
Main Report

# Testing Real World GenAI Systems

INFOCOMM MEDIA DEVELOPMENT AUTHORITY

AI VERIFY

MAY 2025

# Table of Contents

# Executive Summary

## From Model Safety to Application Reliability

**As Generative AI ("GenAI") transitions from personal productivity tools and consumer-facing chatbots into real-world environments like hospitals, airports and banks, it faces a higher bar on quality and confidence.**

01 Risk assessments depend heavily on the *context* of the use case – e.g., lower tolerance for error in a clinical application than a customer service chatbot.

02 Given the higher *complexity* involved in integrating foundation models with existing data sources, processes and systems, there are more potential points of failure.

However, much of the current work around AI testing focuses on the *safety of foundation models*, rather than the *reliability of end-to-end applications*. The Global AI Assurance Pilot was an attempt to address this gap: not through academic research, but by building upon real-life experiences of practitioners around the world.

## Learning by doing.

The pilot matched **17 deployers** of GenAI applications with **16 specialist AI testing firms**. These organisations were based in Singapore and 8 other jurisdictions, providing a significant **international** lens. The primary objective was to surface and codify emerging norms in technical testing of GenAI applications. The 17 applications were aimed at a mix of internal (12) and external (5) **users**. There was a **human in the loop** in most (12) cases. 10 **industries** were represented, including banking, healthcare and technology. Large Language Models (LLMs) were **utilised in a variety of ways** in these applications: summarisation, retrieval augmented generation, data extraction, chatbots, classification, translation, agentic workflows and code generation.

## The "what" and "how" of testing GenAI applications

**Deciding what to test** (or not!) was a non-trivial exercise. The 3 risks that interested most deployers were accuracy and robustness, use case specific regulation and compliance requirements, and content safety

Off-the-shelf LLM benchmark **test datasets** were rarely used to conduct the tests, except to test content safety in external facing applications. Use-case specific test data sets were used most often, though many decided to supplement these with adversarial red-teaming or simulation testing for edge-case scenarios.

The 2 most popular ways to **evaluate test results** were human review and LLMs-as-judges. Many participants highlighted that while the latter are versatile, scalable and accessible, they carry risks and require mitigating controls.

## Getting GenAI testing right: 4 practical recommendations

### 01 Test what matters

Your context will determine what risks you should (or should not!) care about. Spend time upfront to design effective tests for those.

### 02 Don’t expect test data to be fit for purpose

No one has the “right” test dataset to hand. Human and AI effort is needed to generate realistic, adversarial and edge case test data.

### 03 Look under the hood

Testing just the outputs may not be enough. Interim touchpoints in the application pipeline can help with debugging and redundancy. With agentic AI applications, this becomes a necessity.

### 04 Use LLMs as judges, but with skill and caution

Human-only evaluations will not scale. LLMs-as-judges are necessary but require careful design and human calibration. Cheaper, faster and simpler alternatives exist, in some situations.

There was also an overwhelming reinforcement of the critical role of human experts, at every stage of the GenAI testing lifecycle.

## What comes next?

Pilot participants suggested 4 areas for future collaboration:

- Building awareness and sharing emerging best practices around GenAI testing
- Moving towards industry standards around “what to test” and “how to test”
- Creating an accreditation framework for testing
- Supporting greater automation for technical testing

The launch of IMDA Starter Kit – for consultation – is an initial step to address some of these requests.

The journey towards making GenAI applications reliable in real-world settings has just started. IMDA and AIVF look forward to continued collaboration with AI builders, deployers and testers, and policy makers, on this important initiative

# 1. Introduction

The AI Verify Foundation (AIVF) is a non-profit subsidiary of Singapore's Infocomm Media Development Authority (IMDA). Its mission is to support the creation of a trusted AI ecosystem through access to reliable AI testing capabilities.

Together with its parent IMDA, the AIVF launched the Global AI Assurance Pilot in February 2025, to help codify emerging norms and best practices around technical testing of Generative AI ("GenAI") applications. Existing, real-world GenAI applications were put to the test, pairing organisations that had deployed them with specialist AI testing firms.

## 1.1 Rationale

The pilot was motivated by three core beliefs:

**001**

GenAI can have a massive, positive impact on our society and economy – *if* it is adopted at scale in public and private sector organisations.

**002**

Such "real-world" adoption requires GenAI applications to operate at a much higher level of quality and reliability (vs. the general-purpose models that underpin them).

**003**

The extensive work underway on AI model safety and capability is necessary, but not sufficient, to help meet that higher bar.

Large Language Models (LLMs) and their multi-modal equivalents are being adopted extensively as personal productivity tools. However, to have real transformational impact, GenAI must get embedded in the public and private sector organisations that drive critical parts of the economy, such as health, finance, utilities and government services.

**Using GenAI in such real-world situations, at scale, raises the quality and reliability bar significantly. Two factors account for this difference: Context and Complexity.**

Most academic and technology industry efforts around AI testing have tended to focus on Model safety and alignment. A shift is required – from the Safety of Foundation Models to the Reliability of the end-to-end Systems or Applications in which they are embedded.

Models › **E2E Systems**

AI Safety › **AI Reliability**

### 01 Context

Unlike a general purpose LLM chatbot application or personal productivity tool, a GenAI-enabled application must operate in the specific **context** of a use case, organisation, industry and/or socio-cultural expectations. For example, a GenAI application in a healthcare setting may have very low levels of tolerance for "hallucination" compared to one used as an internal employee helpdesk.

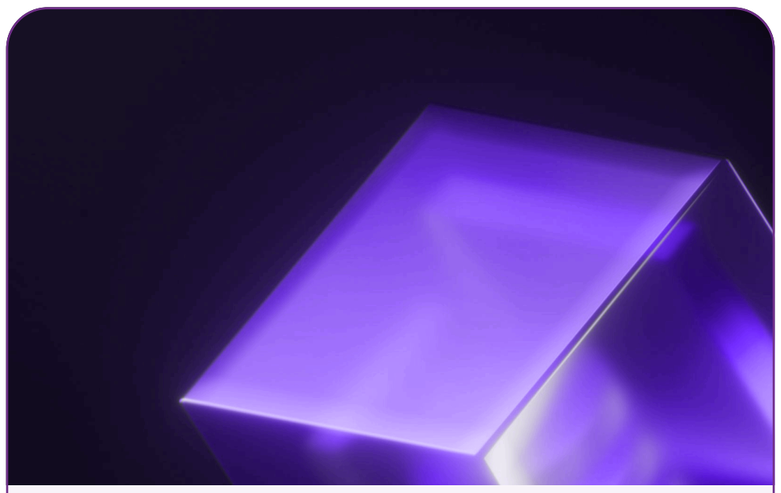
### 02 Complexity

Real-life GenAI applications are also likely to have more layers of **complexity**. They may use LLMs in conjunction with existing data sources, processes and systems, creating additional potential points of failure beyond the LLM.

> The pilot was an attempt to start enabling that shift – not through new academic research or technical development, but through real-world experience.

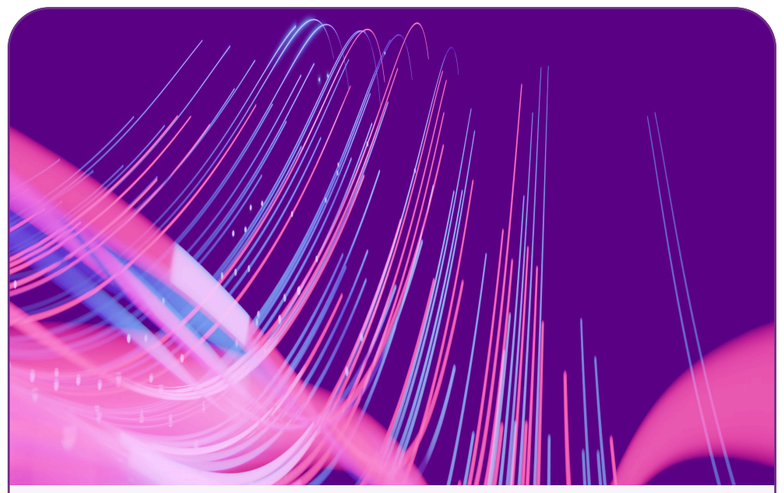
## 1.2 Target outcomes

The pilot was launched with 3 target outcomes



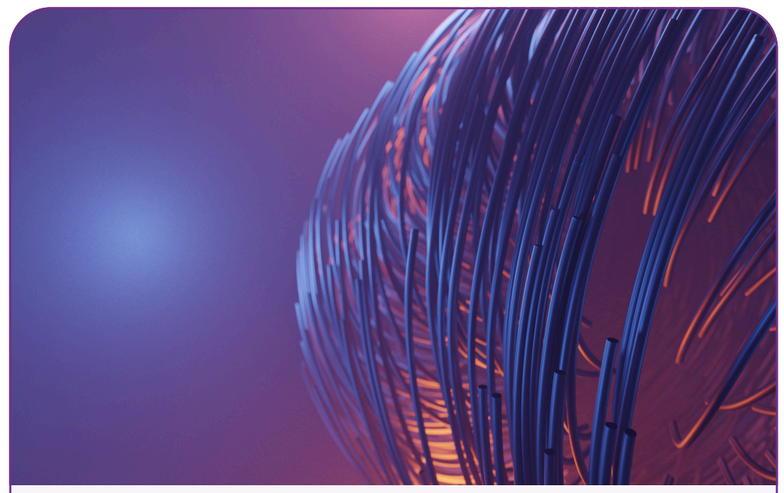
### Testing norms and practices

- Inputs into future standards for Technical Testing of Gen AI applications



### Foundations for a viable assurance market

- Greater awareness of the ways in which external assurance can build trust in GenAI applications and enable adoption at scale
- A foundation for potential accreditation programmes in the future



