UserServiceTest.java

```java
package com.hotelbooking.userservice.service;

import static org.junit.jupiter.api.Assertions.assertEquals;

@SpringBootTest(properties = "eureka.client.enabled=false")
public class UserServiceTest {

    @InjectMocks
    private UserService userService = new UserServiceImpl();

    @Mock
    private UserDao userDao;

    @Test
    public void getUserDetailsTest() {

        User user = new User();

        user.setUserName("Rohit");
        user.setMobile("9856231470");
        user.setUserId(100);
        user.setEmail("gdcgf@gmail.com");
        user.setAddress("ehgfchkuyjh");
        user.setRole("customer");

        when(userDao.findById(100)).thenReturn(Optional.of(user));

        User testuser = userService.getUserById(100);

        assertEquals("Rohit", testuser.getUserName());

    }

    @Test
    public void testGetUserDetailsException() {

        when(userDao.findById(101)).thenThrow(new ResourceNotFoundException("User details not found with id 101"));
        assertThrows(ResourceNotFoundException.class, () -> userService.getUserById(101));
    }

    @Test
    public void getAllUserDetailsTest() {

```

Problems @ Jav

hotel-service - Hote

```
2024-05-06T17:1
2024-05-06T17:1
Hibernate: sele
Hibernate: sele
Hibernate: sele
Hibernate: sele
Hibernate: sele
Hibernate: sele
Hibernate: sele
Hibernate: sele
Hibernate: sele
2024-05-06T17:2
2024-05-06T17:2
```

UserServiceTest.java

```java
    @Test
    public void getAllUserDetailsTest() {

        User user = new User();

        user.setUserName("Rohit");
        user.setMobile("9856231470");
        user.setUserId(100);
        user.setEmail("gdcgf@gmail.com");
        user.setAddress("ehgfchkuyjh");
        user.setRole("customer");

      User user1 = new User();

        user1.setUserName("Rohit");
        user1.setMobile("9856231470");
        user1.setUserId(101);
        user1.setEmail("gdcgf@gmail.com");
        user1.setAddress("ehgfchkuyjh");
        user1.setRole("customer");

        User user2 = new User();

        user2.setUserName("Rohit");
        user2.setMobile("9856231470");
        user2.setUserId(102);
        user2.setEmail("gdcgf@gmail.com");
        user2.setAddress("ehgfchkuyjh");
        user2.setRole("customer");

        List<User> users = new ArrayList<>();

        users.add(user2);
        users.add(user1);
        users.add(user);

        when(userDao.findAll()).thenReturn(users);

        List<User> testlist =userService.getAllUsers();

        assertEquals(3, testlist.size());

    }
```

Problems Javadoc De

server - ServerApplication [Sprin

```
2024-05-06T17:17:05.910
2024-05-06T17:18:05.914
2024-05-06T17:19:05.923
2024-05-06T17:20:05.925
2024-05-06T17:21:05.934
2024-05-06T17:22:05.943
2024-05-06T17:23:05.956
2024-05-06T17:24:05.958
2024-05-06T17:25:05.969
2024-05-06T17:26:05.971
2024-05-06T17:27:05.973
2024-05-06T17:28:05.973
2024-05-06T17:29:05.974
2024-05-06T17:30:05.974
```

```java
    @Test
    public void deleteUserTest() {

        User user = new User();

        user.setUserName("Rohit");
        user.setMobile("9856231470");
        user.setUserId(100);
        user.setEmail("gdcgf@gmail.com");
        user.setAddress("ehgfchkuyjh");
        user.setRole("customer");

        doNothing().when(userDao).delete(user);;


        when(userDao.findById(100)).thenReturn(Optional.of(user));
        userService.deleteUser(100);

        verify(userDao, times(1)).findById(100);

        verify(userDao, times(1)).delete(user);

    }

    @Test
    public void deleteUserExceptionTest() {
        User user = new User();

        when(userDao.findById(100)).thenThrow(new ResourceNotFoundException("User Details Not found with id 100"));

        assertThrows(ResourceNotFoundException.class, () -> userService.deleteUser(100));
        verify(userDao, times(0)).delete(user);

    }

}
```

```
2024-05-06T16:49:10.1
2024-05-06T16:54:10.2
2024-05-06T16:59:10.2
2024-05-06T17:04:10.2
2024-05-06T17:09:10.2
2024-05-06T17:14:10.2
2024-05-06T17:19:10.2
Hibernate: select b1_
2024-05-06T17:24:10.2
2024-05-06T17:29:10.2
```

```java
package com.hotelbooking.paymentservice.service;

import static org.junit.jupiter.api.Assertions.assertEquals;

@SpringBootTest(properties = "eureka.client.enabled=false")
public class PaymentServiceTest {

    @Mock
    private PaymentDao paymentDao;

    @Mock
    private BookingServiceConsumer bookingService;

    @InjectMocks
    private PaymentServiceImpl paymentService;

    @Test
    public void testMakePayment() {
        Payments payment = new Payments();
        payment.setBookingId(1);
        payment.setPAmount(100.0);

        BookingResponse bookingResponse = new BookingResponse();
        bookingResponse.setAmount(100.0);

        when(bookingService.getBookingDetailsById(1)).thenReturn(bookingResponse);
        when(paymentDao.existsByBookingId(1)).thenReturn(false);
        when(paymentDao.save(payment)).thenReturn(payment);

        PaymentResponse response = paymentService.makePayment(payment);

        assertEquals("Payment Successfull", response.getMessage());
        assertEquals(bookingResponse, response.getBookingResponse());
    }

    @Test
    public void testMakePaymentPaymentAlreadyExistException() {
        Payments payment = new Payments();
        payment.setBookingId(1);

        when(paymentDao.existsByBookingId(1)).thenReturn(true);

        assertThrows(PaymentFailedException.class, () -> {
```

