CASE STUDY

How a Large Indian Bank Reduced E-Pricing Processing from Days to Hours

Sector: Banking / Pricing Operations / FinTech Infrastructure

Days → 6–7 Hours

10M+
Records Processed

CASE BACKGROUND

A ₹100 charge on a bank statement looks straightforward.

Behind it sits a much larger operation.

The bank first has to establish whether the charge applies to that account. It may need to examine the account type, average/minimum balance, service usage, previous deductions, billing cycle, and applicable exemptions. It must calculate GST, determine whether sufficient funds are available, record partial or failed deductions, update the core banking system, and prepare the information required for customer communication.

Now run that process across millions of accounts.

For one of India's largest private sector banks, this was the scale behind its e-pricing operations, the systems responsible for calculating and applying charges associated with services such as average balance maintenance, debit cards, credit cards and SMS alerts.

The calculation itself was only one step. Customer and account data had to travel through several on-premise and cloud-based systems, pass multiple validations, return from external processing services, and reconcile correctly before the bank could apply a charge.

The existing platform had supported this process for years. As the customer base and data volumes grew, however, a daily pricing cycle could take several days to complete.
By the time one cycle finished, the next had already begun.

What E-Pricing Actually Involves

E-pricing determines the service charges applicable to individual customer accounts.

The rules differ by service and customer context.

A savings account may be assessed against average monthly or quarterly balance requirements. A debit card may attract an annual or renewal fee. SMS alerts may carry usage-based charges. Annual fees follow their respective billing cycles. Joint and linked accounts introduce additional rules around which account should be charged.

Each account record must be evaluated against the relevant criteria before a deduction is initiated.

The process then has to answer a second set of questions.

Has the customer already been charged during the current cycle? Is the account active? Is it exempt? Does it contain enough money to cover the entire amount? If the applicable charge is ₹100 but the available balance is ₹40, how should the partial amount and remaining liability be recorded?

GST must also be calculated and reconciled before the final amount reaches the core banking platform for deduction.

> *This meant the bank was not running a single calculation over a uniform dataset. It was coordinating several categories of files, rules, validations, and outcomes across a large customer base every day.*

THE PROBLEM

Why the Existing Platform Had Reached Its Limit

The earlier system had been built on ageing technologies and licensed software that had become expensive to maintain. Some components were approaching the end of their supported lifecycle, increasing the long-term risk of continuing with the same setup.

Volume created immediate pressure.

The platform was processing files of up to approximately 30 GB, representing close to 10 million records in benchmarked runs. The existing architecture struggled to handle that load within an acceptable processing window.

A cycle could extend across several days, sometimes reaching close to a week once the end-to-end movement; calculation and downstream handling were considered. That delay created more than a slow notification.

A charge that should have been calculated and applied on a particular day remained pending while subsequent records entered the next processing cycle. Accounts could appear repeatedly in the pipeline. Backlogs accumulated. Customer visibility lagged behind actual account activity.

The bank also remained dependent on a costly external platform for a process that had become central to daily operations.

Continuing to add capacity to the existing system would have preserved the same limitations. The bank needed a scalable in-house platform that could handle growing volumes, reduce processing time, and remove dependency on technology that was becoming increasingly difficult to support.

THE APPROACH

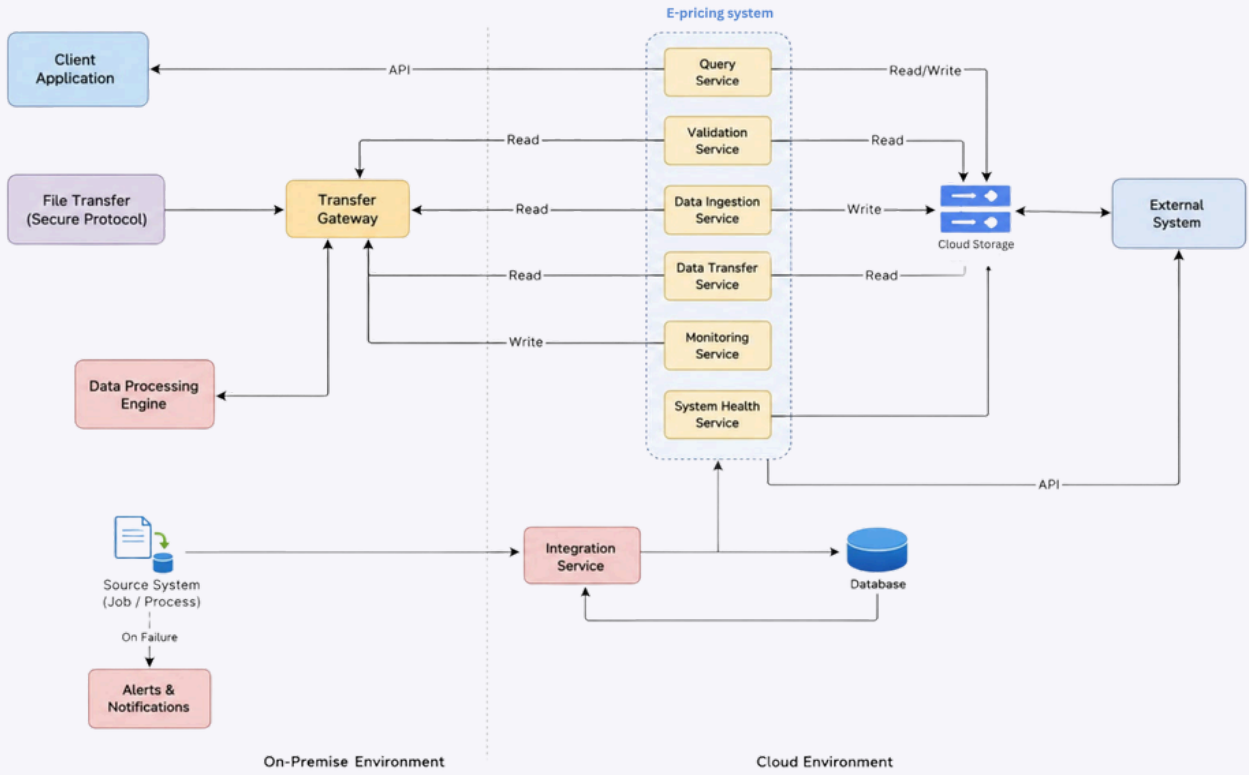
Designing Around the Bank's Existing Systems

The platform could not be rebuilt in isolation.

