

Actividad 6.2

Documentación de Ejercicios

A01794935 - Sergio Enrique Pulido Morales

El presente trabajo ha sido realizado cumpliendo las políticas del curso y con los criterios de evaluación de la actividad. Asimismo, establezco que el contenido de este trabajo ha sido documentado en fuentes bibliográficas autorizadas, por tanto, la información redactada no ha sido plagiada de otro documento o trabajo ajeno ni de cualquier otra fuente de carácter confidencial.

Exercise #1: Reservation System

☒ Req 1. Implement a set of classes in Python that implements two abstractions:

1. Hotel
2. Reservation
3. Customers

☒ Req 2. Implement a set of methods to handle the next persistent behaviors (stored in files):

1. Hotels
 - ☒ a. Create Hotel
 - ☒ b. Delete Hotel
 - ☒ c. Display Hotel information
 - ☒ d. Modify Hotel Information
 - ☒ e. Reserve a Room
 - ☒ f. Cancel a Reservation
2. Customer
 - ☒ a. Create Customer
 - ☒ b. Delete a Customer
 - ☒ c. Display Customer Information
 - ☒ d. Modify Customer Information
3. Reservation
 - ☒ a. Create a Reservation (Customer, Hotel)
 - ☒ b. Cancel a Reservation

You are free to decide the attributes within each class that enable the required behavior.

☒ Req 3. Implement unit test cases to exercise the methods in each class. Use the unittest module in Python.

☒ Req 4. The code coverage for all unittests should accumulate at least 85% of line coverage.

☒ Req 5. The program shall include the mechanism to handle invalid data in the file. Errors should be displayed in the console and the execution must continue.

☒ Req 6. Be compliant with PEP8.

☒ Req 7. The source code must show no warnings using Fleak and PyLint.

- Results

Exist 3 files for persist Hotel, Customer and Reservations

```
hotels.txt M x
A01794935_A6.2 > reservation_system > data > hotels.txt
You, 2 minutes ago | 1 author (You)
1 1;Cleveland;Bogota;20;[{ 'id': 1, 'status': 'reserved', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': None}]
2 2;Hilton;Bogota;15;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': None}]
3

customers.txt M x
A01794935_A6.2 > reservation_system > data > customers.txt
You, 2 minutes ago | 1 author (You)
1 1;Sergio;Pulido;Colombiana
2

reservations.txt U x
A01794935_A6.2 > reservation_system > data > reservations.txt
1 1;1;Cleveland;2;1;Sergio;cancelled
2
```

When the start of the application loads this file and build each file in the respective class

```
reservation.py M x
A01794935_A6.2 > reservation_system > reservation.py > Reservation > __init__
You, 1 second ago | 1 author (You)
3 class Reservation:
4     def __init__(self, id, hotel_id, name_hotel, room_id, customer_id, name_customer, status):
5         """Initializes the Reservation class."""
6         self.id = id
7         self.hotel_id = hotel_id
8         self.name_hotel = name_hotel
9         self.room_id = room_id
10        self.customer_id = customer_id
11        self.name_customer = name_customer
12        self.status = status
13

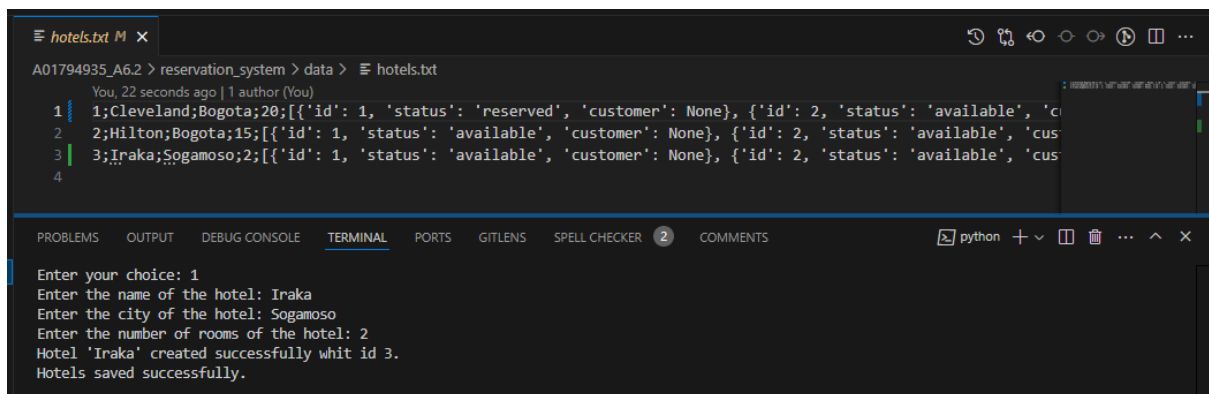
hotel.py x
A01794935_A6.2 > reservation_system > hotel.py > Hotel > __init__
3
You, 19 hours ago | 1 author (You)
4 class Hotel:
5     def __init__(self, id, name, city, num_rooms, rooms=[]):
6         """Initializes the Hotel class."""
7         self.id = id
8         self.name = name
9         self.city = city
10        self.num_rooms = num_rooms
11        self.rooms = rooms

customer.py M x
A01794935_A6.2 > reservation_system > customer.py > Customer > __init__
3 class Customer:
4     def __init__(self, id, name, last_name, nationality):
5         """Initializes the Customer class."""
6         self.id = id
7         self.name = name
8         self.last_name = last_name
9         self.nationality = nationality
```

```
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> python reservation_system/app.py
Hotels loaded successfully.
Customers loaded successfully.
Reservations loaded successfully.
```

```
Reservation System Menu
1. Create Hotel
2. List Hotels
3. Show Hotel
4. Modify Hotel
5. Delete Hotel
6. Create Customer
7. List Customers
8. Show Customer
9. Modify Customer
10. Delete Customer
11. Create Reservation
12. List Reservations
13. Cancel Reservation
0. Exit
Enter your choice: █
```

