Building and Securing a REST API for MoMo SMS Data

(Team DEVSQUAD)

1. Introduction to API Security

Application Programming Interfaces (APIs) allow different software systems to exchange data.
Because APIs expose valuable data over the internet, **security is critical** to prevent unauthorized access, data breaches, and abuse.

Common security concerns include:

- **Authentication** – verifying that a client is who they claim to be.
- **Authorization** – enforcing what each authenticated client can access or modify.
- **Data confidentiality** – ensuring sensitive data is protected in transit.
- **Integrity** – preventing tampering with requests and responses.

Our assignment demonstrates these concepts by protecting API endpoints with **Basic Authentication**, which requires a valid username and password for every request.

2. API Endpoints

The API provides CRUD (Create, Read, Update, Delete) access to parsed MoMo transaction data.
All endpoints require **Basic Auth** (`devsquad / devsquadpass`) and return **JSON** responses.

Method	Endpoint	Description	Example Request	Example Response
GET	`/transactions`	List all transactions	`curl -u devsquad:devsquadpass`	`[ { "transaction_id"`

			`http://localhost:8000/transactions`	`:"76662021700", ...}, ...]`
GET	`/transactions/{id}`	Retrieve one transaction by ID	`curl -u devsquad:devsquadpass http://localhost:8000/transactions/76662021700`	`{ "transaction_id" :"76662021700", "amount":2000.0, ...}`
POST	`/transactions`	Add a new transaction	`curl -u devsquad:devsquadpass -H "Content-Type: application/json" -d @new_tx.json -X POST http://localhost:8000/transactions`	`{ "transaction_id" :"TEST-0001", "amount":5000, ...}`
PUT	`/transactions/{id}`	Update an existing transaction	`curl -u devsquad:devsquadpass -H "Content-Type: application/json" -d '{"amount":6000}' -X PUT http://localhost:8000/transactions/TEST-0001`	`{ "transaction_id" :"TEST-0001", "amount":6000, ...}`
DELETE	`/transactions/{id}`	Delete a transaction	`curl -u devsquad:devsquadpass -X DELETE http://localhost:8000/transactions/TEST-0001`	*(no content, HTTP 204)*

Error Codes

Code	**Meaning**
`200` OK	Successful GET or PUT
`201` Created	New transaction added
`204` No Content	Successful delete
`400` Bad Request	Malformed JSON or missing fields

`401` Unauthorized	Missing or invalid credentials
`404` Not Found	Transaction does not exist
`409` Conflict	Transaction ID already exists

---

3. Results of Data Structure & Algorithm (DSA) Comparison

To evaluate search efficiency, we compared:

- **Linear Search** – iterates through the list of transactions to find a match.
- **Dictionary Lookup** – stores transactions in a Python dictionary (`id → transaction`) for constant-time access.

We measured the time to perform 20 repeated lookups on a dataset of ~100 records.

Method	Total Time (sec)	Relative Speed
Linear Search	**0.324**	baseline
Dictionary Lookup	**0.012**	~27× faster

Dictionary lookup is significantly faster because it uses a hash table, providing *O(1)* average lookup time versus *O(n)* for linear search.

---

4. Reflection on Basic Auth Limitations

Basic Authentication was sufficient to demonstrate **concepts** of authentication, but it is **not secure for production**:

- **Plain credentials** – Username and password are sent with every request (even if base64 encoded, it is easily decoded).

- **Replay attacks** – Credentials can be captured and reused if the connection is not encrypted.

- **No token expiration** – If credentials are leaked, they remain valid until manually changed.

Stronger Alternatives

- **HTTPS + JWT (JSON Web Tokens):** Clients authenticate once and receive a signed token with expiration and claims.

- **OAuth 2.0:** Widely used for third-party access (e.g., Google/Facebook login).

- **Mutual TLS:** Client and server authenticate with certificates.

For real deployments, using **HTTPS with JWT or OAuth2** is recommended to protect sensitive financial data.

5. AI Assistance Disclosure

Language editing and formatting of this report were assisted by **ChatGPT (OpenAI, 2025)**. All code, API endpoints, data structures, and measurements were designed and implemented by the **DEVSQUAD** team.