Uguralp Onbasli - 7798416

Tri Pham – 8060220

Upadhyay Hinali – 8119158

Jasvir Kaur – 8133530

Professor Igor Pustylnick

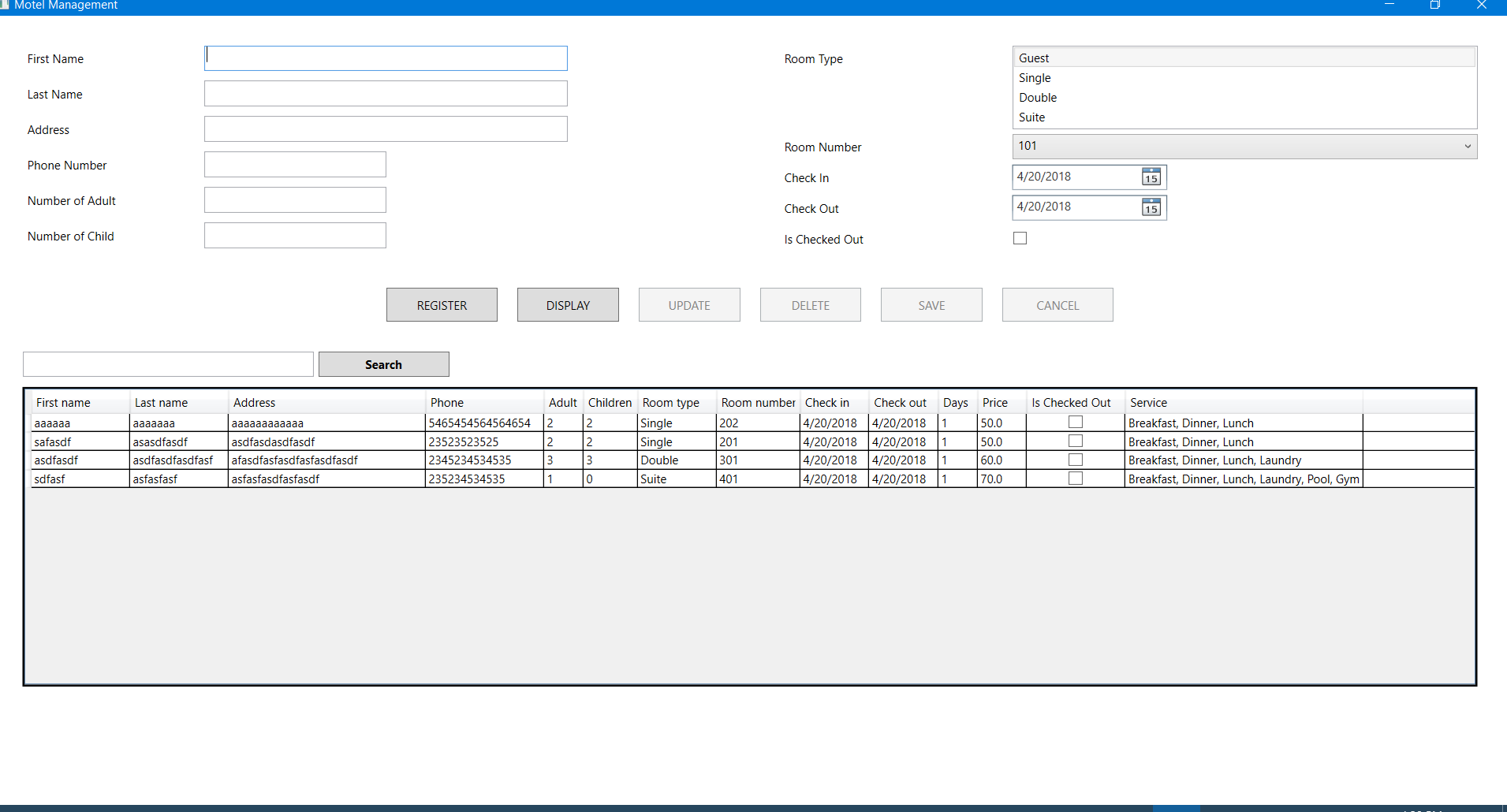
PROG8145 - Software Development Techniques

April 20, 2018

**The Motel Management**

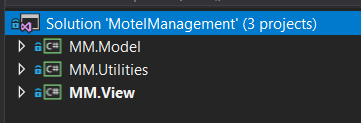
**Overview**

The purpose of the Motel Management System is to allow booking rooms for the customers and list them all. The system allows the user to register, update and delete the all information of the bookings. Operations were designed well for the Motel Management System.

****

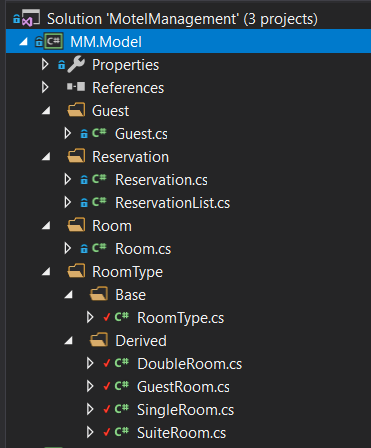
**The structure of the project**

The solution includes 3 separated projects. One is called Model, containing definitions (classes) of Guest, Reservation, Room and RoomType. One is called Utilities which is used for error and converter handling, read and write XML… And the final one is called View, which is the main UI of the project.



