L2 Evaluation — Assignment — ACL Digital

GitHub Link: https://github.com/Suraj7241/L2_Evaluation_ACL_Digital

Q. 1] How do you create a deadlock scenario programmatically in Java?

```java
package com.DeadLock;

class DeadLock {
      static final String resource1= "Printer";
      static final String resource2= "Scanner";

      public static void main(String[] args) {
        ThreadDemo1 Thread1 = new ThreadDemo1();
        ThreadDemo2 Thread2 = new ThreadDemo2();
        Thread1.start();
        Thread2.start();
      }

      private static class ThreadDemo1 extends Thread {
        public void run() {
          synchronized (resource1) {
            System.out.println("Desktop "+ ": locked" + resource1);
            try {
                Thread.sleep(1000);
            } catch (Exception e) {
            }
          }
          System.out.println("Desktop " + ": waiting for" + resource2+".....");
          synchronized (resource2) {
            System.out.println("Desktop "+ ": locked" + resource2);
            }
          }
        }
```

```java
    private static class ThreadDemo2 extends Thread {
      public void run() {
        synchronized (resource2) {
          System.out.println("Laptop "+": locked" + resource2);
          try {
            Thread.sleep(100);
          } catch (Exception e) {
          }
        }
        System.out.println("Laptop " + ": waiting for" + resource1+".....");
        synchronized (resource1) {
          System.out.println("Laptop "+ ": locked" + resource1);
        }
      }
    }
  }
```

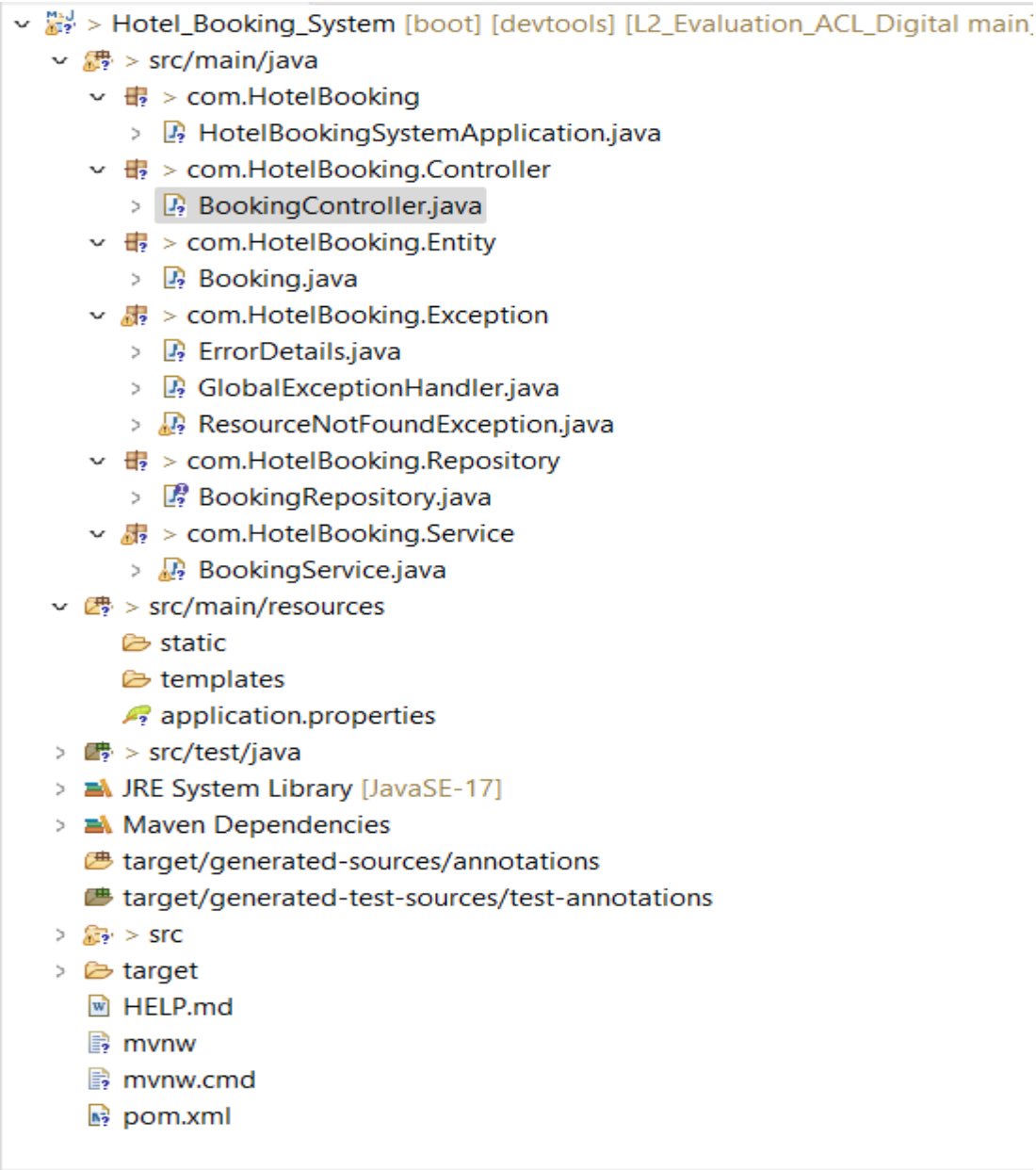
OUTPUT :

```
Problems  @ Javadoc  Declaration  Console X
<terminated> DeadLock [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe  (26-Jun-2024, 6:12:05 pm – 6:12:06 pm) [pid: 15664]
Laptop : lockedScanner
Desktop : lockedPrinter
Laptop : waiting forPrinter.....
Laptop : lockedPrinter
Desktop : waiting forScanner.....
Desktop : lockedScanner
```

