Project Document – Car Rental System

Muhammad Aulia Firmansyah
mauliafirmansyah@uchicago.edu

System Overview

For this project, I propose a concept of a simple car rental system. This system aims to streamline the rental process for both customers and the car rental service provider. There are three key components of this systems, which are booking system, fleet management system, and payment system. In real life system, there should be many more systems involved, such as maintenance, insurance, and GPS tracking. However, for this project, I decided to define these three which is the most essential of this system.

The booking system allows customers to browse available cars based on their preferences such as car type and rental location. Users make bookings directly through the booking app. This system manages bookings and responds to customers while communicating with both fleet management system and payment management system.

The fleet management system oversees the entire cars inventory in multiple locations. It tracks and updates each car's availability, and location. When a vehicle is booked or the booking is cancelled, the system updates its status accordingly to avoid double-bookings. This is crucial to ensure the fleet meets customer demand effectively.

The payment system manages transactions for each booking. This system supports multiple payment methods, such as credit cards and digital wallets. Transaction records are saved upon payment completion. This system is integrated with the booking system to handle booking payments throughout the booking process.

Design Patterns

There are several design patterns which relates to this system:

1. Singleton: Defines a single fleet of cars.
2. Template Method: Standardizes the booking processes across different locations.
3. Composite: Organizes the cars into a hierarchical structure such as locations and fleet.
4. Strategy: Provides various ways for payments during booking process.
5. Iterator: Iterates through list of cars or booking orders.
6. Factory: Generates cars for each rental location.
7. State: Manages the status of booking orders, such as created, booked, paid, cancelled).

Furthermore, this system also utilizes many EIP patterns:

1. Channel Adapter: Accommodates booking requests from multiple apps into the system.
2. Message Translator: Translates booking request into car or payment request.
3. Content-Based Router: Routes booking orders based on rental location.
4. Request-Reply: Used for synchronous interactions, such as booking or payment confirmations.
5. Event-Driven Consumer: Triggers actions based on user's interaction with the app.

6. Multicast: Sends cancellation messages to both booking app and fleet management system at the same time.
7. Dead Letter Channel: Handles failed bookings due to various reasons, such as invalid booking or unavailable car.
8. Competing Consumer: Implements multiple processing instances to handle concurrent bookings.

Diagram