**Problem Requirements:**

The aim of this exercise is to implement a vehicle repair ordering API so that claim officer can request repair for the vehicle to the repairer. Claim officers can use this API to place repair orders, view existing repair order and cancel repair order before it is delivered.

A basic Repair order has the following attributes:

• orderId – A unique ID for identifying a repair order

• Start date time – The date and time when repair will start

• Duration – The duration of the repair order (e.g. Duration of 3 hours means , repair activity will take 3 hours from the start date time).

As part of this exercise, we will need to:

1. Design and implement a REST API to accept new repair orders from a claim officer.

2. Design and implement an API for cancelling an existing order if it hasn’t been delivered.

3. Design and implement an API so claim officers can query existing orders. When querying orders, the claim officers should be able to see the status of each order.

Possible status of a repair order:

• Requested – “Order has been placed but not yet delivered.”

• InProgress – “Order is being delivered right now.”

• Delivered – “Order has been delivered.”

• Cancelled – “Order was cancelled before delivery.”

4. The API must ensure the repair orders for a vehicle do not overlap.

• For example, if Repairer X already has an order for 30 Jan 2019 starting at 8am with a 3 hours duration, it should not allow repairer X to place an order starting at

10am on the same day.

5. To simulate repairing delivery, our application should output a line each time the status of a repair order changes. This include –

• When a new repair order is placed;

• When a repair order starts (Start date time of the order);

• When a repair order is delivered (i.e. start date time + duration);

• When a repair order is cancelled;

• As an example:

i. Assume it is currently 10am, and a repair order is created with a start time of 12pm and finishing at 2pm.

ii. At 10am, when the order is placed, our application’s log should print “New repair order for Repairer xyz created”

iii. At 12pm, log should print “Repair delivery for vehicle at repairer xyz started.”

iv. At 2pm, log should print “Repair delivery for vehicle at repairer xyz stopped.”

**Why this problem?**

This problem represents typical and generic scenario in enterprise level applications across various domains. Although on the face of it, it looks like straightforward problem, but it has tricky scenarios like time overlap, dealing with dates, database polling as a part of solution. I choose this to showcase my abilities for API design, writing production ready, maintainable, extensible and readable code.

**Design and Approach**

Web Oriented architecture is used to provide basic CRUD operations to the claim user. For database polling operation, solution has been implemented using spring integration. Application has been designed keeping SOLID principles in mind.

**Webservice Architecture**

**H2 in memory db**

**Persistence layer (JPA)**

Business Layer

Spring service

**REST API**

Browser

**Data Integration architecture**

DB

Inbound channel

**Patterns used**

* Designed by using WOA (web oriented architecture)
* DAO Pattern (data access object) is used to separate low level data accessing API or operations from high level business services (spring data repositories)
* DTO Pattern (Data transfer object) this has been used to decouple presentation layer object and entity object. We can then control what and how much data we want to transfer and expose to clients.
* DI (dependency injection ) spring beans