Backend Engineer Test Result

Devi Handevi | devihandevi@gmail.com | +62 857 9540 3931

2020/12/03 - 2020/12/05

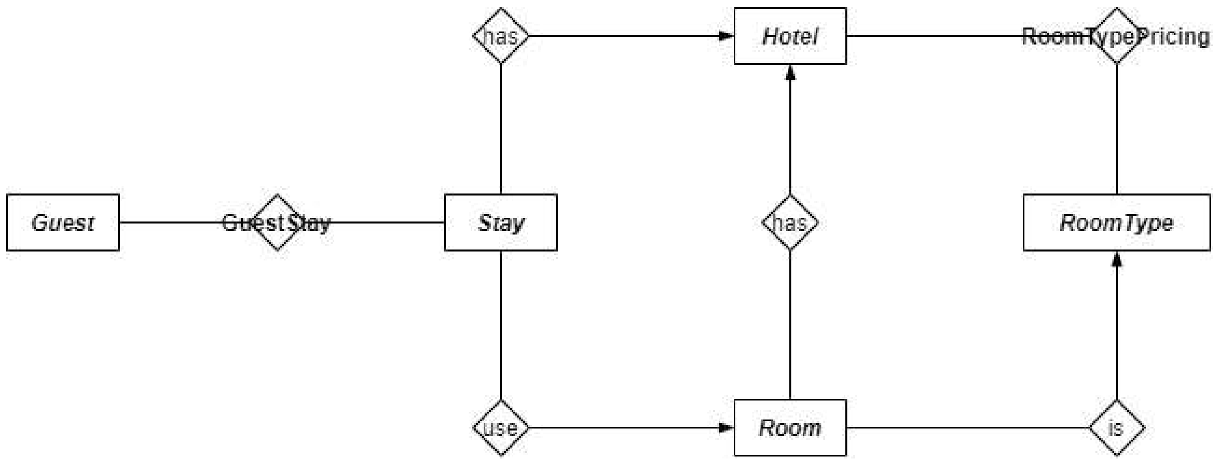
1. Hotel Management System

With minimum requirements:

1. Manage multiple hotels
2. Manage dynamic room pricing
3. Manage room pricing up to one next year

Database Design

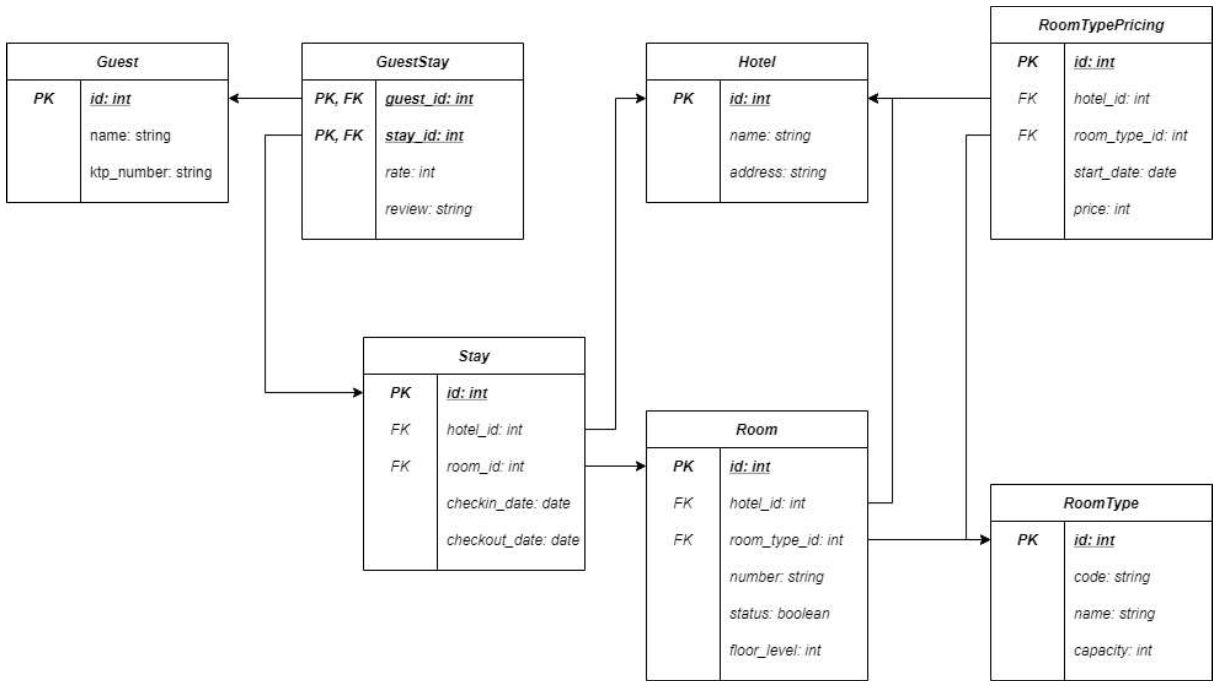
Entity Relationship Diagram



Notes:

1. Each guest in each room can have their own rating and review.
2. Reservations, Customers (who booked), Billing, and other entities are not used in this case, but should exist in the actual system.
3. The entity relationships with many-to-many conditions (no-arrow to no-arrow) will have a relationship table.

Table Design



Dynamic Pricing

Default

The design allows us to save different pricing for different:

1. Dates
2. Room Type
3. Hotel

The pricing records save the latest (or future) pricing for the room type in the specified hotel. So it will be able to manage pricing any time for any location for any room type.

Advanced

For more dynamic pricing, it should not just be designed in the entities alone. If it is needed, we could add functionality in the backend to:

1. Adjust pricing based on the demand. If it is in high season, the price could go up dynamically and automatically.
2. Adjust pricing based on the room availability.
3. Adjust pricing based on the all-time or seasonal room rate or even review (needs more advanced language processing to extract the positivity of a review)
4. Adjust pricing based on how often loyal guests have stayed.

But the pricing calculation should be researched and modeled first.

