Hotel Booking System
(Backend System)

Submitted By :

Md Meraj Alam(1706134)
CSE
Intern at OYO

Under the guidance of :

Dara Singh
Sr. Engr.. Manager, OYO Workspaces
Mentor :
Debashish Karmakar (SDE-2 ,OYO)


MAJOR PROJECT
" HOTEL
BOOKING SYSTEM".

ABSTRACT

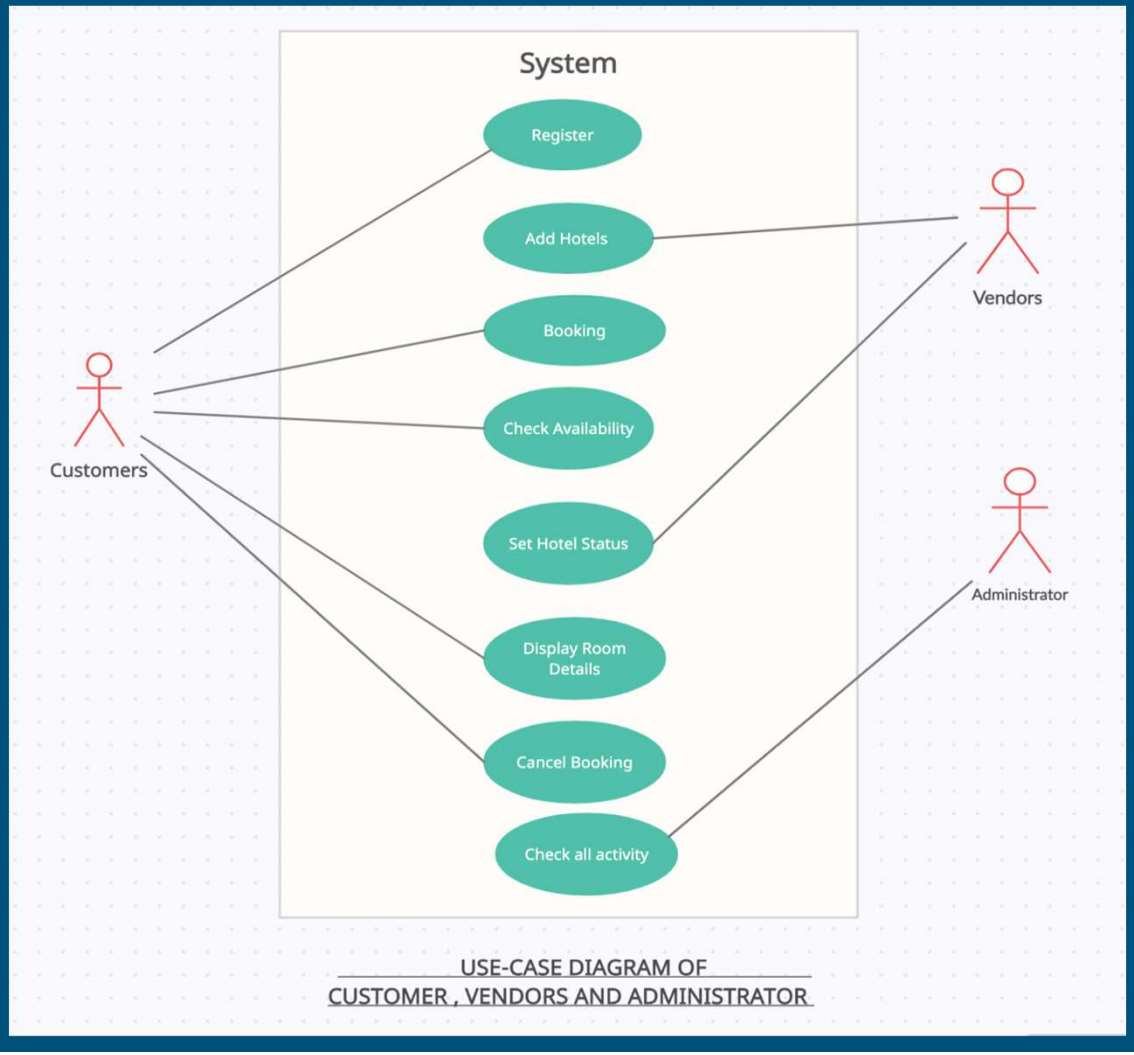
- This project aims at creating an Online Hotel Booking System. Which can be used by Customer to reserve hotels and vendors can add their hotels.
- Customer can check the availability of rooms and facilities.
- Customer can book rooms and are also able to cancel them.
- Vendors can also add their hotels and set the status of their hotel to be Active or In-Active state.