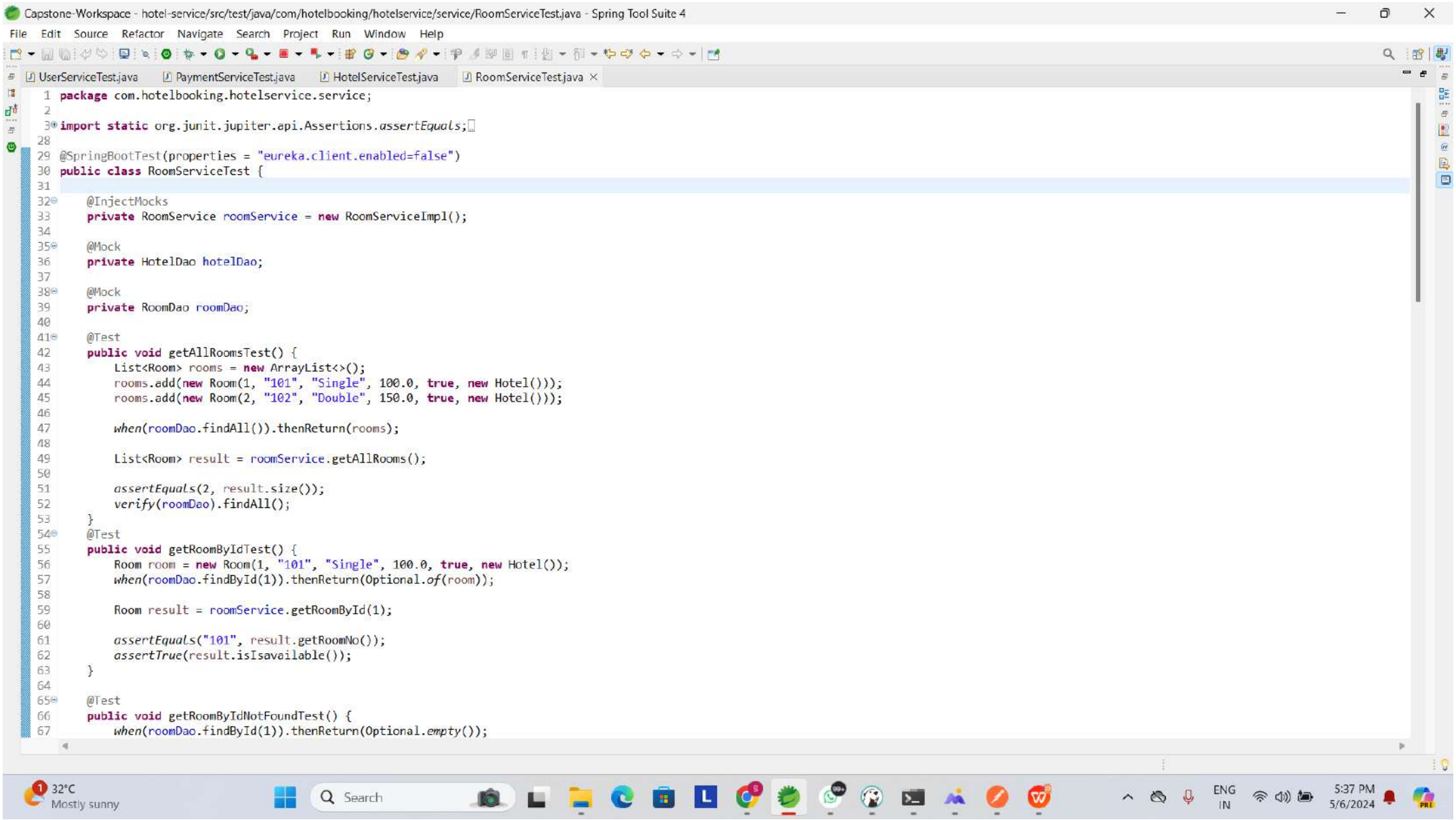
```java
@Test
public void testMakePayment() {
    Payments payment = new Payments();
    payment.setBookingId(1);
    payment.setPAmount(100.0);

    BookingResponse bookingResponse = new BookingResponse();
    bookingResponse.setAmount(100.0);

    when(bookingService.getBookingDetailsById(1)).thenReturn(bookingResponse);
    when(paymentDao.existsByBookingId(1)).thenReturn(false);
    when(paymentDao.save(payment)).thenReturn(payment);

    PaymentResponse response = paymentService.makePayment(payment);

    assertEquals("Payment Successfull", response.getMessage());
    assertEquals(bookingResponse, response.getBookingResponse());
}

@Test
public void testMakePaymentPaymentAlreadyExistException() {
    Payments payment = new Payments();
    payment.setBookingId(1);

    when(paymentDao.existsByBookingId(1)).thenReturn(true);

    assertThrows(PaymentFailedException.class, () -> {
        paymentService.makePayment(payment);
    });
}

@Test
public void testViewPaymentsById() {
    Payments payment = new Payments();
    payment.setPaymentId(1);

    when(paymentDao.findById(1)).thenReturn(Optional.of(payment));

    Payments result = paymentService.viewPaymentsById(1);

    assertEquals(payment, result);
}
```

