TEST CASE SUMMARY (RESTAURANT-MANAGEMENT System)(Group 4)

- **Test Case 1:**
  - **Description:** Verify successful creation of a new order with valid data.
  - **Expected Outcome:** Order object is created successfully in the database.
  - **Result:** Passed.

- **Test Case 2:**
  - **Description:** Test order creation with missing required fields.
  - **Expected Outcome:** Order creation fails with appropriate validation errors.
  - **Result:** Passed.

- **Test Case 3:**
  - **Description:** Verify successful retrieval of a specific order by ID.
  - **Expected Outcome:** Order object with matching ID is retrieved correctly.
  - **Result:** Passed.

- **Test Case 4:**
  - **Description:** Test retrieval of an order with an invalid ID.
  - **Expected Outcome:** Order retrieval fails with an appropriate error (e.g., OrderNotFound).
  - **Result:** Passed.

- **Test Case 5:**
  - **Description:** Verify successful addition of a dish to an existing order.
  - **Expected Outcome:** Dish is successfully added to the order.
  - **Result:** Passed.

- **Test Case 6:**
  - **Description:** Test updating an order status (e.g., from "Placed" to "In Preparation").
  - **Expected Outcome:** Order status is updated successfully in the database.
  - **Result:** Passed.

- **Test Case 7:**
  - **Description:** Test deletion of an order.
  - **Expected Outcome:** Order is successfully deleted from the database.
  - **Result:** Passed.

- **Test Case 8:**
  - **Description:** Verify successful creation of a new menu item.
  - **Expected Outcome:** Menu item object is created successfully in the database.

- **Result:** Passed.

- **Test Case 9:**

  - **Description:** Test updating the price of a menu item.
  - **Expected Outcome:** Menu item price is updated successfully in the database.
  - **Result:** Passed.

- **Test Case 10:**

  - **Description:** Test deletion of a menu item.
  - **Expected Outcome:** Menu item is successfully deleted from the database.
  - **Result:** Passed.

How the tests are carried out:

OrderModelTest:

Tests order creation with valid and invalid data.

Tests retrieving and deleting orders.

DishModelTest:

Tests dish creation, price updates, and deletion.

MenuModelTest:

Tests menu creation and deletion.

Assertions:

self.assertEqual(): Checks if two values are equal.

self.assertRaises(): Checks if an exception (like ValidationError or DoesNotExist) is raised.

setUp() Method:

In OrderModelTest, the setUp() method creates a sample Menu and Dish before each test, ensuring consistent test data.

Code Illustration:

```
from django.test import TestCase

from .models import Order, Dish, Menu
```

```
class OrderModelTest(TestCase):
    def setUp(self):
        self.menu = Menu.objects.create(name="Lunch Menu")
        self.dish = Dish.objects.create(name="Pizza", price=10.00, menu=self.menu)

    def test_create_order(self):
        order = Order.objects.create(customer_name="John Doe")
        order.dishes.add(self.dish)
        order.save()
        self.assertEqual(order.customer_name, "John Doe")
        self.assertEqual(order.dishes.count(), 1)

    def test_create_order_with_missing_customer(self):
        with self.assertRaises(ValidationError):
            Order.objects.create()

    def test_get_order_by_id(self):
        order = Order.objects.create(customer_name="Jane Smith")
        retrieved_order = Order.objects.get(pk=order.id)
        self.assertEqual(retrieved_order, order)

    def test_get_order_with_invalid_id(self):
        with self.assertRaises(Order.DoesNotExist):
            Order.objects.get(pk=9999)

class DishModelTest(TestCase):
```

```
    def test_create_dish(self):
        menu = Menu.objects.create(name="Dinner Menu")
        dish = Dish.objects.create(name="Pasta", price=8.00, menu=menu)
        self.assertEqual(dish.name, "Pasta")
        self.assertEqual(dish.price, 8.00)
        self.assertEqual(dish.menu, menu)

    def test_update_dish_price(self):
        dish = Dish.objects.create(name="Salad", price=5.00)
        dish.price = 7.00
        dish.save()
        self.assertEqual(dish.price, 7.00)

    def test_delete_dish(self):
        dish = Dish.objects.create(name="Burger")
        dish_id = dish.id
        dish.delete()
        with self.assertRaises(Dish.DoesNotExist):
            Dish.objects.get(pk=dish_id)

class MenuModelTest(TestCase):
    def test_create_menu(self):
        menu = Menu.objects.create(name="Breakfast Menu")
        self.assertEqual(menu.name, "Breakfast Menu")

    def test_delete_menu(self):
        menu = Menu.objects.create(name="Dessert Menu")
```

```
menu_id = menu.id
menu.delete()
with self.assertRaises(Menu.DoesNotExist):
    Menu.objects.get(pk=menu_id)
```