

## Case Study

### "Project Gram-Uday"

**Bharat-Setu Bank** has secured **\$200M in Series B funding** to **bridge** the **"Digital Divide"** in India.

The **mission** is **dual-pronged**: delivering a **high-speed "Super-App"** for **Tier-1 urban youth** while providing **resilient "Micro-Credit"** for **Tier-4 rural farmers**.

**The Deadline**: You have **180 days** to go live. **Failure to launch** within this window results in the **revocation** of the **RBI banking license**.

### Strategic Objectives

#### 1. The Urban "Speed" Play

- **Target**: **5 million** Gen-Z & Millennial users.
- **The Product**: A **"Super-App"** providing **instant personal loans (< 2 mins)**, **UPI**, and **wealth management**.
- **Performance Targets**:
  - **Throughput**: Must **sustain 50,000 TPS** during **national sales (e.g., Diwali)**.
  - **Latency**: **API response** for **UPI** must be **< 100ms** (*99th percentile*).
  - **Scalability**: **Auto-scale** from **5,000 to 50,000 TPS** in under **3 minutes**.

#### 2. The Rural "Inclusion" Play

- **Target**: **10,000 "Gram-Sakhi" agents** serving **50,000 farmers** in **"Dark Zones"** (*No/Low internet*).
- **The Product**: **Biometric handheld tablets** and **"Bank-in-a-Box" kiosks** for **cash-in/cash-out** and **micro-loans**.
- **Performance Targets**:
  - **Offline Resilience**: **Agents** must be able to **process** up to **100 offline transactions** **before** requiring a **sync**.
  - **Package Size**: **Rural Android APK** must be **< 20MB** to **allow updates** over **2G speeds**.
  - **Sync Integrity**: **Zero data loss** during **"Store-and-Forward" syncs**; must **resolve conflicts** (*Double-Spending*) in **< 1 second** upon **reconnection**.

#### 3. The "Architect's Guardrails" (Constraints)

##### A. Regulatory & Sovereignty

- **Data Residency**: **100% of data** (*including logs/metadata*) must **stay within Indian Geography**.
- **Aadhaar Vault**: **Implement** an isolated **Aadhaar Data Vault (ADV)**. **Never store a customer's real Aadhaar number** in your **main database**.

## B. Integration

- **NPCI Connectivity (The Payment Links)**: Your bank must "speak the same language" as the rest of India.
- You must connect to:
  - **UPI for instant mobile payments.**
  - Rural **farmers withdraw cash** using just their **fingerprint**.

## C. Operational & Security

- **High Availability**: **99.99% Uptime**. No maintenance windows allowed.
- **Deployment**: Mandatory **Blue-Green deployment** for the **core** and **Canary** for the **Super-App**.
- **Security**: **Zero-Trust** architecture. **Every internal call must use mTLS.**
- **Fraud**: A **real-time Fraud Management** hook must **intercept** and score **transactions** in **< 50ms**.

## 4. Infrastructure & Resilience Constraints

To comply with RBI's **Framework**, the architecture must implement a strict **Data Center (DC) and Disaster Recovery (DR)** model within India.

- **DC-DR Architecture**:
  - **Primary DC**: **Active site** (e.g., Mumbai).
  - **Secondary DR**: **Hot-Standby site** (e.g., Hyderabad) with **real-time data replication**.
  - **RPO (Recovery Point Objective)**: **< 1 second** (*Max data loss in a catastrophe*).
  - **RTO (Recovery Time Objective)**: **< 15 minute** (*Max time to switch operations to DR*).

## 5. Detailed Latency & Performance Targets

The Solution Architect must ensure the system meets these end-to-end response times to avoid transaction timeouts at the NPCI or user level:

Channel / Service	Target Latency (99th Percentile)	Success Threshold
<b>UPI Payment</b>	<b>&lt; 100ms</b> (Internal processing)	<b>&lt; 2s</b> (End-to-End NPCI)
<b>ATM / AePS Withdrawal</b>	<b>&lt; 200ms</b> (Authorization)	Zero-Timeout at Terminal
<b>Mobile Banking App</b>	<b>&lt; 300ms</b> (Dashboard Load)	Under 3s on 3G
<b>Web Banking (Urban)</b>	<b>&lt; 500ms</b> (Transaction History)	No flickering / Instant UI
<b>Fraud Check (FMS)</b>	<b>&lt; 50ms</b>	Must block <i>before</i> ledger write
<b>Video KYC (Async)</b>	<b>&lt; 5 seconds</b> (Initial Buffer)	90% completion rate

## The Assignment: 12 Detailed Tasks

### Section 1: Strategic Mapping

- **Assignment 1:** Business Vision to Technical Vision.
- **Assignment 2:** Functional & Non-Functional Requirements.

### Section 2: Architectural Selection

- **Assignment 3:** Select Paradigm.
- **Assignment 4:** Select Model.
- **Assignment 5:** Select Architecture Style.
- **Assignment 6:** Select Architecture Pattern.

### Section 3: Technical Design & Flow

- **Assignment 7:** High-Level Design (HLD).
- **Assignment 8:** Low-Level Design (LLD).
- **Assignment 9:** Component & Service Selection.
- **Assignment 10:** Create 3 ADRs (Architectural Decision Records).

### Section 4: Visualizing the Flow

- **Assignment 11:** Create System Flow.
- **Assignment 12:** Final Architecture Picture.

