

RAPTORS CONFERENCE 2026

83% Faster, 10x Fewer Errors

Engineering a Self-Healing Payment Saga Tracker

Tarun Kumar Chatterjee • Payments Integration

Distributed Payments Are Fragile by Default

A single failed step in one service can corrupt the entire transaction — with no automatic recovery.



Inconsistent State

A failed mid-flow step leaves payments stuck between "charged" and "confirmed."



Manual Reconciliation

Engineers chase logs across services to reconstruct what actually happened.



No Self-Healing

Every error requires human intervention rather than automatic compensation.

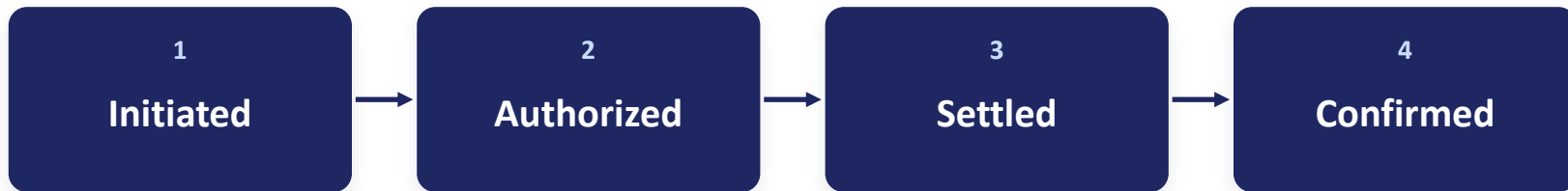


Slow Recovery

Mean-time-to-recovery measured in hours, not seconds — burning engineering time and trust.

The Solution: Event-Driven Saga Pattern

Break the transaction into independent steps. Each step publishes an event. Failure triggers automatic compensation.



Automatic Compensation on Failure

If any stage fails the tracker rolls back completed steps, restoring system integrity — no manual cleanup.

Under the Hood

The architectural building blocks that make self-healing possible.



Pub/Sub Event Backbone

Services publish state-change events (Initiated, Settled, Failed) and react independently.



Compensation Logic

Each step defines its own rollback action, triggered automatically on downstream failure.



Real-Time State Tracking

The saga tracker maintains a live view of every transaction's journey end to end.

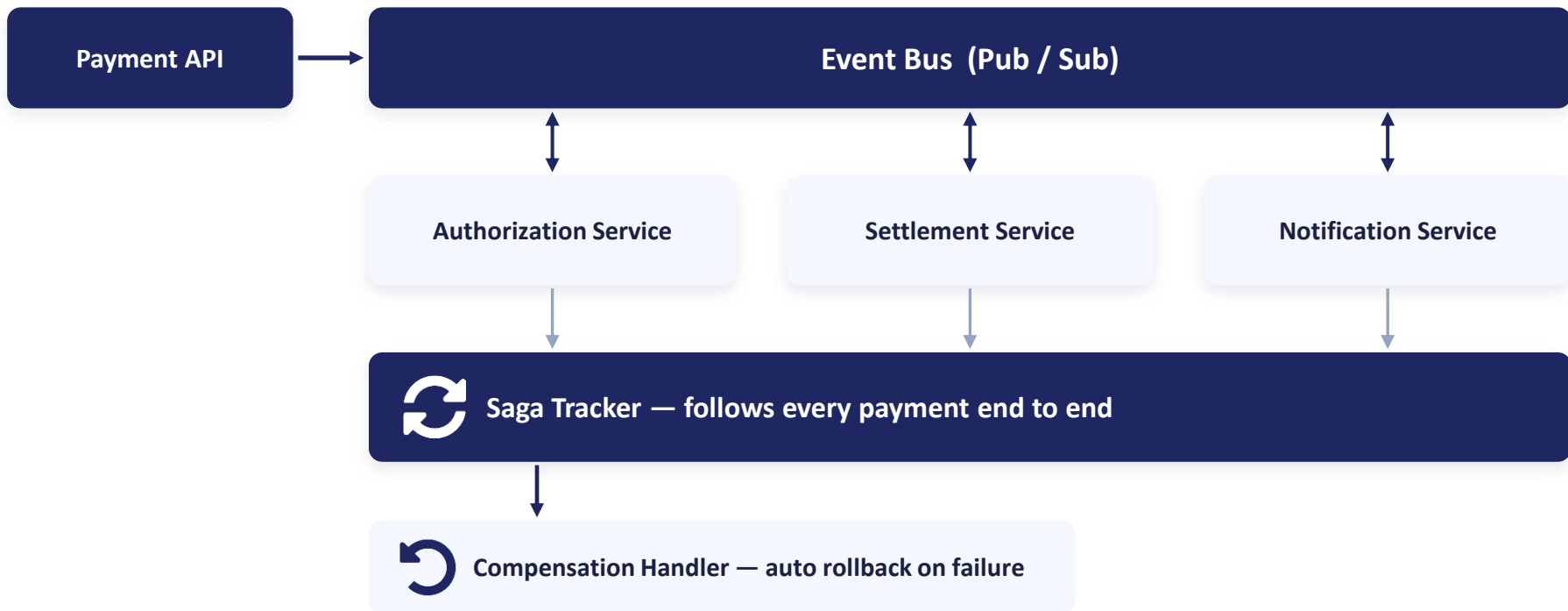


Secure, Scalable Storage

Transaction and event data secured with industry best practices for compliance and scale.

