wipro intelligence™

proof over promise.

From Intent to Production

MCP Gateway Patterns for Regulated Banking

A practitioner's retrospective on building a four-layer agentic banking system — the patterns that worked, the failures along the way, and the MCP protocol gaps worth knowing about.

Hariskumar Panakkal | DMTS, Senior Principal Member & Enterprise Architect

Jun 2026

wipro intelligence™
proof over promise.

THE PROBLEM

MCP is easy in demos.
It gets hard in production.

POC MCP Server

- Run it locally
- Maybe OAuth, maybe not
- Stateful? Stateless? Works either way
- Logs in the terminal
- Tools and data live in the same process

Banking MCP System

- Deployed behind a gateway, in Kubernetes
- OAuth 2.0 or strict mTLS, non-negotiable
- Stateful tool calls → pod affinity matters
- Immutable audit, cryptographic receipts
- SSN, DOB, government IDs must never leak

wipro intelligence™
proof over promise.

WHAT I'LL SHARE

Five patterns. Three failures. Three MCP gaps.

Everything in this talk comes from code that shipped.

01	**The four-layer architecture**	*UI → Agent → Gateway → MCP Server*
02	**Why my first three deployments failed**	*Session affinity and the mcp-session-id header*
03	**Keeping SSN out of the LLM**	*A tool wrapper pattern for PII safety*
04	**Consent as a first-class MCP tool**	*With a signed receipt, not bundled into submit*
05	**Three gaps I hit in MCP**	*Honest notes for the community*

wipro intelligence™
proof over promise.

PATTERN 01 · THE FOUR-LAYER ARCHITECTURE

What I built, and why it's split this way

Who connects:

Partner AI agents

Partner ops teams

Amazon · Walmart · BestBuy
React portal

React Portal
Partner UI · :5173
Zod validation, PII masking

LangGraph Agent
FastAPI · :8080
14 REST endpoints, 11 workflows

MCP Gateway
Microsoft · :8000
OAuth, session affinity, K8s routing

FastMCP Server
Python · :8001
19 tools · AES-256-GCM vault

Why split this way:

- Each layer one job
- Security boundary
- Scale independently
- Swap components

wipro intelligence™
proof over promise.

FAILURE 01 · SESSION AFFINITY

My first three gateway deployments failed

MCP is a stateful protocol. I treated it like a REST API. The pods broke.

The Bug

Tool call 1 → Pod A (starts session)
Tool call 2 → Pod B
Pod B: "Who are you? No session."

Every multi-call workflow broke randomly — **depending on which pod Kubernetes routed to.**

The Fix

mcp-session-id
— an HTTP header that tells the gateway which pod owns this session.

On MCP initialize:
1. Gateway assigns session ID
2. Client sends it with every request
3. Gateway routes to the same pod

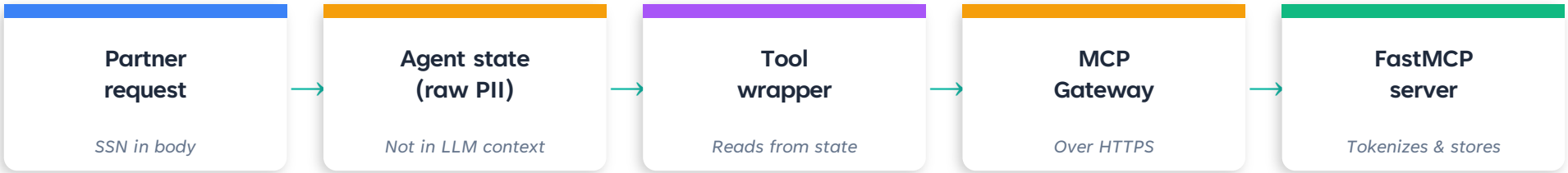
→ ***Stateful tool calls just work.***

wipro intelligence™
proof over promise.

PATTERN 02 · PII SAFETY

How to keep raw SSN out of the LLM

The LLM still reasons about the workflow. It just never sees the value.