2. Hotel Room Search Availability

Some notes while I was doing the task:

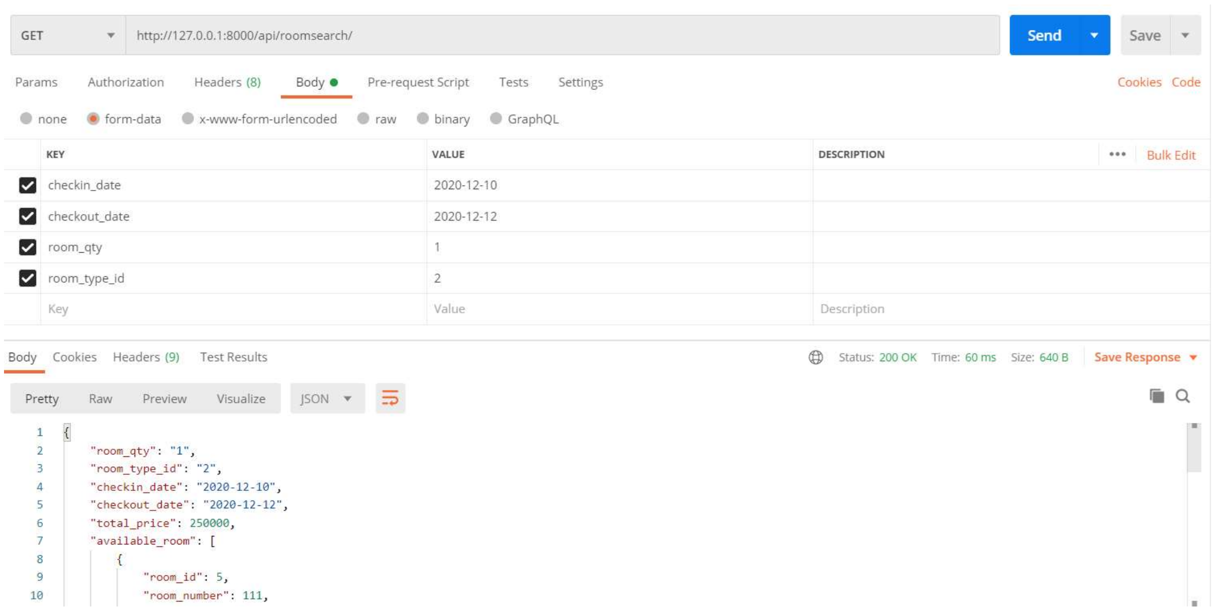
1. I assumed that the second task was more focused on the search functionality, not on the database model. For that reason, I am not too detailed on the data validation and assume that they are all valid. For example:
    a. Check-in date should be before check-out date
    b. Room selection in the Stay model should be from the reservation hotel
    c. Date in the Stay model should be between check-in and check-out date
    d. Unique room numbers
2. I made an assumption that the `room_id` in the Stay is the same as the `room_id` in the StayRoom and I apply only in StayRoom instead.
3. I did not add `order_id` to the Registration model because of the lack of information about the Order model and that the Order model was not needed to finish the task.
4. I assume that the Price saves price changes with the date start it is applied.
5. I assumed that the given table structure cannot be changed (have to be used as is). In my humble opinion, there are some things that could be changed to make the data management better, e.g. the Stay and StayRoom are merged into one table and it would still have the same functionality. However, it is only within my assumptions context, I do not know the overall considerations that made the decision.

Example Using Postman

URL: http://127.0.0.1:8000/api/roomsearch/

Notes: Using different cases from the assignment cases but the program is able to handle the given cases.

Case 1: Overlap first days to other people



http://127.0.0.1:8000/api/roomsearch/

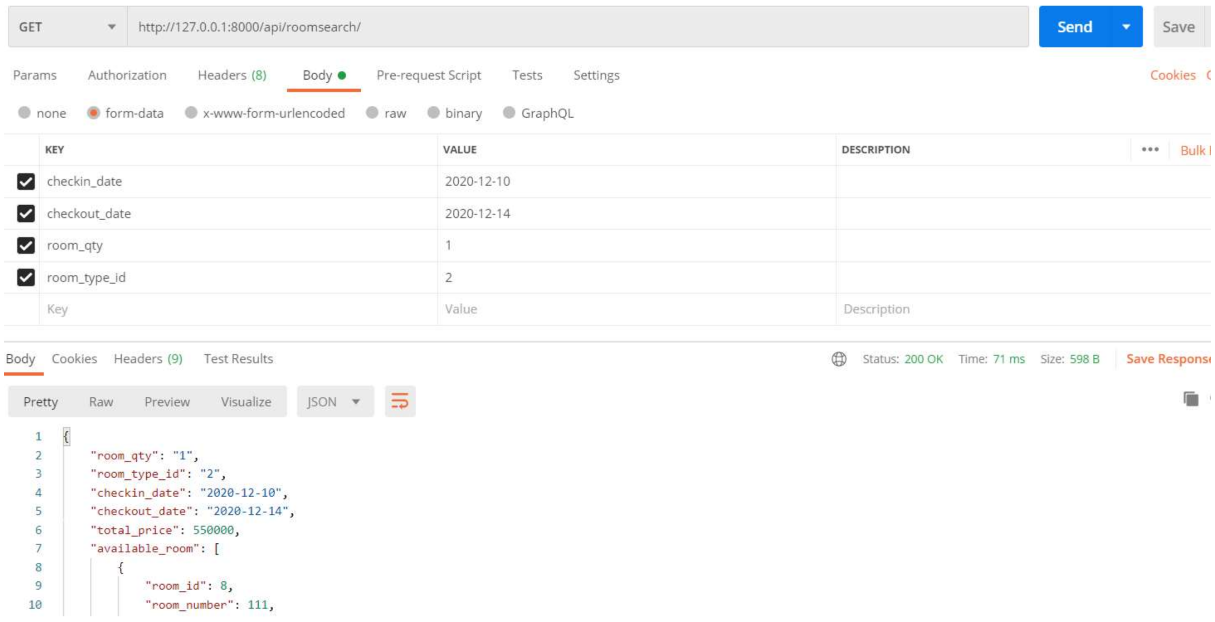
KEY	VALUE	DESCRIPTION
checkin_date	2020-12-10	
checkout_date	2020-12-12	
room_qty	1	
room_type_id	2	
Key	Value	Description

