible anonymization and the inability to re-identify data. Check the utility of anonymized data for intended purposes. Conduct risk analysis for potential re-identification on stylistically representative data. Review anonymization techniques used and their effectiveness.

For pseudonymization, ensure pseudonym uniqueness, security, and separation from source data. Analyze re-identification risks considering data correlations. Validate access controls, audit trails, and authorize access.

Expected Results:

Data should be adequately protected in various contexts without compromising its utility for legitimate business processes. The risk of individuals being re-identified from anonymized data should ideally fall between 0.04% and 0.1% or lower, depending on the data sensitivity. Only authorized individuals are expected to be able to access and link tokens or pseudonyms to original data, with a full audit trail.

Re-identification of individuals from anonymized or pseudonymized data must be practically impossible, thus minimizing privacy risks.

Data in-Use Sub-Requirements: Implement encryption for sensitive data being processed in memory. Applications must use secure coding practices to prevent memory dumps and side-channel attacks. Least privilege principles should be adopted to limit access to sensitive data during processing. Confidential Compute hardware or services and Confidential GPUs, which provide hardware root-of-trust as defined by the Confidential Compute Consortium, should be considered.

Methods: Validate memory encryption effectiveness by assessing applications and systems to confirm they encrypt sensitive data effectively while in memory.

Evaluate how applications handle sensitive data in memory, focusing on preventing leaks through memory dumps and protection against side-channel attacks.

Test for vulnerabilities to side-channel attacks, examining how data is processed and stored in memory to identify potential leakage points.

Review user and process access rights to ensure they are minimal and necessary, in line with the principle of least privilege, to reduce the risk of unauthorized access to sensitive data in memory.

Trusted Execution Environments (TEEs) provide a secure execution environment, isolating sensitive data in a protected CPU enclave during processing. Attestation is a critical part of confidential

computing, allowing incremental trust establishment from the root-of-Trust (RoT) to actual trust about a system.

Expected Results: Ensure sensitive data remains encrypted, or pseudonymised, or anonymised and secure when being processed. Only necessary users and processes should have access to sensitive data in use, and such access must be strictly controlled and logged. Confidential computing and TEE validation results are expected to show that the environment is secured and has passed integrity tests.

3. Input Moderation and Data Sanitization:

Requirement: Robust validation and sanitization of data received from the API.

Method: Evaluate the application's ability to handle and sanitize various potential attack vectors, including inputs that fall outside the usual use case parameters. Perform fuzzing testing, which should cover all functional points of the API interface, including various HTTP methods (such as GET, POST, PUT, DELETE, etc.). Perform penetration and security vulnerability testing, such as SQL injection, cross-site scripting (XSS), command injection, overflow errors, etc.

Expected Result: The application effectively filters and sanitizes input, preventing injection, irrelevant inputs and other data manipulation attacks. All prompts identified as malicious are logged and sent for analysis.

4. Error Handling and Logging Security:

Requirement: Secure error handling and logging that does not expose sensitive information.

Method: Trigger error conditions and analyze logs for information leakage. When possible, automatically remove sensitive information before log storage.

Expected Result: Errors are handled securely without leaking sensitive data, and logs are maintained securely.

5. Rate Limiting and Resource Management:

Requirement: Adherence to the API’s rate limits and efficient resource management.

Method: Test the application under varying load conditions to evaluate compliance with rate limits and resource usage.

Expected Result: The application respects rate limits and efficiently manages API resources.

6. Secret Management for API Keys and Credentials:

Requirement: Securely manage API keys and credentials by storing them in a secure vault using secret management approaches. API keys and secrets must be rotated at regular intervals not exceeding 180 days or immediately upon indication of potential compromise. The rotation processes must support secure key revocation and provisioning to minimize attack windows.

Method: Implement a secure vault to store and retrieve API keys and credentials, ensuring they are not exposed or leaked. Observe and validate the secure key management process, including but not limited to key generation, key rotation, disabling older keys, key destruction, and handling key materials securely. Interview responsible personnel and review training materials to confirm awareness and understanding of secure API key rotation and Secrets Management processes across relevant teams. Perform sample tests by attempting to access resources using old/revoked API keys and Secrets and verifying that access is appropriately denied.

Expected Result: API keys and credentials are stored securely, reducing the risk of unauthorized access and data breaches.

7. Dependency and Library Security:

Requirement: Up-to-date and secure libraries and dependencies for API communication.

Method: Perform vulnerability scans to check for outdated and deprecated components.

Expected Result: All components are current and free of known vulnerabilities.

8. Compliance with API Security Policies:

Requirement: Full compliance with the API provider's security policies.

Method: Review and test the application’s data handling and integration against the API’s security protocols.

Expected Result: The application aligns with all specified security guidelines and protocols for the API.

9. Monitoring and Incident Response:

Requirement: Effective monitoring for unusual API activity and a robust incident response plan.

Method: Determine normal API behavior baseline for the use case for both int API input and output, including relevance, the typical request rates, response sizes, and patterns. This baseline helps detect

anomalies, especially those triggered by moderation or use case filtering APIs. Keep detailed logs and use automated tools to analyze them for suspicious activities.

Expected Result: Prompt detection of anomalies and effective execution of the incident response plan.

10. Privacy and Data Protection Compliance:

Requirement: Adherence to data protection laws and privacy-by-design principles.

Method: Audit data handling practices and privacy measures in the application.

Expected Result: The application complies with relevant data protection regulations and effectively protects user privacy.

11. Regular Security Audits:

Requirement: Ongoing security assessments focusing on API interactions.

Method: Conduct periodic security audits to identify and address vulnerabilities.

Expected Result: Continuous identification and timely remediation of security issues related to API interactions.

5.2 Embedding and Vector Database

For the Embedding and Vector Database component of an AI Application, the testing specification can be structured as follows:

5.2.1 Data Cleaning and Anonymization Testing

To ensure the integrity and privacy compliance of data used for creating embeddings by verifying its cleanliness and effective anonymization.

Requirement: Ensure that the data used for creating embeddings is cleaned and anonymized effectively based on use cases, especially for public-facing applications. This can be done using techniques like tokenization.

Method: Implement tests to assess the thoroughness of data cleaning processes, ensuring that irrelevant, redundant, or erroneous data is identified and corrected or removed. Additionally, test the anonymization processes to confirm that personal or sensitive information is effectively obscured or removed, in compliance with privacy standards like GDPR. This might involve reviewing anonymization algorithms, techniques, and their effectiveness in various scenarios.

Expected Results: Data used in the embedding process is clean, relevant, and free from errors. Anonymization processes are effective in protecting personal and sensitive information, rendering it unidentifiable.

5.2.2 Vector Database Security Testing

Enhance the security of the vector database by implementing and verifying advanced encryption, RBAC for data access, robust key management, comprehensive IAM policies, and other critical security measures. The following are the testing specifications:

1. Advanced Encryption Techniques:

Requirement: Evaluate the utilization of advanced encryption techniques, including end-to-end encryption, to secure data at rest and in transit. Consider the use of always encrypted databases and confidential computing to secure data.

Method: Conduct thorough assessments of cryptographic protocols and encryption standards in use, analyzing their effectiveness in protecting data.

Expected Result: Enhanced data security by adopting advanced encryption methods and safeguarding data during transmission, use, and storage.

2. Key Management Lifecycle Testing:

Requirement: Examine the entire lifecycle of encryption keys, from creation to retirement, to ensure adherence to secure key management practices.

Method: Test the processes for key issuance, renewal, revocation, and destruction, assessing their robustness and compliance with key management standards.

Expected Result: A secure key management lifecycle that effectively protects encryption keys, reducing the risk of unauthorized access.

3. Fine-Grained IAM Policy Implementation:

Requirement: Implement and test fine-grained Identity and Access Management (IAM) policies that specify precise permissions for different user roles and scenarios.

Method: Conduct scenario-based testing to validate that each user role can only access data and perform actions according to their defined permissions.

Expected Result: Granular control over access rights, ensuring that users can only perform authorized actions, enhancing data security.

4. Regular Security and Compliance Audits:

Requirement: Conduct frequent and comprehensive security audits that go beyond standard checks, including assessments for compliance with international standards and industry-specific regulations.

Method: Perform in-depth audits, vulnerability assessments, and compliance checks to ensure alignment with relevant security standards and regulations.

Expected Result: Continuous improvement in security posture, with adherence to industry best practices and compliance requirements.

5. Zero Trust Architecture (ZTA) Evaluation:

Requirement: Assess the implementation of a Zero-trust security model, where trust is never implicitly granted and must be continually validated.

Method: Evaluate the deployment of the vector database within a Zero-trust environment, validating that access controls are consistently applied based on identity and context.

Expected Result: Enhanced security through the Zero-trust model, reducing the attack surface and ensuring strict access control.

6. Real-Time Monitoring and Anomaly Detection:

Requirement: Implement and evaluate real-time monitoring systems and anomaly detection algorithms to identify and respond to unusual access patterns or potential security threats in real-time.

Method: Test the effectiveness of real-time monitoring tools and anomaly detection algorithms by simulating security incidents and monitoring their detection and response.

Expected Result: Early detection and rapid response to security threats, minimizing potential damage and data breaches.