What the LLM sees (and doesn't)

✓ DOES see:
- field names (ssn, dob)
- masked references (last-4 only)
- correlation_id, partner_id
- validation results, consent IDs

✗ NEVER sees:
- raw SSN values
- raw DOB, government ID numbers
- OAuth tokens or secrets
- anything in tool system prompts

wipro intelligence™
proof over promise.

PATTERN 03 · TRACEABILITY

One correlation_id. Every layer. Every log.

When something breaks in production, I need to trace it in seconds — not hours.

React UI	Generated on form submit	d0a3c16c-51c6-40f1-bac3-d92ef53d4206
Agent	Bound to structlog context	d0a3c16c-51c6-40f1-bac3-d92ef53d4206
Gateway	Injected as HTTP header	d0a3c16c-51c6-40f1-bac3-d92ef53d4206
MCP Server	Part of every audit event	d0a3c16c-51c6-40f1-bac3-d92ef53d4206
Issuer queue	Signed into payload	d0a3c16c-51c6-40f1-bac3-d92ef53d4206

One grep. One UUID. Complete traceability across four layers and an issuer.

wipro intelligence™
proof over promise.

PATTERN 04 · CONSENT DESIGN

Consent is its own MCP tool. Not a field on submit.

✗ My first attempt

```
submit_application(...,
  consent_given=true
)
```

Problems:
- No proof the user actually agreed
- One-line revoke is impossible
- Regulators can't audit the moment
- Boolean flag flattens nuance (purpose, version, expiry)

✓ What I landed on

```
capture_consent(...)
  → ConsentReceipt
    {id, hash, version}
submit_application(...,
  consent_receipt_id)
```

Wins:
- Tamper-evident via SHA-256
- State machine enforces the gate
- revoke_consent works independently

PART 05 · PROTOCOL GAPS

Three things I wish MCP handled natively

Honest notes for the community — not criticism.

01

Stateful session routing

Pain: MCP has session IDs but no standard for how gateways should use them for pod affinity.
Workaround: Microsoft MCP Gateway handles it via mcp-session-id. Every gateway implements it differently.

02

Consent as a protocol primitive

Pain: Every MCP builder in a regulated domain rebuilds consent tools from scratch.
Workaround: I built record_consent + verify_consent_receipt. It could be a spec-level primitive.

03

Audit event emission

Pain: No standard way for tools to emit auditable events. Each server rolls its own format.
Workaround: Added emit_audit_event() internally. A protocol-level hook would let observability tools plug in.

wipro intelligence™
proof over promise.

TAKEAWAYS

What I'd keep doing

Treat PII like a boundary, not a best-effort

Tool wrappers that physically prevent PII from entering the LLM context — not system prompts that ask nicely.

Pydantic schemas are non-optional

Every tool input and output is a typed, validated model. Catches bad data at the edge, documents the contract for free.

Make consent its own tool, not a field

Verifiable receipts, versioned, revocable. You'll thank yourself at audit time.

Thread a correlation_id through every layer

UI, agent, gateway, MCP server, issuer queue. One UUID turns hour-long incident forensics into minutes.

wipro intelligence™
proof over promise.

FOR THE COMMUNITY

If you're building with MCP and a gateway, heading toward production —

Four layers. One correlation ID. **Zero PII leaks.**

Consent as a first-class tool. Session affinity done right.

These are the patterns — and scars — I wish I'd had when I started.

You can apply them to banking, healthcare, insurance,
or any domain where MCP needs to grow up.

wipro intelligence™

proof over promise.

Thank you.

Questions, critiques, better patterns — all welcome.

Hariskumar Panakkal

Enterprise Architect · DMTS

sessionize.com/hariskumarpanakkal · LinkedIn: /in/hariskumarpanakkal

Click to add title

fdfd

wipro intelligence™

proof over promise.

Agentic Banking
AI-DLC Meets MCP Gateway

From Intent to Production: Building a Governed, Agent-Driven Credit Card Platform

Hariskumar Panakkal | DMTS, Senior Principal Member & Enterprise Architect

Apr 2026

wipro intelligence™
proof over promise.

The Challenge

Building agent-driven banking at production scale requires solving four problems at once.

Partner AI Agents

Amazon, Walmart, BestBuy AI systems need a standardized way to submit credit applications — not custom REST integrations per partner.