Q. 2] Create Hotel Booking APIs in Spring Boot using Post, Put, Delete and Get also

Create a Table as per your understanding and have relevant Error Handling and Exception Handling.

Folder structure of the project:



Above is the Folder structure of the Hotel Booking System using Spring Boot.

1. Project Root

- pom.xml (for Maven projects): This is the Project Object Model file where you define project dependencies, plugins, and configurations for the build process.

2. src Directory

- src/main
  - java

- Contains the main source code of the application.
- com.HotelBooking
  - HotelBookingSystemApplication.java
    - This is the main class that contains the main method to run the Spring Boot application. It is usually annotated with @SpringBootApplication.

```java
package com.HotelBooking;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class HotelBookingSystemApplication {

    public static void main(String[] args) {
        SpringApplication.run(HotelBookingSystemApplication.class, args);
    }

}
```

- This is the base package, and it typically follows the reverse domain name convention. Inside this package, you have several sub-packages:
  - Entity
    - Contains entity classes that represent the database tables.
    - Example: Booking.java

```java
package com.HotelBooking.Entity;

import jakarta.persistence.*;
import lombok.*;

@AllArgsConstructor
@NoArgsConstructor
@Setter
@Getter
@Data
@Entity
public class Booking {
    @Id
    @GeneratedValue(strategy =GenerationType.IDENTITY)
    private Long id;
    private String guestName;
    private String roomType;
    private String checkInDate;
    private String checkoutDate;
    private Double price;
}
```

  - Repository
    - Contains repository interfaces for data access. These usually extend Spring Data JPA interfaces like JpaRepository.
    - Example: BookingRepository.java

```java
package com.HotelBooking.Repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.HotelBooking.Entity.Booking;

public interface BookingRepository extends JpaRepository<Booking,Long> {

}
```

- Service
  - Contains service classes that contain business logic and interact with repositories.
  - Example: BookingService.java

```java
package com.HotelBooking.Service;

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.HotelBooking.Entity.Booking;
import com.HotelBooking.Exception.ResourceNotFoundException;
import com.HotelBooking.Repository.BookingRepository;

@Service
public class BookingService {

    @Autowired
    private BookingRepository bookingRepository;

    public List<Booking> getAllBooking(){
        return bookingRepository.findAll();
    }

    public Optional <Booking> getBookingById (long id){
        return bookingRepository.findById(id);
    }

    public Booking NewBooking(Booking booking) {
        return bookingRepository.save(booking);
    }

    public Booking updateBooking(long id,Booking bookedDetails) {
        Booking booking = bookingRepository.findById(id)
                .orElseThrow(()-> new ResourceNotFoundException("Booking Not Found,Please Enter Correct Id."));
    booking.setGuestName(bookedDetails.getGuestName());
    booking.setRoomType(bookedDetails.getRoomType());
    booking.setCheckInDate(bookedDetails.getCheckInDate());
    booking.setCheckoutDate(bookedDetails.getCheckoutDate());
    booking.setPrice(bookedDetails.getPrice());
    return bookingRepository.save(booking);
    }
```

-

```java
    public void deleteAllBookings() {
        bookingRepository.deleteAll();
    }

    public void deleteBookingById(Long id) {
        Booking booking = bookingRepository.findById(id)
                .orElseThrow(()-> new ResourceNotFoundException("Booking Not Found,Please Enter Valid Id."));
        bookingRepository.deleteById(id);
    }

}
```

