Database Management System Project

Topic : Cellular network services

VIGNESH SINGARAVELU**(PES1UG22CS691)**

VEDANTH PADMARAMAN**(PES1UG22CS682)**

Purpose

This product implements the Databse sofware server side for managing and keeping track of a cellular network, meant for use by network operators. Operators may host this software on their own servers in order to monitor services used by customers. This product handles with high-level aspects of running the cellular network, such as status monitoring, data logging, user interfaces and so on.

Scope

The primary function of the product is to monitor and control high-level aspects of the cellular network, such as plan and contract selection, billing, and so on. In addition to providing an interface for businesses and end users, the software also records data useful for analytics and maintanence, such as usage patterns, per-tower traffic and status reports and error logs. While the product handles high-level aspects such as the user interface, data collection and monitoring, it does not deal with the lower-level technological aspects of the cellular network, such as modems, MSCs (mobile switching centres) and cellular towers. It complements cellular network infrastructure in order to ease management and service sale for the operators.

Description

The server orchestrates the interaction between the cellular network and its users, giving full control to the network operators. Two types of users are recognized: end users and business users. And end user is an individual, identified a user ID, who subscribes to the network, by means of one or more phone numbers, for personal use. To aid communication between the network operators, the user's e-mail ID and address are collected. Additionally, for regulatory compliance, a document for the proof of identity of the individual is stored; further, the IMEI and ownership of the user's device(s) are also recorded In order to avail network services, the user buys service plans for his/her phone numbers. Each plan allows calls, SMS messages and Internet access to varying limits, and is priced accordingly. A business user is an entity that uses the network for commercial purposes. Each business is identified by a unique ID and associated with a business name and GSTIN number. Business users avail network services through service contracts. Each contract, identified by a unique ID, has a fixed start and end date; other details of the contract are decided in a flexible manner on a per-case basis, unlike with end user service plans. To facilitate health monitoring, the location and capacity of each tower maintained by the network operator is stored along with a unique ID. Additionally, to enable analysis of usage patterns, usage statistics are collected per-device.

List of software and tools used :

- Python
- Tkinter
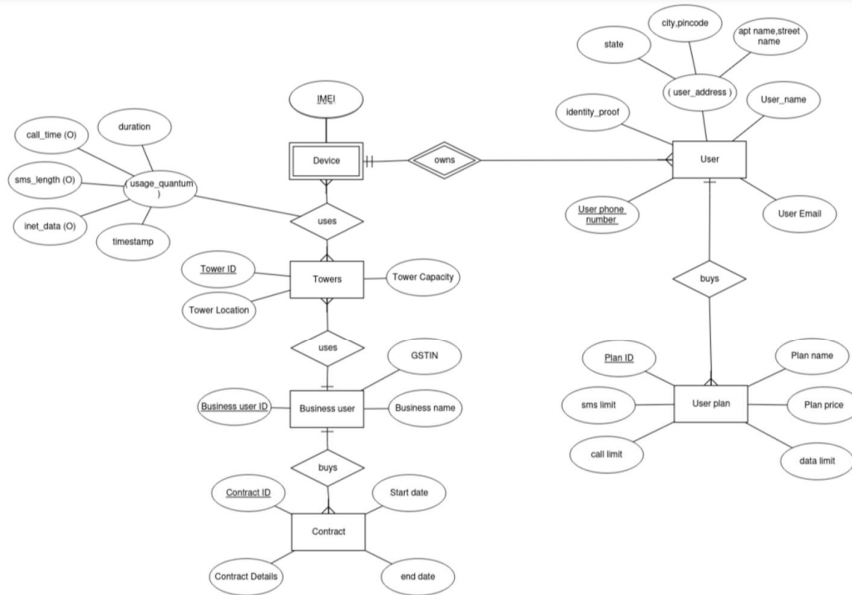- Mysql
- Mysql connector