Privacy & Compliance

SSN, government IDs, and financial data demand tokenization, consent capture, and immutable audit trails from day one.

Governed Velocity

Traditional SDLC is too slow; autonomous AI coding is too risky. Need a methodology that delivers fast AND safe.

End-to-End Platform

Not just tools — need UI for ops, agent for orchestration, gateway for security, and servers for banking logic.

wipro intelligence™
proof over promise.

Project Muon

A 4-component platform for agent-driven credit card onboarding — built using AI-DLC

React UI

project-muon-ui

- Partner portal (port 5173)
- 8 screens: Dashboard, Apply, Audit
- React 18 + Vite + shadcn/ui
- TanStack Query + Zod

FastAPI + LangGraph

project-muon-agent

- Orchestration layer (port 8080)
- 14 REST endpoints, 11 workflows
- 1 submit + 10 query workflows
- PII-safe: LLM never sees raw SPI

Microsoft MCP Gateway

mcp-gateway

- Secure front door (port 8000)
- .NET 8, OAuth 2.0, session affinity
- K8s adapter orchestration
- SSE/stdio protocol translation

FastMCP Server

project-muon-mcp

- Banking logic (port 8001)
- 19 MCP tools, 5 domains
- Pydantic schemas, SPI tokenization
- In-memory stores, audit hashing

wipro intelligence™
proof over promise.

AI-DLC: The Methodology

AI operates as a collaborator across the entire lifecycle — humans shape direction, validate outcomes, own every critical decision.

INTENT

State the goal. AI asks clarifying questions to shape scope and surface constraints.

Human-gated

INCEPTION

AI decomposes intent into units of work. Generates user stories, HLDs. Humans approve.

Human-gated

CONSTRUCTION

AI generates LLDs, code, and tests per unit. Humans review at every checkpoint.

Human-gated

OPERATIONS

AI maintains observability, audit, and runbooks. Humans validate production readiness.

Human-gated

wipro intelligence™
proof over promise.

AI-DLC in Claude Code — Slash Commands

Each phase becomes a repeatable slash command. Same flow applied to all 4 Project Muon repos.

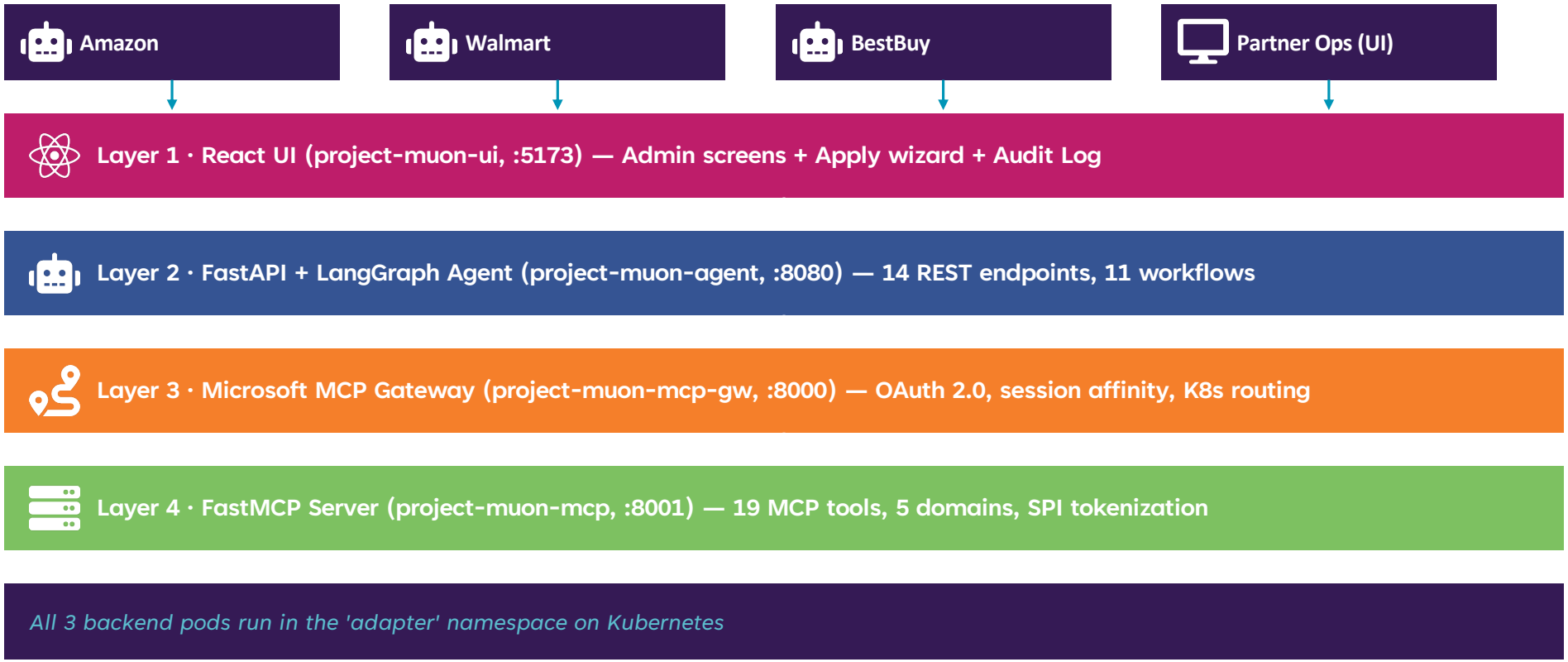
Command	AI-DLC Phase	Role	Output
/project:inception	Inception	Product Manager	overview_user_stories.md + assumptions.md
/project:uow-plan	Construction Planning	Software Architect	units/unit-{1..N}.md
/project:hld [unit]	High-Level Design	Solution Architect	construction/{unit}/hld.md
/project:lld [unit]	Low-Level Design	Technical Architect	construction/{unit}/lld.md
/project:implement [unit]	Implementation	Software Engineer	src/ code + tests/