- Exception
  - Contains custom exception classes and global exception handlers.
  - Example: ResourceNotFoundException.java, GlobalExceptionHandler.java

```java
package com.HotelBooking.Exception;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class ErrorDetails {

	private String message;
	private String details;
}

```

```java
package com.HotelBooking.Exception;

public class ResourceNotFoundException extends RuntimeException{

    public ResourceNotFoundException(String message) {
        super(message);
    }
}
```

```java
package com.HotelBooking.Exception;

import org.springframework.http.HttpStatus;

@ControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(ResourceNotFoundException.class)
     public ResponseEntity<?> resourceNotFoundException(ResourceNotFoundException ex, WebRequest request) {
        ErrorDetails errorDetails = new ErrorDetails(ex.getMessage(), request.getDescription(false));
        return new ResponseEntity<>(errorDetails, HttpStatus.NOT_FOUND);
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<?> globalExceptionHandler(Exception ex, WebRequest request) {
        ErrorDetails errorDetails = new ErrorDetails(ex.getMessage(), request.getDescription(false));
        return new ResponseEntity<>(errorDetails, HttpStatus.INTERNAL_SERVER_ERROR);
    }

}
```

- Controller
  - Contains REST controllers that handle HTTP requests and map them to service methods.
  - Example: BookingController.java

```java
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.HotelBooking.Entity.Booking;
import com.HotelBooking.Exception.ResourceNotFoundException;
import com.HotelBooking.Service.BookingService;

@RestController
@RequestMapping("/bookings")
public class BookingController {

    @Autowired
    private BookingService bookingService;

    //get all bookings
    @GetMapping
    public ResponseEntity<List<Booking>> getAllBooking(){
        List<Booking> bookings = bookingService.getAllBooking();
        return ResponseEntity.ok().body(bookings);
    }

    //get by id Booking
    @GetMapping("/{id}")
    public ResponseEntity<Booking> getById(@PathVariable Long id){
        Booking booking = bookingService.getBookingById(id)
                .orElseThrow(()-> new ResourceNotFoundException("Booking Not Found,Please Enter Valid id !!") );
        return ResponseEntity.ok().body(booking);
    }
```

```java
    //create new Booking
    @PostMapping
    public ResponseEntity<Booking> newBooking(@RequestBody Booking booking){
        Booking newBookings = bookingService.NewBooking(booking);
        return ResponseEntity.ok(newBookings);
    }

    //update Extisting booking
    @PutMapping("/{id}")
    public ResponseEntity<Booking> updateBooking (@PathVariable long id, @RequestBody Booking booking){
        Booking updateBooking = bookingService.updateBooking(id, booking);
    return ResponseEntity.ok(updateBooking);
    }

    //delete by id
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deteteById(@PathVariable Long id){
        bookingService.deleteBookingById(id);
        return ResponseEntity.noContent().build();
    }

    //delete All booking
    @DeleteMapping
    public ResponseEntity<Void> deleteAll(){
        bookingService.deleteAllBookings();
        return ResponseEntity.noContent().build();
    }

}
```

- resources
  - Contains configuration files and static resources.
  - application.properties or application.yml
    - Configuration file where you define properties like database connection details, server port, etc.

```yaml
#Application Name
spring.application.name=Hotel_Booking_System

#Post Configuration
server.port= 9898

#MySQL Database Cofiguration
spring.datasource.url=jdbc:mysql://localhost:3306/Hotel_Booking
spring.datasource.username=root
spring.datasource.password=root

#hibernate JPA configuration
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

OUTPUT :

1.MySQL Database and Table :

