MoMo REST API – Report

1. Introduction to API Security
- Basic Auth sends credentials (base64-encoded) on every request.
- Weaknesses:
  - No built-in expiry/rotation.
  - Credentials stored on client; easily leaked if reused.
  - If not served over HTTPS, credentials can be intercepted.
- Stronger alternatives:
  - JWT (stateless tokens, expirations, refresh tokens).
  - OAuth2 (authorization framework for delegated access).
  - Mutual TLS (client certificates) in high-security contexts.

---

2. Endpoint Documentation

GET /transactions
- Returns a paginated list of all transactions.
- 200 OK → JSON list of transactions.
- 401 Unauthorized → If no/invalid credentials.

GET /transactions/{id}
- Returns a single transaction by ID.
- 200 OK → Transaction JSON.
- 404 Not Found → If ID does not exist.
- 401 Unauthorized → If no/invalid credentials.

POST /transactions
- Creates a new transaction.
- 201 Created → JSON of created record.
- 400 Bad Request → Missing or invalid fields.
- 401 Unauthorized → If no/invalid credentials.

PUT /transactions/{id}
- Updates an existing transaction by ID (partial update allowed).
- 200 OK → Updated transaction JSON.
- 404 Not Found → If ID does not exist.
- 401 Unauthorized → If no/invalid credentials.

DELETE /transactions/{id}
- Deletes a transaction by ID.
- 204 No Content → Success, no body.

- 404 Not Found → If ID does not exist.
- 401 Unauthorized → If no/invalid credentials.

---

3. DSA Comparison (Linear vs Dictionary)

- Linear search: O(n). Scans each item until a match.
- Dictionary lookup: Average O(1), uses hashing to retrieve directly.

Benchmark results (10,000 lookups on `data.json`):
- Linear search: 0.006612 s
- Dict lookup:0.001190 s
- Dict is ≈ 5.56× faster on this dataset.

Why is dict faster?
- Uses hashing to compute an index into a bucket array, avoiding scans.
- Collisions are handled internally (e.g., open addressing or chaining).

Alternatives
- Sorted list + binary search → O(log n).
- B-Tree/SQLite index → Efficient persistence and range queries.
- Trie → Efficient for prefix searches on sender/receiver fields.

---

4. Testing Evidence

Screenshots were captured to verify functionality:
1. Authorized GET /transactions → 200 OK
2. Unauthorized GET /transactions → 401 Unauthorized
3. POST /transactions → 201 Created
4. PUT /transactions/{id} → 200 OK
5. DELETE /transactions/{id} → 204 No Content

👉 Screenshots are included in the `screenshots/` folder as part of the submission.

---

5. Reflection on Basic Auth Limitations
- Best used over HTTPS in controlled environments.
- No token revocation, no scopes/permissions.
- Credentials are static and sent on every request.
- Consider JWT for stateless APIs or OAuth2 when third-party apps integrate.

Authorisation: REST API Authentication.

The reason why Basic Authentication is Weak.

The above API implementation makes use of Basic Authentication wherein a client puts the username and password in each request. The credentials are merely coded with Base64 which is not encryption and can be easily recovered.

The main weaknesses are:

Credentials are revealed on each request- The username and password are sent on the network each time. When they are not encrypted using HTTPS, they can be intercepted.

No expiration- Once a client possesses the credentials, they are valid till they have been altered manually.

Replay attacks - When an attacker captures the credentials, then he can reuse them to impersonate the user.

Hard-coded credentials - In this case, user name and password are embedded in the software and they are insecure and hard to maintain.

Due to these reasons, Basic Authentication can only be used in testing and with internal systems and not in production.

Stronger Alternatives

1. JWT (JSON Web Token)

When the user logs in once with his/her credentials, a signed token is given by the server.

This token is then sent in the Authorization header by the client instead of credentials.

Expiration time of tokens can be set, which minimizes the possibility of abuse.

The token is signed and thus, cannot be manipulated.

Advantages:

No requirement of sending username/ password on each request.

Scalable: tokens may be applied to various services.

Allows expiration and refresh tokens.

2. OAuth2

An industry-standard framework that is more advanced and used in Google, GitHub, Facebook etc.

Offers a token based system of authentication that has various grant types (Authorization Code, Client Credentials, etc.).

Enables third-party applications to retrieve user data safely without having to provide the password of the user.

Advantages:

Highly secure and flexible.

Popularized on huge platforms.

Tokens may be scoped (only restricted to certain permissions).