

## Week 2 Assignment – Task 2

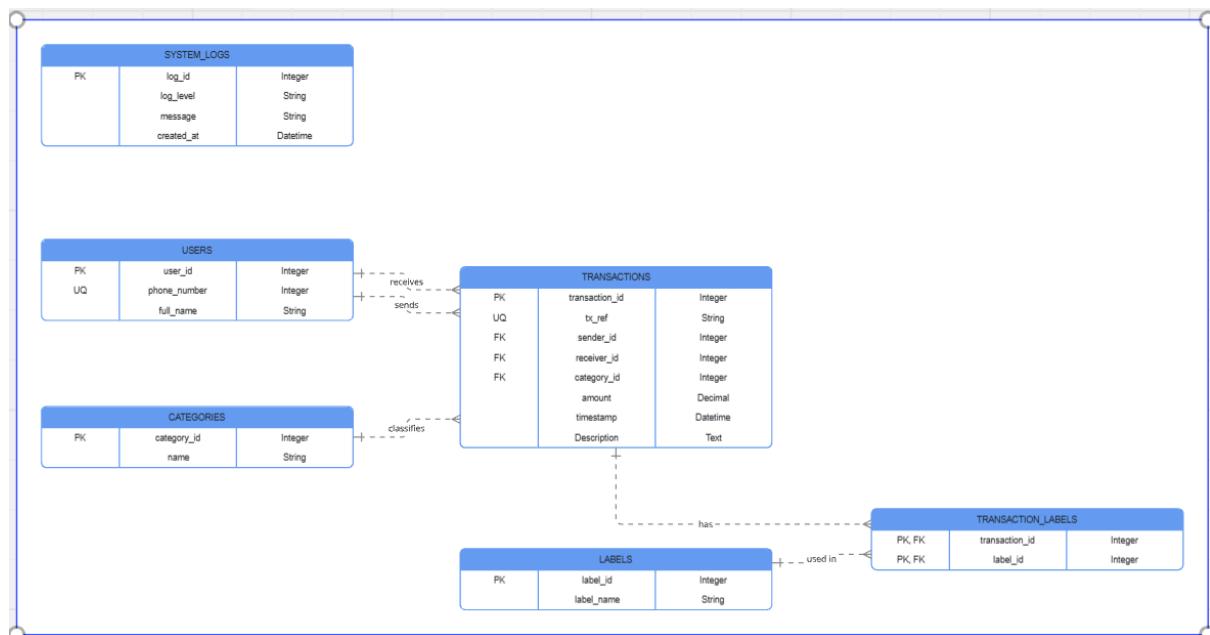
Group:

### 1. Entity Relationship Diagram (ERD)

<https://miro.com/app/board/uXjVGL5Z6Nk=/>

**Figure 1: Complete ERD**

(Entities: Users, Categories, Transactions, Labels, Transaction\_Labels, System\_Logs)



### 2. Design Rationale and Justification

The database schema was designed to faithfully represent the MoMo SMS transaction processing requirements while ensuring performance, data integrity, and future scalability.

Core entities were derived directly from the ERD:

- **Users** stores sender/receiver information with `phone_number` as UNIQUE BIGINT for fast lookup and indexing.
- **Categories** classifies transaction types (airtime, P2P, bill payment, etc.) to enable grouping and reporting.
- **Transactions** is the central fact table with composite relationships to **Users** (sender & receiver) and **Categories**. `tx_ref` is UNIQUE VARCHAR to prevent duplicates from SMS parsing. Amount uses DECIMAL(15,2) to handle large transfers accurately.
- **Labels** and **Transaction\_Labels** implement a many-to-many relationship for flexible tagging (fraud, high-value, promo, etc.).
- **System\_Logs** captures parsing/processing audit trail, linked optionally to transactions.

Key decisions:

- InnoDB engine for referential integrity and ACID compliance.
- FOREIGN KEY constraints with RESTRICT on core relationships to prevent orphan records, CASCADE on junction/logs for cleanup.
- CHECK (amount >= 0) prevents invalid negative values.
- Indexes on timestamp, sender\_id, receiver\_id, tx\_ref, and composite (category\_id + timestamp) optimize common queries (history by date, per user, per category).
- Phone numbers as BIGINT improve numeric indexing vs VARCHAR. Column comments and ENUM for log\_level enhance maintainability.

