MoMo SMS Data Processor — Report

1. Introduction to API security

This report documents the development, testing, and validation of a REST API for processing MoMo SMS transactions. The API allows clients to securely interact with transaction data and demonstrates efficient search operations using Python data structures.

The API uses **Basic Authentication** to restrict access. While suitable for a prototype:

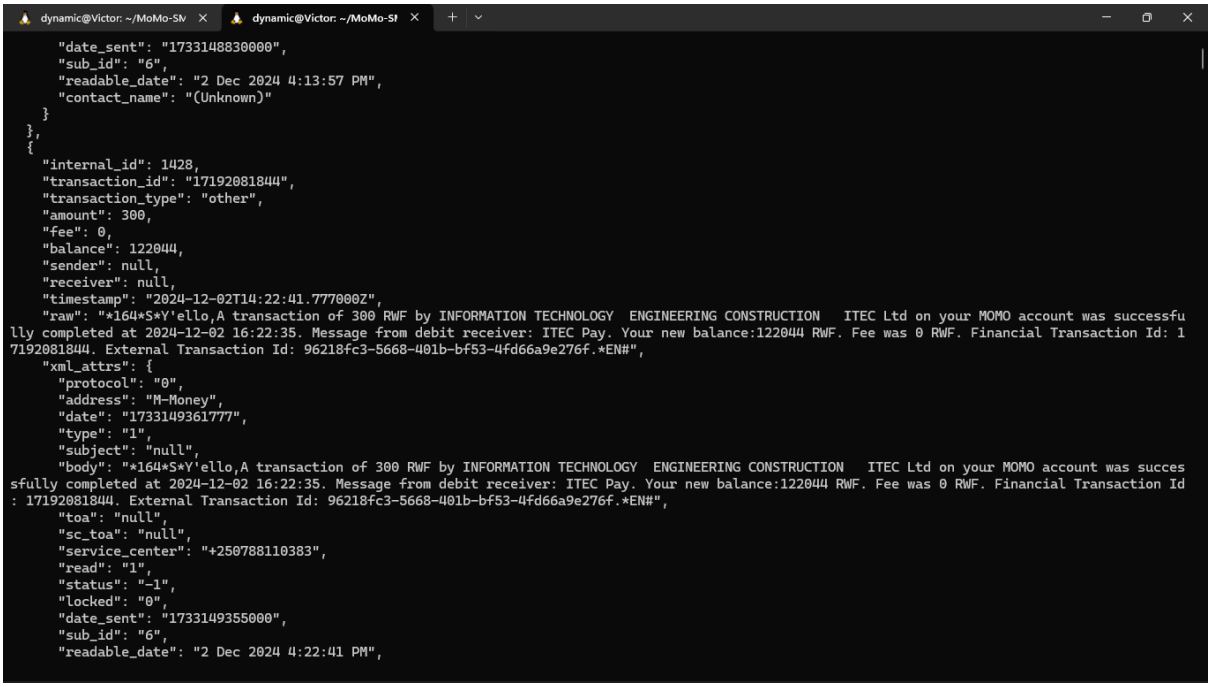
- Credentials are sent in every request (even if Base64-encoded).
- Weakness: Susceptible to interception over plain HTTP.

Recommended improvements for production:

- **HTTPS** for encrypted communication.
- **JWT (JSON Web Tokens)** or **OAuth2** for token-based, stateless authentication.
- Rate limiting and input validation to prevent misuse.

