Momo App Backend Report

Introduction

The Momo App Backend is a secure and efficient system designed for handling Mobile Money transactions. This report outlines the architectural choices made during development to ensure security, performance, and reliability.

Technical Design Choices

1. Authentication: JWT vs. Basic Auth

We implemented **JSON Web Tokens (JWT)** for user authentication instead of traditional Basic Auth.

Rationale: Basic Auth requires sending credentials with every request, which increases the risk of interception. JWT allows for a single login event that generates a temporary, cryptographically signed token.

Security Benefits:

- By using tokens, the server remains stateless and does not need to store session data
- We use a blacklist to immediately invalidate tokens when a user logs out
- This approach significantly reduces the exposure of user credentials

Error Handling: To prevent account enumeration attacks, the system provides a generic error message ("Invalid email or password") regardless of whether the email or password was the incorrect field.

2. Data Security: Password Hashing

User passwords are never stored in plain text. We utilize the **bcrypt algorithm** for hashing.

Implementation Details:

- Bcrypt automatically incorporates a unique "salt" for every password
- This ensures that identical passwords used by different users result in different hashes
- The hashing process is one-way, meaning passwords cannot be retrieved from their hashes
- This provides security even if attackers somehow obtain the hash values

3. **Performance: Transaction Lookup Optimization**

   Optimizing data retrieval was a primary focus for this project. We evaluated two search methods for transaction records:

   Linear Search (Initial Approach)

   The system initially retrieved transactions by iterating through a list until a match was found.

   Limitation:

   - Time complexity: **O(n)**, meaning search time increases linearly with the number of transactions
   - For 10,000 records, the system might need to scan every record in the worst case
   - Performance degrades significantly as the dataset grows

   Dictionary Lookup - Indexed (Optimized Approach)

   To improve performance, we implemented an indexed lookup using a Python Dictionary.

   Optimization:

   - Time complexity: **O(1)**, ensuring nearly instantaneous retrieval
   - Performance remains consistent whether there are ten or ten million records
   - Dramatically improves user experience and system scalability

API Reference

The following table summarizes the primary endpoints available in the system:

Endpoint	**Method**	**Auth**	**Purpose**
/auth/register	POST	No	Creates a new account
/auth/login	POST	No	Authenticates user and generates a JWT
/auth/logout	POST	Yes	Token invalidation and session termination.

/transactions/	POST	Yes	Initiates a money transfer.
/transactions/me	GET	Yes	Retrieves the current user’s history.
/transactions/<id>	GET	Yes	Retrieve by ID (Linear Search).
/indexed_transactions/<id>	GET	Yes	Retrieve by ID (Dictionary Lookup).
/transactions/<id>	PUT	Admin	Update transaction metadata (Admin only).
/transactions/<id>	DELETE	Admin	Remove a transaction record (Admin only).