HOW WE DROVE ENTERPRISE REINVENTION FOR ONE OF THE WORLD'S LARGEST BANKS

One of the largest in the world, serves over 120 million customers across its extensive network of branches, ATMs, and digital channels. Known for its innovation in retail and corporate banking, this bank was keen to address challenges stemming from its existing architecture. The bank's growing digital footprint highlighted performance bottlenecks, dependency on proprietary systems, and operational inefficiencies that hindered its ability to innovate and scale rapidly.

BANKING IN THE DIGITAL AGE

Modern banking operates at the crossroads of finance and technology. The Core Banking System (CBS) acts as the backbone of a bank's operations, facilitating real-time transactions across branches, ATMs, mobile banking, and net banking channels. While traditional CBS frameworks were designed to ensure reliability and centralization, the exponential growth in customer expectations, transaction volumes, and the rise of fintech innovation have exposed the limitations of monolithic architectures.

As customers demand faster, seamless, and more personalized services, banks worldwide are rethinking their technology stacks to embrace flexibility, scalability, and cost-efficiency. Open-source solutions, microservices, and event-driven architectures are becoming pivotal in enabling these transformations.

MONOLITHIC SYSTEM CONSTRAINTS

A monolithic system is characterized by tightly coupled components, meaning all functions within the system are interconnected. This architecture has several drawbacks:

Risk of System-Wide Disruptions

Since components are interdependent, updating or deploying even a small feature affects the entire system. For example, introducing a new feature in mobile banking could unintentionally impact net banking or other channels.

Bottlenecks During High-Volume Transactions

A surge in transaction volumes can overload certain parts of the system, slowing down or even halting services for users.

Illustrative Example

Imagine a customer who uses a bank primarily for account-related services, such as checking balances or transferring money. These services operate through one part of the system. However, if the bank's deposit service (responsible for handling deposits) experiences downtime, the entire system may slow down.

Why? In a monolithic architecture, all components share resources and dependencies. Even though the customer is not directly using the deposit service, its failure creates a ripple effect, impacting unrelated services. This results in frustration for customers and operational challenges for the bank.

PERFORMANCE AND SCALABILITY ISSUES

The centralized, read-heavy system struggled to handle increasing transaction volumes, causing delays in processes like approvals and impacting customer experiences.

Illustrative Example

In a typical database system, operations are divided into two categories: reading (viewing or retrieving information) and writing (updating or adding new information). Imagine a scenario where 90% of the traffic is from users reading data, such as checking account balances, while only 10% involves writing, like updating transaction records.

When too many users simultaneously read data, the system prioritizes these requests, leaving insufficient resources for writing or updating data. Over time, this imbalance creates congestion, causing significant slowdowns in both reading and writing operations. Ultimately, the entire system performance degrades, affecting critical processes like transaction approvals.

KEY CHALLENGES

CONFIGURATION MANAGEMENT LIMITATIONS

Managing customer-specific settings, such as adjusting transaction limits, required manual updates. These processes were slow, error-prone, and further complicated by reliance on a shared database architecture.

Illustrative Example

In this system, the database was shared for multiple operations, including Business Rules Engine (BRE) processes that handled transactions. If a user needed their transaction limit updated, the change had to be applied manually to the database. Since the same database was also processing live transactions through BRE, the manual update competed for resources, slowing down the entire process. This inefficiency caused delays, further impacting customer service and operational effectiveness.

