Yinka Omole

Building Scalable Systems in a Complex Compliance World

LDX3 2025

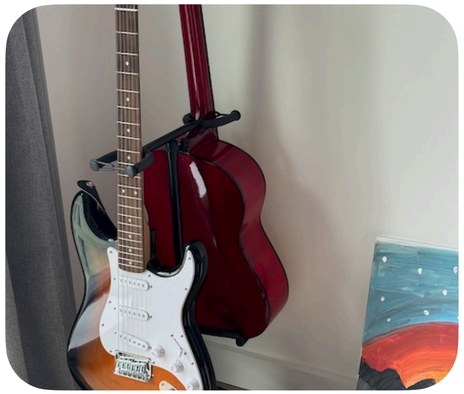
Outline

About Me

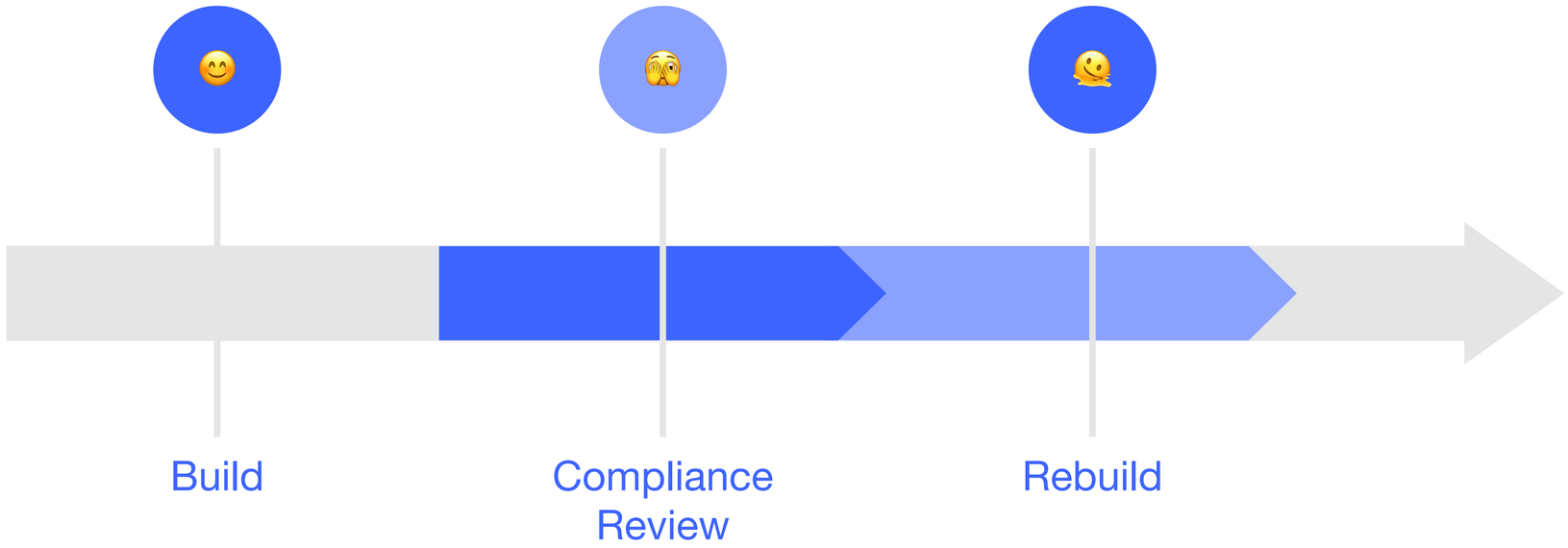
Goldman Sachs, Personio

Software engineer who enjoys building products that help people get things done.

Spent 6+ years dealing with regulatory complexity across banking and HR tech.



