# Building and Securing a REST API Report

## 1. Introduction to API Security

This report documents the implementation, security considerations, documentation, and data-structure performance comparison for a REST API built in plain Python (http.server) to expose mobile money SMS transaction records parsed from an XML dataset. The API implements CRUD endpoints and Basic Authentication for access control.

Security summary:
- Endpoints are protected with HTTP Basic Authentication.
- On invalid or missing credentials, the API returns 401 Unauthorized.
- Basic Auth is easy to implement but weak if TLS is not enforced.
- Recommended: use HTTPS, JWT, OAuth2, credential rotation, and logging.

This project implements a complete REST API system for processing SMS transaction data from a mobile money service. It includes:

- XML Data Parsing: Convert XML transaction records to JSON format
- CRUD API Endpoints: Full Create, Read, Update, Delete operations
- Basic Authentication: Secure endpoint access with credentials
- Data Structure Algorithms: Performance comparison between linear search and dictionary lookup
- Comprehensive Testing: Automated test suites and manual testing tools
- Complete Documentation: Detailed API documentation with examples

### Project Structure

REST API/

├── api/ # API implementation

│ ├── server.py # Main REST API server

│ └── test_api.py # Automated API testing

├── dsa/ # Data Structures & Algorithms

│ └── xml_parser.py # XML parsing and search comparison

├── data/ # Data files

│ └── modified_sms_v2.xml # Sample SMS transaction data

├── docs/ # Documentation

│ └── api_docs.md # Complete API documentation

├── screenshots/ # Test screenshots (to be added)

├── test_api_curl.bat # curl testing script

└── README.md # README

### Authentication

The API uses **Basic Authentication** with the following credentials:

| Username | Password |

|----------|----------|

| admin | password123 |

| user | user123 |

| demo | demo123 |

Authorization: Basic YWRtaW46cGFzc3dvcmQxMjM=

## 2. API Documentation — Endpoints

**Base URL: `http://localhost:8000`**

**GET /transactions** → List all SMS transactions.

**GET /transactions/{id}** → Retrieve one transaction by id.

**POST /transactions** → Add a new transaction (JSON body).

**PUT /transactions/{id}** → Update a transaction.

**DELETE /transactions/{id}** → Delete a transaction.

## 3. Request / Response Examples

Example POST request body (JSON):
{"id": "tx1001", "type": "transfer", "amount": 1250.50, "sender": "+250788000001", "receiver": "+250788000002", "timestamp": "2025-09-27T14:13:00Z"}

Example successful response (201 Created):

{"status":"created","transaction":{"id":"tx1001","type":"transfer","amount":1250.5,...}}

Unauthorized example: 401 Unauthorized with header WWW-Authenticate: Basic realm="Protected".

## 4. Data Structures & Algorithm Comparison

Two lookup methods were implemented:
- Linear Search (O(n)): iterate through list to find transaction by id.
- Dictionary Lookup (O(1)): store transactions in a dictionary keyed by id.

Benchmark Results (20 lookups on dataset with 20 transactions):

- Linear Search: 0.000049 seconds (average)

- Dictionary Lookup: 0.0000068 seconds (average)

- Performance Improvement: 7.25x faster with dictionary lookup

## 5. Testing & Validation

Testing was performed using curl/Postman:
- Successful GET with authentication
- Unauthorized request (wrong credentials)
- Successful POST, PUT, DELETE

## 6. Deliverables & Repository Structure

Repository includes:
- api/ → API code
- dsa/ → XML parsing & DSA code
- docs/api_docs.md → API documentation
- screenshots/ → test screenshots
- README.md → setup instructions
- modified_sms_v2.xml → dataset
- RestAPI_Report.docx → this report

## 7. Reflection on Basic Auth

**Recommended improvements:**

- Implement HTTPS/TLS encryption

- Use JWT tokens with expiration

- Add rate limiting and request logging

- Implement OAuth2 for enterprise use

**Limitations:**
- Credentials sent with every request.
- No token expiry or session management.
- Poor for large-scale apps.