- Create Hotel

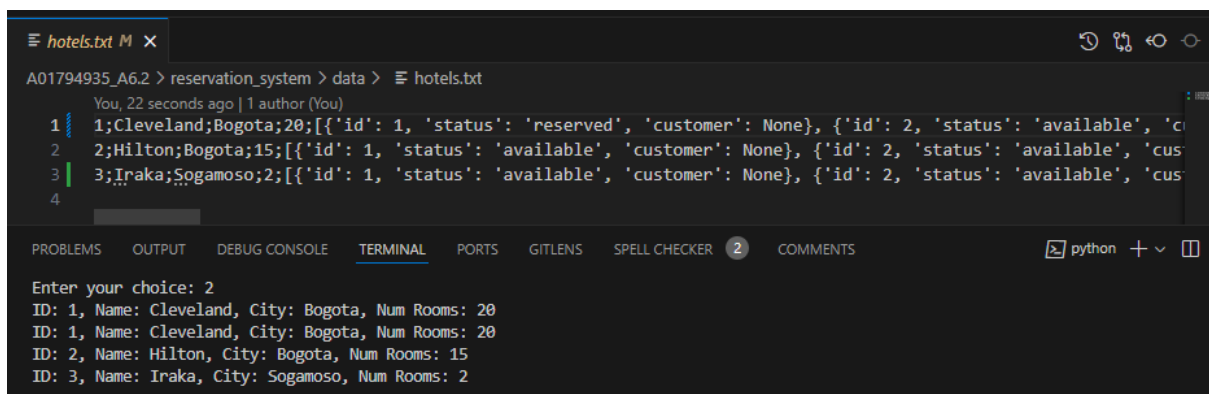


```
hotels.txt M X
A01794935_A6.2 > reservation_system > data > hotels.txt
You, 22 seconds ago | 1 author (You)
1 1;Cleveland;Bogota;20;[{ 'id': 1, 'status': 'reserved', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': None}]
2 2;Hilton;Bogota;15;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': None}]
3 3;Iraka;Sogamoso;2;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': None}]
4

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 2 COMMENTS python + v

Enter your choice: 1
Enter the name of the hotel: Iraka
Enter the city of the hotel: Sogamoso
Enter the number of rooms of the hotel: 2
Hotel 'Iraka' created successfully whit id 3.
Hotels saved successfully.
```

- List Hotels



```
hotels.txt M X
A01794935_A6.2 > reservation_system > data > hotels.txt
You, 22 seconds ago | 1 author (You)
1 1;Cleveland;Bogota;20;[{ 'id': 1, 'status': 'reserved', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': None}]
2 2;Hilton;Bogota;15;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': None}]
3 3;Iraka;Sogamoso;2;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': None}]
4

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 2 COMMENTS python + v

Enter your choice: 2
ID: 1, Name: Cleveland, City: Bogota, Num Rooms: 20
ID: 1, Name: Cleveland, City: Bogota, Num Rooms: 20
ID: 2, Name: Hilton, City: Bogota, Num Rooms: 15
ID: 3, Name: Iraka, City: Sogamoso, Num Rooms: 2
```

- Get Hotel by Id

```
hotels.txt M X
A01794935_A6.2 > reservation_system > data > hotels.txt
You, 22 seconds ago | 1 author (You)
1 1;Cleveland;Bogota;20;[{ 'id': 1, 'status': 'reserved', 'customer': None}, { 'id': 2, 'status': 'available'
2 2;Hilton;Bogota;15;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available',
3 3;Iraka;Sogamoso;2;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available',
4

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 2 COMMENTS python +

Enter your choice: 3
Enter the id of the hotel: 2
ID: 2, Name: Hilton, City: Bogota, Num Rooms: 15
```

