CASE STUDY

HEALTH AND LIFE INSURER TRANSFORMS TEST DATA MANAGEMENT

Overview

A leading health and life insurance provider in the United Kingdom faced mounting challenges in managing test data as it modernized its IT systems. The company had recently transitioned to a microservices-based architecture to enable faster innovation and greater scalability. However, this modernization effort introduced new complexities into their software testing practices, particularly around Test Data Management.

While microservices-based architecture brings many advantages at many levels, testing these systems is incredibly complex for several reasons, including test data. The test data requirements for microservices-based testing are more complex and dynamic than traditional applications due to the distributed nature of the architecture. The chart below highlights how microservices testing differs from traditional database testing.

AREA	TRADITIONAL DATABASE APPLICATION	MICROSERVICES-BASED SYSTEM
Architecture	Monolithic, tightly coupled	Distributed, loosely coupled
Testing Scope	Primarily UI, Backend, and Database interactions	Individual services, interactions, contracts, and messaging
Data Management	Centralized DB (easy to manage)	Decentralized DBs, polyglot persistence
Test Complexity	Low-to-medium; fewer dependencies	High; numerous interdependent microservices
Test Environment Setup	Typically static and easier to configure	Complex orchestration; dynamic environments
Deployment Impact	Less frequent deployments, lower complexity	Frequent deployments, each needing independent tests
Service Dependencies	Limited dependencies	Multiple dependencies, external APIs, message brokers
Performance Testing	Simpler, single DB bottlenecks	Complex; multiple points of failure
Integration Complexity	Moderate, mostly DB-oriented	High; numerous communication channels

Traditional methods—such as pre-loading test databases with anonymized production data—were proving insufficient, slow, and increasingly risky under strict GDPR regulations.

The organization sought a solution that would allow them to accelerate testing cycles, improve test coverage, and maintain full compliance with data privacy standards, all while adapting to the dynamic, decentralized nature of their new architecture.

The Test Data Challenge

As the insurer expanded its services and technology footprint, its testing teams encountered significant obstacles:

Complex Test Data Needs Across Microservices: Each microservice had distinct data requirements, often involving intricate interdependencies. Traditional approaches, like batch loading data into staging environments, were cumbersome and error prone.

Manual Data Creation Bottlenecks: Creating advanced test scenarios—such as those supporting wellness reward programs, lifestyle-based underwriting, and claims adjudication workflows—required extensive manual effort. This process was time-consuming and lacked consistency.

API-Centric Testing Requirements: The microservices model demanded that test data be injected directly into APIs during test execution, rather than being staged in a database ahead of time.

Compliance Risks: Continued reliance on production-like data, even when anonymized, posed GDPR compliance concerns, amplifying the urgency for a safer and more flexible alternative.

The company needed a modern test data solution that was dynamic, scalable, and fully compliant with evolving privacy laws.

The GenRocket Synthetic Data Solution

To address these challenges, the insurer turned to GenRocket—a leading provider of real-time synthetic test data generation technology.

GenRocket's solution was implemented with a focus on dynamic, on-demand data provisioning. Instead of preparing datasets in advance, synthetic data was generated at runtime and injected directly into microservices via their APIs. This was accomplished using GenRocket's RESTAPI Receiver, which seamlessly created and delivered context-aware data at the moment it was needed, eliminating the need for persistent storage in any environment.

At the core of this implementation was GenRocket's unique *Design-Driven Synthetic Data* paradigm—a transformative approach to test data provisioning that empowers testing teams to design exactly the data they need for any scenario. This design-driven strategy ensured synthetic data was not only timely and compliant, but also intelligently aligned to business logic, workflows, and test objectives.

The key elements of GenRocket's synthetic test data deployment included:

Test Data Modeling via Self-Service Test Data Cases: Testing teams used GenRocket's self-service capabilities to design *Test Data Cases* that reflected realistic and comprehensive insurance scenarios. These included:

- Configurations for health and life insurance policies
- Applicant profiles with diverse health attributes and risk classifications
- Eligibility paths for wellness reward programs
- Positive, negative, and edge cases to test business rules and exception handling

Referential Integrity Maintained Across Services: GenRocket's engine ensured that data dependencies across services remained intact without relying on pre-stored datasets, an essential feature for end-to-end workflow testing in a distributed system.

Integration with CI/CD Pipelines: GenRocket's real-time data generation plugs seamlessly into CI/CD workflows, enabling automated and continuous testing. Traditional TDM tools typically struggle in this area, as they often depend on static, pre-generated datasets that don't align well with agile pipelines.

Direct Data Injection into Microservices: GenRocket eliminates the need for intermediate scripts or staging databases by injecting synthetic data directly into microservices at runtime. This is a major contrast to traditional TDM tools, which often rely on complex scripts to extract and load data, introducing extra layers of complexity and risk.

Data Consistency Across Systems: Testing microservices-based systems requires tailored test data strategies that differ from traditional applications. Each microservice manages its own data, so test data must be service-specific yet consistent across services for end-to-end testing. GenRocket is the only TDM tool that natively supports simulating APIs and publishing to MQs as part of test data delivery pipelines. And the platform is very easy to scale horizontally, ideal for distributed, containerized environments.

A Successful Outcome

The introduction of GenRocket revolutionized the company's test data management strategy.

Test data that once took days to prepare was now provisioned in seconds, tailored exactly to each test's requirements and injected directly into the appropriate microservice.
The QA teams could easily simulate a wide variety of real-world insurance scenarios, while automation engineers integrated dynamic data generation seamlessly into CI/CD pipelines, greatly accelerating software release cycles.

Importantly, the company achieved full GDPR compliance by using completely synthetic, non-identifiable data throughout the testing process—eliminating the need to mask or subset production data.

Key Benefits Realized

No Data Storage Required: Test data is generated on demand and delivered at runtime, with no persistence in databases or staging environments.

Faster Test Cycles: Data provisioning time was reduced from days to minutes, significantly accelerating test execution and feedback loops.

Higher Test Coverage: The ability to model positive, negative, and boundary conditions enriched test coverage and improved defect detection earlier in the SDLC.

GDPR Compliance: By eliminating the use of production data, the solution fully adheres to GDPR requirements, reducing legal and security risks.

Scalability and Flexibility: The dynamic data generation model easily scales across new microservices as the insurer's technology ecosystem grows.

Take-Aways

By adopting GenRocket's synthetic data generation platform, this UK-based health and life insurer successfully transformed its approach to Test Data Management. In doing so, they not only modernized their testing practices to match the demands of a microservices architecture but also improved compliance, increased testing efficiency, and empowered their DevOps initiatives.

By generating synthetic data dynamically and tying it directly to the specific test case being executed, the insurer was able to overcome the challenges of API-driven microservices testing without compromising data privacy or quality.

With GenRocket's *Design-Driven Synthetic Data* at the foundation of its test data strategy, the company now views test data not as a bottleneck, but as a strategic enabler for faster innovation, higher software quality, and greater agility in the competitive insurance market.