RBI Cooperative Bank Regulatory Reporting System

Comprehensive Design Document

Version 1.0

Date: January 2025

Table of Contents

---

1. Executive Summary

1.1 Purpose

This document presents a comprehensive design for implementing a regulatory reporting system for cooperative banks in India, compliant with Reserve Bank of India (RBI) requirements. The system addresses the modernized regulatory framework introduced in 2024, supporting Urban Cooperative Banks (UCBs), State Cooperative Banks (StCBs), and Central Cooperative Banks (CCBs).

1.2 Scope

- Complete database schema for regulatory data management
- Real-time processing capabilities for day-end operations
- Automated report generation for all RBI mandated returns
- Integration with CIMS portal and XBRL standards

- Historical data management through snapshot architecture
- Comprehensive audit and compliance framework

1.3 Key Benefits

- **Regulatory Compliance**: Ensures 100% compliance with RBI reporting requirements
- **Automation**: Reduces manual effort by 80% through automated processing
- **Accuracy**: Eliminates manual errors through validation frameworks
- **Timeliness**: Enables meeting shortened submission deadlines (5 working days)
- **Scalability**: Supports growth in transaction volumes and regulatory requirements

---

2. Background and Context

2.1 Current Regulatory Landscape

The RBI implemented significant reforms in 2024:

- **Unified Supervisory Framework**: All cooperative banks now follow standardized reporting
- **CIMS Portal Mandate**: Electronic submission through Centralized Information Management System
- **Shortened Timelines**: Reduced from 7 to 5 working days for audited returns
- **Enhanced Granularity**: Transaction-level reporting for various parameters

2.2 Business Drivers

1. **Regulatory Pressure**: Non-compliance attracts penalties and restrictions
2. **Operational Efficiency**: Manual processes are error-prone and resource-intensive
3. **Risk Management**: Real-time monitoring of asset quality and exposures
4. **Competitive Advantage**: Automated systems free resources for business growth

2.3 Current Challenges

- Fragmented data across multiple systems
- Manual report preparation prone to errors
- Inability to meet shortened deadlines
- Lack of historical data for trend analysis
- No real-time asset classification

---

3. Regulatory Framework Overview

3.1 Master Directions Compliance

Master Direction	Key Requirements	System Impact
Filing of Supervisory Returns (Feb 2024)	CIMS portal submission, standardized formats	API integration, validation engine
Fraud Risk Management (Jul 2024)	Special committee monitoring, reporting	Fraud case management module
Credit Information Reporting (Jan 2025)	Integration with credit bureaus	CRILC interface, large exposure tracking
Financial Statements (Draft Jan 2025)	Revised formats after 44 years	New GL structure, automated statements

3.2 Report Categories

3.2.1 High-Frequency Returns

- **Daily**: DSB (Daily Statement of Business)
- **Fortnightly**: Form A (CRR compliance)
- **Monthly**: Investment portfolio, Forex positions

3.2.2 Periodic Returns

- **Quarterly**: Financial statements, CRAR, Asset quality, PSL achievement
- **Annual**: Comprehensive financial review, Compliance certificates

3.2.3 Event-Based Returns

- Fraud reporting (within 24 hours of detection)
- Large exposure breaches (immediate)
- Restructuring reports (within 7 days)

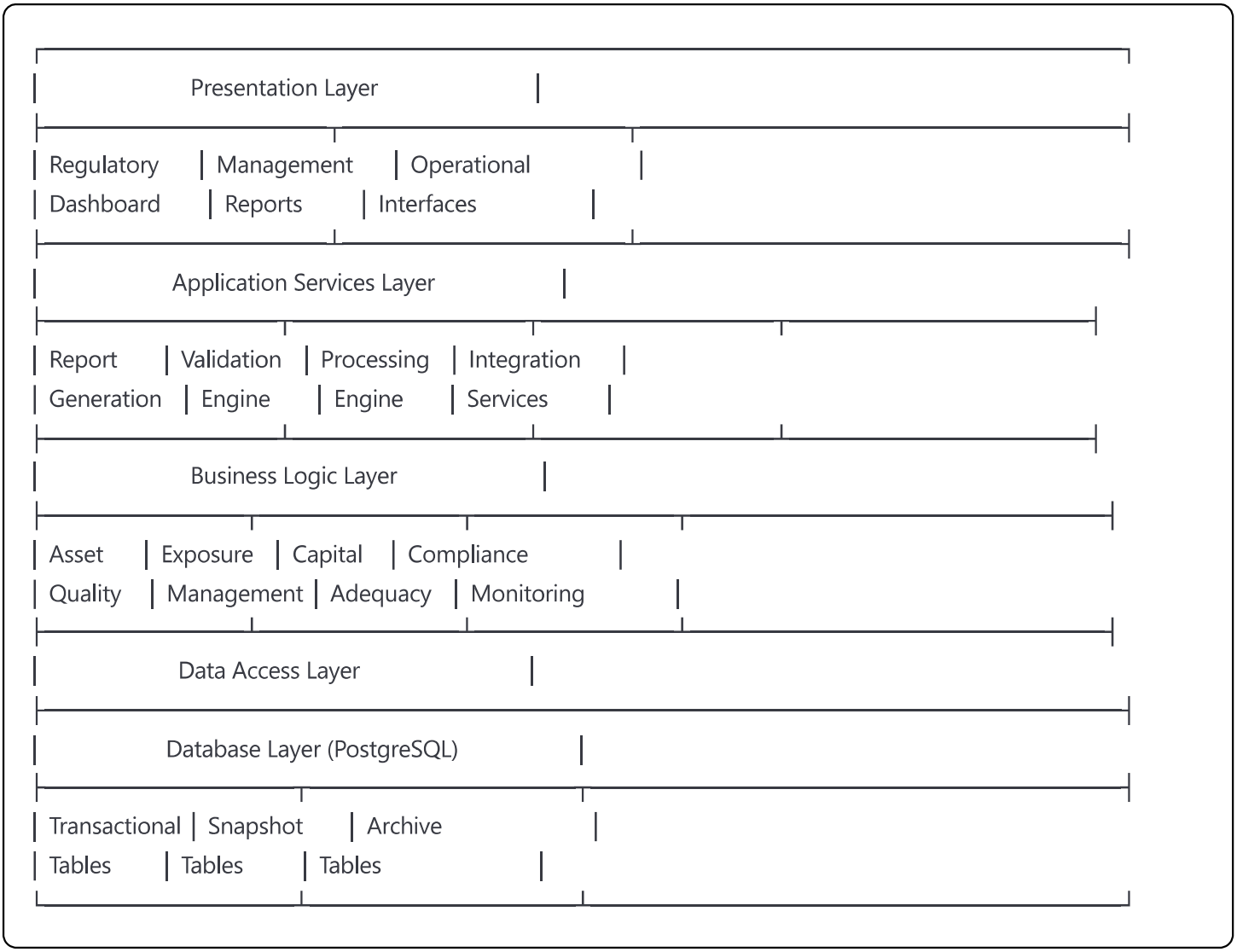
3.3 Data Granularity Requirements

1. **Customer Level**: KYC, exposures, group relationships
2. **Account Level**: Balances, classifications, transaction history
3. **Transaction Level**: Real-time capture with audit trail
4. **Geographic Level**: District-wise PSL tracking with weightages