Each command writes a plan with checkboxes, pauses for human approval, then executes step-by-step.

wipro intelligence™
proof over promise.

End-to-End Architecture

How a credit card application flows through the 4-layer Project Muon stack



LAYER 4 project-muon-mcp MCP Server

wipro intelligence™
proof over promise.

19 MCP Tools Across 5 Banking Domains

Admin	Applications	Consent	Webhooks	Audit
create_tenant	submit_application	capture_consent	register_webhook	get_audit_record
get_tenant	get_application_status	get_consent_status	get_webhook_delivery_history	query_audit_events
list_tenants	get_application_lifecycle	revoke_consent	trigger_decision	
suspend_tenant	list_applications			
reinstate_tenant	cancel_application			
	get_validation_report			

Python 3.12 · FastMCP 3.2 · AES-256-GCM for SPI · Pydantic validation · Immutable audit with SHA-256 hash

LAYER 3 mcp-gateway MCP Gateway

wipro intelligence™
proof over promise.

Secure Front Door for Agent Traffic

Gateway Capabilities

OAuth 2.0 + Azure AD
Bearer token validation or dev mode headers

Session Affinity
Stateful MCP routing via mcp-session-id header

Adapter Routing
/adapters/bank-mcp-services/mcp → pod

Protocol Translation
HTTP ↔ stdio, streaming tool responses

Rate Limiting + RBAC
Per-partner quotas, scoped access

Tech Stack

.NET 8 · ASP.NET Core
ModelContextProtocol.AspNetCore
Redis / Cosmos DB (session store)
3 projects: Service, Management, Tools

Kubernetes Deployment

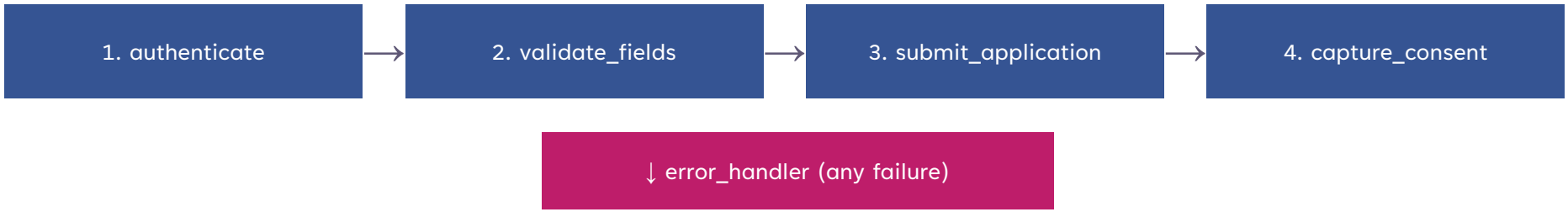
Namespace: adapter
Runtime: minikube + Podman
Adapter creates: StatefulSet + Service
Dev mode: X-Dev-* headers, no token