## Model Answer

### 1. Business Vision to Technical Vision Mapping

Business Objective	Business Goal	Technical Vision	Strategy & Success Metric
<b>Delivery Speed (180 Days)</b>	<b>Complete</b> the <b>banking ecosystem</b> within <b>6 months</b> for the RBI license.	<b>Rapid Appliance Deployment</b>	<b>Deploy pre-configured, bank-grade software</b> stacks on <b>private servers</b> . Use modular services for the core ledger to minimize custom coding.
<b>Peak Load Management</b>	Sustain <b>50,000 TPS</b> during national peak periods.	<b>Private Cloud Elasticity</b>	Implement a <b>Private Cloud Orchestrator</b> on <b>bare metal</b> to spin up <b>additional service instances instantly</b> using pre-provisioned hardware.
<b>Real-time Transaction Flow</b>	<b>Internal Latency &lt; 100ms</b> for <b>&lt; 2s end-to-end UPI success</b> .	<b>Low-Latency Hardware Bus</b>	Use <b>high-speed In-Memory Data Grids</b> and a <b>physical 10Gbps/40Gbps backbone</b> to minimize network hops between internal services.
<b>Zero-Connectivity Access</b>	Process <b>100+ Transactions per device</b> in <b>network-dead zones</b> .	<b>Edge-to-Core Persistence</b>	Local-first <b>data storage</b> on <b>tablets</b> with a <b>Secure Queueing System</b> that <b>pushes data</b> to the <b>DC</b> the moment a <b>connection</b> is <b>detected</b> .
<b>Continuous Operations</b>	<b>99.99% Uptime</b> with zero downtime for updates.	<b>Hardware Redundancy &amp; Blue-Green</b>	Maintain <b>two identical production environments (A/B)</b> on separate hardware racks; switch traffic at the physical load balancer level.
<b>Extreme Disaster Recovery</b>	<b>RPO &lt; 1s</b> and <b>RTO &lt; 15 mins</b> during site failure.	<b>Metro-Cluster Mirroring</b>	Use <b>Synchronous Hardware Replication</b> over <b>dedicated fiber between two physical sites</b> ( <i>Mumbai DC and Hyderabad DR</i> ) for zero-lag data copies.
<b>Immediate Security Scoring</b>	Detect and <b>block fraud</b> in <b>&lt; 50ms</b> .	<b>Hardware-Accelerated Validation</b>	Deploy <b>security modules</b> directly on the entry firewall/load balancer <b>to filter</b> and score <b>transactions before</b> they <b>touch</b> the <b>application layer</b> .
<b>Regulatory Sovereignty</b>	<b>100% data residency</b> and zero raw PII in main storage.	<b>Physical Air-Gapping</b>	<b>Store sensitive IDs</b> in a <b>Physically Isolated Vault</b> ( <i>Dedicated Rack</i> ) with <b>restricted network access</b> ; use tokens for all cross-rack communication.

## 2. Mapping requirements

### i. Functional Requirements (The "What")

These are the core features required to bridge the Urban-Rural divide.

ID	Domain	Requirement Description
FR-01	Urban Payments	<b>Support</b> for <b>UPI transactions</b> with <b>instant debit/credit notifications</b> .
FR-02	Rural Access	<b>Biometric Authentication</b> for cash-out <b>using fingerprint</b> scanners on agent tablets.
FR-03	Offline Credit	<b>Enable agents</b> to <b>approve</b> and record <b>micro-loans</b> in <b>Offline Mode</b> with local digital signatures.
FR-04	KYC Compliance	Integrated <b>Video-KYC</b> and <b>Aadhaar-based E-KYC</b> flows for instant account opening.
FR-05	Data Privacy	A "Consent Management" module allowing users to grant or revoke data access to third parties.

### ii. Non-Functional Requirements (The "How Well")

These are the constraints that define the system's quality, performance, and reliability.

Category	ID	NFR Description	Technical Target
Availability	NFR-01	High Availability	<b>99.99% Uptime</b> ( <i>Calculated over a rolling 12-month window</i> ).
Performance	NFR-02	Throughput (Peak)	Must process <b>50,000 TPS</b> at <b>70% CPU utilization</b> .
Latency	NFR-03	Internal Turnaround	<b>&lt; 100ms</b> for <b>internal API calls</b> ( <i>P99</i> ).
Security	NFR-04	Zero-Trust mTLS	All <b>internal data-in-motion</b> must be <b>encrypted</b> via <b>Mutual TLS</b> .
Resilience	NFR-05	Disaster Recovery	<b>RPO &lt; 1s</b> and <b>RTO &lt; 15 mins</b> using Hot-Standby sites.
Compliance	NFR-06	Data Sovereignty	<b>0%</b> of <b>PII data</b> or <b>logs</b> stored <b>outside On-Premise India DCs</b> .
Consistency	NFR-07	Data Integrity	<b>Strict ACID</b> for <b>financial ledgers</b> ; <b>Eventual Consistency</b> for <b>Rewards</b> .
Scalability	NFR-08	Vertical/Horizontal	Ability to <b>scale</b> to <b>100,000 TPS</b> by adding physical blades to the rack.

### iii. On-Premises Constraint Mapping

This table maps the **Non-Functional Requirements (NFRs)** to specific **Physical Infrastructure** choices required in a private Data Center environment.

NFR ID	Requirement	Infrastructure Choice (On-Premises)	Implementation Detail
NFR-02	50,000 TPS	Hardware Load Balancers (L4/L7)	Use dedicated hardware appliances for <b>SSL termination</b> and <b>traffic distribution</b> to offload CPU from application servers.
NFR-05	RPO < 1s / RTO < 15m	Fiber-Channel (FC) Backbone	<b>Dedicated dark-fiber link</b> between <b>Mumbai</b> and <b>Hyderabad</b> for <b>synchronous SAN-to-SAN storage replication</b> .
NFR-06	Sovereignty	Localized Monitoring Stack	Deployment of self-hosted <b>observability tools</b> ( <i>Log management/Metrics</i> ) <b>strictly</b> on <b>internal server</b> racks; zero external API calls.
NFR-08	Scalability	Hyper-Converged Infrastructure (HCI)	Use of <b>modular "blades"</b> that allow for <b>rapid physical expansion</b> of Compute and Storage without re-wiring the DC.