The Compliance Challenge

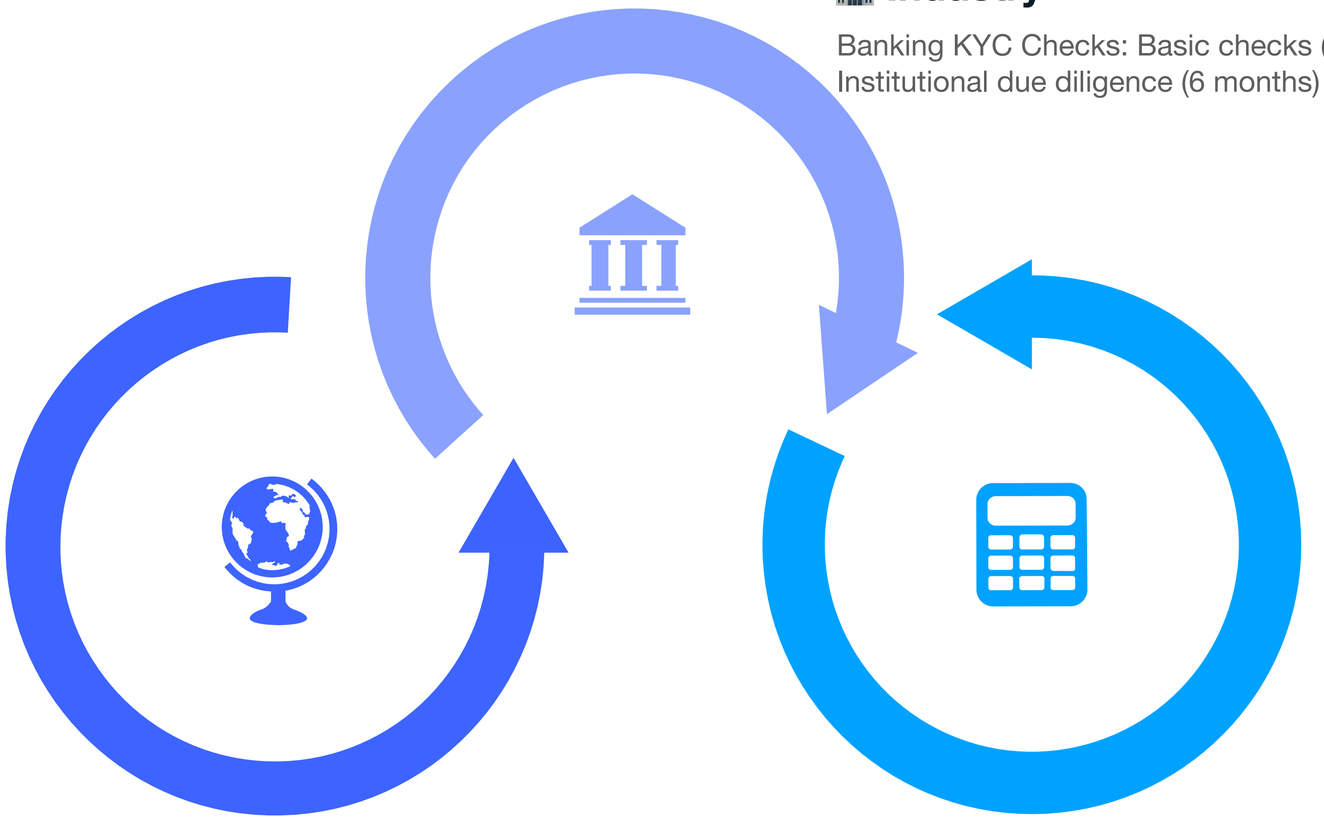


Yinka Omole

Compliance Complexity

🏛️ Industry

Banking KYC Checks: Basic checks (5 minutes) vs Institutional due diligence (6 months)



🌍 Jurisdiction

GDPR “you can delete everything" vs Financial "keep everything"

📈 Business Context

Startup vs Public Company

But here's what I've learned:
compliance constraints shouldn’t be obstacles. They can guide us to better architecture.

At Personio, implementing one of the patterns I'll share today, helped reduce our international expansion timeline.

Let me show you how...

