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
UPI Payment	**< 100ms** (Internal processing)	< 2s (End-to-End NPCI)
ATM / AePS Withdrawal	**< 200ms** (Authorization)	Zero-Timeout at Terminal
Mobile Banking App	**< 300ms** (Dashboard Load)	Under 3s on 3G
Web Banking (Urban)	**< 500ms** (Transaction History)	No flickering / Instant UI
Fraud Check (FMS)	**< 50ms**	Must block *before* ledger write
Video KYC (Async)	**< 5 seconds** (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
Delivery Speed (180 Days)	**Complete** the **banking ecosystem** within **6 months** for the RBI license.	**Rapid Appliance Deployment**	**Deploy pre-configured**, **bank-grade software** stacks on **private servers**. Use modular services for the core ledger to minimize custom coding.
Peak Load Management	Sustain **50,000 TPS** during national peak periods.	**Private Cloud Elasticity**	Implement a **Private Cloud Orchestrator** on **bare metal** to spin up **additional service** i**nstances instantly** using pre-provisioned hardware.
Real-time Transaction Flow	**Internal Latency < 100ms** for **< 2s end-to-end UPI success**.	**Low-Latency Hardware Bus**	Use **high-speed In-Memory Data Grids** and a **physical 10Gbps/40Gbps backbone** to minimize network hops between internal services.
Zero-Connectivity Access	Process **100+ Transactions per device** in **network-dead zones**.	**Edge-to-Core Persistence**	Local-first **data storage** on **tablets** with a **Secure Queueing System** that **pushes data** to the **DC** the moment **a connection** is **detected**.
Continuous Operations	**99.99% Uptime** with zero downtime for updates.	**Hardware Redundancy & Blue-Green**	Maintain **two identical production environments** (*A/B*) on separate hardware racks; switch traffic at the physical load balancer level.
Extreme Disaster Recovery	**RPO < 1s** and **RTO < 15 mins** during site failure.	**Metro-Cluster Mirroring**	Use **Synchronous Hardware Replication** over **dedicated fiber between two physical sites** (*Mumbai DC and Hyderabad DR*) for zero-lag data copies.
Immediate Security Scoring	Detect and **block fraud** in **< 50ms.**	**Hardware-Accelerated Validation**	Deploy **security modules** directly on the entry firewall/load balancer **to filter** and score **transactions *before*** they **touch** the **application layer**.
Regulatory Sovereignty	**100% data residency** and zero raw PII in main storage.	**Physical Air-Gapping**	**Store sensitive IDs** in a **Physically Isolated Vault** (*Dedicated Rack*) with **restricted network access**; 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**	**Support** for **UPI transactions** with **instant debit/credit notifications**.
FR-02	**Rural Access**	**Biometric Authentication** for cash-out **using fingerprint** scanners on agent tablets.
FR-03	**Offline Credit**	**Enable agents** to **approve** and record **micro-loans** in **Offline Mode** with local digital signatures.
FR-04	**KYC Compliance**	Integrated **Video-KYC** and **Aadhaar-based E-KYC** 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**	**99.99% Uptime** (*Calculated over a rolling 12-month window*).
Performance	**NFR-02**	**Throughput (Peak)**	Must process **50,000 TPS** at **70% CPU utilization**.
Latency	**NFR-03**	**Internal Turnaround**	**< 100ms** for **internal API calls** (*P99*).
Security	**NFR-04**	**Zero-Trust mTLS**	All **internal data-in-motion** must be **encrypted** via **Mutual TLS**.
Resilience	**NFR-05**	**Disaster Recovery**	**RPO < 1s** and **RTO < 15 mins** using Hot-Standby sites.
Compliance	**NFR-06**	**Data Sovereignty**	**0%** of **PII data** or **logs** stored **outside** **On-Premise India DCs**.
Consistency	**NFR-07**	**Data Integrity**	**Strict ACID** for **financial ledgers**; **Eventual Consistency** for **Rewards**.
Scalability	**NFR-08**	**Vertical/Horizontal**	Ability to **scale** to **100,000 TPS** 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 **SSL termination** and **traffic distribution** to offload CPU from application servers.
NFR-05	**RPO < 1s / RTO < 15m**	**Fiber-Channel (FC) Backbone**	**Dedicated dark-fiber link** between **Mumbai** and Hyderabad for **synchronous SAN-to-SAN storage replication**.
NFR-06	**Sovereignty**	**Localized Monitoring Stack**	Deployment of self-hosted **observability tools** (*Log management/Metrics*) **strictly** on **internal server** racks; zero external API calls.