### iv. Traceability Matrix

This matrix ensures that every **Business Vision** from Step 1 is directly supported by the **Requirements** defined in Step 2.

Strategic Vision	Functional Requirements (FR)	Non-Functional Requirements (NFR)	Architectural Impact
"Rural Inclusion"	<b>FR-03:</b> Offline Credit & Biometric Auth.	<b>NFR-07:</b> Consistency/Sync Integrity.	Requires an <b>"Offline-First" sync engine</b> that <b>handles local data persistence</b> on tablets.
"Urban Speed"	<b>FR-01:</b> High-Speed UPI & Wealth Apps.	<b>NFR-02:</b> 50k TPS. <b>NFR-03:</b> <100ms Latency.	Requires <b>in-memory data grids</b> and <b>optimized network paths</b> within the DC.
"Regulatory Trust"	<b>FR-05:</b> Consent Management. <b>FR-04:</b> Video KYC.	<b>NFR-04:</b> Zero-Trust mTLS. <b>NFR-06:</b> Data Sovereignty.	<b>Requires physical air-gapping</b> of the <b>Aadhaar Vault</b> and internal traffic encryption.

### 3. Select Paradigm

We will utilize four distinct programming models to address the "Bi-Modal" nature of the bank.

Programming Model	Application Area	Technical Logic (How it works)	Real-World Banking Example	Key Benefit for the Project
<b>Reactive</b>	<b>Urban UPI &amp; Real-time Payments</b>	<b>Non-blocking I/O &amp; Event Loops:</b> Instead of 1 thread per user, it uses a small pool of threads to handle thousands of concurrent connections.	During a "Diwali Flash Sale," 50,000 urban users click "Pay" at once. The system processes them as events without crashing the server.	<b>High Throughput:</b> Reaches <b>50,000 TPS</b> using significantly less on-premises hardware.
<b>Object-Oriented (OOP)</b>	<b>Core Banking Ledger &amp; Loan Management</b>	<b>Encapsulation &amp; Inheritance:</b> Wraps data and logic into "Classes" (Blueprints) that represent real-world bank entities.	A RuralMicroLoan class inherits basic math from BaseLoan but adds specific "Seasonal Repayment" logic for farmers.	<b>Maintainability:</b> Makes the ledger organized and easy to audit for the <b>180-day RBI launch</b> .
<b>Functional</b>	<b>Fraud Detection &amp; Interest Calculation</b>	<b>Pure Functions &amp; Immutability:</b> Functions have no "side effects." The input always determines the output without changing external data.	A transaction for ₹5,000 is checked for fraud. The function calculates a score in <b>&lt; 50ms</b> without ever accidentally modifying the user's balance.	<b>Precision:</b> Ensures interest and fraud scores are <b>100% bug-free</b> and mathematically accurate.
<b>Declarative</b>	<b>DC-DR &amp; Infrastructure Automation</b>	<b>Desired State Configuration:</b> You define "what" you want (the end goal) rather than "how" to step-by-step configure it.	You define: "I need 100 payment services active." If a physical blade server in Mumbai fails, the system automatically starts 100 new ones in Hyderabad.	<b>Resilience:</b> Maintains <b>99.99% Uptime</b> and meets <b>RTO &lt; 15 mins</b> through automated self-healing.

## 4. Select Model

### i. Core Domains & Bounded Contexts

We have identified three primary domains to meet the **180-day launch** goal:

Domain	Context Type	Responsibility
<b>Core Banking Domain</b>	<b>Core</b>	The "Golden Ledger." Handles double-entry bookkeeping, interest accrual, and regulatory reporting. (Strict OOP & ACID).
<b>Payment &amp; UPI Domain</b>	<b>Generic</b>	High-speed message switching between the bank and NPCI. (Reactive & Functional).
<b>Customer Inclusion Domain</b>	<b>Supporting</b>	Manages Rural Agent (Gram-Sakhi) workflows, Offline-Sync, and Biometric (AePS) identity.

### ii. Domain Structural Model (Data & Behavior)

We decompose the bank into Bounded Contexts where **Data (Attributes)** and **Logic (Behaviors)** are strictly encapsulated.

Entity (Aggregate Root)	Attributes (Data)	Relationships	Behaviors (Logic)
<b>Account</b>	AccountID (Tokenized), Balance, Currency, Status, KYC_Level	1:N with <b>Transactions</b> , 1:1 with <b>Customer</b>	debit(), credit(), calculateInterest(), applyFreeze()
<b>Customer</b>	CIF_Number, Tokenized_Aadhaar, Consent_Artifact, Risk_Score	1:N with <b>Accounts</b> , 1:1 with <b>Biometric_Profile</b>	updateConsent(), upgradeKYC(), verifyIdentity()
<b>Transaction</b>	Txn_UUID, Amount, Timestamp, Channel_Type, Sync_Status	N:1 with <b>Account</b> , 1:1 with <b>Fraud_Report</b>	authorize(), idempotentCheck(), generateReceipt()

### iii. DDD Tactical Elements Applied to Bharat-Setu

This table defines how we maintain data integrity and consistency across the Urban and Rural segments.

DDD Element	Application in Bharat-Setu	Technical Purpose
<b>Aggregate Root</b>	<b>The "Account" Entity</b>	Acts as the gatekeeper. All transactions (UPI or Rural Cash) must pass through the Account Aggregate to ensure the balance never goes negative.
<b>Domain Events</b>	<b>TransactionCaptured</b>	A message triggered whenever a Rural agent performs an offline withdrawal. This event is "stored" and "forwarded" to the DC later.
<b>Repositories</b>	<b>Ledger Repository</b>	Encapsulates the logic for saving to the on-premises ACID-compliant RDBMS, hiding the database complexity from the business logic.
<b>Domain Services</b>	<b>Interest Calculator</b>	A stateless service that calculates complex farm-loan interest. It doesn't belong to a single "Account" but operates on many.

