# Problem Overview

Aisle offers **10 free Likes daily** to each user. These Likes must be **automatically reset every day at 12:00 PM**, **based on the user's local time**.
However, Aisle has:

- Millions of users
- Spanning across 150+ countries
- Operating in 24+ timezones

## Key Challenges:

- How to **automatically reset Likes** at the correct **local noon** for each user?
- How to make it **secure**, **scalable**, and **accurate**?

## Functional Requirements

- Each user gets **10 Likes per day**.
- Likes **reset at 12:00 PM (local time)** for each user.
- **Unused Likes do not carry over**.
- Reset must happen **automatically**, even if the user doesn't open the app.
- The system must be **cheat-proof** (users shouldn't manipulate device time).

## Technical Constraints

- Device time is **not trusted** (can be tampered).
- All logic must depend on **server-side time**.
- Should **scale** to handle millions of users across different timezones.
- Should support **real-time reset** when user opens the app after 12 PM.

## Scalable Architecture – Step-by-Step Approach

**Step 1: Capture and Store the User's Timezone**

- When a user **signs up** or **logs in**, capture their current timezone and store it in the backend database.

**iOS (Swift):**

- let timezone = TimeZone.current.identifier  // e.g., "Asia/Kolkata"

**Payload to backend:**

```
{
"timezone": "Asia/Kolkata"
}
```

Backend should persist this value alongside user profile data.

**Step 2: Group Users by Timezone**

- Instead of creating a reset job for each user (which is inefficient), **group users by their timezone**.
- Example: All users in "Asia/Kolkata" will be reset together at 12:00 PM IST.
- This gives us around **24 groups**, each with their own reset schedule.

**Step 3: Schedule Daily Cron Jobs by Timezone**

- Use your backend scheduler (like **AWS Lambda + EventBridge**, **Celery**, **Firebase Functions**, etc.) to run a daily job at **12 PM per timezone**.

**Example logic:**

```
// pseudo code for cron running at 12 PM Asia/Kolkata

if (currentServerTime.inTimezone("Asia/Kolkata").isNoon()) {

   resetLikes(forTimezone: "Asia/Kolkata")
```

```
}
```

**Backend Update Logic:**

```
// user document update

{

  "likesRemaining": 10,

  "lastResetDate": "2025-07-19"

}
```

Reset only if lastResetDate < today in that timezone.

**Step 4: Real-time Reset (When App is Opened)**

- If the user **opens the app after 12 PM** and hasn't been updated yet, reset on-the-fly.

**Backend Pseudo Logic:**

```
if (nowInUserTimezone() >= 12:00 PM && lastResetDate < today) {

    likesRemaining = 10;

    lastResetDate = today;

}
```

This fallback ensures even if cron misses someone, they get their Likes updated.

**Step 5: Block Device Time Cheats**

- Never rely on iOS local time like:

```
Date()
```

Always compare server time adjusted to user's timezone.

**Example:**

```
// convert UTC → user's timezone

let userLocalTime = utcTime.inTimezone(user.timezone)


if (userLocalTime.hour >= 12 && lastResetDate < today) {

    resetLikes()
```

## User Data Structure (Suggested)

```
{
  "userId": "u123",
  "timezone": "Asia/Kolkata",
  "likesRemaining": 3,
  "lastResetDate": "2025-07-18" }
```

## Summary: Why This Works

| Feature | Why it's Ideal |
|---|---|
| Timezone stored per user | Ensures accurate local reset |
| Grouped cron jobs (by zone) | Efficient scaling across millions of users |
| Server time only | Prevents time manipulation by users |
| Fallback real-time check | Enhances reliability and user experience |