```
Hibernate:
    create table booking (
        id bigint not null auto_increment,
        check_in_date varchar(255),
        checkout_date varchar(255),
        guest_name varchar(255),
        price float(53),
        room_type varchar(255),
        primary key (id)
    ) engine=InnoDB
```

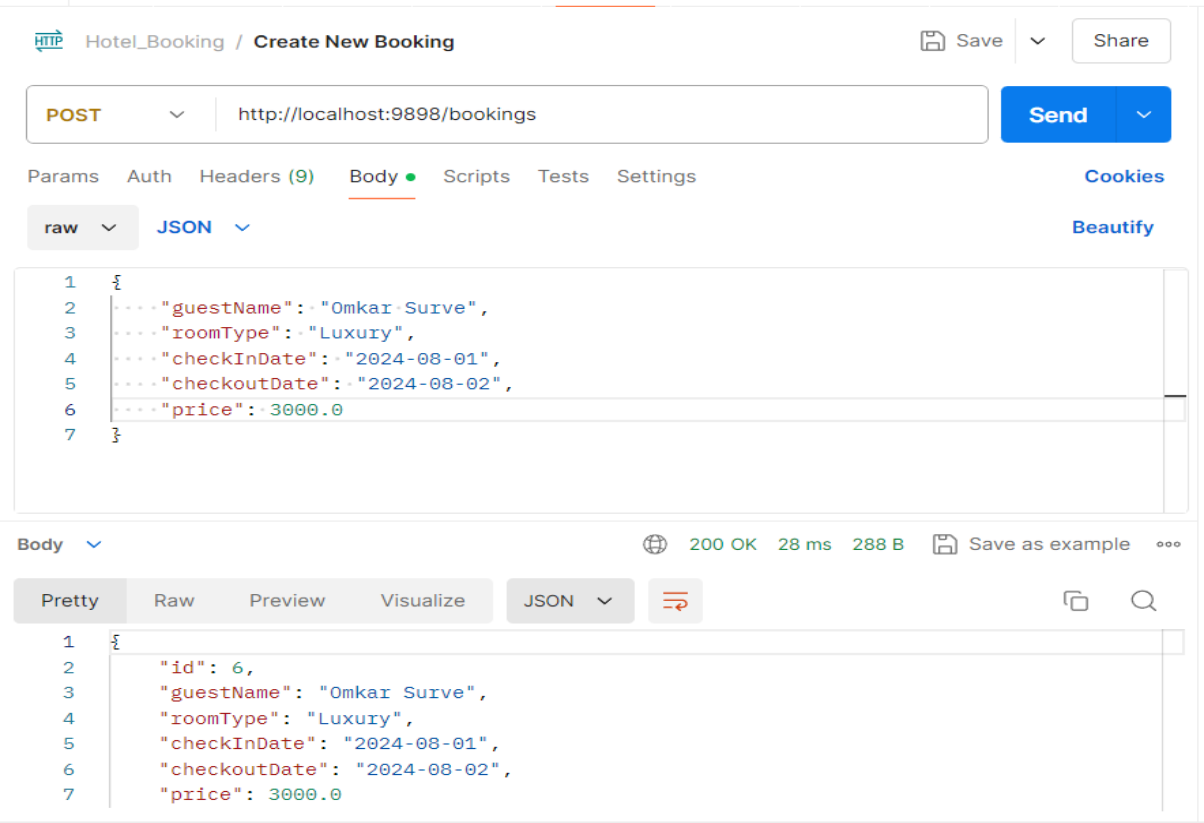
Result Grid | Filter Rows: | Edit: | Export/Imp

id	check_in_date	checkout_date	guest_name	price	room_type
1	2024-07-01	2024-07-02	Suraj Surve	1500	Deluxe
2	2024-08-01	2024-08-02	Saurabh Kapure	2000	Deluxe
3	2024-06-01	2024-06-02	Smit patil	1200	Non-AC
4	2024-06-09	2024-06-10	Omkar Mayekar	2000	Deluxe
5	2024-07-30	2024-07-31	Sahil Sarang	1200	Non-AC
6	2024-08-01	2024-08-02	Omkar Surve	3000	Luxury
NULL	NULL	NULL	NULL	NULL	NULL

API Testing Using Postman Client :

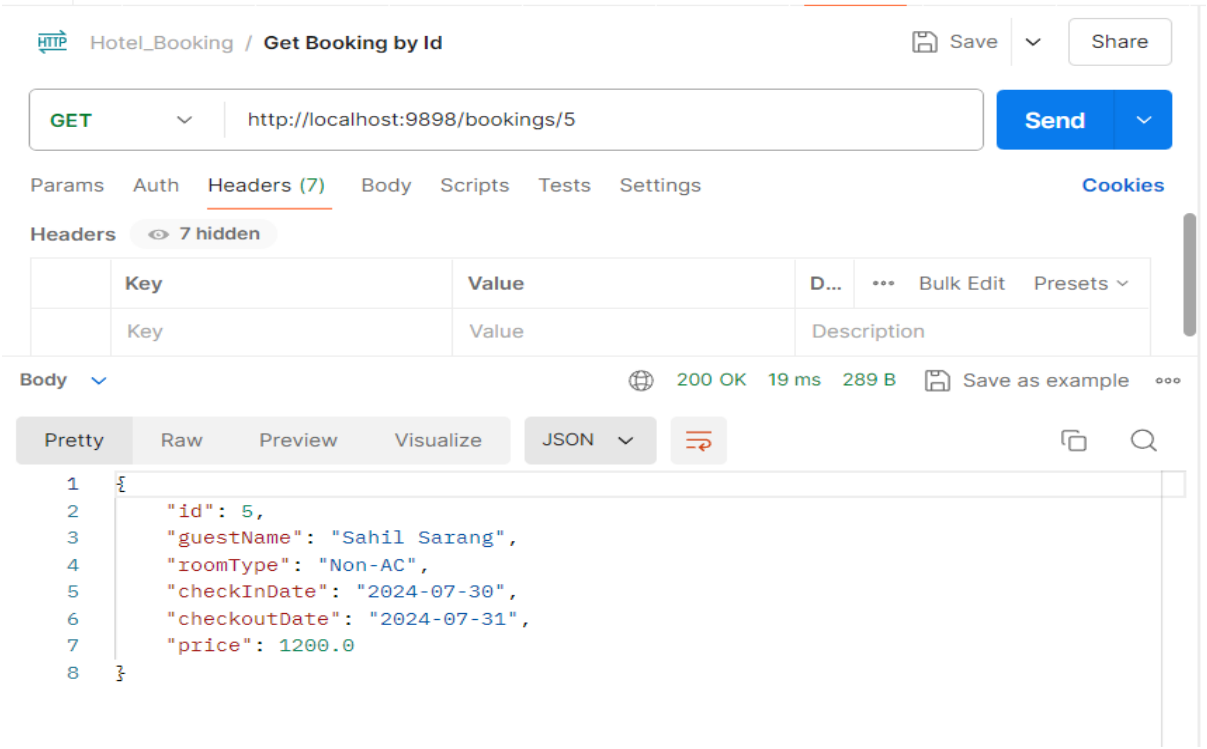
1.Create New Booking :

Api : http://localhost:9898/bookings



2. Get Booking By Id

API : http://localhost:9898/bookings/5



Exception when we call using given invalid Id

API : http://localhost:9898/bookings/7

Hotel_Booking / **Get Booking by Id**

Save | Share

GET http://localhost:9898/bookings/7 Send

Params Auth Headers (7) Body Scripts Tests Settings Cookies

Headers 7 hidden

Key	Value	D...
Key	Value	Description

Bulk Edit Presets

Body 404 Not Found 31 ms 255 B Save as example

Pretty Raw Preview Visualize JSON

