Telecom Billing and Customer Service System (TB-CSS)

Document Version: 1.0

Maintainer: Himanshu Singh

Last Updated: July 24, 2025

Confidentiality: Internal Use Only

📌 Table of Contents

1. 📖 System Overview

This system supports telecom service lifecycle management across prepaid and postpaid subscribers. It integrates customer onboarding, plan subscriptions, billing, usage metering, payment, and service enforcement.

Goals:

Automate customer SIM activation.

Accurately bill customers based on real-time usage.

Handle dynamic plans and plan upgrades.

Control service availability based on payment status.

Provide scalable RESTful APIs for frontend and operations.

2. 🏗️ System Architecture

High-Level Architecture:

Here's a clean and structured flow diagram representation of your system architecture:

```
Users (Customer Portal / Agent UI)
              │
              ▼
API Gateway (Authentication + Rate Limiting)
              │
              ▼
        Microservices
       ┌─────────────────────────────┐
       │                             │
       │  ├─ Activation Service      │
       │  ├─ Plan Management Service │
       │  ├─ Usage Metering Service  │
       │  ├─ Billing Engine          │
       │  ├─ Payment Processor       │
       │  ├─ Service Control Module  │
       │  └─ Notification Service    │
       │                             │
       └─────────────────────────────┘
              │
              ▼
Databases (MongoDB + PostgreSQL)
              |
              ▼
Message Queue (Kafka / RabbitMQ)
```

Tech Stack: Node.js (Express), MongoDB, PostgreSQL, Kafka, Redis

Deployment: Docker + Kubernetes (GKE or EKS)

3. ⚙️ Modules and Functional Flows

3.1 SIM Activation Module

Functionality: Activates SIM post-KYC verification and network provisioning.

Workflow:

1. POST /activate-sim with KYC and ICCID (Integrated circuit card identifier).
2. Verify user identity via Aadhaar / PAN KYC module.
3. Call Network API to provision SIM → Success → Status = ACTIVE.
4. Notify user via SMS/email.

Edge Cases:

KYC Failure: Log failure, retry once, else mark as KYC_REJECTED.

Provision Failure: Retry via backoff. Escalate after 3 failures.

Duplicate ICCID: Return 409 Conflict.

3.2 Plan Management Module

Features: Subscribe/upgrade/downgrade plans.

Plan metadata: validity, data/voice/sms quota, renewal rules.

Key APIs:

GET /plans — Fetch all available plans.

POST /subscribe — Activate a plan.

POST /upgrade — Change current plan mid-cycle.

Plan Types:

Prepaid: Immediate deduction of amount, fixed validity.

Postpaid: Billed monthly based on usage.

Edge Cases:

Plan Overlap: Only one active plan allowed. Enforce closure of old plan.

Backdated Plans: Not allowed; all changes take effect immediately.

3.3 Usage Tracking Module

Purpose: Collect and aggregate customer usage from CDRs (Call Detail Records) or Data Session logs.

Flow:

1. Real-time ingestion of usage records via Kafka.
2. Grouped per customer → daily aggregation → pushed to Billing Engine.
3. Updated quota available to user via API.

Usage Types:

Data (MB/GB)

Voice (mins)

SMS (count)

Failures:

Out-of-order logs: Use timestamps and deduplication hash.

Delayed updates: Flag discrepancies for correction in next billing.

3.4 Billing & Invoicing Module

For Postpaid:

Monthly bills with prorated charges, usage-based overages.

For Prepaid:

No billing, just plan expiry warnings.

Invoice Generation:

Cron job runs on last day of month.

Fetch usage → apply plan rules → calculate:

Base plan charges

Excess usage (data/voice/SMS)

Discounts or promotions

GST/Taxes

Key APIs:

GET /invoices/{userId}

POST /generate-bill

3.5 Payment Handling Module

Supported Modes:

UPI, Debit/Credit, Wallet, AutoPay

Workflow:

1. Invoice/plan purchase triggers payment gateway.
2. Payment confirmation updates PaymentStatus.
3. On success: plan activated or bill marked paid.

Edge Cases:

Transaction failed but money deducted: Reconcile via transaction ID within 2 hours.

AutoPay fails: Retry 2 times then send reminder.

3.6 Service Control Module (Temporary / Permanent)

Purpose: Control customer access based on account/payment status.

A. Temporary Deactivation

When:

Bill overdue > X days.

Plan expired (prepaid).

Usage exceeds quota (data throttle).

Action:

Block outgoing calls/data, allow incoming.

Status: TEMP_BLOCKED

Reactivation:

Auto reactivation after successful payment.