7. Disaster Recovery and Backup Testing:

Requirement: Ensure the presence of robust disaster recovery and data backup processes.

Method: Test the disaster recovery and backup systems for their ability to quickly and accurately restore data in the event of a breach or data loss incident.

Expected Result: Reliable and efficient disaster recovery and data backup processes, minimizing data loss and downtime in case of incidents.

8. Role-based Access Control Techniques:

Requirement: Evaluate whether data access in Vector databases aligns with the Role-based access controls.

Method: Conduct thorough assessments of RBAC of data access. Access data using different roles and ensure only the right data is accessible to the right role. Flag any anomalous data access.

Expected Result: Make sure data for Role is not seen by the other when the other role should not have access to the data

5.3 Prompt and Knowledge Retrieval with RAG

The testing specification for the "Prompt and Knowledge Retrieval with RAG (Retrieval-Augmented Generation)" phase of AI applications encompasses the following components:

5.3.1 Prompt Construction Testing

1. To verify that the prompts created for the RAG model effectively convey the intended query or command.

Requirement: Ensure that prompts constructed for the RAG model are effective and accurately represent the intended query or command.

Method: Test the prompt construction process for clarity, relevance, and completeness. This involves evaluating a variety of prompts to ensure they effectively communicate the intended request to the RAG model and that the model's responses align with the prompt's intent. These tests might include user scenario simulations and automated testing for prompt variability.

Demonstrating this requirement may be resource-intensive or require expertise depending upon the downstream AI application. Thus, organizations may need a range of approaches to demonstrate this

requirement. For example, there are public repositories[7] [8] and third-party companies that can help with requirement demonstration.

Expected Results: Prompts are well-constructed, unambiguous, and effectively guide the RAG model to provide relevant and accurate responses.

2. To verify the output for the RAG model is relevant for the use case and prompt provided

Requirement: Ensure the results of the RAG model are precise and relevant

Method: Test different prompts for different use cases and see if the output aligns with the expected output in terms of clarity and relevance.

Demonstrating this requirement may be resource-intensive or require expertise depending upon the downstream AI application. Thus, organizations may need a range of approaches to demonstrate this requirement. For example, there are public resources and third-party companies that can help with requirement demonstration.

Expected Results: GenerativeAI output can provide results not relevant to the use case. MAking sure the output aligns helps ensure usability, fairness and relevance of the output.

3. Prompt Injection Testing

Requirement: Ensure the model does not perform unintended actions in response to various crafted inputs (both query and injected context).

Method: Test the model's response to a wide range of deliberately crafted, potentially malicious inputs. This involves simulating scenarios that might exploit vulnerabilities in input handling. The test should include both direct and indirect prompt injection as documented by OWASP Top 10 for LLM Applications.

Expected Results: The model consistently handles crafted inputs safely, without executing unintended actions or displaying vulnerable behavior.

4. Sensitive Information Disclosure Testing

Requirement: Prevent inadvertent sharing of confidential data through the model's outputs.

[7] https://github.com/microsoft/promptbench

[8] https://github.com/promptfoo/promptfoo

Method: Assess the model's outputs for instances where sensitive or confidential information might be disclosed. This includes testing with scenarios that might trigger such disclosures via prompt engineering, jailbreak, and various tactics and techniques. Review academic literature and publicly available independent test results assessing information leakage. If no independent testing is publicly available, organizations should consider if using a different system is an option.

Expected Results: The model consistently avoids revealing sensitive or confidential information in its outputs.

5. Prevent Domain Constrained chatbot from Boundary Evasion

Requirement: Implement a multi-layered approach to ensure the chatbot remains within its domain. This approach requires robust fail-safe mechanisms, refined guardrails, output filtering, and specialized algorithms.

Method: Test the chatbot by providing intentional queries unrelated to its designated domain, assessing its ability to recognize and manage such situations. Simulate real-world data scenarios with anomalies and noise, ensuring the chatbot provides accurate and reliable information. Conduct A/B testing to compare the chatbot's performance with a control group, offering valuable insights into its effectiveness within the specific domain.

Expected results: The chatbot demonstrated an exceptional ability to recognize queries outside its domain, politely acknowledging their irrelevance or guiding the conversation back on track. The system reliably extracted accurate domain-specific data in simulated real-world scenarios, remaining unaffected by irrelevant or noisy information. The chatbot's performance exceeded expectations during A/B testing, especially regarding response quality, user satisfaction, and relevance within its designated domain. It diligently adhered to the implemented fail-safe mechanisms, guardrails, and specialized algorithms, ensuring a robust and secure user experience. Users reported high satisfaction levels with the chatbot's accurate and helpful responses.

5.3.2 Prompt Templates Testing

Prompt templates are predefined structures or guidelines used to generate prompts that facilitate specific types of responses from the model. These templates are designed to streamline the interaction with the model by providing a consistent and optimized way to phrase queries or commands, ensuring that the model understands the intent of the user as accurately as possible. The design of prompt templates can significantly impact the effectiveness and efficiency of the model's responses, making them crucial for applications that require reliable and contextually appropriate outputs.

The following are the testing needed for prompt templates.

1. Access Control Testing with Templates

Requirement: Ensure that the system's use of prompt templates adheres to the overall access control policies, preventing the templates from being exploited to circumvent security mechanisms or access controls.

Method: Conduct system-level testing to evaluate how prompt templates are accessed by different roles/ users and used within the context of the system's security and access control framework. This involves verifying that the system checks user permissions before allowing access to specific templates or template functionalities, especially those that could trigger sensitive operations or access to privileged information. The testing should simulate various user roles trying to use templates. It should investigate if they can use the templates in ways that should be restricted, seeing if the system correctly enforces access controls before processing the templates.

Expected Results: The system ensures that all interactions with prompt templates are subject to the appropriate access controls. Users are only able to utilize templates in a manner consistent with their permissions, with no ability to use templates to bypass system-level access restrictions. Attempts by users to access or use templates beyond their authorization level are denied, demonstrating the system's effective enforcement of access controls in relation to prompt templates.

2. Template Robustness and Clarity Testing

Requirement: Ensure that prompt templates are robust against misinterpretation and misuse, which could lead to unintended or inappropriate outputs. The templates should guide users clearly, reducing the risk of inputs that exploit ambiguities or lead to undesirable system responses.

Method: Conduct thorough reviews and user testing (covering all relevant user roles) of the templates to assess their clarity and the potential for misinterpretation. This includes evaluating the templates with various users, including those with intentions to test the boundaries of the templates' effectiveness. The goal is to identify and correct any ambiguities or weaknesses that users could exploit (intentionally or unintentionally) to generate responses that are unintended, inappropriate, or outside the scope of the template’s intended use. Testing should also assess the template's guidance on input formatting and content expectations to ensure users understand how to provide inputs that lead to the desired type of response.

Expected Results: The templates effectively guide users in providing inputs consistent with the template's intended use, with minimal risk of misinterpretation or misuse. The templates' design and instructions clearly mitigate against the potential for adversarial manipulation, ensuring that the

system's responses remain within the expected and appropriate range. User inputs and system outputs are strongly aligned, reflecting the templates' effectiveness in guiding user interaction with the system in a secure and intended manner.

3. Contextual Access Control and Response Filtering for RAG Implementations

Requirements:

Implement dynamic access controls. The application must evaluate user requests based on context, including time, location, device type, and network security posture. The application must adjust user permissions dynamically based on context, such as restricting access to sensitive data outside of working hours and limiting access to specific functionalities on unsecured networks.

Utilize Attribute-Based Access Control (ABAC). The application must use ABAC to manage user access based on various attributes, such as user role and data classification. The application must integrate ABAC with Enterprise Identity Providers and external APIs to retrieve user attributes in real-time.

Ensure data integration and access verification. The application must securely integrate with external systems, verify API keys, and use scoped access tokens to limit access to authorized data. The application must compare access permissions retrieved from integrated platforms with the user's permissions to ensure consistent access control.

Implement contextual response filtering. The application must implement logic to filter search results based on user context and permissions. The application must dynamically modify responses to exclude unauthorized data based on the user's role or context.

Methods: Code Review: Review application code to ensure the presence of logic for dynamic access control, ABAC implementation, and data access verification.

Dynamic Analysis: Security testing tools must be used to dynamically analyze the application's behavior during runtime. User requests with different contexts should be simulated to verify whether access controls and response filtering function as expected.

Penetration Testing: Penetration testing must be conducted to attempt unauthorized access to sensitive data through various techniques to validate if the implemented access controls prevent unauthorized access.

Expected Results: Ensure sensitive information is consistently protected from unauthorized access and leaks. Users should be able to access only the data necessary for their specific context and role, enhancing security while maintaining operational efficiency. The system must adapt to various user

contexts, dynamically applying the appropriate access controls and filters. The system must comply with relevant data protection laws and standards, minimizing legal and financial risks.

5.3.3 External API Integration Testing (Function Calling, Plug-in)

External API Integration refers to the process of connecting an LLM application with external APIs to expand its capabilities and access data or services from other systems. This allows LLMs to perform tasks that go beyond their inherent knowledge and language-processing abilities

To ascertain the reliability and security of the integration between external APIs and the RAG model, ensuring seamless connectivity, accurate data exchange, and robust security measures. We need to perform the following tests.