LAYER 2 project-muon-agent LangGraph

Intelligent Orchestration with PII Safety

wipro intelligence™
proof over promise.

Submit Application Workflow (LangGraph)



11 Total Workflows · 14 REST Endpoints

- 1 Submit workflow (5 nodes)
- 10 Query workflows (2 nodes each)
- LLM configurable: Claude or Azure OpenAI
- langchain-mcp-adapters for gateway calls
- correlation_id + idempotency enforced

PII Safety Boundary

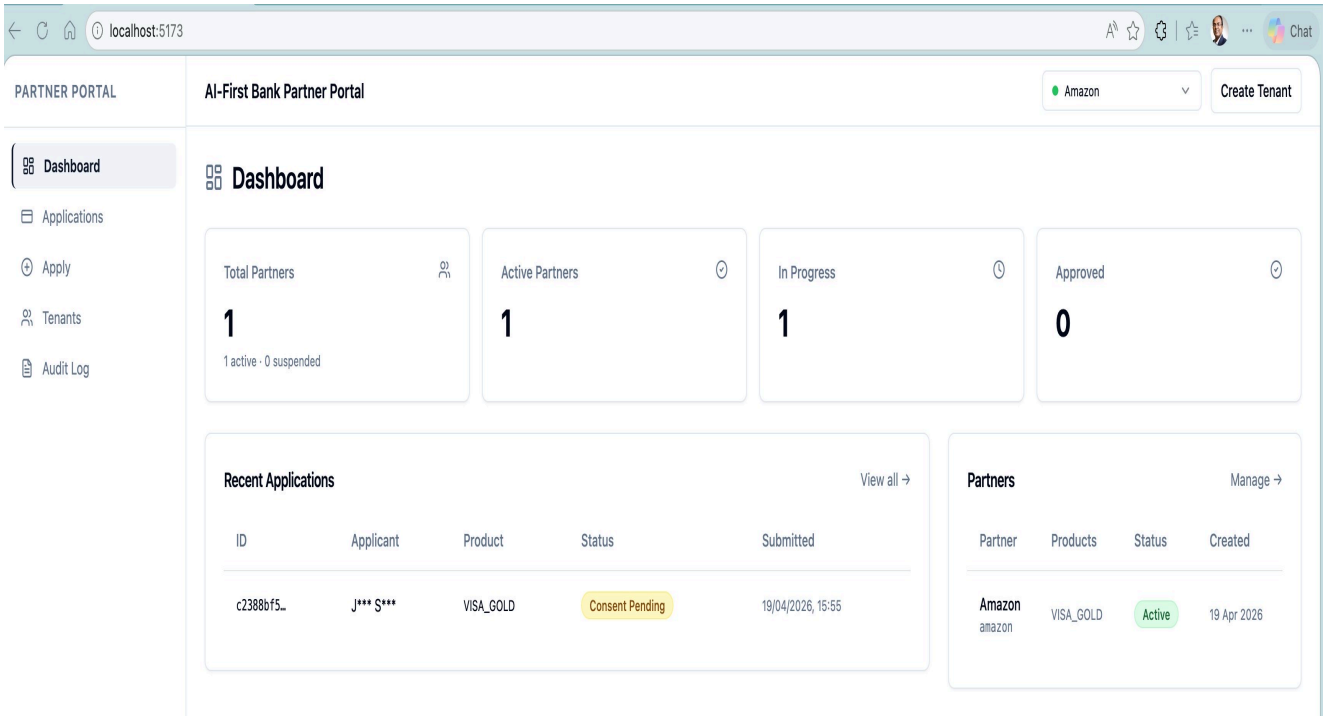
- Raw SSN / DOB never enter LLM context
- Tool wrappers pass PII directly to gateway
- Field names logged, values never logged
- AgentState masks values (last 4 digits only)
- Trimmed response strips token + raw data

LAYER 1 project-muon-ui Partner Portal

wipro intelligence™
proof over promise.

