How a state government modernized its payroll and pension mainframe on Microsoft Azure

Learn how a state government agency partnered with OZ Digital to modernize its mission-critical payroll and pension system supporting millions of people.

The Challenge

A state government's legacy mainframe system, supporting payroll and pension processing, that millions of people depend on had reached a breaking point. Over the years, it had spawned a highly complex environment: millions of lines of code, tightly woven dependencies, and thousands of batch jobs running rigid workflows. The technical debt wasn't just old code, it was an entire ecosystem: Natural and Adabas applications, JCL-based batch processing, and very little usable documentation. Much of the business logic existed only in source code and in the heads of a few long-tenured SMEs. That combination creates fragility. Even small changes become slow, risky, and expensive, which compounds over time.

At the same time, it was critical to statewide operations, so the stakes are different. If a consumer app goes down, it's annoying. If payroll or pensions go down, it's a crisis. These systems support legal obligations, labor agreements, and public trust. So modernization here wasn't about chasing innovation but about reducing risk while keeping the lights on.

> It's tempting to say, "If it's working, don't touch it. But doing nothing is still a decision and usually the riskiest one. Costs continue to rise, skills become harder to find, and every policy or regulatory change takes longer to implement. Over time, the system becomes less resilient, not more. This is how stable systems quietly become single points of failure."
>
> — Chris Pietschmann
> Director - Practice Leader, Digital & App Innovation, and 14-time Microsoft MVP
> OZ Digital

Technologies deployed

- Microsoft Azure
- Azure Kubernetes Service (AKS)
- Azure SQL/Managed Instance
- Azure API Management
- Azure Functions
- Logic Apps
- Azure Service Bus
- Azure DevOps
- Azure Monitor & Defender
- Azure Sentinel
- Microsoft Entra ID
- GitHub Copilot

The Solution

OZ Digital led a full-scale modernization of the legacy system, moving beyond the traditional lift-and-shift to a complete refactor into an Azure cloud-native architecture.

Microsoft and Azure matter here because they support incremental modernization. This wasn't a rip-and-replace. Azure provides enterprise-grade security, compliance, and operational tooling, while GitHub Enterprise gives us governance, auditability, and controlled AI usage. The goal was not just to modernize code but to modernize safely, transparently, and in a way that government stakeholders could trust.

GitHub Copilot was not asked to "rewrite the mainframe." Nor "push button modernization," or Not replace SMEs and developers.

Instead, it acted as a collaborator helping explain legacy code, surface patterns, generate transformations, and assist with testing. The SMEs stayed in control. Copilot simply reduced friction and accelerated understanding, which is where modernization efforts usually stall.

Another big challenge we overcame was the lack of documentation. Business rules had evolved over decades and were embedded directly in code and screen flows. Without understanding those rules, modernization is guesswork. Traditionally, this phase alone can take years. But AI became a practical accelerator speeding things up.

Step 1: CoPilot to analyze code

In this phase, mainframe SMEs paired with GitHub Copilot to analyze Natural programs directly. Copilot helped explain what the code was doing in plain language, identify business rules, and surface dependencies. This created living documentation derived from the source itself. Engineers and executives alike could finally see how the system actually worked—not how it was assumed to work.

Step 2: Logic transformation

Instead of hand-rewriting thousands of programs, our SMEs used Copilot to help build Python-based tooling that translated Natural logic into C#. This wasn't blind automation. The AI assisted with pattern recognition and mapping, while SMEs validated correctness. The result was a repeatable, auditable conversion process that dramatically reduced manual effort and risk.

Step 3: Batch modernization

Batch processing is often overlooked, but it's critical in payroll systems. We used Copilot-assisted tooling to analyze JCL workflows and convert them into Python and C# orchestration logic. This allowed batch jobs to run in Azure with modern scheduling, logging, and error handling without changing underlying business outcomes.