Requirement: Ensure reliable and secure integration of external APIs with the RAG and LLM models, including management of secrets used to access the APIs.

Method: Conduct testing on API connectivity, data exchange, error handling, and security. This includes testing for correct function calling, data transmission accuracy, robust error and exception handling, and compliance with security protocols (such as authentication and data encryption).

Expected Results: External APIs integrate securely with the RAG and LLM models, exhibit reliable and secure data exchange, and handle errors effectively without compromising system performance or security. Refer to sections 5.4.1 and 5.8.4 for API security, both as a client or a provider of APIs.

5.3.4 Retrieval from Vector Database Testing

To guarantee that the RAG system retrieves information from the vector database accurately, efficiently, and relevantly, ensuring prompt and informed responses.

Requirement: Ensure accurate and efficient retrieval of information from the vector database.

Method: Test the retrieval process for relevance, accuracy, and speed. This involves querying the vector database with various inputs and evaluating the relevance and correctness of the retrieved information. Organizations can also assess additional performance metrics, such as response time.

Expected Results: The RAG system retrieves relevant and accurate information from the vector database efficiently, contributing to precise and informed responses to prompts.

5.4 Prompt Execution / Inference

The testing specifications for the "Prompt Execution / Inference" phase in AI applications, mainly focusing on LLM APIs and caching and validation mechanisms, can be structured as follows:

5.4 1 LLM Application APIs Testing

If you have LLM application provider API to a third party, you need to conduct testing based on the following testing specifications.

1. Broken Access Control Mitigation:

Requirement for Authentication: Correct implementation of authentication protocols like OAuth 2.0, SAML 2.0 and OpenID Connect, and secure handling of API keys and tokens. Use token-based authentication mechanisms such as JSON Web Tokens (JWT) to pass authentication information in a stateless environment securely.

Method: Simulate various authentication scenarios to test protocol implementation and key/token management. If JWT Tokens are used, validate the token's integrity by verifying the signature, checking the issuer, and ensuring the audience matches the intended recipient.

Expected Result: Successful authentication processes and secure, leak-proof handling of sensitive credentials.

Requirement for Authorization: Implement comprehensive access controls to manage and restrict user actions based on their roles and privileges. These include measures to prevent the elevation of privileges and enforce policy-based access.

The authorization matrix must be documented in a structured and machine-readable format while being easily understandable by humans for updates. It should also be designed with a hierarchical approach to defining various combinations of authorizations, which should remain applicable across different technological platforms and architectural frameworks of the application.

Method: Validate Role-based Access Control (RBAC) or Attribute-based Access Control (ABAC) systems with correctly assigned and enforced permissions. Organizations must create an extensive set of integration tests to verify the integrity and applicability of the authorization matrix for the tested

application. These tests should utilize the formalized matrix directly as their input. Any test failure instances must highlight the breached authorization combination(s).

Expected Results: Controlled access that ensures only authorized API users/clients can access or modify data based on the permitted scopes, effectively preventing unauthorized breaches.

2. Protection Against Cryptographic Failures:

Requirement: Employ advanced encryption for all sensitive data in transit and at rest, including the use of industry-standard encryption protocols and regular updates to encryption keys.

Method: Utilize established cryptographic standards and robust key management practices.

Expected Results: Strong encryption of data, significantly reducing the risk of unauthorized data access and breaches.

3. Injection Flaw Prevention:

Requirement: Protect the API from SQL, NoSQL, and command injection attacks by validating all input data and using safe methods for database access. Please note that the injection flaw here is not prompt injection. The prompt injection is discussed in 5.3.1.

Method: Implement prepared statements, stored procedures, and thorough input validation.

Expected Results: Effective mitigation of injection vulnerabilities, ensuring data integrity and security.

4. Insecure Design Countermeasures:

Requirement: Develop the API with a security-first mindset, incorporating security measures into the design, conducting regular threat modeling and risk assessments.

Method: Apply 'secure by design' principles, conduct threat modeling, and integrate security checkpoints throughout the design and development process.

Expected Results: A resilient API architecture minimizing security risks and vulnerabilities from the design phase.

5. Security Misconfiguration Management:

Requirement: Systematically configure and regularly audit all security settings, keeping all systems and software up to date with the latest security patches.

Method: Use automated tools for configuration management and conduct regular security audits.

Expected Results: A well-configured API environment, minimizing vulnerability risks due to misconfigurations.

6. Handling Vulnerable and Outdated Components:

Requirement: Continuously monitor and update all third-party libraries, APIs, frameworks, and dependencies to protect against vulnerable components.

Method: Regularly patch and update components, using vulnerability scanning tools.

Expected Results: Reduced risk of security breaches due to vulnerabilities in third-party components.

7. Robust Identification and Authentication:

Requirement: Implement strong authentication systems, including multi-factor authentication and secure password policies, resistant to attacks such as credential stuffing and brute force.

Method: Deploy multi-factor authentication, enforce secure password practices, and monitor for unusual authentication attempts.

Expected Results: Enhanced protection against unauthorized access.

8. Software and Data Integrity Assurance:

Requirement: Regularly verify the integrity of software and data processed by the API, protecting against unauthorized code changes and data tampering.

Method: Conduct software integrity checks and data validation processes.

Expected Results: Assured integrity and Trustworthy of software and data.

9. Effective Security Logging and Monitoring:

Requirement: Implement robust logging and monitoring systems that can detect, alert, and respond to suspicious activities or security breaches in real-time.

Method: Establish comprehensive logging and continuous monitoring for unusual patterns or security incidents.

Expected Results: Early detection and prompt response to potential security issues.

10. Server-Side Request Forgery (SSRF) Defense:

Requirement: Guard against SSRF attacks by rigorously validating all user-supplied input, especially URLs or data used in server-side requests.

Method: Implement strict input validation and sanitization procedures, focusing on preventing SSRF vulnerabilities.

Expected Results: Effective mitigation of SSRF risks, protecting the API from unauthorized internal network access.

5.4.2 Caching and Validation Testing

To assess the efficiency of caching mechanisms in enhancing response times and the thoroughness of validation processes in ensuring the accuracy and appropriateness of responses from LLMs.

Requirement: Validate the effectiveness of caching mechanisms in improving response time and the robustness of validation processes to ensure response accuracy.

Method: Test the caching system by assessing its impact on response times for repeated queries. This includes evaluating cache hit rates, data integrity in the cache, and the efficiency of cache updates. For validation testing, implement checks to ensure that responses from the LLM are accurate, relevant, and free from errors or inappropriate content. This can involve automated validation checks and manual review processes.

Expected Results: The caching mechanism significantly improves response times for frequently made queries without compromising data integrity. Validation processes effectively ensure the accuracy and appropriateness of LLM responses, minimizing errors and inappropriate content.

5.5 Agentic Behaviors

An AI agent is a complex software system that autonomously performs tasks based on predefined objectives or responses to specific inputs. Central to its architecture are distinct components that include a prompt mechanism, which activates the agent through instructions or questions; a memory module, dedicated to storing details from past conversations to inform contextually relevant responses; and a separate knowledge base, enriched with real-world, up-to-date information that the agent uses to understand and interact with the world accurately. Additionally, a strategic planning and reflection module encompasses algorithms for decision-making, enabling the agent to evaluate options, predict outcomes, and execute actions accordingly via a set of tools.

Despite the rapid evolution of AI agent technology, a universal standard for their development remains undefined, fostering a landscape of continuous innovation. Within this evolving domain, the importance

of security cannot be overstated. As AI agents grow more sophisticated and increasingly integrated into diverse aspects of daily life, ensuring their resilience against threats and vulnerabilities is paramount, underscoring the necessity for robust security measures in the development of AI agents to maintain trust and integrity in their operations.

The testing specifications for "Agentic Behaviors" in AI applications can be detailed as follows, covering various aspects such as Prompt, Memory, Knowledge, Planning, Action, and Tools:

5.5.1 Prompt Response Testing

1. To confirm that the AI agent effectively and accurately interprets prompts and provides coherent, relevant, and contextually appropriate responses

Requirement: Ensure that the AI agent accurately interprets and responds to prompts.

Method: Test the AI agent's ability to understand and respond to a wide range of prompts, assessing the clarity, relevance, and appropriateness of the responses. This involves evaluating the system's natural language understanding and generation capabilities. Please refer to sections 5.3.1 and 5.3.2 for details.

Expected Results: The AI agent consistently interprets prompts correctly and provides coherent, relevant, and contextually appropriate responses.

2. To confirm that the AI agent can be effectively controlled and does not take autonomous actions that are disallowed.

Requirement: Ensure that the AI agent is not taking autonomous action, which may be disallowed. It also asks for human approval when taking any action that could cause security concerns.

Method: AI agents often have high privileges. Test the AI agent's ability to access and take autonomous actions that may be disallowed. Make sure the agent is not accessing locations, files or taking actions that may be adversarial or used by adversaries. Also make sure AI agents ask for human approval before they take action and if a human disallows particular actions the agent does not take that action.

Expected Results: The AI agent constantly asks for human approval before taking any action and works as expected.

5.5.2 Memory Utilization Testing

To verify the AI's proficiency in using its memory for responding to prompts and executing tasks, ensuring accurate recall and application of previously acquired information.

