SKY IT Coding Challenge

Assumptions:

1. The mileage of each vehicle will be updated only once a day.
2. The mileage for the preceding day(s) is equal to or greater than the previous day's mileage.
3. No authentication(permission), anyone can access these APIs.

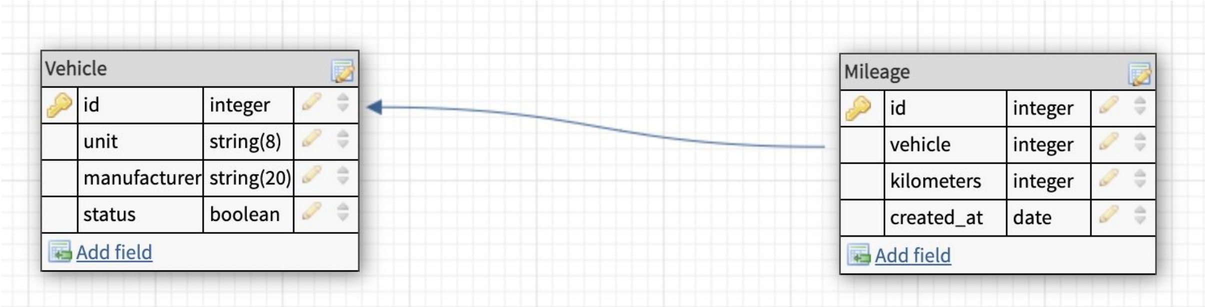
Data Available in Database:

2 Vehicle are available in Vehicle Table
Mileage for vehicle 1 is available from Jan 03 to Jan 10(current date of server).
Mileage for vehicle 2 is available from Jan 09 to Jan 10.
You have to add the mileage after Jan 10 if you're testing it on Jan 11. Otherwise, till now data is up to date and ready for testing.

Code Explanation:

Models:

Each class in the models.py file represents a table in the database.
There are two models in the models.py file one is for vehicles and one is for mileage.
The relation of the vehicle to the mileage model is one to many. So, each day end-user selects the vehicle and updates the mileage.



Vehicle	
id	integer
unit	string(8)
manufacturer	string(20)
status	boolean

Mileage	
id	integer
vehicle	integer
kilometers	integer
created_at	date

Views:

There are 4 classes in the Views.py file which retrieve users' requests and return responses according to the request.

Note: Views 1 and 2 (VehicleView and UpdateVehicleView) are the additional views I preferred to add and update vehicles by RestApi instead of the admin panel.

1. **VehicleView**
   This is a class-based view that lists all the vehicles and it can also create a new vehicle.
2. **UpdateVehicleView**
   This class-based view takes the id of the vehicle as an argument in URL and retrieves the object against the given id by the end-user and can update that object.
3. **CreateMileageView**
   This class-based view is used to create mileage for vehicles every day and the post method is overridden in this view because according to our assumption vehicle mileage will be updated only once so that's why if the end-user try to update the mileage again on the same day then the post function return error response that data is added for today.

4. **CalculateMileageView**

   This view takes the id of the vehicle and date as an argument in the URL and search object against that date and calculates the distance covered between the user-entered date and the current date.

   So if the server date is changed and the current date mileage of a vehicle is not updated for any reason then the mileage covered on the previous date will be returned.

Serializers:

Serializer is used to convert the queryset into a format that is understandable by the Frontend.

There are two serializers in serializer.py one is for the Vehicle model and one is Mileage Model. To show the mileage of the vehicle when rendering the Vehicle I used nested Serializer by adding the Mileage serializer in Vehicle Serializer.