### iv. Tactical Design: Entity vs. Value Object

Distinguishing between these two is critical for memory management on the "Gram-Sakhi" tablets and the 50,000 TPS Urban engine.

Concept	Definition	Banking Example	Architectural Behavior
<b>Entity</b>	Defined by a unique, thread-safe Identity that persists over time.	<b>Customer (CIF Number)</b>	Even if a customer changes their name or address, their ID remains the same in the <b>Aadhaar Vault</b> .
<b>Value Object</b>	Defined by its attributes; it has no identity and is <b>Immutable</b> .	<b>Transaction Amount</b>	A value of "₹500 INR" is just data. If you change it to ₹600, you create a <i>new</i> value object; you don't modify the old one.

#### v. Process Model: New User Registration Flow (Urban vs. Rural)

Registration is the "Front Door" of the bank and must satisfy the **180-day launch** requirement and **Aadhaar Vault** constraints.

Step	Urban Registration (Gen-Z)	Rural Registration (Farmer)	Data Compliance Action
<b>1. Identity</b>	User enters Aadhaar Number.	Agent scans Farmer's Fingerprint/Aadhaar.	Data is encrypted at the source (Tablet/Phone).
<b>2. Vaulting</b>	Sent to <b>Aadhaar Vault (ADV)</b> via secure API.	Queued for Sync (if offline) or sent to ADV.	<b>Tokenization:</b> Real Aadhaar is swapped for a Token.
<b>3. KYC</b>	<b>Async Video-KYC</b> (<5s initial buffer).	<b>E-KYC</b> via biometric authentication (AePS).	Complies with <b>RBI Master Directions</b> .
<b>4. Verification</b>	AI-based Document OCR & Face Match.	Account Aggregator (AA) pulls history.	Functional logic ensures <b>100% Accuracy</b> .
<b>5. Activation</b>	Instant Digital Virtual Debit Card.	Welcome Kit / Physical Card issued by Agent.	<b>Consent Artifact</b> created for DPDP 2023.

#### vi. Process Model: Transaction Process (Online vs. Offline Sync)

Phase	Urban Online (UPI/App)	Rural Offline (Tablet Sync)	Architectural Action
<b>1. Capture</b>	Real-time API request from Super-App.	Local database write on the Agent's Tablet (Edge).	<b>Offline:</b> Uses "Store & Forward" pattern on 2G.
<b>2. Validation</b>	Immediate Fraud Hook (<50ms).	Biometric match (Local/Cached) or Delayed Validation.	<b>Online:</b> Handled at DC; <b>Offline:</b> Handled at Device.
<b>3. Execution</b>	Funds blocked in Core Ledger (ACID).	Transaction queued in "Pending Sync" local bucket.	Uses <b>Reactive model</b> for Urban to handle 50k TPS.
<b>4. Synchronization</b>	N/A (Instant).	<b>Trigger:</b> App detects 3G/Wi-Fi connection.	<b>Conflict Resolution:</b> System checks for "Double Spending" in <1s.
<b>5. Consolidation</b>	Real-time DC-DR mirroring.	Batch replay of Domain Events to the DC Ledger.	Ensuring <b>RPO &lt;1s</b> once data reaches the Data Center.

### vii. Process Model: Banking Process Model (The Life of a Transaction)

This model describes the logical flow of a transaction from the "Edge" (User/Agent) to the "Core" (DC-DR Ledger).

Phase	Process Step	Architectural Action
<b>1. Initiation</b>	User triggers UPI or Agent captures Biometric.	Request hits the <b>Hardware Load Balancer</b> at the Primary DC.
<b>2. Validation</b>	Fraud & Identity Check.	<b>Fraud Hook (&lt;50ms)</b> and <b>Aadhaar Vault</b> tokenization occur.
<b>3. Processing</b>	Bounded Context Execution.	The <b>Payment Domain</b> executes the Reactive logic to verify funds.
<b>4. Persistence</b>	The "Golden Write".	The <b>Core Ledger (ACID)</b> records the transaction. Data is mirrored to the <b>DR Site (RPO &lt;1s)</b> .
<b>5. Integration</b>	External Handshake.	System communicates with <b>NPCI</b> (UPI/AePS) via ISO 8583/JSON.
<b>6. Fulfillment</b>	Notification & Rewards.	<b>Eventual Consistency</b> kicks in; User gets a push notification and "Reward Points" are updated.

## 5. Select Architectural Style

### i. Architecture Selection: Clean Microservices

Style / Pattern	Implementation	Justification for On-Premises
<b>Microservices</b>	Decoupled, autonomous services (UPI, Loan, KYC, Ledger).	<b>Physical Isolation:</b> We can dedicate specific physical blade servers to the "UPI Service" to handle <b>50,000 TPS</b> without high-intensity Video-KYC processing slowing down payments.
<b>Layered Architecture</b>	Internal 4-layer separation (Domain, App, Interface, Infra).	<b>Hardware Independence:</b> The "Infrastructure Layer" contains the drivers for our specific on-premises SAN storage and hardware load balancers.