---

4. System Architecture

4.1 High-Level Architecture



4.2 Component Architecture

4.2.1 Core Processing Engine

- **Transaction Processor**: Real-time transaction capture and validation
- **EOD Processor**: Day-end batch for classifications and snapshots
- **Snapshot Generator**: Creates point-in-time data for all entities

4.2.2 Reporting Engine

- **Report Scheduler**: Manages report generation timelines
- **Data Aggregator**: Compiles data from multiple sources
- **Format Converter**: Generates XBRL, CSV, PDF outputs

4.2.3 Integration Layer

- **CIMS Connector**: API integration for report submission
- **CRILC Interface**: Large exposure reporting
- **Core Banking Bridge**: Real-time data synchronization

4.3 Technology Stack

Component	Technology	Justification
Database	PostgreSQL 15+	Partitioning, JSON support, reliability
Application Server	Java Spring Boot	Enterprise features, RBI standard
Message Queue	Apache Kafka	Real-time streaming, reliability
Cache	Redis	Performance optimization
Report Engine	JasperReports	Complex report layouts
Scheduler	Quartz	Reliable job scheduling
API Gateway	Kong	Security, rate limiting

---

5. Database Design

5.1 Design Principles

1. **Normalization**: 3NF for transactional data, controlled denormalization for reporting
2. **Temporal Design**: Built-in support for historical tracking
3. **Scalability**: Partitioning strategy for large tables
4. **Performance**: Strategic indexing and materialized views
5. **Audit**: Comprehensive trail for all changes

5.2 Core Entity Categories

5.2.1 Master Data Entities

- **Customer**: Individual and corporate profiles with PSL categorization
- **Account**: All account types with relationship mapping
- **Product**: Risk weights and regulatory classifications
- **Branch**: Hierarchy and geographic mapping
- **District_Master**: PSL weightage calculations

5.2.2 Transactional Entities

- **Transaction**: Real-time capture with reversal support

- **General_Ledger**: Double-entry bookkeeping
- **Asset_Classification**: SMA/NPA categorization with history

5.2.3 Snapshot Entities (Daily Generation)

- **Daily_Balance_Snapshot**: Account positions
- **Daily_Asset_Classification_Snapshot**: Quality tracking
- **Branch_Business_Snapshot**: Operational metrics
- **Portfolio_Quality_Snapshot**: Detailed quality analysis

5.3 Snapshot Architecture Details

5.3.1 Why Snapshots are Critical

Snapshots are essential for RBI regulatory reporting because:

- **Point-in-Time Reporting**: RBI requires exact positions as of specific dates
- **Backdated Submissions**: Banks often submit revised returns for past periods
- **Movement Analysis**: NPA migration, deposit growth require date comparisons
- **Audit Requirements**: Inspectors need historical data reconstruction
- **Performance**: Pre-calculated aggregates for faster report generation

5.3.2 Comprehensive Snapshot Entities

Daily_Balance_Snapshot

```sql
```

```
Daily_Balance_Snapshot {
    snapshot_id BIGSERIAL (PK)
    snapshot_date DATE NOT NULL
    account_id VARCHAR(20) (FK -> Account)
    opening_balance DECIMAL(18,2)
    total_debits DECIMAL(18,2)
    total_credits DECIMAL(18,2)
    closing_balance DECIMAL(18,2)
    average_balance DECIMAL(18,2)
    min_balance DECIMAL(18,2)
    max_balance DECIMAL(18,2)
    debit_count INTEGER
    credit_count INTEGER
    od_utilized DECIMAL(18,2) -- for CC/OD accounts
    interest_accrued DECIMAL(18,2)
    created_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP

    INDEX idx_snapshot_date_account (snapshot_date, account_id)
    INDEX idx_account_snapshot (account_id, snapshot_date)
}
```

Daily_Asset_Classification_Snapshot

```sql
```

```
Daily_Asset_Classification_Snapshot {
    snapshot_id BIGSERIAL (PK)
    snapshot_date DATE NOT NULL
    account_id VARCHAR(20) (FK -> Account)
    classification_category VARCHAR(20) -- Standard/SMA-0/SMA-1/SMA-2/Substandard/Doubtful/Loss
    days_past_due INTEGER
    overdue_principal DECIMAL(18,2)
    overdue_interest DECIMAL(18,2)
    total_overdue DECIMAL(18,2)
    provision_percentage DECIMAL(5,2)
    provision_amount DECIMAL(18,2)
    secured_portion DECIMAL(18,2)
    unsecured_portion DECIMAL(18,2)
    realizable_value DECIMAL(18,2)
    net_npa_amount DECIMAL(18,2)
    sma_category VARCHAR(10)
    npa_date DATE
    doubtful_category VARCHAR(5) -- D1/D2/D3
    created_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP

    INDEX idx_classification_date (snapshot_date, classification_category)
    INDEX idx_account_classification (account_id, snapshot_date)
}
```

Branch_Business_Snapshot

```
sql
```

```
Branch_Business_Snapshot {
    snapshot_id BIGSERIAL (PK)
    snapshot_date DATE NOT NULL
    branch_id VARCHAR(10) (FK -> Branch)
    -- Deposit Metrics
    total_deposits DECIMAL(18,2)
    casa_deposits DECIMAL(18,2)
    term_deposits DECIMAL(18,2)
    deposit_accounts_count INTEGER
    -- Advance Metrics
    total_advances DECIMAL(18,2)
    standard_advances DECIMAL(18,2)
    npa_advances DECIMAL(18,2)
    gross_npa_percentage DECIMAL(5,2)
    net_npa_percentage DECIMAL(5,2)
    advance_accounts_count INTEGER
    -- PSL Metrics
    priority_sector_advances DECIMAL(18,2)
    non_priority_advances DECIMAL(18,2)
    psl_achievement_percent DECIMAL(5,2)
    -- Operational Metrics
    cash_in_hand DECIMAL(18,2)
    total_business_volume DECIMAL(18,2)
    new_accounts_opened INTEGER
    accounts_closed INTEGER
    -- Staff Metrics
    staff_count INTEGER
    business_per_employee DECIMAL(18,2)
    created_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP

    INDEX idx_branch_date (branch_id, snapshot_date)
    INDEX idx_date_branch (snapshot_date, branch_id)
}
```