System Analysis

- This System is used by three types of User (Modules):-

1. Customer
2. Vendors
3. Administrator

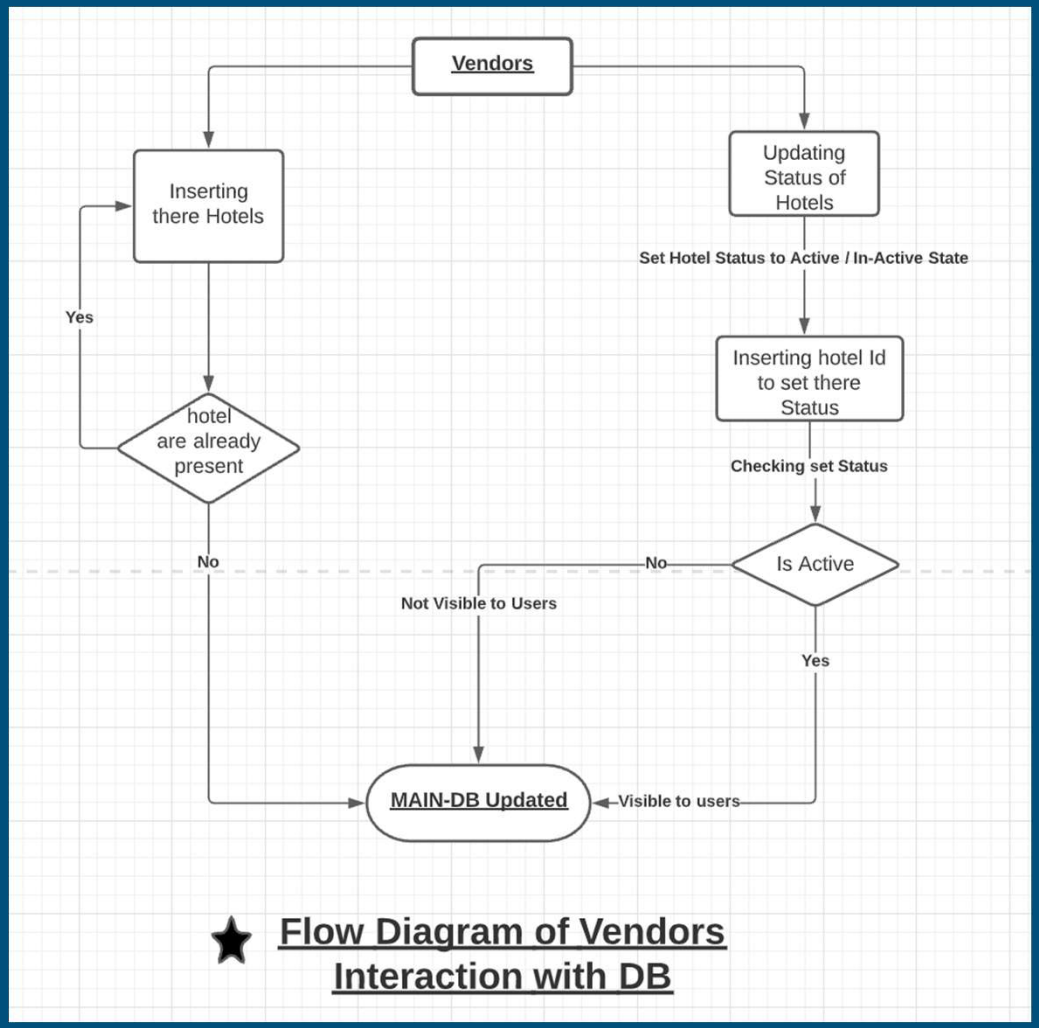
USE-CASE DIAGRAM



USE-CASE DIAGRAM OF
CUSTOMER , VENDORS AND ADMINISTRATOR

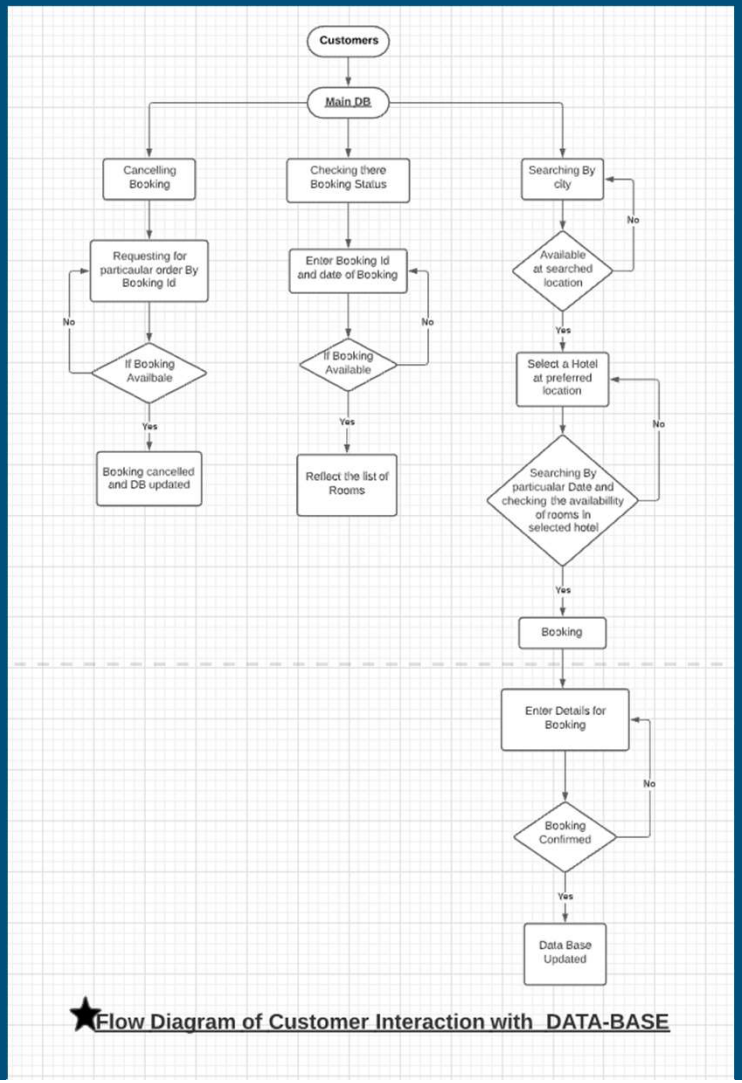
- Here is the use case diagram which shows the Customer , Vendors and Administrator interaction with the system .

FLOW CHART OF VENDORS



- Here is the flow chart of Vendors interaction with Main DB and the flow of interaction with the project