- Modify Hotel

```
hotels.txt M X
A01794935_A6.2 > reservation_system > data > hotels.txt
You, 1 second ago | 1 author (You)
1 1;Cleveland;Bogota;20;[{ 'id': 1, 'status': 'reserved', 'customer': None}, { 'id': 2, 'status': 'available',
2 2;Hilton;Bogota;15;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available', 'c
3 3;Iraka;Sogamoso;20;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available',
4

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 2 COMMENTS python +

Enter your choice: 4
Enter the id of the hotel: 3
ID: 3, Name: Iraka, City: Sogamoso, Num Rooms: 2
Enter the new name of the hotel (leave blank to keep current):
Enter the new city of the hotel (leave blank to keep current):
Enter the new number of rooms of the hotel (leave blank to keep current): 20
Hotel 'Iraka' modified successfully.
ID: 3, Name: Iraka, City: Sogamoso, Num Rooms: 20
Hotels saved successfully.
```

- Delete Hotel

```
hotels.txt M X
A01794935_A6.2 > reservation_system > data > hotels.txt
You, 1 second ago | 1 author (You)
1 1;Cleveland;Bogota;20;[{ 'id': 1, 'status': 'reserved', 'customer': None}, { 'id': 2, 'status': 'available'
2 3;Iraka;Sogamoso;20;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available',
3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 2 COMMENTS python +

Enter your choice: 5
Enter the id of the hotel: 2
ID: 2, Name: Hilton, City: Bogota, Num Rooms: 15
Hotel 'Hilton' deleted successfully.
Hotels saved successfully.
```

- Create Customer

```
customers.txt M X
A01794935_A6.2 > reservation_system > data > customers.txt
You, 8 seconds ago | 1 author (You)
1 1;Sergio;Pulido;Colombiana
2 2;Erika;Velandia;Colombiana
3
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL
Enter your choice: 6
Enter the name of the customer: Erika
Enter the last name of the customer: Velandia
Enter the nationality of the customer: Colombiana
Customer 'Erika' created successfully whit id 2.
Customers saved successfully.
```

- List of Customers

```
customers.txt M X
A01794935_A6.2 > reservation_system > data > customers.txt
You, 57 seconds ago | 1 author (You)
1 1;Sergio;Pulido;Colombiana
2 2;Erika;Velandia;Colombiana
3
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 4
Enter your choice: 7
ID: 1, Name: Sergio, Last Name: Pulido, Nationality: Colombiana
ID: 2, Name: Erika, Last Name: Velandia, Nationality: Colombiana
```

- Get Customer by Id

```
customers.txt M X
A01794935_A6.2 > reservation_system > data > customers.txt
You, 2 minutes ago | 1 author (You)
1 1;Sergio;Pulido;Colombiana
2 2;Erika;Velandia;Colombiana
3
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SP
Enter your choice: 8
Enter the id of the customer: 3
Customer with id 3 not found.
```

- Modify Customer

```

customers.txt M X
A01794935_A6.2 > reservation_system > data > customers.txt
You, 1 second ago | 1 author (You)
1 1;Sergio;Pulido;Colombiana You, 2 minutes ago • Uncommitted changes
2 2;Erika;Parra;Italiana
3
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 2 COMMENTS

Enter your choice: 9
Enter the id of the customer: 2
ID: 2, Name: Erika, Last Name: Velandia, Nationality: Colombiana
Enter the new name of the customer (leave blank to keep current):
Enter the new last name of the customer (leave blank to keep current): Parra
Enter the new nationality of the customer (leave blank to keep current): Italiana
Customer 'Erika' modified successfully.
ID: 2, Name: Erika, Last Name: Parra, Nationality: Italiana
Customers saved successfully.

```

- Delete Customer

```

customers.txt M X
A01794935_A6.2 > reservation_system > data > customers.txt
You, 1 second ago | 1 author (You)
1 1;Sergio;Pulido;Colombiana You, 1 minute ago • Uncommitted changes
2
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER CO

Enter your choice: 10
Enter the id of the customer: 2
ID: 2, Name: Erika, Last Name: Parra, Nationality: Italiana
Customer 'Erika' deleted successfully.
Customers saved successfully.

```

- Reserved Hotel

```

reservations.txt U X
A01794935_A6.2 > reservation_system > data > reservations.txt
1 1;1;Cleveland;Bogota;20;1;Sergio;cancelled
2 2;3;Iraka;1;1;Sergio;active

hotels.txt M X
A01794935_A6.2 > reservation_system > data > hotels.txt
You, 1 second ago | 1 author (You)
1 1;Cleveland;Bogota;20;[{ 'id': 1, 'status': 'reserved', 'customer': None }, { 'id': 2, 'status': 'available',
2 3;Iraka;Sogamoso;20;[{ 'id': 1, 'status': 'reserved', 'customer': 1 }, { 'id': 2, 'status': 'available',
3
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 3 COMMENTS python

Enter your choice: 11
Enter the id of the hotel: 3
ID: 3, Name: Iraka, City: Sogamoso, Num Rooms: 20
Enter the id of the customer: 1
ID: 1, Name: Sergio, Last Name: Pulido, Nationality: Colombiana
Room 1 reserved successfully.
Reservation '2' created successfully.
ID: 2, Hotel ID: 3, Name Hotel: Iraka, Room ID: 1, ID Customer: 1, Name Customer: Sergio, Status: active
Hotels saved successfully.