NFR-08	**Scalability**	**Hyper-Converged Infrastructure (HCI)**	Use of **modular "blades"** that allow for **rapid physical expansion** 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"	**FR-03:** Offline Credit & Biometric Auth.	**NFR-07:** Consistency/Sync Integrity.	Requires an **"Offline-First" sync engine** that **handles local data persistence** on tablets.
"Urban Speed"	**FR-01:** High-Speed UPI & Wealth Apps.	**NFR-02:** 50k TPS. **NFR-03:** <100ms Latency.	Requires **in-memory data grids** and **optimized network paths** within the DC.
"Regulatory Trust"	**FR-05:** Consent Management. **FR-04:** Video KYC.	**NFR-04:** Zero-Trust mTLS. **NFR-06:** Data Sovereignty.	**Requires physical air-gapping** of the **Aadhaar Vault** 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
Reactive	**Urban UPI & Real-time Payments**	**Non-blocking I/O & Event Loops:** 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.	**High Throughput:** Reaches **50,000 TPS** using significantly less on-premises hardware.
Object-Oriented (OOP)	**Core Banking Ledger & Loan Management**	**Encapsulation & Inheritance:** 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.	**Maintainability:** Makes the ledger organized and easy to audit for the **180-day RBI launch**.
Functional	**Fraud Detection & Interest Calculation**	**Pure Functions & Immutability:** 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 **< 50ms** without ever accidentally modifying the user's balance.	**Precision:** Ensures interest and fraud scores are **100% bug-free** and mathematically accurate.
Declarative	**DC-DR & Infrastructure Automation**	**Desired State Configuration:** 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.	**Resilience:** Maintains **99.99% Uptime** and meets **RTO < 15 mins** 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
Core Banking Domain	**Core**	The "Golden Ledger." Handles double-entry bookkeeping, interest accrual, and regulatory reporting. (Strict OOP & ACID).
Payment & UPI Domain	**Generic**	High-speed message switching between the bank and NPCI. (Reactive & Functional).
Customer Inclusion Domain	**Supporting**	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)
Account	AccountID (Tokenized), Balance, Currency, Status, KYC_Level	1:N with **Transactions**, 1:1 with **Customer**	debit(), credit(), calculateInterest(), applyFreeze()
Customer	CIF_Number, Tokenized_Aadhaar, Consent_Artifact, Risk_Score	1:N with **Accounts**, 1:1 with **Biometric_Profile**	updateConsent(), upgradeKYC(), verifyIdentity()
Transaction	Txn_UUID, Amount, Timestamp, Channel_Type, Sync_Status	N:1 with **Account**, 1:1 with **Fraud_Report**	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
Aggregate Root	**The "Account" Entity**	Acts as the gatekeeper. All transactions (UPI or Rural Cash) must pass through the Account Aggregate to ensure the balance never goes negative.
Domain Events	**TransactionCaptured**	A message triggered whenever a Rural agent performs an offline withdrawal. This event is "stored" and "forwarded" to the DC later.
Repositories	**Ledger Repository**	Encapsulates the logic for saving to the on-premises ACID-compliant RDBMS, hiding the database complexity from the business logic.
Domain Services	**Interest Calculator**	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
Entity	Defined by a unique, thread-safe Identity that persists over time.	**Customer (CIF Number)**	Even if a customer changes their name or address, their ID remains the same in the **Aadhaar Vault**.
Value Object	Defined by its attributes; it has no identity and is **Immutable**.	**Transaction Amount**	A value of "₹500 INR" is just data. If you change it to ₹600, you create a *new* 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
1. Identity	User enters Aadhaar Number.	Agent scans Farmer's Fingerprint/Aadhaar.	Data is encrypted at the source (Tablet/Phone).
2. Vaulting	Sent to **Aadhaar Vault (ADV)** via secure API.	Queued for Sync (if offline) or sent to ADV.	**Tokenization:** Real Aadhaar is swapped for a Token.
3. KYC	**Async Video-KYC** (<5s initial buffer).	**E-KYC** via biometric authentication (AePS).	Complies with **RBI Master Directions**.
4. Verification	AI-based Document OCR & Face Match.	Account Aggregator (AA) pulls history.	Functional logic ensures **100% Accuracy**.
5. Activation	Instant Digital Virtual Debit Card.	Welcome Kit / Physical Card issued by Agent.	**Consent Artifact** created for DPDP 2023.