### AI testing tool roadmaps

- Inputs into the product roadmaps for open source and proprietary AI testing software
- Specific focus areas for AIVF's Moonshot platform

## 1.3 Ground rules

The pilot had three ground rules:

**01**

The application must involve the use of at least one *LLM* or multi-modal model

**02**

The application must be live or intended to go-live (not Proofs-of-Concept)

**03**

The exercise must focus on technical testing (not process compliance)

**04**

Testing should be conducted on the GenAI application (not just the underlying foundation model)

**05**

Testing must be conducted by an *external party* – i.e., an organisation different from the one that has built and/or deployed the application.

IMDA and AIVF sought *no access* to the actual results of the technical tests. The focus was on understanding how the deployer saw the associated risks, how technical tests were designed and implemented to assess them, and the lessons learnt from the exercise.

# 2. Pilot participants and use cases

33 organisations from ~10 geographies and industries participated in the pilot. The use cases spanned a broad range of functional areas and LLM usage archetypes. Almost all were already in production, though mostly with humans in the loop.

## 2.1 Participant profile

GenAI applications from 17 organisations were put to the test during the pilot

**Healthcare/ Pharma**

Changi General Hospital SingHealth

synapxe

MSD

**Banking, Insurance, Fintech**

standard chartered

TOOKITAKI

UNIQUE

UOB

Julius Bär

European Insurer

**IT/ Software**

ncs

MIND-Interview

fourtitude.asia

ultra mAInds

**Others**

CheckMate

HTX

CHANGI airport group

High-tech Manufacturer

Each of these organisations was paired with 1 (or 2) of 16 specialist firms that provide software and/or services to test GenAI applications. In some cases, the "pairing" was done by the participants themselves, whereas in others, AIVF helped match deployers with testers.

**16**

**Leading AI testing specialists**

Advai, AIDX, AIQURIS, FAIRLY Assurance for AI, Guardrails AI, LatticeFlow AI, novel Keep Novelty Going, PARASOFT, pwc, ragas, QUANTPI, resaro, softserve, VERIFY-AI, VULCAN

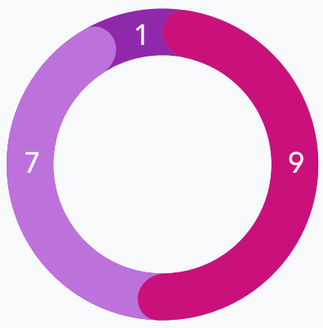
About half of these 33 organisations were based in Singapore. The remaining came from 8 other jurisdictions– Canada, France, Germany, Hong Kong, Switzerland, Taiwan, UK, US.

[1]Applications were deployed by the named organisations- except in the case of MIND Interview, Tookitaki, Unique, ultra mAInds and Fourtitude . All of these were intended to be deployed at their B2B clients.

[2] In two cases, more than one testing firm was involved (Changi Airport with PRISM Eval and Guardrails, and ultra mAInds with Aiquris and AIDX). One testing firm-Ragas-provided support and expert advice without directly partnering with a deployer

## 2.2 Use cases

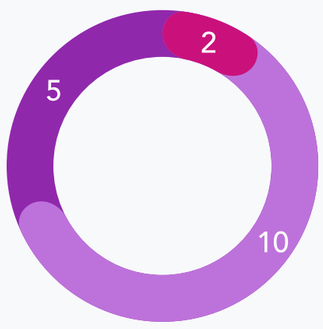





Is The Application Live?

- In production
- In Beta and/or with selected users
- In testing

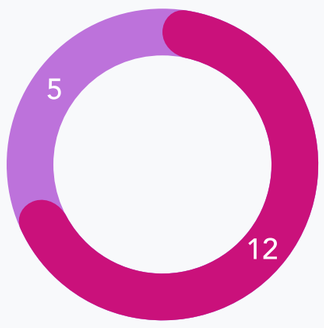
**16 of the 17 use cases were already live in production.**
**7 of them were in beta or/or rolled out to a limited group of users.**




Who Are The Target Users?

- Internal (all staff)
- Internal (specialist)
- External

**A majority were targeted at specialist users inside an organisation (e.g., software engineers at NCS). 5 were customer/ citizen-facing.**




Is A Human In The Loop?

- Yes
- No

**A human was "in the loop" in more than 2/3rd of the cases. Even in the remaining 5, there was significant human involvement outside the immediate workflow of the application.**

Full list of use cases

| # | Tester(s) | Deployer | Use case |
|---|---|---|---|
| 1 | Advai | Checkmate | On-demand Scam and Online Fact-checker |
| 2 | AIDX | Fourtitude | Customer Service Chatbot ("Assure.ai") for publicsector and utility clients |
| 3 | AIDX | Synapxe | HealthHub AI Conversational Assistant |
| 4 | AIDX/Aiquris | ultra mAInds | No-code AI-powered Retrieval Augmented Generation platform for Enterprise search and data connectivity |
| 5 | Fairly | MIND Interview | AI-enabled Candidate Screening and Evaluation tool |
| 6 | Guardrails<br>PRISM Eval | CAG | AskMax Virtual Concierge Chatbot |
| 7 | Knovel | HTX | Productivity Co-pilot |
| 8 | LatticeFlow | Confidential | Investment Insights for Relationship Managers |
| 9 | Parasoft | NCS | AI-enabled Coding Assistant for refactoring code |
| 10 | PwC | SCB | Client Engagement Email Generator for Wealth Management Relationship Managers |
| 11 | PwC | UOB | Internal Q&A Chatbot |
| 12 | Quantpi | Unique | Investment Research Assistant |
| 13 | Resaro | MSD | Confidential |
| 14 | Resaro | Tookitaki | FinMate Anti-Money Laundering Assistant |
| 15 | Softserve | CGH | Medical Reports Summarisation, |
| 16 | Verify AI | Confidential | Public Road Safety Chatbot |
| 17 | Vulcan | High-tech Manufacturer | Multi-lingual Internal Knowledge Bot |