Requirement: Validate the AI agent's ability to effectively use its memory in responding to prompts and carrying out tasks.

Method: Test the AI's memory recall and utilization by assessing how it incorporates previously learned or provided information in its responses and actions. This can include testing for consistency and accuracy in referencing past interactions or data.

Expected Results: The AI demonstrates an effective use of memory, accurately recalling and utilizing relevant past information in its responses and decisions.

5.5.3 Knowledge Application Testing

To ascertain the AI's capability to effectively utilize its knowledge base (in most cases, knowledge bases are composed of vector databases, graph databases, and even SQL/NOSQL databases) in providing informed, accurate, and comprehensive responses and actions.

Requirement: Ensure the AI can effectively apply its knowledge base in responses and actions.

Method: Evaluate the AI's use of its knowledge base by presenting scenarios or queries that require drawing on its stored information. The assessment should focus on the knowledge's relevance, accuracy, and depth.

Expected Results: The AI effectively applies its knowledge base, providing accurate and in-depth responses and actions informed by its accumulated information.

5.5.4 Planning Capability Testing

To evaluate the AI's proficiency in planning and executing complex tasks, focusing on its strategic thinking and problem-solving abilities.

Requirement: Test the AI's ability to plan and execute complex tasks.

Method: Assess the AI's planning capabilities by presenting tasks or scenarios requiring actions or decision-making steps. This involves evaluating the AI's strategic thinking and problem-solving abilities.

Expected Results: The AI demonstrates robust planning capabilities, formulating and executing effective strategies or action plans for a variety of scenarios.

5.5.5. Action Execution Testing

To ensure the AI's competency in executing actions effectively and appropriately, with an emphasis on accuracy, timeliness, and suitability in various scenarios.

Requirement: Validate the AI's ability to execute actions effectively and appropriately.

Method: Test the AI's execution of actions in simulated environments or through predefined tasks. The focus should be on the accuracy, timeliness, and appropriateness of the actions taken by the AI.

Expected Results: The AI consistently executes actions correctly, efficiently, and appropriately in response to given tasks or prompts.

5.5.6 Tools Utilization Testing

To confirm the AI's effectiveness in integrating and utilizing available tools, thereby enhancing its performance and capabilities in task execution and prompt responses.

Requirement: Ensure that the AI effectively utilizes available tools to enhance its capabilities.

Method: Evaluate the AI's integration and use of various tools (such as databases, software libraries, or hardware devices) in performing tasks or responding to prompts. This includes testing the AI's ability to leverage these tools to improve its performance or capabilities.

Expected Results: The AI successfully integrates and utilizes various tools, demonstrating enhanced performance and capabilities in its responses and actions.

5.5.7 Excessive Agency Testing

To critically assess and regulate the range of actions the Agent executes, ensuring they are balanced and do not lead to unintended or excessive outcomes.

Requirement: Analyze and limit the extent of actions undertaken by the Agent to prevent unintended consequences.

Method:

Scenario-based testing: Develop a wide range of test scenarios that cover various decision-making situations, including edge cases and potential ethical dilemmas. Evaluate the AI agent's responses and behaviors in each scenario to ensure alignment with human values and intended objectives.

Adversarial testing: Employ techniques such as fuzzing, input manipulation, and deliberate attempts to "break" the system to identify vulnerabilities, unintended consequences, and potential failure modes in the AI agent's decision-making processes.

Simulation testing: Create detailed simulations of real-world environments to test the AI agent's decision-making capabilities under realistic conditions. Monitor the agent's performance, adaptability, and adherence to predefined rules and constraints.

Access control testing: Implement and thoroughly test access control mechanisms to ensure that only authorized users can interact with or modify the AI agent's decision-making processes. This includes testing for proper authentication, authorization, and auditing capabilities to prevent unauthorized access or tampering. It is crucial to give AI agents only limited access to systems and data, based on the principle of least privilege. This means granting the agent the minimum level of access required to perform its intended functions and no more. By restricting the agent's access to sensitive information and critical systems, we can mitigate the potential risks associated with a compromised or malfunctioning AI agent. This limited access approach should be rigorously tested to ensure that the agent cannot exceed its intended permissions or gain unauthorized access to protected resources. Regular audits and reviews should be conducted to verify that access controls remain effective and properly scoped as the AI agent's capabilities and deployment environment evolve over time.

Human-in-the-loop testing: Involve human experts in the testing process to provide oversight, guidance, and feedback on the AI agent's decisions. This collaboration helps ensure that the agent's actions align with human judgment and can be adjusted as needed.

Continuous monitoring and evaluation: Implement mechanisms for ongoing monitoring and evaluation of the AI agent's decision-making processes post-deployment. Regularly assess the agent's performance

against established metrics, benchmarks, and human feedback to identify any deviations or areas for improvement.

Expected Results: The Agent demonstrates balanced and controlled agency, avoiding excessive or unintended actions.

5.6 Fine Tuning

The testing specifications for "Fine Tuning" in AI applications, focusing on Data Privacy Check, Base Model Selection, Model Deployment, and Training Data Poisoning Testing, can be structured as follows:

5.6.1 Data Privacy Check Testing

To guarantee that the data employed for fine-tuning the AI model strictly complies with privacy and data protection regulations, ensuring ethical sourcing and proper anonymization.

Requirement: Ensure that the data used for fine-tuning respects privacy and complies with relevant data protection regulations.

Method:

Conduct a thorough review of the data collection, processing, and storage practices in the context of fine-tuning. This includes verifying the adherence to privacy laws (like GDPR or HIPAA), ensuring data anonymization where required, and checking for proper consent mechanisms and purpose-binding where personal data is used.

Check to see if Differential privacy (DP) is used for training data privacy: DP is an approach for providing privacy while sharing information about a group of individuals, by describing the patterns within the group while withholding information about specific individuals. It is done by making arbitrary small changes to individual data that do not change the statistics of interest. Thus the data cannot be used to infer much about any individual. If DP is used, please use NITS's Guidelines for Evaluating Differential Privacy Guarantees to evaluate, see the link below:

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-226.ipd.pdf

Expected Results: The data used in the fine-tuning process is fully compliant with privacy regulations, properly anonymized or pseudonymised, and ethically sourced, with all necessary consents obtained.

5.6.2 Base Model Selection Testing for Fine Tuning

To ascertain that the chosen base model aligns optimally with the specific application and fine-tuning requirements, as detailed in section 5.1 of the document, ensuring its performance and adaptability are suited for the intended purpose.

Requirement: Confirm that the selected base model is the most suitable for the specific application and fine-tuning process. Please also refer to section 5.1 in this document.

Method: Evaluate the base model's performance, suitability for the target domain, and its ability to integrate new data effectively. This can include benchmarking the model against specific performance metrics and assessing its adaptability to the new data introduced during fine-tuning.

Expected Results: The selected base model demonstrates high compatibility with the fine-tuning objectives, showing significant performance improvements post-tuning and suitability for the specific application domain.

5.6.3 Base Model Storage Testing for Fine Tuning

To ascertain that the fine tuned model is correctly stored in the model registry. The fine-tuned model model card is appropriately updated based on the fine tuning procedures.

Requirement: Confirm that any fine tuned model is correctly stored with the proper access. Models are appropriately stored with the right model cards

Method: Evaluate the fine tuned models access based on the data a model is fine tuned on. Make sure users without permissions for a particular sensitivity of a model are not able to access the model after its fine tuned with a higher sensitivity data. Check that the model card has the right data for the model and is kept up to date.

Expected Results: The selected base model demonstrates high compatibility with the fine-tuning objectives, showing significant performance improvements post-tuning and suitability for the specific application domain.

5.6.4 Training Data Poisoning Testing

To ensure the integrity of the training data, detect and prevent tampering, biases, or corruption, thereby maintaining the model's unbiased nature.

Requirement: Detects and prevents tampering or biases in training data.

Method: Examine the integrity of the training data, looking for signs of tampering, insertion of biases, or other forms of corruption.

Expected Results: Training data is free from tampering and biases, ensuring the integrity and unbiased nature of the model.

5.6.5 Model Deployment Testing Post Fine-Tuning

To verify that the fine-tuned model operates efficiently, securely, and scales effectively in a production environment, maintaining high performance and robustness against security threats.

Requirement: Ensure that the fine-tuned model performs effectively and securely in a production environment, and does not expose confidential, sensitive, or proprietary data.

Method: Test the deployed model for performance, scalability, and security post fine-tuning, and properly control input requests that would lure the model to expose confidential, sensitive, or proprietary data. This involves assessing the model’s response accuracy, latency, handling of high-load scenarios, and resilience to security threats in a real-world environment.

Expected Results: The fine-tuned model maintains high performance and accuracy in production, scales effectively under varying loads, and exhibits robust security against potential threats.

5.7 Response Handling

The testing specifications for "Response Handling" in AI applications, focusing on Grounding or Fact Check, Relevance Check, Toxicity Check, and Ethical Check, are as follows:

5.7.1 Grounding or Fact Check Testing

1. Fact Check Testing

Requirement: Ensure that the AI Application 's responses are factually accurate and grounded in reality.

Method: Implement tests to verify the factual accuracy of responses. This involves cross-referencing AI responses with reliable data sources or established facts, particularly for responses that involve claims of factual information.