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

Phase	Urban Online (UPI/App)	Rural Offline (Tablet Sync)	Architectural Action
1. Capture	Real-time API request from Super-App.	Local database write on the Agent's Tablet (Edge).	**Offline:** Uses "Store & Forward" pattern on 2G.
2. Validation	Immediate Fraud Hook (<50ms).	Biometric match (Local/Cached) or Delayed Validation.	**Online:** Handled at DC; **Offline:** Handled at Device.
3. Execution	Funds blocked in Core Ledger (ACID).	Transaction queued in "Pending Sync" local bucket.	Uses **Reactive model** for Urban to handle 50k TPS.
4. Synchronization	N/A (Instant).	**Trigger:** App detects 3G/Wi-Fi connection.	**Conflict Resolution:** System checks for "Double Spending" in <1s.
5. Consolidation	Real-time DC-DR mirroring.	Batch replay of Domain Events to the DC Ledger.	Ensuring **RPO <1s** 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
1. Initiation	User triggers UPI or Agent captures Biometric.	Request hits the **Hardware Load Balancer** at the Primary DC.
2. Validation	Fraud & Identity Check.	**Fraud Hook (<50ms)** and **Aadhaar Vault** tokenization occur.
3. Processing	Bounded Context Execution.	The **Payment Domain** executes the Reactive logic to verify funds.
4. Persistence	The "Golden Write".	The **Core Ledger (ACID)** records the transaction. Data is mirrored to the **DR Site (RPO <1s)**.
5. Integration	External Handshake.	System communicates with **NPCI** (UPI/AePS) via ISO 8583/JSON.
6. Fulfillment	Notification & Rewards.	**Eventual Consistency** 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
Microservices	Decoupled, autonomous services (UPI, Loan, KYC, Ledger).	**Physical Isolation:** We can dedicate specific physical blade servers to the "UPI Service" to handle **50,000 TPS** without high-intensity Video-KYC processing slowing down payments.
Layered Architecture	Internal 4-layer separation (Domain, App, Interface, Infra).	**Hardware Independence:** The "Infrastructure Layer" contains the drivers for our specific on-premises SAN storage and hardware load balancers.