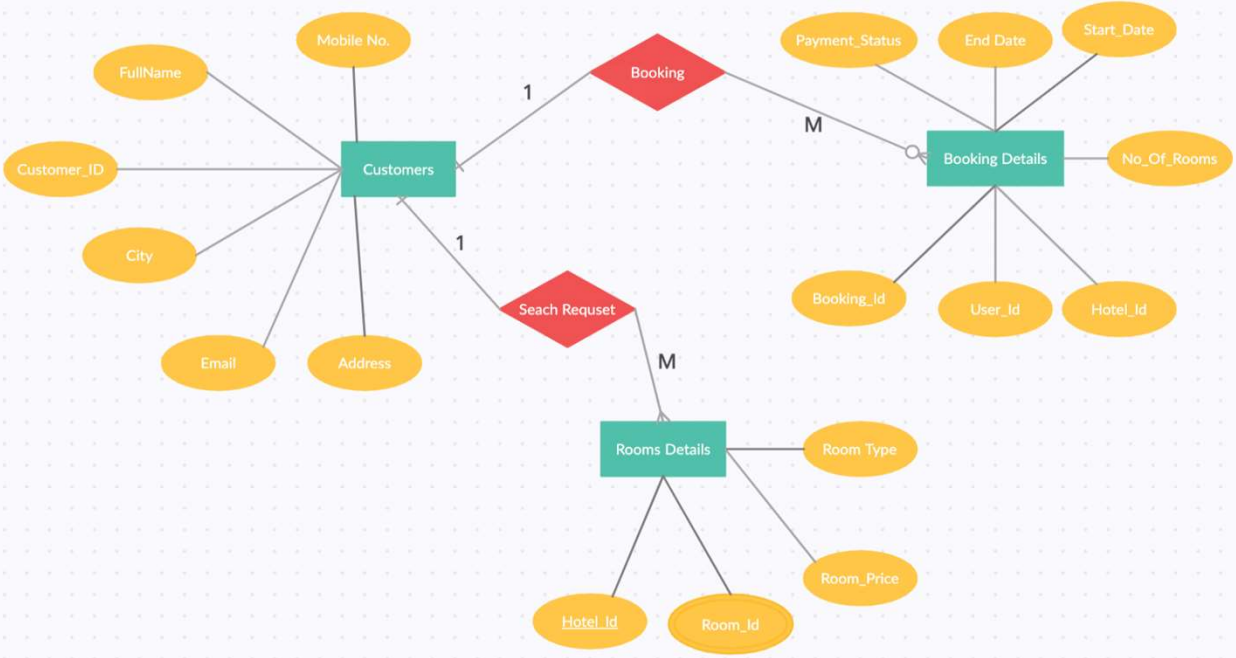
FLOW CHART OF CUSTOMER



★Flow Diagram of Customer Interaction with DATA-BASE

- Here is the flow chart of customer interaction with DB and flow of interaction with project

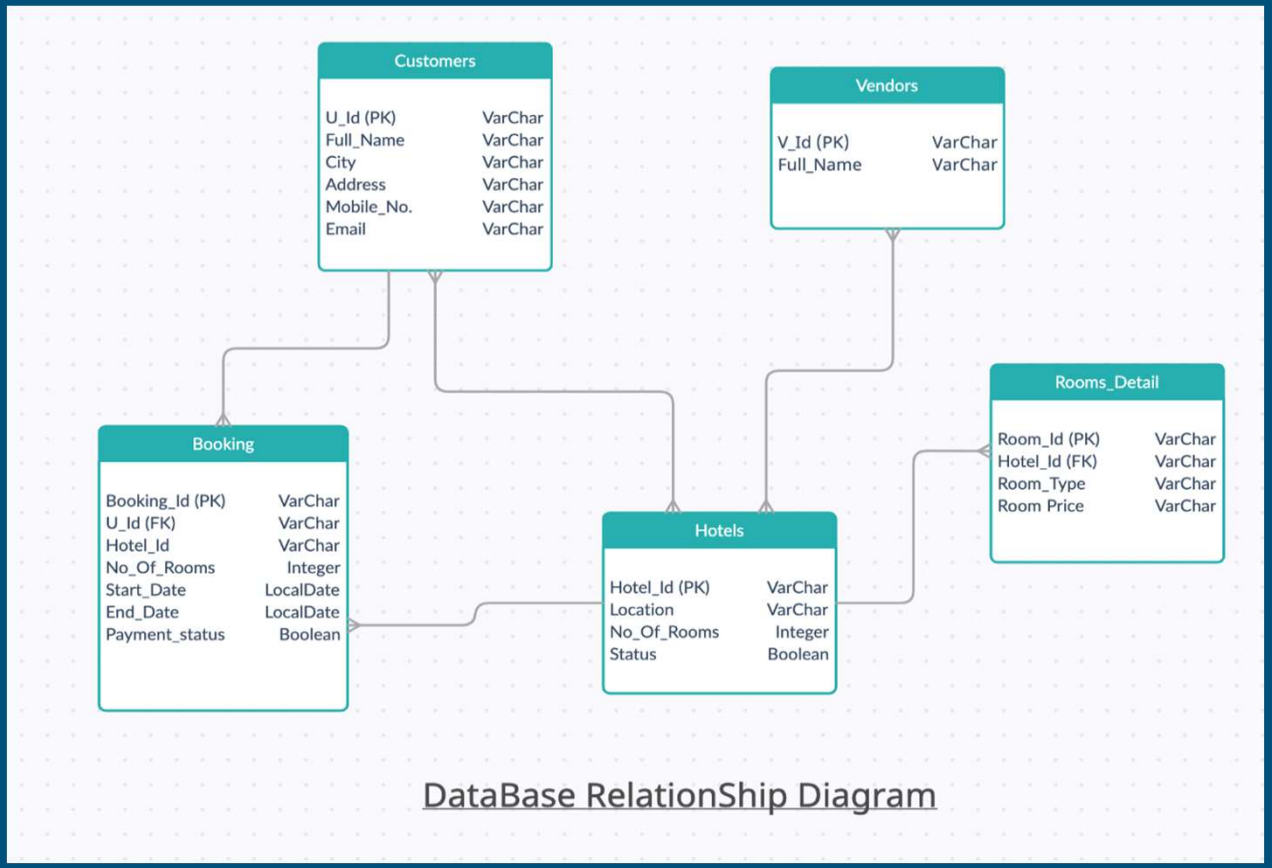
ER-DIAGRAM OF CUSTOMER



ER-Diagram of Customer Interaction

- Here is the ER (Entity Relationship) Diagram of customer for booking and search request of rooms type.

DB-RELATIONSHIP DIAGRAM



- This diagram show the relationship between the table like one to one , one to many and many to many.