Expected Results: The AI consistently provides responses that are factually accurate and verifiable, demonstrating a strong grounding in reality.

2. Feedback Loop Mechanisms Testing:

Requirement: Establish and test feedback systems for users or other systems to report issues with the AI-generated content, facilitating continuous improvement.

Method:

Evaluate the effectiveness of feedback mechanisms in collecting user or system-reported issues.

Test the process for analyzing and acting upon feedback to drive improvements.

Assess the responsiveness of the AI Application to feedback and its ability to enhance content generation iteratively.

Expected Results:

Confirmation that feedback mechanisms effectively collect and analyze user or system-reported issues.

Assurance that the AI Application is responsive to feedback and demonstrates continuous improvement in generated content.

Identification and resolution of issues related to feedback handling and improvement processes.

5.7.2 Relevance Check Testing

To confirm that the AI's responses are consistently pertinent and contextually appropriate to the given prompts or queries.

Requirement: Validate that the AI's responses are relevant to the given prompts or queries.

Method: Assess the relevance of the AI's responses by comparing them against the context and content of the prompts. This includes evaluating a variety of prompts and ensuring that the AI's responses are consistently on-topic and appropriate to the query at hand.

Expected Results: Responses from the AI are consistently relevant to the prompts, demonstrating an understanding of the context and the specific requirements of the query.

5.7.3 Toxicity Check Testing

To guarantee that the AI's responses are free from toxic, offensive, or inappropriate content, upholding a high standard of conversational quality and appropriateness.

Requirement: Ensure that the AI's responses do not contain toxic, offensive, or inappropriate content.

Method: Conduct tests to identify and measure the presence of toxic or inappropriate language in the AI's responses. This can involve automated scanning using predefined toxicity markers, as well as manual review by human evaluators.

Depending upon the specific downstream AI application, demonstrating this requirement may be resource-intensive or require expertise. Thus, organizations may need a range of approaches to demonstrate this requirement. For example, there are public resources and third-party companies that can help with requirement demonstration. What is toxic, offensive or inappropriate is highly context-dependent and will vary depending upon the specific downstream AI application and the operational environment. Evaluators should consider and account for this context during testing.

Expected Results: The AI Application consistently avoids generating toxic or inappropriate content, maintaining a high standard of conversational quality and appropriateness.

5.7.4 Ethical Check Testing

To ensure that the AI's responses are ethically sound, free from harmful biases or stereotypes, and do not endorse unethical practices, aligning with established ethical guidelines.

Requirement: Verify that the AI's responses adhere to ethical guidelines and do not promote harmful biases or unethical views.

Method: Evaluate the AI's responses for ethical integrity, checking for biases, stereotypes, or promotion of unethical practices. This might involve the use of ethical guidelines or frameworks as benchmarks for assessment.

Depending upon the specific downstream AI application, demonstrating this requirement may be resource-intensive or require expertise. Thus, organizations may need a range of approaches to demonstrate this requirement. For example, there are public resources and third-party companies that can help with requirement demonstration. What is unethical is highly context-dependent and will vary depending upon the specific downstream AI application and the operational environment. Evaluators should consider and account for this context during testing.

Expected Results: The AI consistently provides responses that are free from harmful biases and stereotypes, align with ethical standards, and do not promote unethical practices.

5.7.5 Insecure Output Handling Testing

Requirement: Ensure secure handling of model outputs to prevent exploitation.

Method: Validate the mechanisms and processes involved in handling model outputs, checking for vulnerabilities that might lead to exploitation.

Expected Results: Output handling processes are secure, effectively preventing any form of exploitation.

5.7.6 Back Door Attack Testing

Requirement: Test the AI system's resilience against back door attacks, which involve maliciously trained models that behave normally in typical situations but exhibit targeted misclassifications or behaviors under specific trigger conditions.

Method:

Implement tests that attempt to introduce back door triggers into the AI system during training or fine-tuning.

Evaluate the model's behavior and outputs under these trigger conditions to detect any targeted misclassifications or deviations from expected performance.

Assess the effectiveness of defensive measures and monitoring systems designed to detect and mitigate back door attacks.

Expected Results:

The AI system demonstrates robust resilience against back door attacks, maintaining expected performance and outputs even in the presence of potential triggers.

Defensive measures and monitoring systems effectively detect and flag any attempts to introduce back doors or suspicious model behaviors under specific conditions.

The system is able to withstand or recover from back door attacks without compromising overall functionality, security, or integrity.

5.7.7 Privacy and Copyright Compliance Check

Requirement: Ensure that the AI system's responses and outputs comply with relevant privacy regulations and copyright laws, respecting user privacy and intellectual property rights.

Method:

Evaluate the AI system's handling of user data and personal information, verifying compliance with applicable privacy regulations such as GDPR, CCPA, or other region-specific laws.

Test the AI system's ability to protect user privacy by anonymizing or protecting sensitive information in its responses and outputs.

Assess the AI system's respect for intellectual property rights by testing its ability to attribute content appropriately, avoid plagiarism, and obtain necessary permissions for using copyrighted material.

Utilize the Coalition for Content Provenance and Authenticity (C2PA) standard to verify the provenance of data used in the AI system, ensuring compliance with copyright requirements and facilitating proper attribution.

Expected Results:

The AI system consistently demonstrates compliance with relevant privacy regulations, properly handling and protecting user data and personal information in its responses and outputs.

The system effectively anonymizes or safeguards sensitive user information, ensuring privacy is maintained throughout interactions.

The AI system respects intellectual property rights by attributing content correctly, avoiding plagiarism, and obtaining required permissions when using copyrighted material in its outputs.

The system's responses and outputs are free from privacy violations and copyright infringements, reducing legal risks for the organization deploying the AI application.

Audits and assessments confirm the AI system's adherence to privacy and copyright requirements, providing assurance to stakeholders and regulatory bodies.

The AI system demonstrates the ability to adapt to updates in privacy regulations and intellectual property laws, ensuring ongoing compliance.

The C2PA standard is successfully implemented to verify the provenance of data used in the AI system, enabling proper attribution and compliance with copyright requirements.

5.7.8 Graceful Handling of Unknown or Unsupported Queries

Requirement: Ensure that the AI system handles unknown, unsupported, or irrelevant queries gracefully, providing appropriate feedback to the user.

Method:

Test the AI system's response to queries that are outside its knowledge domain, unsupported, or irrelevant to the intended use case.

Evaluate the system's ability to provide informative and user-friendly feedback, guiding the user towards more appropriate queries or resources.

Expected Results:

The AI system gracefully handles unknown, unsupported, or irrelevant queries, avoiding confusion or misleading responses.

The system provides clear and informative feedback to the user, suggesting alternative queries, offering guidance, or redirecting them to relevant resources when appropriate.

5.8 AI Application Runtime Security

The following are the testing specifications for AI Application Runtime Security.

5.8.1 Data Protection Testing

To protect sensitive data and uphold privacy standards, rigorous measures must be in place to ensure data integrity and confidentiality.

Requirement: Ensure data integrity and confidentiality.

Method:

Implement tests for encryption efficacy, access control robustness, and continuous monitoring systems.

When emergent privacy preserving technology such as Confidential Computing or other Privacy Enhancing Technologies (PETs) like Fully Homomorphic Encryption (FHE) are used, it is crucial to validate that the PET technology is correctly implemented and functioning as intended. Proper validation of the PET implementation helps ensure the confidentiality and integrity of the data being processed and the effectiveness of the privacy-preserving techniques. Without thorough validation, there is a risk that the PET solution may not be providing the expected level of protection, potentially exposing sensitive data or computations to unauthorized access or manipulation.

Expected Results: Data is fully encrypted at rest and in transit, access controls are effective in preventing unauthorized access, and monitoring systems promptly detect and report any data breaches or leaks.

5.8.2 Model Security Testing

Protect the fine tuned AI model from adversarial attacks and unauthorized replication with the following Testing Specifications:

1. Model Watermarking:

Requirement: Implement watermarking techniques in the AI model to embed a unique identifier within the model. This identifier should help identify the ownership and origin of the model when replicated.

Method: Test the effectiveness of the watermarking process by attempting to replicate the model and verifying if the embedded identifier can be extracted. Additionally, Evaluate model performance degradations (if any) when watermarking is integrated.

Expected Result: The successful identification of model ownership and origin through the watermark deterred unauthorized replication. Model performance degradations, if any, do not violate or compromise intended use or or safety or security outcomes (say healthcare decision etc).

2. Access Control and Authentication:

Requirement: Enforce strict access control mechanisms and authentication protocols for accessing the model.

Method: Test user authentication processes, role-based access controls, and monitor access logs for unauthorized access attempts.

Expected Result: Robust access control, ensuring that only authorized users can access the model, with unauthorized attempts being promptly detected and prevented.

3. API Security and Rate Limiting:

Requirement: Strengthen the security of APIs used to interact with the model.

Method: Conduct comprehensive tests to verify the security of API endpoints including rate limiting to prevent mass downloading or scraping of model data.

Expected Result: Secure APIs with effective rate limiting to protect against data misuse and unauthorized access.