Partner Portal — Dashboard

8 screens: Dashboard, Tenants, Create Tenant, Tenant Detail, Apply (5-step wizard), Applications, Audit Log, Audit Detail



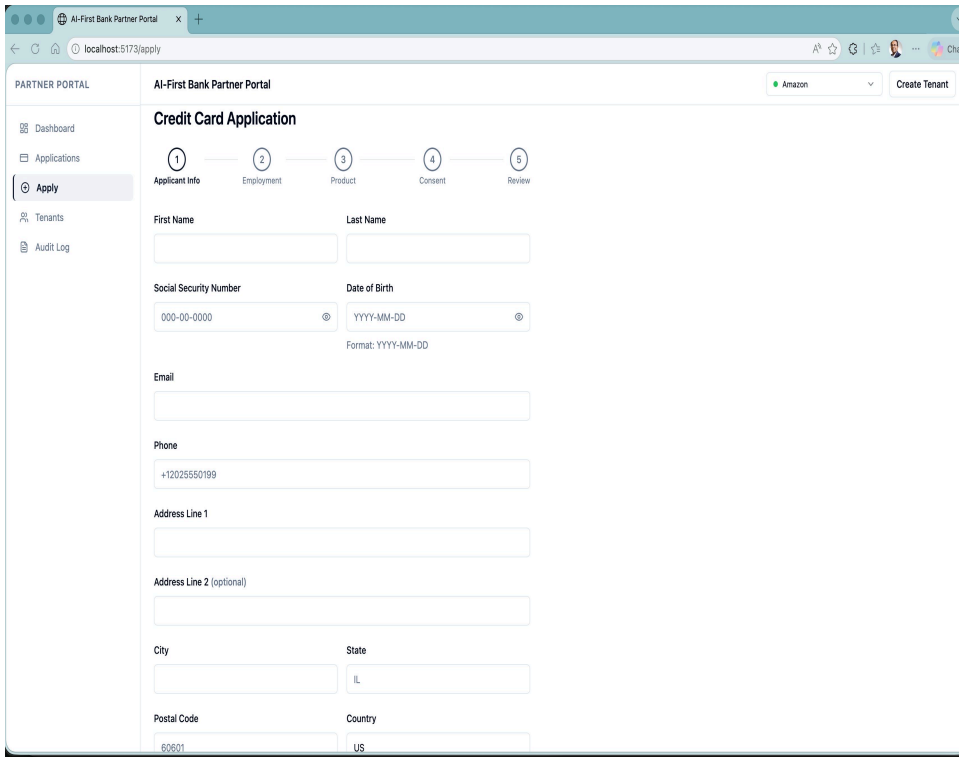
Tech Stack

- React 18 + Vite
- TypeScript (strict)
- shadcn/ui + Tailwind
- React Router v6
- TanStack Query v5
- React Hook Form + Zod
- Axios (typed)
- No localStorage for PII
- SSN masked as •••-••-XXXX
- Client secret shown once