```

- List Reservations

```
reservations.txt U X
A01794935_A6.2 > reservation_system > data > reservations.txt
1 1;1;Cleveland;2;1;Sergio;cancelled
2 2;3;Iraka;1;1;Sergio;active

hotels.txt M X
A01794935_A6.2 > reservation_system > data > hotels.txt
You, 1 second ago | 1 author (You)
1 1;Cleveland;Bogota;20;[{ 'id': 1, 'status': 'reserved', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': 1}]
2 3;Iraka;Sogamoso;20;[{ 'id': 1, 'status': 'reserved', 'customer': 1}, { 'id': 2, 'status': 'available', 'customer': 1}]
3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 3 COMMENTS python

Enter your choice: 12
ID: 1, Hotel ID: 1, Name Hotel: Cleveland, Room ID: 2, ID Customer: 1, Name Customer: Sergio, Status: cancelled
ID: 2, Hotel ID: 3, Name Hotel: Iraka, Room ID: 1, ID Customer: 1, Name Customer: Sergio, Status: active
```

- Cancel Reservation

```
reservations.txt U X
A01794935_A6.2 > reservation_system > data > reservations.txt
1 1;1;Cleveland;2;1;Sergio;cancelled
2 2;3;Iraka;1;1;Sergio;cancelled
3

hotels.txt M X
A01794935_A6.2 > reservation_system > data > hotels.txt
You, 1 second ago | 1 author (You)
1 1;Cleveland;Bogota;20;[{ 'id': 1, 'status': 'reserved', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': 1}]
2 3;Iraka;Sogamoso;20;[{ 'id': 1, 'status': 'available', 'customer': None}, { 'id': 2, 'status': 'available', 'customer': 1}]
3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 3 COMMENTS

0. Exit
Enter your choice: 13
Enter the id of the reservation: 2
ID: 3, Name: Iraka, City: Sogamoso, Num Rooms: 20
Room 1 canceled successfully.
Reservation '2' cancelled successfully.
Hotels saved successfully.
```

- Exit

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 3 COMMENTS

7. List Customers
8. Show Customer
9. Modify Customer
10. Delete Customer
11. Create Reservation
12. List Reservations
13. Cancel Reservation
0. Exit
Enter your choice: 0
Exit the program.
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2>
```


- UnitTest

Implement UnitTest for Hotel and HotelManagement Class

```
test_management_hotel.py U X  management_hotel.py M
tests > test_management_hotel.py > TestHotelManagement > test_cancel_room_no_rooms
1  import unittest
2  from reservation_system.management_hotel import HotelManagement
3  from reservation_system.hotel import Hotel
4
5  class TestHotelManagement(unittest.TestCase):
6
7      def setUp(self):
8          self.hotel_management = HotelManagement()
9
10     def test_load_hotels(self):
11         self.hotel_management.load_hotels()
12         self.assertEqual(len(self.hotel_management.hotels), 0)
13
14     def test_save_hotels(self):
15         self.hotel_management.save_hotels()
16         self.assertEqual(len(self.hotel_management.hotels), 0)
17
18     def test_add_hotel(self):
19         self.hotel_management.add_hotel("Test Hotel", "Test City", 10)
20         self.assertEqual(len(self.hotel_management.hotels), 1)
21         self.assertEqual(self.hotel_management.hotels[0].name, "Test Hotel")
22         self.assertEqual(self.hotel_management.hotels[0].city, "Test City")
23         self.assertEqual(self.hotel_management.hotels[0].num_rooms, 10)
24         self.assertEqual(self.hotel_management.hotels[0].id, 1)
25
26     def test_add_multiple_hotels(self):
27         self.hotel_management.add_hotel("Hotel One", "City One", 5)
28         self.hotel_management.add_hotel("Hotel Two", "City Two", 15)
29         self.assertEqual(len(self.hotel_management.hotels), 2)
```

Also implement negative cases

```
test_management_hotel.py U X
tests > test_management_hotel.py > TestHotelManagement > test_load_hotels_no_file
5 class TestHotelManagement(unittest.TestCase):
93     self.assertIsNone(room['customer'])
94
95     def test_load_hotels_no_file(self):
96         self.hotel_management.load_hotels()
97         self.assertEqual(len(self.hotel_management.hotels), 0)
98
99     def test_save_hotels_no_hotels(self):
100         self.hotel_management.save_hotels()
101         self.assertEqual(len(self.hotel_management.hotels), 0)
102
103     def test_reserve_room_no_hotel(self):
104         room = self.hotel_management.reserve_room(None, "Test Customer")
105         self.assertIsNone(room)
106
107     def test_reserve_room_no_rooms(self):
108         self.hotel_management.add_hotel("Hotel One", "City One", 0)
109         hotel = self.hotel_management.get_hotel_by_id('1')
110         room = self.hotel_management.reserve_room(hotel, "Test Customer")
111         self.assertIsNone(room)
112
113     def test_reserve_room_some_reserved(self):
114         self.hotel_management.add_hotel("Hotel One", "City One", 5)
115         hotel = self.hotel_management.get_hotel_by_id('1')
116         hotel.rooms[0]['status'] = 'reserved'
117         room = self.hotel_management.reserve_room(hotel, "Test Customer")
118         self.assertEqual(room['status'], 'reserved')
119         self.assertEqual(room['customer'], 'Test Customer')
120
121     def test_cancel_room_no_hotel(self):
122         room = self.hotel_management.cancel_room(None, None)
123         self.assertIsNone(room)
124
125     def test_cancel_room_no_rooms(self):
```

Implement UnitTest for Customer and CustomerManagement Class