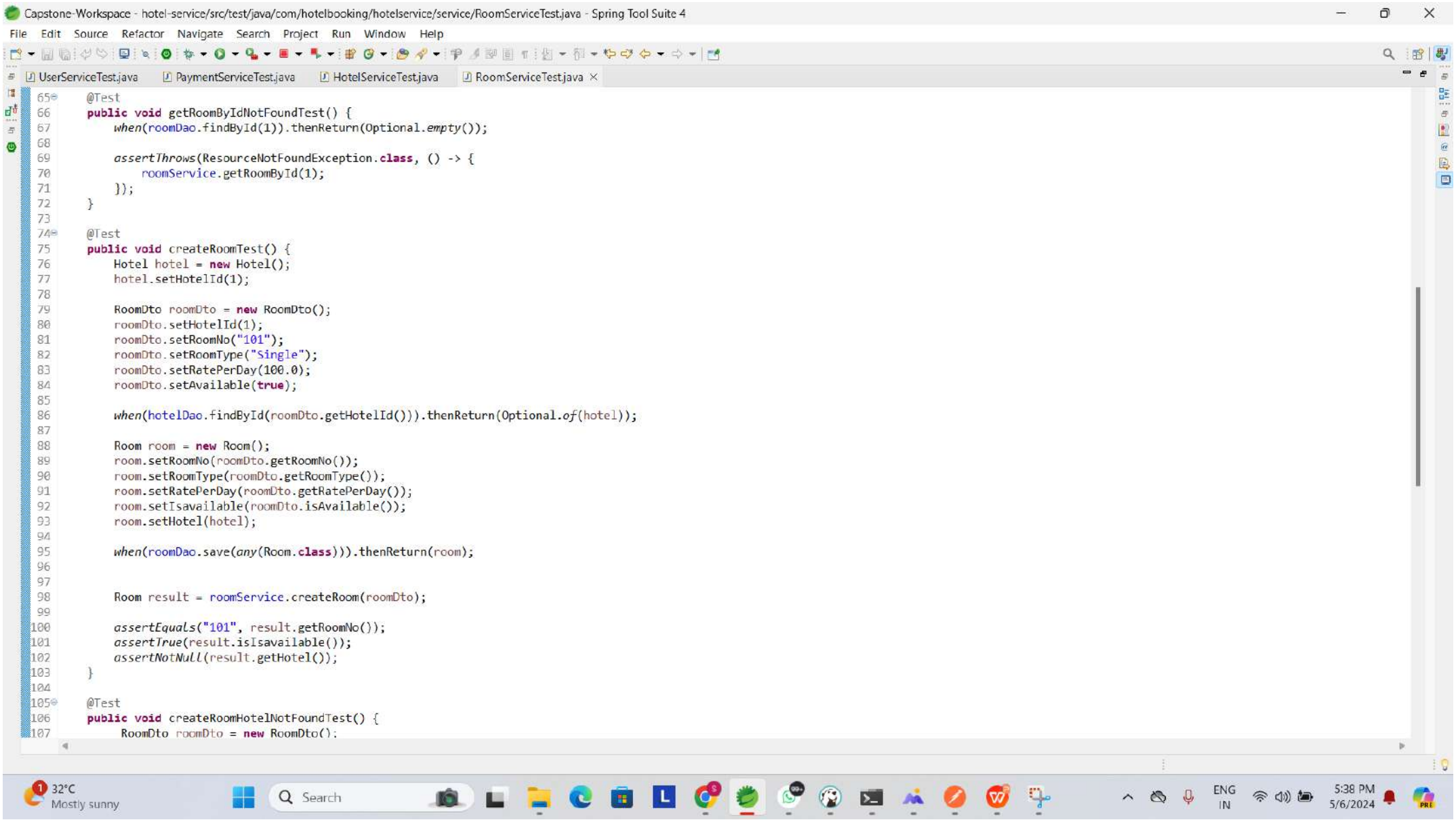
```
    }

    @Test
    public void testViewPaymentsByIdPaymentNotFoundException() {
        when(paymentDao.findById(1)).thenReturn(Optional.empty());

        assertThrows(ResourceNotFoundException.class, () -> {
            paymentService.viewPaymentsById(1);
        });
    }

    @Test
    public void testViewAllPayments() {
        List<Payments> payments = new ArrayList<>();
        payments.add(new Payments());
        payments.add(new Payments());

        when(paymentDao.findAll()).thenReturn(payments);

        List<Payments> result = paymentService.viewAllPayments();

        assertEquals(payments.size(), result.size());
    }

    @Test
    public void testMakePaymentAmountMismatchException() {
        Payments payment = new Payments();
        payment.setBookingId(1);
        payment.setPAmount(100.0);

        BookingResponse bookingResponse = new BookingResponse();
        bookingResponse.setAmount(200.0);

        when(bookingService.getBookingDetailsById(1)).thenReturn(bookingResponse);

        assertThrows(PaymentFailedException.class, () -> {
            paymentService.makePayment(payment);
        });
    }

}
```