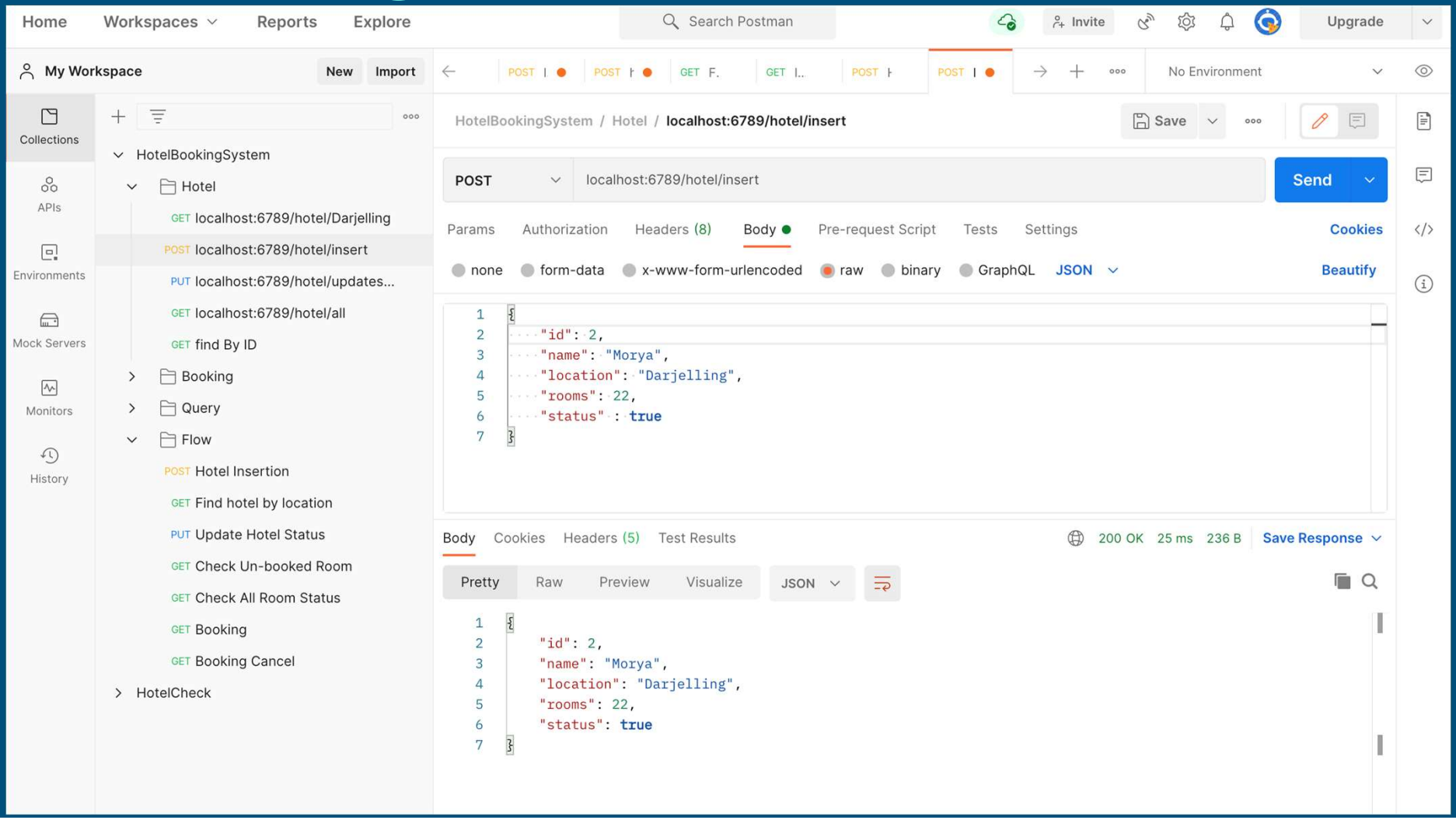
- I have done every thing only through the backend so Now below is the code part of my project and I tested their API by hitting them from the postman.

Vendors - > Inserting Hotels

```java
package com.meraj.springboot.controllers;


import ...

@RestController
@RequestMapping("/hotel")
public class HotelController {

    @Autowired
    HotelRepository hotelRepo;

    @Autowired
    SupportRepository supportRepo;

    @GetMapping("/all")
    public List<Hotel> findAll() { return hotelRepo.findAll(); }

    @PostMapping("/insert")
    public Hotel createHotel(@RequestBody Hotel hotel){
        Hotel insertedHotel = hotelRepo.insert(hotel);

        long id = hotel.getId();
        HashMap<LocalDate,int[]> roomStatus = new HashMap<LocalDate,int[]>();
        Support support = new Support(id,roomStatus);
        supportRepo.insert(support);
        return insertedHotel;
    }
```

- Here is the part of code through which vendors can Add their Hotels to the web-App

Testing(Vendors -> Enter their Hotels)