Status: 200 OK Time: 60 ms Size: 640 B

```
{
    "room_qty": "1",
    "room_type_id": "2",
    "checkin_date": "2020-12-10",
    "checkout_date": "2020-12-12",
    "total_price": 250000,
    "available_room": [
        {
            "room_id": 5,
            "room_number": 111,
```

```
{
    "room_qty": "1",
    "room_type_id": "2",
    "checkin_date": "2020-12-10",
    "checkout_date": "2020-12-12",
    "total_price": 250000,
    "available_room": [
        {
            "room_id": 5,
            "room_number": 111,
            "price": [
                {
                    "date": "2020-12-10",
                    "price": 100000
                },
                {
                    "date": "2020-12-11",
                    "price": 150000
                }
            ]
        },
        {
            "room_id": 8,
            "room_number": 111,
            "price": [
                {
                    "date": "2020-12-10",
                    "price": 100000
                },
                {
                    "date": "2020-12-11",
                    "price": 150000
                }
            ]
        }
    ]
}
```

Case 2: Overlap first days and last days to other people



http://127.0.0.1:8000/api/roomsearch/

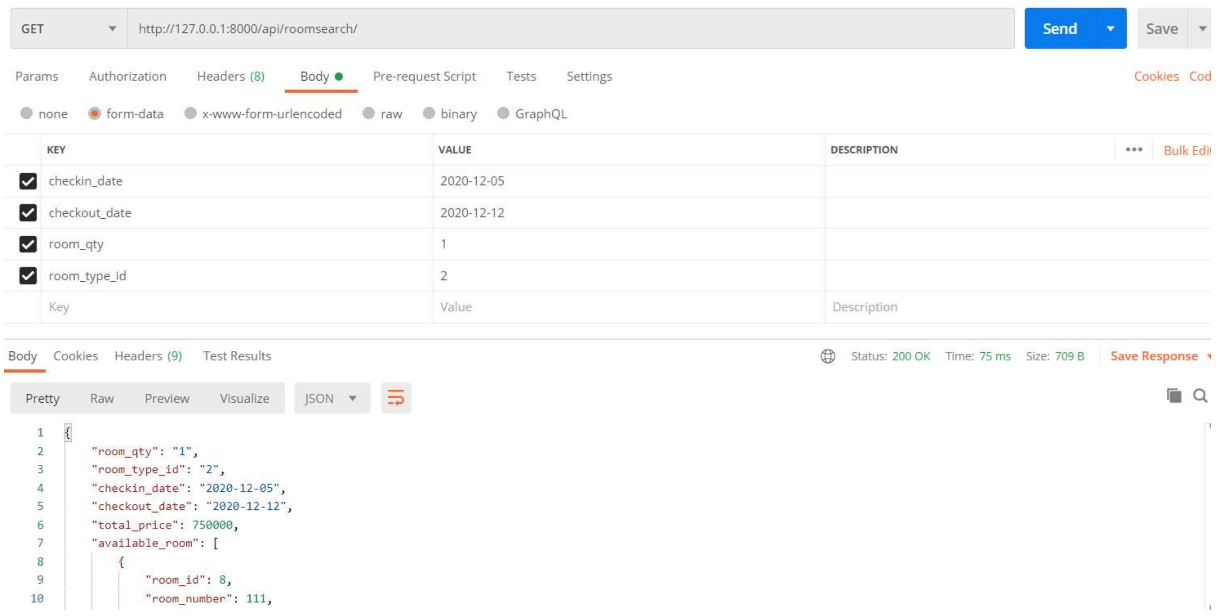
	KEY	VALUE	DESCRIPTION
✓	checkin_date	2020-12-10	
✓	checkout_date	2020-12-14	
✓	room_qty	1	
✓	room_type_id	2	
	Key	Value	Description

Status: 200 OK Time: 71 ms Size: 598 B

```
{
    "room_qty": "1",
    "room_type_id": "2",
    "checkin_date": "2020-12-10",
    "checkout_date": "2020-12-14",
    "total_price": 550000,
    "available_room": [
        {
            "room_id": 8,
            "room_number": 111,
```

```
{
    "room_qty": "1",
    "room_type_id": "2",
    "checkin_date": "2020-12-10",
    "checkout_date": "2020-12-14",
    "total_price": 550000,
    "available_room": [
        {
            "room_id": 8,
            "room_number": 111,
            "price": [
                {
                    "date": "2020-12-10",
                    "price": 100000
                },
                {
                    "date": "2020-12-11",
                    "price": 150000
                },
                {
                    "date": "2020-12-12",
                    "price": 150000
                },
                {
                    "date": "2020-12-13",
                    "price": 150000
                }
            ]
        }
    ]
}
```

Case 3: Overlap middle to other people



http://127.0.0.1:8000/api/roomsearch/

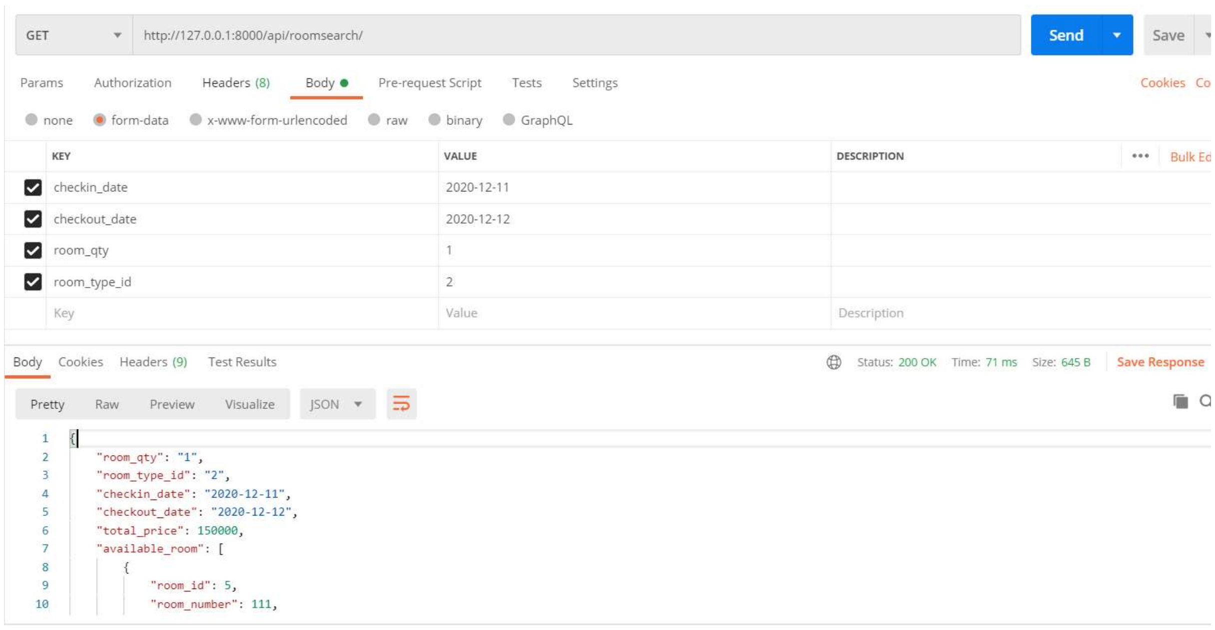
KEY	VALUE	DESCRIPTION
✓ checkin_date	2020-12-05	
✓ checkout_date	2020-12-12	
✓ room_qty	1	
✓ room_type_id	2	
Key	Value	Description

