Traffic Management System 1.0

HONGTAO AND VINCENT →

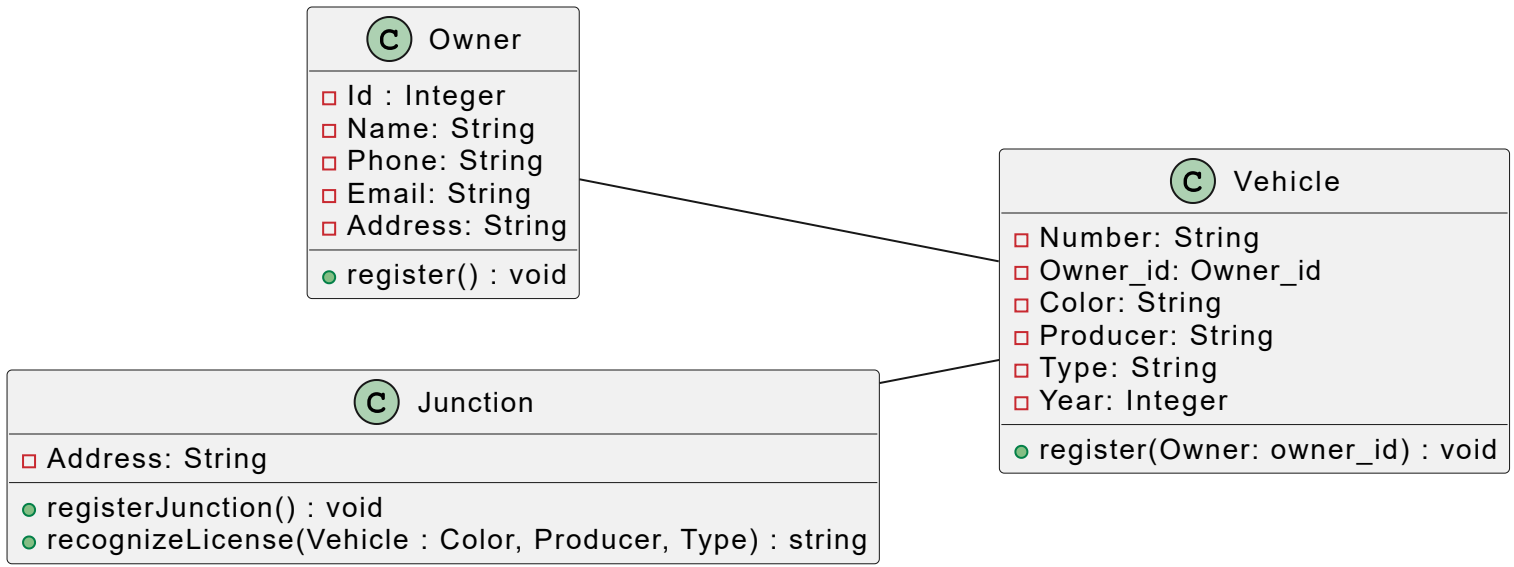
29/01/2024

Table of contents

Requirements

`I`	Register Vehicle
`II`	Simulate Recognizing License

Class Diagram



We defined three classes: Junction, Vehicle, Owner.

We use phone number to distinguish each owner.

We use owner_id to connect each vehicle to its owner.

We use "color + producer + type" to recognize vehicles.

Register Vehicle Highlights

Django view function for handling a POST request to register a vehicle.

- Checks if required fields exist and if their types are correct.
- Ensures that the plate number, color, producer, and type are strings and do not contain numbers or special characters.
- Verifies that the year is an integer.

Rigester Vehicle Details

Show how to define the details of cars, See more.

`nubmer`	models.CharField(max_length=10)
`owner_id`	database allocation id
`color`	models.CharField(max_length=10)
`producer`	models.CharField(max_length=10)
`type`	models.CharField(max_length=10)
`year`	models.IntegerField

Code

Check code here directly or check full project.