## 6. Select Architectural Pattern

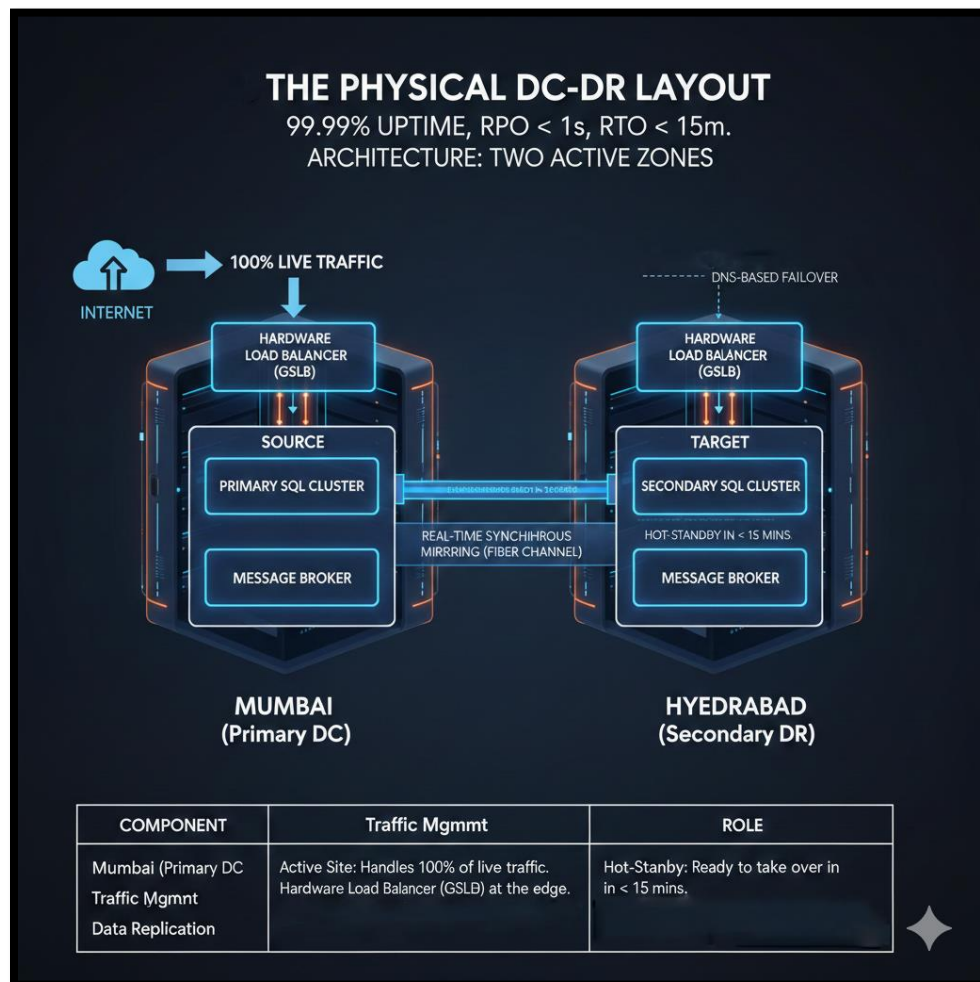
Pattern	Application Area	Justification (The "How it helps")	Example
<b>BFF (Backend for Frontend)</b>	<b>Mobile App vs. Agent Tablet</b>	<ul style="list-style-type: none"> <li>Creates <b>separate "Gateways"</b> for the <b>Urban App</b> and <b>Rural Tablet</b>.</li> <li>This keeps the <b>Urban App fast</b> and the <b>Rural Tablet light (&lt;20MB)</b>.</li> </ul>	<ul style="list-style-type: none"> <li>The Urban BFF handles heavy AI-wealth data; the Rural BFF only sends small, compressed binary data for 2G sync.</li> </ul>
<b>Sidecar Pattern</b>	<b>Security &amp; Observability</b>	<ul style="list-style-type: none"> <li><b>Offloads "non-banking" tasks</b> (mTLS, Logging, Fraud Hook) to a separate container.</li> </ul>	<ul style="list-style-type: none"> <li>Meets <b>Zero-Trust (mTLS)</b> and the <b>&lt;50ms Fraud Check</b> without slowing down the main Banking Java code.</li> </ul>
<b>Circuit Breaker</b>	<b>NPCI &amp; External Links</b>	<ul style="list-style-type: none"> <li>Prevents a failure in one external link (e.g., UPI Switch) from crashing the entire bank.</li> </ul>	<ul style="list-style-type: none"> <li>If the NPCI UPI server is slow, the Circuit Breaker "trips," instantly showing users a "Maintenance" message instead of a "Loading" spinner.</li> </ul>
<b>Saga (Orchestration)</b>	<b>Distributed Transactions</b>	<ul style="list-style-type: none"> <li>Ensures "All-or-Nothing" consistency across different microservices without a shared database.</li> </ul>	<ul style="list-style-type: none"> <li><b>Loan Disbursal:</b> Step 1: Debit Ledger Microservice → Step 2: Update Loan Microservice. If Step 2 fails, Step 1 is automatically reversed.</li> </ul>
<b>CQRS</b>	<b>High-Volume Urban Traffic</b>	<ul style="list-style-type: none"> <li>Segregates "Writes" (Payments) from "Reads" (Balance Checks).</li> </ul>	<ul style="list-style-type: none"> <li>Users check their balance 10x more than they pay. CQRS keeps the "Read" traffic from slowing down the "Payment" engine to hit <b>50,000 TPS</b>.</li> </ul>
<b>Observer</b>	<b>Real-time Notifications</b>	<ul style="list-style-type: none"> <li>Allows a service to "listen" for changes in another service without being tightly coupled.</li> </ul>	<ul style="list-style-type: none"> <li>When the <b>Ledger</b> finishes a transaction, the <b>Notification Service</b> "observes" this event and instantly sends an SMS/Push notification to the user.</li> </ul>

## 7. Create High-Level Design (HLD)

### i. The Physical DC-DR Layout

To meet the **99.99% Uptime** and **RPO < 1s / RTO < 15m** requirements, the architecture is split into two active zones.