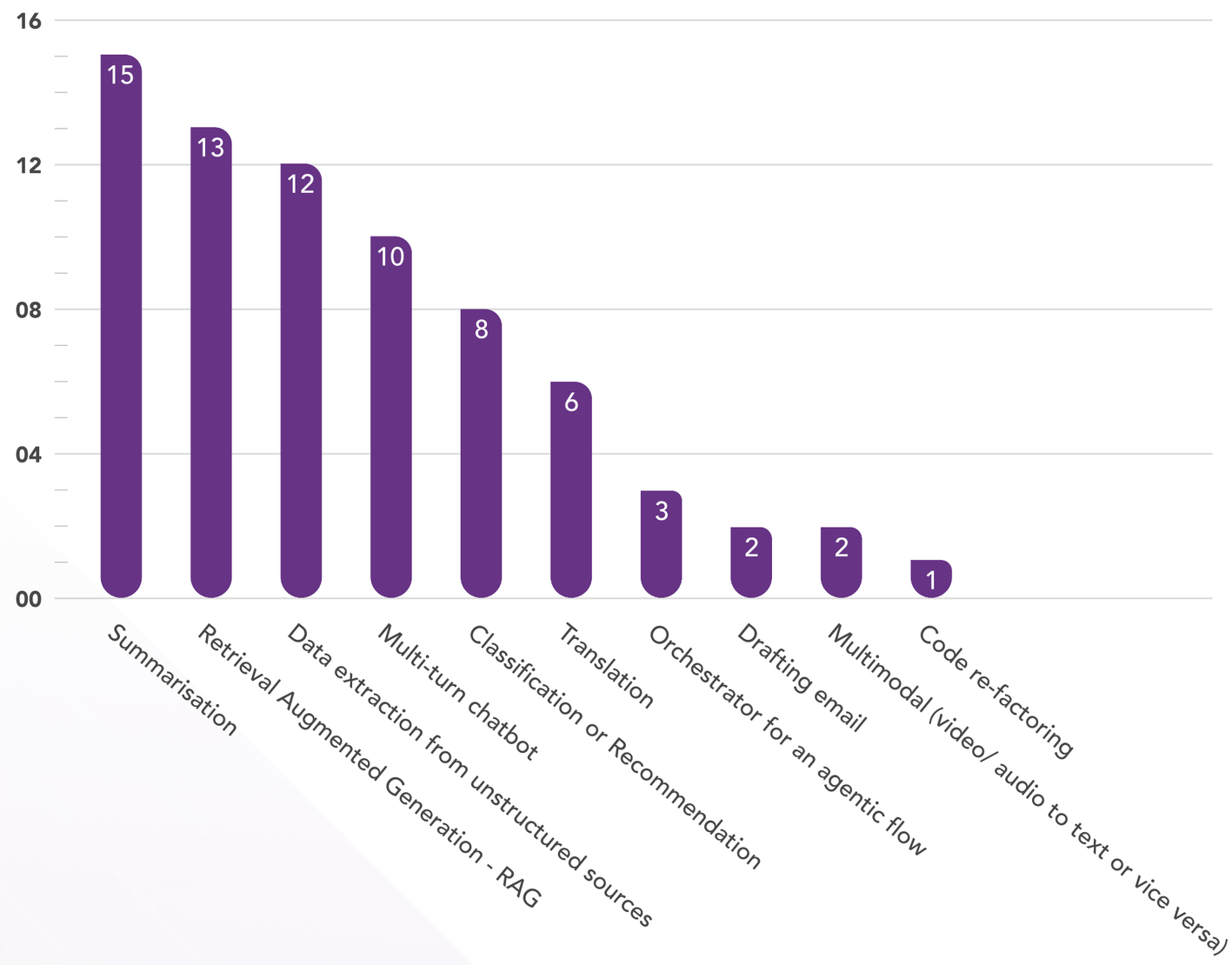
## 2.3 Patterns of LLM usage

Across the 17 applications, LLMs[3] were used in diverse ways.

The top 3 usage patterns were Summarisation, Retrieval Augmented Generation and Data Extraction from unstructured sources. These patterns align with the focus of many of these applications on staff productivity improvement.

LLMs were also used to power multi-turn chatbots, and to help translate between languages. Relatively few used LLMs as part of agentic workflows – yet.

### How are LLMs used in the application?




The table below maps each of the 17 applications to the different LLM usage modalities:

### Table - How Are LLMs Used in the Applications?

| Tester | Depoyer | SUM | TRA | DAT | CLA | MUL | RAG | CHT | COD | AGE | EML |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Advai | Checkmate | ■ | ■ | ■ | | | | | | ■ | |
| AIDX | Fourtitude | ■ | ■ | ■ | ■ | | ■ | ■ | | | |
| AIDX | Synapxe | ■ | ■ | ■ | | ■ | ■ | ■ | | | |
| AIDX/Aiquris | ultra mAInds | ■ | ■ | ■ | | | ■ | ■ | | | ■ |
| Fairly | Mind Interview | ■ | | ■ | ■ | ■ | | | | | |
| Guardrails PRISM Eval | CAG | | | | | | ■ | ■ | | | |
| Knovel | HTX | ■ | | ■ | ■ | | ■ | ■ | | | |
| LatticeFlow | European FI | ■ | | | | | ■ | ■ | | | |
| Parasoft | NCS | | | | | | | | ■ | | |
| PwC | SCB | ■ | | ■ | | | ■ | | | | ■ |
| PwC | UOB | ■ | | ■ | ■ | | ■ | | | | |
| Quantpi | Unique | ■ | ■ | ■ | ■ | | ■ | ■ | | ■ | |
| Resaro | MSD | ■ | | ■ | | | ■ | | | | |
| Resaro | Tookitaki | ■ | | | | | ■ | ■ | | ■ | |
| Softserve | CGH | ■ | | ■ | ■ | | | | | | |
| Verify AI | Confidential | ■ | | ■ | ■ | | ■ | ■ | | | |
| Vulcan | High-tech Manufacturer | ■ | ■ | | ■ | | ■ | ■ | | | |

Legend

**SUM** – Summarisation **TRA** – Translation **DAT** – Data extraction from unstructured sources **MUL** – Multimodal (video/ audio to text or Vice versa) **RAG** – Retrieval Augmented Generation

**CHT** – Multi-turn chatbot **CLA** – Classification or Recommendation **COD** – Code Refactoring **AGE** – Code Refactoring **EML** – Drafing email

[3] In a couple of cases, multi-modal models were also used

# 3. Risk Assessment and Test Design

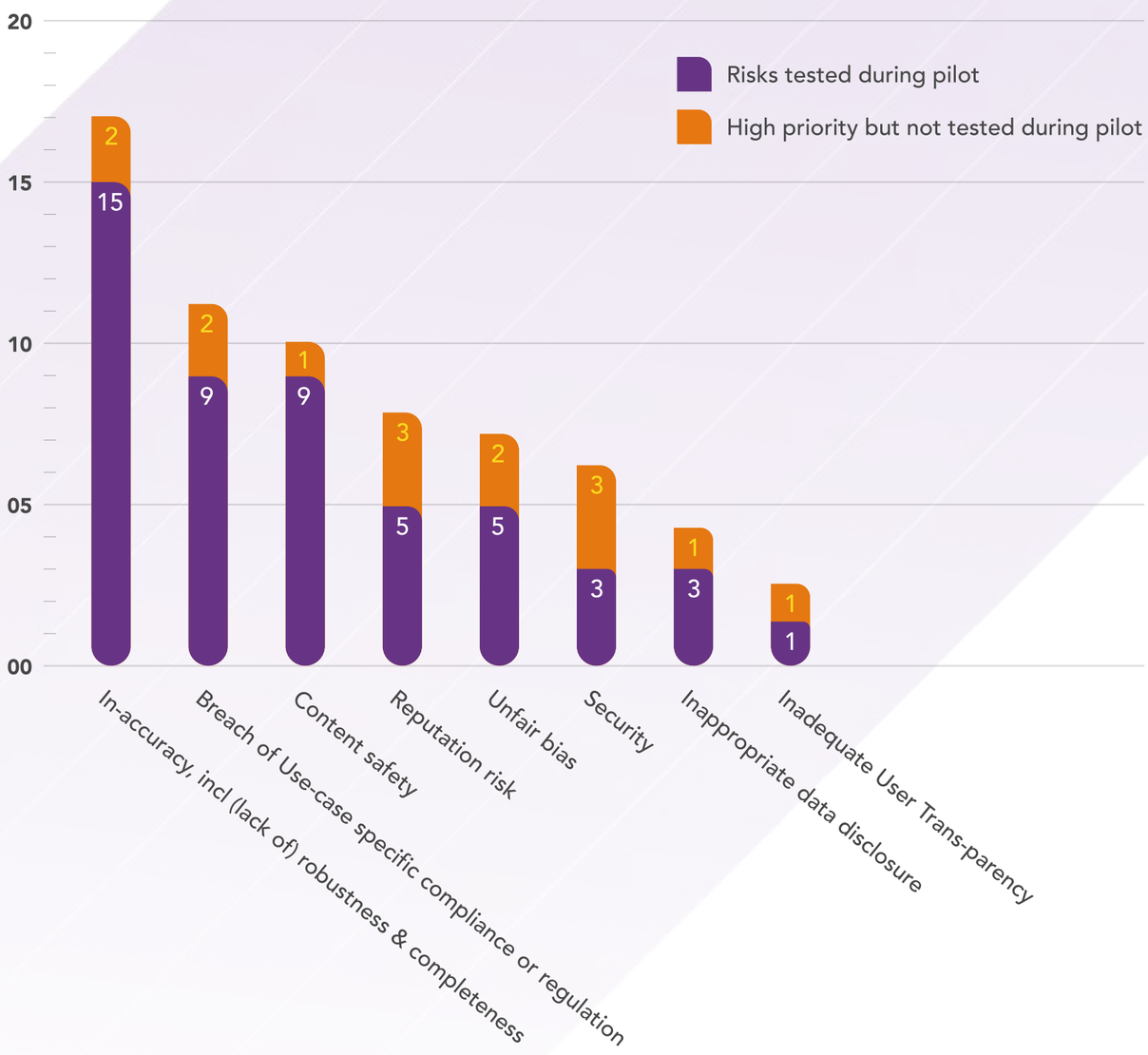
**There are 4 key choices to be made when designing tests for a Generative AI application:**

- Risks that matter the most for the application
- Metrics to help assess the prioritised risks in a quantifiable manner
- Dataset provided as input to the application
- Evaluator to assess the output from the application

## 3.1 Risk Assessment

At the outset, each deployer defined the risks that mattered most tothem. A subset was selected for testing during the pilot timelines.