```

test_management_customer.py U X
tests > test_management_customer.py > TestCustomerManagement > test_add_multiple_customers
A62\test_management_customer.py • Untracked
2 from reservation_system.management_customer import CustomerManagement
3
4 class TestCustomerManagement(unittest.TestCase):
5
6     def setUp(self):
7         self.customer_management = CustomerManagement()
8
9     def test_load_customers(self):
10         self.customer_management.load_customers()
11         self.assertEqual(len(self.customer_management.customers), 0)
12
13     def test_save_customers(self):
14         self.customer_management.save_customers()
15         self.assertEqual(len(self.customer_management.customers), 0)
16
17     def test_add_customer(self):
18         self.customer_management.add_customer("Test Customer", "Test LastName", "Test Nation")
19         self.assertEqual(len(self.customer_management.customers), 1)
20         self.assertEqual(self.customer_management.customers[0].name, "Test Customer")
21         self.assertEqual(self.customer_management.customers[0].last_name, "Test LastName")
22         self.assertEqual(self.customer_management.customers[0].nationality, "Test Nation")
23         self.assertEqual(self.customer_management.customers[0].id, 1)
24
25     def test_add_multiple_customers(self):
26         self.customer_management.add_customer("Customer One", "LastName One", "Test Nation")
27         self.customer_management.add_customer("Customer Two", "LastName Two", "Test Nation")
28         self.assertEqual(len(self.customer_management.customers), 2)
29         self.assertEqual(self.customer_management.customers[0].name, "Customer One")
30         self.assertEqual(self.customer_management.customers[1].name, "Customer Two")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS GITLENS SPELL CHECKER COMMENTS

Also implement negative cases

```

test_management_customer.py U X
tests > test_management_customer.py > TestCustomerManagement > test_delete_customer_by_invalid_id
4 class TestCustomerManagement(unittest.TestCase):
5     def test_delete_customer_by_id(self):
6         self.assertEqual(self.customer_management.customers[0].name, "Customer Two")
7
8     def test_load_customers_no_file(self):
9         self.customer_management.load_customers()
10        self.assertEqual(len(self.customer_management.customers), 0)
11
12    def test_save_customers_no_customers(self):
13        self.customer_management.save_customers()
14        self.assertEqual(len(self.customer_management.customers), 0)
15
16    def test_get_customer_by_id_invalid(self):
17        customer = self.customer_management.get_customer_by_id('a')
18        self.assertIsNone(customer)
19
20    def test_delete_customer_by_invalid_id(self):
21        self.customer_management.add_customer("Customer One", "City One", 5)
22        customer = self.customer_management.get_customer_by_id('a')
23        self.assertIsNone(customer)
24
25

```

Implement UnitTest for Reservation and ReservationManagement Class

```
test_management_reservation.py U X
tests > test_management_reservation.py > TestReservationManagement > test_get_reservation_by_id_invalid
1 import unittest
2 from reservation_system.management_reservation import ReservationManagement
3
4 class TestReservationManagement(unittest.TestCase):
5
6     def setUp(self):
7         self.reservation_management = ReservationManagement()
8
9     def test_load_reservations(self):
10         self.reservation_management.load_reservations()
11         self.assertEqual(len(self.reservation_management.reservations), 2)
12
13     def test_save_reservations(self):
14         self.reservation_management.load_reservations()
15         self.reservation_management.save_reservations()
16         self.assertEqual(len(self.reservation_management.reservations), 2)
17
18     def test_create_reservation(self):
19         self.reservation_management.create_reservation(1, "Test Hotel", 1, 1, "Test Customer")
20         self.assertEqual(len(self.reservation_management.reservations), 1)
21         self.assertEqual(self.reservation_management.reservations[0].hotel_id, 1)
22         self.assertEqual(self.reservation_management.reservations[0].name_hotel, "Test Hotel")
23         self.assertEqual(self.reservation_management.reservations[0].room_id, 1)
24         self.assertEqual(self.reservation_management.reservations[0].customer_id, 1)
25         self.assertEqual(self.reservation_management.reservations[0].name_customer, "Test Customer")
26         self.assertEqual(self.reservation_management.reservations[0].status, "active")
27         self.assertEqual(self.reservation_management.reservations[0].id, 1)
28
29     def test_list_reservations(self):
30         self.reservation_management.create_reservation(1, "Test Hotel", 1, 1, "Customer One")
```

