

```
1 package com.hotelbooking.userservice.service;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4
22
23 @SpringBootTest(properties = "eureka.client.enabled=false")
24 public class UserServiceTest {
25
26     @InjectMocks
27     private UserService userService = new UserServiceImpl();
28
29     @Mock
30     private UserDao userDao;
31
32     @Test
33     public void getUserDetailsTest() {
34
35         User user = new User();
36
37         user.setUserName("Rohit");
38         user.setMobile("9856231470");
39         user.setUserId(100);
40         user.setEmail("gdcgf@gmail.com");
41         user.setAddress("ehgfchkuyjh");
42         user.setRole("customer");
43
44         when(userDao.findById(100)).thenReturn(Optional.of(user));
45
46         User testuser = userService.getUserById(100);
47
48         assertEquals("Rohit", testuser.getUserName());
49     }
50
51
52     @Test
53     public void testGetUserDetailsException() {
54
55         when(userDao.findById(101)).thenReturn(new ResourceNotFoundException("User details not found with id 101"));
56         assertThrows(ResourceNotFoundException.class, () -> userService.getUserById(101));
57     }
58
59     @Test
60     public void getAllUserDetailsTest() {
61
```

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

Hibernate: sele

2024-05-06T17:2

2024-05-06T17:2

UserServiceTest.java ×

```

59 @Test
60 public void getAllUserDetailsTest() {
61
62     User user = new User();
63
64     user.setUserName("Rohit");
65     user.setMobile("9856231470");
66     user.setUserId(100);
67     user.setEmail("gdcgf@gmail.com");
68     user.setAddress("ehgfchkuyjh");
69     user.setRole("customer");
70
71     User user1 = new User();
72
73     user1.setUserName("Rohit");
74     user1.setMobile("9856231470");
75     user1.setUserId(101);
76     user1.setEmail("gdcgf@gmail.com");
77     user1.setAddress("ehgfchkuyjh");
78     user1.setRole("customer");
79
80     User user2 = new User();
81
82     user2.setUserName("Rohit");
83     user2.setMobile("9856231470");
84     user2.setUserId(102);
85     user2.setEmail("gdcgf@gmail.com");
86     user2.setAddress("ehgfchkuyjh");
87     user2.setRole("customer");
88
89     List<User> users = new ArrayList<>();
90
91     users.add(user2);
92     users.add(user1);
93     users.add(user);
94
95     when(userDao.findAll()).thenReturn(users);
96
97     List<User> testlist =userService.getAllUsers();
98
99     assertEquals(3, testlist.size());
100
101 }

```

Problems @ Javadoc De

server - ServerApplication [Spring  
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-

```

@Test
public void deleteUserTest() {

    User user = new User();

    user.setName("Rohit");
    user.setMobile("9856231470");
    user.setId(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)).thenReturn(Optional.empty());

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

}

```

```

2024-05-06T10:44:10.1
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_1
2024-05-06T17:24:10.2
2024-05-06T17:29:10.2