Architecture at a Glance

How events flow from the API down to automatic compensation.



MongoDB: The Saga Event Store

Every state transition is persisted as an immutable event document — giving full auditability and instant replay.

saga_events collection

```
{
  sagaId:    "txn_8f3a92c",
  step:      "SETTLEMENT",
  status:    "COMPLETED",
  timestamp: ISODate(...),
  payload:   { amount: 149.99,
               currency: "USD" },
  retryCount: 0,
  compensated: false
}
```



Immutable Log

Events are append-only — every state change is preserved for full audit history.



Fast Queries

Indexed on sagaId + step for sub-millisecond saga state lookups.



Event Replay

Replay any saga from its events to reconstruct state or debug failures.

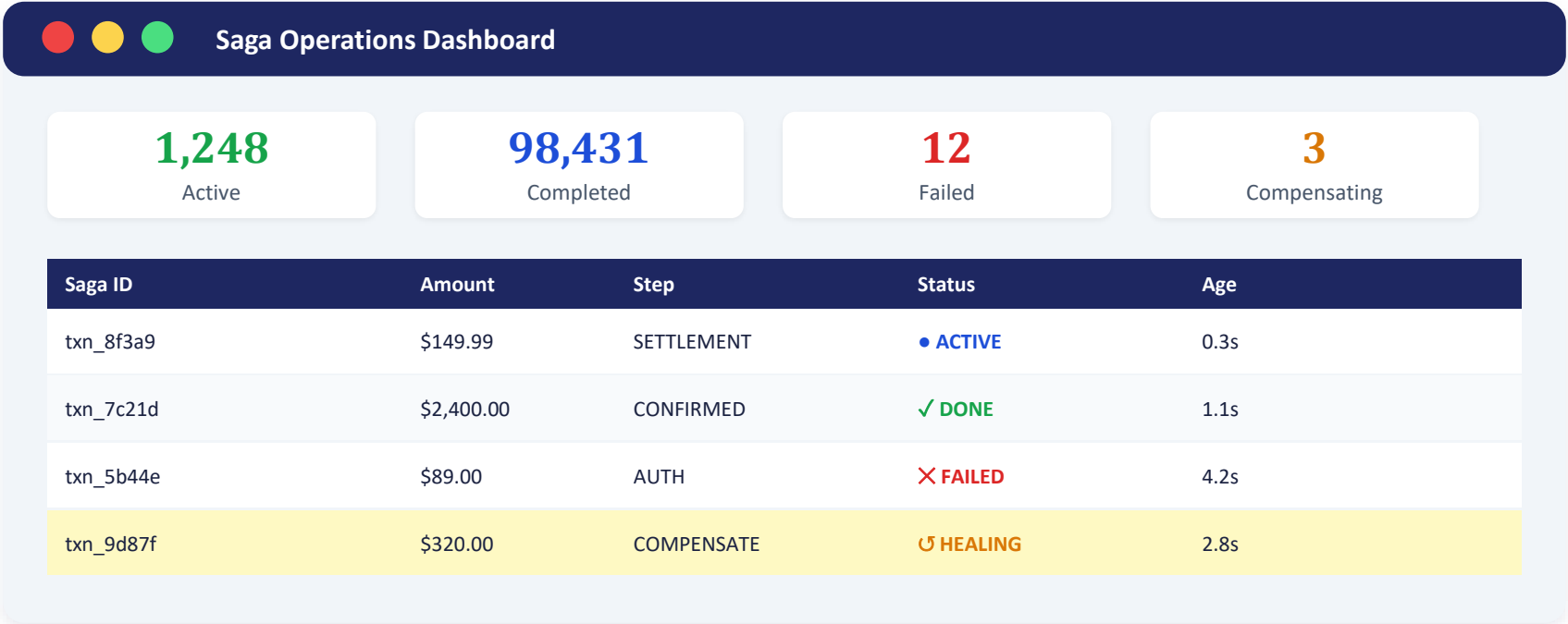


TTL & Archival

TTL indexes auto-expire hot data; cold events archive to Blob Storage.

Real-Time Saga Dashboard

A live operator view showing every saga's status — making automated recovery visible and trustworthy.