6. Select Architectural Pattern

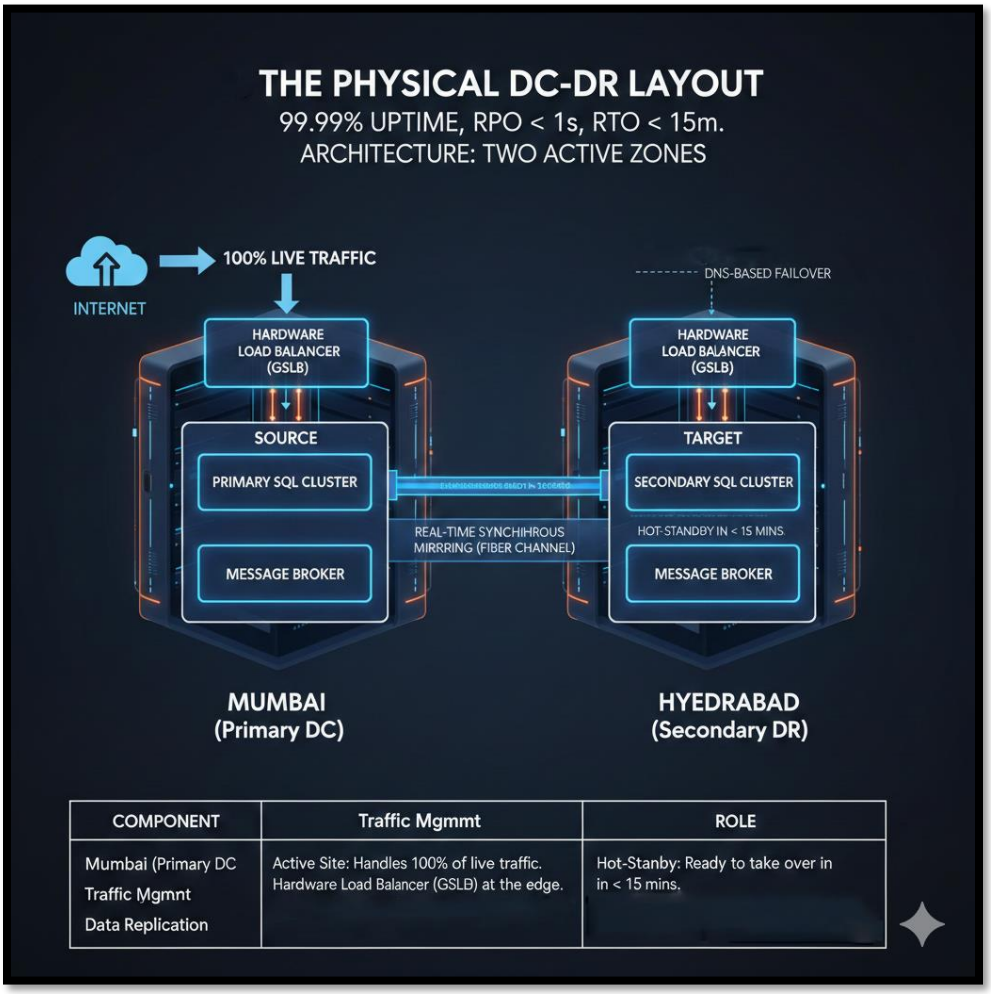
Pattern	Application Area	Justification (The "How it helps")	Example
BFF (Backend for Frontend)	**Mobile App vs. Agent Tablet**	• Creates **separate "Gateways"** for the **Urban App** and **Rural Tablet**. • This keeps the **Urban App fast** and the **Rural Tablet light (<20MB)**.	• The Urban BFF handles heavy AI-wealth data; the Rural BFF only sends small, compressed binary data for 2G sync.
Sidecar Pattern	**Security & Observability**	• **Offloads "non-banking" tasks** (mTLS, Logging, Fraud Hook) to a separate container.	• Meets **Zero-Trust (mTLS)** and the **<50ms Fraud Check** without slowing down the main Banking Java code.
Circuit Breaker	**NPCI & External Links**	• Prevents a failure in one external link (e.g., UPI Switch) from crashing the entire bank.	• If the NPCI UPI server is slow, the Circuit Breaker "trips," instantly showing users a "Maintenance" message instead of a "Loading" spinner.
Saga (Orchestration)	**Distributed Transactions**	• Ensures "All-or-Nothing" consistency across different microservices without a shared database.	• **Loan Disbursal:** Step 1: Debit Ledger Microservice → Step 2: Update Loan Microservice. If Step 2 fails, Step 1 is automatically reversed.
CQRS	**High-Volume Urban Traffic**	• Segregates "Writes" (Payments) from "Reads" (Balance Checks).	• Users check their balance 10x more than they pay. CQRS keeps the "Read" traffic from slowing down the "Payment" engine to hit **50,000 TPS**.
Observer	**Real-time Notifications**	• Allows a service to "listen" for changes in another service without being tightly coupled.	• When the **Ledger** finishes a transaction, the **Notification Service** "observes" this event and instantly sends an SMS/Push notification to the user.

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	**Active Site:** Handles 100% of live traffic.	**Hot-Standby:** Ready to take over in < 15 mins.
Traffic Management	Hardware Load Balancer (GSLB) at the edge.	Synchronized GSLB for DNS-based failover.
Data Replication	**Source:** Primary SQL Cluster + Message Broker.	**Target:** 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.