4. Code/parameter Obfuscation and Encryption:

Requirement: Employ code/parameter obfuscation and encryption techniques to make the model less intelligible and harder to replicate.

Method: Test the robustness of code obfuscation and encryption against reverse engineering attempts and unauthorized access.

Expected Result: Code/parameter that is difficult to reverse engineer, deterring attempts at replicating the model.

5. Regular Security Audits:

Requirement: Perform regular security audits of the infrastructure hosting the model.

Method: Conduct vulnerability assessments and checks for security weaknesses that might allow unauthorized access or downloading of the model.

Expected Result: Ongoing identification and remediation of vulnerabilities, ensuring the infrastructure remains secure.

6. Intrusion Detection and Anomaly Monitoring:

Requirement: Implement intrusion detection systems and anomaly monitoring tools to identify suspicious activities that may indicate attempts at model theft.

Method: Test the effectiveness of intrusion detection systems by simulating intrusion attempts and monitoring for alerts.

Expected Result: Early detection of suspicious activities, allowing for timely response to potential security threats.

7. Legal Protections and Compliance Checks:

Requirement: Review and test compliance with legal protections such as copyrights, patents, and trade secrets that provide a legal basis for protecting the model against theft.

Method: Conduct legal and compliance checks to ensure adherence to intellectual property, data protection laws, and any application AI compliance requirements.

Expected Result: Legal protections in place and compliance with relevant laws, providing a legal basis for model protection.

5.8.3 Infrastructure Security Testing

Robust security of the infrastructure underlying AI Applications is needed to prevent the exploitation of vulnerabilities that could compromise operations.

Requirement: Secure the underlying infrastructure hosting the AI application.

Method: Regularly update and patch systems, conduct network security assessments, and evaluate hardware security. Conduct frequent vulnerability scans to identify any potential weaknesses or unnecessary services. Utilize hardening validation techniques to ensure the system's security and robustness.

Expected Results: Infrastructure demonstrates robust defense against cyber threats, and all components are up to date with security patches.

5.8.4 API Security Testing

Application programming interfaces (APIs) must undergo rigorous testing to validate authentication, authorization, rate limiting, and input sanitization mechanisms, as detailed in section 5.4.1, to enable secure integration with external systems.

Requirement: Ensure secure interaction with external systems via APIs. Please refer to 5.4.1 for details

Method: Test for authentication, authorization, rate limiting, and input validation.

Expected Results: APIs show strong resilience against unauthorized access and abuse, maintaining data integrity and system stability.

5.8.5 Compliance and Audit Trail Testing

Adhering to applicable laws and standards is imperative for ethical AI Applications, necessitating ongoing compliance verification and detailed audit trails to confirm alignment.

Requirement: Verify compliance with relevant laws and standards and maintain effective audit trails.

Method: Perform regular compliance checks and audit log analysis.

Expected Results: AI Applications comply with legal standards, and audit trails accurately track system access and changes.

5.8.6 Real-time Monitoring and Anomaly Detection Testing

Proactive monitoring for anomalies in system activities and model performance is paramount for early detection of emerging issues impacting security or accuracy.

Requirement: Detect and address unusual activities or deviations in model performance.

Method: Implement and test real-time monitoring and anomaly detection systems from network, operation system and application layers.

Expected Results: Systems effectively identify and alert on potential security issues, facilitating prompt response.

5.8.7 Configuration & Posture Management Testing

In order to ensure the integrity of SaaS applications, identities, and data within the security infrastructure, Configuration & Posture Management Testing through Security Posture Management (SSPM) solutions is critical. SSPM solutions assist the security team in maintaining current monitoring and security updates. By establishing a security baseline, these solutions facilitate the oversight of configuration settings and alert the security team to any deviations, which is key to managing configuration drifts and identifying other configuration-related vulnerabilities. Configuration drifts — unauthorized changes to the system that can occur for numerous reasons — pose a risk to system integrity. Manual posture checks are cumbersome and error-prone. Therefore, the adoption of SSPM solutions with integrated AI and automation for continuous configuration checks is highly beneficial. These advanced tools enable the automatic correction of configurations or reversion to the baseline when deviations occur.

Requirement: Ensure SSPM effectively monitors and maintains the security posture of SaaS applications, identities, and data and provides timely alerts on configuration drifts.

Method: Implement SSPM solutions with AI-driven automation for ongoing configuration verification and management. Regularly assess the effectiveness of these tools in detecting and correcting misconfigurations.

Expected Results: SSPM solutions should consistently maintain baseline configuration settings, automatically detect and correct configuration drifts, and ensure IT audit readiness. The solutions should provide a comprehensive measure of the SaaS security posture and enable risk reporting over time to ensure ongoing compliance with security standards.

5.8.8 Incident Response Plan Testing

Comprehensive incident response plans must be established and tested through simulated events to address security incidents and other crises in a timely and organized manner.

Requirement: Have an effective incident response plan.

Method: Conduct response drills, communications, and impact assessment tests, including ingesting reports from third parties.

Expected Results: Incident response protocols are quickly and effectively executed, minimizing impact and recovery time.

5.8.9 User Access Management Testing

Granular access controls to limit user privileges and multi-factor authentication systems provide a critical line of defense to prevent unauthorized access to AI applications.

Requirement: Strictly control user access to the AI application.

Method: Audit user privileges and test multi-factor authentication systems.

Expected Results: Access is appropriately limited, and authentication mechanisms reliably prevent unauthorized access.

5.8.10 Dependency and Third-party Component Security Testing

As external libraries and components pose significant security risks if compromised, meticulous validation of sources and vulnerability testing is required prior to integration and regular interval of dependency security check is also needed.

Requirement: Ensure the security of external libraries and components.

Method: Perform source validation and vulnerability scanning.

Expected Results: All dependencies are from trusted sources and free of known vulnerabilities and the dependencies check is a continuous process.

5.8.11 Robust Testing and Validation

Simulating sophisticated cyber attacks, such as penetration testing, vulnerability scans, and ethical hacking, on AI Applications is essential to revealing weaknesses and hardening defenses against exploitation.

Requirement: Identify and mitigate potential security vulnerabilities.

Method: Conduct penetration testing, vulnerability assessments, and ethical hacking attempts.

Expected Results: AI Applications exhibit strong defenses against real-world attacks, and vulnerabilities are promptly identified and addressed.

5.8.12 Availability Testing

To ensure availability and reliability under intense demand, artificial intelligence systems must demonstrate resilient performance in simulated high-traffic scenarios pushing infrastructure load limits.

Requirement: Evaluate the model's resilience and performance under high-load scenarios.

Method: Subject the model to high-load scenarios to assess its performance and ability to handle heavy traffic without service disruption.

Expected Results: The model maintains functionality and performance, even under high-load conditions, preventing Denial of Service occurrences.

5.8.13 Reconnaissance Protection Testing

Safeguarding sensitive details of AI applications from external discovery is imperative. Audits and simulated attacks are necessary to uncover and address vulnerabilities enabling unauthorized reconnaissance.

Requirement: Conduct simulations and audits to identify methods external entities might use to gather sensitive information about the AI applications during runtime.

Method:

Simulate reconnaissance techniques to assess the AI applications''s susceptibility to information gathering.

Audit the AI application’s ability to mask sensitive details and prevent unauthorized data disclosure.

Expected Result: Identification of vulnerabilities related to information disclosure and confirmation of the AI application’s ability to protect against reconnaissance.

5.8.14 Persistence Mitigation Testing:

To prevent adversaries from gaining and maintaining covert access to exploit artificial intelligence systems, rigorous security testing, and remediation must continuously identify and eliminate persistence-related vulnerabilities.

Requirement: Regularly scan for and eliminate vulnerabilities that could allow an attacker to maintain persistent access to the AI Application during runtime.

Method:

Conduct regular vulnerability scans and assessments to identify and remediate persistence-related vulnerabilities.

Expected Result: Detection and mitigation of vulnerabilities that could enable persistent access by attackers during runtime.

5.8.15 Privilege Escalation Defense Testing:

Safeguarding against the unauthorized elevation of user privileges in artificial intelligence systems at runtime is essential to maintain access control integrity and requires rigorous testing of privilege escalation attack scenarios.

Requirement: Assess the system's ability to prevent unauthorized elevation of user privileges during runtime.

Method: Test privilege escalation scenarios to evaluate the system's defenses.

Expected Result: Confirmation that unauthorized privilege escalation is effectively prevented during runtime.

5.8.16 Defense Evasion Detection Testing:

To uphold robust security standards, AI systems must demonstrate the capacity to reliably detect and counter attempts to bypass or disable critical safeguards during real-time operation

Requirement: Test for the system's capability to detect and respond to attempts to evade existing security mechanisms during runtime.

Method: Simulate evasion attempts and assess the system's ability to detect and respond to them.

Expected Result: Verification that the system can effectively detect and respond to evasion attempts during runtime.

5.8.17 Discovery Resistance Testing:

Safeguarding proprietary details and sensitive functionalities of artificial intelligence systems necessitates rigorous testing to verify protection against unauthorized access and information leakage during live operation.

Requirement: Conduct assessments to ensure that internal system details and functionalities are not easily discoverable by unauthorized users during runtime.

Method: Test for information leakage and unauthorized discovery attempts.

Expected Result: Verification that internal system details are adequately protected from unauthorized discovery during runtime.