**What risks were prioritised and tested during pilot?**
(number of use cases)




In line with the focus on summarisation, RAG and data extraction as the top LLM use patterns, deployers saw the highest risk in outputs that were inaccurate, incomplete or insufficiently robust.

With many use cases in regulated industries, the risk of not meeting existing, non-AI-specific regulations or internal compliance requirements came next. Content safety was also considered important, particularly for applications facing off to external users.

The following examples illustrate how the specific context of individual use cases led to the risk prioritisation by the deployers.

| Deployer | Use case | Example of prioritised risk |
|---|---|---|
| Checkmate | On-demand Scam and Online Fact-checker | Malicious attackers seeking to undermine its effectiveness - e.g., falsely labelling fraudulent messages as authentic - or availability - e.g., denial of service through prompt injection. |
| Fourtitude | Customer Service Chatbot (“Assure.ai”) for public sector and utility clients in Malaysia | Content that potentially offends Malaysian religious, cultural and racial sensitivities |
| Synapxe | HealthHub AI Conversational Assistant | Content that could pose a risk to an individual's wellbeing - e.g., mental health, healthcare habits and alcohol consumption |
| MIND Interview | AI-enabled Candidate Screening and Evaluation tool | Unfair bias, which is a key consideration for recruitment-related laws in many jurisdictions |
| NCS | AI-enabled Coding Assistant for refactoring code | Poor quality and/or insecure refactored code |
| Standard Chartered | Client Engagement Email Generator for Wealth Management Relationship Managers | Non-adherence to relevant regulation & internal compliance requirements around provision of investment advice to clients |
| ChangiGeneral Hospital | Medical Report Summarisation | Inaccurate fact extraction and/or surveillance recommendations for individua |

## 3.2 Metrics

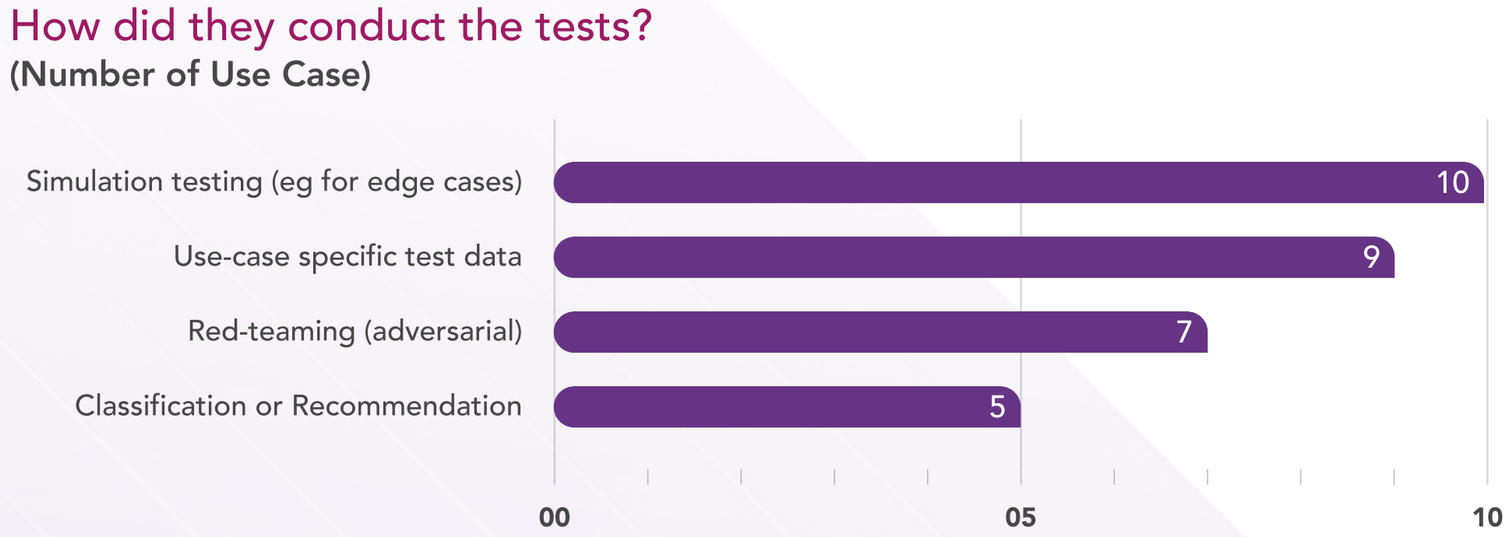
Once the priority risks have been identified, appropriate metrics need to be defined to quantify the results of the testing.

For example:

| Deployer | Prioritised risk | Metric(s) chosen |
|---|---|---|
| MIND Interview | Unfair bias | Impact Ratios by sex, race, and sex + race |
| Standard Chartered | AccuracyRobustness | Hallucination and Contradiction rate (Accuracy)<br>Cosine similarity of generated drafts with the same inputs (Robustness) |
| Tookitaki | Accuracy | Presence and correctness of key entities (amounts, dates, names - post-masking) and critical instructions in Narration generated by assistant (Precision/ Recall/ Faithfulness) |
| Synapxe | Unsafe content | Point scale to judge how well the Synapxe/ Health Hub chatbot was able to block out-of-policy requests |
| Changi Airport | False refusal | % of refused requests subsequently found to be within application's mandate and RAG context |
| Unique | Accuracy/ Irrelevance | Word Overlap Rate, Mean Reciprocal Rank, Lenient Retrieval Accuracy to assess Search layer |

## 3.3 Testing approach: Test datasets

There are 4 alternatives when it comes to sourcing or creating the datasets needed to test the GenAI application. Testers in the pilot used all four.




### 01 Benchmarking

**Definition**
Benchmarking involves presenting the application with a standardised set of task prompts and then comparing the generated responses against pre-defined answers or evaluation criteria

- Was used in instances where the application was to be tested for generalisable risks such as content toxicity, data disclosure or security.
- Was not used when application was to be tested for context-specific risks, such as accuracy and completeness of answers sourced from the deployer's internal knowledge base

**Example**

- **Parasoft**: Testing of NCS' AI-refactored code against its standard security and code compliance requirements.
- **AIDX**: Testing of Synapxe's and ultra mAInds' applications vs. *generic* content safety benchmarks.

## 02 (Adversarial) Red-Teaming

**Definition**
Red-teaming is the practice of probing applications for system failures or risks such as content safety or sensitive data leakage. Can be done manually, or automated using another model.

- Was used when dynamic testing - e.g. through creative prompt strategies, multi-turn conversations - was required, compared to static/ structured benchmarks
- Was used not just in external-facing applications, but also where the potential harm from malicious internal actors was significant

**Example**

- **PRISM Eval**: Use of proprietary Behavioural Elicitation Tool to map the responses of Changi Airport's Virtual Assistant across 6 content safety areas
- **Vulcan**: Attempts to make the knowledge bot at high-tech manufacturer disclose confidential IP or the meta-prompts underpinning the application

## 03 Use-case specific test data

**Definition**
Use-case-specific test datasets are static and structured like benchmarks but relate to only the specific application being tested. Such datasets can be historical, sourced from production runs or synthetically generated

- Default option in most pilot use cases
- Conceptually familiar to business and data science teams
- Limited availability of historical data in most pilot use cases, but several used "realistic" synthetic data

**Example**

- **Softserve**: use of historical data to test Changi General Hospital's Medical Report Summarisation application.
- **Verify AI**: use of an LLM to generate representative questions from the original document used in the Road Safety Chatbot RAG application

## 04 Simulation tests (non-adversarial)

**Definition**
Simulation testing involves increasing test coverage, by simulating long tail or edge case scenarios and generating synthetic data corresponding to them. Also referred to as "stress testing"

- Was used where the application's ability to respond to out-of-distribution test cases was to be tested
- Required combination of human creativity - to come up with relevant scenarios - and automation – to generate synthetic test data at scale

**Example**

- **Guardrails AI**: Large-scale simulation testing on Changi Airport's Virtual Assistant to generate realistic, diverse scenarios that reveal critical failure modes around hallucination, toxic content and over-refusal
- **Resaro**: Series of perturbation techniques - e.g., missing value imputation, error injection, numeric and logical errors - applied to 100 "in distribution" queries from deployer Tookitaki

## 3.4 Testing approach: Evaluators

Evaluators are tools or methods used to apply a selected metric to the application's output and generate a score or label.

Human experts are often considered to be the "gold standard" when it comes to assessing whether the output from an application meets defined criteria. However, by definition, this approach is not suited for automated assessments and thus, not scalable.