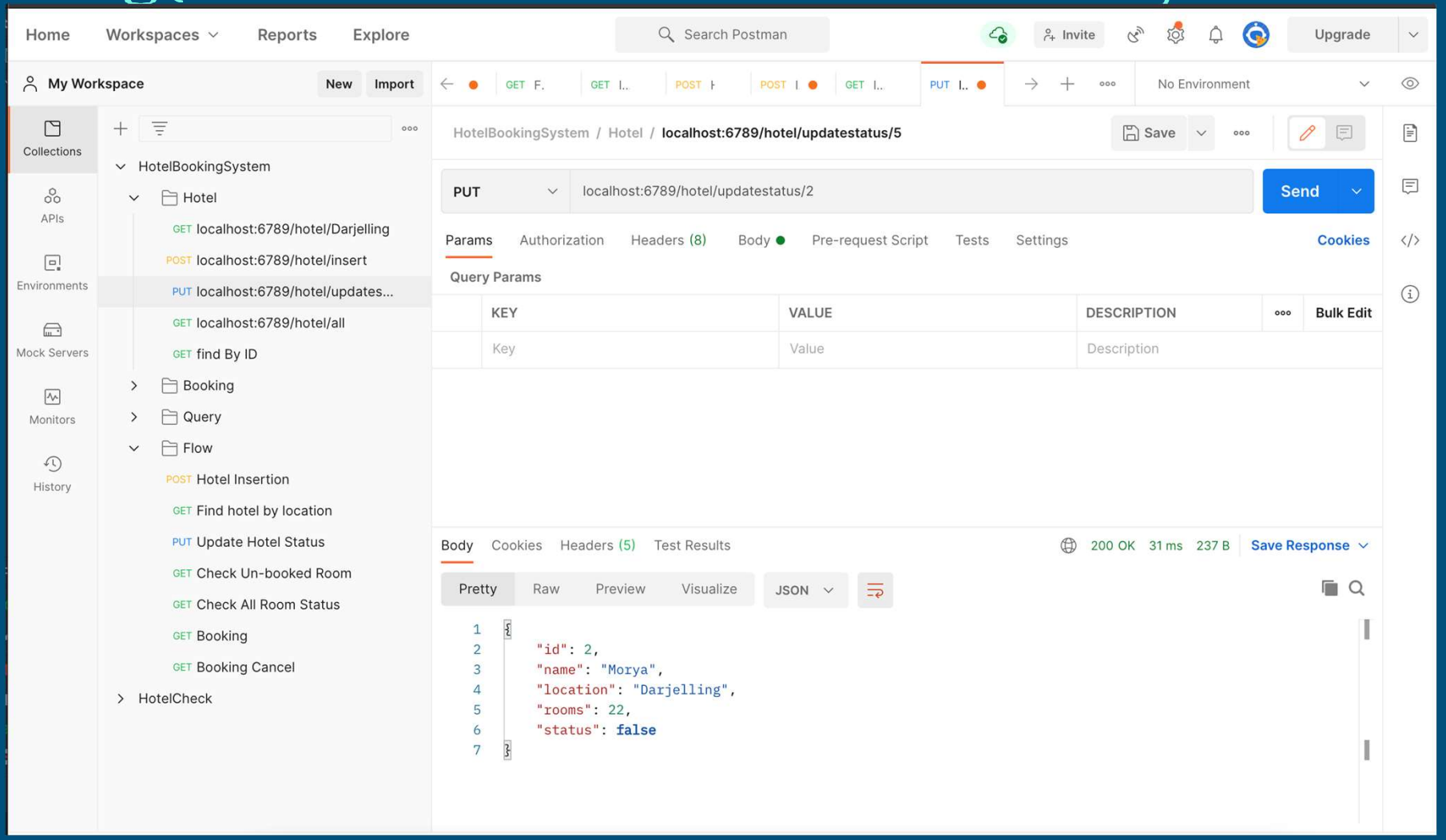


Administrator -> Update status

```java
package com.meraj.springboot.controllers;


import ...

@RestController
@RequestMapping("/hotel")
public class HotelController {

    @Autowired
    HotelRepository hotelRepo;

    @Autowired
    SupportRepository supportRepo;

    @PutMapping("/updatestatus/{id}")
    public Hotel updateStatus(@PathVariable long id) {
        Hotel hotel = hotelRepo.findById(id).get();
        hotel.setStatus(!hotel.isStatus());
        hotelRepo.save(hotel);
        return  hotel;
    }

```

- Here is the part of code through administrator update the status of hotel on the request of vendors.

Testing(ADMINISTRATOR -> Activate / In-Activate)

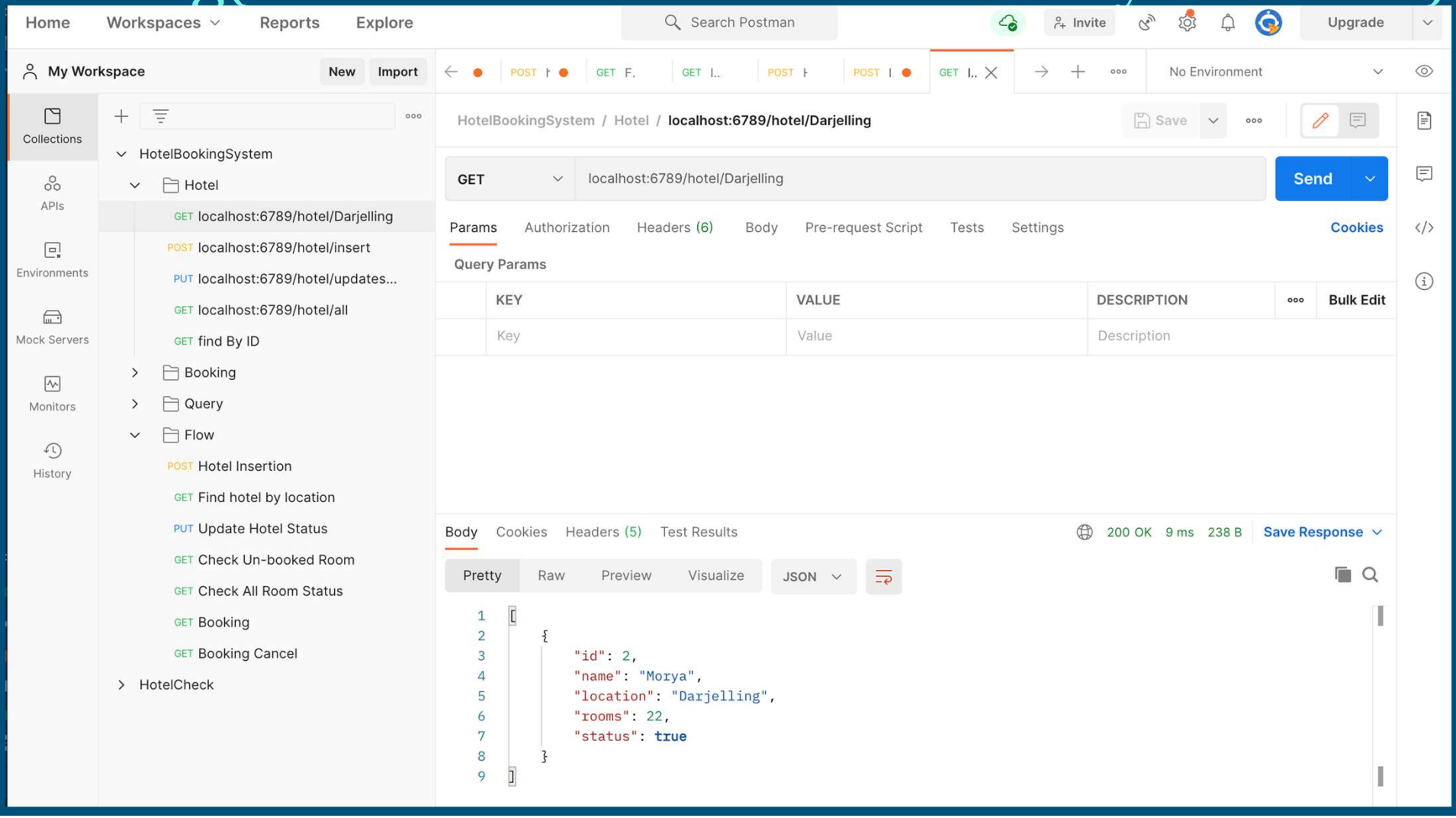


Customer->Searching Hotels

```java
package com.meraj.springboot.controllers;


import ...

@RestController
@RequestMapping("/hotel")
public class HotelController {

    @Autowired
    HotelRepository hotelRepo;

    @Autowired
    SupportRepository supportRepo;

    @GetMapping("/all")
    public List<Hotel> findAll() {
        return hotelRepo.findAll();
    }


    @GetMapping("/{location}")
    public List<Hotel> findByLocation(@PathVariable String location){
        return hotelRepo.findByLocationAndStatus(location, b: true);
    }
}
```

- Here is the part of code through customer can search the hotel in its preferred location and its only return the list of hotels which are in Active state and present in the searched location.