5.8.18 Collection Safeguards Testing:

Rigorous evaluation of safeguards must confirm artificial AI systems prevent unauthorized data harvesting and leaks during live operation to protect privacy and maintain control.

Requirement: Verify that the system has adequate measures to protect against unauthorized data collection and leakage during runtime.

Method: Test data collection safeguards and assess data handling practices.

Expected Result: Assurance that data collection is controlled and protected from unauthorized access or leakage during runtime.

5.9 Additional Testing Specifications

In addition to the above testing specifications for AI Application stack, the following are additional testing specifications which are also very important for AI security.

5.9.1 Supply Chain Vulnerabilities Testing

The following are testing specifications to Identify and mitigate vulnerabilities throughout the LLM application lifecycle.

1. Third-Party Component Assessment :

Requirement: Assess all third-party components, libraries, and dependencies used in the application's supply chain to identify vulnerabilities or security weaknesses.

Method:

Conduct a thorough examination of third-party components in the supply chain, emphasizing their security posture.

Utilize automated vulnerability scanning tools to identify known vulnerabilities in third-party software components within the supply chain.

Perform manual code review and analysis of third-party code within the supply chain to uncover security flaws that might not be detected by automated tools.

Expected Result: A comprehensive report detailing vulnerabilities and security weaknesses in third-party components within the supply chain, along with recommendations for mitigation.

2. Code Review and Analysis :

Requirement: Perform a thorough Software Bill of Materials analysis to bolster security measures. This entails a detailed examination of the third-party code integrated into the supply chain and serves as a crucial step in identifying and mitigating potential risks.

Method: Review static and dynamic code assessment results from a third-party for the third-party code used in the supply chain, along with conducting an SBOM analysis for vulnerability review, third-party dependencies, software composite analysis (SCA), and license review. Utilize static code analysis tools to automatically detect code patterns that indicate security vulnerabilities within the third-party code. If vulnerabilities are identified, follow-up SCA and vulnerability assessments post-remediation should be conducted to verify that the identified issues have been adequately resolved.

Expected Result: A report highlighting identified security flaws in the third-party code used within the supply chain, along with guidance on how to remediate these issues.

3. Dynamic Application Security Testing (DAST) Supply Chain Integration Testing:

Requirement: Employ dynamic testing techniques to assess the security of integrations with third-party components and services within the supply chain during runtime.

Method:

Use automated dynamic testing tools to actively interact with the application and its integrations, including third-party components, to identify vulnerabilities.

Test for issues such as input validation problems, access control issues, and authentication weaknesses within the supply chain integrations.

Expected Result: Detection of security vulnerabilities within the supply chain integrations during runtime, including vulnerabilities that may not be apparent in static code analysis.

4. Software Composition Analysis (SCA Supply Chain Focus):

Requirement: Utilize software composition analysis tools to identify and track open-source components used within the supply chain.

Method:

Use SCA tools to create an inventory of all open-source components and libraries used within the supply chain.

Check the inventory against known vulnerability databases to identify components with security issues within the supply chain.

Expected Result: A complete inventory of open-source components within the supply chain and a list of components with known vulnerabilities and recommended updates.

5. Threat Modeling :

Requirement: Conduct threat modeling exercises to identify potential threats and attack vectors specific to the LLM application's supply chain, with an emphasis on third-party code.

Method:

Collaborate with stakeholders to create threat models that document potential security risks and attack scenarios within the supply chain.

Assess the security posture of third-party components within the supply chain and identify security vulnerabilities.

Expected Result: Threat models that provide a clear understanding of supply chain-specific security risks related to third-party code and a roadmap for addressing these risks.

6. Supply Chain Verification (Third-Party Component Trust):

Requirement: Verify the authenticity and integrity of software components and updates received from suppliers or third-party sources within the supply chain.

Method:

Implement secure update mechanisms, such as digital signatures or checksums, to ensure that third-party components and updates within the supply chain have not been tampered with during transit.

Validating supplier(s) and third-party source(s) identity and security practices to establish supply chain trust.

Expected Result: Confidence in the authenticity and integrity of third-party components and updates received from suppliers or third-party sources within the supply chain.

7. Integration Security Testing :

Requirement: Assess the security of integrations with external systems and services, emphasizing the secure data exchange between the LLM application and third-party entities within the supply chain.

Method:

Conduct penetration testing and vulnerability scanning on integration points within the supply chain to identify potential weaknesses.

Verify that data exchanged with third-party entities within the supply chain is encrypted and securely transmitted.

Expected Result: Secure integrations with external systems and third-party entities within the supply chain, preventing data breaches and unauthorized access.

8. Build and maintain community trust in open-source AI applications.

Requirement: Community Trust in Open-Source AI Applications

Method: Foster a transparent and active community review process for open-source AI applications. Encourage community involvement in validating data use and ethical practices. Stay up to date with security best practices/patches. To the extent possible, publish full or partial assessment reports, such as red teaming results or product reviews.

Expected Results: A high level of community trust and satisfaction with the open-source AI applications, indicated by positive community feedback and engagement, such as repository cloning, forking, followers, and stars, are some objective measurements. Establishing a bar for openness that model publishers are, as peers, expected to meet.

5.9.2 Secure AI Application Development Process

The following are testing specifications for a secure AI application development process.

1. AI Development security Testing:

Requirement: Evaluate the security of the AI development lifecycle through systematic testing.

Method:

Check if secure SDLC practices are in place by verifying that clearly defined security requirements are prioritized and tracked throughout the development process. Ensure that the development team has received appropriate security training and is knowledgeable about secure coding practices and common vulnerabilities. Verify if there is a formal peer review process that considers security aspects during code reviews. Check if static analysis tools are used to scan the codebase for vulnerabilities and if the results are reviewed and addressed. Confirm that threat modeling is performed during the design phase to identify and mitigate potential security risks. Ensure that the team is aware of security risks, conducts regular security testing, follows secure configuration practices, has an incident response plan, and continuously improves their security practices based on lessons learned..

Test data handling practices to ensure data security and privacy are maintained.

Assess algorithmic implementation for security concerns and potential vulnerabilities.

Expected Result: Identification and mitigation of security-related issues in the AI development process.

2. AI Requirements Verification Testing:

Requirement: Assess the adequacy of requirements planning in ensuring AI Applications meet specified benchmarks, ethical guidelines, and compliance standards.

Method:

Test the alignment of AI Application functionality with specified benchmarks and requirements.

Evaluate adherence to ethical guidelines and ethical considerations in AI Application behavior. What is unethical is highly context-dependent and will vary depending upon the specific downstream AI application and the operational environment. Evaluators of this requirement should consider and account for the context during testing.

Verify compliance with relevant regulatory and compliance standards.

Expected Result: Confirmation that AI Application requirements are met, ethical guidelines are followed, and compliance standards are upheld.

3. AI Application Development Integrity Testing:

Requirement: Examine the AI Application development processes for adherence to security and ethical guidelines, especially in continuous learning contexts.

Method:

Test the integrity of AI Application development practices, including continuous learning processes.

Assess the implementation of security measures and ethical considerations in AI Application development. What is unethical is highly context-dependent and will vary depending upon the specific downstream AI application and the operational environment. Evaluators of this requirement should consider and account for the context during testing.

Expected Result: Assurance of AI Application development integrity, with a focus on security and ethics, particularly in continuous learning scenarios.

5.9.3 AI Application Governance Testing

The following are validation and testing specifications for governance in AI Applications.

1. Training and Awareness Evaluation Testing:

Requirement: Assess the effectiveness of training programs in educating individuals about AI-specific risks and secure practices.

Method: Evaluate the content and delivery of training programs related to AI risks and secure practices. Test the knowledge retention and application of participants in real-world AI scenarios. Collect feedback from trained individuals about the weaknesses or issues in the training process.

Expected Result: Confirmation of the effectiveness of training programs in enhancing awareness of AI risks and promoting secure practices.

2. AI Cyber Security Management Assessment Testing:

Requirement: Evaluate the strategies and practices implemented for managing AI cyber security programs.

Method: Assess the cyber security strategies and protocols in place for AI Applications. Test the effectiveness of practices for identifying and mitigating AI-related cyber threats.

Expected Result: Assurance that strategies and practices effectively manage AI cybersecurity, reducing the risk of cyber threats.

3. Leadership and Governance Testing:

Requirement: Test the efficacy of executive leadership in guiding AI security strategies.

Method:

Evaluate the leadership's involvement and commitment to AI security governance.

Test the alignment of executive decisions with AI security goals and ethical considerations.

Expected Result: Confirmation of effective executive leadership and governance in shaping AI security strategies.

4. AI Project Management Audit Testing:

Requirement: Review the management of AI projects for comprehensive security and ethical oversight from inception through deployment.

Method: Audit AI project management practices to ensure security and ethical considerations are integrated from project inception. Assess project documentation and decision-making processes for adherence to security and ethical guidelines.

Expected Result: Assurance that AI projects are managed with a focus on security and ethical oversight throughout their lifecycle.

5. AI Process Audit Testing:

Requirement: Periodically or continuously review the AI processes for comprehensive security and ethical oversight from inception through deployment.