**The MM.Model project**

This is the place containing all the classes of the project



**The Guest class:**

It contains all the information of a guest, such as First name, last name, address and phone number.

**The Room class:**

It contains properties of Room number and IsCheckedOut, to represent the current status of the room.

* When IsCheckedOut = true: the room is booked / reserved
* When IsCheckedOut = false: the room is vacancy.

**The RoomType class:**

In this class, there are RoomTypeName, list of Room, Price and Service of each type of Room.

This is the abstract class which is inherited by other classes: DoubleRoom, GuestRoom, SingleRoom and SuiteRoom.

**The DoubleRoom, GuestRoom, SingleRoom and SuiteRoom classes:**

Each class is derived from the base class RoomType. Therefore they all have RoomTypeName, list of Room, Price and Service. Each type of room has its different name, price and service.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Class** | **Name** | **Price** | **Service** | **ExtraService** | **Note** |
| RoomType | N/A | N/A | Breakfast, Dinner | N/A | Base class |
| GuestRoom | Guest | 35 | Same as RoomType | N/A | Derived from RoomType |
| SingleRoom | Single | 50 | Same as RoomType | Lunch | Derived from RoomType |
| DoubleRoom | Double | 60 | Same as RoomType | Lunch, Laundry | Derived from RoomType |
| SuiteRoom | Suite | 70 | Same as RoomType | Lunch, Laundry, Pool, Gym | Derived from RoomType |

**Each type of room has its own rooms:**

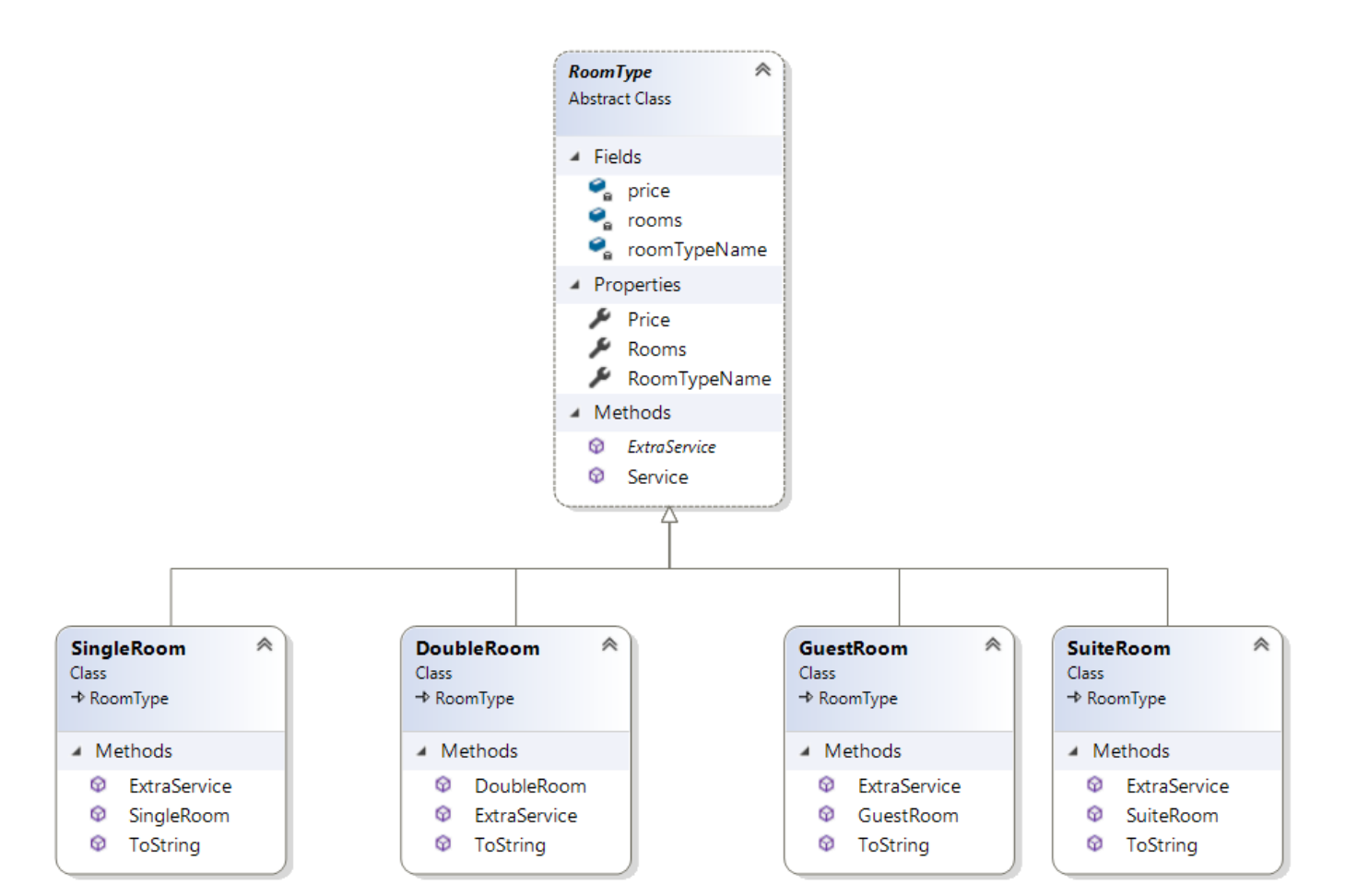
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Room Type** | **Guest** | **Single** | **Double** | **Suite** |
| **Room Number** | 101 | 201 | 301 | 401 |
| 102 | 202 | 302 | 402 |
| 103 | 203 | 303 | 403 |
| 104 | 204 | 304 | 404 |