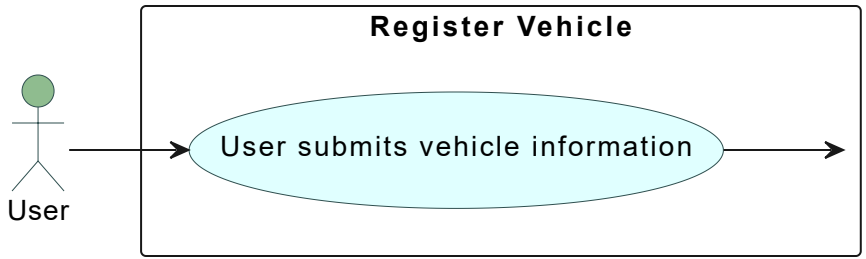
```
class Vehicle(models.Model):
    Number = models.CharField(max_length=10)
    Owner = models.ForeignKey(Owner, on_delete=models.SET_
    Color = models.CharField(max_length=10)
    Producer = models.CharField(max_length=10)
    Type = models.CharField(max_length=10)
    Year = models.IntegerField()
    def __str__(self) -> str:
        return self.Number
```

```
@csrf_exempt
def register_vehicle(request):
    if request.method == 'POST':
        try:
            data = json.loads(request.body.decode("utf-8")

            number = data.get("number")
            owner_id = data.get("owner_id")
            color = data.get("color")
            producer = data.get("producer")
            type = data.get("type")
            year = data.get("year")

            if not number or number == "":
                return JsonResponse({'error': "Plate_numbe
            if not isinstance(number, str):
                return JsonResponse({'error': "Plate_numbe
```

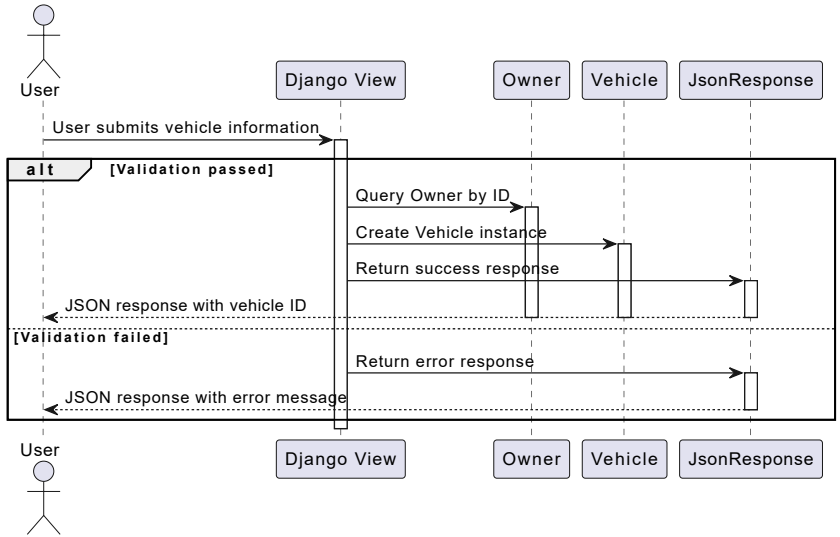
Register_Vehicle: Usecase Diagram



Use Case Diagram:

The interaction between the "User" and the "Register Vehicle" use case. The user submits vehicle information, which triggers the "Register Vehicle" use case.