Testing(Customer -> Search Hotels By Location)

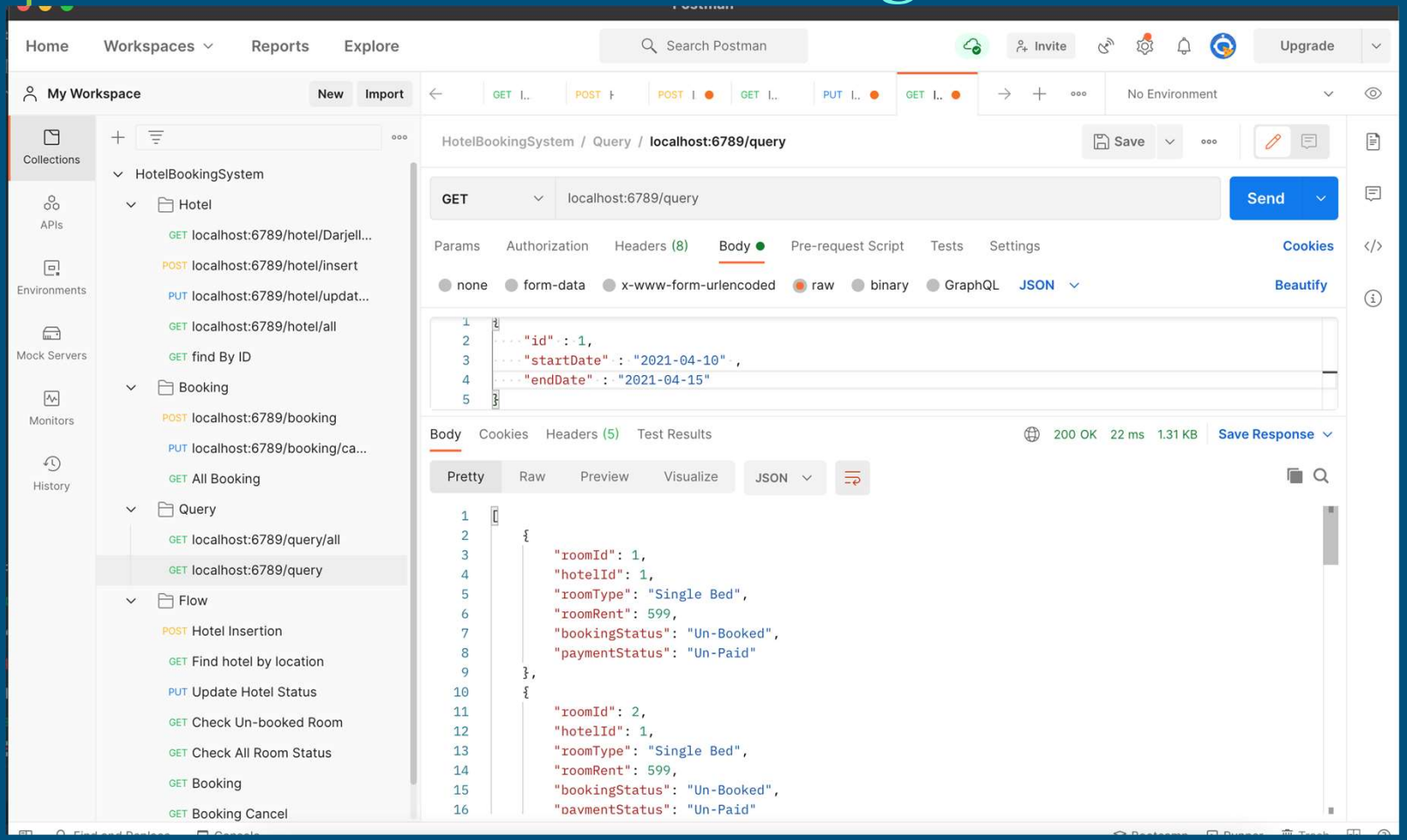


Searching query

```java
package com.meraj.springboot.controllers;

import ...

@RestController
@RequestMapping("/query")
public class QueryController {

    @Autowired
    SupportRepository supportRepo;
    @Autowired
    HotelRepository hotelRepo;

    @GetMapping
    public List<Room> findRoomStatus(@RequestBody Query query){
        LocalDate startDate = query.getStartDate();
        LocalDate endDate = query.getEndDate();

        LocalDate today = LocalDate.now();
        if(startDate.isBefore(today)  || endDate.isBefore(today) || endDate.isBefore(startDate)) {
            return null;
        }
        List<Room> result  = new ArrayList<~>();
        long hotelId = query.getId();
        Hotel hotel  = hotelRepo.findById(hotelId).get();
        int noOfRooms = hotel.getRooms();
        Support support = supportRepo.findById(hotelId).get();
        int[] finalRoomResult = new int[noOfRooms];
        HashMap<LocalDate,int[]> currentStatus = support.getRoomStatus();


        for(LocalDate date = startDate; date.isBefore(endDate); date = date.plusDays(1)) {
            if(currentStatus.containsKey(date)) {
                int[] currentRoomStatus = currentStatus.get(date);
                for(int i=0;i<currentRoomStatus.length;i++) {
                    finalRoomResult[i]= currentRoomStatus[i];
```

- For searching the No. of un-booked room in particular date range I had apply the first Merge sort algorithm on start and end date then apply O(N) time traversing approach to count the number of un-booked room.

