# MoMo App Development Report

## 1. Introduction

The MTN MoMo SMS Data Processor & Dashboard is a project aimed at automating the extraction, categorization, and visualization of MTN MoMo transaction messages. The primary goal was to help users track financial trends, analyze transactions, and gain valuable insights into their economic activities. The app processes data from SMS-based transactions, categorizes them, and presents them via an interactive web-based dashboard.

Team Members:

i. David Muotoh Francis
ii. Nformi
iii. Josue Byiringiro

## 2. Approach

### Frontend Development:

The app's frontend was built using HTML, CSS, and JavaScript, focusing on providing a clean, responsive interface for users to interact with the financial data.

The dashboard includes key features such as transaction type analysis (via bar charts), a recent transaction table (sortable and searchable), and customizable timeframes for users to filter the data and view analytics.

### Backend Development:

For the backend, Flask was chosen due to its simplicity and efficiency for rapid development.

SQLite was used for the database, which proved to be a lightweight solution, handling the structured transaction data with ease.

Data extraction from SMS was performed using XML parsing, where raw SMS data was converted into structured records, allowing for better categorization and analysis.

## 3. Challenges Encountered

### Data Extraction and Categorization:

A significant challenge was correctly extracting all the necessary data from raw SMS messages. Transaction IDs, amounts, and transaction types were inconsistently formatted, and handling this variability required creating robust parsing and cleaning techniques.

The regex logic had to be fine-tuned to handle different cases, such as extracting the transaction ID only when it followed the "TxId:" label. This issue also extended to amounts, where RWF (Rwandan Franc) appeared after most amounts, but there were exceptions that required careful handling.

**Handling Special Cases (OTP and Data Bundles):**

Some SMS messages contained OTP (One-Time Password) codes, which were not relevant to the transaction data but needed to be identified and filtered out.

Additionally, there was an inconsistency in the categorization of data bundles. In some cases, data bundles were categorized as transactions, while in others, they were categorized as payments. A step-by-step categorization process helped isolate these discrepancies and reclassify them correctly.

**Error Handling and Logging:**

Tracking and handling errors in data extraction was crucial. Malformed messages or incomplete data required specific logging and review procedures to ensure all valid messages were processed correctly.

**4. Key Decisions**

**Frontend Development:**

Task delegation played a significant role in the development process. The frontend was designed to be intuitive, with charts and filters that allow users to explore transaction data effectively. JavaScript dynamically updated the charts and tables based on the selected filters.

**Backend and Data Processing:**

Flask was selected for the backend due to its minimal setup and ease of use for small to medium-sized applications. SQLite was used for the database because it has minimal setup requirements and is well-suited for this small-scale web application.

The task of categorizing the data into transaction types and handling various special cases (OTP, data bundles) was prioritized to ensure the integrity of the data.

**Project Management:**

The project was divided into frontend and backend tasks to ensure rapid development. The team worked collaboratively, sharing responsibilities to implement features quickly and effectively.

**5. Conclusion**

The MoMo SMS Data Processor & Dashboard was implemented using a full-stack approach with Flask, SQLite, and JavaScript. The main challenges of data extraction, categorization, and visualization were effectively addressed, and the app provides valuable insights into financial transactions. In the future, additional features such as advanced filters and deeper analytics can be added to further enhance the platform's functionality.