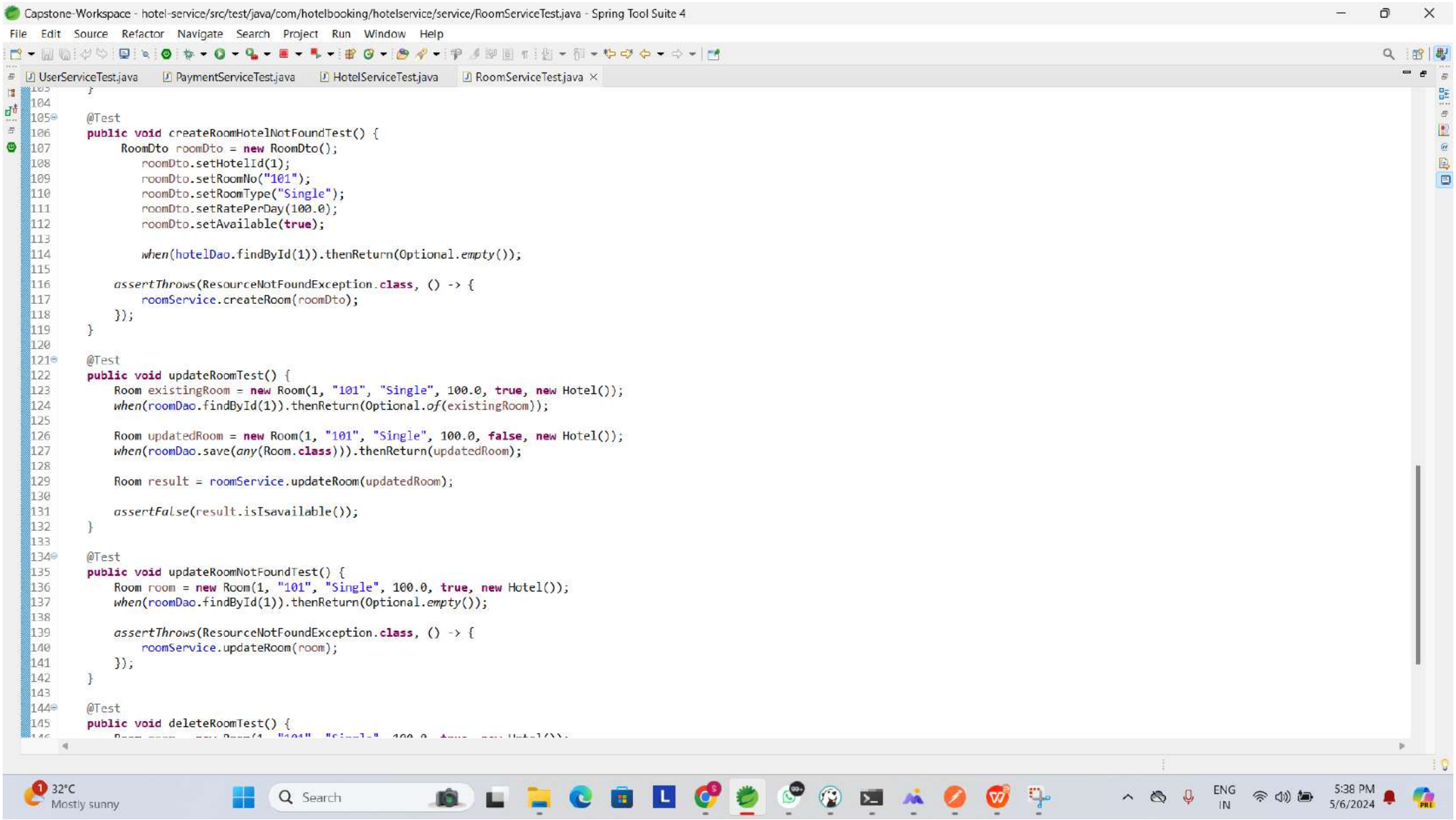
```java
package com.hotelbooking.hotelservice.service;

import static org.junit.jupiter.api.Assertions.assertEquals;

@SpringBootTest(properties = "eureka.client.enabled=false")
public class HotelServiceTest {

    @InjectMocks
    private HotelService hotelService = new HotelServiceImpl();

    @Mock
    private HotelDao hotelDao;

    @Test
    public void getAllHotelsTest() {
        List<Hotel> hotels = new ArrayList<>();
        hotels.add(new Hotel(1, "New York", "Hotel California", "Some address", "Nice place", 200.0,
                "contact@hotelcal.com", "1234567890", "0987654321", "www.hotelcal.com", new ArrayList<Room>()));
        hotels.add(new Hotel(2, "Los Angeles", "Hotel New York", "Another address", "Great stay", 150.0,
                "info@hotelnyc.com", "1112223330", "4445556661", "www.hotelnyc.com", new ArrayList<Room>()));

        when(hotelDao.findAll()).thenReturn(hotels);

        List<Hotel> result = hotelService.getAllHotels();

        assertEquals(2, result.size());
        assertEquals("New York", result.get(0).getCity());
        verify(hotelDao).findAll();
    }

    @Test
    public void getHotelByIdTest() {
        Hotel hotel = new Hotel(1, "New York", "Hotel California", "Some address", "Nice place", 200.0,
                "contact@hotelcal.com", "1234567890", "0987654321", "www.hotelcal.com", new ArrayList<Room>());
        when(hotelDao.findById(1)).thenReturn(Optional.of(hotel));

        Hotel result = hotelService.getHotelById(1);

        assertEquals("Hotel California", result.getHotelName());
        assertEquals("New York", result.getCity());
    }

    @Test
```