Testing(Customer -> Search Un-Booked rooms in Particular Date Range)

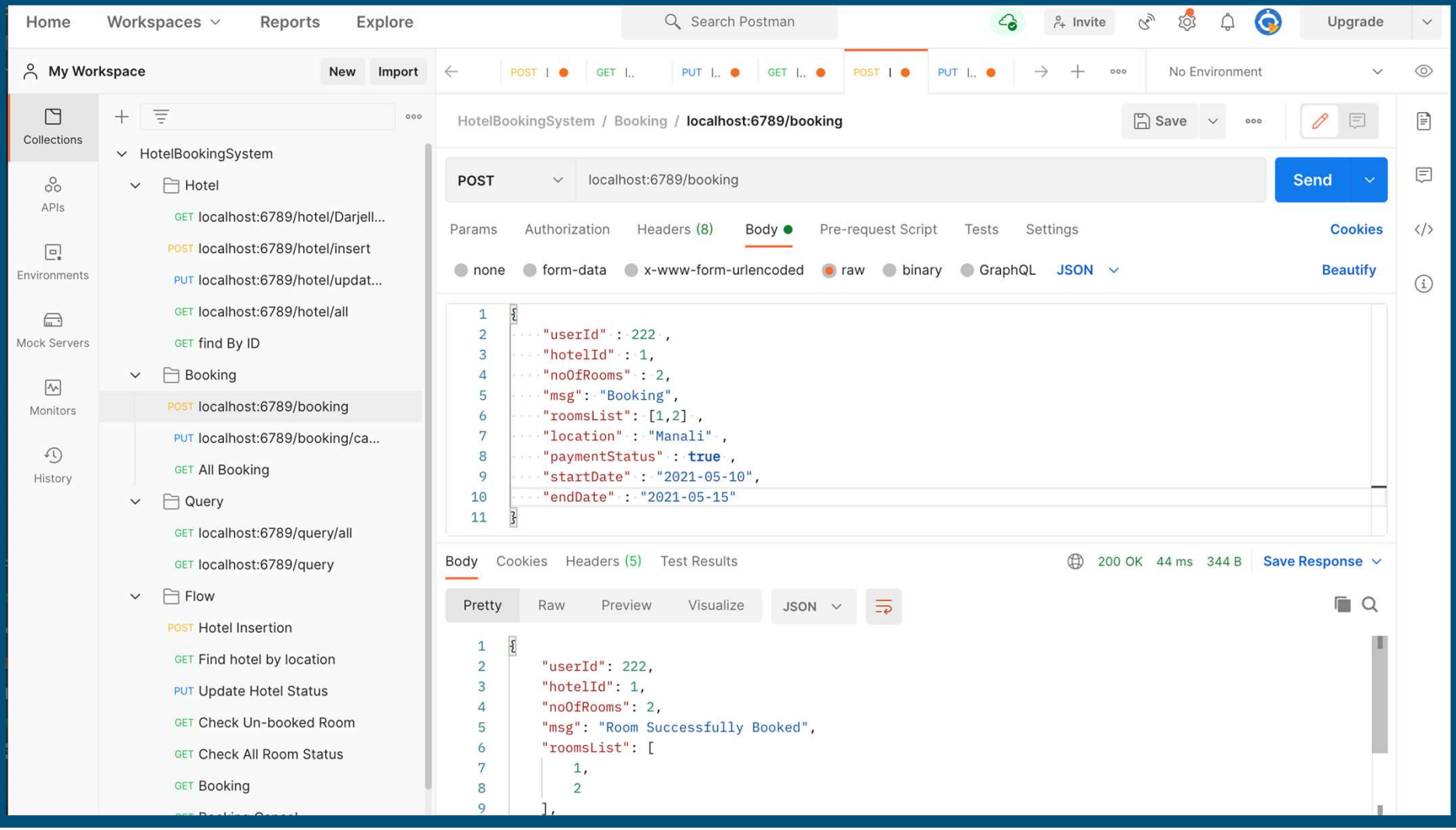


Booking

```
44
45      @PostMapping
46  @   public BookingDetails bookingHotel(@RequestBody BookingDetails booking) {
47          LocalDate startDate = booking.getStartDate();
48          LocalDate endDate = booking.getEndDate();
49          LocalDate today = LocalDate.now();
50
51          if(startDate.isBefore(today)  || endDate.isBefore(today) || endDate.isBefore(startDate)) {
52              booking.setHotelId(0);
53              booking.setUserId(0);
54              booking.setLocation(null);
55              booking.setEndDate(null);
56              booking.setNoOfRooms(0);
57              booking.setPaymentStatus(false);
58              booking.setRoomsList(null);
59              booking.setStartDate(null);
60
61              booking.setMsg("*** Error *** Enter Date is Not in proper manner");
62              return booking;
63          }
64          List<Integer> roomRequest = booking.getRoomsList();
65          int noOfRooms = booking.getNoOfRooms();
66          if (noOfRooms != roomRequest.size()) {
67              booking.setHotelId(0);
68              booking.setUserId(0);
69              booking.setLocation(null);
70              booking.setEndDate(null);
71              booking.setNoOfRooms(0);
72              booking.setPaymentStatus(false);
73              booking.setRoomsList(null);
74              booking.setStartDate(null);
75
76              booking.setMsg("*** Error *** noOfRooms and roomRequest list length is not equale");
77              return booking;
78          }
79
80          long hotelId = booking.getHotelId();
```

- Here is the part of code through customer can book the rooms , here I have used the FCFS Algorithm for booking the rooms ,means booking are done for those customer who have apply first.