```
{
    "message": "Booking Not Found,Please Enter Valid id !!",
    "details": "uri=/bookings/7"
}
```

3. Get All Bookings

Api : http://localhost:9898/bookings

Hotel_Booking / **Get All Bookings**

Save | Share

GET http://localhost:9898/bookings Send

Params Auth Headers (7) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description

Bulk Edit

Body Cookies Headers (5) Test Results 200 OK 197 ms 920 B Save as example

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "guestName": "Suraj Surve",
        "roomType": "Deluxe",
        "checkInDate": "2024-07-01",
        "checkoutDate": "2024-07-02",
        "price": 1500.0
    },
    {
        "id": 2,
        "guestName": "Saurabh Kapure",
        "roomType": "Deluxe",
        "checkInDate": "2024-08-01",
        "checkoutDate": "2024-08-02",
        "price": 2000.0
```

```
{
    "id": 3,
    "guestName": "Smit patil",
    "roomType": "Non-AC",
    "checkInDate": "2024-06-01",
    "checkoutDate": "2024-06-02",
    "price": 1200.0
},
{
    "id": 4,
    "guestName": "Omkar Mayekar",
    "roomType": "Deluxe",
    "checkInDate": "2024-06-09",
    "checkoutDate": "2024-06-10",
    "price": 2000.0
},
```

```
{
    "id": 5,
    "guestName": "Sahil Sarang",
    "roomType": "Non-AC",
    "checkInDate": "2024-07-30",
    "checkoutDate": "2024-07-31",
    "price": 1200.0
},
{
    "id": 6,
    "guestName": "Omkar Surve",
    "roomType": "Luxury",
    "checkInDate": "2024-08-01",
    "checkoutDate": "2024-08-02",
    "price": 3000.0
}
```