DEPENDENCY ON PROPRIETARY TECHNOLOGY

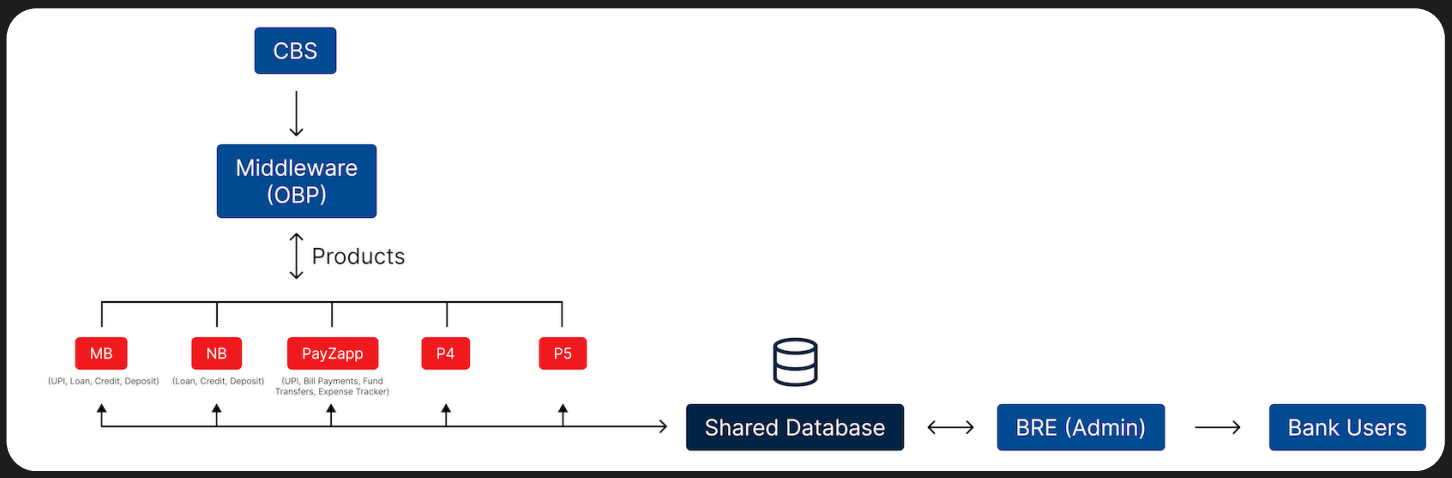
The banks reliance on proprietary technology limited its flexibility, increased costs, and delayed the adoption of newer, more efficient solutions.

Illustrative Example

The system used on-premises servers and heavily depended on Core Banking System (CBS) platforms like Finacle or Oracle Banking Platform (OBP). If the bank needed a new feature or an update, it had to wait for the proprietary platform's provider to implement the change. This process was often slow, and the system lacked the flexibility to quickly adapt to evolving business needs or accommodate customizations.

For instance, if a new compliance requirement arose, Finacle/OBP updates would take considerable time, leaving the bank struggling to keep up with regulations. Additionally, the rigidity of the proprietary system hindered the bank's ability to experiment with innovative solutions, keeping it locked into a single technology stack

PREVIOUS ARCHITECTURE



THE SOLUTION

To address these challenges, transformative solutions were implemented:

Decoupling Services Through Microservices

By transitioning to a microservices architecture, services such as credits, debits, UPI, and bill payments were separated into independent modules. This allowed upgrades to individual services without affecting others, minimizing downtime and ensuring continuous availability.

Centralized Configuration with Event-Driven Architecture

A central configuration module was introduced, where customer-specific rules could be defined and approved through a maker-checker process. Changes were propagated seamlessly to mobile and net banking channels via event-driven mechanisms, eliminating manual dependencies and reducing error margins.

Migration to Open-Source Platforms

Replacing the legacy database with an open-source alternative reduced costs, increased scalability, and eliminated dependency on a single vendor. Advanced migration tools ensured real-time data synchronization, maintaining accuracy and continuity.

Technology Stack Modernization

The outdated backend and frontend were revamped with modern frameworks, improving usability, performance, and response times. REST APIs enabled secure and efficient integration with downstream systems.

THE RESULTS

50% Faster Deployment Cycles

Modular systems enabled faster rollout of new features and updates.

Scalable Performance

The system handled higher transaction volumes with minimal latency, enhancing customer satisfaction.