```

```

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, () -> {

```

```

@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);  
    });  
}
```

```
}
```

```

1 package com.hotelbooking.hotel.service.service;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4
23
24 @SpringBootTest(properties = "eureka.client.enabled=false")
25 public class HotelServiceTest {
26
27     @InjectMocks
28     private HotelService hotelService = new HotelServiceImpl();
29
30     @Mock
31     private HotelDao hotelDao;
32
33     @Test
34     public void getAllHotelsTest() {
35         List<Hotel> hotels = new ArrayList<>();
36         hotels.add(new Hotel(1, "New York", "Hotel California", "Some address", "Nice place", 200.0,
37             "contact@hotelcal.com", "1234567890", "0987654321", "www.hotelcal.com", new ArrayList<Room>()));
38         hotels.add(new Hotel(2, "Los Angeles", "Hotel New York", "Another address", "Great stay", 150.0,
39             "info@hotelnyc.com", "1112223330", "4445556661", "www.hotelnyc.com", new ArrayList<Room>()));
40
41         when(hotelDao.findAll()).thenReturn(hotels);
42
43         List<Hotel> result = hotelService.getAllHotels();
44
45         assertEquals(2, result.size());
46         assertEquals("New York", result.get(0).getCity());
47         verify(hotelDao).findAll();
48     }
49
50     @Test
51     public void getHotelByIdTest() {
52         Hotel hotel = new Hotel(1, "New York", "Hotel California", "Some address", "Nice place", 200.0,
53             "contact@hotelcal.com", "1234567890", "0987654321", "www.hotelcal.com", new ArrayList<Room>());
54         when(hotelDao.findById(1)).thenReturn(Optional.of(hotel));
55
56         Hotel result = hotelService.getHotelById(1);
57
58         assertEquals("Hotel California", result.getHotelName());
59         assertEquals("New York", result.getCity());
60     }
61
62     @Test

```

```

@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

```



```

@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

```
1 package com.hotelbooking.hotelservice.service;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4
28
29 @SpringBootTest(properties = "eureka.client.enabled=false")
30 public class RoomServiceTest {
31
32 @InjectMocks
33 private RoomService roomService = new RoomServiceImpl();
34
35 @Mock
36 private HotelDao hotelDao;
37
38 @Mock
39 private RoomDao roomDao;
40
41 @Test
42 public void getAllRoomsTest() {
43     List<Room> rooms = new ArrayList<>();
44     rooms.add(new Room(1, "101", "Single", 100.0, true, new Hotel()));
45     rooms.add(new Room(2, "102", "Double", 150.0, true, new Hotel()));
46
47     when(roomDao.findAll()).thenReturn(rooms);
48
49     List<Room> result = roomService.getAllRooms();
50
51     assertEquals(2, result.size());
52     verify(roomDao).findAll();
53 }
54 @Test
55 public void getRoomByIdTest() {
56     Room room = new Room(1, "101", "Single", 100.0, true, new Hotel());
57     when(roomDao.findById(1)).thenReturn(Optional.of(room));
58
59     Room result = roomService.getRoomById(1);
60
61     assertEquals("101", result.getRoomNo());
62     assertTrue(result.isAvailable());
63 }
64
65 @Test
66 public void getRoomByIdNotFoundTest() {
67     when(roomDao.findById(1)).thenReturn(Optional.empty());
```

32°C Mostly sunny

Search

ENG IN

5:37 PM 5/6/2024



UserServiceTest.java PaymentServiceTest.java HotelServiceTest.java RoomServiceTest.java ×

```
65 @Test
66 public void getRoomByIdNotFoundTest() {
67     when(roomDao.findById(1)).thenReturn(Optional.empty());
68
69     assertThrows(ResourceNotFoundException.class, () -> {
70         roomService.getRoomById(1);
71     });
72 }
73
74 @Test
75 public void createRoomTest() {
76     Hotel hotel = new Hotel();
77     hotel.setHotelId(1);
78
79     RoomDto roomDto = new RoomDto();
80     roomDto.setHotelId(1);
81     roomDto.setRoomNo("101");
82     roomDto.setRoomType("Single");
83     roomDto.setRatePerDay(100.0);
84     roomDto.setAvailable(true);
85
86     when(hotelDao.findById(roomDto.getHotelId())).thenReturn(Optional.of(hotel));
87
88     Room room = new Room();
89     room.setRoomNo(roomDto.getRoomNo());
90     room.setRoomType(roomDto.getRoomType());
91     room.setRatePerDay(roomDto.getRatePerDay());
92     room.setIsavailable(roomDto.isAvailable());
93     room.setHotel(hotel);
94
95     when(roomDao.save(any(Room.class))).thenReturn(room);
96
97     Room result = roomService.createRoom(roomDto);
98
99     assertEquals("101", result.getRoomNo());
100    assertTrue(result.isIsavailable());
101    assertNotNull(result.getHotel());
102 }
103
104
105 @Test
106 public void createRoomHotelNotFoundTest() {
107     RoomDto roomDto = new RoomDto();
```

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