Step 4: UI transformation

Many users still interacted with the system through emulator screens. Using AI-generated tooling, we analyzed screen flows and mapped them to modern web-based interfaces. The goal wasn't cosmetic—it was functional equivalence with better usability. Users kept the same workflows, but the system became more accessible, maintainable, and extensible.

Step 5: AI-assisted functional testing

Testing is where modernization efforts often fail. Copilot was used to generate functional test cases that compared mainframe outputs with Azure-based outputs. Payroll calculations, pension disbursements, and exception handling were validated side by side. This is how confidence was built through proof.

The modern architecture separated concerns cleanly. Web interfaces were decoupled from business logic, which ran in C# services. Data moved to Azure SQL and managed services, and integration relied on messaging rather than tight coupling. This architecture improved resilience, scalability, and observability without changing the business rules.

There was no big bang. Components were modernized incrementally and run in parallel with the mainframe. Canary releases, rollback mechanisms, and side-by-side comparisons ensured safety. This approach allowed modernization to proceed without risking payroll or pension operations, which is non-negotiable in government systems.

AI use was fully governed used as a controlled, auditable tool within an approved framework. GitHub Enterprise ensured code isolation, access control, and auditability while Azure Policy and Defender provided security posture management that aligned with NIST and state regulatory requirements.

Technologies leveraged

We leveraged the following Azure-native services to enable environment isolation (dev, test, prod), zero-trust security, and a flexible, modern foundation.

- **Azure Kubernetes Service (AKS)** for containerized microservices with autoscaling
- **Azure SQL Managed Instance** for enterprise-grade data management with high availability
- **Azure API Management** for secure API exposure and governance
- **Azure Front Door & Application Gateway** for global load balancing and web application security
- **Azure Functions & Logic Apps** to replace legacy batch processing
- **Azure Service Bus & Queue Storage** for event-driven orchestration
- **Azure DevOps** for automated CI/CD pipelines
- **Azure Monitor & Sentinel** for full observability and threat detection

Impact

Our client is already seeing results: development cycles have shortened. Documentation finally exists. Their dependency on a shrinking pool of mainframe specialists has also decreased.
More importantly, the state has gained a platform that can adapt to future policy changes without increasing risk.
Here's what changed:

- **Faster modernization cycles:** the modernization has enabled greater developer productivity and innovation by adopting a modern architecture based on microservices and APIs, accelerating release cycles and improving the state's ability to deliver new features and enhancements.
- **Scalability:** By transitioning from a fixed-capacity mainframe to an elastic cloud model, resulting in improved system responsiveness improved alongside the ability to handle peak demand efficiently.
- **Operational efficiency:** There were efficiency gains through automated deployments and streamlined infrastructure management, replacing manual batch workflows with event-driven processing.
- **Reduced reliance on niche skills:** A shrinking team of mainframe specialists was making it harder to manage these systems. By modernizing these systems, these dependencies have reduced.
- **Security and compliance:** Security and compliance were strengthened through the integration of Microsoft Entra ID, role-based access control, and Azure Key Vault, alongside centralized monitoring and SIEM capabilities enabled by Azure Sentinel which aligns with NIST and state regulatory requirements.
- **Improved test coverage:** Testing is where modernization efforts often fail. Copilot was used to generate functional test cases that compared mainframe outputs with Azure-based outputs. Payroll calculations, pension disbursements, and exception handling were validated side by side. This is how we built confidence through proof and not mere promises
- **Reliability:** The solution also improved reliability and availability by introducing built-in failover and disaster recovery mechanisms, reducing the risk of downtime for critical payroll operations.
- **Clear documentation:** Created for the first time in decades. Copilot helped explain what the code was doing in plain language, identify business rules, and surface dependencies. This created living documentation derived from the source enabling executives to finally see how the system actually worked, not assume how it worked.

OZ Digital will continue supporting the platform post-launch, ensuring there's continuous optimization, security compliance, and long-term value realization.