# MTN MoMo SMS Dashboard Report

## Introduction

This project focuses on processing and analyzing MTN MoMo SMS data. It aims to help users better understand their mobile money transactions by automatically extracting key information, categorizing transactions, and displaying trends on a clean, simple dashboard.

## Project Overview

The system reads raw MoMo SMS messages in XML format, extracts and cleans the transaction details, and stores them in a structured database. Users can view and search transactions, filter by category, and visualize spending patterns over time.

## Technologies Used

- **Python** – for data processing and XML parsing
- **SQLite** – lightweight, file-based relational database
- **Flask** – simple backend API
- **JavaScript + Chart.js** – for data visualization
- **HTML/CSS** – clean frontend layout

# System Architecture

## Database Design Decisions

We used a relational database (SQLite) for simplicity and easy integration. Tables were created for transactions with fields like amount, type, party, and timestamp. Indexes were added on date and amount for faster querying.

## Data Processing Approach

A Python script parses each XML file, extracts useful information (like sender, amount, and purpose), and stores it in the database. Duplicates are skipped, and malformed entries are logged separately.

## Frontend Architecture

The dashboard is built using plain HTML/CSS and JavaScript. Chart.js renders graphs, and all data is fetched from a simple Flask API. The layout is responsive and mobile-friendly.

# Implementation Details

## Key Algorithms Used

- **Regex Extraction** – used to find values like phone numbers and currency amounts
- **Categorization Logic** – keywords in the SMS body are matched to assign a category (e.g., withdrawal, airtime, payment)

## Challenges Faced and Solutions

- **Parsing Issues** – Some SMS formats were inconsistent. We added fallback patterns and logs for debugging.
- **Performance** – As the database grew, queries slowed down. Indexing key fields fixed this.

## Data Categorization Logic

Transaction messages were categorized using keyword matching. For example:

- “Cash in” → Deposit
- “Buy airtime” → Airtime
- “Send money” → Transfer
  If no keyword matched, the category was set to “Uncategorized”.

# Results and Insights

## Transaction Patterns Discovered

- Airtime top-ups and transfers were the most common transaction types.
- Certain numbers were frequently reused—likely service numbers.

## Performance Considerations

- The app works well with thousands of records, but pagination and lazy loading could help scale to tens of thousands.
- API endpoints are basic and would benefit from caching for repeat queries.

### Future Improvements

- Add user authentication for personal dashboards
- Enable CSV export
- Refine categorization with machine learning or more advanced logic

## Conclusion

### Lessons Learned

- A consistent SMS structure is crucial for automation
- Lightweight tools (like SQLite and plain JS) are effective for small-scale tools
- Testing with real data early helps avoid parsing issues later