```
103 }
104
105 @Test
106 public void createRoomHotelNotFoundTest() {
107     RoomDto roomDto = new RoomDto();
108     roomDto.setHotelId(1);
109     roomDto.setRoomNo("101");
110     roomDto.setRoomType("Single");
111     roomDto.setRatePerDay(100.0);
112     roomDto.setAvailable(true);
113
114     when(hotelDao.findById(1)).thenReturn(Optional.empty());
115
116     assertThrows(ResourceNotFoundException.class, () -> {
117         roomService.createRoom(roomDto);
118     });
119 }
120
121 @Test
122 public void updateRoomTest() {
123     Room existingRoom = new Room(1, "101", "Single", 100.0, true, new Hotel());
124     when(roomDao.findById(1)).thenReturn(Optional.of(existingRoom));
125
126     Room updatedRoom = new Room(1, "101", "Single", 100.0, false, new Hotel());
127     when(roomDao.save(any(Room.class))).thenReturn(updatedRoom);
128
129     Room result = roomService.updateRoom(updatedRoom);
130
131     assertFalse(result.isIsavailable());
132 }
133
134 @Test
135 public void updateRoomNotFoundTest() {
136     Room room = new Room(1, "101", "Single", 100.0, true, new Hotel());
137     when(roomDao.findById(1)).thenReturn(Optional.empty());
138
139     assertThrows(ResourceNotFoundException.class, () -> {
140         roomService.updateRoom(room);
141     });
142 }
143
144 @Test
145 public void deleteRoomTest() {
146     Room room = new Room(1, "101", "Single", 100.0, true, new Hotel());
147     when(roomDao.findById(1)).thenReturn(Optional.of(room));
148     when(roomDao.delete(room)).thenReturn(1L);
149     roomService.deleteRoom(room);
150 }
```

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

```
116     assertThrows(ResourceNotFoundException.class, () -> {
117         roomService.createRoom(roomDto);
118     });
119 }
120
121 @Test
122 public void updateRoomTest() {
123     Room existingRoom = new Room(1, "101", "Single", 100.0, true, new Hotel());
124     when(roomDao.findById(1)).thenReturn(Optional.of(existingRoom));
125
126     Room updatedRoom = new Room(1, "101", "Single", 100.0, false, new Hotel());
127     when(roomDao.save(any(Room.class))).thenReturn(updatedRoom);
128
129     Room result = roomService.updateRoom(updatedRoom);
130
131     assertFalse(result.isAvailable());
132 }
133
134 @Test
135 public void updateRoomNotFoundTest() {
136     Room room = new Room(1, "101", "Single", 100.0, true, new Hotel());
137     when(roomDao.findById(1)).thenReturn(Optional.empty());
138
139     assertThrows(ResourceNotFoundException.class, () -> {
140         roomService.updateRoom(room);
141     });
142 }
143
144 @Test
145 public void deleteRoomTest() {
146     Room room = new Room(1, "101", "Single", 100.0, true, new Hotel());
147     when(roomDao.findById(1)).thenReturn(Optional.of(room));
148     doNothing().when(roomDao).deleteById(1);
149
150     String result = roomService.deleteRoom(1);
151
152     assertEquals("Room Deleted with id 1", result);
153     verify(roomDao).deleteById(1);
154 }
155
156 }
157 }
158 }
```

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 X