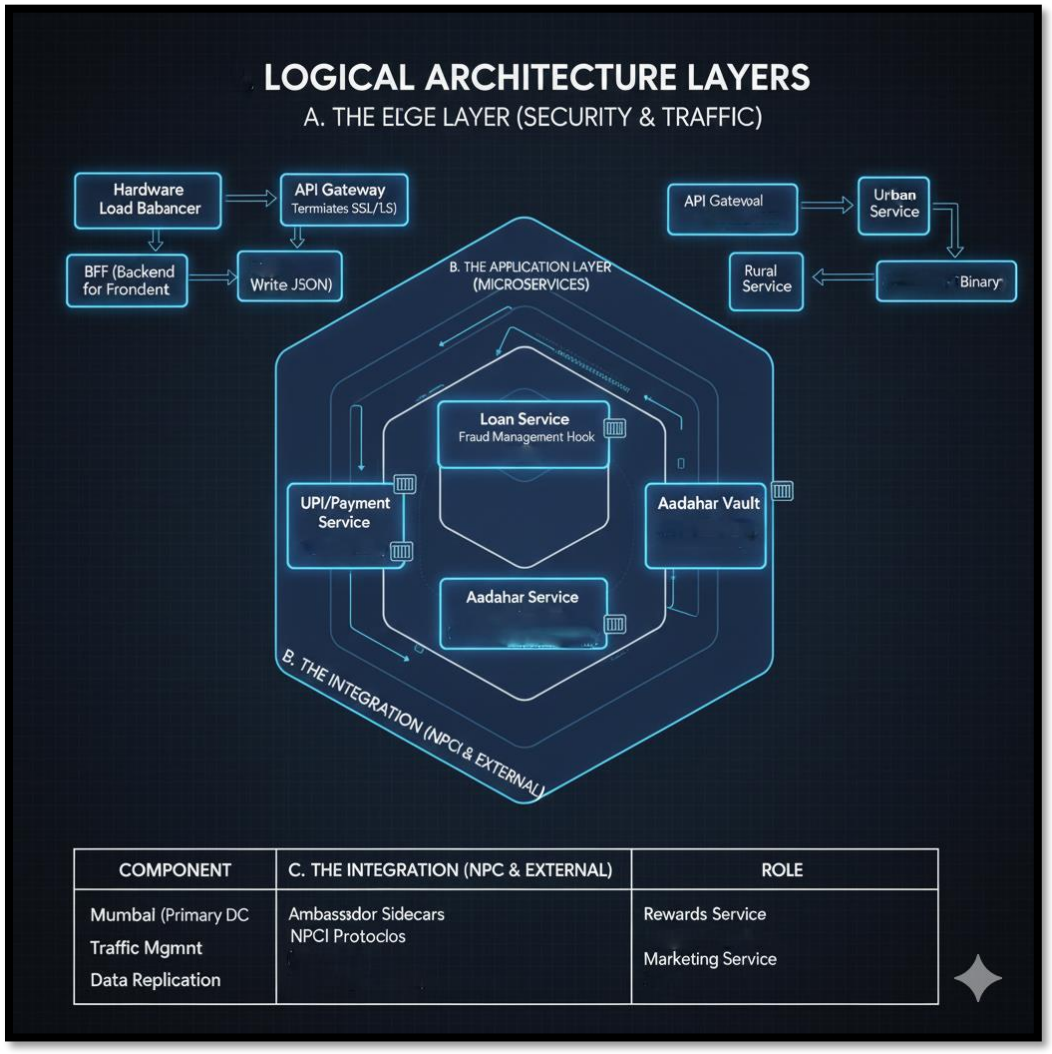
b) The Application Layer (Microservices)

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**	**Synchronous Mirroring:** Every "Write" must be acknowledged by the Hyderabad DR before it is finalized in Mumbai.
Urban Dashboard	**In-Memory Cache (Redis)**	**Async Replication:** Fast reads for the urban dashboard.
Rural Sync	**Outbox / Persistent Queue**	**Exactly-once Delivery:** 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
Atomic Committer	Ensures data integrity on the tablet.	BEGIN TRANSACTION; SAVE TransactionData; SAVE OutboxEvent(UUID, Payload); COMMIT;	**Local SQLite:** txn_table + outbox_table
Connectivity Monitor	Detects 2G/3G/Wi-Fi signal.	If (SignalStrength > -100dBm) {TriggerSync();}	**Native OS API:** Android Connectivity Manager
Sync Worker	Pushes data to DC.	Reads outbox_table by created_at ASC; Sends via gRPC; Deletes on HTTP 200.	**Buffer:** Queue-based (FIFO)
Idempotency Guard	Prevents double spending at DC.	If (Redis.Exists(EventUUID)) {Return Success;} Else {ProcessAndCache(EventUUID);}	**Cache:** 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
1. Ingress	Encryption	Encrypt raw Aadhaar using the Public Key of the Vault.	Encrypted_Blob
2. Verification	Checksum Logic	Validate Aadhaar.	Boolean (True/False)