Entity Relationship Diagram

Figure 1: Entity-Relation Diagram
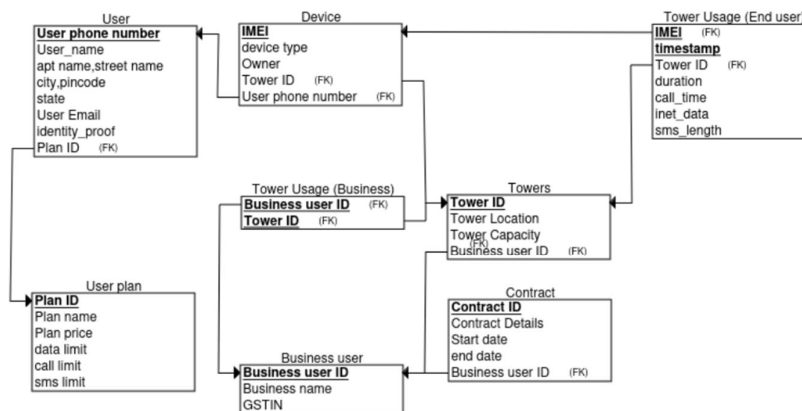
Relational Schema

Figure 2: Relational Schema

DDL commands :

CREATE DATABASE IF NOT EXISTS cellular_network;

USE cellular_network;

CREATE TABLE IF NOT EXISTS Device (

   device_type ENUM ("smartphone", "featurephone") NOT NULL,

   IMEI CHAR(15) NOT NULL,

   PRIMARY KEY (IMEI)

);

CREATE TABLE IF NOT EXISTS Business_user (

   Business_user_ID INT NOT NULL,

   Business_name VARCHAR(50) NOT NULL,

   GSTIN CHAR(15) NOT NULL,

   PRIMARY KEY (Business_user_ID)

);

CREATE TABLE IF NOT EXISTS Contract (

   Contract_ID INT NOT NULL,

   Contract_Details INT NOT NULL,

   Start_date TIMESTAMP NOT NULL,

   end_date TIMESTAMP NOT NULL,

   Business_user_ID INT NOT NULL,

   PRIMARY KEY (Contract_ID),

   FOREIGN KEY (Business_user_ID) REFERENCES Business_user(Business_user_ID),

   CHECK (end_date > Start_date)

);

CREATE TABLE IF NOT EXISTS User (

   User_name VARCHAR(50) NOT NULL,

   User_phone_number CHAR(10) NOT NULL,

   apt_name VARCHAR(50) NOT NULL,

   street_name VARCHAR(50) NOT NULL,

```sql
    city VARCHAR(50) NOT NULL,

    pincode VARCHAR(6) NOT NULL,

    state VARCHAR(50) NOT NULL,

    User_Email VARCHAR(100) NOT NULL,

    identity_proof INT NOT NULL,

    IMEI CHAR(15) NOT NULL,

    PRIMARY KEY (User_phone_number, IMEI),

    FOREIGN KEY (IMEI) REFERENCES Device(IMEI)
);


CREATE TABLE IF NOT EXISTS Towers (

    Tower_ID INT NOT NULL,

    Tower_Location VARCHAR(100) NOT NULL,

    Tower_Capacity INT NOT NULL,

    Business_user_ID INT NOT NULL,

    PRIMARY KEY (Tower_ID),

    FOREIGN KEY (Business_user_ID) REFERENCES Business_user(Business_user_ID),

    CHECK (Tower_Capacity > 0)
);


CREATE TABLE IF NOT EXISTS plan_spec (

    Plan_ID INT NOT NULL,

    Plan_name VARCHAR(50) NOT NULL,

    Plan_price INT NOT NULL,

    data_limit_GB INT NOT NULL,

    call_limit INT NOT NULL,

    sms_limit INT NOT NULL,

    PRIMARY KEY (Plan_ID),

    CHECK (Plan_price >= 0),

    CHECK (data_limit_GB >= 0),

    CHECK (call_limit >= 0),

    CHECK (sms_limit >= 0)
);
```

```sql
CREATE TABLE IF NOT EXISTS User_plan (

    Plan_ID INT NOT NULL,

    IMEI CHAR(15) NOT NULL,

    User_phone_number CHAR(10) NOT NULL,

    date_of_purchase TIMESTAMP NOT NULL,

    FOREIGN KEY (User_phone_number, IMEI) REFERENCES User(User_phone_number, IMEI),

    FOREIGN KEY (Plan_ID) REFERENCES plan_spec(Plan_ID)

);


CREATE TABLE IF NOT EXISTS uses (

    usage_quantum INT NOT NULL,

    IMEI CHAR(15) NOT NULL,

    Tower_ID INT NOT NULL,

    time_stamp TIMESTAMP NOT NULL,

    PRIMARY KEY (IMEI, Tower_ID),

    FOREIGN KEY (IMEI) REFERENCES Device(IMEI),

    FOREIGN KEY (Tower_ID) REFERENCES Towers(Tower_ID),

    CHECK (usage_quantum >= 0)

);


CREATE VIEW plan_popularity AS

    SELECT Plan_ID, COUNT(*) AS user_count

    FROM User_plan

    GROUP BY Plan_ID

    ORDER BY user_count DESC;


DELIMITER |

CREATE FUNCTION IF NOT EXISTS tower_load (tower_id INT, period_start TIMESTAMP, period_end TIMESTAMP)

RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN
```

```sql
DECLARE usage_total INT;

DECLARE capacity INT;


SELECT COALESCE(SUM(usage_quantum), 0)

INTO usage_total

FROM uses

WHERE

    uses.Tower_ID = tower_id AND

    uses.time_stamp > period_start AND

    uses.time_stamp < period_end;


SELECT Tower_Capacity

INTO capacity

FROM Towers

WHERE Towers.Tower_ID = tower_id;


IF capacity IS NULL OR capacity = 0 THEN

    RETURN 0;

END IF;


RETURN CAST(usage_total AS DECIMAL(10,2)) / CAST(capacity AS DECIMAL(10,2));

END |

DELIMITER ;
```