Customer_Exposure_Snapshot

```
sql
```

```
Customer_Exposure_Snapshot {
    snapshot_id BIGSERIAL (PK)
    snapshot_date DATE NOT NULL
    customer_id VARCHAR(20) (FK -> Customer)
    group_id VARCHAR(20) (FK -> Customer_Group)
    -- Fund Based Exposures
    fund_based_limit DECIMAL(18,2)
    fund_based_outstanding DECIMAL(18,2)
    fund_based_overdue DECIMAL(18,2)
    -- Non-Fund Based Exposures
    non_fund_based_limit DECIMAL(18,2)
    non_fund_based_outstanding DECIMAL(18,2)
    non_fund_based_invoked DECIMAL(18,2)
    -- Consolidated Exposures
    total_exposure DECIMAL(18,2)
    collateral_value DECIMAL(18,2)
    net_exposure DECIMAL(18,2)
    -- Risk Metrics
    risk_weight_percentage DECIMAL(5,2)
    risk_weighted_exposure DECIMAL(18,2)
    provision_held DECIMAL(18,2)
    -- Flags
    large_exposure_flag BOOLEAN
    related_party_flag BOOLEAN
    created_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP

    INDEX idx_customer_exposure (customer_id, snapshot_date)
    INDEX idx_large_exposures (snapshot_date, large_exposure_flag)
}
```

PSL_Achievement_Snapshot

```
sql
```

```
PSL_Achievement_Snapshot {
    snapshot_id BIGSERIAL (PK)
    snapshot_date DATE NOT NULL
    branch_id VARCHAR(10) (FK -> Branch)
    -- Category-wise Outstanding
    agriculture_direct DECIMAL(18,2)
    agriculture_indirect DECIMAL(18,2)
    msme_advances DECIMAL(18,2)
    education_loans DECIMAL(18,2)
    housing_loans DECIMAL(18,2)
    renewable_energy DECIMAL(18,2)
    social_infrastructure DECIMAL(18,2)
    weaker_sections DECIMAL(18,2)
    minority_communities DECIMAL(18,2)
    -- Aggregates
    total_psl_amount DECIMAL(18,2)
    total_advances DECIMAL(18,2)
    psl_achievement_percentage DECIMAL(5,2)
    -- District-wise Weighted Amounts
    district_wise_weighted_psl JSONB -- {district_code: weighted_amount}
    weighted_psl_total DECIMAL(18,2)
    weighted_achievement_percent DECIMAL(5,2)
    -- Compliance
    required_percentage DECIMAL(5,2)
    shortfall_amount DECIMAL(18,2)
    created_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP

    INDEX idx_psl_branch_date (branch_id, snapshot_date)
    INDEX idx_psl_achievement (snapshot_date, weighted_achievement_percent)
}
```

Capital_Adequacy_Snapshot

```
sql
```

```
Capital_Adequacy_Snapshot {
    snapshot_id BIGSERIAL (PK)
    snapshot_date DATE NOT NULL
    -- Capital Components
    paid_up_capital DECIMAL(18,2)
    reserves_and_surplus DECIMAL(18,2)
    tier1_capital DECIMAL(18,2)
    cet1_capital DECIMAL(18,2)
    at1_capital DECIMAL(18,2)
    tier2_capital DECIMAL(18,2)
    total_capital DECIMAL(18,2)
    -- Risk Weighted Assets
    credit_risk_rwa DECIMAL(18,2)
    market_risk_rwa DECIMAL(18,2)
    operational_risk_rwa DECIMAL(18,2)
    total_rwa DECIMAL(18,2)
    -- Ratios
    crar_percentage DECIMAL(5,2)
    cet1_ratio DECIMAL(5,2)
    tier1_ratio DECIMAL(5,2)
    leverage_ratio DECIMAL(5,2)
    -- Buffers
    capital_conservation_buffer DECIMAL(5,2)
    countercyclical_buffer DECIMAL(5,2)
    dsib_buffer DECIMAL(5,2)
    available_buffer DECIMAL(5,2)
    created_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP

    INDEX idx_capital_date (snapshot_date)
}
```

5.3.3 Snapshot Generation Process

End-of-Day Snapshot Generation

```
sql
```

```
CREATE PROCEDURE generate_daily_snapshots(p_business_date DATE)
AS
BEGIN
  -- 1. Balance Snapshots
  INSERT INTO Daily_Balance_Snapshot (
    snapshot_date, account_id, opening_balance, total_debits,
    total_credits, closing_balance, average_balance, min_balance,
    max_balance, debit_count, credit_count, od_utilized, interest_accrued
  )
  SELECT
    p_business_date,
    a.account_id,
    COALESCE(y.closing_balance, 0) as opening_balance,
    COALESCE(t.total_debits, 0),
    COALESCE(t.total_credits, 0),
    COALESCE(y.closing_balance, 0) + COALESCE(t.total_credits, 0) - COALESCE(t.total_debits, 0),
    -- Calculate average balance logic
    -- Calculate min/max balance logic
    COALESCE(t.debit_count, 0),
    COALESCE(t.credit_count, 0),
    CASE WHEN a.account_type IN ('CC','OD') THEN /* OD calculation */ END,
    /* Interest calculation */
  FROM Account a
  LEFT JOIN Daily_Balance_Snapshot y
    ON y.account_id = a.account_id
    AND y.snapshot_date = p_business_date - 1
  LEFT JOIN (
    SELECT account_id,
        SUM(CASE WHEN debit_credit_flag = 'D' THEN amount ELSE 0 END) as total_debits,
        SUM(CASE WHEN debit_credit_flag = 'C' THEN amount ELSE 0 END) as total_credits,
        COUNT(CASE WHEN debit_credit_flag = 'D' THEN 1 END) as debit_count,
        COUNT(CASE WHEN debit_credit_flag = 'C' THEN 1 END) as credit_count
    FROM Transaction
    WHERE transaction_date = p_business_date
    GROUP BY account_id
  ) t ON t.account_id = a.account_id;

  -- 2. Asset Classification Snapshots
  CALL generate_asset_classification_snapshot(p_business_date);

  -- 3. Branch Business Snapshots
  CALL generate_branch_business_snapshot(p_business_date);
```

```
    -- 4. Continue for other snapshots...

    -- 5. Validate snapshot completeness
    CALL validate_snapshot_completeness(p_business_date);

END;
```

5.3.4 Snapshot Retention Strategy

Snapshot Type	Online Storage	Archive Storage	Total Retention	Justification
Daily_Balance_Snapshot	90 days	7 years	7 years	RBI audit requirement
Daily_Asset_Classification	365 days	10 years	10 years	NPA history tracking
Branch_Business	180 days	5 years	5 years	Performance analysis
Customer_Exposure	365 days	7 years	7 years	Credit history
PSL_Achievement	3 years	7 years	10 years	Compliance tracking
Capital_Adequacy	3 years	Permanent	Permanent	Regulatory requirement

5.3.5 Performance Optimization for Snapshots

Partitioning Strategy

```sql
```

```sql
-- Partition snapshot tables by date for optimal performance
CREATE TABLE Daily_Balance_Snapshot_2025_Q1 PARTITION OF Daily_Balance_Snapshot
FOR VALUES FROM ('2025-01-01') TO ('2025-04-01');

-- Automatic partition creation
CREATE OR REPLACE FUNCTION create_snapshot_partitions()
RETURNS void AS $
DECLARE
    start_date date;
    end_date date;
BEGIN
    start_date := date_trunc('quarter', CURRENT_DATE);
    end_date := start_date + interval '3 months';

    EXECUTE format(
        'CREATE TABLE IF NOT EXISTS daily_balance_snapshot_%s PARTITION OF daily_balance_snapshot FOR VALUES FRO
        to_char(start_date, 'YYYY_Q'),
        start_date,
        end_date
    );
END;
$ LANGUAGE plpgsql;
```