Status: 200 OK Time: 75 ms Size: 709 B

```
{
    "room_qty": "1",
    "room_type_id": "2",
    "checkin_date": "2020-12-05",
    "checkout_date": "2020-12-12",
    "total_price": 750000,
    "available_room": [
        {
            "room_id": 8,
            "room_number": 111,
```

```
{
    "room_qty": "1",
    "room_type_id": "2",
    "checkin_date": "2020-12-05",
    "checkout_date": "2020-12-12",
    "total_price": 750000,
    "available_room": [
        {
            "room_id": 8,
            "room_number": 111,
            "price": [
                {
                    "date": "2020-12-05",
                    "price": 100000
                },
                {
                    "date": "2020-12-06",
                    "price": 100000
                },
                {
                    "date": "2020-12-07",
                    "price": 100000
                },
                {
                    "date": "2020-12-08",
                    "price": 100000
                },
                {
                    "date": "2020-12-09",
                    "price": 100000
                },
                {
                    "date": "2020-12-10",
                    "price": 100000
                },
                {
                    "date": "2020-12-11",
                    "price": 150000
                }
            ]
        }
    ]
}
```

Case 4: No overlap


GET http://127.0.0.1:8000/api/roomsearch/

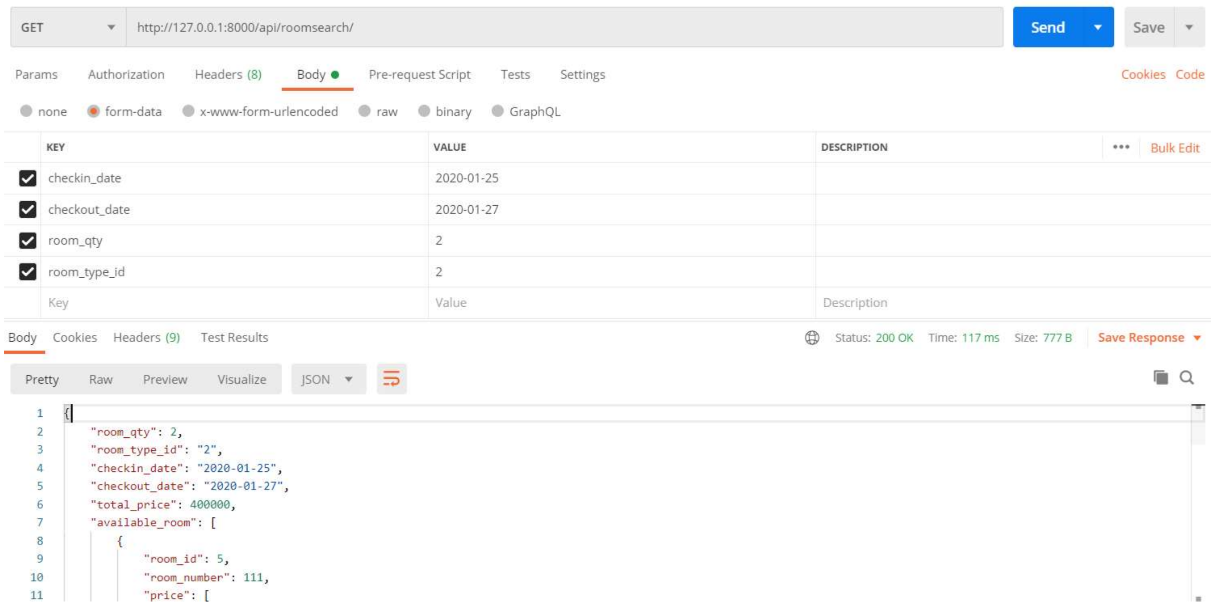
Body: form-data

KEY	VALUE	DESCRIPTION
checkin_date	2020-12-11	
checkout_date	2020-12-12	
room_qty	1	
room_type_id	2	

Status: 200 OK Time: 71 ms Size: 645 B

```
{
    "room_qty": "1",
    "room_type_id": "2",
    "checkin_date": "2020-12-11",
    "checkout_date": "2020-12-12",
    "total_price": 150000,
    "available_room": [
        {
            "room_id": 5,
            "room_number": 111,
```

```
{
    "room_qty": "1",
    "room_type_id": "2",
    "checkin_date": "2020-12-11",
    "checkout_date": "2020-12-12",
    "total_price": 150000,
    "available_room": [
        {
            "room_id": 5,
            "room_number": 111,
            "price": [
                {
                    "date": "2020-12-11",
                    "price": 150000
                }
            ]
        },
        {
            "room_id": 6,
            "room_number": 112,
            "price": [
                {
                    "date": "2020-12-11",
                    "price": 150000
                }
            ]
        },
        {
            "room_id": 8,
            "room_number": 111,
            "price": [
                {
                    "date": "2020-12-11",
                    "price": 150000
                }
            ]
        }
    ]
}
```



