Parking Lot API Contract

Api Definitions

BaseApi: `your-domain`

ParkingLot

Create Parking Lot

base: `/api/ParkingLots`
verb: `post`
payload(body):

```json
{
        "id": "65e72adb1a811501c45afd72",
        "capacity": 10
}
```

response:

```json
{
        "isSuccess": true,
        "response": {
                "id": "65e72adb1a811501c45afd72",
                "capacity": 10,
                "isActive": true
        }
}
```

Constraints:

i. Capacity should not be higher than 2000 or lower than 0
ii. Input validations
iii. id should be a hexadecimal string ideally 24 in length

Parking

park

url: `/api/Parkings`
verb: `post`
Payload:

```
{
        "parkingLotId": "65e72adb1a811501c45afd72",
        "registrationNumber": "MH12A1234",
        "color": "YELLOW"
}
```

Response:

```
{
        "isSuccess": true,
        "response": {
                "slotNumber": 1,
                "status": "PARKED"
        }
}
```

Constraints:

i. `registrationNumber` should be a valid registration number
    i. For now you can consider that each state will have a maximum of 20 districts.
    ii. The leading alphabet after the district code should only be one in length.
    iii. Total length should be 9.
ii. `status` will be limited to `PARKED` and `LEFT`
iii. `parkingLotId` should correspond an active parkingLot
iv. Only the following colored cars are allowed in the parking lot
    - RED
    - GREEN
    - BLUE
    - BLACK
    - WHITE
    - YELLOW
    - ORANGE

Leave

url: `/api/Parkings`
verb: `delete`
Payload:

```
{
        "parkingLotId": "65e72adb1a811501c45afd72",
        "registrationNumber": "MH12A1234"
}
```

Response:

```
{
        "isSuccess": true,
        "response": {
                "slotNumber": 1,
                "registrationNumber": "MH12A1234",
                "status": "LEFT"
        }
}
```

Constraints:

- Same constraints as create parking

Registration Number by Color

url: `api/Parkings?color=WHITE&parkingLotId=65e72adb1a811501c45afd72`
verb: `get`
queryParams: `color` `parkingLotId`
Response:

```
{
        "isSuccess": true,
        "response": {
                "registrations": [
                        {
                                "color": "BLUE",
                                "registrationNumber": "MH15A4567"
                        },
                        {
                                "color": "BLUE",
                                "registrationNumber": "MH13K4567"
                        }
                ]
        }
}
```

Constraints:

i. `registrations` array should follow natural ordering based on the db insertion
ii. If cars of specified color (say WHITE) is not available then your api should respond with the following error:

```
{
        "isSuccess": false,
        "error": {
                "reason": "No car found with color WHITE"
        }
}
```

Slot

Fetch slots by color

url: `/api/Slots?color=BLACK&parkingLotId=65e72adb1a811501c45afd72`
verb: `get`
queryParams: `color` `parkingLotId`
Response:

```
{
        "isSuccess": true,
        "response": {
                "slots": [
                        {
                                "color": "BLACK",
                                "slotNumber": 2
                        },
                        {
                                "color": "BLACK",
                                "slotNumber": 3
                        }
                ]
        }
}
```

Constraints:

i. `slots` should be served in increasing order by `slotNumber`
ii. If an invalid color is provided to the api to then your api should respond with

```
{
        "isSuccess": false,
        "error": {
                "reason": "Invalid Color"
        }
}
```

Notes

- The construct of the completer url would follow `BaseApi` + `base` for all individual apis.
- Before each round of evaluation please purge all the relevant tables to avoid data corruption. The evaluation algorithm expects a clean state for all tables.
- Renaming of fields in the contract might lead to inconsistent evaluation.
- Some of the corner cases are only mentioned in the test cases, a complete report of which you would receive after each evaluation.
- Your apis should always respond with status code 200 for any of the specified edge cases, with the response structure adhering to the following standard:

```
{
        "isSuccess": false,
        "error": {
                "reason": ""
        }
}
```