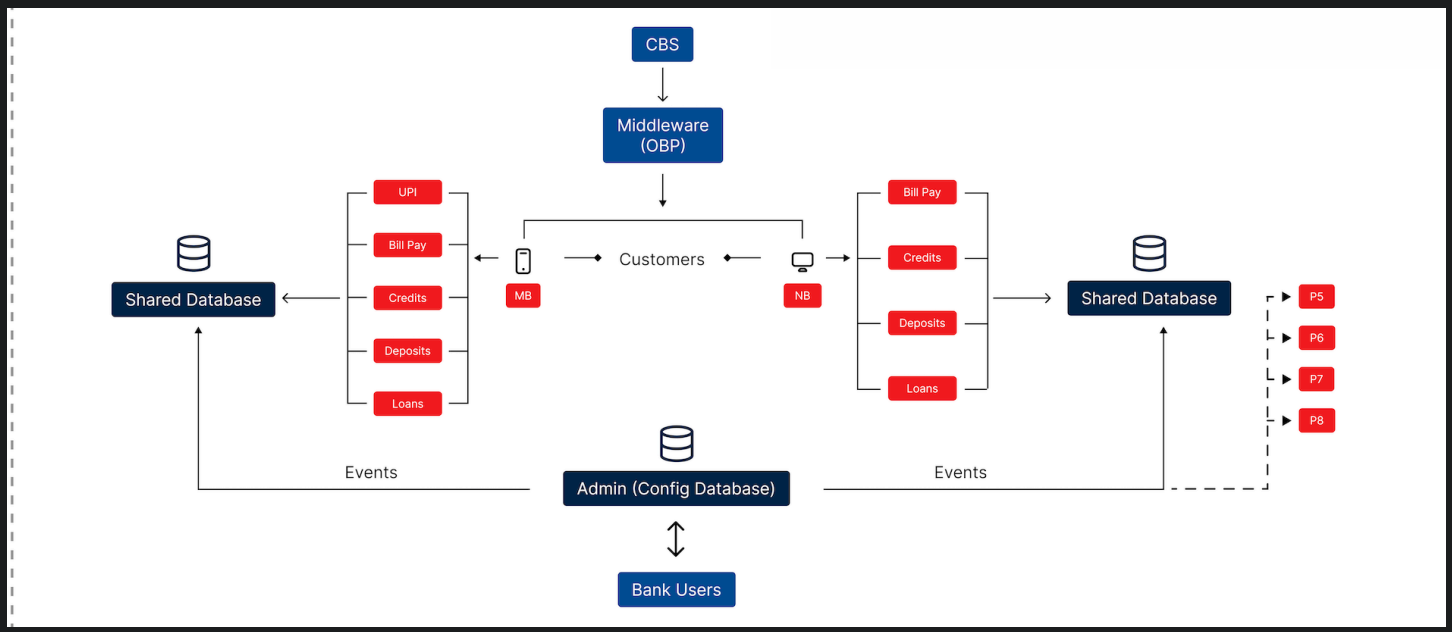
Enhanced Customer Experience

Event-based synchronization provided seamless access to updated configurations.