4. Update Existing Booking

Api : http://localhost:9898/bookings/6

Existing Booking

Hotel_Booking / **Get Booking by Id**

Save | Share

GET http://localhost:9898/bookings/6 **Send**

Params | Auth | Headers (7) | Body | Scripts | Tests | Settings | Cookies

Headers | 7 hidden

	Key	Value	D...
	Key	Value	Description

Bulk Edit | Presets

Body | 200 OK | 17 ms | 288 B | Save as example

Pretty | Raw | Preview | Visualize | JSON

```
{
    "id": 6,
    "guestName": "Omkar Surve",
    "roomType": "Luxury",
    "checkInDate": "2024-08-01",
    "checkoutDate": "2024-08-02",
    "price": 3000.0
}
```

After Updating Booking:

Hotel_Booking / **update Existing Booking**

Save | Share

PUT http://localhost:9898/bookings/6 **Send**

Params | Auth | Headers (9) | Body | Scripts | Tests | Settings | Cookies

raw | JSON | Beautify

```
{
    "guestName": "Omkar chavan",
    "roomType": "Non-AC",
    "checkInDate": "2024-08-11",
    "checkoutDate": "2024-08-12",
    "price": 1200.0
}
```

Body | Cookies | Headers (5) | Test Results | 200 OK | 83 ms | 289 B | Save as example

Pretty | Raw | Preview | Visualize | JSON

```
{
    "id": 6,
    "guestName": "Omkar chavan",
    "roomType": "Non-AC",
    "checkInDate": "2024-08-11",
    "checkoutDate": "2024-08-12",
    "price": 1200.0
```

For Exception If we are given an invalid Id:

Hotel_Booking / **update Existing Booking**

Save | Share

PUT http://localhost:9898/bookings/7 Send

Params Auth Headers (9) Body Scripts Tests Settings Cookies

raw JSON Beautify

```
{
    "id": 6,
    "guestName": "Omkar Surve",
    "roomType": "Luxury",
    "checkInDate": "2024-08-01",
    "checkoutDate": "2024-08-02",
    "price": 3000.0
}
```

Body Cookies Headers (5) Test Results 404 Not Found 15 ms 255 B Save as example

Pretty Raw Preview Visualize JSON

```
{
    "message": "Booking Not Found,Please Enter Correct Id.",
    "details": "uri=/bookings/7"
}
```

5. Delete Booking By ID:

Hotel_Booking / **delete booking by id**

Save | Share

DELETE http://localhost:9898/bookings/6 Send

Params Auth Headers (9) Body Scripts Tests Settings Cookies

raw JSON Beautify

```
{
    "guestName": "Omkar chavan",
    "roomType": "Non-AC",
    "checkInDate": "2024-08-11",
    "checkoutDate": "2024-08-12",
    "price": 1200.0
}
```

Body 204 No Content 39 ms 112 B Save as example

Pretty Raw Preview Visualize Text

Result Grid | Filter Rows: | Edit: | Export/Import:

id	check_in_date	checkout_date	guest_name	price	room_type
1	2024-07-01	2024-07-02	Suraj Surve	1500	Deluxe
2	2024-08-01	2024-08-02	Saurabh Kapure	2000	Deluxe
3	2024-06-01	2024-06-02	Smit patil	1200	Non-AC
4	2024-06-09	2024-06-10	Omkar Mayekar	2000	Deluxe
5	2024-07-30	2024-07-31	Sahil Sarang	1200	Non-AC
NULL	NULL	NULL	NULL	NULL	NULL