Testing(Customer -> Room Booking)

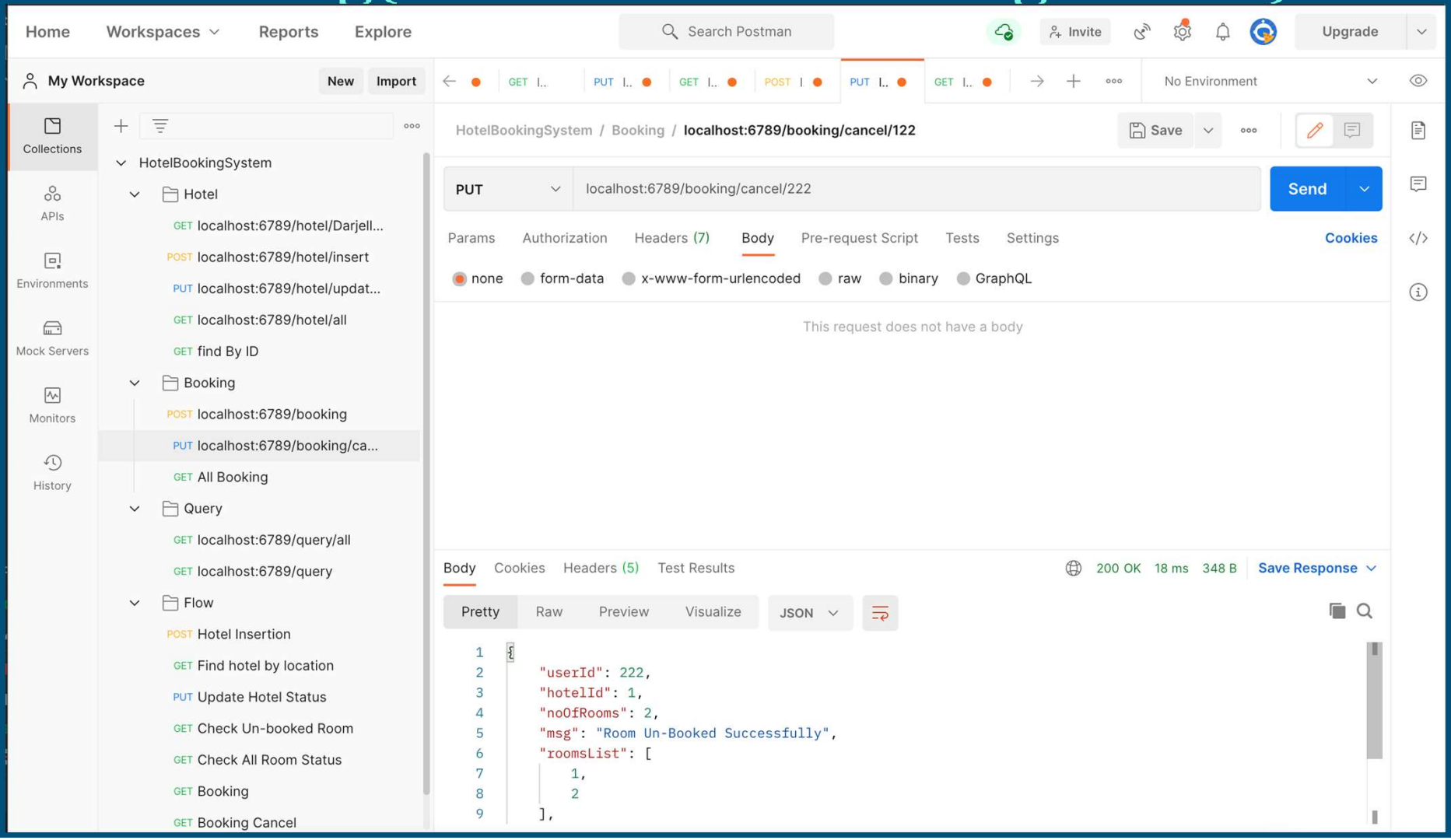


Booking -cancellation

```java
@SuppressWarnings("null")
@PutMapping("/cancel/{id}")
public BookingDetails cancelBooking(@PathVariable long id) {
    if(bookingRepo.findById(id)==null){
        BookingDetails bookingDetails = new BookingDetails( userId: 0, hotelId: 0, noOfRooms: 0, msg: "Us
        bookingDetails.setMsg("UserId not found");
        return bookingDetails;
    }
    BookingDetails bookingDetails = bookingRepo.findById(id).get();
    if(bookingDetails==null) {
        bookingDetails.setMsg("No any Booking is done with this Id");
        return bookingDetails;
    }
    long hotelId = bookingDetails.getHotelId();
    Support support = supportRepo.findById(hotelId).get();
    HashMap<LocalDate,int[]> roomStatus = support.getRoomStatus();
    LocalDate startDate = bookingDetails.getStartDate();
    LocalDate endDate = bookingDetails.getEndDate();
    List<Integer> roomList = bookingDetails.getRoomsList();
    boolean paymentStatus = bookingDetails.isPaymentStatus();
    for (LocalDate date = startDate; date.isBefore(endDate); date = date.plusDays(1)) {
        if(roomStatus.containsKey(date)) {
            int [] currentRooms = roomStatus.get(date);
            for(Integer num:roomList) {
                currentRooms[num-1]=0;%
            }
            roomStatus.replace(date, currentRooms);
        }
    }
    support.setRoomStatus(roomStatus);
    supportRepo.save(support);
    bookingDetails.setMsg("Room Un-Booked Successfully");
    if(paymentStatus==true) {
        bookingDetails.setPaymentStatus(false);
    }
    bookingRepo.save(bookingDetails);
```

- Here is the part of code through administrator can cancel the booking by booking ID on the request of customer, And after that list is updated

Testing(Customer -> Booking Cancel)



Technology Stack

- Spring Boot
- Java
- Mongo DB
- Postman
- IntelliJ

- **GitHub Link - https://github.com/mdmeraj13/Hotel-Booking-System**

Conclusion

- While developing this project I have learnt a lot about Hotel Booking System.
- The Online Hotel Booking system was developed to replace the manual process of booking for a hotel room or any other facility of hotel.
- The Old system does not serve the customer in a better way , rather it makes customer data vulnerable.
- The new system keeps proper record of customer for emergency and security purpose.
- I have also learnt how to make a system user friendly.

References

- Spring : https://spring.io/
- Spring Boot : https://spring.io/projects/spring-boot
- Apache Thrift : https://thrift.apache.org/
- Apache Kafka Architecture view in depth: https://medium.com/swlh/apache-kafka-in-a-nutshell-5782b01d9ffb
- Apache Kafka : https://kafka.apache.org/
- SonarQube Code Analysis : https://medium.com/code-to-express/sonarqube-static-code-analysis-8b6ca5226d2b
- Code Coverage: https://stackoverflow.com/questions/195008/what-is-code-coverage-and-how-do-you-measure-it#:~:text=Code%20coverage%20is%20a%20measurement,tests%20against%20the%20instrumented%20product.
- Code Vulnerability : https://www.codegrip.tech/productivity/what-is-code-vulnerability/


THANK YOU