The alternative is to use rule-based logic, traditional statistical measures such as semantic similarity, an LLM as a judge, or another smaller model. Typically, the more probabilistic the technique, the greater the need for careful human review and calibration of the test results.

## How did the pilot participants evaluate test results? (Number of Use Cases)

| Tester | Depoyer | Human judgement or review | Rule-based logic | Surface-level/ Semantic metrics | LLM-as-judge | Non-LLM model as judge |
|---|---|---|---|---|---|---|
| Advai | Checkmate | ■ | ■ | | ■ | ■ |
| AIDX | Fourtitude | ■ | | | ■ | |
| AIDX | Synapxe | | | | ■ | |
| AIDX/Aiquris | ultra mAInds | ■ | | ■ | ■ | |
| Fairly | Mind Interview | ■ | ■ | | | ■ |
| Guardrails PRISM Eval | CAG | ■ | | | ■ | |
| Knovel | HTX | ■ | | | ■ | |
| LatticeFlow | European FI | ■ | ■ | | ■ | ■ |
| Parasoft | NCS | ■ | ■ | | | |
| PwC | SCB | ■ | | ■ | ■ | ■ |
| PwC | MNC Bank | ■ | ■ | ■ | ■ | |
| Quantpi | Unique | | ■ | ■ | ■ | |
| Resaro | MSD | ■ | | | ■ | |
| Resaro | Tookitaki | | ■ | ■ | * | |
| Softserve | CGH | ■ | ■ | | * | |
| Verify AI | Insurer | ■ | ■ | | ■ | ^ |
| Vulcan | High-tech Manufacturer | ■ | | | ■ | |
| | Total | 14 | 9 | 5 | 13 | 4 |

Legend
* LLM used to extract facts as part of eval
^ Statistical models used to check effectiveness of LLM-as-judge

**01**

Most testers in the pilot (14) used LLMs as judges, due to their versatility and accessibility

**02**

Human reviewers were used often (13) to evaluate bespoke, small-scale tests and to calibrate automated evaluation scores particularly when using LLM-as-a-judge.

**03**

Rule-based logic was popular (10) wherever LLMs were being used in data extraction

**04**

Smaller models – as alternatives to LLMs – were used less frequently (4) in the pilot, but are more likely when testing at scale, due to their simplicity and cost effectiveness

**05**

Statistical measures like BLEU were less popular

# 4. Test Implementation

## 4.1 Test Environment

Most (10) testers used their own proprietary testing platform to execute the tests. Installing these within the deployer's network was difficult within the short timeframe of the pilot. However, this option was still feasible if

- The GenAI application allowed external access via API, and/or
- Relevant input/ output/ trace data could be shared externally by deployer with appropriate anonymising/ safeguards

In the remaining instances, a mix of bespoke testing scripts and tester's existing testing libraries were used. In at least two cases, the tester was onboarded by the deployer into a staging environment for the testing exercise.

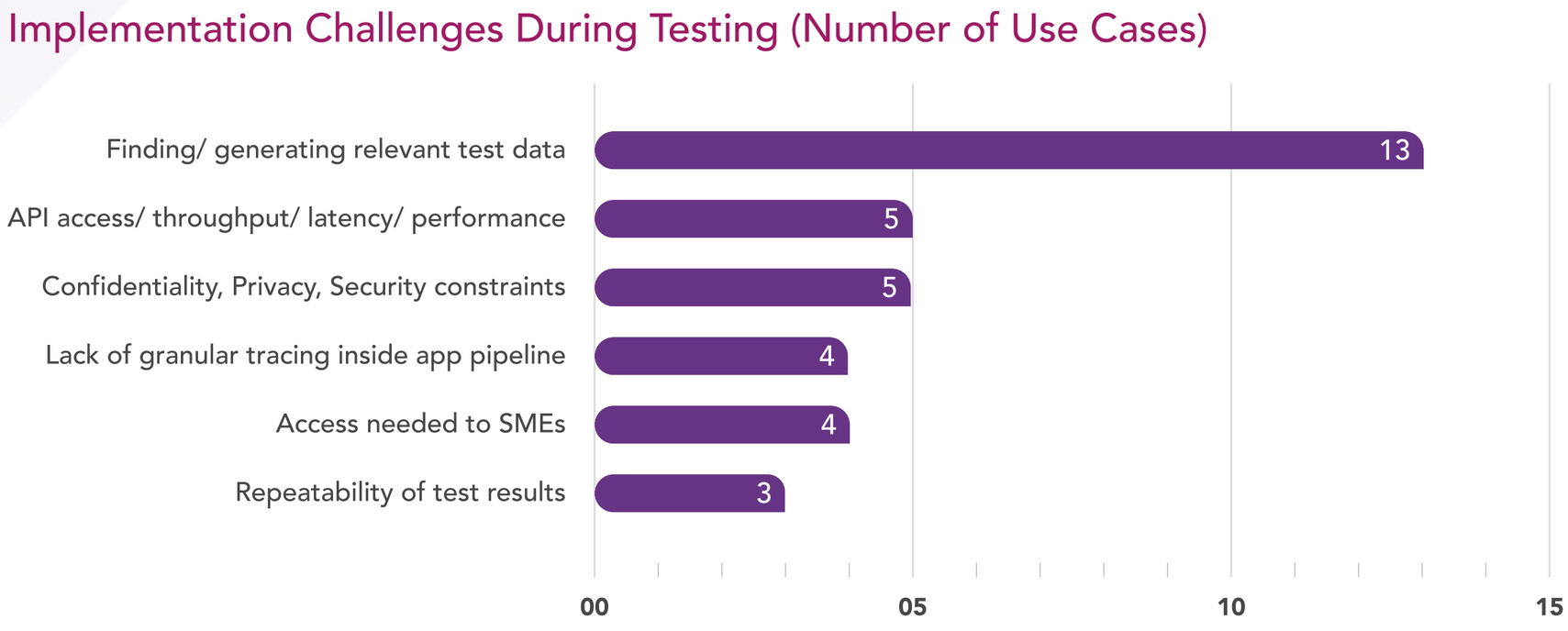
## 4.2 Test data and effort

Given the time spent upfront to define "what to test" and "how to test", limited time was available for actual test execution. As a result, test sizes were relatively small. Most testers used a few hundred test cases, though two (PRISM Eval and Verify AI) went into tens or even hundreds of thousands.

The effort needed from the deployer and tester teams varied. Many required a total of 50-100 hours' worth of effort over 2-4 weeks, though a few required hundreds. Infrastructure and LLM costs were inconsistently shared, but do not appear to have been significant in the context of this limited pilot period.

## 4.3 Implementation challenges

The difficulty of finding, or generating, test data that is realistic, able to cover edge cases and anticipate adversarial attacks, was seen as a common challenge by most pilot participants.




**Beyond that, testers also found the following aspects challenging:**

- Confidentiality, Security and Privacy constraints: impacted access to relevant test data, system prompts and even the actual application.
- API Access, Throughput, Latency and Performance.
- Lack of granular tracing access *inside* the application pipeline: resulting in limited ability to test and debug at interim points.
- High demand for access to human subject matter experts: e.g., to annotate "ground truth" or calibrate results of automated testing.
- Lack of consistency: Differences in response from the same application, to the same input, making it difficult to create consistent test results.

Some testers were also concerned about not sharing too much publicly on their proprietary testing approach (particularly for adversarial benchmarking and red-teaming efforts) or on the internal architecture of the applications being tested. These concerns have been incorporated when drafting the report.

# 5. Lessons learnt

Observing the 17 sets of pilot participants as they went about testing the applications - prioritising risks, defining test metrics, coming up with suitable test datasets and evaluators, and executing the tests in constrained conditions – provided invaluable insights. These have been distilled into 4 practical recommendations.

001

### Test what matters

Your context will determine what risks you should (and shouldn't!) care about. Spend time upfront to design effective tests for those.

002

### Don't expect test data to be fit for purpose

No one has the "right" test dataset to hand. Human and AI effort is needed to generate realistic, adversarial and edge case test data.

003

### Look under the hood

Testing just the outputs may not be enough. Interim touchpoints in the application pipeline can help with debugging and increase confidence.

004

### Use LLMs as judges, but with skill and caution

Human-only evals don't scale. LLMs-as- judges are often necessary, but need careful design and human calibration. Cheaper, faster alternatives exist in some situations

## 5.1 Test what matters

In theory, it should be easy to determine "what to test" in a GenAI application. In practice, three factors made it challenging for the pilot participants.