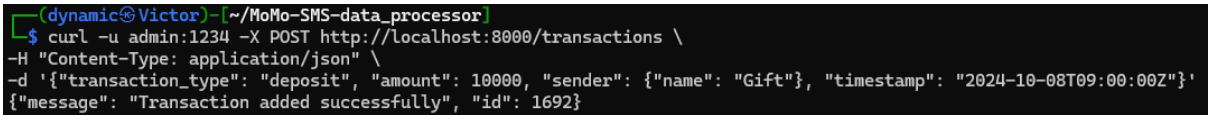
2. Documentation of endpoints

- GET /transactions — all transactions

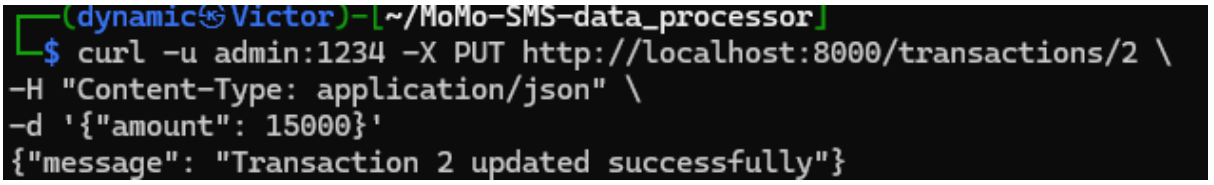


- GET /transactions/{id} — a single transaction by ID

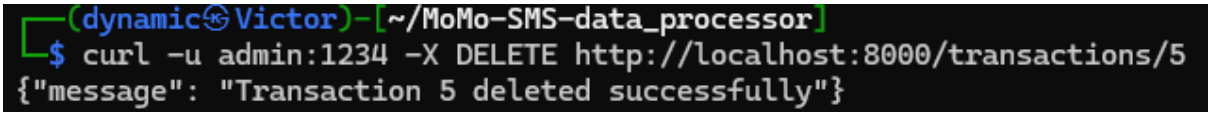
- POST /transactions — add a new transaction



- PUT /transactions/{id} — update a transaction



- DELETE /transactions/{id} — remove a transaction



3. DSA Comparison — Search Efficiency

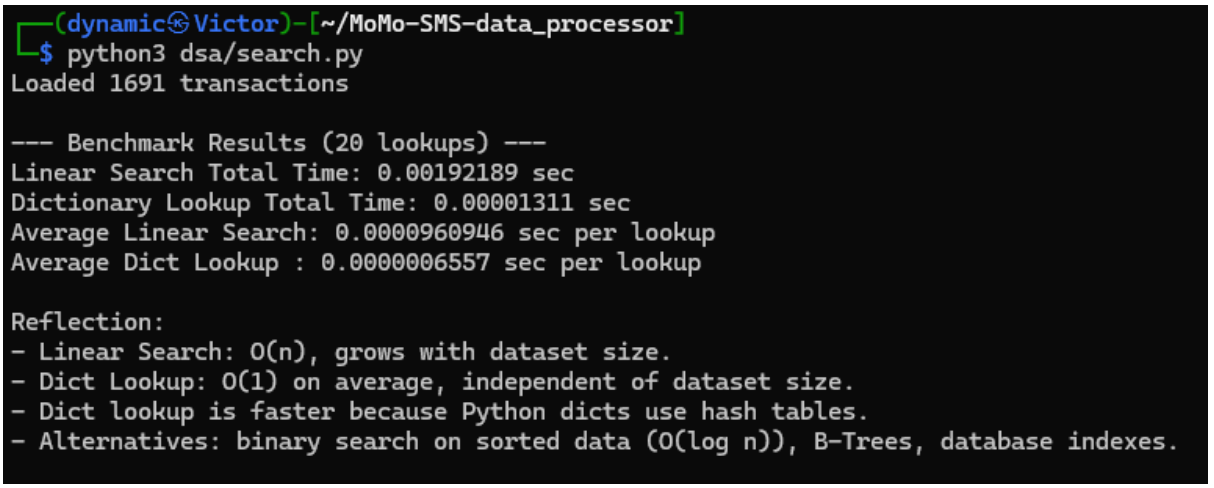
Setup

Transactions parsed from `modified_sms_v2.xml` (1691 records). Two search methods were compared:

1. **Linear Search:** Sequential scan of the list.
2. **Dictionary Lookup:** Use transaction IDs as keys in a Python dictionary (hash table).

Reflection:

- Linear search time grows with dataset size (O(n)).
- Dictionary lookup is nearly constant (O(1)) thanks to hash tables.
- Alternative data structures for efficient search:
    - Binary search on a sorted list (O(log n))
    - B-Trees or database indexes
    - Hash tables for in-memory lookup



4. Reflection on Basic Auth limitations

Basic Authentication provides a simple way to secure API endpoints by requiring a username and password for each request. It is easy to implement, test, and integrate with tools like `curl` or Postman, making it suitable for prototypes and small internal systems.

However, for production environments, stronger methods are preferred. Using HTTPS ensures that credentials are transmitted securely, while token-based authentication (JWT) or OAuth2 provides better control, session management, and scalability. These approaches enhance security without adding much complexity to client-server communication.