CRUD operation

```
mysql> -- Add devices
mysql> INSERT INTO Device (device_type, IMEI) VALUES
    ->     ('smartphone', '1234567890012345'),
    ->     ('smartphone', '234567890123456'),
    ->     ('featurephone', '345678901234567'),
    ->     ('smartphone', '456789012345678'),
    ->     ('smartphone', '567890123456789'),
    ->     ('featurephone', '678901234567890'),
    ->     ('smartphone', '789012345678901'),
    ->     ('smartphone', '890123456789012'),
    ->     ('featurephone', '901234567890123'),
    ->     ('smartphone', '012345678901234');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Add business users
mysql> INSERT INTO Business_user (Business_user_ID, Business_name, GSTIN) VALUES
    ->     (1, 'Zara Fashion Ltd', '29AADCU9604R1ZJ'),
    ->     (2, 'Levi Strauss India Pvt Ltd', '27AADCU9604R1ZJ'),
    ->     (3, 'McDonalds India Pvt Ltd', '33AADCU9604R1ZJ'),
    ->     (4, 'H&M India Pvt Ltd', '22AADCU9604R1ZJ');
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Add contracts for business users
mysql> INSERT INTO Contract (Contract_ID, Contract_Details, Start_date, end_date, Business_user_ID) VALUES
    ->     (1, 1001, '2023-01-01 00:00:00', '2024-12-31 23:59:59', 1),
    ->     (2, 1002, '2023-02-01 00:00:00', '2024-12-31 23:59:59', 1),
    ->     (3, 2001, '2023-01-15 00:00:00', '2024-12-31 23:59:59', 2),
    ->     (4, 3001, '2023-03-01 00:00:00', '2024-12-31 23:59:59', 3),
    ->     (5, 4001, '2023-04-01 00:00:00', '2024-12-31 23:59:59', 4);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> -- Add towers
mysql> INSERT INTO Towers (Tower_ID, Tower_Location, Tower_Capacity, Business_user_ID) VALUES
    ->     (1, 'Mumbai West', 1000, 1),
    ->     (2, 'Mumbai East', 1200, 1),
    ->     (3, 'Delhi North', 1500, 2),
    ->     (4, 'Delhi South', 1300, 2),
    ->     (5, 'Bangalore', 1400, 3),
    ->     (6, 'Chennai', 1100, 3),
    ->     (7, 'Kolkata', 1000, 4),
    ->     (8, 'Hyderabad', 1200, 4);
Query OK, 8 rows affected (0.00 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Add users
mysql> INSERT INTO User (User_name, User_phone_number, apt_name, street_name, city, pincode, state, User_Email, identity_proof, IMEI) VALUES
    ->     ('Rahul Kumar', '9876543210', 'Green Park', 'MG Road', 'Mumbai', '400001', 'Maharashtra', 'rahul@email.com', 12345, '1234567890012345'),
    ->     ('Priya Singh', '9876543211', 'Blue Bells', 'Park St', 'Delhi', '110001', 'Delhi', 'priya@email.com', 12346, '234567890123456'),
    ->     ('Amit Patel', '9876543212', 'Sea View', 'Marine Dr', 'Mumbai', '400002', 'Maharashtra', 'amit@email.com', 12347, '345678901234567'),
    ->     ('Deepa Gupta', '9876543213', 'Sky Tower', 'Ring Road', 'Bangalore', '560001', 'Karnataka', 'deepa@email.com', 12348, '456789012345678'),
    ->     ('Suresh Reddy', '9876543214', 'Palm Grove', 'Anna Salai', 'Chennai', '600001', 'Tamil Nadu', 'suresh@email.com', 12349, '567890123456789'),
    ->     ('Meera Iyer', '9876543215', 'River View', 'MG Road', 'Bangalore', '560002', 'Karnataka', 'meera@email.com', 12350, '678901234567890'),
    ->     ('Rajesh Shah', '9876543216', 'Sun City', 'SV Road', 'Mumbai', '400003', 'Maharashtra', 'rajesh@email.com', 12351, '789012345678901'),
    ->     ('Anita Sharma', '9876543217', 'Moon Tower', 'CP', 'Delhi', '110002', 'Delhi', 'anita@email.com', 12352, '890123456789012');
Query OK, 8 rows affected (0.00 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Add plan specifications
mysql> INSERT INTO plan_spec VALUES
    ->     (1, "Balanced", 200, 10, 80, 200, 60),
    ->     (2, "Entertainment", 300, 40, 60, 60, 80),
    ->     (3, "Talktime Add-on", 50, 0, 100, 0, 40),
    ->     (4, "Data Add-on", 50, 10, 0, 0, 30);
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

