Preface

About this Document

This document describes the architecture of Careem Shipping Model for B2B Customers.

**Contents**

[1 Introduction 3](#_Toc475804875)

[2 Technology Stack 3](#_Toc475804876)

[3 Architecture 4](#_Toc475804877)

[3.1 Customer Set Up 5](#_Toc475804878)

[3.2 Ordering App 5](#_Toc475804879)

[3.3 Delivery App 6](#_Toc475804880)

[3.3.1 Life cycle of an DELIVERY 6](#_Toc475804881)

[3.4 Notification Application 7](#_Toc475804882)

# Introduction

This is high level design document to build a platform that can work with commerce companies who need a shipment delivery service.

The main features of this architecture is to provide solution for

* Customer Set Up
* Decision Making Unit
* Customer specific prices
* Payment on credit sales
* Managing Vendors/Partners
* Smart shipments (i.e. truckloads)
* Reoccurring purchases
* Long lasting relationship between customer and manufacturer
* Buying as part of the job
* Buyers as part of an organization with a relationship defined by a contract, terms and conditions

# Technology Stack

Technology stack which I shall be using in the application:

* Spring Boot Application
* Spring REST Controller
* JPA-Hibernate
* MySql Database
* Spring Basic Authentication (would like to implement Spring Oauth 2 in real time)
* Maven
* FTL : Free marker template for generating notification messages

# Architecture

To solve the above problems this system is divided into four decoupled sub systems. The sub systems can be deployed/developed parallelly in same/different servers and will communicate through each other via REST APIs.

## Customer Set Up

A customer needs to be set up. A customer exists under an **entity**. Eg: Flipkart. This entity can have multiple offices across different locations. Each office can have individual shipping requirements. Each office act as a **“Customer”**. The customer needs to be validated via proper KYC checks. A credit limit should be assigned to each customer and a rating is also given at Customer Level.

REST APIs

1. Create Entity
2. Edit Entity
3. Deactivate an Entity
4. Define Customers in Entity
5. Get the Customer Id
6. Deactivate a customer

## Ordering App

The Ordering App allows the ACTIVE customers to enter the order details. Each Order detail will contain information about a order in terms of pick up location, drop location, size, quantity.

Every Order will have its lifecycle:

1. DRAFT: A customer logins and creates an order. He/she gets the calculated charges based on order details.
2. PENDING: A user submits a proposal (order detail) which allows him to enter his/her quotation. The Order remains in PENDING status.
3. APPROVED: The backend system user verifies the proposal. This proposal can be again sent to DRAFT if it requires further input from the customer.
4. REJECTED: The backend system user can REJECT an invalid order details. That would be considered end of that order lifecycle.
5. CLOSED: Once the order is completed end to end.

DRAFT

PENDING

APPROVED

CLOSED

REJECTED

The REST APIS will be to written for :

1. Creating an order
2. Modifying an order
3. Getting order list
4. Getting an order
5. Approve an Order: This will fire the Delivery Application.
6. Cloning an Oder: A new order will be cloned in DRAFT stage. This feature enables re-ordering with editing features.

## Delivery App

The delivery app is the heart of the application. The delivery app will look into the transportation resources to find the most cost-effective and reliable way to deliver shipments.

It accepts an order and sets up the delivery workflow for the Order. The delivery route is broken down for an Order. One Bulk order can involve multiple mode of delivery.

One bulk delivery from location A to C may involve

1. Air Transport from A to B
2. Transport via trucks from B to C

The delivery model should also allow quotation by the partners/vendors to take the entire delivery or in parts.

### Life cycle of an DELIVERY

Once an **Order** is approved by the backend executive, a delivery with status “DRAFT” is created. The delivery is broken down into routes and quotations from vendors/partners are fetched in time bound manner. Ideally every quotation can have its own lifecycle but for simplicity I am assuming every quotation is final. Once the quotation is finalized, the status of the delivery changes to “IN-PROCESS”.

Each Delivery has status as

1. DRAFT
2. IN-PROCESS: Multiple Routes will complete their lifecycles in one Delivery.
3. COMPLETED: If the delivery completes the destination successfully.

The system will break each delivery into one or multiple “**route**”.

Each route has status as

1. CHECKED IN: The product is verified and approved by the owner.
2. IN-PROCESS: The product is in transit.
3. CHECHKED OUT: The product has completed the route and approved by the owner.

The REST APIs for delivery app are

1. Create a delivery : The delivery is created by the system in DRAFT status.
2. Get delivery List
3. Fetch quotation for routes
4. Finalize quotation
5. Update a delivery
6. Get Routes for a delivery
7. Update Route Status

## Notification Application

Since the transaction involves multiple parties which are represented by executives of different companies. The system allows generation of

1. Contracts: For each order a contract is generated and sent to respective customer.
2. Invoices and bills: The bills and invoices needs to be generated for every order.
3. The contract and invoices: The vendors and participating partners need contract and invoices generated to them.
4. Notification Message: A notification message should be send to customer when the product gets delivered. The preferred mode of communication can be defaulted to email or can be selected while defining customer.

# 