**The Reservation class**

This class contains the main activity of the project. It includes:

* ReservationID: created by GUID, which is unique for each reservation.
* Guest: information of the guest that is booking a room
* Number of Adult: Minimum is 1 and maximum is 10 for each room
* Number of Child: Minimum is 0 and maximum is 10 for each room.
* Check In, Check out (datetime): Check Out always equal or after Check In
* RoomType: type of a reserved room
* NumberOfDays: is the number of days between CheckOut and CheckIn
* TotalPrice = [Price of each type of room] \* [NumberOfDays]
* Service: service of each type of room.

**Class Diagram:**

****

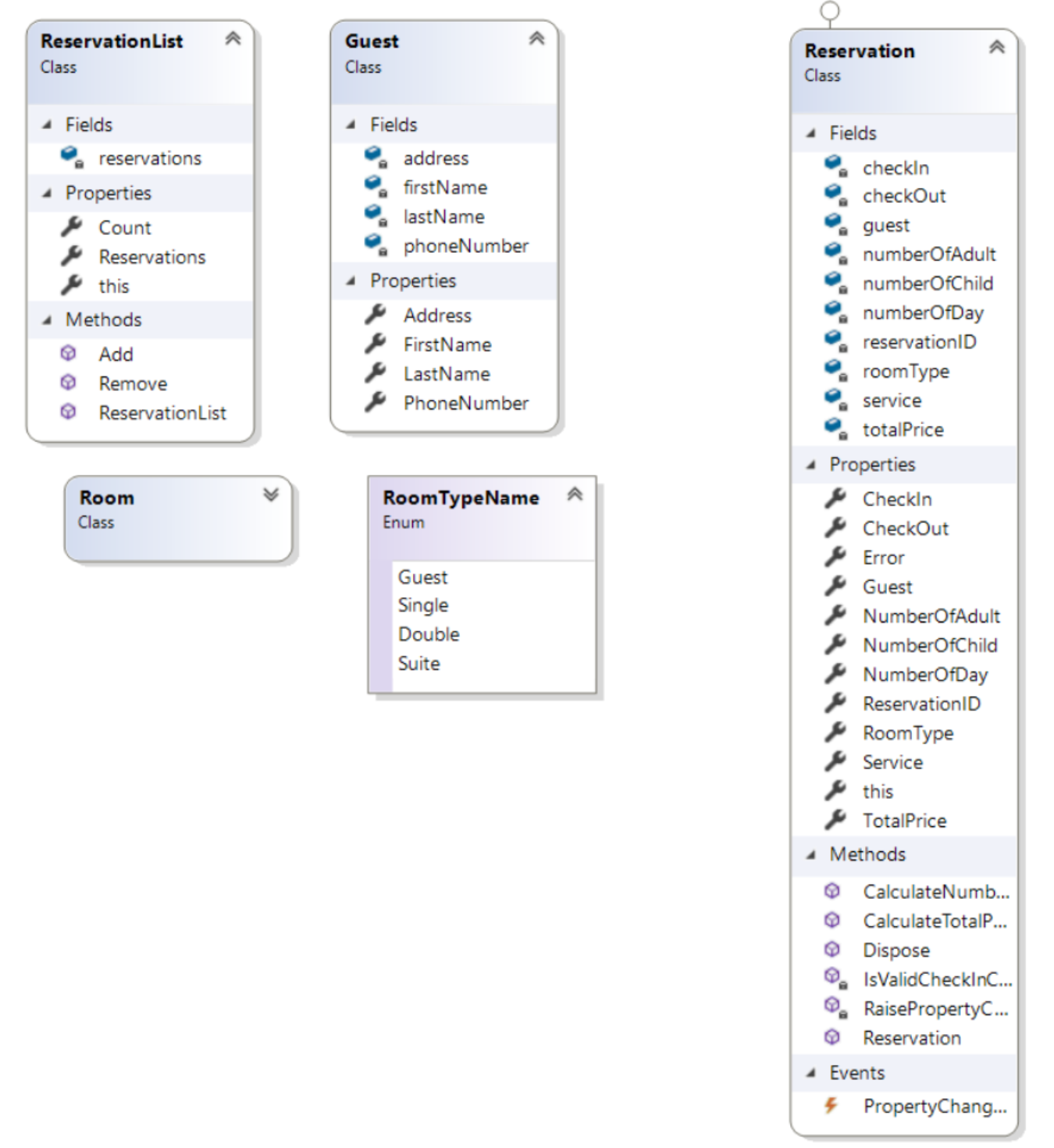
****

Figure 1: Class Diagram for the Motel Management