```
mysql>
mysql> -- Add user plans (subscriptions)
mysql> INSERT INTO User_plan (Plan_ID, IMEI, User_phone_number, date_of_purchase) VALUES
    ->     (1, '1234567890012345', '9876543210', '2024-10-15 10:00:00'),
    ->     (2, '2345678900123456', '9876543211', '2024-11-11 11:00:00'),
    ->     (1, '3456789001234567', '9876543212', '2024-10-17 12:00:00'),
    ->     (2, '4567890012345678', '9876543213', '2024-09-18 13:00:00'),
    ->     (3, '1234567890012345', '9876543210', '2024-11-12 14:00:00'),
    ->     (4, '5678901234456789', '9876543214', '2024-10-20 15:00:00'),
    ->     (1, '6789012345567890', '9876543215', '2023-11-13 16:00:00'),
    ->     (2, '7890123456678901', '9876543216', '2023-09-22 17:00:00'),
    ->     (3, '8901234567789012', '9876543217', '2024-10-23 18:00:00');
Query OK, 9 rows affected (0.00 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Add usage data
mysql> INSERT INTO uses (usage_quantum, IMEI, Tower_ID, time_stamp) VALUES
    ->     (100, '1234567890012345', 1, '2024-10-15 10:30:00'),
    ->     (150, '2345678900123456', 3, '2024-10-16 11:30:00'),
    ->     (200, '3456789001234567', 2, '2024-10-17 12:30:00'),
    ->     (350, '4567890012345678', 5, '2024-10-18 13:30:00'),
    ->     (200, '5678901234456789', 6, '2024-10-20 15:30:00'),
    ->     (250, '8901234567789012', 3, '2024-10-23 18:30:00'),
    ->     (400, '1234567890012345', 2, '2024-11-15 10:30:00'),
    ->     (350, '2345678900123456', 4, '2024-11-16 11:30:00'),
    ->     (450, '3456789001234567', 1, '2024-11-17 12:30:00'),
    ->     (500, '4567890012345678', 6, '2024-11-18 13:30:00');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

List of functionalities :

Register new device

Register user :