Two Patterns That Work

State Machines

Workflow-Based Compliance

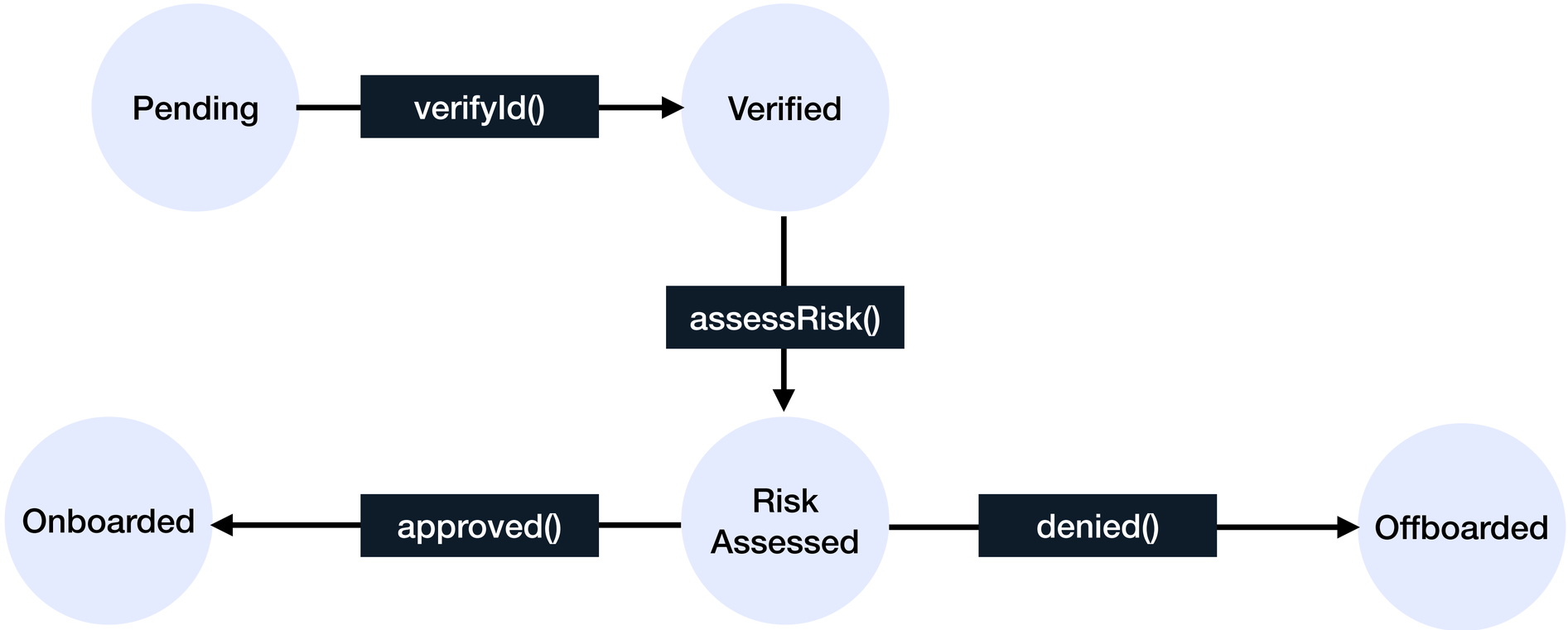
Compliance Libraries

Shared Compliance Logic

Yinka Omole

State Machines

Workflow-Based Compliance



Yinka Omole

State Machines

Workflow-Based Compliance

✅ Advantages

- Clear visibility
- Audit trails built in
- Easily extensible

⚠️ Watch Out For

- Mixing business rules with compliance gates

Yinka Omole

Compliance Libraries

Shared Compliance Logic

Before

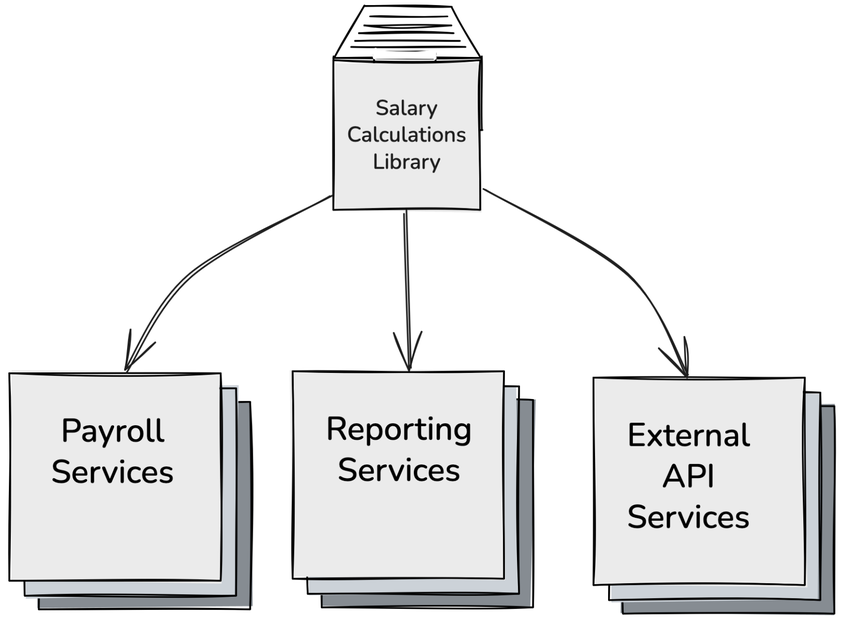
```
// Payroll Service
fun calculateSalary(employee: Employee): Money {
    // 200 lines of compliance logic
}

// Reporting Service
fun calculateSalary(employee: Employee): Money {
    // Same 200 lines, slightly different
}

// External API Service
fun calculateSalary(employee: Employee): Money {
    // Same 200 lines, with bugs
}
```

VS

After



Yinka Omole

Compliance Libraries

Shared Compliance Logic

✅ Advantages

- New services immediately compliance-ready
- Consistency / SSOT
- Easy to implement

⚠️ Watch Out For

- Less flexibility for service-specific logic
- Requires coordination for updates
- Needs language homogeneity

When to Use Which Pattern

🔄 **Multi-step workflows with gates? → State Machines**