**Main Page:**

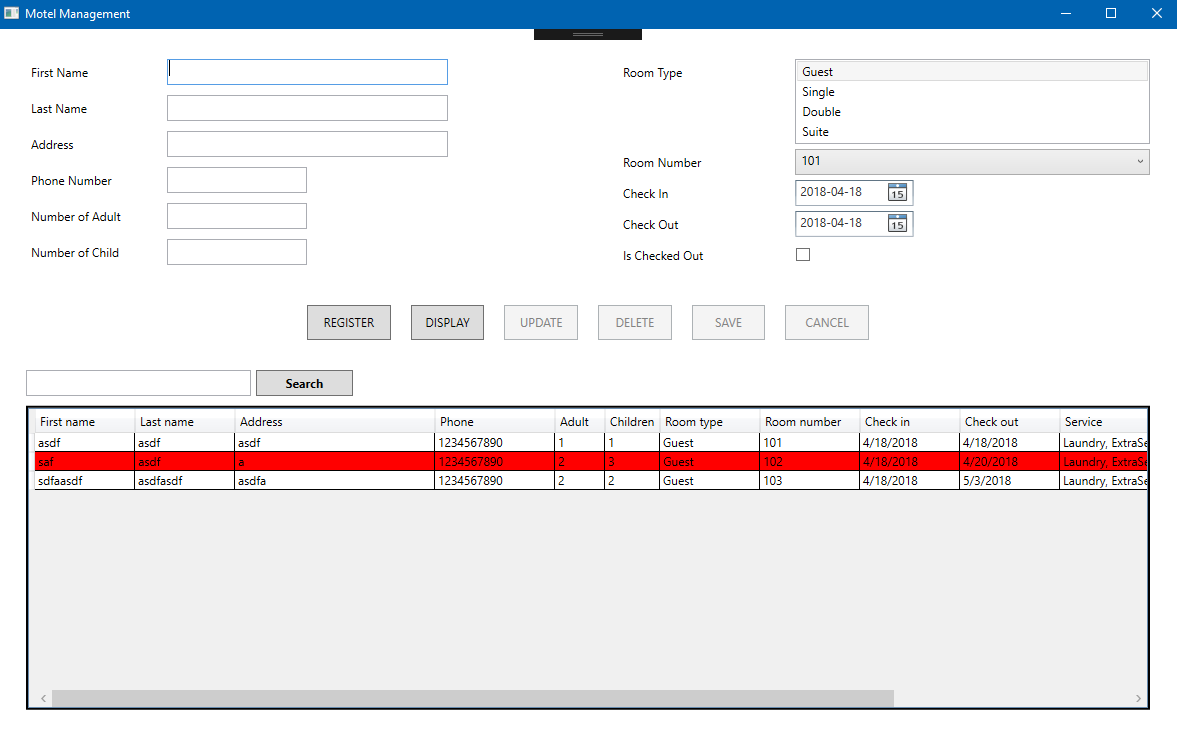


Figure 2: Main Page of the Application

This is the main page of the application. It has two sections. The section on the top is for adding new customers. The other section on the bottom shows the information list for the customers in the datagrid.

**Adding Data:**

After giving the right inputs, the user clicks the “REGISTER” button to save the data. If the data is wrong, the user can’t save the data and get an error. The data saves in an XML file properly by using the serialization.

**Selecting Data:**

Firstly, the user needs to select the existing data from the Datagrid. After selecting one of the rows, information appears on the top section of the app. By default, the top area is disabled.