Also implement negative cases

```
test_management_reservation.py U X
tests > test_management_reservation.py > TestReservationManagement > test_get_reservation_by_id_invalid
4 class TestReservationManagement(unittest.TestCase):
51     def test_cancel_reservation(self):
58         self.assertEqual(reservation.name_customer, "Customer One")
59         self.assertEqual(reservation.status, "cancelled")
60
61     def test_get_reservation_by_id_invalid(self):
62         reservation = self.reservation_management.get_reservation_by_id('a')
63         self.assertIsNone(reservation)
64
65     def test_cancel_reservation_invalid_reservation(self):
66         self.reservation_management.create_reservation(1, "Test Hotel", 1, 1, "Customer One")
67         reservation = self.reservation_management.get_reservation_by_id('a')
68         self.assertIsNone(reservation)
69
70 if __name__ == '__main__':
71     unittest.main()
```

Before that implement any cases for principal app.py

```
test_app.py M x
tests > test_app.py > TestReservationSystem > test_create_reservation
0 sys.path.insert(0, os.path.abspath(os.path.
7 from reservation_system.app import run_rese
8
9 You, 1 minute ago | 1 author (You)
10 class TestReservationSystem(unittest.TestCase):
11
12     @patch('builtins.input', side_effect=['0'])
13     @patch('sys.stdout', new_callable=StringIO)
14     @patch('management_hotel.HotelManagement')
15     @patch('management_customer.CustomerManagement')
16     @patch('management_reservation.ReservationManagement')
17     def test_main(self, MockHotelManagement, MockCustomerManagement, MockReservationManagement, mock
18         main()
19
20
21     @patch('builtins.input', side_effect=['1', 'NameTest', 'CityTest', 10, '0'])
22     @patch('sys.stdout', new_callable=StringIO)
23     def test_create_hotel(self, mock_stdout, mock_input):
24         mock_management_hotel = MagicMock()
25         mock_management_customer = MagicMock()
26         mock_management_reservation = MagicMock()
27
28         run_reservation_system(mock_management_hotel, mock_management_customer, mock_management_res
29
30         self.assertIn("Reservation System Menu", mock_stdout.getvalue())
31         mock_management_hotel.add_hotel.assert_called_once()
32
33
34     @patch('builtins.input', side_effect=['2', '0'])
35     @patch('sys.stdout', new_callable=StringIO)
```

All test implements are successful

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS GITLENS SPELL CHECKER ... powershell + v [ ] [ ] ... ^
ID: 2, Hotel ID: 1, Name Hotel: Test Hotel, Room ID: 2, ID Customer: 2, Name Customer: Customer Two, Status: active
ID: 1, Hotel ID: 1, Name Hotel: Test Hotel, Room ID: 1, ID Customer: 1, Name Customer: Customer One, Status: active
ID: 2, Hotel ID: 1, Name Hotel: Test Hotel, Room ID: 2, ID Customer: 2, Name Customer: Customer Two, Status: active
.Reservations loaded successfully.
.Reservations loaded successfully.
.
-----
Ran 50 tests in 0.097s
OK
PS C:\Users\CHECH\OneDrive\Documents\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> coverage run -m unittest discover
```

And report the coverage is high


```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  TEST RESULTS  PORTS  GITLENS  SPELL CHECKER  ...  powershell +
OK
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> coverage report
Name                               Stmts  Miss  Cover
-----
reservation_system\__init__.py      0      0   100%
reservation_system\app.py          137      7    95%
reservation_system\customer.py       8      0   100%
reservation_system\hotel.py         13      0   100%
reservation_system\management_customer.py  57      4    93%
reservation_system\management_hotel.py  82      4    95%
reservation_system\management_reservation.py  43      3    93%
reservation_system\reservation.py    11      0   100%
tests\__init__.py                   0      0   100%
tests\test_app.py                  183      1    99%
tests\test_management_customer.py     77      1    99%
tests\test_management_hotel.py       113      1    99%
tests\test_management_reservation.py    60      1    98%
-----
TOTAL                               784     22    97%
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2>

```

- Pylint

When execute pylint appear many issues:

```

reservation_system\management_hotel.py:45:30: W0622: Redefining built-in 'id' (redefined-builtin)
***** Module A01794935_A6.2.reservation_system.management_reservation
reservation_system\management_reservation.py:10:0: C0303: Trailing whitespace (trailing-whitespace)
reservation_system\management_reservation.py:16:0: C0301: Line too long (115/100) (line-too-long)
reservation_system\management_reservation.py:17:0: C0301: Line too long (135/100) (line-too-long)
reservation_system\management_reservation.py:22:0: C0303: Trailing whitespace (trailing-whitespace)
reservation_system\management_reservation.py:27:0: C0301: Line too long (194/100) (line-too-long)
reservation_system\management_reservation.py:32:0: C0301: Line too long (110/100) (line-too-long)
reservation_system\management_reservation.py:36:0: C0303: Trailing whitespace (trailing-whitespace)
reservation_system\management_reservation.py:48:0: C0303: Trailing whitespace (trailing-whitespace)
reservation_system\management_reservation.py:55:0: C0303: Trailing whitespace (trailing-whitespace)
reservation_system\management_reservation.py:4:0: E0401: Unable to import 'reservation' (import-error)
reservation_system\management_reservation.py:7:0: C0115: Missing class docstring (missing-class-docstring)
reservation_system\management_reservation.py:16:20: W0622: Redefining built-in 'id' (redefined-builtin)
reservation_system\management_reservation.py:14:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
reservation_system\management_reservation.py:25:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
reservation_system\management_reservation.py:29:4: R0913: Too many arguments (6/5) (too-many-arguments)
reservation_system\management_reservation.py:29:4: R0917: Too many positional arguments (6/5) (too-many-positional-arguments)
reservation_system\management_reservation.py:31:8: W0622: Redefining built-in 'id' (redefined-builtin)
reservation_system\management_reservation.py:37:36: W0622: Redefining built-in 'id' (redefined-builtin)
reservation_system\management_reservation.py:49:4: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
***** Module A01794935_A6.2.reservation_system.reservation
reservation_system\reservation.py:13:0: C0303: Trailing whitespace (trailing-whitespace)
reservation_system\reservation.py:15:0: C0304: Final newline missing (missing-final-newline)
reservation_system\reservation.py:15:0: C0301: Line too long (207/100) (line-too-long)
reservation_system\reservation.py:3:0: C0115: Missing class docstring (missing-class-docstring)
reservation_system\reservation.py:4:4: R0913: Too many arguments (8/5) (too-many-arguments)
reservation_system\reservation.py:4:4: R0917: Too many positional arguments (8/5) (too-many-positional-arguments)
reservation_system\reservation.py:4:23: W0622: Redefining built-in 'id' (redefined-builtin)
reservation_system\reservation.py:3:0: R0903: Too few public methods (1/2) (too-few-public-methods)

-----
Your code has been rated at 6.72/10

PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> pylint reservation_system/

```

Finally, after many changes pylint isn't issues:

```
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> pylint reservation_system/
-----
Your code has been rated at 10.00/10 (previous run: 9.97/10, +0.03)
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2>
```

As change part of the code, rerun the tests and adjust problems that appear, and finally run all test successfully and the coverage is ok

```
Reservation '2' created successfully.
ID: 2, Hotel ID: 1, Name Hotel: Test Hotel, Room ID: 2, ID Customer: 2, Name Customer: Customer Two, Status: active
ID: 1, Hotel ID: 1, Name Hotel: Test Hotel, Room ID: 1, ID Customer: 1, Name Customer: Customer One, Status: active
ID: 2, Hotel ID: 1, Name Hotel: Test Hotel, Room ID: 2, ID Customer: 2, Name Customer: Customer Two, Status: active
.Reservations loaded successfully.
.Reservations loaded successfully.
.
-----
Ran 51 tests in 0.243s

OK
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> coverage run -m unittest discover
```

```
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> coverage report
Name                               Stmts  Miss  Cover
-----
reservation_system\__init__.py      0      0  100%
reservation_system\app.py           122     7    94%
reservation_system\customer.py       9      0  100%
reservation_system\hotel.py          14      0  100%
reservation_system\management_customer.py  57     4    93%
reservation_system\management_hotel.py  83     4    95%
reservation_system\management_reservation.py  44     3    93%
reservation_system\reservation.py    12      0  100%
tests\__init__.py                   0      0  100%
tests\test_app.py                   200    42    79%
tests\test_management_customer.py     77     1    99%
tests\test_management_hotel.py       113     1    99%
tests\test_management_reservation.py   62     1    98%
-----
TOTAL                               793    63    92%
```

And execute the program

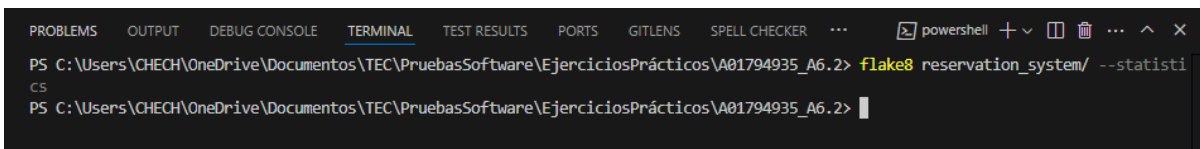
```
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> python -m reservation_system.app
Hotels loaded successfully.
Customers loaded successfully.
Reservations loaded successfully.