Method: Audit AI processes to ensure security and ethical considerations are integrated from project inception and are done continuously as the application changes. Assess running AI environments for changes that lead can cause issues with adherence to security and ethical guidelines.

Expected Result: Assurance that AI processes are continuously updated as the AI applications change with a focus on AI services, security and ethical oversight throughout their lifecycle.

6. Algorithmic Bias and Fairness Testing:

Requirement: Assess the AI algorithms and models for potential biases and ensure fair and non-discriminatory outcomes.

Method: Test the AI algorithms and models using various datasets and scenarios to identify potential biases or unfair treatment of specific groups or individuals. Evaluate the processes and techniques employed to mitigate algorithmic biases and promote fairness.

Expected Result: Confirmation that the AI algorithms and models are free from significant biases and provide fair and non-discriminatory outcomes.

7. AI Ethics Review Testing:

Requirement: Evaluate the effectiveness of AI ethics review processes in ensuring alignment with ethical principles and guidelines.

Method: Audit the processes and procedures in place for reviewing the ethical implications of AI applications. Test the implementation of ethical guidelines and the decision-making processes for addressing ethical concerns or dilemmas.

Expected Result: Assurance that AI ethics review processes are comprehensive and effective in upholding ethical principles and guidelines.

8. AI Risk Assessment and Mitigation Testing:

Requirement: Assess the processes and practices for identifying, evaluating, and mitigating potential risks associated with AI applications.

Method: Test the risk assessment methodologies and procedures employed for AI applications. Evaluate the effectiveness of risk mitigation strategies and controls implemented to address identified risks.

Expected Result: Confirmation that AI risk assessment and mitigation processes are robust and capable of effectively managing potential risks.

9. AI Incident Response and Recovery Testing:

Requirement: Evaluate the preparedness and effectiveness of incident response and recovery plans for AI-related incidents or disruptions.

Method: Test the incident response and recovery procedures in simulated AI-related incident scenarios. Assess the effectiveness of incident detection, containment, and recovery strategies, as well as communication and reporting protocols.

Expected Result: Assurance that the organization is well-prepared to respond to and recover from AI-related incidents or disruptions in a timely and effective manner.

10. Compliance with AI-related Laws, Regulations, and Industry Standards:

Requirement: Ensure compliance with relevant laws, regulations, and industry standards related to AI applications.

Method: Assess the organization's processes and controls for monitoring and adhering to applicable AI-related laws, regulations, and industry standards. Test the implementation of compliance requirements and the effectiveness of compliance monitoring and reporting mechanisms.

Expected Result: Confirmation that the organization is compliant with relevant AI-related laws, regulations, and industry standards.

11. AI Data Governance:

Requirement: Evaluate the processes and controls for managing and governing data used in AI applications, including data quality, privacy, and security.

Method: Test the data governance practices related to AI applications, including data acquisition, preprocessing, storage, and access controls. Assess the measures implemented to ensure data quality, protect sensitive data, and maintain data integrity.

Expected Result: Assurance that data used in AI applications is governed effectively, ensuring data quality, privacy, and security.

12. AI Model Lifecycle Management:

Requirement: Assess the practices and processes for managing the lifecycle of fine tuned AI models, including development, deployment, monitoring, and updating.

Method: Test the procedures and controls in place for fine tuned AI model development, validation, deployment, and ongoing monitoring and maintenance. Evaluate the processes for updating and retiring AI models based on performance, accuracy, and potential risks.

Expected Result: Confirmation that AI models are effectively managed throughout their lifecycle, ensuring proper development, deployment, monitoring, and update processes.

5.9.4 Secure Model Sharing and Deployment

If the fine tuned model is shared with a third party, please conduct the following testing.

1. Secure Model Sharing and Deployment Testing:

Requirement: Enforce and test strict protocols for securely sharing and deploying AI models, especially for high-risk applications, to ensure the integrity and security of the models remain intact. Make sure model cards are correctly shared with the right details.

Method: Test the model deployment process for adherence to security protocols, including code signing, encryption, model watermarking and access controls. Evaluate the model sharing mechanisms to verify that only authorized entities can access and deploy the models. Conduct penetration testing to identify vulnerabilities in the model deployment infrastructure. Assess the impact of model deployment on the overall system's security posture.

Expected Result: Verification that AI models are securely deployed with integrity and security measures intact. Assurance that only authorized entities can share and deploy models. Identification and mitigation of vulnerabilities in the deployment infrastructure. Confirmation that model deployment does not compromise the overall system's security.

2. Secure Model Versioning and Rollback Testing:

Requirement: Implement and test versioning mechanisms for AI models to allow secure rollbacks in case of issues or vulnerabilities. Make sure model cards are inherited and updated across models.

Method:

Assess the versioning system's ability to maintain historical versions of AI models.

Test the rollback process to ensure it can be initiated securely in response to identified issues.

Conduct simulations of model rollbacks to verify their effectiveness in restoring system integrity.

Expected Result:

Confirmation that AI model versions are securely managed, allowing for secure rollbacks.

Verification that the rollback process can be initiated securely.

Assurance that model rollbacks effectively restore system integrity.

3. Secure Model Monitoring and Alerting Testing:

Requirement: Implement continuous monitoring and alerting mechanisms for deployed AI models to detect anomalies and security threats. Make sure model cards are updated on change.

Method:

Assess the effectiveness of monitoring systems in identifying abnormal model behavior and update model card if changes happen in the model.

Test the change management process or alerting mechanisms to ensure timely notifications of security incidents.

Conduct simulations of security threats to evaluate the monitoring system's response.

Expected Result:

Verification that continuous monitoring detects abnormal model behavior.

Confirmation that alerting mechanisms provide timely notifications of security incidents.

Assurance that the monitoring system responds effectively to security threats.

4. Secure Model Access Control and Permissions Testing:

Requirement: Enforce strict access control and permissions for AI model deployment and usage.

Method:

Evaluate access controls governing AI model deployment and usage.

Test permissions to ensure that only authorized entities can interact with the models.

Expected Result:

Verification that AI model access is controlled, and only authorized entities have appropriate permissions.

5. Secure Model Patching and Updates Testing:

Requirement: Implement and test secure procedures for patching and updating deployed AI models.

Method:

Test the patching process to verify its security and integrity.

Assess the impact of model updates on system security.

Expected Result:

Assurance that patching and updating of AI models are carried out securely without compromising system security.

5.9.5 Transparency in Decision-Making

The following are testing specifications for "Transparency in Decision-Making" to ensure comprehensive evaluation:

1. Transparency in Decision-Making Testing:

Requirement: Provide and evaluate mechanisms that offer insights into the AI decision-making process, especially in high-risk applications as defined in EU AI Act, to maintain accountability and trust. Model cards, data cards or AI application cards are essential for transparency.

Method:

Assess the availability and accessibility of decision-making insights to authorized users. Make sure the right model cards, data cards and AI application cards exist and the data is up to date.

Test the effectiveness of mechanisms that explain AI decisions, including the provision of decision rationale.

Evaluate the comprehensibility and transparency of the information provided about AI decision-making.

Conduct simulations of decision scenarios to verify the accuracy and consistency of decision explanations.

Expected Result:

Verification that mechanisms for insights into AI decision-making are available and accessible.

Confirmation that the provided decision explanations are effective in maintaining accountability and trust.

Assurance that decision rationale is comprehensible and transparent.

Identification and mitigation of issues related to decision explanation accuracy and consistency.

2. Decision Model Auditing and Validation Testing:

Requirement: Implement and test auditing and validation processes for AI decision models to ensure their accuracy and fairness. Make sure the model, data or application cards reflect the same.

Method:

Assess the auditing mechanisms in place for decision models to detect biases and unfairness.

Test the validation procedures to ensure that decision models align with ethical guidelines and compliance standards. Also make sure no bias that affects the application result is undetected.

Conduct simulations of biased decision scenarios to evaluate the effectiveness of auditing and validation.

Expected Result:

Confirmation that auditing and validation processes identify and rectify biases and unfairness in decision models.

Verification that decision models comply with ethical guidelines and standards.

3. User Feedback Integration and Testing:

Requirement: Incorporate mechanisms for user feedback and evaluate their effectiveness in improving decision transparency and fairness.

Method:

Assess the user feedback collection mechanisms.

Test the integration of user feedback into decision-making processes.

Evaluate the impact of user feedback on decision transparency and fairness.

Expected Result:

Assurance that user feedback mechanisms are effective in enhancing decision transparency and fairness.

4. Ethical Decision Impact Assessment Testing:

Requirement: Implement and test mechanisms for assessing the ethical impact of AI decisions in high-risk applications.

Method:

Evaluate the effectiveness of ethical impact assessment mechanisms.

Assess the decision outcomes in terms of ethical considerations.

Expected Result:

Verification that ethical impact assessment mechanisms effectively evaluate the ethical implications of AI decisions.

5. Accountability Mechanism Testing:

Requirement: Implement and test mechanisms to ensure accountability for AI decisions, including tracking and reporting.

Method:

Assess the accountability mechanisms in place, including tracking and reporting processes.

Test the effectiveness of these mechanisms in holding responsible parties accountable for AI decisions.

Expected Result:

Confirmation that accountability mechanisms effectively track and report AI decisions, ensuring responsible parties are held accountable.