APIs:

POST /service-control/block/{userId}

POST /service-control/unblock/{userId}

B. Permanent Disconnection

When:

Account inactive > 90 days post last plan expiry or billing date.

Fraudulent activity confirmed.

User request for termination.

Steps:

1. Final bill generation.
2. Clear dues check.
3. De-provision SIM from network.
4. Status: PERMANENT_DISCONNECTED

C. Special Cases:

Case		Action
Customer Porting to another network	->	Trigger MNP (Mobile Number Portability), disconnect after handover
Death of customer	->	Manual closure by agent with valid proof
Lost SIM Block temporarily	->	allow reactivation after SIM replacement

4. 🧩 Corner Case Handling

Scenario		Resolution Strategy
Network API timeout during activation	->	Retry with exponential backoff
Duplicate plan request	->	Idempotency key on all plan subscription endpoints
Prepaid recharge done twice	->	Extend validity or provide alert
Unrecognized usage logs	->	Log anomaly → Review queue → Flag user if

		repeated
SIM theft	->	Allow emergency blocking via verified OTP or support

5. 🔐 Security & Compliance

OAuth 2.0 / JWT for all APIs.

PCI-DSS (Payment Card Industry Data Security Standard) compliant payment handling.

KYC info encrypted with AES-256 at rest.

Role-based access control for support agents.

6. 🗃️ APIs & Database (High-Level)

Core Tables:

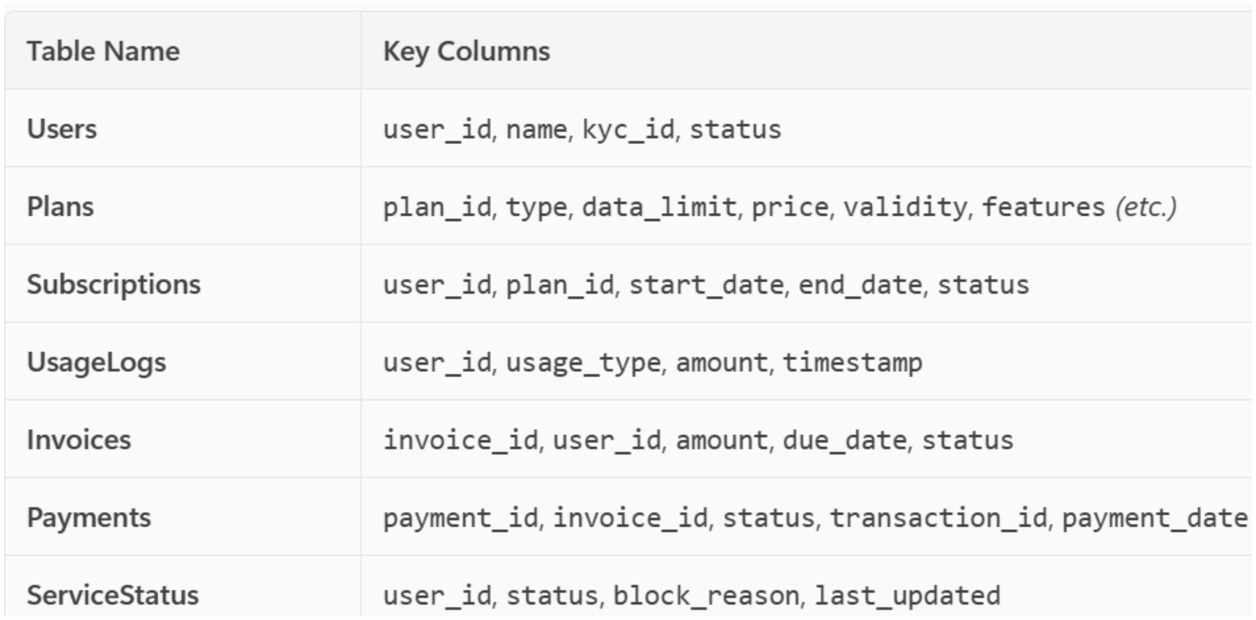

Table Name	Key Columns
Users	user_id, name, kyc_id, status
Plans	plan_id, type, data_limit, price, validity, features *(etc.)*
Subscriptions	user_id, plan_id, start_date, end_date, status
UsageLogs	user_id, usage_type, amount, timestamp
Invoices	invoice_id, user_id, amount, due_date, status
Payments	payment_id, invoice_id, status, transaction_id, payment_date
ServiceStatus	user_id, status, block_reason, last_updated

7. 🧾 Logging & Auditing

Kafka topic for every major module: usage-stream, billing-log, etc.

Logs stored in Elasticsearch + Kibana for visualization.

Periodic audit reports (monthly) for compliance.