http://127.0.0.1:8000/api/roomsearch/

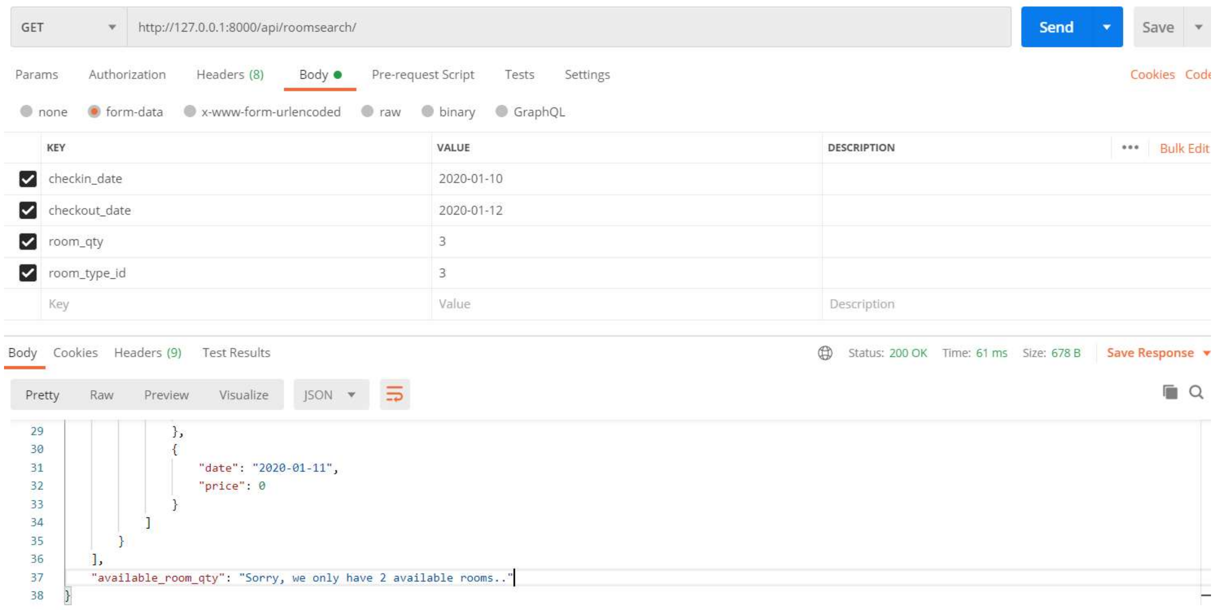
KEY	VALUE	DESCRIPTION
checkin_date	2020-01-25	
checkout_date	2020-01-27	
room_qty	2	
room_type_id	2	
Key	Value	Description

Status: 200 OK Time: 117 ms Size: 777 B

```
{
    "room_qty": 2,
    "room_type_id": "2",
    "checkin_date": "2020-01-25",
    "checkout_date": "2020-01-27",
    "total_price": 400000,
    "available_room": [
        {
            "room_id": 5,
            "room_number": 111,
            "price": [
```

```json
{
    "room_qty": 2,
    "room_type_id": "2",
    "checkin_date": "2020-01-25",
    "checkout_date": "2020-01-27",
    "total_price": 400000,
    "available_room": [
        {
            "room_id": 5,
            "room_number": 111,
            "price": [
                {
                    "date": "2020-01-25",
                    "price": 100000
                },
                {
                    "date": "2020-01-26",
                    "price": 100000
                }
            ]
        },
        {
            "room_id": 6,
            "room_number": 112,
            "price": [
                {
                    "date": "2020-01-25",
                    "price": 100000
                },
                {
                    "date": "2020-01-26",
                    "price": 100000
                }
            ]
        },
        {
            "room_id": 8,
            "room_number": 111,
            "price": [
                {
                    "date": "2020-01-25",
                    "price": 100000
                },
                {
                    "date": "2020-01-26",
                    "price": 100000
                }
            ]
        }
    ],
    "available_room_qty": 3
}
```

Case 5: Number of available rooms is not enough for the request


GET http://127.0.0.1:8000/api/roomsearch/

Body: form-data

KEY	VALUE	DESCRIPTION
checkin_date	2020-01-10	
checkout_date	2020-01-12	
room_qty	3	
room_type_id	3	

Status: 200 OK Time: 61 ms Size: 678 B

```
                },
                {
                    "date": "2020-01-11",
                    "price": 0
                }
            ]
        }
    ],
    "available_room_qty": "Sorry, we only have 2 available rooms.."
}
```

```
{
    "room_qty": 3,
    "room_type_id": "3",
    "checkin_date": "2020-01-10",
    "checkout_date": "2020-01-12",
    "total_price": 200000,
    "available_room": [
        {
            "room_id": 3,
            "room_number": 201,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 50000
                },
                {
                    "date": "2020-01-11",
                    "price": 50000
                }
            ]
        },
        {
            "room_id": 4,
            "room_number": 202,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 50000
                },
                {
                    "date": "2020-01-11",
                    "price": 50000
                }
            ]
        }
    ],
    "available_room_qty": "Sorry, we only have 2 available rooms.."
}
```

How To

1. Git repository: https://github.com/DeviHandevi/bobobox_backend_test
2. Read the README.md document in the project to run the app.
3. Use Postman Collection in the file named “Hotel Room Search.postman_collection.json” or use your custom input.

3. Promotion Service

Input Output Definition

Expected Input: send in “params” key with the value of JSON

```
{
    "promo_id": 1,
    "total_price": 170000,
    "rooms": [
        {
            "room_id": 3,
            "room_number": 201,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 30000
                },
                {
                    "date": "2020-01-11",
                    "price": 40000
                }
            ]
        },
        {
            "room_id": 4,
            "room_number": 202,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 50000
                },
                {
                    "date": "2020-01-11",
                    "price": 50000
                }
            ]
        }
    ]
}
```

Expected Output

```
{
    "promo_id": 1,
    "total_price": 170000,
    "total_promo_price": 50000,
    "total_final_price": 120000,
    "rooms": [
        {
            "room_id": 3,
            "room_number": 201,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 30000,
                    "promo_price": 10000,
                    "final_price": 20000
                },
                {
                    "date": "2020-01-11",
                    "price": 40000,
                    "promo_price": 10000,
                    "final_price": 30000
                }
            ]
        },
        {
            "room_id": 4,
            "room_number": 202,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 50000,
                    "promo_price": 10000,
                    "final_price": 40000
                },
                {
                    "date": "2020-01-11",
                    "price": 50000,
                    "promo_price": 20000,
                    "final_price": 30000
                }
            ]
        }
    ]
}
```

Minimum Function

Minimum promotion functionality:

1. Calculate new price after promo, promo could be either on currency or percentage.
2. Promo Quota
    a. Avoid the promo quota to be fully spent on a single day by dividing to the length of promo date start (or current date) to end
    b. Distribute the promo quota to stay date range (e.g. one promo quota for Rp100000 for 2 days stay will have each 50000 per day)
    c. Stay date range is used to filter the check-in date, making sure check-in are in eligible date to stay for the promotion
    d. Single promo quota is tied to a single reservation
3. Promo rules contains
    a. Minimum total nights
    b. Minimum number of rooms
    c. Check-in day: check-in only available for specific selected day (Sun-Sat)

d. Booking day: booking only available for specific selected day (Sun-Sat)
e. Booking hour range: booking only available for specific hour range

Table Design

To fulfill the promo rules, we will use following table design to store promo data:

Promo			Notes
PK	**id**	**int**	Auto incremented ID
	name	string	The name of the promo (e.g. "Christmas 2020")
	value	int	Value of the promotion (e.g. 12 or 150000)
	promo_type	int	Number 1 for currency or 2 for percentage (e.g. 1)
	quota	int	Total number of promo can be used (e.g. 10)
	promo_date_start	date	Promo date start (e.g. 2020-12-10)
	promo_date_end	date	Promo date start (e.g. 2020-12-15)
	stay_date_start	date	Stay date start (e.g. 2021-01-01)
	stay_date_end	date	Stay date start (e.g. 2021-01-30)

and to store promo rule data:

PromoRule			Notes
PK	**id**	**int**	Auto incremented ID
FK	promo_id	int	Foreign key to promo ID
	min_nights	int	Minimum number of nights booked (e.g. 3)
	min_rooms	int	Minimum number of rooms booked (e.g. 1)
	checkin_day	int	Number 0 for Sunday, 1 for Monday, ... 6 for Saturday (e.g. 1)
	booking_day	int	Number 0 for Sunday, 1 for Monday, ... 6 for Saturday (e.g. 5)
	booking_hour_start	time	Booking hour range start (e.g. 08:00)
	booking_hour_end	time	Booking hour range end (e.g. 18:00)

Example Case

Christmas 2020

1. Promo 10%
2. For first 10 people
3. Promo on December 10, 2020 until December 11, 2020 at 08:00-10:00
4. To stay between January 1, 2021 until January 30, 2021

with condition:

1. Booking on Thursday between 08:00-09:00, check-in on Monday, minimal 3 nights for 2 rooms
2. Booking on Friday between 07:00-09:00, check-in on Tuesday, minimal 1 nights for 5 rooms

It will need a promo record with:

1. id = 1
2. name = Christmas 2020
3. value = 10
4. promo_type = 2
5. quota = 10
6. booking_date_start = 2020-12-10
7. booking_date_end = 2020-12-11
8. stay_date_start = 2021-01-01
9. stay_date_end = 2021-01-30

and first promo rule:

1. promo_id = 1
2. min_nights = 3
3. min_rooms = 2
4. checkin_day = 1
5. booking_day = 4
6. booking_hour_start = 08:00
7. booking_hour_end = 09:00

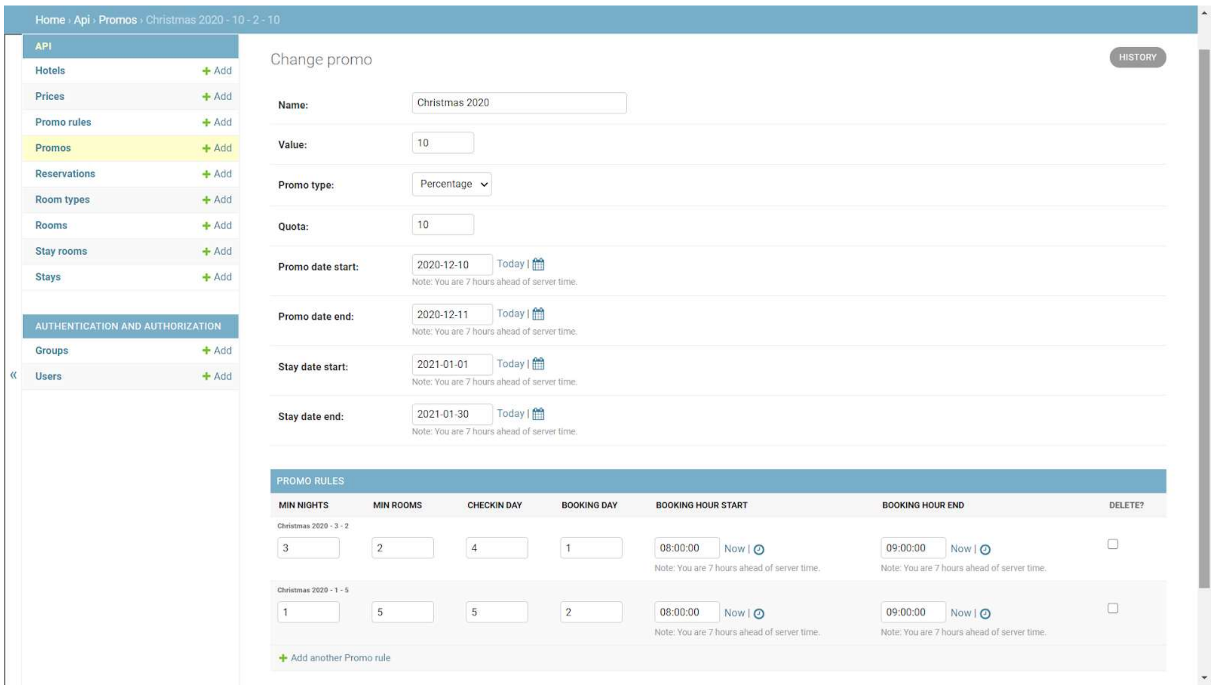
and second promo rule:

1. promo_id = 1
2. min_nights = 1
3. min_rooms = 5
4. checkin_day = 2
5. booking_day = 5
6. booking_hour_start = 07:00
7. booking_hour_end = 09:00

Example Using Postman

URL: http://127.0.0.1:8000/api/applypromo/

Example of the promo and promo rule data:



Home › Api › Promos › Christmas 2020 - 10 - 2 - 10

Change promo

Name:	Christmas 2020
Value:	10
Promo type:	Percentage
Quota:	10
Promo date start:	2020-12-10 Today | Note: You are 7 hours ahead of server time.
Promo date end:	2020-12-11 Today | Note: You are 7 hours ahead of server time.
Stay date start:	2021-01-01 Today | Note: You are 7 hours ahead of server time.
Stay date end:	2021-01-30 Today | Note: You are 7 hours ahead of server time.

