Technical Assessment: Data Engineer / Business Intelligence Engineer

This report outlines the approach taken by Qenehelo Matjama to fulfill the technical assessment requirements for the Data Engineer / Business Intelligence Engineer role. The project involves building a robust, visually engaging, and insight dashboard powered by real-time blockchain token transaction data.

Project Overview

The primary objective of this assessment was to extract token transaction data from XCAP's public blockchain API, process and clean it, derive key insights, and build a Streamlit-powered dashboard accessible to stakeholders via the web. The final deployed dashboard is available at:

Click here to view the live dashboard

https://nwoysq9dt2vrvoqfvndzrh.streamlit.app/

Tools & Technologies Used

- • Python (pandas, requests, matplotlib, openpyxl)
- • Streamlit for dashboard visualization
- • GitHub for version control
- • XCAP blockchain API (https://xcap-mainnet.explorer.xcap.network/api/v2/token-transfers)

Pipeline Design & Architecture

Architecture Diagram

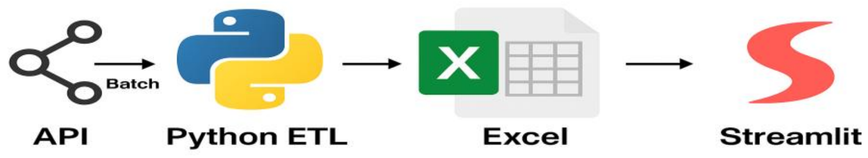

Figure: Architecture diagram showing the batch data pipeline using Python, Excel, and Streamlit.

The architecture follows a batch-oriented design. The ingestion of data from the XCAP API can take several minutes, especially when fetching multiple pages. Hence, the data is fetched periodically and written to a locally saved Excel file which serves as a temporary database. This allows the dashboard to load much faster and provide a smooth user experience.

This Excel file is committed to the GitHub repo and used as the primary source in the dashboard. This is a strategic tradeoff for performance and usability during review, especially considering that the API data doesn't change rapidly in real-time.

Should this pipeline be moved to production, a background scheduler or cron job could automate the ingestion every hour and push updated results to cloud storage or a lightweight database. ETL Pipeline script

Live API ETL

Due to API limitations and response time, fetching and processing 50 pages of blockchain token transactions takes a considerable amount of time. To improve performance and user experience, I implemented a batch update pipeline that stores the cleaned data in an Excel workbook. Streamlit then loads this Excel file at runtime.

This design improves speed while still enabling dynamic filtering, exploration, and visualization of data. It allows for frequent updates to the Excel file through scheduled scripts, maintaining a near-real-time view without API delays.

Key Features of the Dashboard

- • Token metrics such as supply, burned, minted, and transferred tokens.
  • Top 10 token holders, senders, and receivers.
  • Daily transfer volumes and cumulative supply per token.
  • Most traded tokens.
  • Full transaction table with dynamic filters (by token, type, date range, exchange rate).
- **How to Use the Token Analytics Dashboard**
  - **Filters**: Use the sidebar to filter records by Token Symbol, Transaction Type, Date Range, and Exchange Rate Range.
  - **Zooming and Panning**: On all charts, click and drag to zoom in. Double-click to reset zoom.
  - **Data Tables**: Below every chart, you'll find a data table reflecting the chart's values, providing full context. You can sort or scroll through them as needed.
  - **Download**: Right-click any chart in the browser to save it as an image. Tables can also be downloaded using the "Download as CSV" Streamlit context menu (visible on top-right of each table).

GitHub Repository

Click here to access Pipeline files

https://github.com/qmatjama/Data-Engineering-Assesment

📁 File Structure (GitHub)

```
├── API_DE/
│   ├── streamlit_app.py        # Dashboard code
│   ├── requirements.txt        # Dependencies
│   └── README.md
├── DE_Assesment_Results.xlsx    # Batch-ingested live data from API
```

Conclusion

This solution demonstrates practical experience with API ingestion, data transformation, analytical computation, and visualization using Python and Streamlit. The dashboard provides actionable insights into blockchain token activities. It demonstrates the ability to connect APIs, process data efficiently, and present it in a user-friendly and scalable web application.The choice to stage the data in Excel improves usability and avoids long wait times when loading the dashboard. In a production-grade implementation, a more scalable storage solution (MS SQL Server,MySQL, PostgreSQL, cloud RDS, Snowflake or any Cloud warehouse) would replace the Excel intermediary.

Additional Advanced Insights & ML Enhancements (streamlit_app_advanced.py) Proposed by Qenehelo

As part of extending the analytics for deeper understanding, an additional Streamlit app named `streamlit_app_advanced.py` was developed. This version provides enhanced exploratory data insights for stakeholders.

Key Features of streamlit_app_advanced.py

Click this link to view the advanced Dashboard

https://whjnbzx5wc7vk9eb98br6w.streamlit.app/

📁 File Structure (GitHub)

```
Data-Engineering-Assesment/
├── API_DE/
│   ├── streamlit_app_advanced.py
```

1. Weekly & Monthly Aggregated Token Volume

- Displays weekly and monthly token activity volume across the blockchain
- Helps identify seasonal or periodic trends.
- Includes interactive pivot tables below each chart.

2. Spike Detection in Minting & Burning

- Highlights abnormal behavior such as sudden minting or burning events.
- Enables early detection of unusual activities for further investigation.

3. Most Actively Traded Tokens by Count

- Identifies which tokens are most frequently traded.
- Useful for performance benchmarking or liquidity monitoring.

4. Integrated Filters and Pivot Tables

- All visuals respond to Token, Transaction Type, Date Range, and Exchange Rate filters.
- Every chart is paired with a downloadable pivot table.

Machine Learning Opportunities (Future-Ready Enhancements)

Two impactful ML-driven features were identified to further strengthen the dashboard's analytical power:

1. Anomaly Detection (Fraud or Spike Detection)

- Objective: Automatically detect suspicious token activity such as sudden volume spikes.
- Suggested Algorithm: IsolationForest from Scikit-Learn.
- Libraries: scikit-learn, pandas, matplotlib.
- Outcome: Flag transactions or dates with potential irregularities for alerts.

2. Forecasting Token Transfer Volume

- Objective: Predict the next day's transfer volume per token.
- Suggested Algorithm: Prophet or ARIMA from statsmodels
- Libraries: prophet, pandas, plotly
- Outcome: Enables demand prediction, capacity planning, and resource allocation.