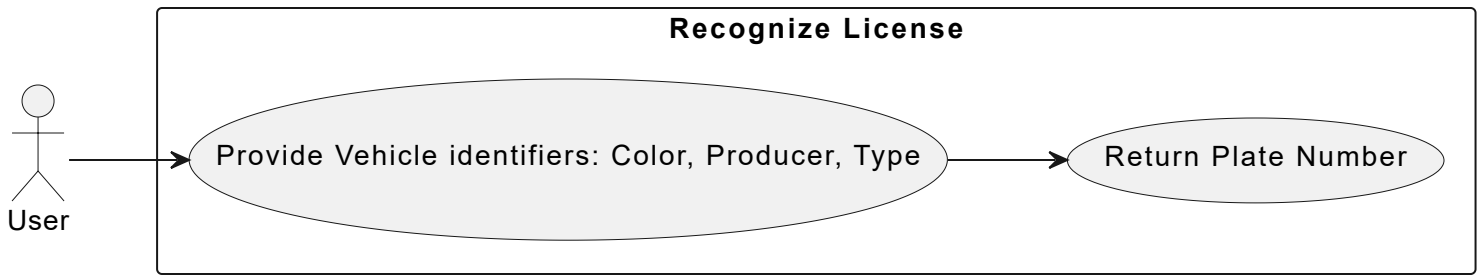
Register_Vehicle: Sequence Diagram



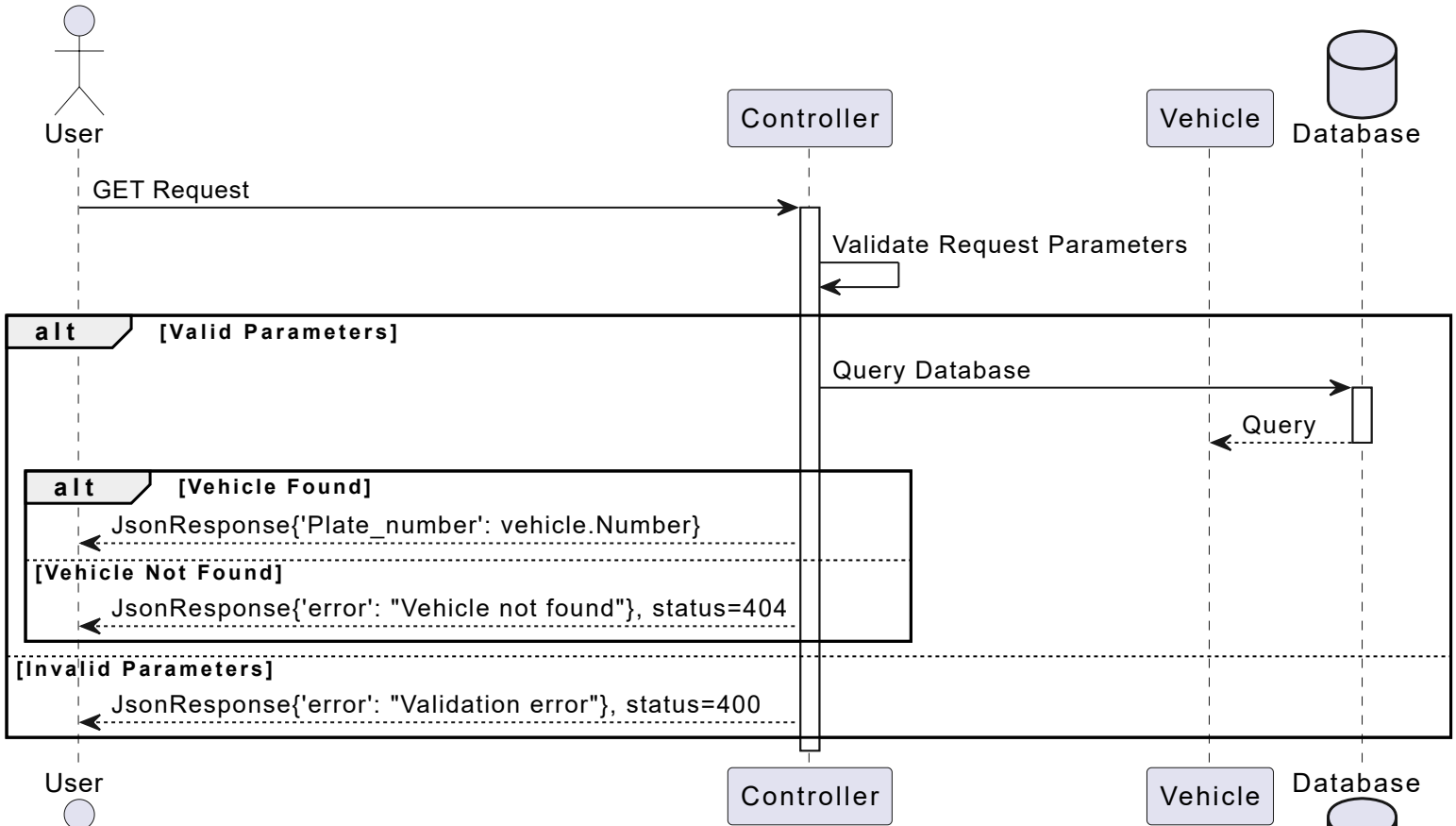
The "User" interacts with the "Django View", "Owner", "Vehicle", and "JsonResponse" during vehicle registration. The user submits vehicle info, which the view processes. If it’s valid, the view creates a vehicle and sends a success response; if not, it sends an error response.

Part II Recognize Vehicle Plate Number

Usecase Diagram



Recognize_Vehicle: Sequence Diagram



Thank you for watching

GitHub