Backend Challenge

Background

Liberet spots a key enhancement in the UI design for the users in the shopping cart module. Now Liberet needs new services for shopping carts. The service must include a way to store and persist data, a way to track user patterns and an API design to interact with the service. The main purpose of the service will be to calculate the total amount to be charged to the user, create the historical data for the user profile and to have the final data to be used by another future service.
The service will aim to support 10k active users, in average the user will complete a shopping cart per day with an average of 5 products per Shopping Cart.

Business decisions:

1. A **user** can only have one active **shopping cart**. The shopping cart is removed only if **it's completed** by the user or **all the products** in the shopping cart are **removed**.
2. A **shopping cart** can have any amount of **products**, as the user wants.
3. The shopping cart service will group all the products in **orders**.
4. An **order** is defined by the unique combination of:
    a. Service Day -> MUST be handled as string in ISO utc -> https://en.wikipedia.org/wiki/ISO_8601
    b. Service Schedule: Liberet will only handle 3 delivery times
        i. 9:00 - 11:00
        ii. 11:00 - 13:00
        iii. 13:00 - 15:00
    c. Supplier: handled as string
    d. Delivery Type: Liberet only handle 3 delivery types:
        i. Pickup
        ii. Reservation
        iii. Delivery
5. An **order** can have any amount of products,
6. The **order** cost is the sum of all products in the order.
7. Each **order** must add an extra amount to the total with the concept of "**Delivery Fee**"
8. The Delivery fee is calculated following these rules:
    a. **DELIVERY FEE** only APPLY for delivery type "**DELIVERY**"
    b. Express Order: an order which is requested less than 4 hours prior its delivery
    c. Scheduled order: an order which is requested with more than 4 hours prior its delivery
    d. Express Order = 28MXN DELIVERY FEE
    e. Schedule Order = 19MXN DELIVERY FEE
    f. Express orders with a cost under 60 MXN (sum of all products in the order) = 38MXN DELIVERY FEE

g. Scheduled Orders with a cost under than 60MXN (sum of all products in the order) = 28MXN DELIVERY FEE
h. Any order with at least one product with a cost higher or equal to 2000MXN = 199MXN DELIVERY FEE
i. All costs and delivery fees are displayed with 2 decimal places, the currency is MXN.

Examples and some data:
https://docs.google.com/spreadsheets/d/115A4_LtnZJ-QeD25GCeOBi-Rk4Hg_fDE5lEu6OSInic/edit?usp=sharing

NOTES:
- You choose how to manage and model your data. Prepare to explain your choices.
- Ids can be handled as you prefer, there is no restriction.
- You can use any framework, ORM or tool of your preference.

Endpoints:

- POST /shoppingCarts/{**userId**}/add
  Add a product to a current active shoppingCart. If the user doesn't have an active it will create a new reference. **NOTE**: user can only have 1 ACTIVE shoppingCart
  Body:

```
{
    productId: “string”
    Amount: number,
    serviceDate: “isoUtcDate”
    serviceSchedule: “isoUtcDate”
    supplier: “string”
    deliveryType: “string”
    productCost: “string”
}
```

  Response 200:

```
{
    orderId:””
}
```

- DEL /shoppingCart/{**userId**}/remove/{**orderId**}/{**productId**}
  Response: 204

- POST /shoppingCart/{**userId**}/complete
  Completes a shopping Cart, if a user adds another product it will be a new shopping cart.
  Response 204

- GET /shoppingCart/{**userId**}
  Obtain current active shopping cart in the following forma
  Response: 200

```
{
    orders:[
        {
            orderId:string,
            products:[
                {
                    productId:”string”,
                    amount:number
                    productPrice:number
                },...
            ],
            orderCost:number,
            orderDeliveryFee:number
        },...
    ],
    totalDeliveryFee: float,
    totalOrdersCost: float,
    totalCost: float
}
```

Time and deliver

You will have 1 day to complete the challenge. Please share the repository with the generated code at the time provided in the email.

Extras

1. Add an authorization technique to the API. Login, Logout and use accessTokens to obtain the userId instead of the url path
2. Upload your API to a cloud solution. Provide the Link.
3. Add Automatic Tests for your endpoints.

Presentation

1. Demo presentation of the API, it can be local, share an ngrok link or a host.
2. You will have 20 minutes to show how it works, and how you develop your work. There will be another 10 minutes of Q/A.