Component	Mumbai (Primary DC)	Hyderabad (Secondary DR)
Role	<b>Active Site:</b> Handles 100% of live traffic.	<b>Hot-Standby:</b> Ready to take over in < 15 mins.
Traffic Management	Hardware Load Balancer (GSLB) at the edge.	Synchronized GSLB for DNS-based failover.
Data Replication	<b>Source:</b> Primary SQL Cluster + Message Broker.	<b>Target:</b> Real-time Synchronous Mirroring (Fiber Channel).



## ii. Logical Architecture Layers

### a) The Edge Layer (Security & Traffic)

- **Hardware Load Balancer:** Terminates SSL/TLS.
- **API Gateway:** Implements the **Fraud Management Hook (<50ms)**.
- **BFF (Backend for Frontend):** Segregates Urban (High-speed JSON) and Rural (Low-speed Binary) traffic.

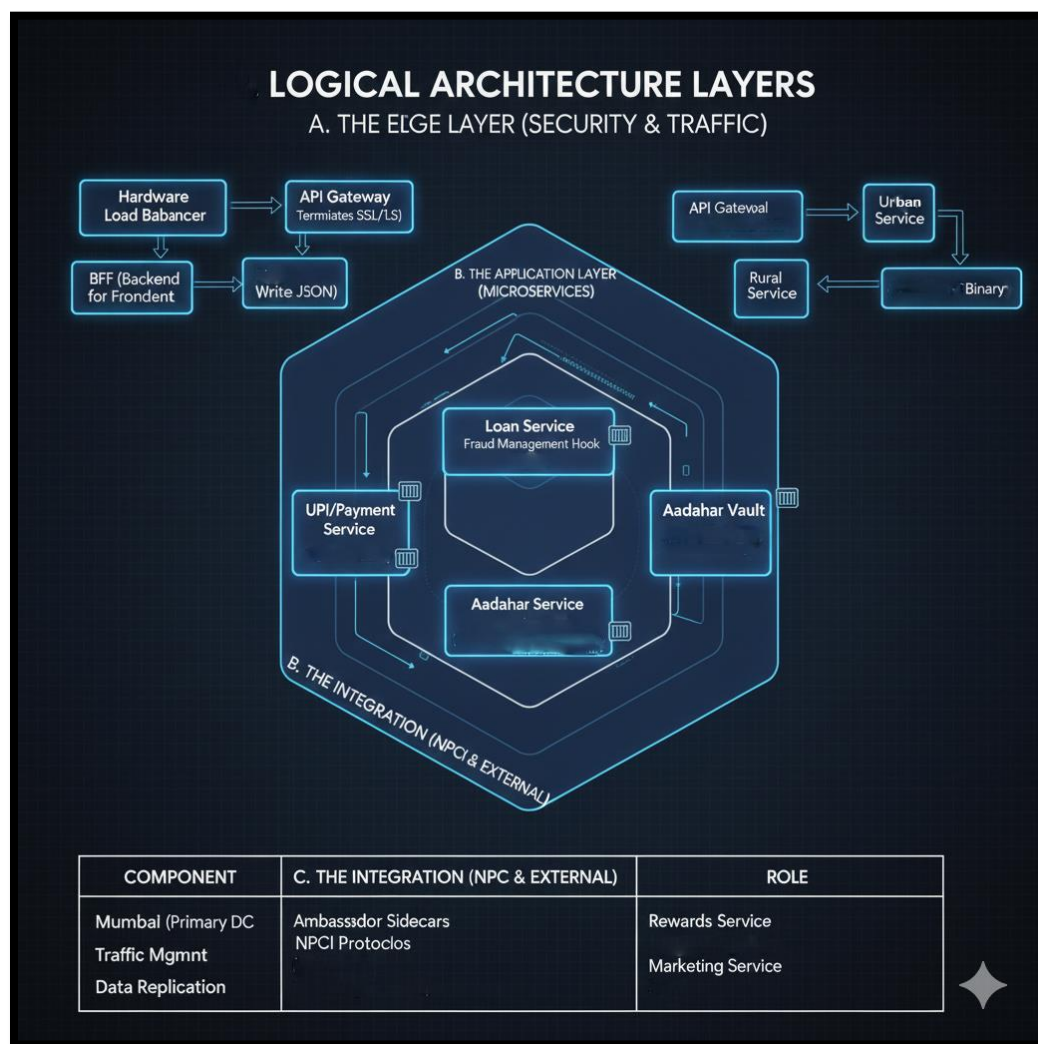
### b) The Application Layer (Microservices)

Each service runs in its own container/pod with a **Sidecar** attached.

- **UPI/Payment Service:** Uses **CQRS** to send "Read" queries to a cache and "Write" commands to the ledger.
- **Loan Service:** Uses the **Saga Orchestrator** to manage multi-step farm loan approvals.
- **Aadhaar Vault Service:** An isolated subnet containing the **Aadhaar Data Vault (ADV)** with tokenization logic.

### c) The Integration Layer (NPCI & External)

- **Ambassador Sidecars:** Act as translators to speak the NPCI protocols (ISO 8583) and connect to **Account Aggregators (AA)**.
- **Observer Hub:** A message broker (Kafka/RabbitMQ) that lets the Rewards and Marketing services "listen" to transaction successes.



### iii. Data & Persistence Model

Service Type	Database Pattern	Replication Strategy
Core Ledger	ACID-compliant RDBMS	<b>Synchronous Mirroring:</b> Every "Write" must be acknowledged by the Hyderabad DR before it is finalized in Mumbai.
Urban Dashboard	In-Memory Cache (Redis)	<b>Async Replication:</b> Fast reads for the urban dashboard.
Rural Sync	Outbox / Persistent Queue	<b>Exactly-once Delivery:</b> Ensures offline transactions are replayed correctly.