Customer and account data originated in the bank's core banking environment. Several downstream systems were responsible for parsing records, applying pricing rules, computing GST, executing deductions, and generating reports. Some of these services remained on-premise, while the new processing components were being introduced in the cloud.

The architecture therefore had to connect two operating environments without weakening data integrity or control.

The shared flow illustrates this hybrid design.



Files originate in the bank's on-premise environment and are deposited on an SFTP landing zone. The HWA integration service initiates and controls the workflow. From there, a set of microservices manage validation, inbound transfer, outbound transfer, and monitoring across the cloud landing zone.

The flow also connects with the bank's service-oriented architecture, pricing engine, GST-processing systems, and core banking environment.

The work was less about moving one application to the cloud and more about creating a dependable pipeline between systems that had to remain where they were.

Turning a Multi-System Flow into a Controlled Pipeline

The first requirement was orchestration.

A pricing run could involve multiple files, each serving a different purpose. Before processing began, the platform had to confirm that the expected files had arrived, that none were empty, and that their basic structure met the required conditions.

Spring Batch was used to manage high-volume file processing. **The workflow then moved through a defined sequence:**

- Files were received from the core banking environment through the on-premise SFTP server.
- Initial validations checked file count, size, and availability.
- Validated files were transferred to the cloud landing zone.
- The platform confirmed that the complete file set had arrived without alteration.
- Relevant services parsed the records and applied account-level pricing rules.
- Files were routed for GST calculation.
- Processed outputs returned through the cloud layer.
- Final files were sent back to the core banking system for deductions and reporting.

Each stage had to be completed successfully before the next could begin.

Status flags were maintained across services so that the workflow could identify where a failure occurred and prevent an incomplete file from moving further through the pipeline. Monitoring services tracked file movement and processing status throughout the run.

This converted a series of loosely connected transfers into a controlled end-to-end process.

Protecting Data Across On-Premise and Cloud Systems

Moving files of this size between environments introduced a second challenge: proving that nothing had changed in transit.

A successful upload was not enough.

The system had to confirm that the file received in the cloud was identical to the file placed on the on-premise server. The same assurance was required when processed files returned from downstream systems.

Checksum validations were therefore introduced at critical stages of the flow.

If a file left the on-premise environment with a defined checksum, the value had to match after it reached the cloud. Record counts, file summaries, and expected outputs were also reconciled before the process could continue.

This mattered because a seemingly minor loss or alteration could affect charges across thousands of accounts.

> *The architecture also avoided retaining unnecessary customer-level information within the new application. Operational logs focused on file movement, processing status and execution outcomes, while the core banking platform remained the system of record for transactions.*

Building for Portability and Internal Control

The bank wanted to reduce dependency on the earlier vendor platform without creating another rigid technology dependency.

Reusable libraries were developed for recurring capabilities such as accessing secrets, managing cloud configurations, and supporting communication between services. These abstractions reduced duplication across the application and made the solution easier to maintain.

The cloud integration layer was also designed with portability in mind. The underlying implementation could work across AWS, Google Cloud or Azure through configuration rather than extensive code changes.

This gave the bank flexibility in how it hosted and evolved the platform over time.

More importantly, the solution was built as an in-house capability. The bank retained control over its pricing workflow, release roadmap and future enhancements instead of routing every change through an external product vendor.

THE OUTCOME

What Changed in Practice

The redesigned pipeline reduced the end-to-end processing window from several days to approximately six or seven hours in benchmarked runs involving close to 10 million records and files of up to 30 GB.

That change altered the operational rhythm of the process.

Pricing cycles could be completed within the same day rather than carrying into subsequent cycles. Backlogs were reduced. Records no longer remained pending for days before charges were applied. Customer communications could reflect account activity much closer to the time it occurred.

The bank also gained a clearer view of failures across the pipeline. Instead of waiting for an entire batch to fail, teams could identify the stage at which a file had stopped, validate its status, and restart the relevant part of the workflow.

> *The new architecture removed dependency on ageing software, improved scalability, and created a foundation the bank could extend as customer volumes and pricing rules evolved.*

KEY TAKEAWAY

The Larger Lesson

A bank charge may be small. The infrastructure required to calculate it correctly across millions of accounts is not.

This project brought together core banking data, pricing rules, GST processing, deductions, file reconciliation, and customer communication across a hybrid environment. Every component had to move in sequence, and every transfer had to preserve the integrity of the underlying data.

We helped the bank replace a slow, vendor-dependent batch process with an in-house processing pipeline that connected its on-premise and cloud systems, validated files at each stage and compressed a multi-day cycle into hours.

The immediate outcome was faster e-pricing. **The more important outcome was operational control over a high-volume banking process that could now grow without carrying out the limitations of the system it replaced.**

That is what this project really solved.

A centralised, governed, searchable system – built to assist rather than obstruct.

www.joshsoftware.com | contact@joshsoftware.com | +91 7887889902