Example: Onboarding, approvals, certifications

📚 **Same rules across services? → Compliance Libraries**

Example: Calculations, validations, formatting, access checks

⚡ **Rules change frequently? → Rules Engine ***

Example: Fraud detection, risk assessment

📋 **Need complete audit history? → Event Sourcing ***

Example: Financial transactions, medical records

*Beyond today's scope

You might need combinations of these, or even completely different patterns.

Start Simple, Measure, and Evolve.

Making It Work

🤝 **People**: Embed compliance experts

Compliance as partners, not reviewers.

📋 **Process**: Front-load requirements

Agile doesn’t work.

🧪 **Testing**: Build rigorous testing infrastructure

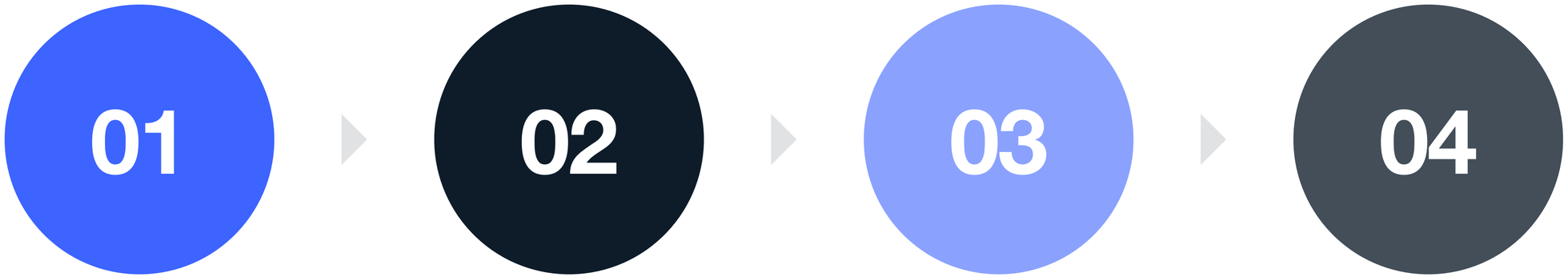
Regression testing is crucial.

Key Takeaways

Compliance isn't something you validate after the fact. It's something that shapes your architecture from day one.

You can turn compliance into a competitive advantage.

Your Next Steps This Week



Audit one critical flow

Find where compliance logic is duplicated

Pick one pattern and prototype it

Schedule coffee with your compliance team

Thank you!

🌐 yinka.dev
💼 LinkedIn: Yinka Omole

**If you're facing technical compliance challenges,
I'd love to chat about them.**