```
27 @SpringBootTest(properties = "eureka.client.enabled=false")
28 public class BookingServiceTest {
29
30     @InjectMocks
31     private BookingServiceImpl bookingService;
32
33     @Mock
34     private BookingDao bookingDao;
35
36     @Mock
37     private UserServiceConsumer userService;
38
39     @Mock
40     private HotelServiceConsumer hotelService;
41
42     @Test
43     public void testAddBooking() {
44
45         User user = new User(1, "Rohit", "1234567890@gmail", "6563133", "user", "84121211", "ghgvnvn");
46         Room room = new Room(1, "101", "Deluxe", 100.0, true, null);
47         Booking booking = new Booking(1, 1, 1, 1, LocalDate.now(), LocalDate.now().plusDays(2),
48             LocalDate.now().plusDays(5), 2, 1, 200.0, "Booked");
49
50         when(userService.getUserById(1)).thenReturn(user);
51         when(hotelService.getRoomById(1)).thenReturn(room);
52
53         // used to return the updated room
54         when(hotelService.updateRoom(any(Room.class))).thenAnswer(invocation -> invocation.getArgument(0));
55         when(bookingDao.save(any(Booking.class))).thenReturn(booking);
56
57         Booking savedBooking = bookingService.addBooking(booking);
58
59         assertEquals("Booked", savedBooking.getStatus());
60         assertFalse(room.isAvailable());
61     }
62
63     @Test
64     void addBookingRoomNotAvailableTest() {
65         Booking booking = new Booking(1, 1, 1, 1, LocalDate.now(), LocalDate.now().plusDays(5),
66             LocalDate.now().plusDays(10), 2, 1, 200.0, "Booked");
67         Room room = new Room(1, "101", "Deluxe", 150.0, false, new Hotel());
68
69         when(hotelService.getRoomById(1)).thenReturn(room);
```

Writable Smart Insert 1:1:0

32°C Mostly sunny Search 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 X

```
65 booking = new Booking(1, 1, 1, 1, LocalDate.now(), LocalDate.now().plusDays(5),
66     LocalDate.now().plusDays(10), 2, 1, 200.0, "Booked");
67 Room room = new Room(1, "101", "Deluxe", 150.0, false, new Hotel());
68
69 when(hotelService.getRoomById(1)).thenReturn(room);
70
71 assertThrows(ResourceNotFoundException.class, () -> bookingService.addBooking(booking));
72 }
73
74 @Test
75 void getBookingDetailsByIdTest() {
76     Booking booking = new Booking(1, 1, 1, 1, LocalDate.now(), LocalDate.now().plusDays(5),
77         LocalDate.now().plusDays(10), 2, 1, 200.0, "Booked");
78     User user = new User(1, "rohit", "rt@gmail.com", "password", "user", "5555555", "gccc");
79     Hotel hotel = new Hotel();
80     hotel.setHotelId(1);
81
82     Room room = new Room(1, "101", "Deluxe", 150.0, true, new Hotel());
83
84     when(bookingDao.findById(1)).thenReturn(Optional.of(booking));
85     when(userService.getUserById(1)).thenReturn(user);
86     when(hotelService.getRoomById(1)).thenReturn(room);
87     when(hotelService.getHotelById(1)).thenReturn(hotel);
88
89     BookingResponse response = bookingService.getBookingDetailsById(1);
90
91     assertEquals("rohit", response.getCustomerName());
92     assertEquals("5555555", response.getMobile());
93     verify(userService).getUserById(1);
94     verify(hotelService).getRoomById(1);
95     verify(hotelService).getHotelById(1);
96 }
97
98 @Test
99 void cancelBookingDeadlinePassedTest() {
100     Booking booking = new Booking(1, 1, 1, 1, LocalDate.now(), LocalDate.now().minusDays(2),
101         LocalDate.now().plusDays(5), 2, 1, 200.0, "Booked");
102
103     when(bookingDao.findById(1)).thenReturn(Optional.of(booking));
104
105     assertThrows(DeadlinePassedException.class, () -> bookingService.cancelBooking(1));
106 }
107 }
```

Writable Smart Insert 57 : 47 : 2105

32°C Mostly sunny Search 5:40 PM 5/6/2024