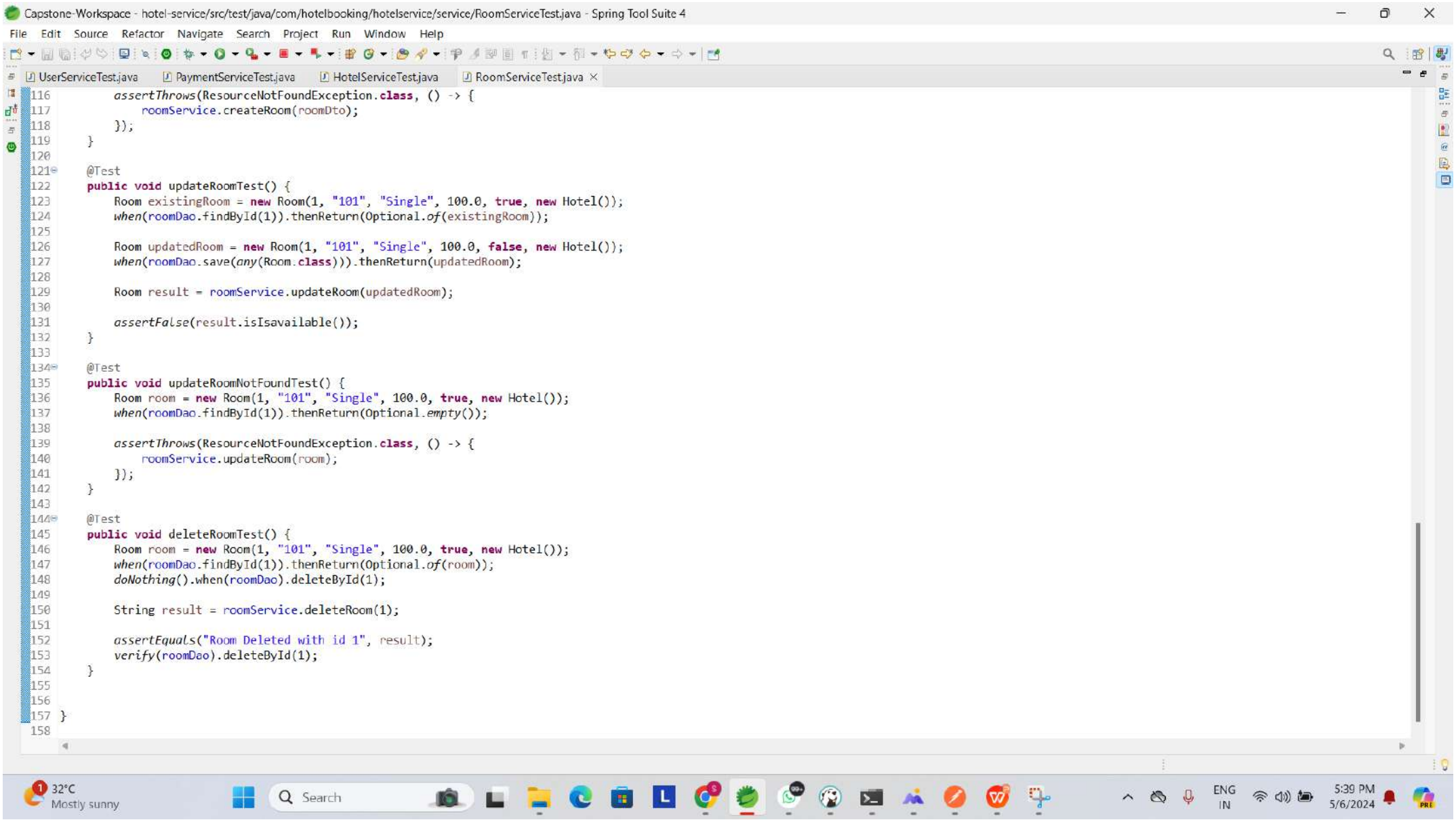
```java
@Test
public void getHotelByIdNotFoundTest() {
    when(hotelDao.findById(1)).thenReturn(Optional.empty());

    assertThrows(ResourceNotFoundException.class, () -> {
        hotelService.getHotelById(1);
    });
}

@Test
public void createHotelTest() {
    Hotel hotel = new Hotel(1, "New York", "Hotel California", "Some address", "Nice place", 200.0,
            "contact@hotelcal.com", "1234567890", "0987654321", "www.hotelcal.com", new ArrayList<Room>());
    when(hotelDao.save(any(Hotel.class))).thenReturn(hotel);

    Hotel result = hotelService.createHotel(hotel);

    assertEquals("Hotel California", result.getHotelName());
    verify(hotelDao).save(hotel);
}

@Test
public void updateHotelTest() {
    Hotel existingHotel = new Hotel(1, "New York", "Hotel California", "Some address", "Nice place", 200.0,
            "contact@hotelcal.com", "1234567890", "0987654321", "www.hotelcal.com", new ArrayList<Room>());
    when(hotelDao.findById(1)).thenReturn(Optional.of(existingHotel));
    when(hotelDao.save(any(Hotel.class))).thenReturn(existingHotel);

    Hotel result = hotelService.updateHotel(existingHotel);

    assertEquals("Hotel California", result.getHotelName());
    verify(hotelDao).save(existingHotel);
}

@Test
```