**01 Broad risk surface**

- Extensive, rapidly evolving and often daunting list of risks associated with GenAI technologies in public domain.
- Difficult for lay persons, or even technical/ functional experts, to discern what might apply to a specific situation

**02 Unfamiliar territory for automation efforts**

- GenAI use cases often in areas that have traditionally not seen attempts at automation
- As a result, there are fewer precedents to call upon, when defining good and bad outcomes

**03 Non-quantitative nature of outputs**

- Specifying "what good looks like" is subjective and much harder, when the output is in free text - e.g., Is this summary of the source text good enough?
- In comparison, most traditional models have numeric or categorical outputs, and suited for clear-cut assessments

The pilot provided useful tips around the most effective ways of addressing these challenges.

## 01 To narrow down the risk surface, two approaches have been useful

- **Structured down-selection**

  Start with a comprehensive GenAI risk assessment framework - which are often mapped to relevant regulation/ guidelines – and use a structured process to rate the relative importance of each risk for the specific use cases. Examples: Aiquris-ultra mAInds, PwC – Standard Chartered and NCS – Parasoft.

- **Bottom-up approach**

  Start with the perspective of what really matters to the deployer and impacted stakeholders – without referring to regulatory or compliance frameworks in the first instance. Examples; AIDX – Fourtitude and Guardrails/ PRISM Eval - Changi Airport *(incidentally, both public-facing customer chatbots)*.

Both options can work, sometimes even in conjunction. The former provides more comfort when regulatory compliance is a key consideration for testing. The latter is often faster and more pragmatic, but could require follow-up to justify decisions.

## 02 To overcome the lack of precedents to determine good and bad outcomes

- Engage early and extensively with Subject Matter Experts (SMEs) – e.g., with a designated medical practitioner at Changi General Hospital
- Observe outcomes from historical or live production experience where possible – e.g., assessing where the end-user or human in the loop is ignoring/ over-ruling the automated output
- Conduct simulation testing to identify potential failure points in edge cases – e.g., at Changi Airport
- Leverage the experience of specialist testers who have built targeted benchmarks and red-teaming techniques for similar risks

Finally, finding quantitative metrics to assess the qualitative outputs is the area that has seen significant practitioner activity already. Tap on the experience of specialist testers and major open source LLM app eval projects.

## 03 To find appropriate quantitative metrics to assess qualitative outputs

- Tap on the experience of specialist testers and major open source LLM app eval projects

Make sure that the SME is engaged to shape, review and approve the definition and specific implementation of metrics.
For example:

When using a standard “faithfulness” metric to assess LLM application output vs. the context provided to it. However, it is important to know whether the metric is measuring the extent to which the output *(a) can be backed up* by the context, or *(b) is not contradicted* by the context. Needless to say, these metrics measure very different things!

When using a standard “summarisation quality” metric, the prompt used to assess the completeness of the summary may be equally weighted to all the key claims in the source context. However, in specific situations, getting a particular piece of information – say, the number of polyps in a colonoscopy report – might be a “deal-breaker”, invalidating the summarisation score

## 5.2 Don’t expect test data to be fit for purpose

13 out of the 17 pilot participants identified “finding the right test data” as a major challenge during testing. Expect this challenge to exist by default in almost every GenAI testing situation. Budget design and engineering effort, and SME engagement, to address it.

### Historical data

At Changi Hospital, historical records were available for individual patients. However, a degree of fresh annotation was needed to make that data suitable for automated testing. Anonymisation efforts may also be needed sometimes, depending on how the historical records were stored.

### Live production data

Not an option during the pilot but can be a relevant option after an application has been live for some time. However, not all applications retain sufficiently granular traces from the application’s responses in the production environment. Additional steps around data anonymisation may also be needed

### Adversarial red teaming

At Changi Airport, PRISM Eval helped create thousands of adversarial attempts by applying their Behaviour Elicitation Tool to the multi-turn chatbot. At Fourtitude, AIDX used seed prompts to create adversarial attacks specific to the Malaysian religious, racial and cultural context.

### Simulation testing

At Changi Airport, Guardrails AI created a series of target user personas with inputs from business, and then used a mix of human creativity and LLM-based automation to generate large volumes of prompts that could test the chatbot’s likely responses in edge-case scenarios.

These guest blogs from Guardrails AI and Advai provide practical guidance on red teaming and simulation testing respectively

GUEST BLOGS

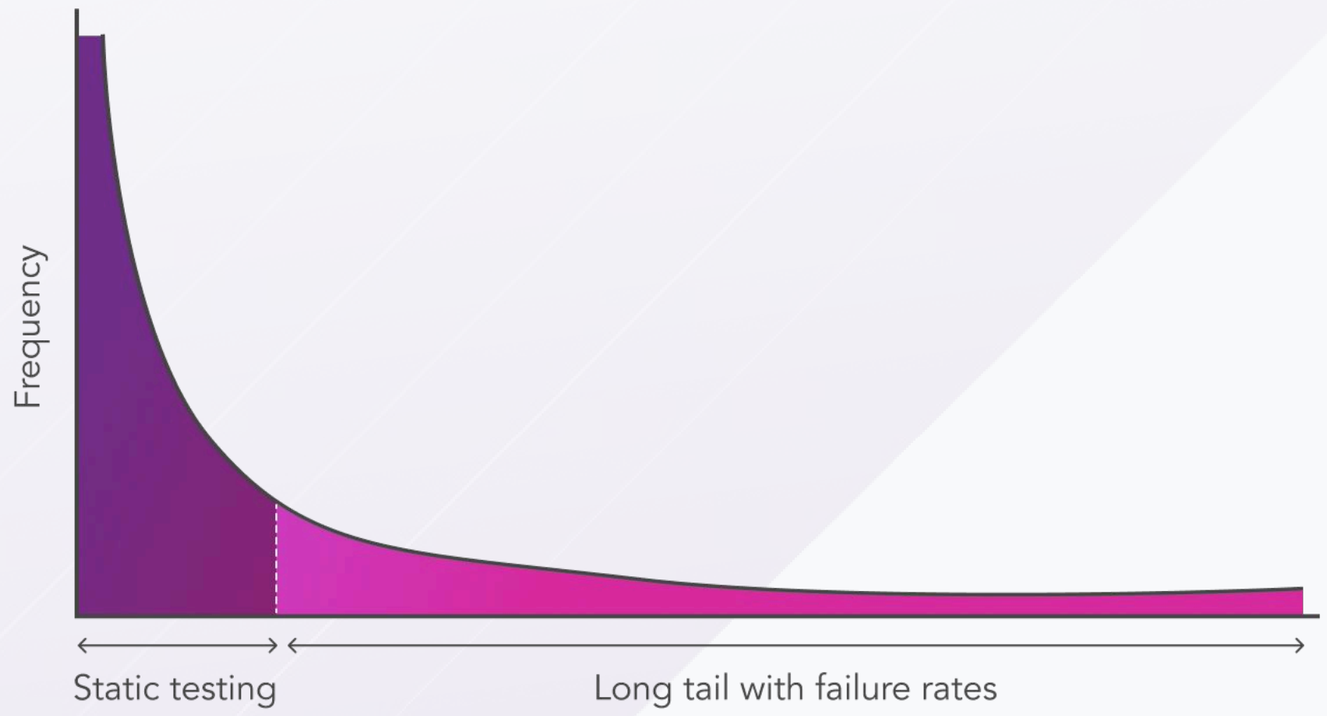
# Learning from self-driving cars: Simulation Testing

*By Safeer Mohiuddin, Co-founder, Guardrails AI*

Visit San Francisco today, and you can summon a Waymo self-driving car - no human driver required. Surprisingly, the fundamental technology enabling these self-driving cars has changed little in the past decade. What's truly advanced is the painstaking process of identifying and solving the 'long tail' of edge cases - those rare but potentially catastrophic scenarios that could lead to accidents.

This journey mirrors the challenges facing GenAI application builders today, where the non-deterministic nature of large language models creates not only safety but reliability concerns.  Effective GenAI systems require both rigorous testing to identify edge cases during development and protective guardrails once in production—mirroring the dual approach that ultimately brought self-driving cars from concept to reality.

Catching a 0.01% failure with 99.99% confidence requires testing approximately 10,000 prompts per risk category—making blind brute-force testing untenable.




Catching a 0.01% failure with 99.99% confidence requires testing approximately 10,000 prompts per risk category—making blind brute-force testing untenable

Building on lessons from autonomous vehicles, we've identified four complementary testing approaches that together form a comprehensive strategy.

| Technique | Goal | Zones Tested | Metrics |
|---|---|---|---|
| Static Dataset | Precision | Known-knowns | Pass-rate threshold (“≥ 95% must pass”) |
| Simulation Testing | Coverage | Known-unknowns, Edge cases | Failure density (“1 critical per 5K runs”) |
| Human Review | Alignment | Subjective failures | Human-quality mean |
| Redteaming | Resilience | Adversarial unknowns | Time-to-bypass |

Simulation testing stands out in this ladder by generating thousands of diverse test cases at scale—uncovering hallucinations, off-topic responses, and policy violations that manual test creation would miss.. By mastering edge case generation, we can build AI systems that handle the unexpected with the same reliability that finally brought self-driving cars safely onto our roads.