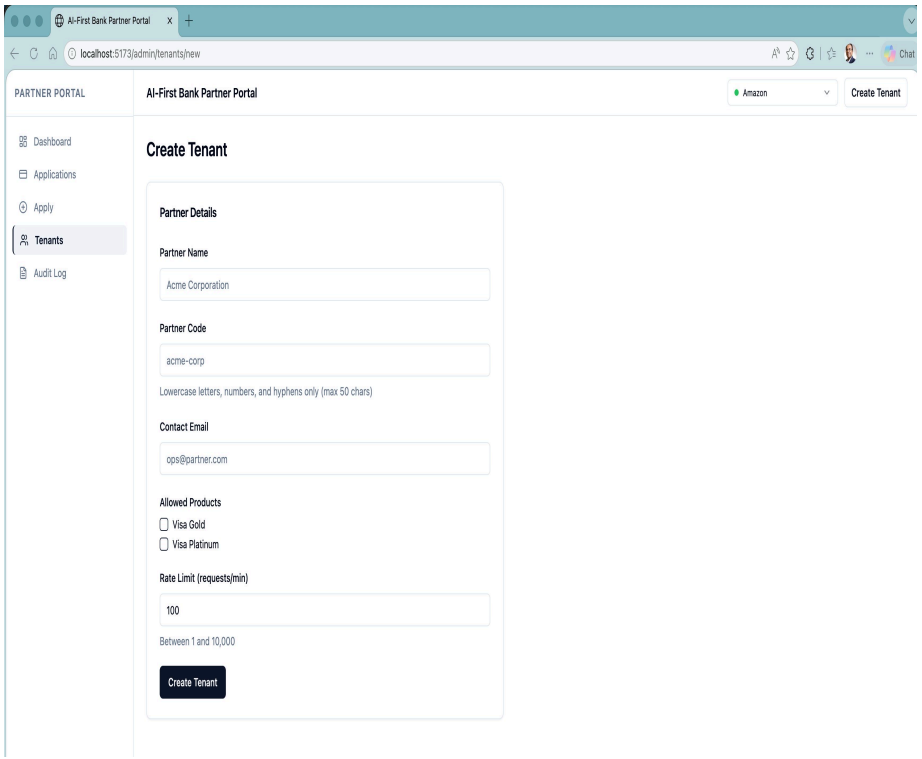
LAYER 1 UI Walkthrough Apply & Tenants

wipro intelligence™
proof over promise.

5-Step Apply Wizard + Multi-Tenant Admin



Apply (5 steps: Applicant → Employment → Product → Consent → Review)

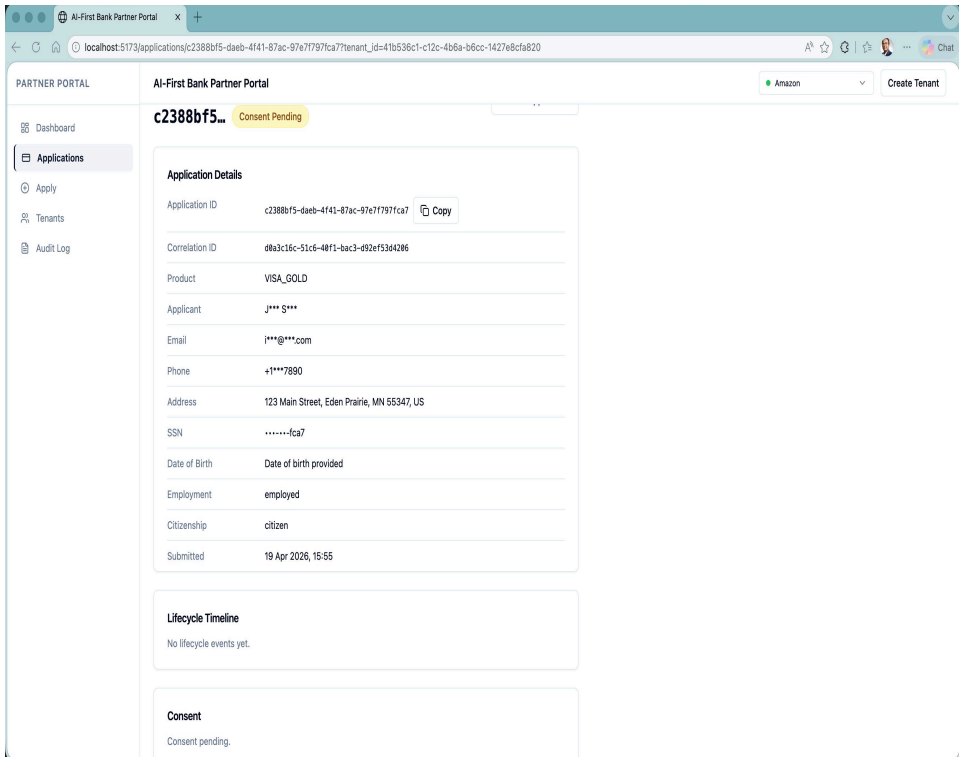


Create Tenant (partner provisioning, OAuth credentials, product allowlist)

