Assignment: Delivery Management System

Background

You are engaged as a backend engineer at a logistics technology company. The business requires a Delivery Management System that manages the entire lifecycle of delivery orders — from order creation through tracking and final delivery. The system should be built using Go and must demonstrate the use of Redis and PostgreSQL for storage and state management. The ability to containerize the application using Docker is considered a bonus.

Business Scenario

Customers place delivery requests on a daily basis. Each request contains details such as:
- The requesting customer.
- The item(s) to be delivered.
- The current state of the order.

Once an order is created, it automatically transitions through multiple stages (e.g., created → dispatched → in transit → delivered). Customers should also be able to cancel their orders before delivery completion. Cancelled orders must stop progressing through their lifecycle.

Users interact with the system in two roles:
- Customer – can register, authenticate, create orders, and view/manage only their own orders.
- Administrator – can view and manage all orders in the system, including cancellation or overrides.

A critical feature is real-time order tracking. After an order is created, the system should asynchronously update its delivery status over time. Multiple orders can be in different states simultaneously, requiring proper concurrency control to ensure correctness and reliability.

Technical Requirements

1. Core Technologies
- Go for implementation.
- PostgreSQL for persistent data storage (users, orders, statuses).
- Redis for caching, real-time tracking, or queue-like mechanisms.

2. Concurrency Handling
- Simulate asynchronous order tracking using goroutines and concurrency primitives.
- Ensure thread-safe updates and consistency of order states.

3. Authentication & Authorization

- Provide secure user authentication.
- Enforce access control (customers vs. administrators).

4. Testing
- Automated tests must validate authentication, order creation, status progression, cancellation, and concurrent handling.

5. Bonus (Optional)
- Provide Dockerfile and (optionally) docker-compose.yml for containerized deployment.

Deliverables

- A working Delivery Management System with the described features.
- Source code in Go with a clean, modular structure (e.g., separating domain logic, persistence, and concurrency).
- Automated test suite covering critical functionality.
- README that includes:
  - Setup instructions.
  - Architectural overview and design decisions.
  - How to run the system and tests.

Evaluation Criteria

- Correctness – Does the system fulfill the functional requirements?
- Code Quality – Is the code idiomatic, modular, and maintainable?
- Concurrency – Are asynchronous operations implemented correctly and safely?
- Persistence – Effective use of PostgreSQL and Redis.
- Testing – Presence and quality of automated tests.
- Documentation – Clarity of instructions and system explanation.
- Bonus – Use of Docker for containerization.

This assignment is intentionally open-ended. You are expected to make your own design decisions regarding APIs, data models, concurrency management, and system organization. The goal is to showcase your ability to build a production-grade system in Go using the specified technologies.