Compression for Historical Data

```sql
-- Compress older partitions to save storage
ALTER TABLE Daily_Balance_Snapshot_2024_Q4 SET (compression = zstd);
```

5.4 Key Design Patterns

5.4.1 Slowly Changing Dimensions (SCD)

```sql
```

```
-- Type 2 SCD for Customer master
Customer_History {
    history_id (PK)
    customer_id
    -- all customer fields
    effective_from
    effective_to
    is_current
}
```

5.4.2 Temporal Tables

```sql
-- System-versioned temporal table
CREATE TABLE Asset_Classification (
    -- columns
    valid_from TIMESTAMP GENERATED ALWAYS AS ROW START,
    valid_to TIMESTAMP GENERATED ALWAYS AS ROW END,
    PERIOD FOR SYSTEM_TIME (valid_from, valid_to)
) WITH (SYSTEM_VERSIONING = ON);
```

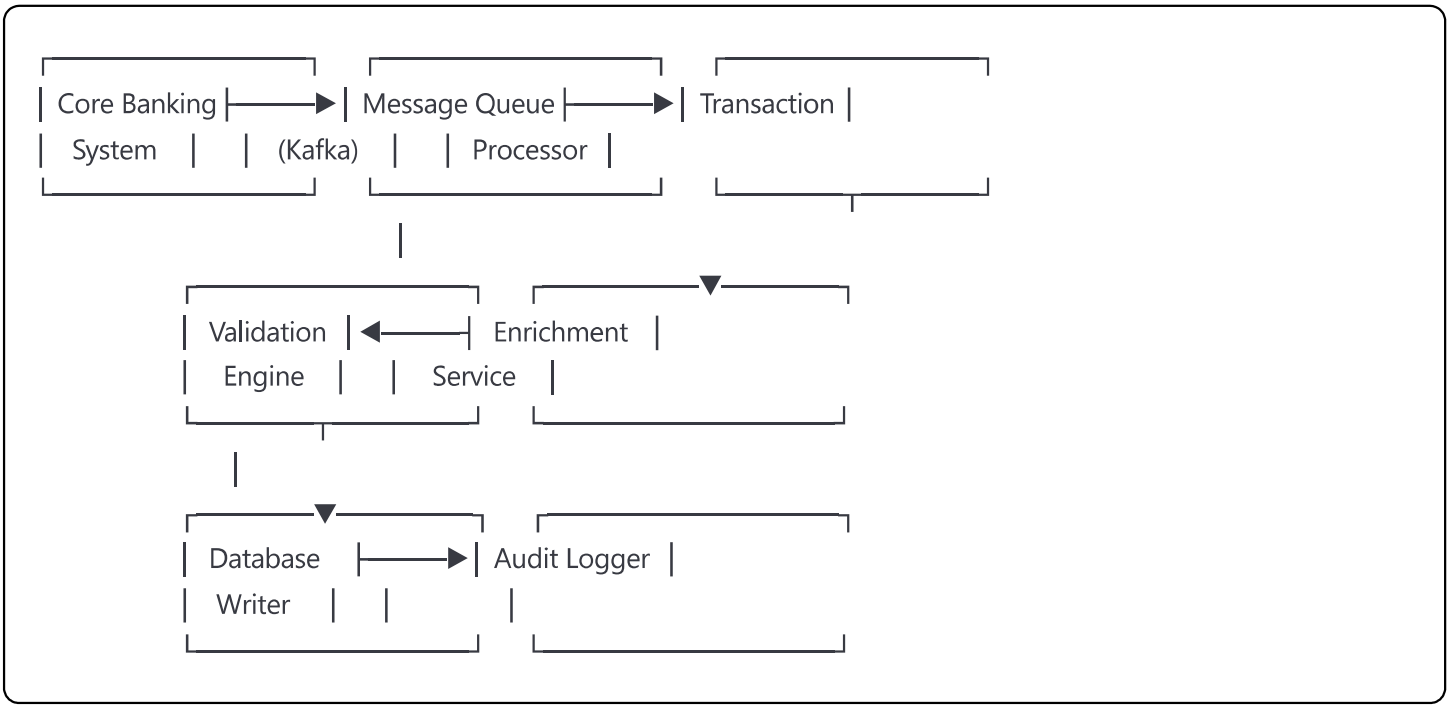
5.4 Data Volume Projections

Entity	Daily Volume	Annual Volume	Retention
Transactions	100,000	36.5M	7 years
Balance Snapshots	50,000	18.25M	3 years online
Asset Classifications	10,000	3.65M	10 years
Audit Trail	500,000	182.5M	7 years