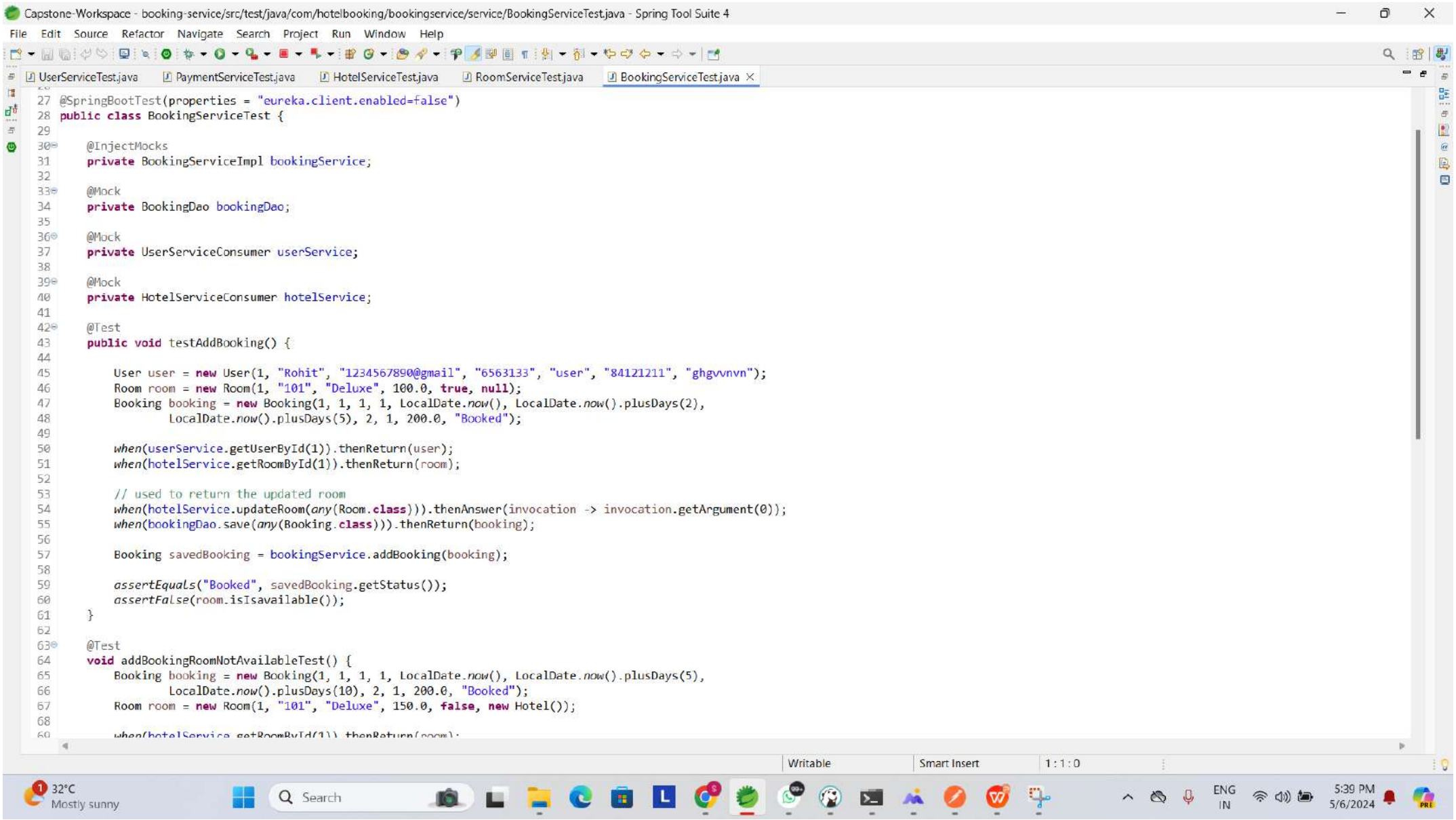
```java
@Test
public void updateHotelNotFoundTest() {
    Hotel newHotel = new Hotel(1, "New York", "Hotel California", "Some address", "Nice place", 200.0,
            "contact@hotelcal.com", "1234567890", "0987654321", "www.hotelcal.com", new ArrayList<Room>());
    when(hotelDao.findById(1)).thenReturn(Optional.empty());

    assertThrows(ResourceNotFoundException.class, () -> {
        hotelService.updateHotel(newHotel);
    });
}

@Test
public void deleteHotelTest() {
    Hotel existingHotel = new Hotel(1, "New York", "Hotel California", "Some address", "Nice place", 200.0,
            "contact@hotelcal.com", "1234567890", "0987654321", "www.hotelcal.com", new ArrayList<Room>());
    when(hotelDao.findById(1)).thenReturn(Optional.of(existingHotel));
    doNothing().when(hotelDao).deleteById(1);

    String result = hotelService.deleteHotel(1);

    assertEquals("Hotel deleted with id 1", result);
    verify(hotelDao).deleteById(1);
}

@Test
public void deleteHotelNotFoundTest() {
    when(hotelDao.findById(1)).thenReturn(Optional.empty());

    assertThrows(ResourceNotFoundException.class, () -> {
        hotelService.deleteHotel(1);
    });
}

}
```