PROMO RULES

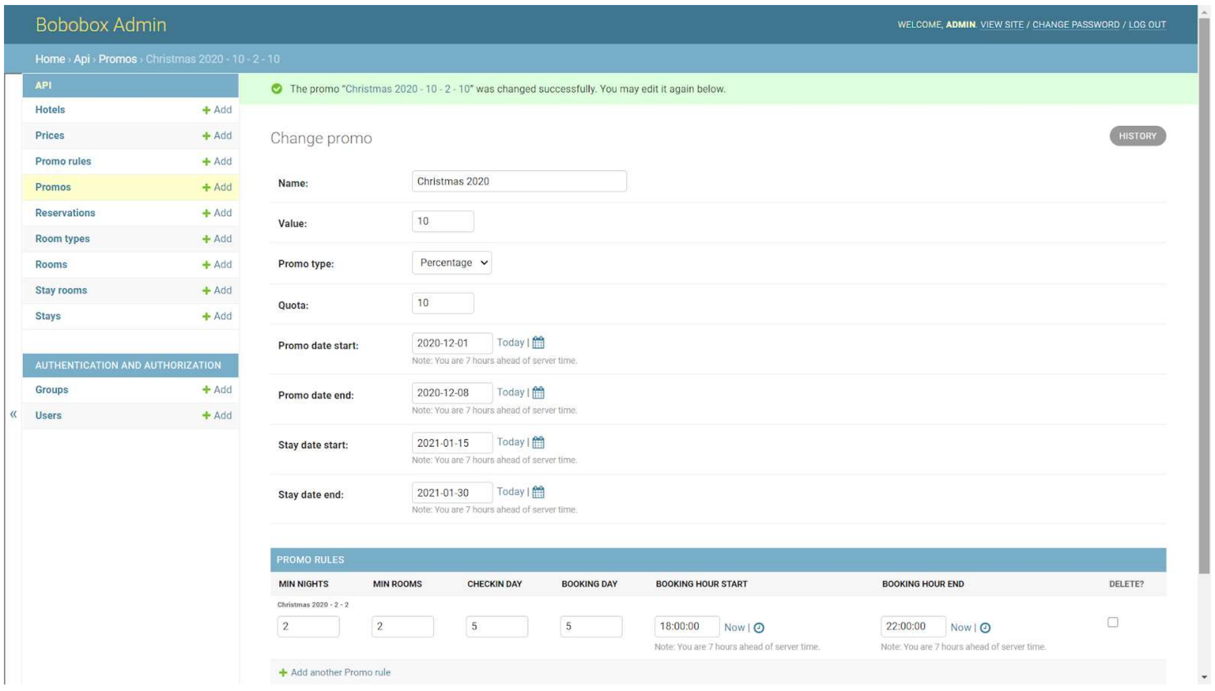
MIN NIGHTS	MIN ROOMS	CHECKIN DAY	BOOKING DAY	BOOKING HOUR START	BOOKING HOUR END	DELETE?
Christmas 2020 - 3 - 2						
3	2	4	1	08:00:00 Now | Note: You are 7 hours ahead of server time.	09:00:00 Now | Note: You are 7 hours ahead of server time.	☐
Christmas 2020 - 1 - 5						
1	5	5	2	08:00:00 Now | Note: You are 7 hours ahead of server time.	09:00:00 Now | Note: You are 7 hours ahead of server time.	☐

+ Add another Promo rule

Example input:

```
{
    "promo_id": 1,
    "total_price": 170000,
    "rooms": [
        {
            "room_id": 3,
            "room_number": 201,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 30000
                },
                {
                    "date": "2020-01-11",
                    "price": 40000
                }
            ]
        },
        {
            "room_id": 4,
            "room_number": 202,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 50000
                },
                {
                    "date": "2020-01-11",
                    "price": 50000
                }
            ]
        }
    ]
}
```

Case 1: Percentage Promo



The promo "Christmas 2020 - 10 - 2 - 10" was changed successfully. You may edit it again below.

Change promo

Name:	Christmas 2020
Value:	10
Promo type:	Percentage
Quota:	10
Promo date start:	2020-12-01 Note: You are 7 hours ahead of server time.
Promo date end:	2020-12-08 Note: You are 7 hours ahead of server time.
Stay date start:	2021-01-15 Note: You are 7 hours ahead of server time.
Stay date end:	2021-01-30 Note: You are 7 hours ahead of server time.