Register New User
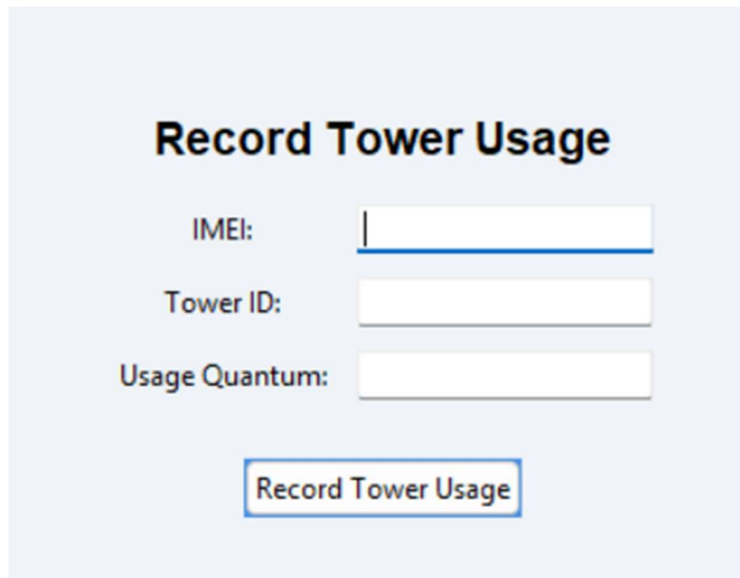
Name:	
Phone:	
Apartment:	
Street:	
City:	
Pincode:	
State:	
Email:	
Identity Proof(Number):	
IMEI:	

Register User

Purchase a new plan

Purchase New Plan

Plan ID:	
IMEI:	
Phone Number:	

Purchase Plan

Record Tower Usage

Triggers :

Trigger for preventing tower overload

```sql
-- trigger for preventing tower overload
DELIMITER //

CREATE TRIGGER prevent_tower_overload
BEFORE INSERT ON uses
FOR EACH ROW
BEGIN
    DECLARE current_load DECIMAL(10,2);
    DECLARE tower_capacity INT;

    -- Modified query to handle NULL cases with COALESCE
    SELECT
        COALESCE(
            CAST(COALESCE(SUM(usage_quantum), 0) AS DECIMAL(10,2)) /
            NULLIF(CAST(Towers.Tower_Capacity AS DECIMAL(10,2)), 0),
            0
        ) AS current_load,
        Towers.Tower_Capacity
    INTO current_load, tower_capacity
    FROM Towers
    LEFT JOIN uses ON uses.Tower_ID = Towers.Tower_ID
    WHERE Towers.Tower_ID = NEW.Tower_ID
    GROUP BY Towers.Tower_ID, Towers.Tower_Capacity;

    -- Check if tower exists
    IF tower_capacity IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Tower does not exist.';
    END IF;
```

```
    -- Check for overload
    IF (current_load + CAST(NEW.usage_quantum AS DECIMAL(10,2)) /
CAST(tower_capacity AS DECIMAL(10,2)) > 1) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Tower capacity exceeded, connection denied.';
    END IF;
END //
```

Trigger for checking whether a user's plan is valid or not :

```
-- Trigger to detect whether the user's plan is valid or not
DELIMITER |
CREATE TRIGGER check_plan_validity
BEFORE INSERT ON uses
FOR EACH ROW
BEGIN
    SELECT COUNT(*)
    INTO @valid_plans
    FROM User_plan, plan_spec
    WHERE
        User_plan.IMEI = NEW.IMEI and
        User_plan.Plan_ID = plan_spec.Plan_ID and
        DATE_ADD(User_plan.date_of_purchase, INTERVAL
plan_spec.validity_days DAY) > NEW.time_stamp;

    IF @valid_plans = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'User does not have any valid pack';
    END IF;
END |
DELIMITER ;
```

Function :

```
-- Function to find load on a given tower in a given time period
DELIMITER |
CREATE FUNCTION IF NOT EXISTS tower_load (tower_id INT, period_start
TIMESTAMP, period_end TIMESTAMP)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE usage_total INT;
    DECLARE capacity INT;

    SELECT COALESCE(SUM(usage_quantum), 0)
    INTO usage_total
    FROM uses
    WHERE
        uses.Tower_ID = tower_id AND
        uses.time_stamp > period_start AND
        uses.time_stamp < period_end;

    SELECT Tower_Capacity
    INTO capacity
    FROM Towers
    WHERE Towers.Tower_ID = tower_id;
```

```
    IF capacity IS NULL OR capacity = 0 THEN
        RETURN 0;
    END IF;

    RETURN CAST(usage_total AS DECIMAL(10,2)) / CAST(capacity AS
DECIMAL(10,2));
END |
DELIMITER ;
```

```
query = "SELECT tower_load(%s, %s, %s) as `load`"
```
invoked using python script