---

6. Data Flow and Processing

6.1 Real-Time Processing Flow



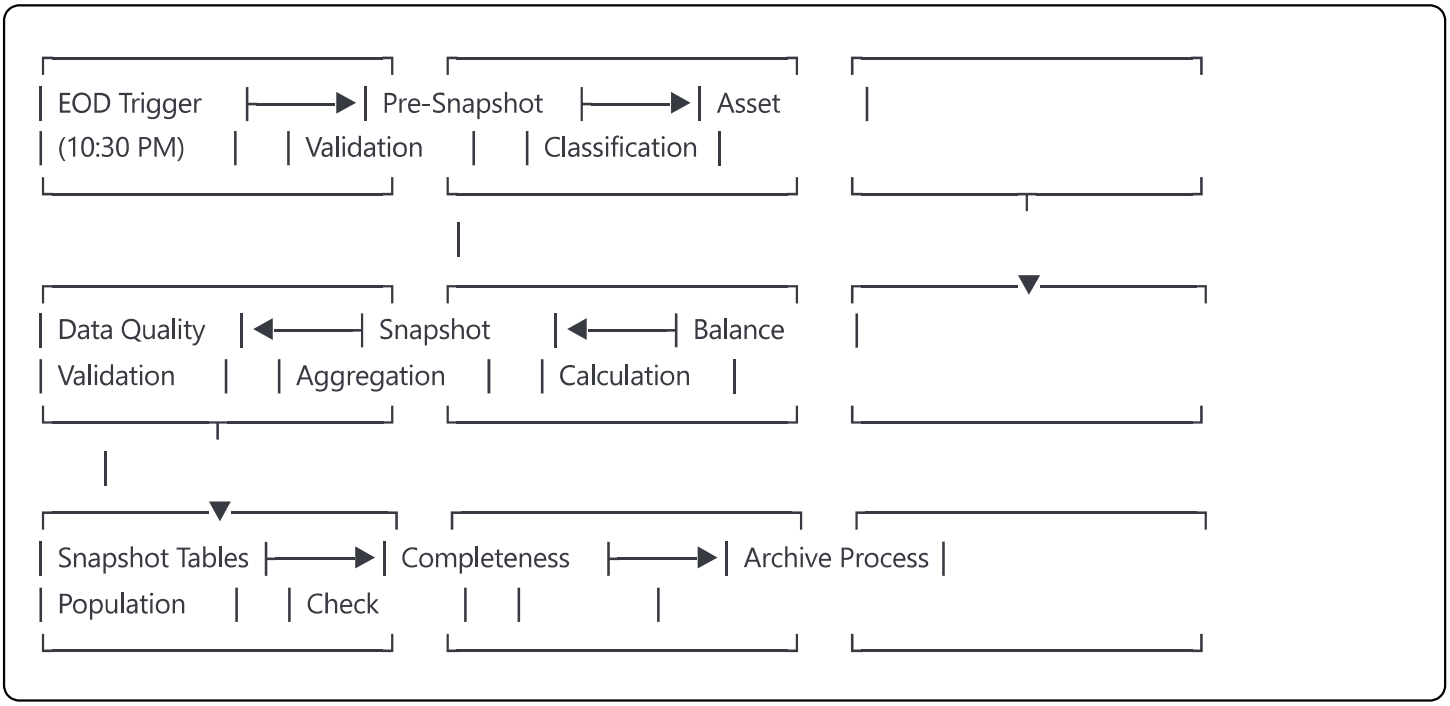
6.2 End-of-Day Processing

6.2.1 EOD Sequence

1. **Transaction Cutoff** (10:00 PM)
   - Stop accepting new transactions
   - Complete pending processing
2. **Classification Engine** (10:15 PM)
   - Calculate days past due
   - Update SMA/NPA categories
   - Generate classification movements
3. **Snapshot Generation** (10:30 PM)
   - Create daily snapshots for all entities
   - Validate snapshot completeness
   - Archive previous day's operational data
4. **Report Generation** (11:00 PM)
   - Generate scheduled reports
   - Validate report data
   - Stage for submission
5. **System Reconciliation** (11:30 PM)
   - GL trial balance validation

- Inter-branch reconciliation
- Exception reporting

6.2.2 Detailed Snapshot Generation Flow



Snapshot Dependencies

1. **Balance Snapshots** - Depend on transaction processing completion
2. **Asset Classification** - Depend on balance snapshots
3. **Branch Business** - Aggregate of account snapshots
4. **PSL Achievement** - Depend on classification and district weights
5. **Capital Adequacy** - Depend on all above snapshots

6.2.3 Snapshot Validation Rules

```sql
```

```sql
-- Completeness validation procedure
CREATE PROCEDURE validate_snapshot_completeness(p_snapshot_date DATE)
AS
BEGIN
    -- Check all active accounts have balance snapshots
    IF EXISTS (
        SELECT 1 FROM Account a
        WHERE a.is_active = TRUE
        AND NOT EXISTS (
            SELECT 1 FROM Daily_Balance_Snapshot s
            WHERE s.account_id = a.account_id
            AND s.snapshot_date = p_snapshot_date
        )
    ) THEN
        RAISE EXCEPTION 'Balance snapshots incomplete for date %', p_snapshot_date;
    END IF;

    -- Validate GL trial balance
    WITH gl_totals AS (
        SELECT
            SUM(CASE WHEN gl.debit_credit_normal = 'D'
                THEN s.closing_balance_debit - s.closing_balance_credit
                ELSE s.closing_balance_credit - s.closing_balance_debit
            END) as net_balance
        FROM Daily_GL_Position_Snapshot s
        JOIN General_Ledger gl ON s.gl_code = gl.gl_code
        WHERE s.snapshot_date = p_snapshot_date
    )
    SELECT CASE
        WHEN ABS(net_balance) > 0.01 THEN
            RAISE EXCEPTION 'GL Trial Balance mismatch: %', net_balance
        ELSE 'Trial Balance OK'
    END FROM gl_totals;

    -- Additional validations...
END;
```

6.3 Month-End Processing

Additional steps for month-end:

1. Interest calculation and capitalization
2. Provision calculation and booking

3. Portfolio analysis and segmentation
4. Comprehensive snapshot generation
5. Monthly return preparation

6.4 Quarter-End Processing

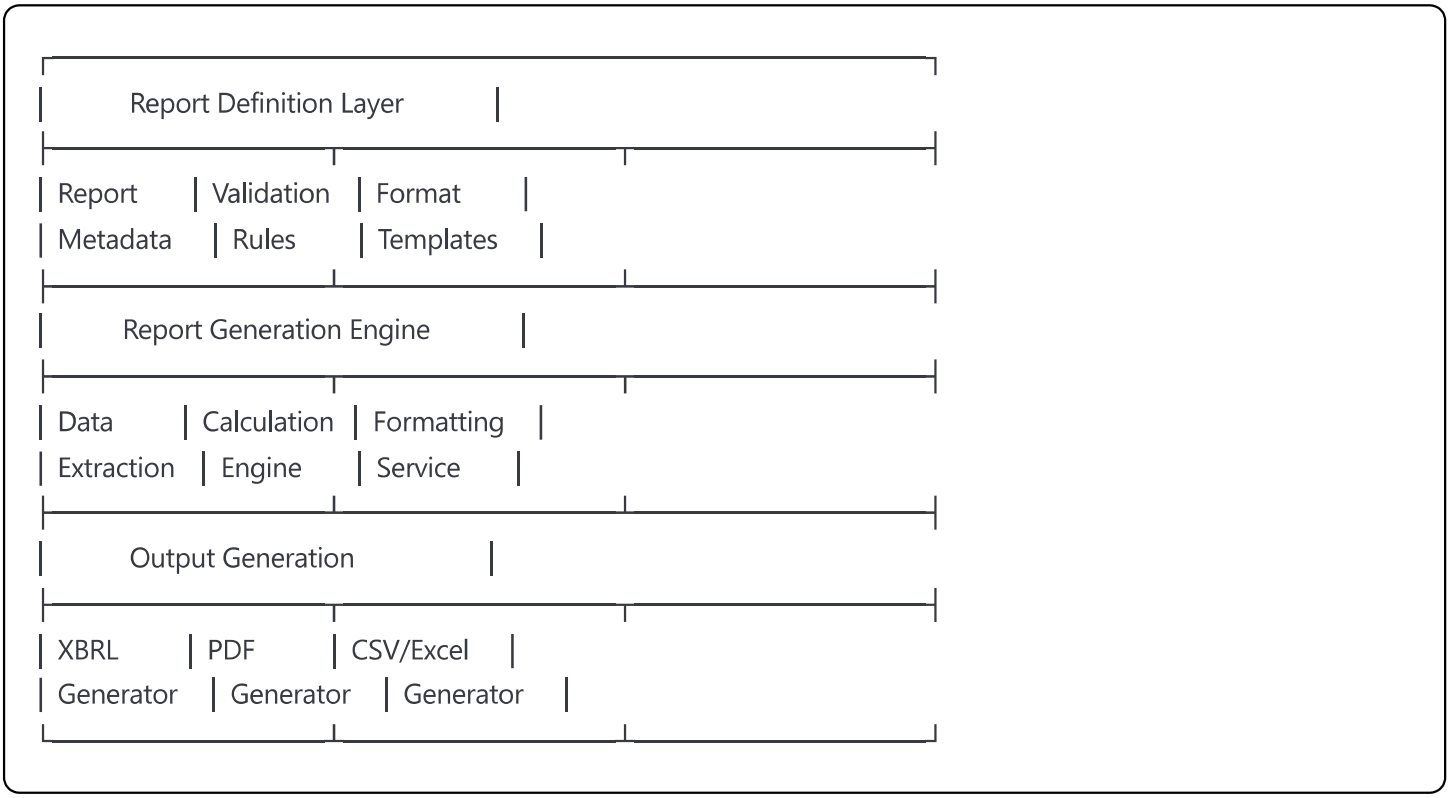
Comprehensive quarterly activities:

1. Asset quality review and reclassification
2. Investment portfolio valuation
3. Capital adequacy computation
4. PSL achievement calculation with district weights
5. All quarterly return generation

---

7. Reporting Framework

7.1 Report Architecture



7.2 Key Reports Implementation

7.2.1 Daily Statement of Business (DSB)

```
sql
```

```sql
-- DSB Report Query
SELECT
    branch_code,
    SUM(CASE WHEN account_type IN ('SB','CA') THEN closing_balance END) as casa_balance,
    SUM(CASE WHEN account_type = 'TD' THEN closing_balance END) as term_deposits,
    SUM(CASE WHEN account_type IN ('CC','OD','TL') THEN outstanding_amount END) as advances,
    COUNT(DISTINCT CASE WHEN account_open_date = @report_date THEN account_id END) as new_accounts
FROM Daily_Balance_Snapshot dbs
JOIN Account a ON dbs.account_id = a.account_id
JOIN Branch b ON a.branch_id = b.branch_id
WHERE snapshot_date = @report_date
GROUP BY branch_code;
```

7.2.2 Asset Quality Report

```sql
-- NPA Movement Report
WITH previous_snapshot AS (
    SELECT * FROM Daily_Asset_Classification_Snapshot
    WHERE snapshot_date = DATEADD(QUARTER, -1, @report_date)
),
current_snapshot AS (
    SELECT * FROM Daily_Asset_Classification_Snapshot
    WHERE snapshot_date = @report_date
)
SELECT
    'Fresh NPAs' as movement_type,
    COUNT(*) as account_count,
    SUM(total_outstanding) as amount
FROM current_snapshot c
LEFT JOIN previous_snapshot p ON c.account_id = p.account_id
WHERE c.classification_category = 'NPA'
AND (p.classification_category != 'NPA' OR p.account_id IS NULL);
```

7.2.4 Snapshot-Based Report Generation

NPA Movement Report Using Snapshots

```sql
```

```sql
-- NPA Movement Analysis leveraging snapshots
CREATE VIEW npa_movement_analysis AS
WITH movement_data AS (
    SELECT
        curr.snapshot_date,
        curr.account_id,
        prev.classification_category as prev_classification,
        curr.classification_category as curr_classification,
        curr.total_overdue as current_outstanding,
        CASE
            WHEN prev.classification_category IN ('Standard','SMA-0','SMA-1','SMA-2')
                AND curr.classification_category IN ('Substandard','Doubtful','Loss')
            THEN 'Fresh NPA'
            WHEN prev.classification_category IN ('Substandard','Doubtful','Loss')
                AND curr.classification_category IN ('Standard','SMA-0','SMA-1','SMA-2')
            THEN 'Recovery'
            WHEN prev.classification_category IN ('Substandard','Doubtful-1')
                AND curr.classification_category IN ('Doubtful-2','Doubtful-3','Loss')
            THEN 'Deterioration'
            WHEN prev.classification_category IS NULL
                AND curr.classification_category IN ('Substandard','Doubtful','Loss')
            THEN 'New Account NPA'
            ELSE 'No Movement'
        END as movement_type
    FROM Daily_Asset_Classification_Snapshot curr
    LEFT JOIN Daily_Asset_Classification_Snapshot prev
        ON curr.account_id = prev.account_id
        AND prev.snapshot_date = curr.snapshot_date - INTERVAL '1 day'
)
SELECT
    snapshot_date,
    movement_type,
    COUNT(*) as account_count,
    SUM(current_outstanding) as total_amount
FROM movement_data
WHERE movement_type != 'No Movement'
GROUP BY snapshot_date, movement_type;

-- Quarterly NPA Movement Report
SELECT
    'Opening NPAs' as particulars,
    COUNT(*) as account_count,
    SUM(total_overdue) as amount
```

```
FROM Daily_Asset_Classification_Snapshot
WHERE snapshot_date = '2024-09-30'
AND classification_category IN ('Substandard','Doubtful','Loss')
UNION ALL
SELECT
   'Fresh NPAs during quarter',
   COUNT(DISTINCT account_id),
   SUM(amount)
FROM npa_movement_analysis
WHERE snapshot_date BETWEEN '2024-10-01' AND '2024-12-31'
AND movement_type = 'Fresh NPA';
```

PSL Achievement Using District Weighted Snapshots

```
sql
```

```sql
-- PSL Achievement Report with District Weights
CREATE PROCEDURE generate_psl_achievement_report(p_reporting_date DATE)
AS
BEGIN
    WITH district_weighted_psl AS (
        SELECT
            b.branch_id,
            b.branch_name,
            psl.category_name,
            psl.outstanding_amount,
            dm.district_name,
            dm.weight_percentage,
            psl.outstanding_amount * dm.weight_percentage / 100 as weighted_amount
        FROM (
            SELECT
                account_id,
                branch_id,
                psl_category as category_name,
                closing_balance as outstanding_amount
            FROM Daily_Balance_Snapshot dbs
            JOIN Account a ON dbs.account_id = a.account_id
            JOIN Loan_Account_Details lad ON a.account_id = lad.account_id
            WHERE dbs.snapshot_date = p_reporting_date
            AND lad.priority_sector_flag = TRUE
        ) psl
        JOIN Branch b ON psl.branch_id = b.branch_id
        JOIN District_Master dm ON b.district_id = dm.district_id
        WHERE dm.effective_date <= p_reporting_date
        AND (dm.end_date IS NULL OR dm.end_date > p_reporting_date)
    )
    SELECT
        branch_name,
        category_name,
        SUM(outstanding_amount) as actual_outstanding,
        SUM(weighted_amount) as weighted_outstanding,
        (SUM(weighted_amount) / total.total_advances * 100) as achievement_percentage
    FROM district_weighted_psl
    CROSS JOIN (
        SELECT SUM(closing_balance) as total_advances
        FROM Daily_Balance_Snapshot
        WHERE snapshot_date = p_reporting_date
        AND account_id IN (SELECT account_id FROM Loan_Account_Details)
    ) total
```

```
    GROUP BY branch_name, category_name, total.total_advances;
END;
```