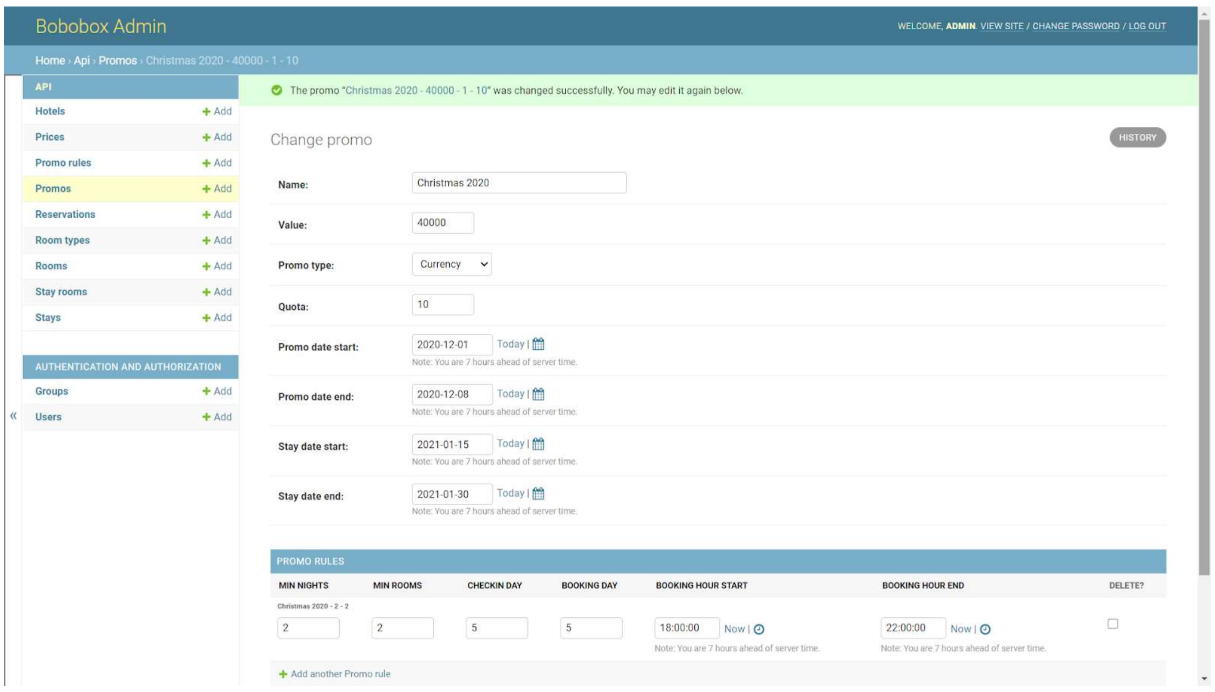
PROMO RULES

MIN NIGHTS	MIN ROOMS	CHECKIN DAY	BOOKING DAY	BOOKING HOUR START	BOOKING HOUR END	DELETE?
Christmas 2020 - 2 - 2						
2	2	5	5	18:00:00 Note: You are 7 hours ahead of server time.	22:00:00 Note: You are 7 hours ahead of server time.	☐

When the promo and rules are valid:

```
{
    "promo_id": 1,
    "total_price": 170000,
    "rooms": [
        {
            "room_id": 3,
            "room_number": 201,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 30000,
                    "promo_price": 3000,
                    "final_price": 27000
                },
                {
                    "date": "2020-01-11",
                    "price": 40000,
                    "promo_price": 4000,
                    "final_price": 36000
                }
            ]
        },
        {
            "room_id": 4,
            "room_number": 202,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 50000,
                    "promo_price": 5000,
                    "final_price": 45000
                },
                {
                    "date": "2020-01-11",
                    "price": 50000,
                    "promo_price": 5000,
                    "final_price": 45000
                }
            ]
        }
    ],
    "total_promo_price": 17000,
    "total_final_price": 153000
}
```

Case 2: Currency Promo



Bobobox Admin

WELCOME, **ADMIN**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Api › Promos › Christmas 2020 - 40000 - 1 - 10

The promo "Christmas 2020 - 40000 - 1 - 10" was changed successfully. You may edit it again below.

Change promo

Name:	Christmas 2020
Value:	40000
Promo type:	Currency
Quota:	10
Promo date start:	2020-12-01 Today | Note: You are 7 hours ahead of server time.
Promo date end:	2020-12-08 Today | Note: You are 7 hours ahead of server time.
Stay date start:	2021-01-15 Today | Note: You are 7 hours ahead of server time.
Stay date end:	2021-01-30 Today | Note: You are 7 hours ahead of server time.

PROMO RULES

Christmas 2020 - 2 - 2

MIN NIGHTS	MIN ROOMS	CHECKIN DAY	BOOKING DAY	BOOKING HOUR START	BOOKING HOUR END	DELETE?
2	2	5	5	18:00:00 Now | Note: You are 7 hours ahead of server time.	22:00:00 Now | Note: You are 7 hours ahead of server time.	☐

When the promo and rules are valid:

```
{
    "promo_id": 1,
    "total_price": 170000,
    "rooms": [
        {
            "room_id": 3,
            "room_number": 201,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 30000,
                    "promo_price": 7058,
                    "final_price": 22941
                },
                {
                    "date": "2020-01-11",
                    "price": 40000,
                    "promo_price": 9411,
                    "final_price": 30588
                }
            ]
        },
        {
            "room_id": 4,
            "room_number": 202,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 50000,
                    "promo_price": 11764,
                    "final_price": 38235
                },
                {
                    "date": "2020-01-11",
                    "price": 50000,
                    "promo_price": 11764,
                    "final_price": 38235
                }
            ]
        }
    ],
    "total_promo_price": 40000,
    "total_final_price": 130000
}
```

When the promo discount is larger than the total price:

```
{
    "promo_id": 1,
    "total_price": 170000,
    "rooms": [
        {
            "room_id": 3,
            "room_number": 201,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 30000,
                    "promo_price": 30000,
                    "final_price": 0
                },
                {
                    "date": "2020-01-11",
                    "price": 40000,
                    "promo_price": 40000,
                    "final_price": 0
                }
            ]
        },
        {
            "room_id": 4,
            "room_number": 202,
            "price": [
                {
                    "date": "2020-01-10",
                    "price": 50000,
                    "promo_price": 50000,
                    "final_price": 0
                },
                {
                    "date": "2020-01-11",
                    "price": 50000,
                    "promo_price": 50000,
                    "final_price": 0
                }
            ]
        }
    ],
    "total_promo_price": 170000,
    "total_final_price": 0
}
```

Notes: When promo price is larger than the room price itself, the promo price will be set to equal to room price instead, so it will not produce a negative price.

Case 3: Invalid Promo

If promo does not exist:

```
{
    "error": "Promo not found"
}
```

If current date is less than promo date start:

```
{
    "error": "Promo has not started yet"
}
```

If current date is more than promo date end:

```
{
    "error": "Promo has finished"
}
```

Case 4: Invalid Promo Rules

Checking the request to each promo rule in the specified promos ID:

1. Number of rooms booked should be greater or equal to the rule minimum rooms
2. Total nights booked should be greater or equal to the rule minimum nights
3. Booking day (using the date of the request) should be the same as the rule booking day
4. Booking hour (using the hour of the request) should be between the rule booking hour start and booking hour end
5. Check-in day (using the earliest date in the room price list) should be the same as the rule check-in day

If no promo rules valid for the request, it returns:

```
{
    "error": "No promo rule valid for this request."
}
```

Case 5: Promo Quota Full

Allowed quota for the day is the promo quota left divided by the number of promo days left. After that, today's quota left will be checked to today's reservations which use the promo ID.

If no more quota left, it returns:

```
{
    "error": "No promo quota left."
}
```

How To

1. Git repository: https://github.com/DeviHandevi/bobobox_backend_test
2. Read the README.md document in the project to run the app.
3. Use Postman Collection in the file named "Apply Promo.postman_collection.json" or use your custom input.

Notes: I provided one expected input on the Postman Collection for an example. Since the response depends on the time the user sends the request, the tester needs to change the promo data manually to adjust to the expected conditions towards the request.