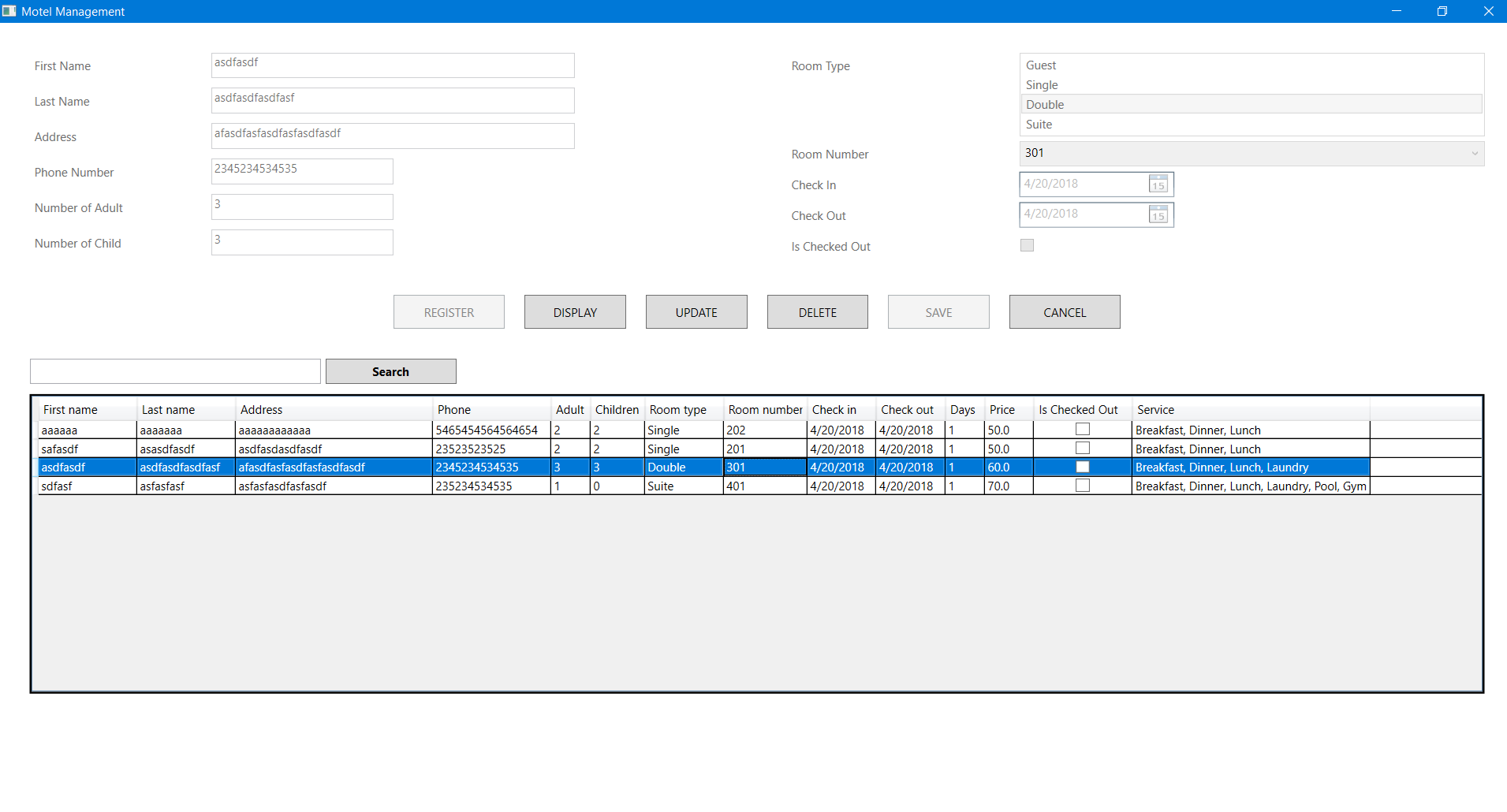


Figure 3: Select the row to edit, update and delete.

**Updating Data:**

After selecting the data, the user clicks the UPDATE button to enable the top area. Then they can edit and click SAVE button to update the data. The data saves in an XML file properly by using the serialization.

**Deleting Data:**

After selecting the data, the user clicks the DELETE button to delete the selected reservation data.

**Searching Data:**

LINQ was implemented for the search feature. The user can search for the following values: First Name, Last Name, Address, Phone Number, Room Type, Room Number, check in date and check out date.