7.3 Report Scheduling Matrix

Report Name	Frequency	Generation Time	Submission Deadline	Format
DSB	Daily	11:30 PM	9:00 AM next day	XML/XBRL
Form A (CRR)	Fortnightly	12:00 AM	Within 7 days	CSV
Asset Quality	Quarterly	1:00 AM	Within 5 days	XBRL
CRAR	Quarterly	2:00 AM	Within 5 days	XBRL
PSL Achievement	Quarterly	3:00 AM	Within 5 days	XML

7.4 XBRL Implementation

```xml
<!-- Sample XBRL Instance Document -->
<xbrl xmlns:rbi="http://www.rbi.org.in/xbrl/2024/supervisory">
  <context id="Q4_2024">
    <entity>
      <identifier scheme="http://www.rbi.org.in/ucb">UCB12345</identifier>
    </entity>
    <period>
      <instant>2024-12-31</instant>
    </period>
  </context>

  <rbi:GrossNPA contextRef="Q4_2024" unitRef="INR" decimals="2">
    1234567.89
  </rbi:GrossNPA>

  <rbi:NetNPA contextRef="Q4_2024" unitRef="INR" decimals="2">
    987654.32
  </rbi:NetNPA>
</xbrl>
```

---

8. Integration Requirements

8.1 External System Integrations

8.1.1 CIMS Portal Integration

```java
@Service
public class CIMSIntegrationService {

    @Value("${cims.api.url}")
    private String cimsApiUrl;

    public SubmissionResponse submitReport(Report report) {
        // Prepare submission package
        SubmissionPackage package = preparePackage(report);

        // Digital signature
        package.sign(certificateService.getCertificate());

        // Submit to CIMS
        RestTemplate restTemplate = new RestTemplate();
        HttpEntity<SubmissionPackage> request = new HttpEntity<>(package, headers);

        return restTemplate.postForObject(
            cimsApiUrl + "/submit",
            request,
            SubmissionResponse.class
        );
    }
}
```

8.1.2 CRILC Integration

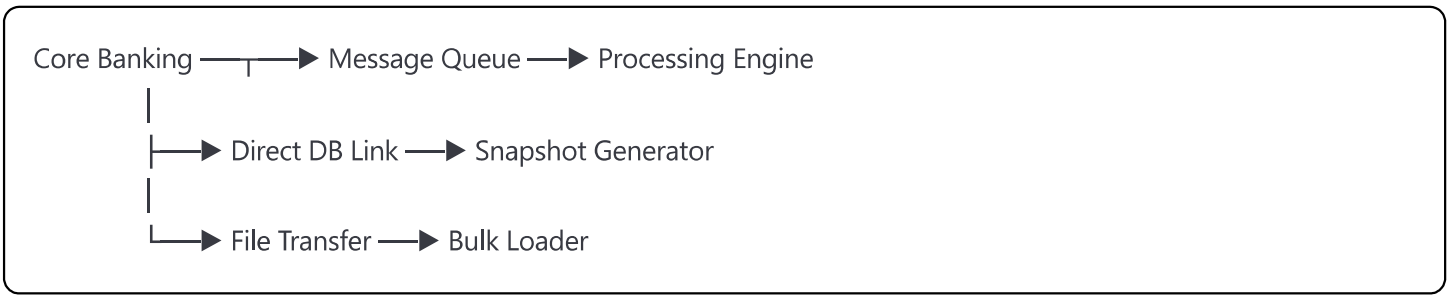
- Real-time large exposure updates
- Threshold-based automatic reporting
- Daily reconciliation of reported exposures

8.1.3 Core Banking System

- Real-time transaction feed via message queue
- Master data synchronization
- EOD balance reconciliation

8.2 Internal System Integrations

8.2.1 Data Flow Architecture



8.2.2 Integration Patterns

1. **Real-time**: Transaction processing, fraud alerts
2. **Near Real-time**: Balance updates, exposure monitoring
3. **Batch**: EOD processing, report generation
4. **On-demand**: Ad-hoc queries, investigations

8.3 API Specifications

8.3.1 REST API for Report Status

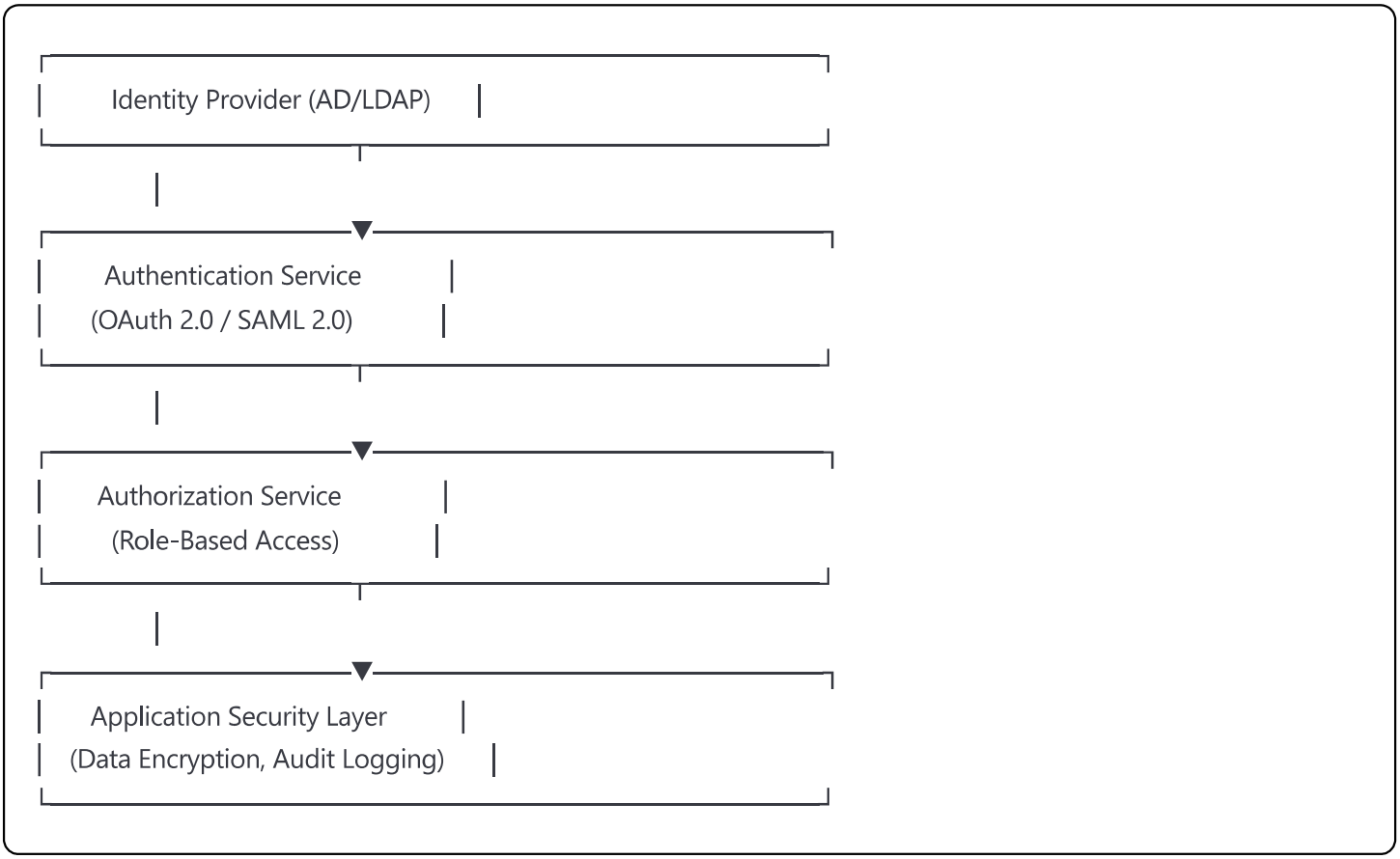
yaml

```yaml
openapi: 3.0.0
paths:
  /api/v1/reports/{reportId}/status:
    get:
      summary: Get report submission status
      parameters:
        - name: reportId
          in: path
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Report status
          content:
            application/json:
              schema:
                type: object
                properties:
                  reportId: string
                  status: enum [PENDING, SUBMITTED, ACCEPTED, REJECTED]
                  submissionTime: datetime
                  acknowledgmentNumber: string
```