This design balances normalization, query performance, and real-world MoMo constraints while allowing easy extension (e.g., adding fraud detection flags).

### 3. Data Dictionary

#### Table: users

- user\_id: INT AUTO\_INCREMENT PRIMARY KEY
- phone\_number: BIGINT NOT NULL UNIQUE COMMENT 'Phone number stored as integer for indexing (e.g. 254712345678)'
- full\_name: VARCHAR(120) NOT NULL COMMENT 'Customer full name'
- created\_at: DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT 'Record creation timestamp'

#### Table: categories

- category\_id: INT AUTO\_INCREMENT PRIMARY KEY
- name: VARCHAR(80) NOT NULL UNIQUE COMMENT 'Transaction type (e.g. Airtime Purchase, P2P Transfer)'
- created\_at: DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP

#### Table: transactions

- transaction\_id: INT AUTO\_INCREMENT PRIMARY KEY
- tx\_ref: VARCHAR(64) NOT NULL UNIQUE COMMENT 'MoMo reference ID from SMS/provider'
- sender\_id: INT NOT NULL FOREIGN KEY → users.user\_id
- receiver\_id: INT NOT NULL FOREIGN KEY → users.user\_id
- category\_id: INT NOT NULL FOREIGN KEY → categories.category\_id
- amount: DECIMAL(15,2) NOT NULL CHECK (amount >= 0) COMMENT 'Transaction amount in KES'
- timestamp: DATETIME NOT NULL COMMENT 'Transaction occurrence time'
- description: TEXT NULL COMMENT 'Optional SMS excerpt or notes'
- created\_at: DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP

#### Table: labels

- label\_id: INT AUTO\_INCREMENT PRIMARY KEY

- label\_name: VARCHAR(80) NOT NULL UNIQUE COMMENT 'Tag e.g. High Value, Potential Fraud'

**Table: transaction\_labels (junction)**

- transaction\_id: INT NOT NULL FOREIGN KEY → transactions
- label\_id: INT NOT NULL FOREIGN KEY → labels
- assigned\_at: DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP
- PRIMARY KEY (transaction\_id, label\_id)

**Table: system\_logs**

- log\_id: INT AUTO\_INCREMENT PRIMARY KEY
- tx\_ref: VARCHAR(64) NULL
- transaction\_id: INT NULL FOREIGN KEY → transactions
- log\_level: ENUM('INFO','WARNING','ERROR','DEBUG') NOT NULL DEFAULT 'INFO'
- message: TEXT NOT NULL
- created\_at: DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP

#### 4. Sample Queries Demonstrating Functionality

Query 1: Recent high-value transactions (SELECT + JOIN)

SQL

```
SELECT t.tx_ref, s.full_name AS sender, r.full_name AS receiver, c.name AS category, t.amount
FROM transactions t
JOIN users s ON t.sender_id = s.user_id
JOIN users r ON t.receiver_id = r.user_id
JOIN categories c ON t.category_id = c.category_id
WHERE t.amount > 5000
ORDER BY t.timestamp DESC
LIMIT 5;
```

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The main area displays a query results grid titled 'Query 1 database\_setup\*'. The results show several transactions from the 'momo\_sms\_db' database. Below the grid, an 'Action History' section lists 30 actions with their timestamp, message, and duration.