#### iv. Compliance & Guardrail Check

- I. **Data Residency:** Both DC and DR sites are physically located in India.
- II. **Aadhaar Privacy:** Real Aadhaar numbers never leave the isolated **ADV Subnet**; only **Tokens** travel to the UPI service.
- III. **Cloud Agnostic:** Every component is containerized. If we need to move to the cloud, we simply "Lift and Shift" the containers and point them to the new cloud database.

### 8. Create High-Level Design (HLD)

#### i. LLD 1: Rural Offline-to-Online Sync Engine

This logic handles the "Dark Zone" scenarios for Gram-Sakhi agents to ensure zero data loss.

Component	Responsibility	Logic / Pseudocode	Data Structure
<b>Atomic Committer</b>	Ensures data integrity on the tablet.	BEGIN TRANSACTION; SAVE TransactionData; SAVE OutboxEvent(UUID, Payload); COMMIT;	<b>Local SQLite:</b> txn_table + outbox_table
<b>Connectivity Monitor</b>	Detects 2G/3G/Wi-Fi signal.	If (SignalStrength > -100dBm) {TriggerSync();}	<b>Native OS API:</b> Android Connectivity Manager
<b>Sync Worker</b>	Pushes data to DC.	Reads outbox_table by created_at ASC; Sends via gRPC; Deletes on HTTP 200.	<b>Buffer:</b> Queue-based (FIFO)
<b>Idempotency Guard</b>	Prevents double spending at DC.	If (Redis.Exists(EventUUID)) {Return Success;} Else {ProcessAndCache(EventUUID);}	<b>Cache:</b> Redis Key-Value (TTL: 24hrs)

#### ii. LLD 2: Security & Aadhaar Tokenization (The Vault)

This ensures the primary database remains compliant with RBI by never storing a real Aadhaar number.

Logical Step	Process Name	Internal Logic	Data Output
<b>1. Ingress</b>	Encryption	Encrypt raw Aadhaar using the Public Key of the Vault.	Encrypted_Blob
<b>2. Verification</b>	Checksum Logic	Validate Aadhaar.	Boolean (True/False)

<b>3. Storage</b>	HSM Vaulting	Store raw value in Hardware Security Module.	Token_ID
<b>4. Detokenization</b>	Restricted Retrieval	Only authorized "Compliance Service" can trade Token for Value.	PII (Restricted Access)

### iii. LLD 3: CQRS Read-Side Projection (Urban Performance)

This logic ensures the Urban Dashboard loads in **< 300ms** even if the main ledger is busy.

Trigger Event	Logic Path	Action	Technology Used
<b>Write Event</b>	Command Side	Debit (AccountID, 500) -> Commit to SQL.	PostgreSQL (ACID)
<b>Event Emit</b>	Message Bus	Publish BalanceChanged event to Broker.	Kafka Topic
<b>Projection</b>	Consumer	Listen to Broker; Fetch current balance; Compute total.	Java/Go Consumer
<b>View Update</b>	Read Store	SET User_Balance_{ID} = NewBalance in Memory.	Redis (In-Memory)

### iv. LLD 4: The Saga Compensating Logic (Distributed Integrity)

Since we use microservices, this "Undo" logic ensures the bank is always in balance.

Current State	Failure Point	Compensating Action (The "Undo")	Success Status
<b>Money Debited</b>	SMS/Notification Fails	<b>Continue:</b> Not a financial failure; retry SMS.	<b>Success</b>
<b>Money Debited</b>	External Switch Fails	<b>Reverse:</b> Credit (AccountID, Amount) with tag SAGA_REVERSAL.	<b>Rollback</b>
<b>Loan Approved</b>	Disbursement Fails	<b>Reverse:</b> Move Loan Status back to Rejected/Pending.	<b>Rollback</b>

### v. LLD 5: Component Interface Definitions (APIs)

Service	Endpoint	Method	payload Example
<b>KYC Service</b>	/v1/identity/tokenise	POST	{"raw_identity": "ENC_BLOB"}
<b>Sync Service</b>	/v1/rural/sync	gRPC	Stream<OfflineEvents>

<b>Payment Service</b>	/v1/pay/initiate	POST	{"token": "...", "amount": 500}
------------------------	------------------	------	---------------------------------

## 9. Component & Service Selection

Architecture Layer	Component / Data Tier	Purpose & Logic	Technology / Protocol	Strategy (On-Premises)
<b>Traffic Management</b>	<b>GSLB &amp; DNS</b>	Global site routing and failover between Mumbai and Hyderabad.	<b>F5 Big-IP</b>	<b>Active-Passive:</b> DNS TTL set to 300s for <15 min RTO failover.
<b>Traffic Management</b>	<b>Local Load Balancer</b>	Distributes traffic across local server blades.	<b>Nginx</b>	<b>Weighted Round Robin:</b> Balances 50k TPS across the rack.
<b>Network Protocol</b>	<b>Urban Channel</b>	High-speed app communication for Gen-Z users.	<b>REST</b>	Optimized for 5G/Fiber with rich data payloads.
<b>Network Protocol</b>	<b>Rural Channel</b>	Low-bandwidth communication for agents in 2G areas.	<b>gRPC</b>	<b>Binary Compression:</b> Reduces payload size by 70% vs JSON.
<b>Network Protocol</b>	<b>Integration / NPCI</b>	Communication with the national banking switch.	<b>ISO 8583</b>	Fixed-length bitmaps required for clearing & settlement.
<b>Persistence Tier</b>	<b>Financial Ledger</b>	The "Golden Source" of all money movement.	<b>PostgreSQL</b>	<b>Synchronous:</b> Fiber-channel mirroring for zero data loss (RPO <1s).
<b>Persistence Tier</b>	<b>Profiles &amp; KYC</b>	Flexible storage for Aadhaar tokens and user metadata.	<b>MongoDB</b>	<b>Asynchronous:</b> Background replication to DR to save bandwidth.
<b>Persistence Tier</b>	<b>Urban Dashboard</b>	High-frequency "Read" access for balance checks.	<b>Redis</b>	<b>CQRS Pattern:</b> Offloads read traffic from the main Ledger.
<b>Persistence Tier</b>	<b>Rural Sync Store</b>	Local outbox storage on agent tablets.	<b>SQLite (On-Device)</b>	<b>Store-and-Forward:</b> Atomic local writes with batch sync.

