# Full Stack MERN Assignment – Smart Reconciliation & Audit System

Assignment Objective:
The objective of this assignment is to evaluate the candidate's ability to design and implement a real-world full stack application using the MERN stack (MongoDB, Express.js, React, Node.js). The focus is on system thinking, data handling, performance, auditability, and role-based access, rather than basic CRUD operations.

## Assignment Duration & Level

- Expected Duration: 3–5 Days

## Problem Statement

Build a Smart Reconciliation & Audit System that allows users to upload transaction data, reconcile it against system records, identify mismatches or duplicates, and maintain a complete audit trail of all actions performed on the data.

## Frontend Requirements (React)

### 1. Reconciliation Dashboard

- Display summary cards:
  - Total records uploaded
  - Matched records
  - Unmatched records
  - Duplicate records
  - Reconciliation accuracy percentage
- Include at least one chart (bar or donut)
- Filters: Date range, status, uploaded by
- Dashboard data must update dynamically based on filters

### 2. File Upload & Column Mapping

- Support CSV or Excel file uploads
- Show preview of first 20 rows before submission
- Allow users to map uploaded columns to system fields
- Mandatory fields: Transaction ID, Amount, Reference Number, Date
- Allow correction of mapping without re-uploading the file

### 3. Reconciliation View

- Display system record vs uploaded record
- Show match status:
  - Matched
  - Partially Matched
  - Not Matched
  - Duplicate

- Highlight mismatched fields in partial matches
- Allow manual correction of records

#### 4. Audit Timeline (UI)

- Show a timeline view for each record
- Capture who made the change, what was changed, and when
- Timeline should not be plain text; use a visual sequence

## Backend Requirements (Node.js + Express)

#### 1. Upload Processing

- Handle files up to 50,000 records
- Processing must be asynchronous and non-blocking
- Upload status: Processing, Completed, Failed
- Synchronous file processing is not acceptable

#### 2. Reconciliation Logic

- Exact Match: Transaction ID + Amount
- Partial Match: Reference Number matches with amount variance ±2%
- Duplicate: Same Transaction ID occurs more than once
- Unmatched: No match found
- Matching rules must be configurable, not hardcoded

#### 3. Idempotency & Data Consistency

- Uploading the same file multiple times must not duplicate records
- Reprocessing should reuse existing results if data has not changed

#### 4. Audit Trail

- Maintain a separate audit log collection
- Capture old value, new value, user, timestamp, and source of change
- Audit logs must be immutable

## Database Requirements (MongoDB)

- Required collections:
  - Users
  - UploadJobs
  - Records
  - ReconciliationResults
  - AuditLogs
- Mandatory indexes:
  - Transaction ID
  - Reference Number
  - Upload Job ID

## Authentication & Authorization

- Roles: Admin, Analyst, Viewer
- Admin: Full access
- Analyst: Upload and reconcile
- Viewer: Read-only
- Role enforcement must exist in both frontend and backend

## Non-Functional Requirements

- System should handle large data volumes efficiently
- UI should remain responsive during uploads
- Partial failures should not break the entire process
- Clear and actionable error messages should be shown

## Deliverables

- GitHub repository
- README explaining architecture, assumptions, trade-offs, and limitations
- Sample input files
- API documentation (Postman or Swagger)

## Evaluation Criteria

- Architecture and system design
- Backend performance and async handling
- Frontend UX and state management
- Data reconciliation accuracy
- Audit trail completeness
- Code quality and documentation