MoMo SMS Dashboard - Fullstack Data Processing and Visualization

Author: Jeremie Star & Grevy-k Date: June 2025

1. Overview

The MoMo SMS Dashboard is a fullstack web application designed to process and visualize mobile money SMS data provided in XML format by MTN Rwanda. The system extracts structured data from raw SMS messages using regular expressions, stores it in a relational PostgreSQL database, and provides an API to serve insights to a frontend dashboard built using Next.js.

The application is structured to support real-time analytics, filtering, and categorization of transactions for reporting or auditing purposes.

2. System Architecture

- Backend: Node.js, Express.js, Prisma ORM
- Database: PostgreSQL
- Frontend: Next.js 14 (App Router)
- Parsing: xml2js, Regex
- ORM: Prisma for schema and DB operations
- Deployment: Planned on Render

3. Implementation Approach

The XML file (~1600 messages) is parsed using xml2js. We define regex patterns for different transaction types and loop through each message to extract key fields such as amount, timestamp, txId, fee, balance, and counterparty.

A shared `enrich()` function is used to normalize the data and fill in any missing fields. The data is stored in PostgreSQL using Prisma ORM.

4. API Design

An Express.js API provides endpoints like:
- GET /api/transactions
- GET /api/transactions/incoming
- GET /api/transactions?type=WITHDRAWAL

These endpoints support filtering and pagination and return structured JSON from the database.

MoMo SMS Dashboard - Fullstack Data Processing and Visualization

Author: Jeremie Star & Grevy-k Date: June 2025

5. Frontend Integration (Next.js)

The frontend is built with Next.js (App Router) and Tailwind CSS. It fetches data via a utility API module and displays tables and charts using components. Environment variables are used to connect to the backend API.

6. Challenges & Solutions

- Regex Matching: Solved with fallback enrichment
- Duplicate txId Errors: Caught with Prisma error handling
- Google Fonts Errors: Switched to <link> tag instead of next/font/google
- Real vs Mock Data: Removed mock generators and confirmed API usage

7. Key Decisions

- Prisma ORM for clean schema + DB handling
- Next.js App Router for modern frontend
- No Bootstrap: used Tailwind for modern look
- Structured enrichment of missing txId, channel, fee, etc.

8. Future Improvements

- Add admin authentication
- Export CSVs
- Add pagination, advanced filters
- Deploy backend + frontend on Render

9. Conclusion

This fullstack project successfully transforms unstructured SMS logs into structured, queryable financial data with a clean dashboard interface. It demonstrates strong backend processing, API design, and frontend visualization skills to solve a real-world challenge.