Momo App Database Documentation

The purpose of this database is to store, organise, and process data in a way that ensures accurate tracking, categorisation, and analysis of the momo sms.

The database contains six entities: transaction entity, user entity, transaction categories, system logs entity, transaction tags, and tags.
The core of the system is the transaction entity that records essential details such as transaction ID, date, amount, sender_user_id, receiver_user_id, category, and status.
This design ensures that every financial activity is uniquely identifiable and linkable to other parts of the system

User entity captures the customer and agent information, like phone number, user_id, account_type, registered_date, and their full name. One user can perform many transactions, and each transaction must have exactly one sender and one receiver. By linking users to transactions both as senders and receivers, we can build a comprehensive picture of money flows between individuals and organizations.

Transaction_categories entity helps us to classify each transaction into meaningful groups: Cash In, Cash Out, Airtime Purchase, or Bill Payment. Each transaction must be categorised into only one category, and one category can contain one or many transactions. This separation simplifies business intelligence and reporting, since transactions can be aggregated by type without repeatedly parsing raw SMS text.

System_Logs entity captures processing events and errors during ETL operations to maintain system reliability. Linking logs to transactions ensures traceability and supports debugging. Additionally, we included a Tags entity with a junction table (Transaction_Tags) to support many-to-many relationships, allowing flexible labeling of transactions (e.g., “Business Expense” and “High Value” simultaneously).

This design balances normalization and usability: it avoids data duplication, supports future scalability, and provides clear pathways for analysis. By separating users, transactions, categories, and logs, the database remains modular, easy to query, and ready to integrate with a dashboard or API in later stages.