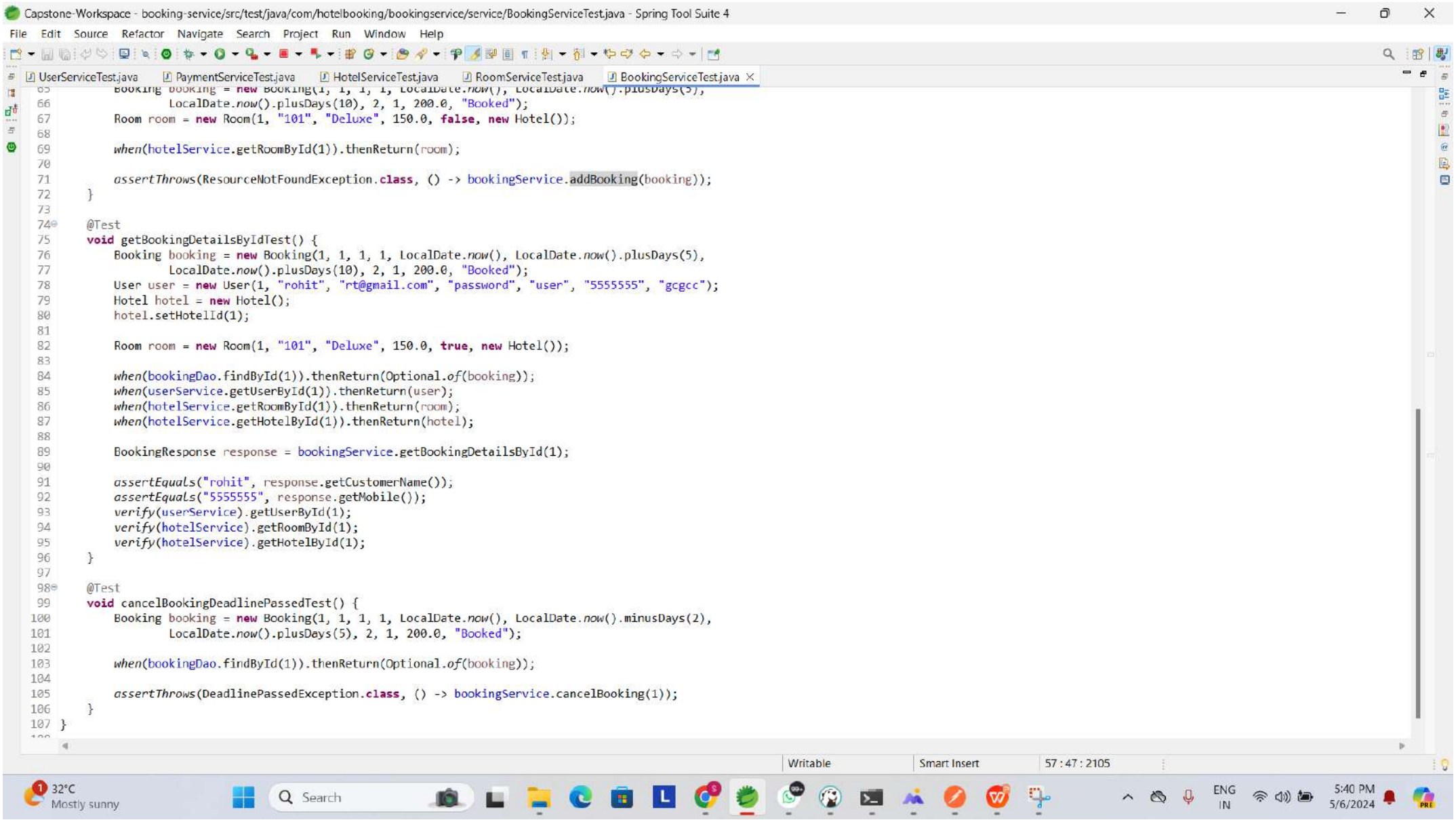

Capstone-Workspace - hotel-service/src/test/java/com/hotelbooking/hotelservice/service/RoomServiceTest.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
UserServiceTest.java
PaymentServiceTest.java
HotelServiceTest.java
RoomServiceTest.java
package com.hotelbooking.hotelservice.service;
import static org.junit.jupiter.api.Assertions.assertEquals;
@SpringBootTest(properties = "eureka.client.enabled=false")
public class RoomServiceTest {
@InjectMocks
private RoomService roomService = new RoomServiceImpl();
@Mock
private HotelDao hotelDao;
@Mock
private RoomDao roomDao;
@Test
public void getAllRoomsTest() {
List<Room> rooms = new ArrayList<>();
rooms.add(new Room(1, "101", "Single", 100.0, true, new Hotel()));
rooms.add(new Room(2, "102", "Double", 150.0, true, new Hotel()));
when(roomDao.findAll()).thenReturn(rooms);
List<Room> result = roomService.getAllRooms();
assertEquals(2, result.size());
verify(roomDao).findAll();
}
@Test
public void getRoomByIdTest() {
Room room = new Room(1, "101", "Single", 100.0, true, new Hotel());
when(roomDao.findById(1)).thenReturn(Optional.of(room));
Room result = roomService.getRoomById(1);
assertEquals("101", result.getRoomNo());
assertTrue(result.isIsavailable());
}
@Test
public void getRoomByIdNotFoundTest() {
when(roomDao.findById(1)).thenReturn(Optional.empty());
32°C
Mostly sunny
Search
ENG
IN
5:37 PM
5/6/2024


Capstone-Workspace - hotel-service/src/test/java/com/hotelbooking/hotelservice/service/RoomServiceTest.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
UserServiceTest.java
PaymentServiceTest.java
HotelServiceTest.java
RoomServiceTest.java
@Test
public void getRoomByIdNotFoundTest() {
when(roomDao.findById(1)).thenReturn(Optional.empty());
assertThrows(ResourceNotFoundException.class, () -> {
roomService.getRoomById(1);
});
}
@Test
public void createRoomTest() {
Hotel hotel = new Hotel();
hotel.setHotelId(1);
RoomDto roomDto = new RoomDto();
roomDto.setHotelId(1);
roomDto.setRoomNo("101");
roomDto.setRoomType("Single");
roomDto.setRatePerDay(100.0);
roomDto.setAvailable(true);
when(hotelDao.findById(roomDto.getHotelId())).thenReturn(Optional.of(hotel));
Room room = new Room();
room.setRoomNo(roomDto.getRoomNo());
room.setRoomType(roomDto.getRoomType());
room.setRatePerDay(roomDto.getRatePerDay());
room.setIsavailable(roomDto.isAvailable());
room.setHotel(hotel);
when(roomDao.save(any(Room.class))).thenReturn(room);
Room result = roomService.createRoom(roomDto);
assertEquals("101", result.getRoomNo());
assertTrue(result.isIsavailable());
assertNotNull(result.getHotel());
}
@Test
public void createRoomHotelNotFoundTest() {
RoomDto roomDto = new RoomDto();
32°C
Mostly sunny
Search
ENG
IN
5:38 PM
5/6/2024


Capstone-Workspace - hotel-service/src/test/java/com/hotelbooking/hotelservice/service/RoomServiceTest.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
UserServiceTest.java
PaymentServiceTest.java
HotelServiceTest.java
RoomServiceTest.java
}
@Test
public void createRoomHotelNotFoundTest() {
RoomDto roomDto = new RoomDto();
roomDto.setHotelId(1);
roomDto.setRoomNo("101");
roomDto.setRoomType("Single");
roomDto.setRatePerDay(100.0);
roomDto.setAvailable(true);
when(hotelDao.findById(1)).thenReturn(Optional.empty());
assertThrows(ResourceNotFoundException.class, () -> {
roomService.createRoom(roomDto);
});
}
@Test
public void updateRoomTest() {
Room existingRoom = new Room(1, "101", "Single", 100.0, true, new Hotel());
when(roomDao.findById(1)).thenReturn(Optional.of(existingRoom));
Room updatedRoom = new Room(1, "101", "Single", 100.0, false, new Hotel());
when(roomDao.save(any(Room.class))).thenReturn(updatedRoom);
Room result = roomService.updateRoom(updatedRoom);
assertFalse(result.isIsavailable());
}
@Test
public void updateRoomNotFoundTest() {
Room room = new Room(1, "101", "Single", 100.0, true, new Hotel());
when(roomDao.findById(1)).thenReturn(Optional.empty());
assertThrows(ResourceNotFoundException.class, () -> {
roomService.updateRoom(room);
});
}
@Test
public void deleteRoomTest() {
32°C
Mostly sunny
Search
ENG
IN
5:38 PM
5/6/2024