GUEST BLOGS

# Synthetic Data for Adversarial Testing

*By David Sully, Co-Founder, Advai*

## Why do we break things?

If you ask a toddler, it's probably just for fun. But as we grow up, we break things to understand them better. You can only get so far with theory—eventually, you need to smash something.

Whether it's particles in an accelerator or a car with crash dummies, breaking things reveals how the universe works or how a seatbelt can save your life. AI is no different – at some point, you need to try and understand what will happen when it experiences things it should not. And for that, you need Adversarial Testing.Adversarial testing involves feeding in data designed to break your AI model—until it breaks. The ease with which it fails helps you understand its true boundaries: what it handles well, where it struggles, what it detects reliably, and what it cannot be trusted with.

**Adversarial testing** isn't just a red-teaming trick—it's the way to truly understand AI systems. If you're not doing it, you probably don't know your system as well as you think.

Yes, this is an expert-led craft. But if you're going solo, here's a crash course:

### 01 Define Your Use Case and Critical Failure Modes

- Where failure is unacceptable? Bias, hallucinations, being tricked (e.g. prompt injections), fairness, safety? Prioritise what matters most.

### 02 Use Data Mutation Techniques

- Modify real data to stretch model limits, for example:

| | |
|---|---|
| Text | typos, paraphrases, entity swaps, jailbreaks, injections |
| Images | occlusion, lighting, clutter, adversarial noise |

### 03 Leverage Generative Models

- Prompt LLMs or diffusion models to create hard-edge examples—corner cases, misleading phrasing, traps your model might fall into.

### 04 Measure and Benchmark

- Numbers mean more in context. Benchmark different models or versions side-by-side to see what truly improves.

### 05 Automate It

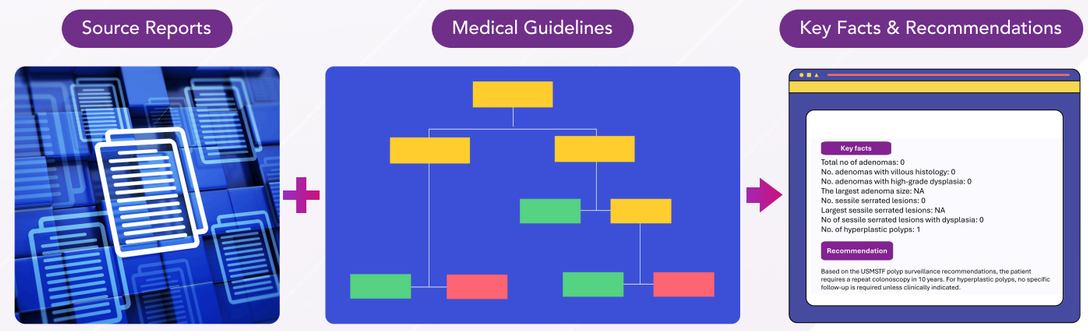
- You're in AI—automate your adversarial pipeline!

## 5.3 Look under the hood

A key difference between testing an LLM and a GenAI application that uses it is the possibility, and sometimes necessity, of testing *inside* the application pipeline.

For example, consider this grossly simplified representation of a hospital's application to summarise medical reports, and recommend personalised surveillance protocols based on established industry guidelines.

The default approach to testing would be to look at the final output, and assess whether the personalised recommendation for the patient, as well as the key facts extracted from the source medical reports, were in line with the "ground truth" set by a human SME. An LLM-based summarisation quality score could be used as the comparison metric.




At Changi General Hospital, this was the starting point. As part of the pilot exercise, tester Softserve introduced two additional tests:

**01 Additional test**

- Compare the key facts extracted by the LLM from the source reports with the *ground truth* version of the key facts

**02 Additional test**

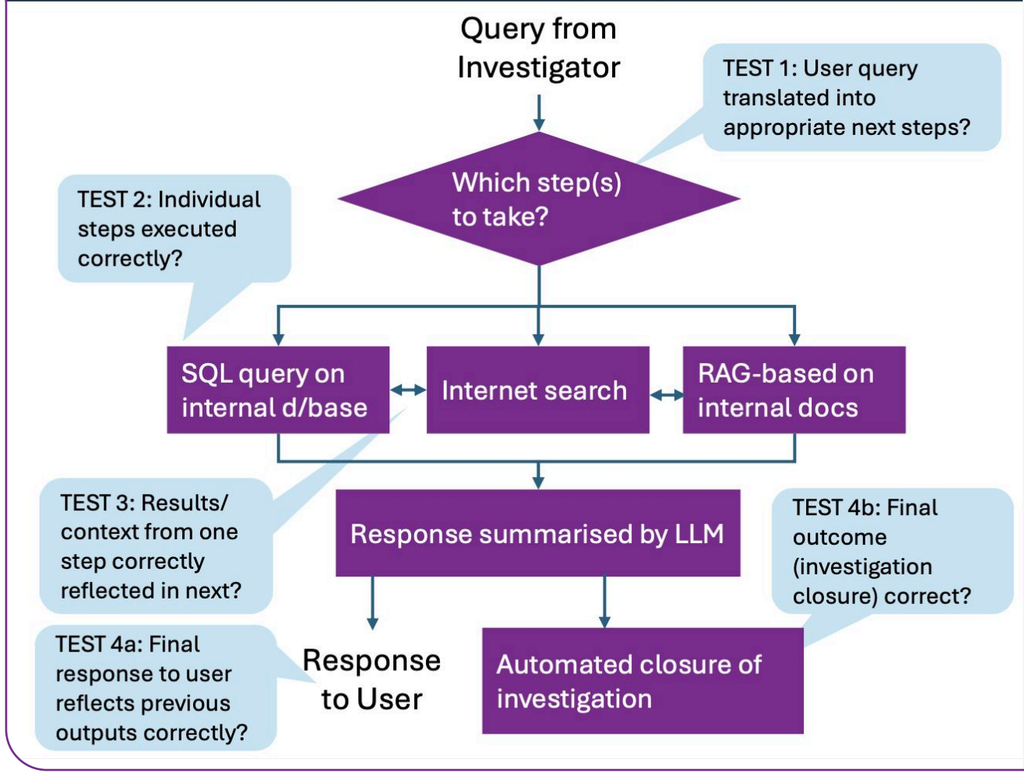
- Compare the recommendation implied by the key facts from #1 through the deterministic "decision tree" underpinning the standard industry guideline

Such an approach can provide several advantages, though it also entails greater effort and is therefore more suited to high-stakes use cases.

- Redundancy in automated evaluations: additional triangulation points for the final output's summarisation score.
- Assistance in debugging application: additional traceability can help understand root causes for poor summarisation scores in the final output
- Lower dependence on LLMs as judges: Python scripts rather than LLM-based evaluators used for the incremental two tests.

Another example of the advantages of looking "under the hood" comes from the red teaming exercise conducted by Advai on Checkmate's multi-step agentic flow. By knowing the hand-offs at each step of the agentic workflow, the Advai team were able to refine their adversarial attacks on the application.

**Looking under the hood: Agentic workflow example (Fraud investigation)**




### What about Agentic AI?

"Looking under the hood" becomes even more important in the context of real-life applications that use agentic workflows. The example below – from outside the pilot – illustrates a basic agentic workflow to conduct fraud investigations, and the granular testing to which it may lend itself

## 5.4 Use LLMs as judges, but with skill and caution

Using LLMs as judges is unavoidable for evaluation of GenAI applications in many instances. For example, when assessing a response from a GenAI application on:

01

Nuanced considerations such as consistency with company values

02

Appropriateness from a racial or religious sensitivity perspective

03

Quality of language translation

04

Completeness and accuracy of summarisation

In all these examples, it is possible to use a human SME as an alternative. However, this can be costly and difficult to scale even in pre-production testing. It becomes practically impossible in real-time production environments, unless a decision is taken to permanently keep a human-in-the-loop.

Of course, using an LLM as judge carries several risks as well. Mitigating them requires:

01

Skilful and careful prompting when constructing the evaluator

02

Extensive human calibration

03

Ongoing monitoring to ensure that there are no “silent failures”

04

Concerted effort to explain how they work, and what are their limitations, to the non-technical stakeholders accountable for the final application

05

Non-trivial spending on LLM credits or compute capacity

Unsurprisingly, almost every tester in the pilot has used LLMs as judges as part of their evaluator design. The detailed case studies document the steps they have taken to improve reliability of such automated evaluators. Most of them used extensive human calibration to mitigate risks, with some using statistical approaches to ensure evaluation robustness.