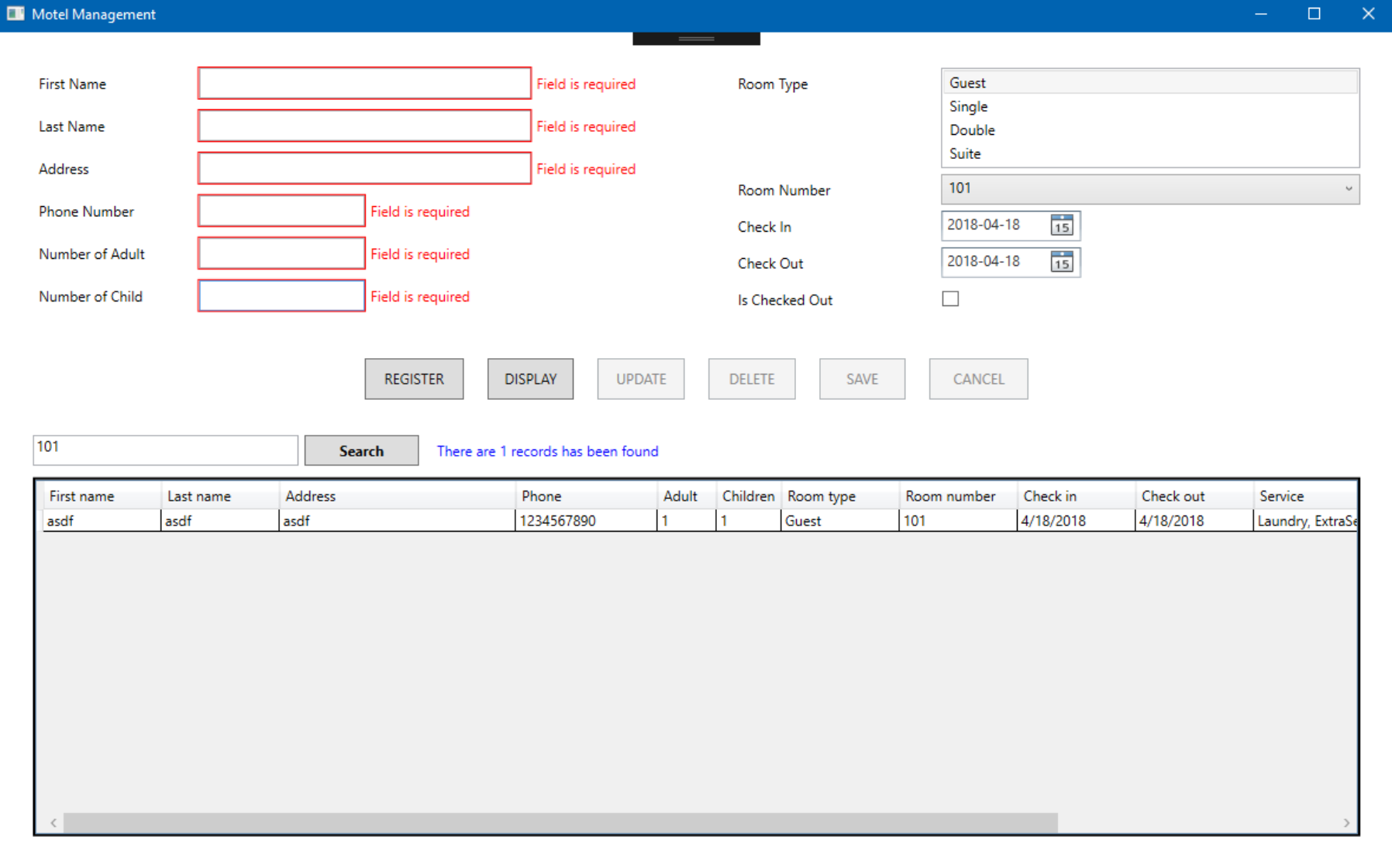


Figure 4: Example for the search.

**Error Handlings:**

Error handlings were implemented based on the rubric. Converters and validation rules work properly in the input fields. Additionally, the user can’t book the room if it has already booked.

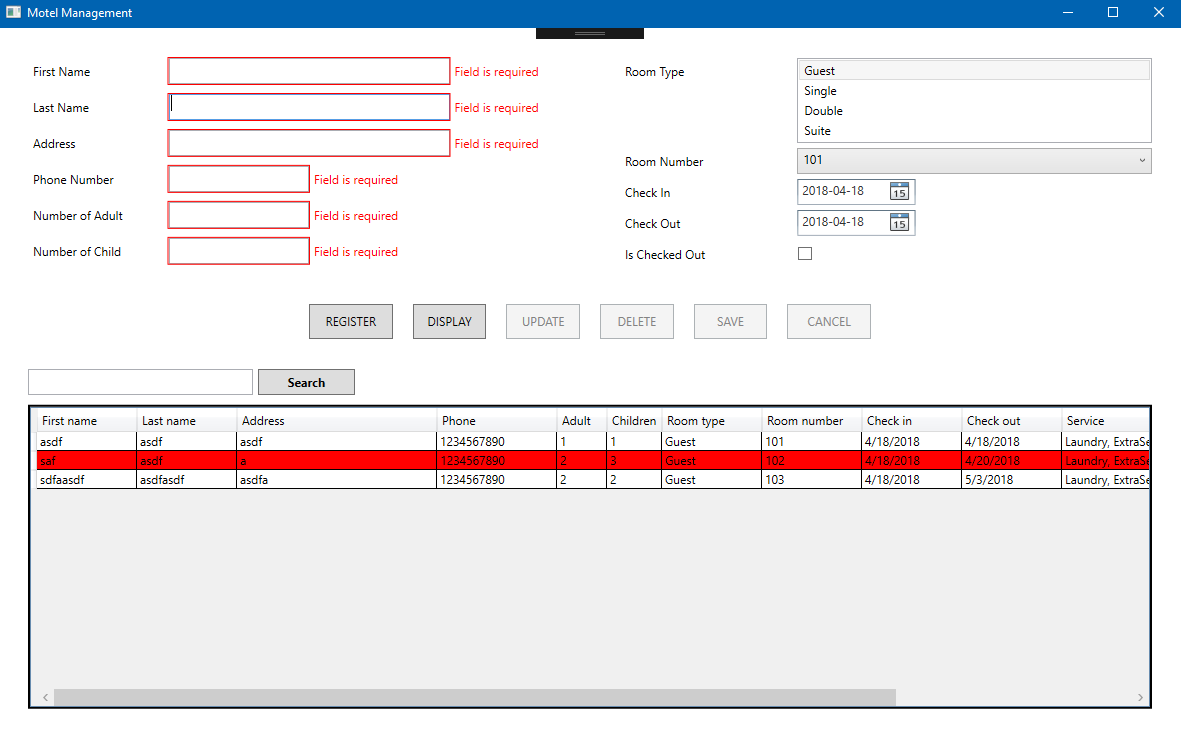


Figure 5: Application gets error when there is no value in the fields.