Capstone-Workspace - hotel-service/src/test/java/com/hotelbooking/hotelservice/service/RoomServiceTest.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
UserServiceTest.java
PaymentServiceTest.java
HotelServiceTest.java
RoomServiceTest.java
assertThrows(ResourceNotFoundException.class, () -> {
roomService.createRoom(roomDto);
});
}
@Test
public void updateRoomTest() {
Room existingRoom = new Room(1, "101", "Single", 100.0, true, new Hotel());
when(roomDao.findById(1)).thenReturn(Optional.of(existingRoom));
Room updatedRoom = new Room(1, "101", "Single", 100.0, false, new Hotel());
when(roomDao.save(any(Room.class))).thenReturn(updatedRoom);
Room result = roomService.updateRoom(updatedRoom);
assertFalse(result.isIsavailable());
}
@Test
public void updateRoomNotFoundTest() {
Room room = new Room(1, "101", "Single", 100.0, true, new Hotel());
when(roomDao.findById(1)).thenReturn(Optional.empty());
assertThrows(ResourceNotFoundException.class, () -> {
roomService.updateRoom(room);
});
}
@Test
public void deleteRoomTest() {
Room room = new Room(1, "101", "Single", 100.0, true, new Hotel());
when(roomDao.findById(1)).thenReturn(Optional.of(room));
doNothing().when(roomDao).deleteById(1);
String result = roomService.deleteRoom(1);
assertEquals("Room Deleted with id 1", result);
verify(roomDao).deleteById(1);
}
}
32°C
Mostly sunny
Search
ENG
IN
5:39 PM
5/6/2024


Capstone-Workspace - booking-service/src/test/java/com/hotelbooking/bookingservice/service/BookingServiceTest.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
UserServiceTest.java
PaymentServiceTest.java
HotelServiceTest.java
RoomServiceTest.java
BookingServiceTest.java
@SpringBootTest(properties = "eureka.client.enabled=false")
public class BookingServiceTest {
@InjectMocks
private BookingServiceImpl bookingService;
@Mock
private BookingDao bookingDao;
@Mock
private UserServiceConsumer userService;
@Mock
private HotelServiceConsumer hotelService;
@Test
public void testAddBooking() {
User user = new User(1, "Rohit", "1234567890@gmail", "6563133", "user", "84121211", "ghgvvnvn");
Room room = new Room(1, "101", "Deluxe", 100.0, true, null);
Booking booking = new Booking(1, 1, 1, 1, LocalDate.now(), LocalDate.now().plusDays(2),
LocalDate.now().plusDays(5), 2, 1, 200.0, "Booked");
when(userService.getUserById(1)).thenReturn(user);
when(hotelService.getRoomById(1)).thenReturn(room);
// used to return the updated room
when(hotelService.updateRoom(any(Room.class))).thenAnswer(invocation -> invocation.getArgument(0));
when(bookingDao.save(any(Booking.class))).thenReturn(booking);
Booking savedBooking = bookingService.addBooking(booking);
assertEquals("Booked", savedBooking.getStatus());
assertFalse(room.isIsavailable());
}
@Test
void addBookingRoomNotAvailableTest() {
Booking booking = new Booking(1, 1, 1, 1, LocalDate.now(), LocalDate.now().plusDays(5),
LocalDate.now().plusDays(10), 2, 1, 200.0, "Booked");
Room room = new Room(1, "101", "Deluxe", 150.0, false, new Hotel());
Writable
Smart Insert
1 : 1 : 0
32°C
Mostly sunny
Search
ENG
IN
5:39 PM
5/6/2024


Capstone-Workspace - booking-service/src/test/java/com/hotelbooking/bookingservice/service/BookingServiceTest.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
UserServiceTest.java
PaymentServiceTest.java
HotelServiceTest.java
RoomServiceTest.java
BookingServiceTest.java
Booking booking = new Booking(1, 1, 1, 1, LocalDate.now(), LocalDate.now().plusDays(5),
LocalDate.now().plusDays(10), 2, 1, 200.0, "Booked");
Room room = new Room(1, "101", "Deluxe", 150.0, false, new Hotel());
when(hotelService.getRoomById(1)).thenReturn(room);
assertThrows(ResourceNotFoundException.class, () -> bookingService.addBooking(booking));
}
@Test
void getBookingDetailsByIdTest() {
Booking booking = new Booking(1, 1, 1, 1, LocalDate.now(), LocalDate.now().plusDays(5),
LocalDate.now().plusDays(10), 2, 1, 200.0, "Booked");
User user = new User(1, "rohit", "rt@gmail.com", "password", "user", "5555555", "gcgcc");
Hotel hotel = new Hotel();
hotel.setHotelId(1);
Room room = new Room(1, "101", "Deluxe", 150.0, true, new Hotel());
when(bookingDao.findById(1)).thenReturn(Optional.of(booking));
when(userService.getUserById(1)).thenReturn(user);
when(hotelService.getRoomById(1)).thenReturn(room);
when(hotelService.getHotelById(1)).thenReturn(hotel);
BookingResponse response = bookingService.getBookingDetailsById(1);
assertEquals("rohit", response.getCustomerName());
assertEquals("5555555", response.getMobile());
verify(userService).getUserById(1);
verify(hotelService).getRoomById(1);
verify(hotelService).getHotelById(1);
}
@Test
void cancelBookingDeadlinePassedTest() {
Booking booking = new Booking(1, 1, 1, 1, LocalDate.now(), LocalDate.now().minusDays(2),
LocalDate.now().plusDays(5), 2, 1, 200.0, "Booked");
when(bookingDao.findById(1)).thenReturn(Optional.of(booking));
assertThrows(DeadlinePassedException.class, () -> bookingService.cancelBooking(1));
}
}
Writable
Smart Insert
57 : 47 : 2105
32°C
Mostly sunny
Search
ENG
IN
5:40 PM
5/6/2024