## 10.ADRs (Architectural Decision Records)

Feature	PostgreSQL	MongoDB	Redis
<b>Decision</b>	Use as the Primary System of Record.	Use for KYC and User Metadata.	Use as a Cache and Session Store.
<b>Rationale</b>	Strict <b>ACID compliance</b> is mandatory for banking licenses. Handles complex SQL joins for audits.	<b>Flexible Schema</b> allows adding new KYC fields (e.g., Video-KYC links) without downtime.	<b>Sub-millisecond latency</b> handles the "Urban Speed" requirement of 50k TPS.
<b>Sync Logic</b>	<b>Synchronous Mirroring</b> to DR site.	<b>Asynchronous Replication</b> (Active-Active).	<b>Local Persistence</b> (AOF) + In-memory replication.
<b>Conflict Handling</b>	Strong Consistency (No conflicts allowed).	Last-Write-Wins (Eventual Consistency).	N/A (Transient data).
<b>Constraint</b>	Fixed schema; migration requires planning.	Max document size 16MB.	Dataset size limited by Physical RAM.

## 11.System Flow

### i. Step-by-Step Transaction Flow (The "Golden Path")

- I. **Urban User** initiates a ₹500 payment via the Super-App.
- II. **API Gateway** runs a **Fraud Check (<50ms)** using a Sidecar.
- III. **Payment Microservice** checks balance via the **CQRS Read-Store**.
- IV. **Saga Orchestrator** starts:
  - a. Command 1: Debit Account (Ledger Service).
  - b. Command 2: Update NPCI Status (Ambassador Service).
- V. **Data Mirroring**: The Ledger write is mirrored to Hyderabad over the **Fiber-Channel** link.
- VI. **Observer**: Once successful, the **Notification Service** observes the event and sends an SMS.

## 12. Final Architecture Picture

### i. The "Bharat-Setu" Unified Architecture Table

This table represents the final state of the bank's technical ecosystem, combining every decision made from Step 1 to Step 11.

Architectural Layer	Selected Component	Final Decision / Integration	Strategic Impact
Edge (Connectivity)	BFF + GSLB	F5 Load Balancers routing to <b>Urban (JSON)</b> and <b>Rural (gRPC)</b> endpoints.	<b>Urban Speed:</b> <100ms Latency.
Security	Zero-Trust Sidecar	Envoy sidecars handling <b>mTLS</b> and <b>HSM-backed Tokenization</b> .	<b>Regulatory Trust:</b> Aadhaar Vault Compliance.
Application Style	Clean Microservices	Polyglot (Java for Ledger, Go for Payments, Python for AI).	<b>180-Day Launch:</b> Parallel development teams.
Data Persistence	Polyglot Trio	<b>Postgres</b> (Ledger), <b>Mongo</b> (KYC), <b>Redis</b> (Cache).	<b>Scale:</b> 50,000 TPS Urban Peak.
Rural Resilience	Offline-First Sync	Transactional Outbox + SQLite Store-and-Forward.	<b>Rural Inclusion:</b> Zero data loss on 2G.
Reliability	DC-DR Mirroring	Mumbai (Active) to Hyderabad (Hot Standby) via Dark Fiber.	<b>99.99% Uptime:</b> RPO <1s / RTO <15m.

### ii. Final High-Level Flow (The "Lifecycle of a Rupee")

To visualize the final architecture, we track a single transaction through the entire on-premises stack:

- Request:** A user in a rural village initiates a ₹1,000 loan repayment on a 2G network.
- Edge (Rural):** The request uses **gRPC (Binary)** to minimize data size.
- BFF Layer:** The Rural BFF validates the device signature and hands it to the **Sync Service**.
- Security Layer:** The **Aadhaar Sidecar** verifies the user's tokenized identity via the **HSM**.
- Core Logic:** The **Ledger Microservice (Java)** performs a double-entry update using **Clean Architecture** rules.
- Persistence:**
  - Write:** Committed to the **PostgreSQL Ledger** and mirrored to Hyderabad.
  - Update:** The **CQRS Projection** updates the **Redis Cache** for the user's dashboard.

7. **Observer Hub:** Kafka emits a PaymentSuccess event.
8. **Downstream:** The **Rewards Service** credits points, and the **SMS Service** notifies the user.

### iii. Strategic "Success Check" (Meeting the Guardrails)

Business Mandate	Architectural Proof	Result
<b>50,000 TPS</b>	<b>CQRS + Redis + Hyper-Converged Infrastructure</b>	Hardware-accelerated read/writes handle urban peaks.
<b>Offline Rural Play</b>	<b>SQLite Outbox + gRPC Binary Compression</b>	Reliable banking in 2G "Dark Zones."
<b>180-Day Launch</b>	<b>Strangler Fig + Modular Microservices</b>	Decoupled services allow for a fast-track MVP launch.
<b>Cloud Agnostic</b>	<b>Infrastructure Abstraction (Hexagonal Layering)</b>	Core logic is 100% portable to Cloud in <30 days.