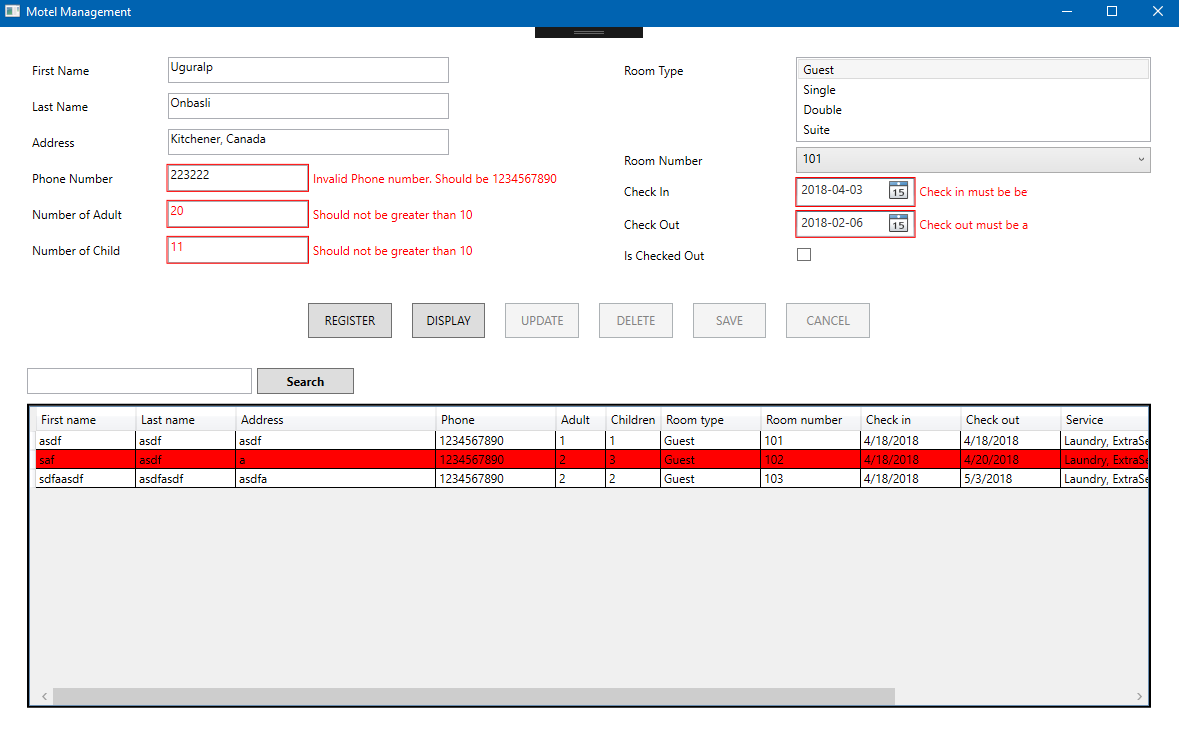


Figure 6: Application gets error when the fields are filled wrong.