Procedure :

Fills the null device type to smartphone

```
DELIMITER $$

CREATE PROCEDURE UpdateNullDeviceType()
BEGIN
    UPDATE Device
    SET device_type = 'smartphone'
    WHERE device_type IS NULL;
END$$

DELIMITER ;
```

mysql> call UpdateNullDeviceType()

Queries :

Q1 : To retrieve plan popularity with names

```
CREATE VIEW plan_popularity_ AS
SELECT ps.Plan_name, COUNT(*) as subscribers
FROM User_plan up
JOIN plan_spec ps ON ps.Plan_ID = up.Plan_ID
GROUP BY ps.Plan_ID, ps.Plan_name
ORDER BY subscribers DESC;
```

Q2 : User distribution across cities

```
CREATE VIEW user_distribution_ AS
SELECT city, COUNT(*) as user_count
FROM User
GROUP BY city
ORDER BY user_count DESC;
```

Q3 : Distribution based on device type

```sql
CREATE VIEW device_distribution_ AS
SELECT device_type, COUNT(*) as device_count
FROM Device
GROUP BY device_type
```

Q4 : Average tower capacity by business users

```sql
CREATE VIEW avg_tower_capac_bu_ AS
SELECT bu.Business_name, AVG(t.Tower_Capacity) as avg_capacity
FROM Towers t
JOIN Business_user bu ON bu.Business_user_ID = t.Business_user_ID
GROUP BY bu.Business_user_ID, bu.Business_name;
```

Q5 : Currently all the running plans

```sql
CREATE VIEW running_plans_ AS
SELECT up.*
FROM User_plan up
JOIN plan_spec ps ON up.Plan_ID = ps.Plan_ID
WHERE DATE_ADD(up.date_of_purchase, INTERVAL ps.validity_days DAY) > NOW();
```

Q6: Users with multiple active plans

```sql
CREATE VIEW multi_plan_ AS
SELECT u.User_name, COUNT(DISTINCT up.Plan_ID) as active_plans
FROM User u
JOIN User_plan up ON u.IMEI = up.IMEI
JOIN running_plans_ rp ON up.IMEI = rp.IMEI
GROUP BY u.User_name
HAVING active_plans > 1;
```

Q7 : List all users

```python
def list_users(self):
    """
    List all users in the database
    """
    query = "SELECT User_name, User_phone_number FROM User"
    try:
        self.cursor.execute(query)
        return self.cursor.fetchall() or {}
    except mysql.connector.Error as err:
        raise Exception(f"Failed to fetch users: {err}")
```

Q8 : Get user details of a particular user

```python
def get_user_details(self, phone_number: str) -> Dict[str, Any]:
    """
    Get details of a specific user.

    Args:
        phone_number: User's phone number
    """
    try:
        query = "SELECT * FROM User WHERE User_phone_number = %s"
        self.cursor.execute(query, (phone_number,))
        return self.cursor.fetchone() or {}
    except mysql.connector.Error as err:
        raise Exception(f"Failed to fetch user details: {err}")
```

Q9 : Get user usage for a particular user

```python
def get_user_usage(self, phone_number):
    """
    Get a single user's usage statistics
    """

    query = """
        SELECT SUM(usage_quantum) AS total_usage, Tower_location
        FROM uses, User, Towers
        WHERE
            User_phone_number = %s
        GROUP BY Towers.Tower_ID
    """
    try:
        self.cursor.execute (query, (phone_number,))
        return self.cursor.fetchall()
    except mysql.connector.Error as err:
        raise Exception(f"Failed to get usage statistics: {err}")
```

Q 10 : Retrieves all the idle users

```python
def get_idle_users (self):
    """
    Get users who have no valid plan
    """
    query = """
        SELECT User_name, User_Email, t1.IMEI
        FROM (
            SELECT * FROM User_plan
            EXCEPT
            SELECT * FROM running_plans_
        ) AS t1, User
        WHERE
            User.IMEI = t1.IMEI AND
            User.User_phone_number = t1.User_phone_number
    """
    try:
```

```python
        self.cursor.execute (query)
        return self.cursor.fetchall()
    except mysql.connector.Error as err:
        raise Exception(f"Failed to get usage statistics: {err}")
```