Results That Matter

Platform-level impact after rollout.



83%

Faster Transaction Processing



10x

Fewer Payment Errors



80%

More Processing Capacity

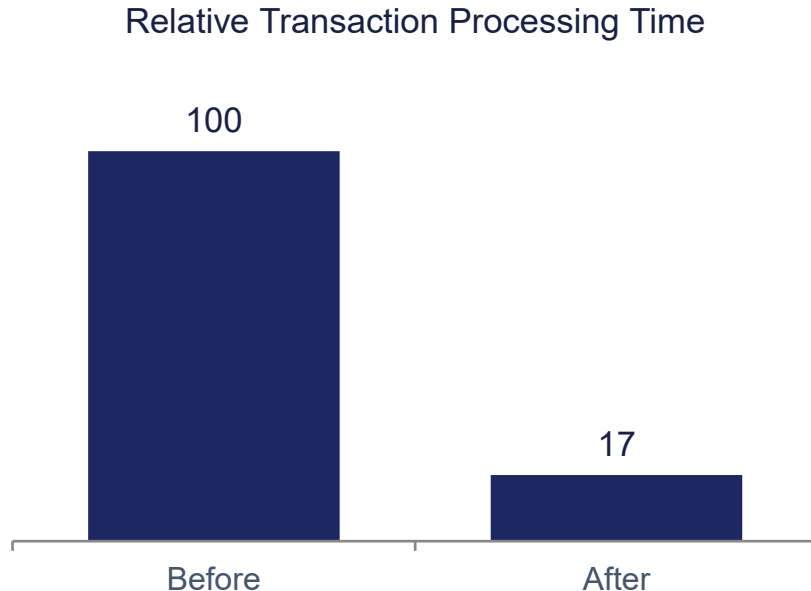


95%

Drop in Security Incidents

Business Impact

Reliability gains translated directly into growth and savings.



 **\$2M**

Additional annual revenue

 **30%**

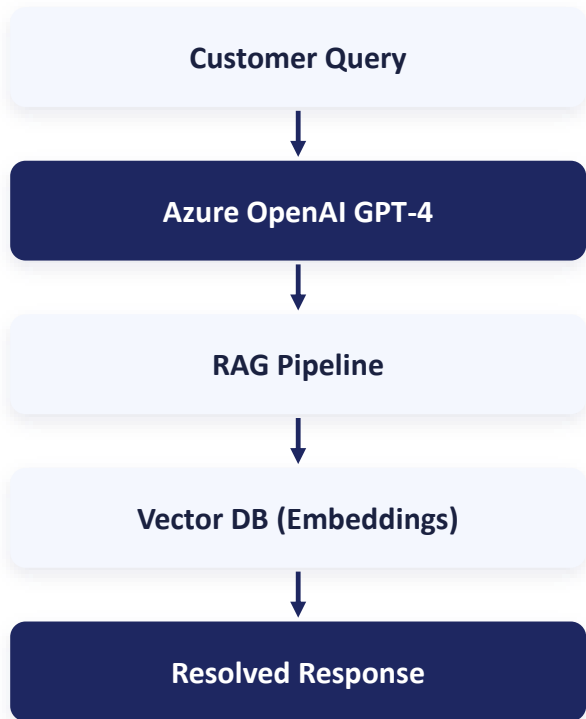
Increase in user base

 **40%**

Cut in payment processing time

AI Integration: Azure OpenAI & RAG Chatbot

A GPT-4 powered chatbot handles 50–70% of customer queries automatically — reducing support costs by 10–15%.



GPT-4 Deployment

Azure AI Foundry hosts the model — full playground for testing and prompt iteration.



RAG Pipeline

Retrieves relevant company docs at query time before generating an answer.



Vector Embeddings

Documents transformed into semantic vectors, stored for sub-100ms retrieval.



Streaming Mode

Responses stream token by token — users see answers forming like ChatGPT.

Lessons Learned

What I'd tell any team building a saga tracker from scratch.



Design Compensations First

Decide how every step rolls back before writing a single line of happy-path code.



Idempotency Is Non-Negotiable

Events get retried. Every handler must be safe to run more than once.



Observability Builds Trust

A real-time saga dashboard is what convinces ops teams to trust automated recovery.



Start With One Critical Flow

Prove the pattern on your highest-risk transaction path first, then expand.

KEY TAKEAWAY

Event-driven saga patterns turn fragile, error-prone distributed transactions into resilient, self-healing systems.

Pair that with MongoDB for durable event storage, a real-time dashboard for trust, and Azure OpenAI for intelligent support — and the savings compound across the entire business.

Thank You

Questions & Discussion

Tarun Kumar Chatterjee • reachtarunchatterjee@gmail.com

LinkedIn: <https://www.linkedin.com/in/tarun-kumar-chatterjee-605963176/>