#	Time	Action	Message	Duration / Fetch
21	20:38:25	INSERT INTO categories (name) VALUES ('Airtime Purchase'), ('P2P Transfer'), ('U...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.016 sec
22	20:38:25	INSERT INTO labels (label_name) VALUES ('Normal'), ('High Value'), ('Potential Fra...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.015 sec
23	20:38:25	INSERT INTO transactions (tx_ref, sender_id, receiver_id, category_id, amount, tim...	7 row(s) affected Records: 7 Duplicates: 0 Warnings: 0	0.016 sec
24	20:38:25	INSERT INTO transaction_labels (transaction_id, label_id) VALUES (1, 1), (1, 2), (3...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.000 sec
25	20:38:25	INSERT INTO system_logs (tx_ref, transaction_id, log_level, message) VALUES (M...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.000 sec
26	20:38:25	SHOW TABLES	6 row(s) returned	0.015 sec / 0.000 sec
27	20:38:26	DESCRIBE users	4 row(s) returned	0.016 sec / 0.000 sec
28	20:38:26	SELECT tx_ref, sfull_name AS sender, rfull_name AS receiver, c.name...	5 row(s) returned	0.000 sec / 0.000 sec
29	20:38:26	SELECT COUNT(*) FROM transactions LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
30	20:38:26	SELECT tx_ref, sfull_name AS sender, rfull_name AS receiver, c.name...	5 row(s) returned	0.000 sec / 0.000 sec

## Query 2: User transaction count

SQL

```
SELECT u.full_name, u.phone_number, COUNT(t.transaction_id) AS transaction_count
FROM users u
LEFT JOIN transactions t ON u.user_id = t.sender_id OR u.user_id = t.receiver_id
GROUP BY u.user_id
ORDER BY transaction_count DESC;
```

The screenshot shows the MySQL Workbench interface. The query results grid displays the transaction count for each user. The results are as follows:

full_name	phone_number	transaction_count
Amina Wanjiku	254712345678	3
Brian Ochieng	254723456789	2
Fatuma Hassan	254734567890	2
David Mwangi	254745678901	2
Esther Njeri	254756789012	2
George Kamau	254767890123	2
Halima Said	254778901234	1

## Query 3: Logs for a specific transaction (READ log)

SQL

```
SELECT log_level, message, created_at
FROM system_logs
WHERE tx_ref = 'MOMO20260124001'
ORDER BY created_at DESC;
```

```

217 ORDER BY transaction_count DESC;
218
219 • SELECT
220     log_level,
221     message,
222     created_at
223     FROM system_logs
224 WHERE tx_ref = 'MOMO20260124001'
225 ORDER BY created_at DESC;

```

log_level	message	created_at
INFO	Transaction parsed and inserted	2026-01-25 20:38:25

## 5. Unique Rules Added for Security & Accuracy (with screenshots)

Rule 1: CHECK constraint on amount (prevents negative/invalid amounts)

SQL

`CHECK (amount >= 0)`

```

1 • SHOW CREATE TABLE transactions;

```

Table	Create Table
transactions	CREATE TABLE `transactions` ( `transaction_id` int NOT NULL AUTO_INCREMENT, `tx_ref` varchar(64) COLLATE utf8mb4...

Rule 2: UNIQUE constraint on tx\_ref (prevents duplicate transactions)

```
transactions CREATE TABLE `transactions` ( `transaction_id` int NOT NULL AUTO_INCREMENT, `tx_ref` varchar(64) COLLATE utf8mb4...
```

Rule 3: FOREIGN KEY constraints with ON DELETE RESTRICT (protects data integrity)

```

CONSTRAINT fk_tx_sender    FOREIGN KEY (sender_id) REFERENCES users(user_id)   ON DELETE RESTRICT
ON UPDATE CASCADE,
CONSTRAINT fk_tx_receiver  FOREIGN KEY (receiver_id) REFERENCES users(user_id)   ON DELETE RESTRICT
ON UPDATE CASCADE,
CONSTRAINT fk_tx_category  FOREIGN KEY (category_id) REFERENCES categories(category_id) ON DELETE
RESTRICT ON UPDATE CASCADE,

```

Rule 4: phone\_number UNIQUE in users (prevents duplicate user records)

```
| CREATE TABLE users (
|     user_id      INT AUTO_INCREMENT PRIMARY KEY,
|     phone_number BIGINT      NOT NULL UNIQUE COMMENT 'Phone number (e.g. 2547xxxxxxxx format as
|     integer)',
```

These rules help maintain data accuracy (no negative money, no duplicates) and basic security (referential integrity).