3. Storage	HSM Vaulting	Store raw value in Hardware Security Module.	Token_ID
4. Detokenization	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
Write Event	Command Side	Debit (AccountID, 500) -> Commit to SQL.	PostgreSQL (ACID)
Event Emit	Message Bus	Publish BalanceChanged event to Broker.	Kafka Topic
Projection	Consumer	Listen to Broker; Fetch current balance; Compute total.	Java/Go Consumer
View Update	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
Money Debited	SMS/Notification Fails	**Continue:** Not a financial failure; retry SMS.	**Success**
Money Debited	External Switch Fails	**Reverse:** Credit (AccountID, Amount) with tag SAGA_REVERSAL.	**Rollback**
Loan Approved	Disbursement Fails	**Reverse:** Move Loan Status back to Rejected/Pending.	**Rollback**

v. LLD 5: Component Interface Definitions (APIs)

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

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

9. Component & Service Selection

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

10.ADRs (Architectural Decision Records)

Feature	PostgreSQL	MongoDB	Redis
Decision	Use as the Primary System of Record.	Use for KYC and User Metadata.	Use as a Cache and Session Store.
Rationale	Strict **ACID compliance** is mandatory for banking licenses. Handles complex SQL joins for audits.	**Flexible Schema** allows adding new KYC fields (e.g., Video-KYC links) without downtime.	**Sub-millisecond latency** handles the "Urban Speed" requirement of 50k TPS.
Sync Logic	**Synchronous Mirroring** to DR site.	**Asynchronous Replication** (Active-Active).	**Local Persistence** (AOF) + In-memory replication.
Conflict Handling	Strong Consistency (No conflicts allowed).	Last-Write-Wins (Eventual Consistency).	N/A (Transient data).
Constraint	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 **Urban (JSON)** and **Rural (gRPC)** endpoints.	**Urban Speed:** <100ms Latency.
Security	**Zero-Trust Sidecar**	Envoy sidecars handling **mTLS** and **HSM-backed Tokenization**.	**Regulatory Trust:** Aadhaar Vault Compliance.
Application Style	**Clean Microservices**	Polyglot (Java for Ledger, Go for Payments, Python for AI).	**180-Day Launch:** Parallel development teams.
Data Persistence	**Polyglot Trio**	**Postgres** (Ledger), **Mongo** (KYC), **Redis** (Cache).	**Scale:** 50,000 TPS Urban Peak.
Rural Resilience	**Offline-First Sync**	Transactional Outbox + SQLite Store-and-Forward.	**Rural Inclusion:** Zero data loss on 2G.
Reliability	**DC-DR Mirroring**	Mumbai (Active) to Hyderabad (Hot Standby) via Dark Fiber.	**99.99% Uptime:** 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:

1. **Request:** A user in a rural village initiates a ₹1,000 loan repayment on a 2G network.
2. **Edge (Rural):** The request uses **gRPC (Binary)** to minimize data size.
3. **BFF Layer:** The Rural BFF validates the device signature and hands it to the **Sync Service**.
4. **Security Layer:** The **Aadhaar Sidecar** verifies the user's tokenized identity via the **HSM**.
5. **Core Logic:** The **Ledger Microservice (Java)** performs a double-entry update using **Clean Architecture** rules.
6. **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
50,000 TPS	**CQRS + Redis + Hyper-Converged Infrastructure**	Hardware-accelerated read/writes handle urban peaks.
Offline Rural Play	**SQLite Outbox + gRPC Binary Compression**	Reliable banking in 2G "Dark Zones."
180-Day Launch	**Strangler Fig + Modular Microservices**	Decoupled services allow for a fast-track MVP launch.
Cloud Agnostic	**Infrastructure Abstraction (Hexagonal Layering)**	Core logic is 100% portable to Cloud in <30 days.