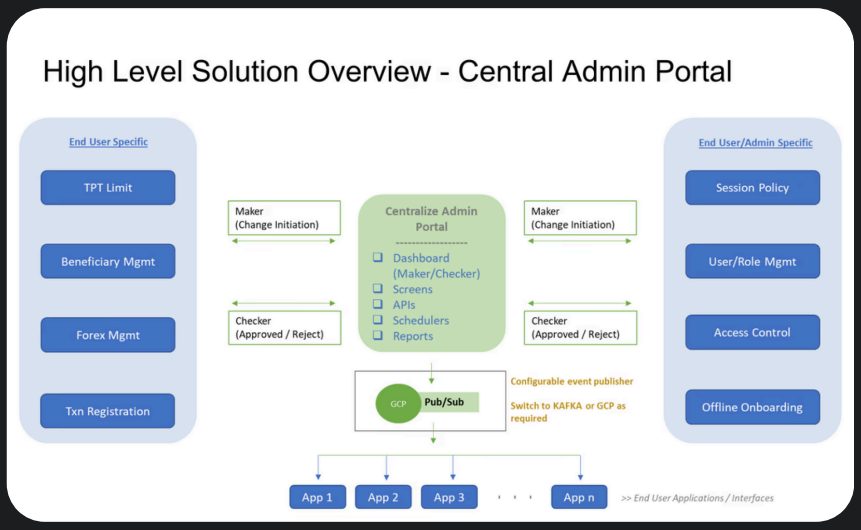
Cost Savings

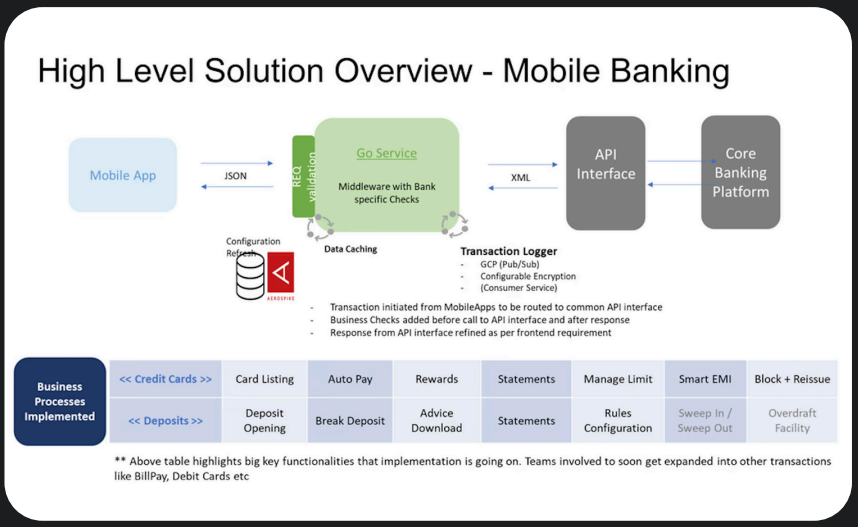
Transitioning to open-source platforms reduced operational expenses significantly.

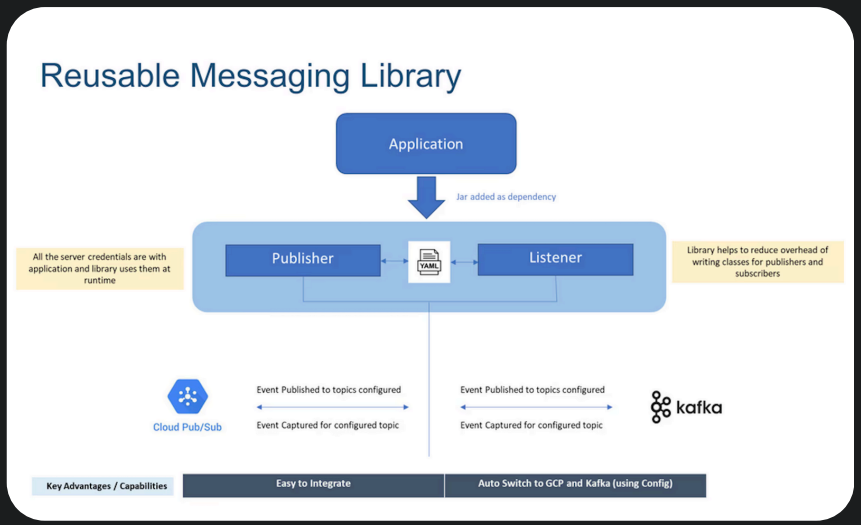
UPGRADED ARCHITECTURE



THE SNAPSHOTS







ADDITIONAL INSIGHTS

While the full deployment is yet to launch, preliminary results indicate the following:

While the full deployment is yet to Over 10 lakh rows of data were successfully migrated initially using custom-built pipelines. A pipeline was generated which facilitates the flow of records in real time.launch, preliminary results indicate the following:

Centralized configurations now ensure uniform rule enforcement across channels, improving both customer and support team efficiency.

This revamped architecture supports independent upgrades, ensuring no downtime for critical banking services.

THE SUMMARY

The journey from a monolithic to a modular, event-driven architecture highlights the importance of aligning technological advancements with operational goals. By addressing performance bottlenecks, reducing vendor lock-in, and modernizing its architecture, the institution is now poised for long-term growth and innovation.

Through this project, Josh Software contributed to enhancing flexibility, scalability, and customer-centricity, ensuring the bank remains at the forefront of digital banking innovation.

THE TECHNOLOGY

Java | Microservices | React | Google Cloud | Biocatch | Camunda