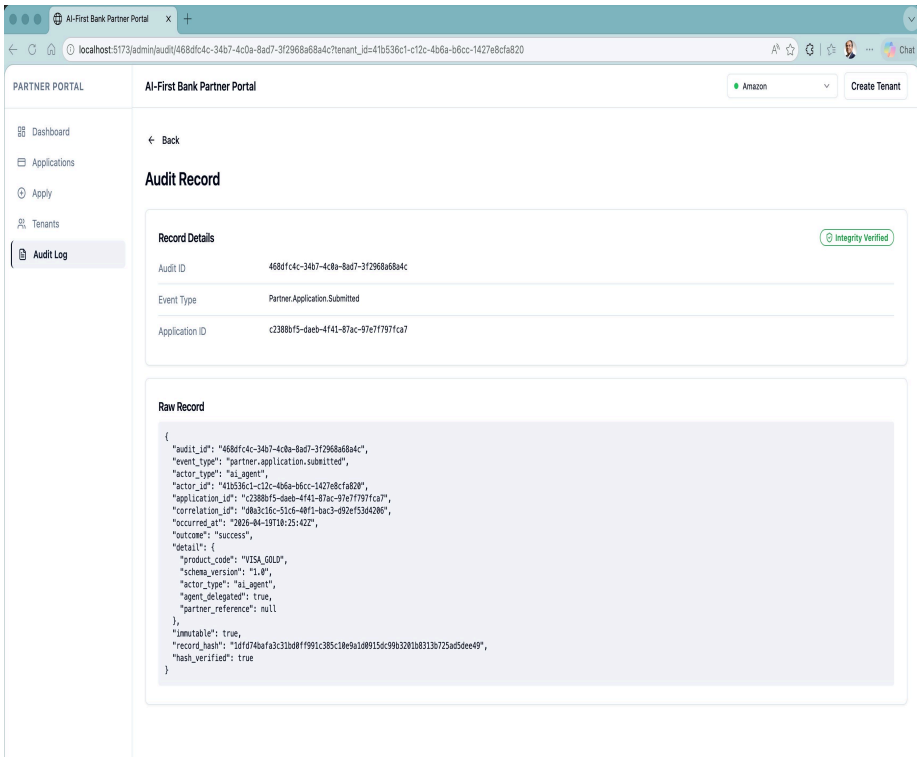
LAYER 1 UI Walkthrough Application & Audit

wipro intelligence™
proof over promise.

PII Masking, Lifecycle Tracking, Cryptographic Audit



Application Detail (SSN masked, lifecycle timeline, consent status)



Audit Record (Integrity Verified badge, record_hash, immutable)

Security, Privacy & Compliance

wipro intelligence™
proof over promise.

Privacy-by-design enforced at every layer — not bolted on

PII & SPI Protection

- SSN, DOB tokenized at MCP ingress
- AES-256-GCM vault
- SecretStr in Pydantic models
- Masked in UI (•••-••-XXXX)

Consent Framework

- 3 consent types captured as receipts
- SHA-256 receipt hashing
- State machine enforces consent gate
- Revocable, versioned (v1.0)

Auth & Authorization

- OAuth 2.0 client credentials
- Azure AD JWT (production)
- Per-partner client_secret (bcrypt)
- RBAC scoped to MCP tools

Audit & Traceability

- Immutable append-only audit trail
- correlation_id through every layer
- SHA-256 record_hash per event
- UI shows Integrity Verified badge

What We Delivered with AI-DLC

wipro intelligence™
proof over promise.

4	19	11	14	8
Repos	MCP Tools	Workflows	REST Endpoints	UI Screens
UI · Agent · Gateway · MCP	Across 5 banking domains	1 submit + 10 query	Agent FastAPI surface	Admin + Partner Ops

AI-DLC + MCP Gateway = Governed velocity for regulated banking.
AI drives the workflow. Humans own every critical decision. PII never enters the LLM. Testing is part of the lifecycle — not an afterthought.

Questions & Live Demo

wipro intelligence™

proof over promise.

Thank you.