zensar

Multinational Insurance Leader Transitions From Uncertainty to Being Future-ready

Case Study

Overview

Modernizing a legacy app

A premium African insurance group, catering to customers in 14 countries across the continent, was using a medical fee application to

- maintain details pertaining to medical practitioners,
- manage tariff and test codes,
- capture and validate tests manually,
- upload tests in bulk using batch jobs,
- make payments to doctors and pathologists,
- generate correspondence of payments made to practitioners, and
- produce accounting and management information system (MIS) reports.

As the system was built on ObjectStar, a legacy technology stack, the company was facing a whole host of challenges: high maintenance cost, lost-opportunity cost, compliance and maintenance issues, insufficient organizational agility and efficiency,

difficulty in handling peak loads, scalability limitations, lack of integration with other systems, complexity in the code base, significant risk of downtime during updates, and risk of security breaches.

Moreover, the ObjectStar vendor had declared the platform would no longer be supported, as it was reaching end of life. This meant that there would be no support provided for the ObjectStar platform in the event of a software failure or bug.

Zensar's brief:

- **Rebuild:** Redesign or rewrite the application component from scratch while preserving its scope and specifications.
- **Re-host:** Redeploy the application component to Amazon Web Services (AWS) cloud.

Beyond the brief:

We adopted a test-driven development model to avoid rework and a multi-agile execution model to align with how the interfacing systems work. In addition, we descoped non-used functionality to avoid redundant features and code and delivered self-service capabilities.

Challenges

Inefficiencies and uncertainty

The IT department was dealing with an aging workforce that was close to retirement, scarcity of skills, tacit knowledge, lack of documentation, cost overheads with an unrationalized portfolio, and the risk of business disruption involved in modernizing ObjectStar applications. And to top it off, there was a lack of clarity on vendor support from ObjectStar after 2020.

Solution

Test-driven development strategy

Approach

We had a unique situation where we did not have a business SPOC who could provide end-to-end requirements. So, we took this solution approach:

- Reverse engineering by performing technical analysis of the existing application and validation of the same from user groups during 8-10 discovery workshops
- Division of the application into two parts: online application and back-end processing
- Creation of an AWS-native architecture, leveraging AWS best practices
- Thorough testing strategy involving 800+ test cases and data migration
- Deployment strategy covering technical and production go-live to minimize impact
- Involvement of business users at every sprint to get feedback during the early stages

Execution

Step 1: Requirement analysis

We conducted an application discovery workshop to identify the challenges and risks associated with the legacy application process. The discovery involved a thorough review of the existing system's codebase, architecture, and data storage. Based on the outcomes of the discovery, we developed a solution plan, using a modernization approach.

Step 2: Design and development

Leveraging modern technologies and practices, we designed an AWS system architecture that's scalable, flexible, and easy to maintain. We then developed a cloud-native application, factoring in secure login, online screens, batch jobs, API integrations, and business logic. Finally, we enabled integration with external systems and seamless data exchange between the new and old medical fee systems.

Step 3: Deployment, data migration, and testing

We deployed the new system code on three different environments on the cloud, before migrating the legacy system's data to the cloud. As the legacy system had a large amount of data and limited support from the cloud service provider, we used open-source tools and techniques to ensure the data was migrated accurately.

Next, we conducted business user testing to ensure that it met all the requirements. We also conducted systems integration testing, performance testing, security testing, and disaster recovery testing to ensure the system could handle increased loads during peak times.

Step 4: Phase-wise go-live

- **Phase 1:** Authorization and authentication by a secure server
- **Phase 2:** Technical release
- **Phase 3:** Product release

Solution enablers

- **PING/active directory (AD)** was used for configuring, authenticating, and providing authentication details of the logged-in user.
- **Amazon API Gateway** was used for all REST (JSON) endpoints directly accessible to web, mobile, and other channels.
- **Amazon Elastic Load Balancing (ELB)** was used to automatically distribute incoming application traffic across multiple targets and virtual appliances in one or more availability zones (AZs).
- **Angular 13.x and Node 16.x** was used to develop a single-page application.
- J**ava 11, Spring boot 2.x, Microservices, Amazon Elastic Container Service (ECS), and Spring Batch** was used for all new services to be developed for any channel or for migrating existing services — where services can be containerized and the containers can be managed through ECS.
- **Amazon Elastic Compute Cloud (EC2)** was used to provide secure, resizable compute capacity in the cloud and make web-scale cloud computing easier for developers.
- **Amazon Relational Database Service (RDS) – PostgreSQL** (transactions database) was used to act like a write cache and hold the transactions from the digital platform, before sending to the respective business systems.
- **Amazon CloudTrail** was used to provide visibility into user activities by recording actions taken on your account.
- **Amazon Batch** was used to enable developers and engineers to run hundreds of thousands of batch computing jobs on AWS easily and efficiently.
- **Amazon CloudWatch** was used to monitor the complete stack (applications, infrastructure, network, and services) and use alarms, logs, and events data to take automated actions and reduce mean time to resolution (MTTR).

Impact

A future-ready IT ecosystem

- 40 percent reduction in CAPEX
- 20 percent improvement in batch runs
- 37 percent lower operational costs

Business outcomes:

The legacy-modernization solution enabled greater user experience, scalability, operational efficiency, and cost -effectiveness. All of this resulted in heightened productivity and future-readiness.

zensar

An RPG Company

At Zensar, we're 'experience-led everything.' We are committed to conceptualizing, designing, engineering, marketing, and managing digital solutions and experiences for over 145 leading enterprises. Using our 3Es of experience, engineering, and engagement, we harness the power of technology, creativity, and insight to deliver impact.

Part of the $4.8 billion RPG Group, we are headquartered in Pune, India. Our 10,000+ employees work across 30+ locations worldwide, including Milpitas, Seattle, Princeton, Cape Town, London, Zurich, Singapore, and Mexico City.

For more information, please contact: **info@zensar.com | www.zensar.com**