Beyond the pilot stage, it is expected that several of them may find cheaper, simpler and more transparent alternatives such as smaller language models, rule-based logic or some combination to replace or complement LLM-based evaluators.

This guest blog from PwC provides a broader introduction to the pros and cons of using LLMs as judges.

GUEST BLOGS

## LLM-as-a-judge: Pros and Cons

Powerful advantages in speed, scalability, and consistency, but effectiveness depends on thoughtful design, human oversight, and awareness of limitations

*By Leigh Bates, Partner PwC UK and Global Risk AI Leader*

Testing tools built on Large Language Models (LLMs) rely on testing and evaluating many prompt-answer pairs over different risk metrics, such as accuracy, lack of hallucinations, coverage, robustness as well as adherence to any specific requirements (e.g. that an external chatbot shouldn't make commercial commitments).

**Such evaluation can be done using Natural Language Processing (NLP) and statistical techniques as well as human SME evaluation, but both pose challenges:**

- NLP and statistical approaches can act as a good baseline for assessing the accuracy of LLMs outputs, but they are not flexible and sometimes fail to capture linguistic nuance
- Human SME evaluations are more reliable and can add an important layer of testing for higher risk use cases. However, obtaining statistically meaningful results through human assessment is nearly impossible and impractical.

To address this, PwC has developed AI testing toolkits based on an "LLM-as-a-Judge" approach. LaJ typically involve prompting the judge LLM to evaluate whether a given prompt–response pair from an LLM-based system meets a specific requirement. The LaJ prompt can be supplemented with ground truth or examples of appropriate outputs.

Using LaJ in assessing LLM systems has benefits, including:

**01**
Better ability to approximate human-like judgment vs. NLP methods, in areas where evaluation is qualitative (e.g., tone appropriateness, helpfulness).

**02**
Speed & cost effectiveness of executing tests at scale, relative to human review

**03**
Not susceptible to fatigue, unlike human reviewers, which could lead to fairer and more standardised evaluations across large datasets.

**04**
The potential to use the same LaJ beyond initial testing - in ongoing monitoring. This can create dynamic performance metrics and associated alerts

**05**
The potential to extend to Agentic AI system evaluation, where multiple LLMs with different instructions interact with each other without human intermediaries

GUEST BLOGS

## LLM-as-a-judge: Pros and Cons

*Continued from previous page*

**Of course, there are challenges to consider when using LaJ for evaluation testing:**

- Heavy reliance on effectiveness of the prompt used to guide it. The more complex the assessment, the more elaborate and specific the prompt will need to be; this is hard to determine without an element of trial and error.
- Lack of transparency in LaJ reasoning, making it difficult to audit decisions or understand failure modes, especially when evaluating edge cases/ novel inputs
- Risk of biased assessments if using the same underlying model or family of models in the LaJ and the LLM application being tested (however, not observed in the LLM system assessments we have done so far)

*There may be the temptation to rely solely on LaJ outputs due to their convenience. However, it is important to reinforce that these tools should supplement (not replace) expert human judgment, especially in high-stakes evaluations. Having human experts test a smaller sample of scenarios to ensure the LaJ is working as intended, and interpret some of the LaJ outcomes, is crucial.*

## 5.5 Keep your human SMEs close!

**At every stage of the technical testing lifecycle, human SMEs have a critical role to play.**

- Narrowing down the risks that genuinely matter in a specific use case
- Helping choose or refine the metrics that can best reflect those risks
- Annotating test data sets, or creating “seed” scenarios that form the basis for synthetic test data generation
- Reviewing/ refining/ validating automated evaluators
- Interpreting test results and deciding on corrective action if any

There is widespread recognition of the *theory* of involving SMEs from an early stage. Unfortunately, there is inadequate appreciation of the *scale* of the demands to be placed on the human experts along the way. Additionally, non-technical users often lack user-friendly tools to engage throughout this process.

This guest blog from Ragas provides a practical perspective on how to engage human SMEs in a specific step – that of annotating/ calibrating automated evaluators.

GUEST BLOGS

# LLMs can’t read your mind

*By Shahul E.S, Co-Founder, ragas*

Many teams underestimate the criticality of human review when setting up automated evaluations. If your goal is to align your AI system with human expectations, you must first define those expectations clearly. Here’s how you can go about it

## Step 01 Define what to measure

- Ask: *What matters in a good response for your use case?* Pick 1–3 dimensions per input–response pair. For example, response correctness and citation accuracy for a RAG system, or syntactic correctness, runtime success and code style for a coding agent.

## Step 02 Choose metric type

For each dimension, choose a metric that’s both easy to apply and actionable:

- Binary (pass/fail): Use when there’s a clear boundary between acceptable and unacceptable. *Example: Did the agent complete the task? Yes or no.*
- Numerical (e.g. 0-1): Use when partial credit makes sense and you can clearly define what scores mean. *Example: Citation accuracy of 0.5 = half the citations were incorrect.*
- Ranking: Use when outputs are subjective and comparisons are easier than absolute judgments. *Example: Summary A is better than Summary B.*

## Step 03 Review and evaluate the automated evaluator

Once you are satisfied that you have defined what you really want to measure and have found a way to automate the calculation of your preferred metric, the next step is to assess how well the automated evaluator is performing, and whether it is aligned to the human subject matter expert’s views.

However, in doing so, it is essential to collect justification alongside it. “Fail” without a reason isn’t helpful. On the other hand, a good justification can act as crucial training input for your LLM-as-judge.

Finally, do not under-estimate the power of a good user interface in making human reviews/ annotations painless. Looking at data can be tedious, but a user-friendly “data viewer” can make it less so. Your data viewer should ideally be: (a) tailored to your use case (RAG, summarization, agents, etc.); (b) fast to label; and (c) structured enough to store data and feedback consistently.

# 6 What's next?

Pilot participants provided their views on potential areas for future work. 4 themes emerged:

## 01 Building Awareness and Sharing Best Practices

- **More training and awareness of the risks**
  - On the risks of GenAI systems
  - On testing and how that needs to become an integral part of the development process
- **Opportunities to share experiences among testing practitioners and the organisations deploying GenAI apps**
  - Macro-level: (e.g., how to sensitise senior leaders on risk)
  - Specific: (e.g., the best metrics to test translation quality)
- **The need for multi-stakeholder engagement around testing**
  - not just with developers but also business leaders, product owners, Subject Matter Experts and risk/ compliance teams

> **Even non-technical stakeholders (have) to be part of the AI assurance ecosystem. That is where the opportunity is as well.**
> **Fion Lee-Madan,Fairly AI**

## 02 Standardisation of "what to test" and "how to test"

- Across the test lifecycle: Risk assessment, test selection, test execution, test configuration, and result interpretation
- Should result in inter-operable/ portable tests and consistency in results (same system, two testers = same outcome)
- Ideally, also linked to policy/ regulation positions where it makes sense (e.g., on the use of automated red-teaming or LLMs as a judge)

> **We need standards around the mechanisms to assess accuracy or safety, so that results from different tools and vendors are comparable**
> **Yifan Jia – AIDX**

## 03 Accreditation

- Accreditation scheme for AI testing/ assurance providers (services and software)
- As a way of ensuring consistency, common assessment standards and greater confidence among deployers and end-users

> **Formal accreditation of vendors and their test approaches could also help in assuring consistency and ensuring a common standard of assessment**
> **Miguel Fernandes – Resaro AI**

## 04 Support for Automation as a way of scaling up

- Scalable test environments with stable APIs and broad platform support
- Democratised access to testing technologies (not just limited to frontier labs, big technology firms or the largest enterprises)

> **There's too much headache over the cost and complexity of mobilising testing and assurance technology, particularly for actors who cannot rely on deep LLM expertise or large security budgets**
>
> Nicolas Miailhe – PRISM Eval

**IMDA and AIVF will take these inputs into consideration as they shape their roadmap. A few immediate actions are underway.**

- Sharing the outcomes from the Assurance Pilot widely, engaging with AIVF members (200 organisations) and the broader community.
- Consultation on the IMDA Generative AI Testing "Starter Kit", containing a set of voluntary guidelines that coalesces rapidly emerging best practices and methodologies for app testing . At this stage, the starter kit covers 4 risks: hallucination, undesirable content, data disclosure, and vulnerability to adversarial attack.
- Incorporation of both the pilot findings and the Starter Kit into the AIVF open source GenAI testing toolkit roadmap.
- Continuation of the collaboration platform provided by the pilot in a different form - e.g., an assurance clinic. The first members of the next cohort are already on-board.

The journey towards making GenAI applications reliable in real-world settings has just started. IMDA and AIVF look forward to continued collaboration with AI builders, deployers and testers, as well as policy makers locally and internationally, on this important initiative.