Reservation System Menu
1. Create Hotel
2. List Hotels
3. Show Hotel
4. Modify Hotel
5. Delete Hotel
6. Create Customer
7. List Customers
8. Show Customer
9. Modify Customer
10. Delete Customer
11. Create Reservation
12. List Reservations
13. Cancel Reservation
0. Exit
Enter your choice: 
```

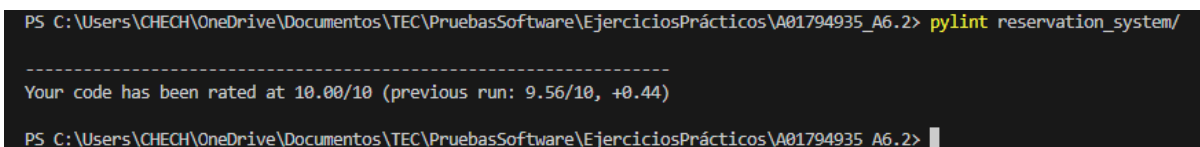
- Flake8

```
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> flake8 reservation_system/ --statistics
CS
reservation_system/app.py:43:80: E501 line too long (81 > 79 characters)
reservation_system/app.py:92:80: E501 line too long (97 > 79 characters)
reservation_system/app.py:108:80: E501 line too long (86 > 79 characters)
reservation_system/app.py:145:80: E501 line too long (90 > 79 characters)
reservation_system/app.py:150:80: E501 line too long (92 > 79 characters)
reservation_system/app.py:172:80: E501 line too long (89 > 79 characters)
reservation_system/app.py:190:80: E501 line too long (82 > 79 characters)
reservation_system/app.py:209:80: E501 line too long (89 > 79 characters)
reservation_system/customer.py:16:5: E303 too many blank lines (2)
reservation_system/hotel.py:5:1: E302 expected 2 blank lines, found 1
reservation_system/hotel.py:15:5: E303 too many blank lines (2)
reservation_system/hotel.py:19:80: E501 line too long (86 > 79 characters)
reservation_system/hotel.py:22:5: E303 too many blank lines (2)
reservation_system/management_customer.py:13:5: E303 too many blank lines (2)
reservation_system/management_customer.py:16:80: E501 line too long (94 > 79 characters)
reservation_system/management_customer.py:18:80: E501 line too long (87 > 79 characters)
reservation_system/management_customer.py:19:80: E501 line too long (86 > 79 characters)
reservation_system/management_customer.py:26:5: E303 too many blank lines (2)
reservation_system/management_customer.py:28:80: E501 line too long (90 > 79 characters)
reservation_system/management_customer.py:37:5: E303 too many blank lines (2)
reservation_system/management_customer.py:42:80: E501 line too long (92 > 79 characters)
reservation_system/management_customer.py:45:5: E303 too many blank lines (2)
reservation_system/management_customer.py:52:5: E303 too many blank lines (2)
reservation_system/management_customer.py:66:5: E303 too many blank lines (2)
reservation_system/management_customer.py:83:5: E303 too many blank lines (2)
reservation_system/management_hotel.py:14:5: E303 too many blank lines (2)
reservation_system/management_hotel.py:17:80: E501 line too long (91 > 79 characters)
reservation_system/management_hotel.py:19:80: E501 line too long (84 > 79 characters)
reservation_system/management_hotel.py:21:80: E501 line too long (82 > 79 characters)
reservation_system/management_hotel.py:28:5: E303 too many blank lines (2)
reservation_system/management_hotel.py:30:80: E501 line too long (87 > 79 characters)
reservation_system/management_hotel.py:40:5: E303 too many blank lines (2)
reservation_system/management_hotel.py:46:80: E501 line too long (83 > 79 characters)
reservation_system/management_hotel.py:49:5: E303 too many blank lines (2)
reservation_system/management_hotel.py:56:5: E303 too many blank lines (2)
reservation_system/management_hotel.py:70:5: E303 too many blank lines (2)
reservation_system/management_hotel.py:70:80: E501 line too long (86 > 79 characters)
```

Check and fix the Flake8 issues, and corroborate that function is executing correctly.



Pylint preserves the successful report



The program is executed successfully


```
CS
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> python -m reservation_system.app
Hotels loaded successfully.
Customers loaded successfully.
Reservations loaded successfully.

Reservation System Menu
1. Create Hotel
2. List Hotels
3. Show Hotel
4. Modify Hotel
5. Delete Hotel
6. Create Customer
7. List Customers
8. Show Customer
9. Modify Customer
10. Delete Customer
11. Create Reservation
12. List Reservations
13. Cancel Reservation
0. Exit
Enter your choice: 2
ID: 1, Name: Cleveland, City: Bogota, Num Rooms: 20
ID: 3, Name: Iraka, City: Sogamoso, Num Rooms: 20
```

And Tests report all ok

```
-----
Ran 51 tests in 0.257s
```

OK

```
PS C:\Users\CHECH\OneDrive\Documentos\TEC\PruebasSoftware\EjerciciosPrácticos\A01794935_A6.2> coverage run -m unittest discover
```