9. Security and Compliance

9.1 Security Architecture

9.1.1 Access Control



9.1.2 Role Matrix

Role	Access Level	Permissions
Report Viewer	Read-only	View reports, dashboards
Report Maker	Create	Generate reports, validate data
Report Checker	Approve	Review and approve reports
Admin	Full	System configuration, user management
Auditor	Read-only + Logs	View all data and audit trails

9.2 Data Security

9.2.1 Encryption

- **At Rest**: AES-256 for database encryption
- **In Transit**: TLS 1.3 for all communications

- **Application Level**: Field-level encryption for sensitive data

9.2.2 Data Masking

```sql
-- Masked view for non-privileged users
CREATE VIEW customer_masked AS
SELECT
    customer_id,
    CONCAT(SUBSTR(name, 1, 3), '***') as name,
    CONCAT('XXXX', SUBSTR(pan_number, -4)) as pan_number,
    CONCAT('XXXX-XXXX-', SUBSTR(aadhaar_number, -4)) as aadhaar_number
FROM customer;
```

9.3 Audit and Compliance

9.3.1 Comprehensive Audit Trail

```sql
CREATE TABLE audit_trail (
    audit_id BIGSERIAL PRIMARY KEY,
    user_id VARCHAR(50) NOT NULL,
    action_timestamp TIMESTAMP NOT NULL,
    action_type VARCHAR(20) NOT NULL,
    table_name VARCHAR(100),
    record_id VARCHAR(100),
    old_values JSONB,
    new_values JSONB,
    ip_address INET,
    session_id VARCHAR(100),
    user_agent TEXT
);

-- Trigger for automatic audit logging
CREATE TRIGGER audit_trigger
AFTER INSERT OR UPDATE OR DELETE ON customer
FOR EACH ROW EXECUTE FUNCTION audit_log_function();
```

9.3.2 Compliance Monitoring

- Automated compliance checks for regulatory limits

- Real-time alerts for threshold breaches
- Periodic compliance reports
- Exception tracking and resolution

9.4 Business Continuity

9.4.1 High Availability Architecture

- Active-Active database replication
- Load-balanced application servers
- Redundant message queues
- Geo-distributed disaster recovery

9.4.2 Backup Strategy

Data Type	Backup Frequency	Retention	Recovery Time
Transactional	Every 15 mins	30 days	< 1 hour
Snapshots	Daily	90 days	< 2 hours
Archives	Weekly	7 years	< 24 hours
Audit Logs	Daily	10 years	< 4 hours

---

10. Implementation Roadmap

10.1 Phase 1: Foundation (Months 1-3)

Deliverables:

1. Database schema implementation
2. Core master data migration
3. Basic transaction processing
4. Daily snapshot generation

Key Milestones:

- Week 4: Database setup complete
- Week 8: Master data migrated
- Week 12: Daily processing operational

10.2 Phase 2: Core Reports (Months 4-6)

Deliverables:

1. DSB report automation
2. Asset quality reports
3. Basic CIMS integration
4. Validation framework

Key Milestones:

- Week 16: First automated DSB submission
- Week 20: Asset quality reports live
- Week 24: CIMS integration tested

10.3 Phase 3: Advanced Features (Months 7-9)

Deliverables:

1. All quarterly reports automated
2. XBRL implementation
3. Real-time monitoring dashboards
4. Advanced analytics

Key Milestones:

- Week 28: Quarterly reports automated
- Week 32: XBRL submission successful
- Week 36: Full dashboard deployment

10.4 Phase 4: Optimization (Months 10-12)

Deliverables:

1. Performance optimization
2. Advanced security features
3. Disaster recovery testing
4. User training completion

Key Milestones:

- Week 40: Performance benchmarks met

- Week 44: DR drill successful
- Week 48: System fully operational

10.5 Success Metrics

Metric	Target	Measurement
Report Accuracy	99.9%	Error rate in submissions
Submission Timeliness	100%	On-time submission rate
System Availability	99.5%	Uptime percentage
Processing Time	< 2 hours	EOD completion time
User Satisfaction	> 4.5/5	Survey scores

---

11. Appendices

Appendix A: Glossary of Terms

Term	Definition
CIMS	Centralized Information Management System
CRAR	Capital to Risk-weighted Assets Ratio
DSB	Daily Statement of Business
NPA	Non-Performing Asset
PSL	Priority Sector Lending
SMA	Special Mention Account
XBRL	eXtensible Business Reporting Language

Appendix B: Regulatory References

1. Master Direction - Filing of Supervisory Returns, 2024
2. Master Direction - Fraud Risk Management, 2024
3. Master Direction - Credit Information Companies, 2025
4. Master Circular - Priority Sector Lending, 2024

Appendix C: Technical Standards

1. **Database Standards**
   - Naming conventions: snake_case for tables/columns
   - Index naming: idx_tablename_columns

- Constraint naming: fk_childtable_parenttable

2. **API Standards**
   - RESTful design principles
   - JSON request/response format
   - OAuth 2.0 authentication
3. **Code Standards**
   - Java coding standards (Google style guide)
   - Unit test coverage > 80%
   - Code review mandatory

Appendix D: Sample Reports

[This section would include mockups of key reports in their final format]

Appendix E: Risk Register

Risk	Probability	Impact	Mitigation
Data migration errors	Medium	High	Phased migration with validation
Integration delays	High	Medium	Early POC development
Performance issues	Medium	High	Load testing and optimization
Regulatory changes	Low	High	Flexible architecture

Document Control

Version	Date	Author	Changes
1.0	Jan 2025	Design Team	Initial version

Approval Sign-offs

Role	Name	Signature	Date
Project Sponsor			
Technical Architect			
Compliance Head			
IT Head			