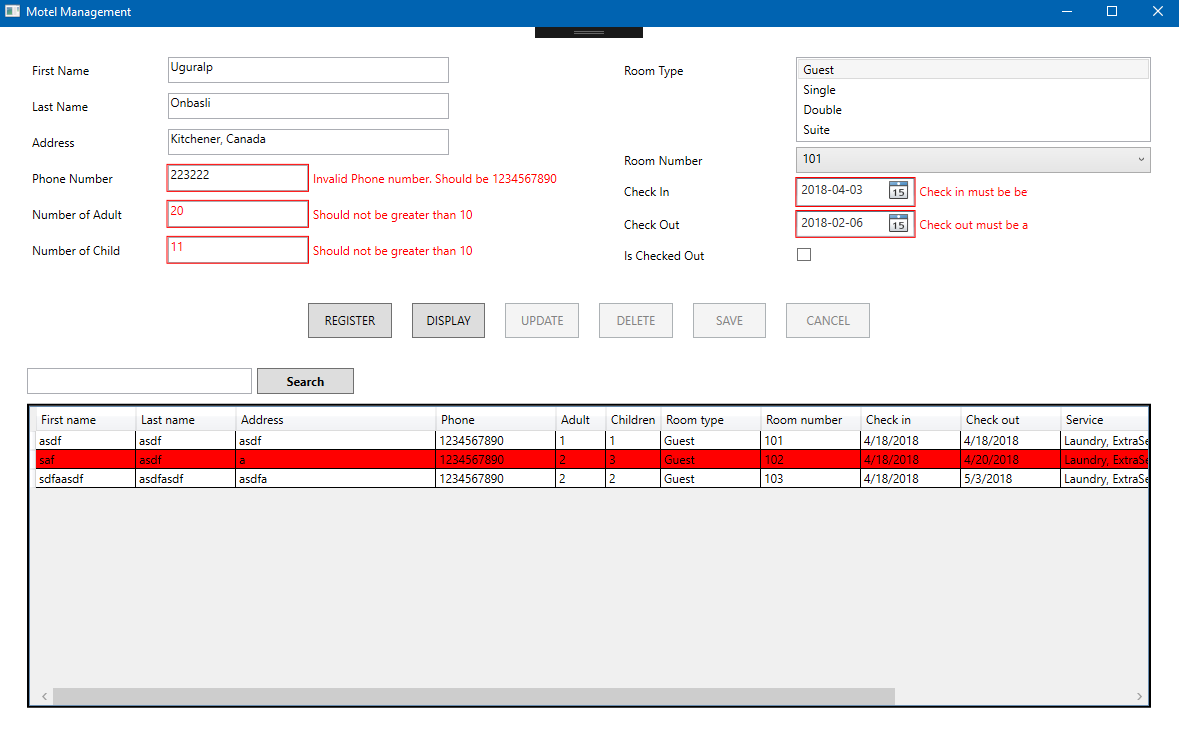
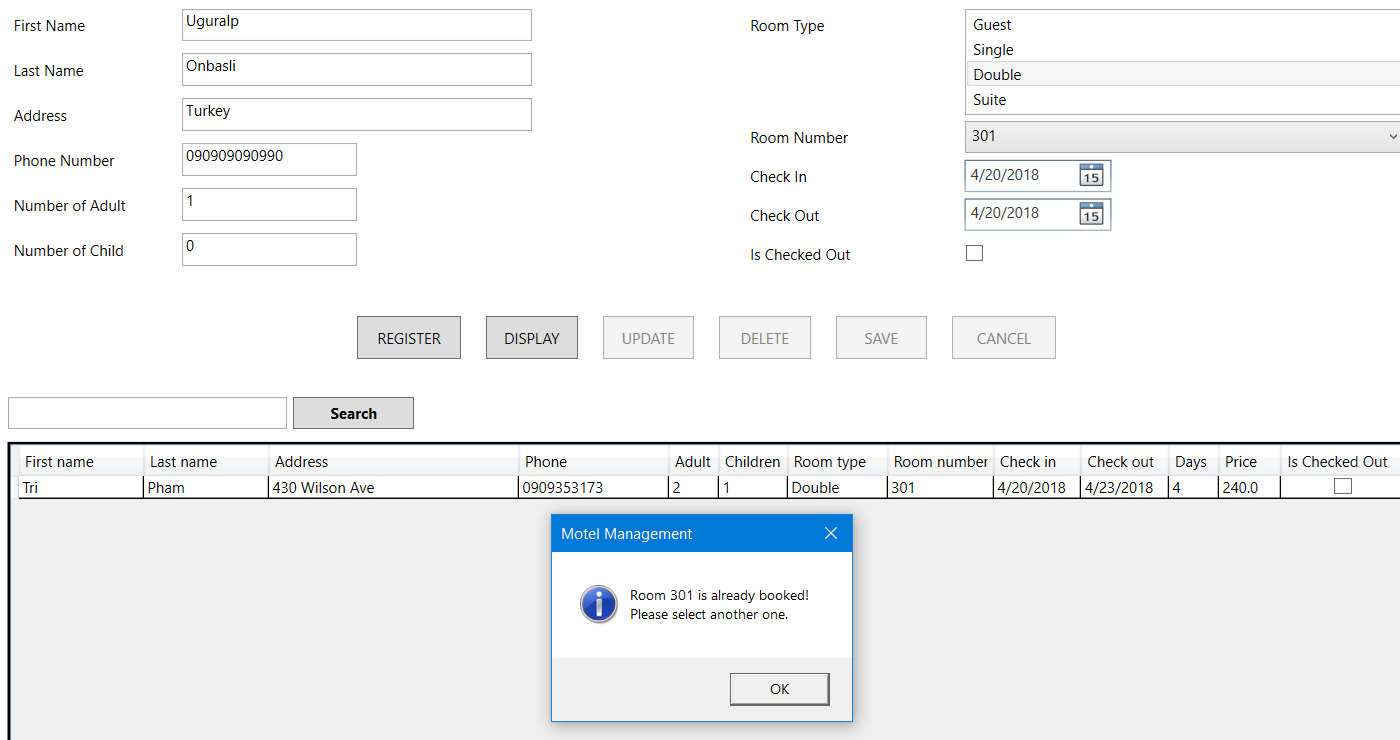


Figure : IsCheckedOut feature on the datagrid.

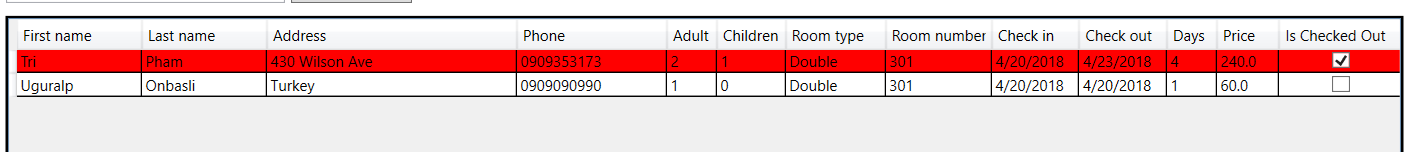
Each record (reservation) has its IsCheckedOut Boolean value. If the record’s IsCheckedOut value on the row is true, the background of the row of the record changes to the red color.

**Checking if the selected room is booked or not**

There is a warning message when register a new reservation with a room that is already booked:



In case there are 2 reservations having the same room. One is checked out and one is not.



There is a warning message when user tries to update the reservation that is checked out (Red) back to not checked out:

