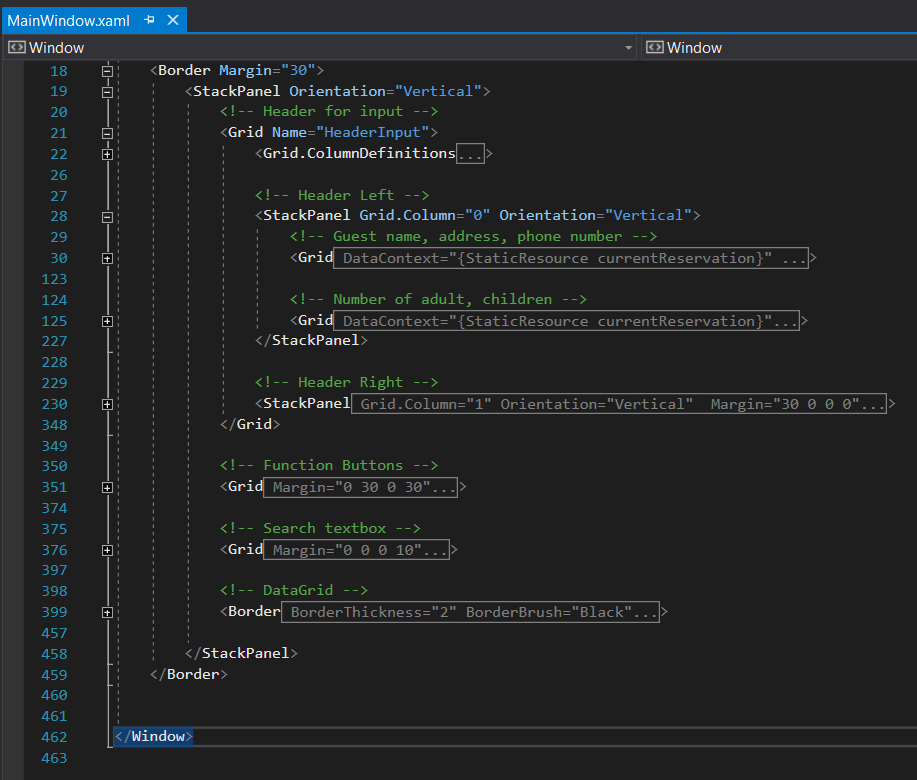
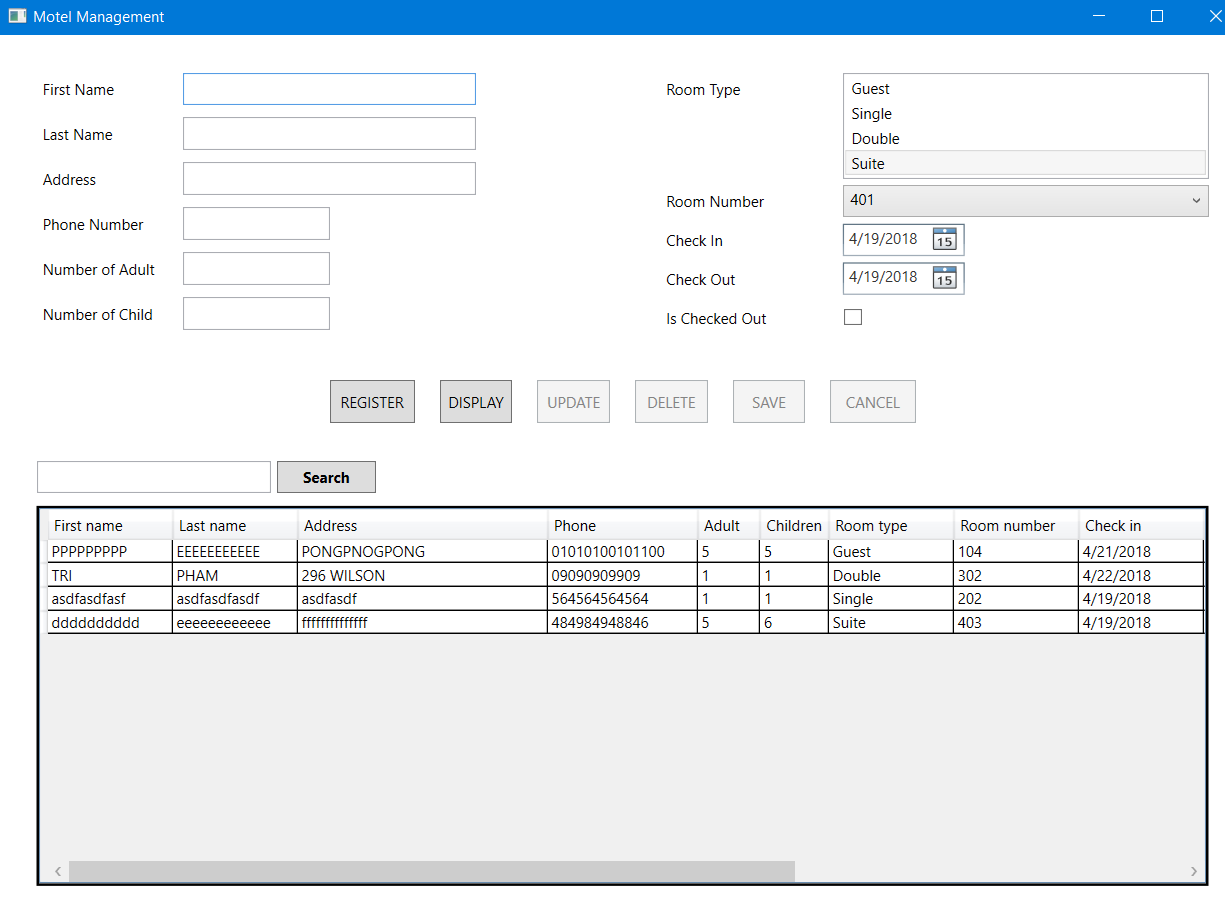
**GUI-Screen Design**

The screen is properly made up. All fields are aligned both vertically and horizontally. All required elements are present

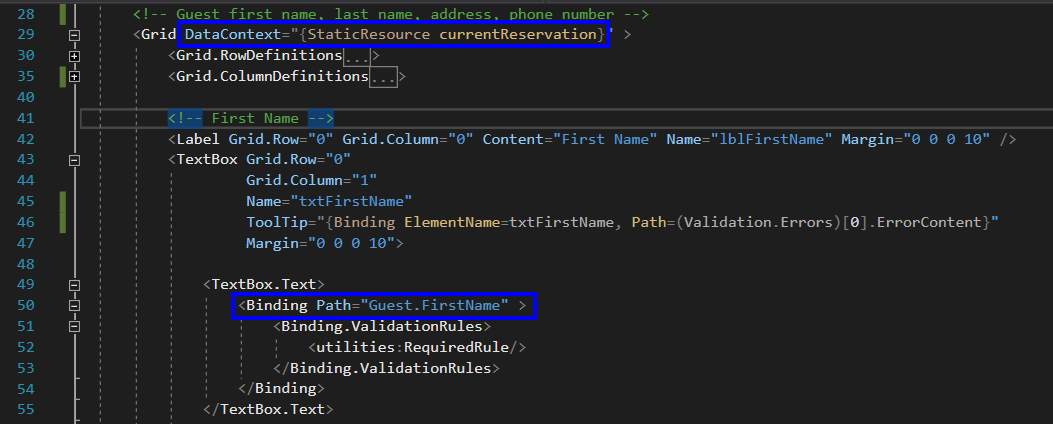




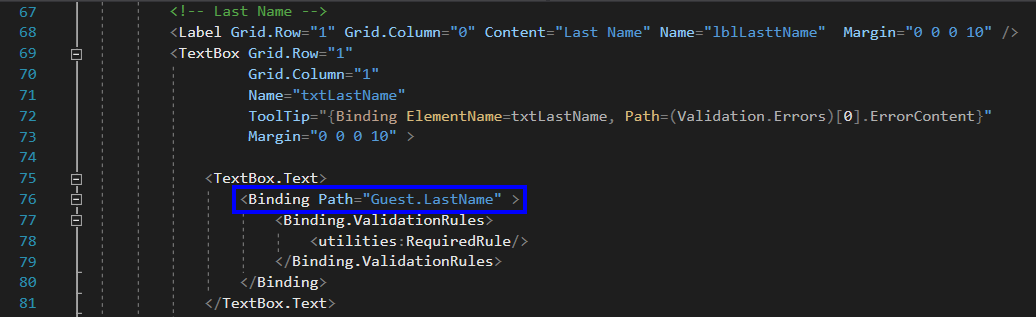
**GUI-Binding**

All controls are bound to the proper data. There are no unbound controls carrying the information

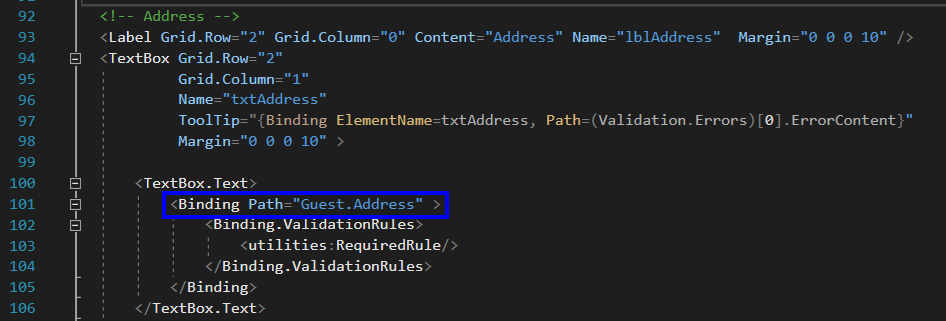
First name



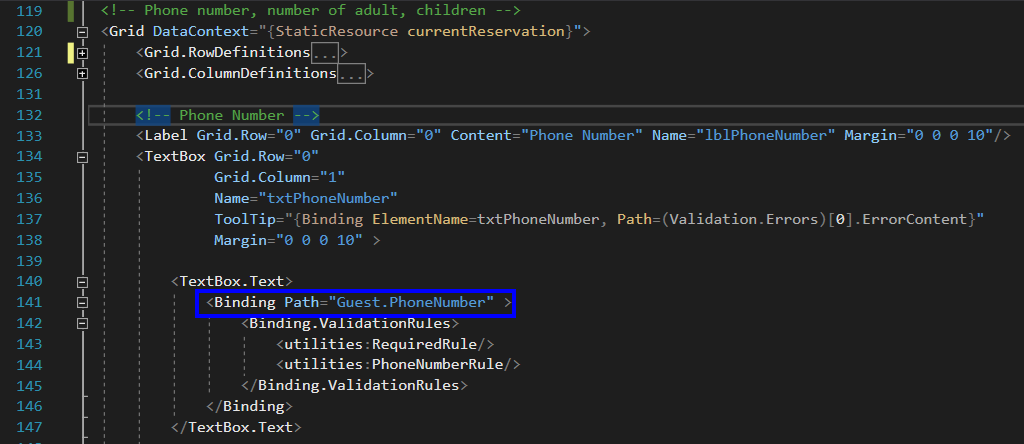
Last name



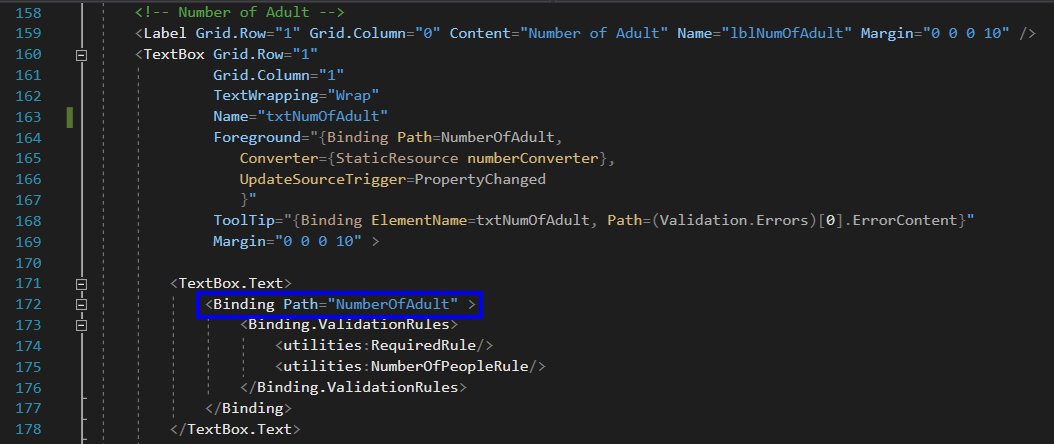
Address



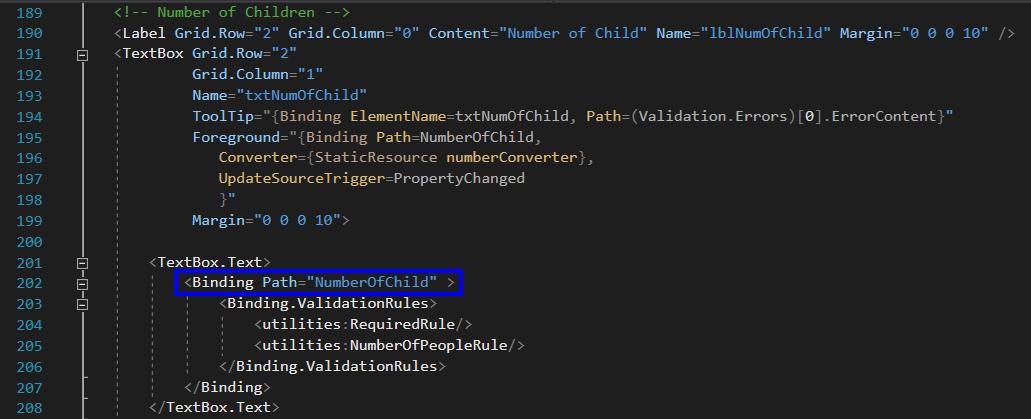
Phone Number



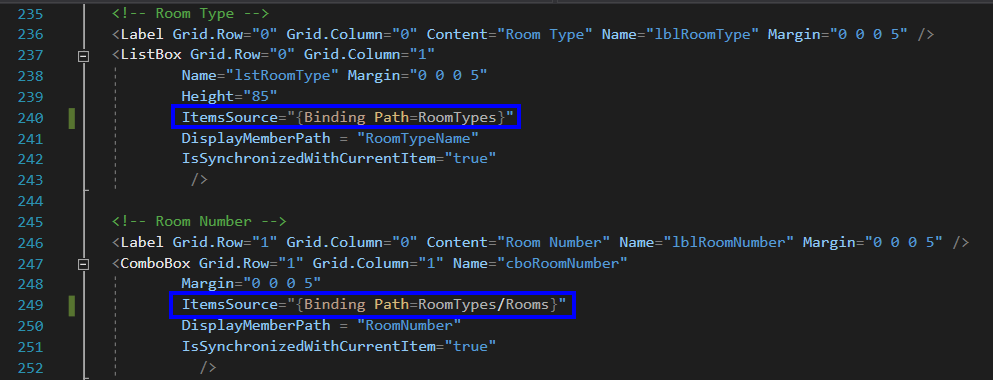
Number of Adult



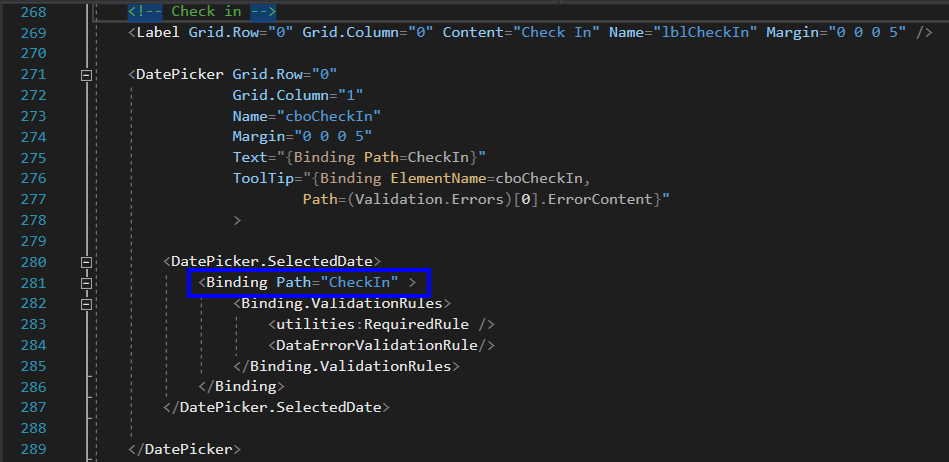
Number of Child



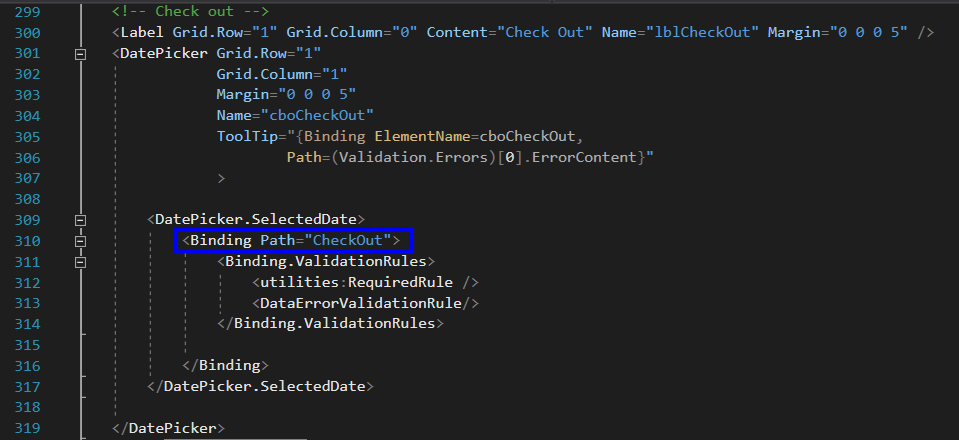
Room type and Room Number



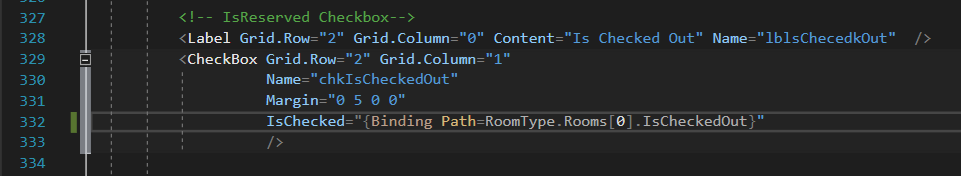
Check In



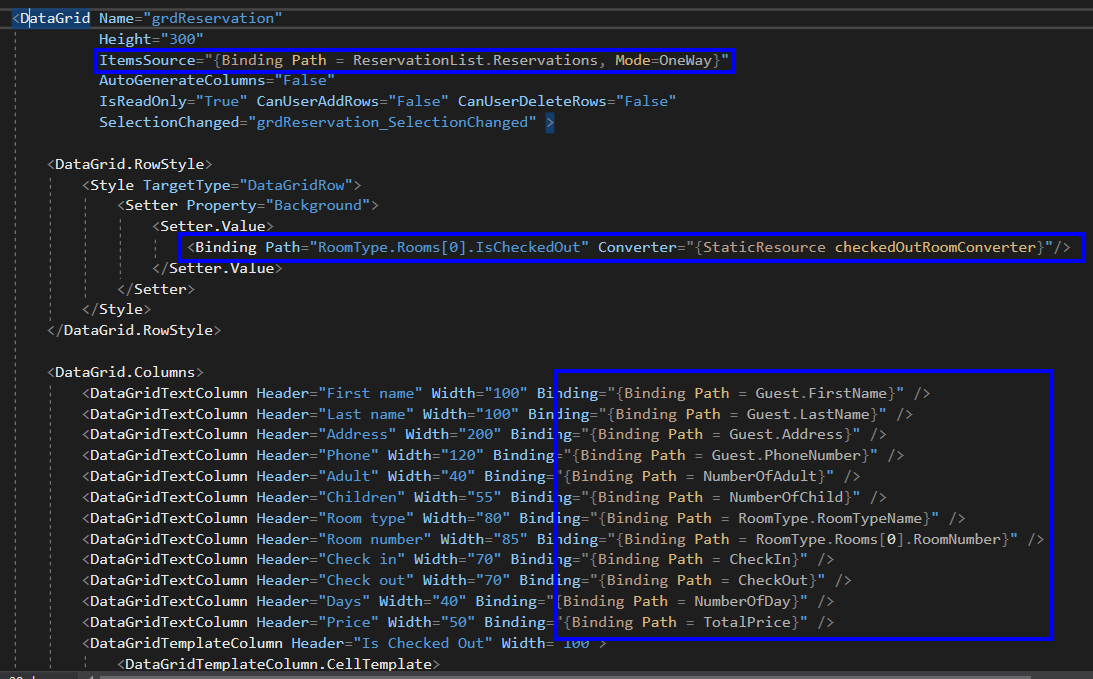
Check Out



Is Checked Out

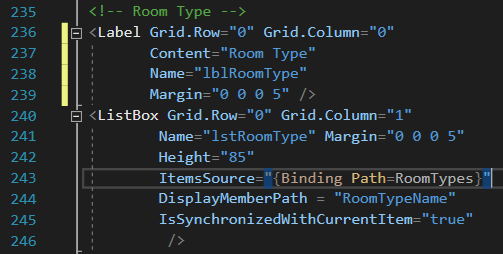


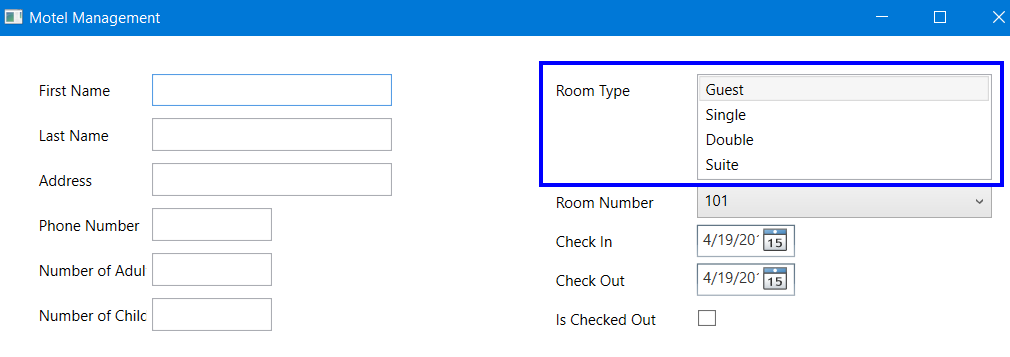
DataGrid



**GUI-Lists**

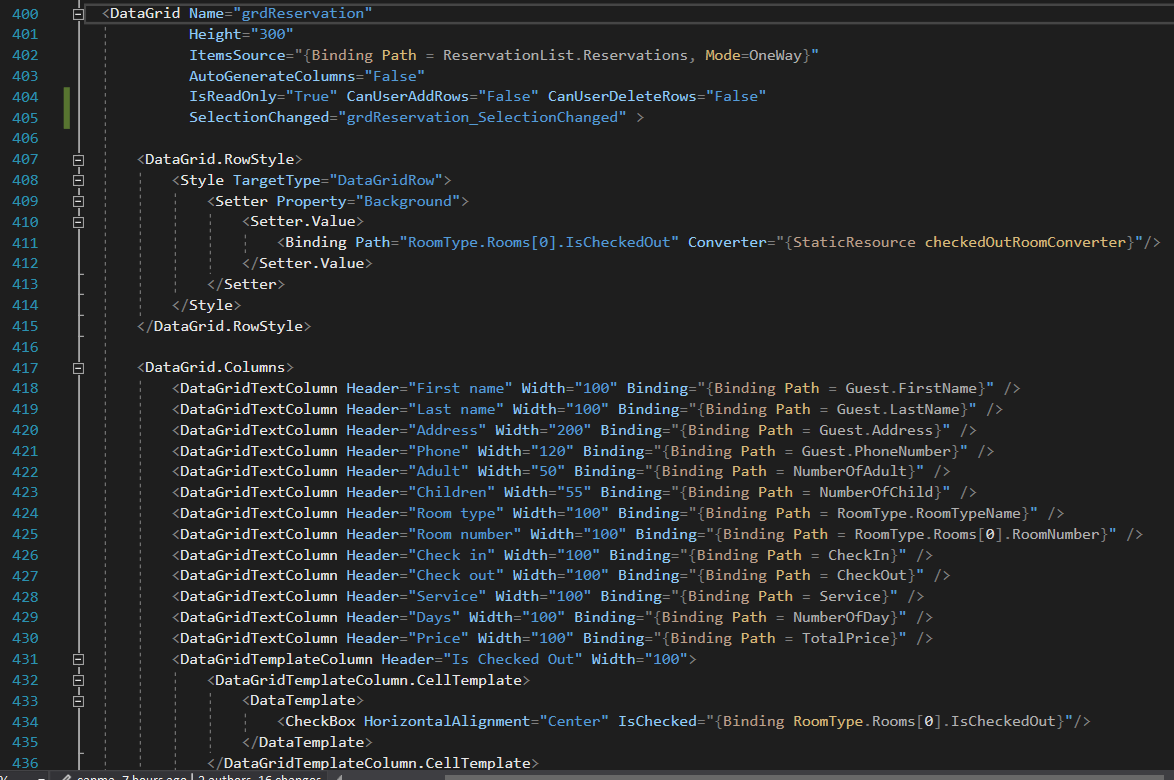
Lists are properly used and properly bound

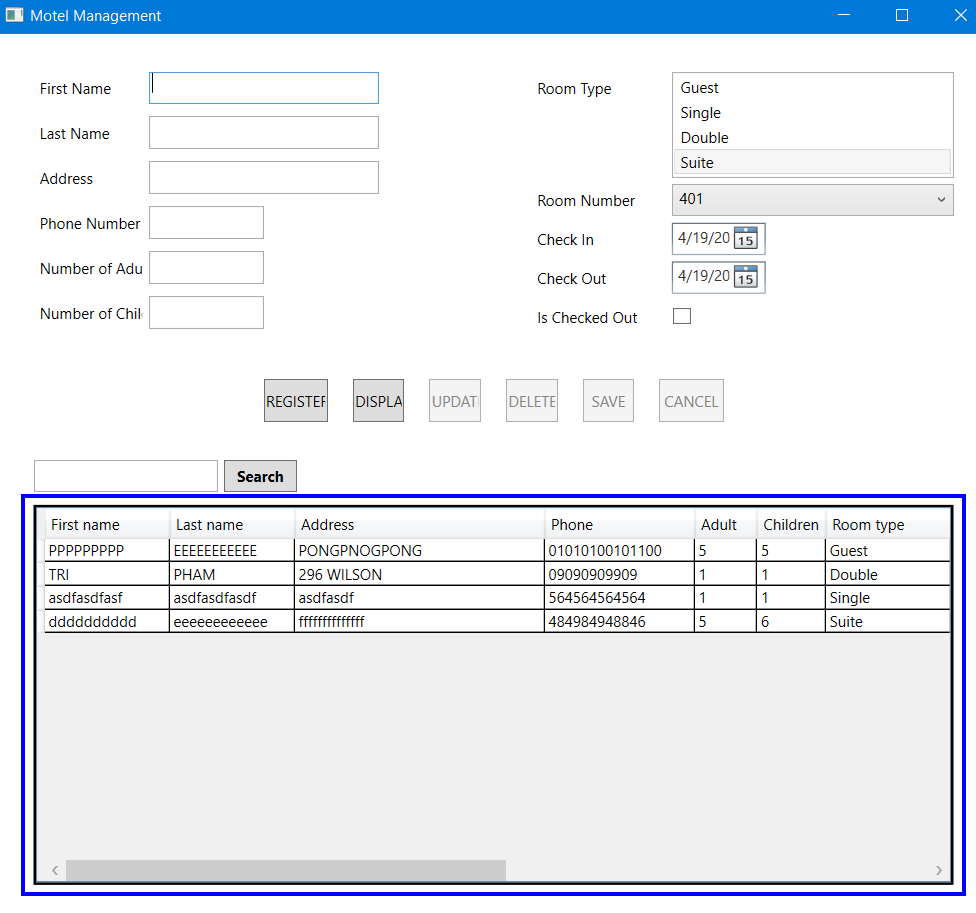




**GUI-Grids**

Data Grid or Grid View is properly used. The data is bound properly and is updated by WPF



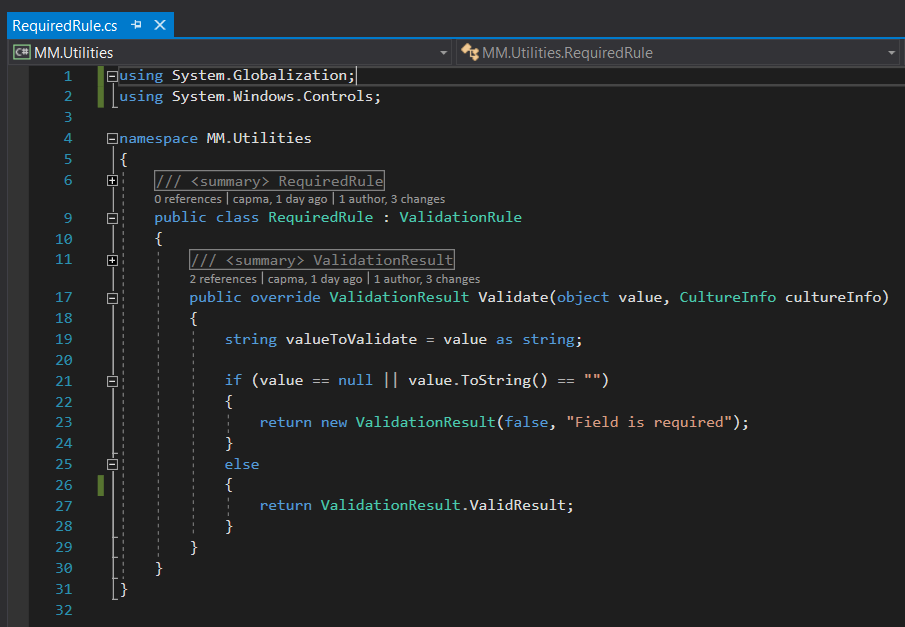


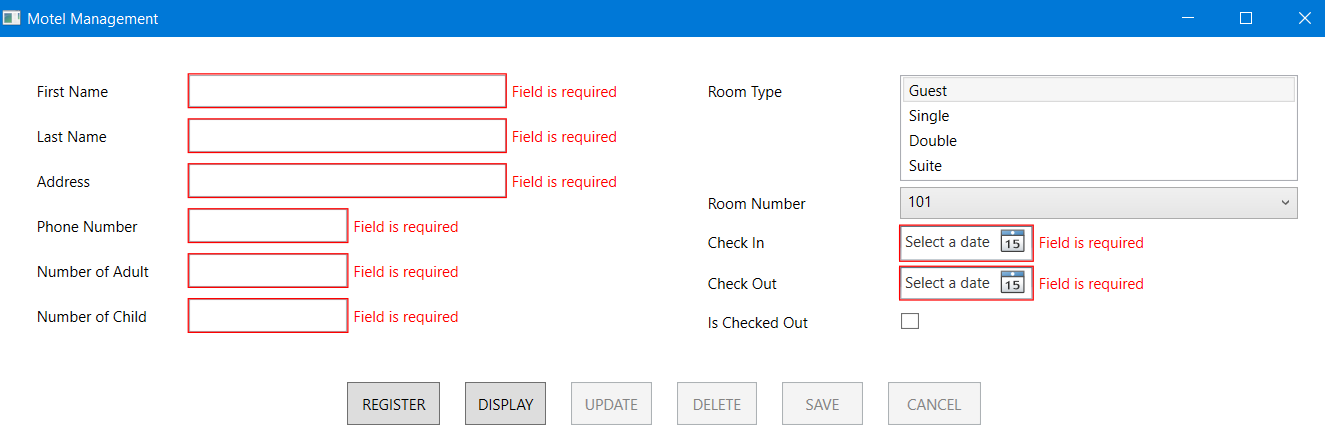
**GUI-Errors and Converters**

All errors are properly signaled. The program uses converters in the entity entry and in the grid

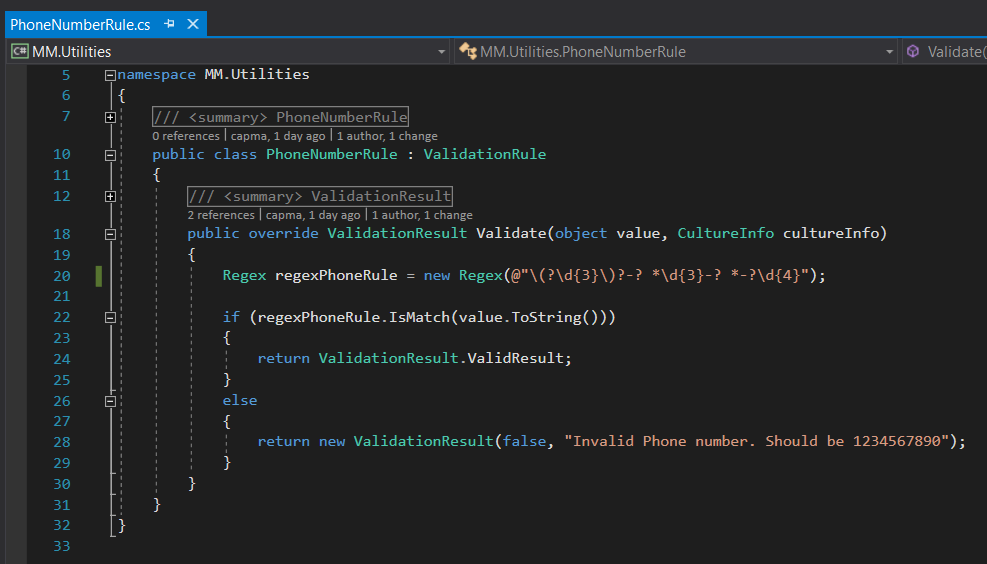
**Error handling**

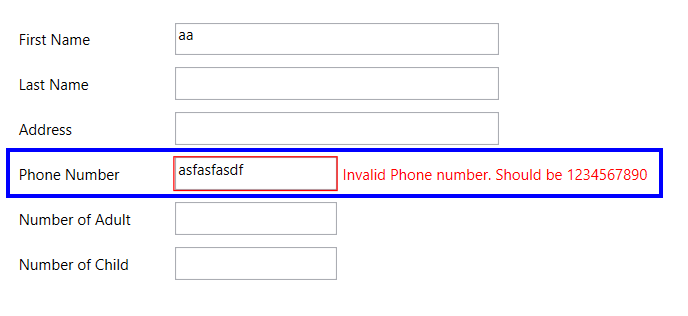
Check required field. Applying for First name, Last name, Address, Phone Number, Number of Adult, Number of Child, Check In, Check Out



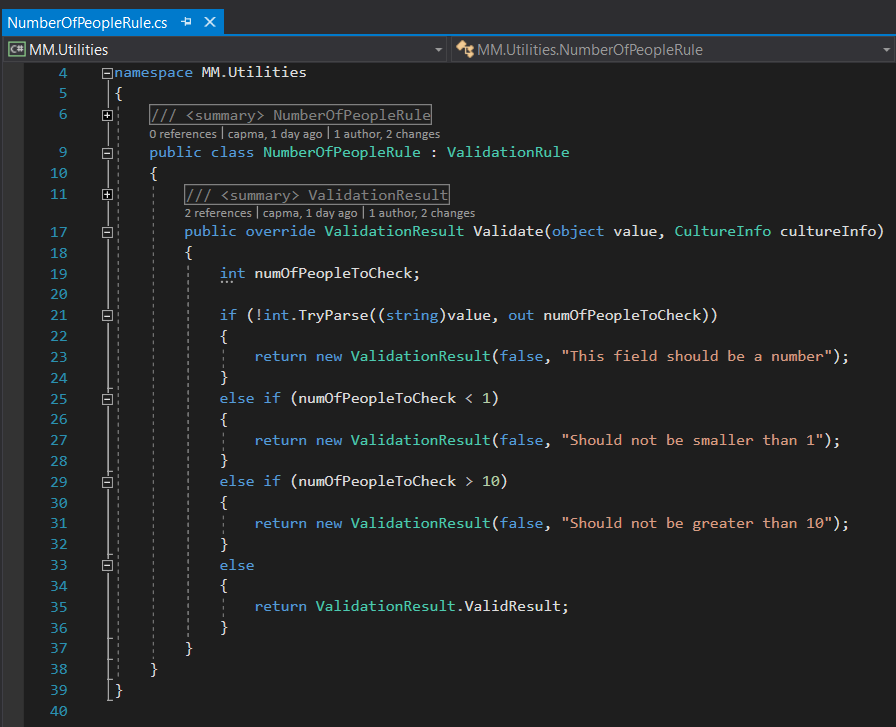


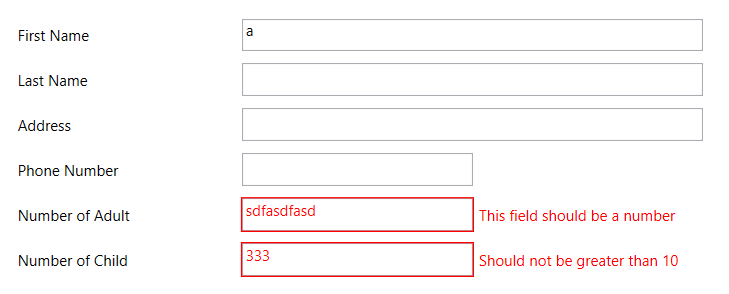
Check phone number is valid or not





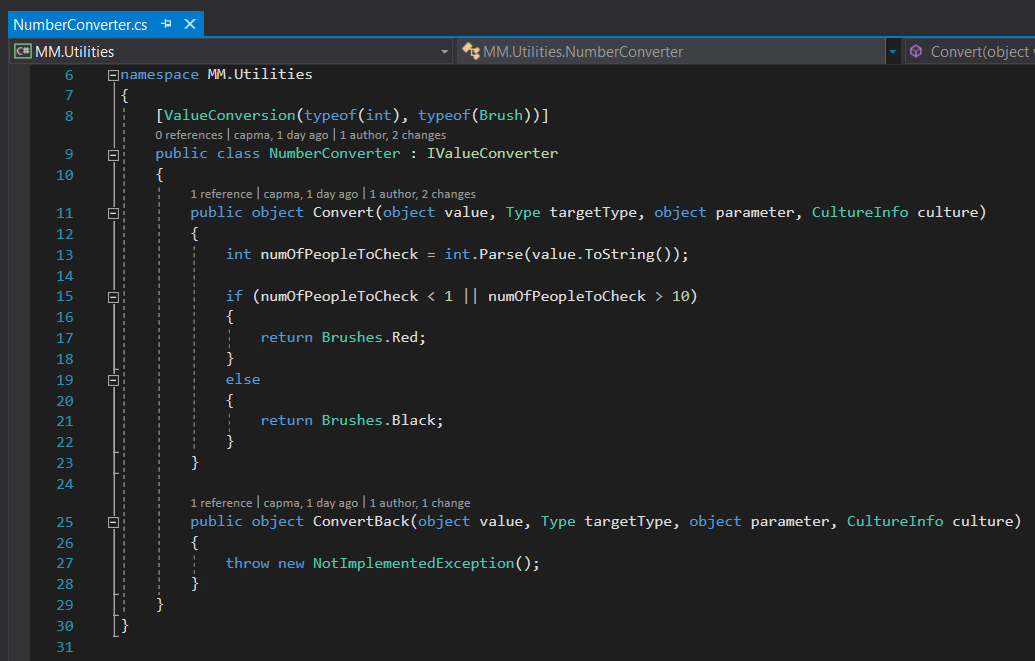
Check Number of Adult and Number of Child are valid or not

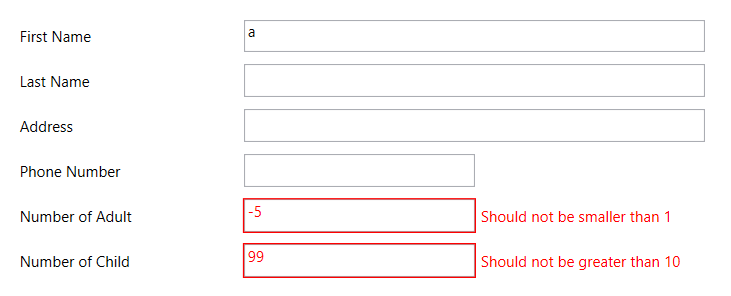


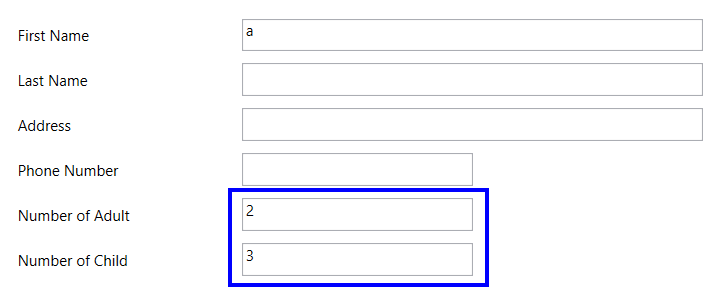


Converter handling

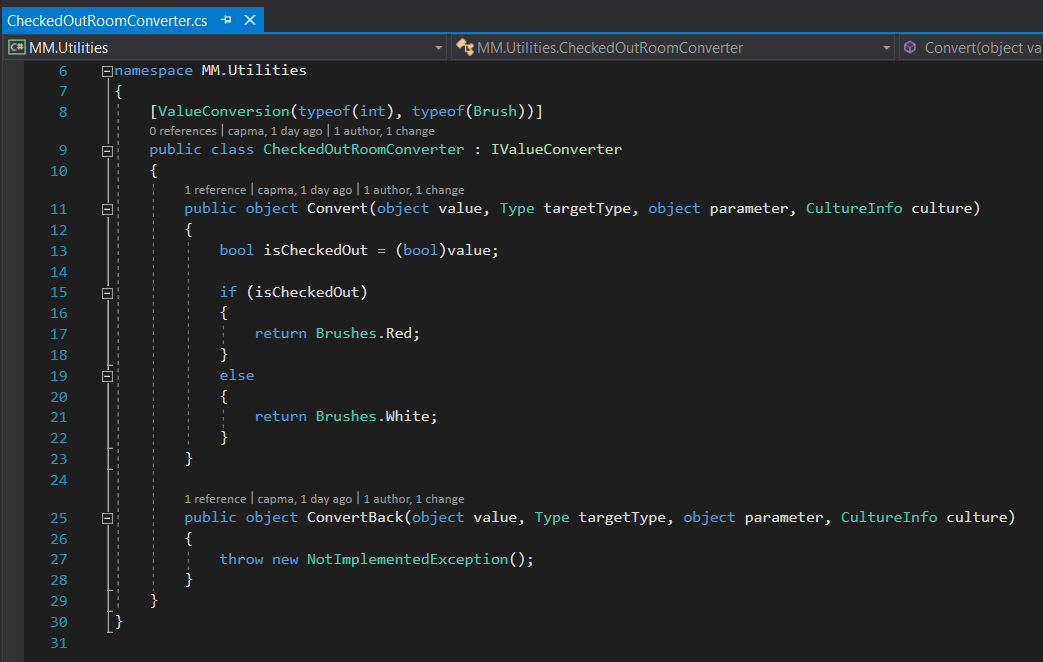
If Number of Adult, Number of Child is invalid then the foreground is Red, or else it would be Black

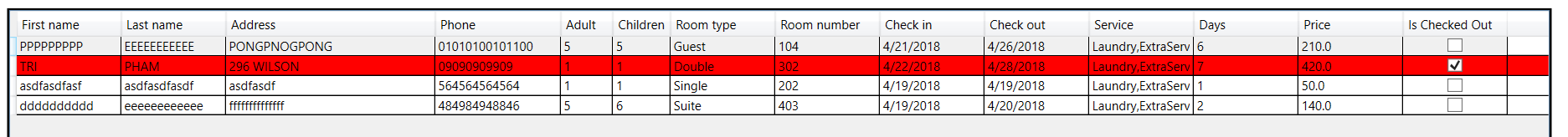






In DataGrid, when a room is checked out, the record is colored to Red, else it would be White





**Code-Proper Distribution**

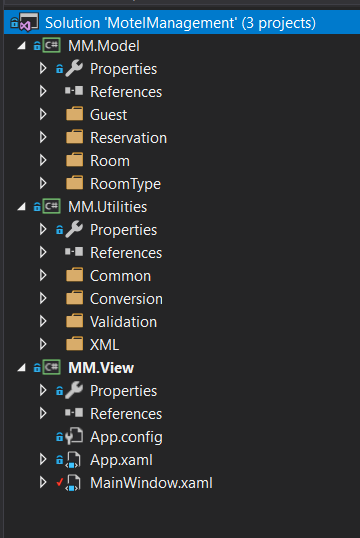
All code is properly distributed between screens and libraries

The project is structured into 3 separated projects

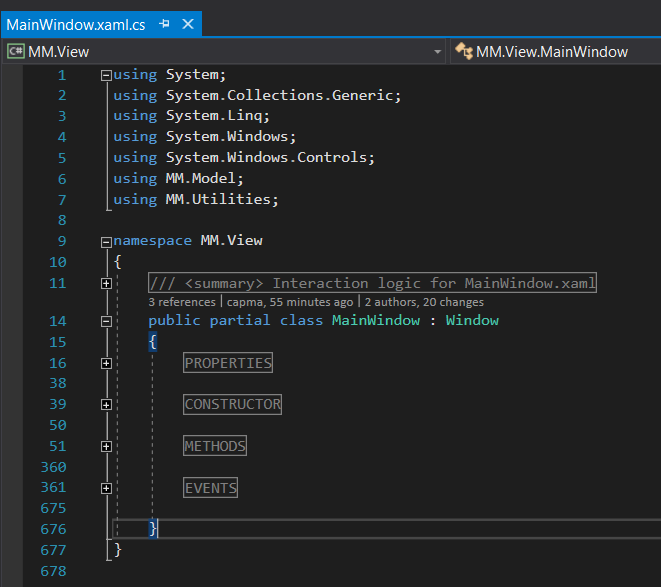
All model classes are placed in the [MM.Model] project.

All the conversion, error handlings are placed in the [MM.Utilities] project.

The main and View is placed in the “MM.View” project

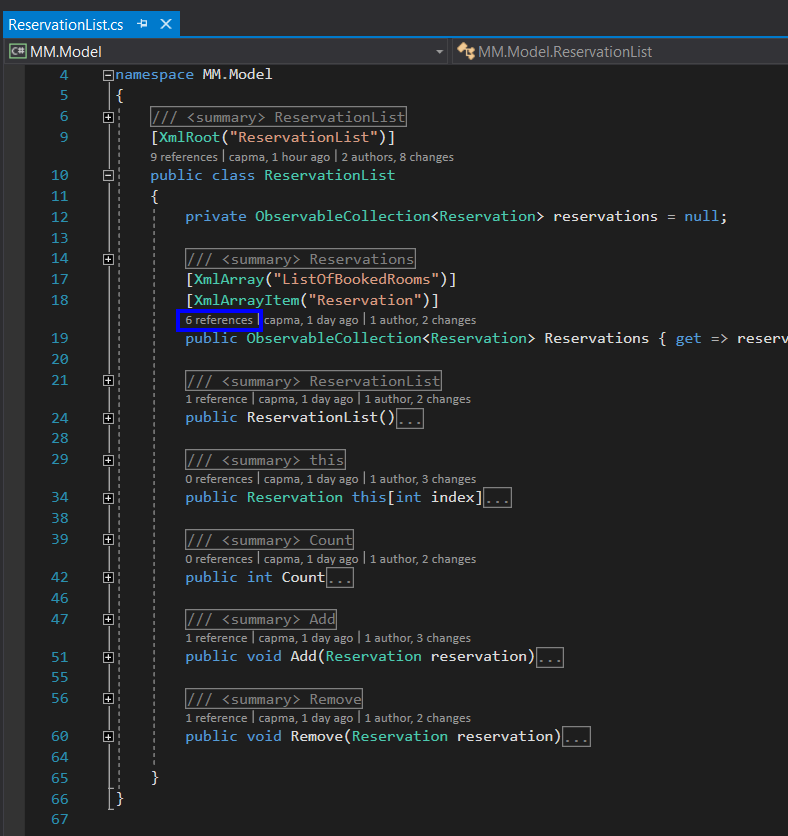


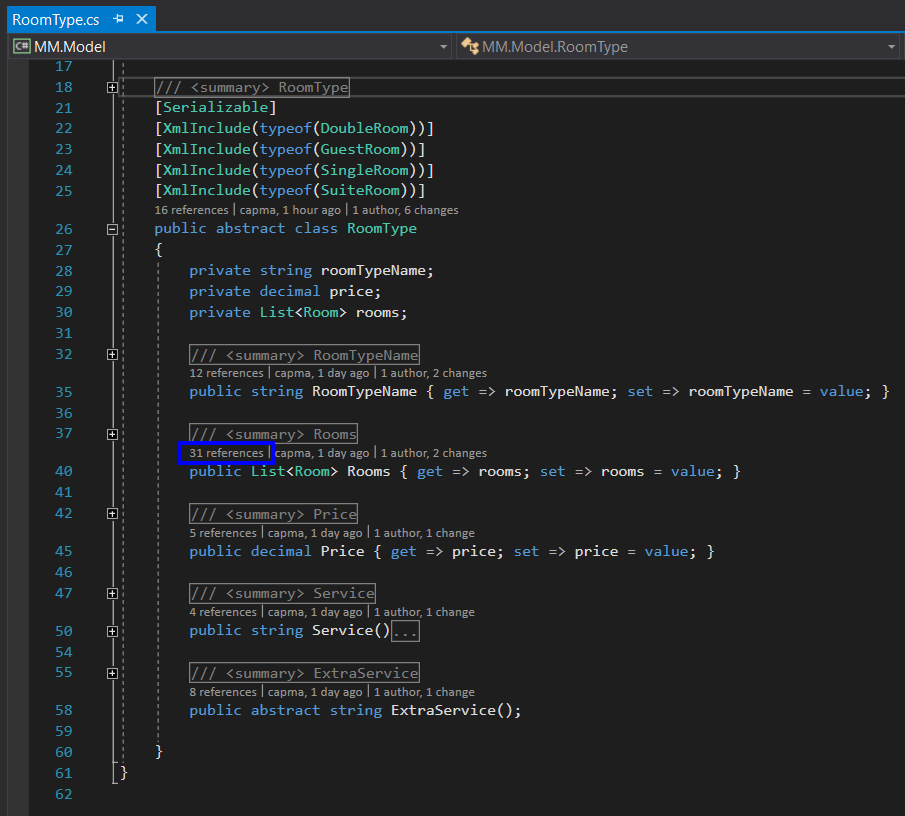
No class in the screen area.

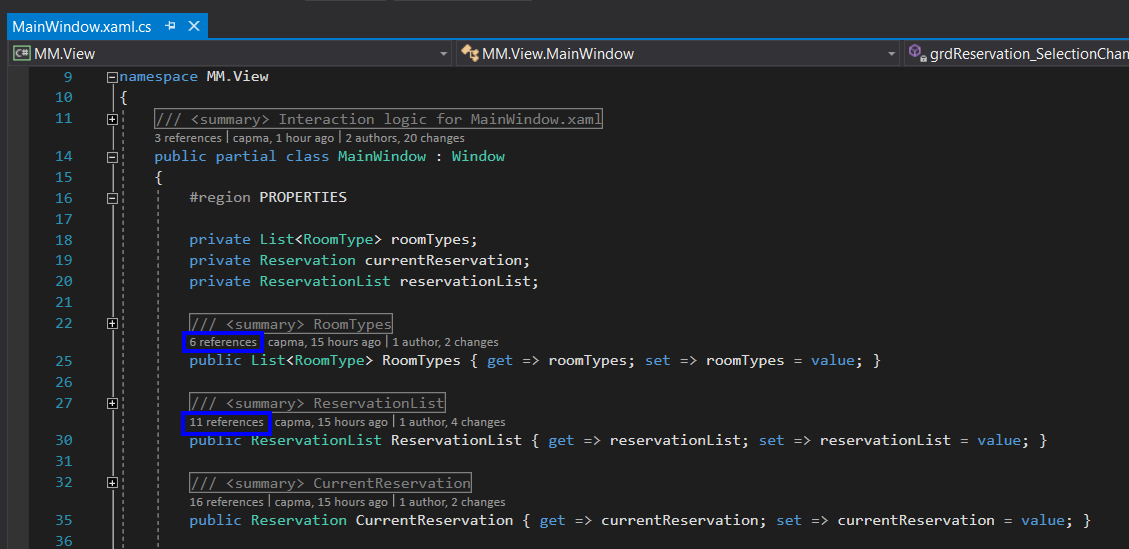


**Code-Use of Collections**

The code uses proper collections for binding. All collections are generic



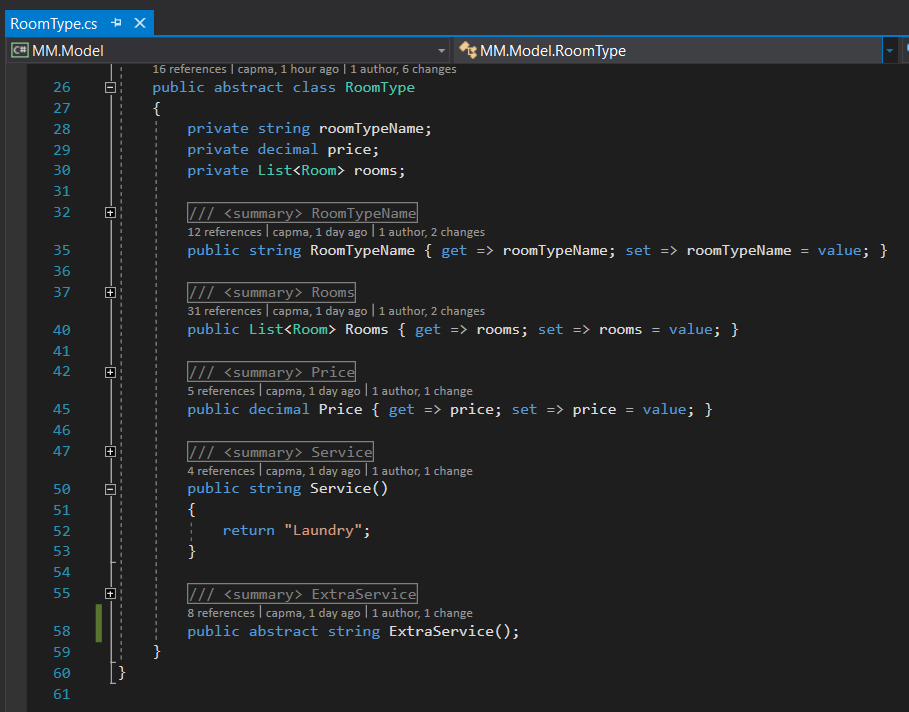




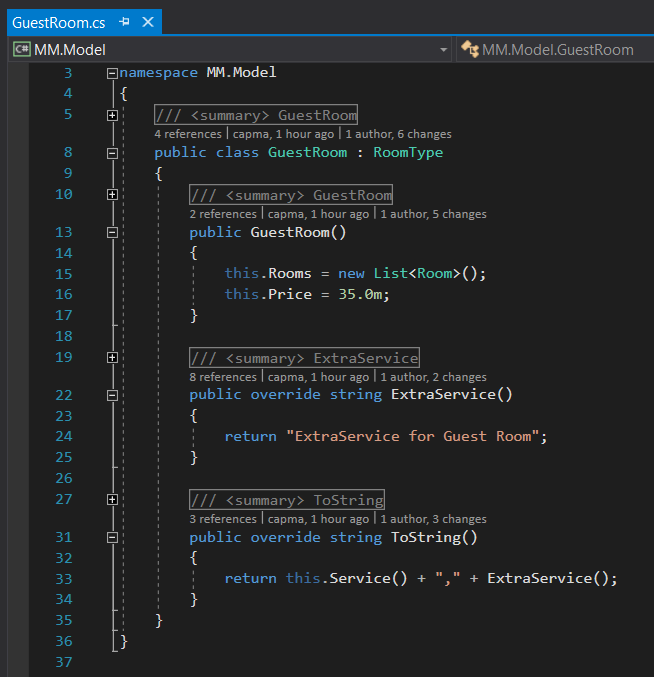
**Code-Class Design**

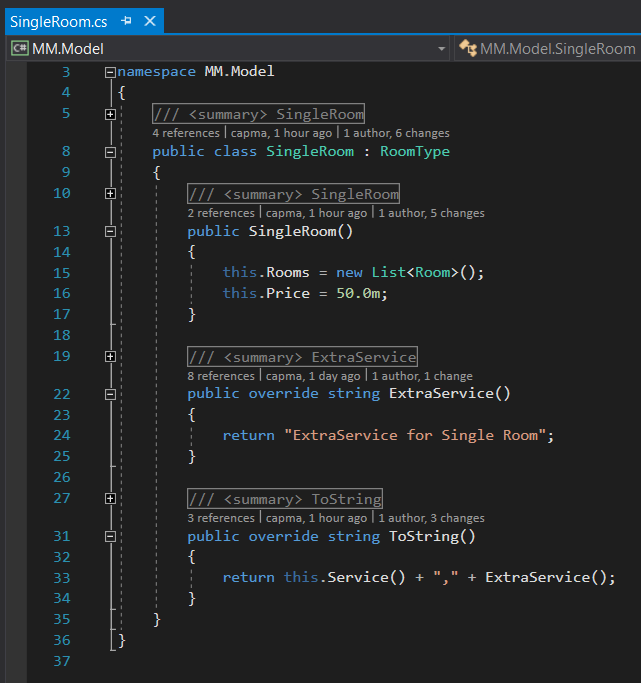
The code uses proper class design. All polymorphic features are in place. There is a proper use of overloading/overriding

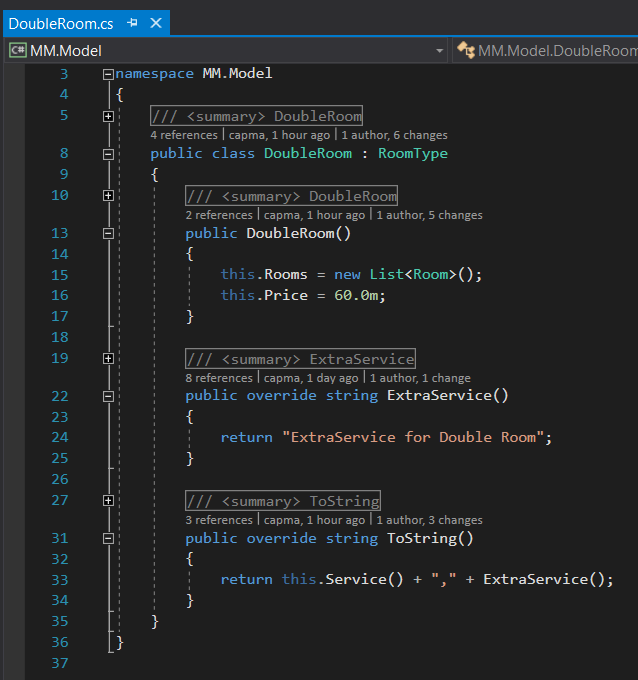
We designed RoomType as an abstract class and it has no constructor.

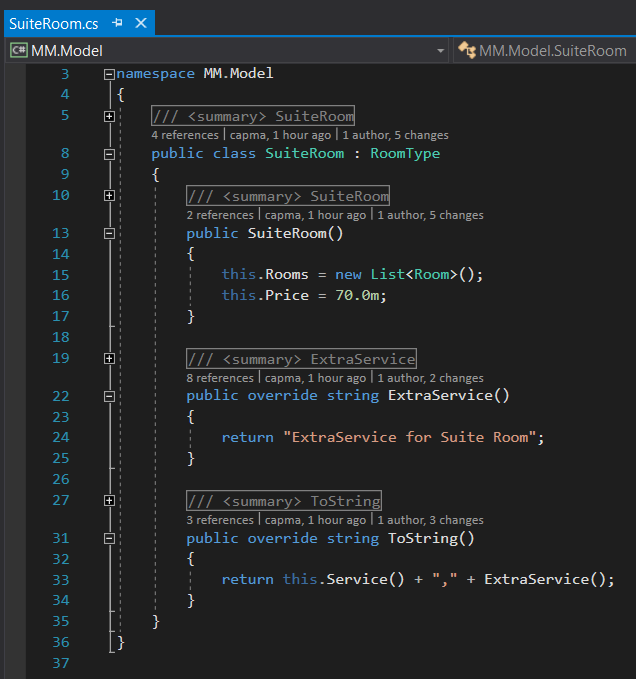


There are 4 derived classes: GuestRoom, SingleRoom, DoubleRoom and SuiteRoom

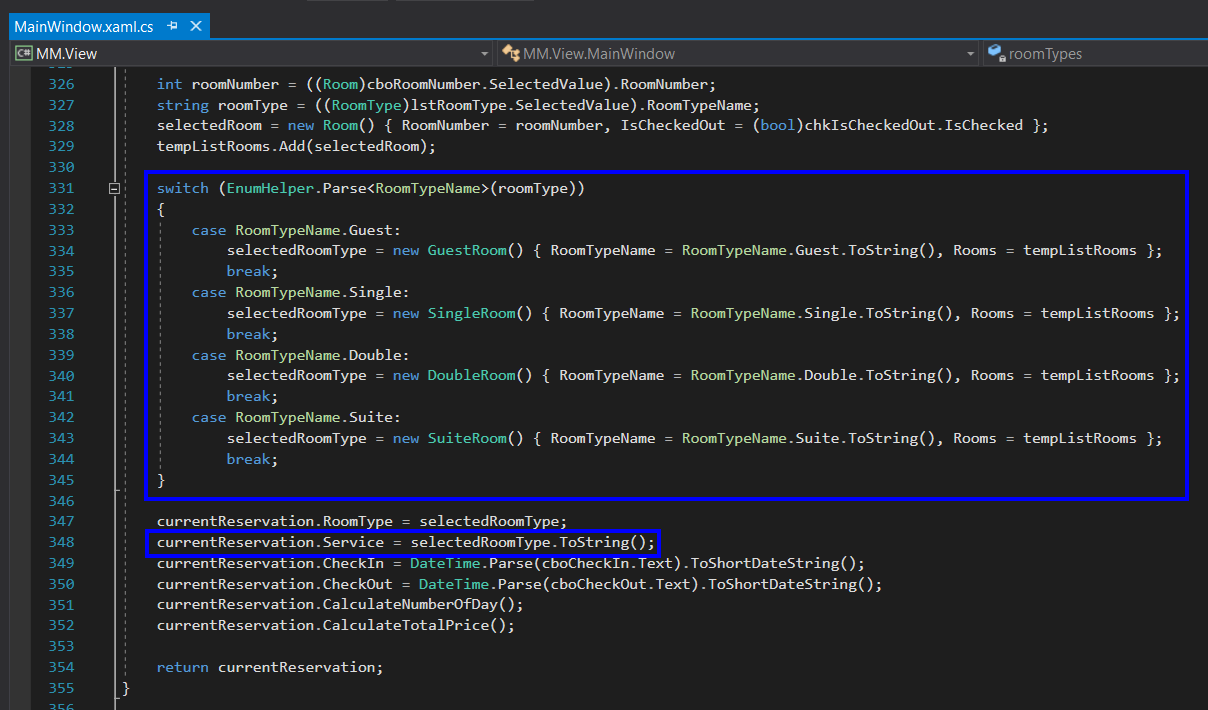






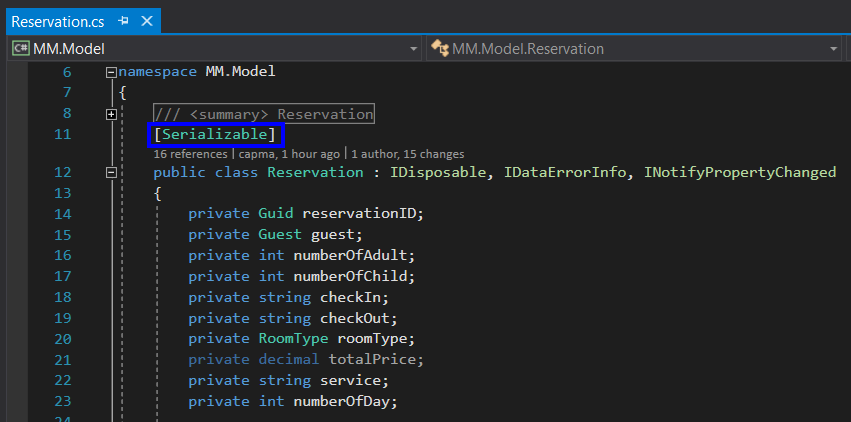


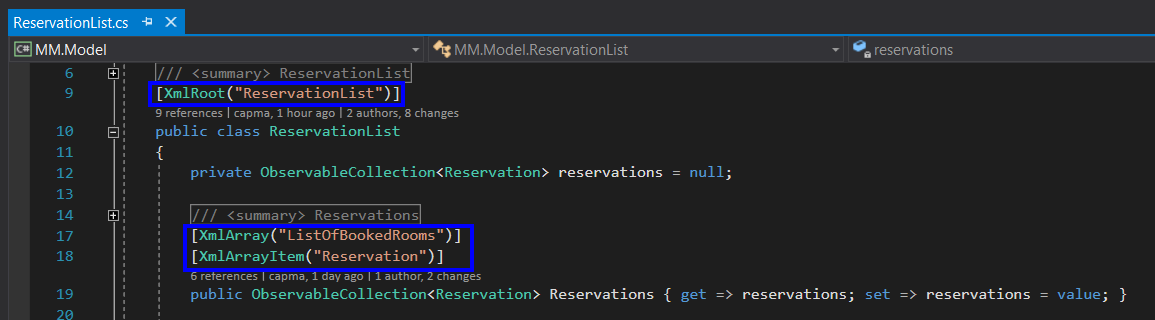
Using:

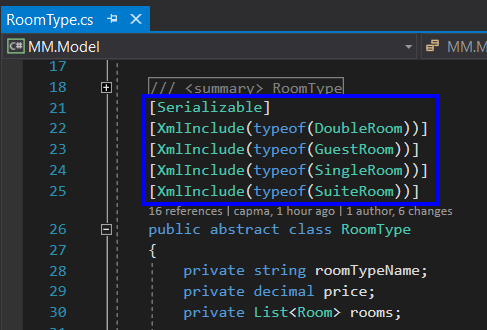


**Files-Saving and Retrieving**

All saving and retrieving of data is done through serialization. No tweaking of data was needed

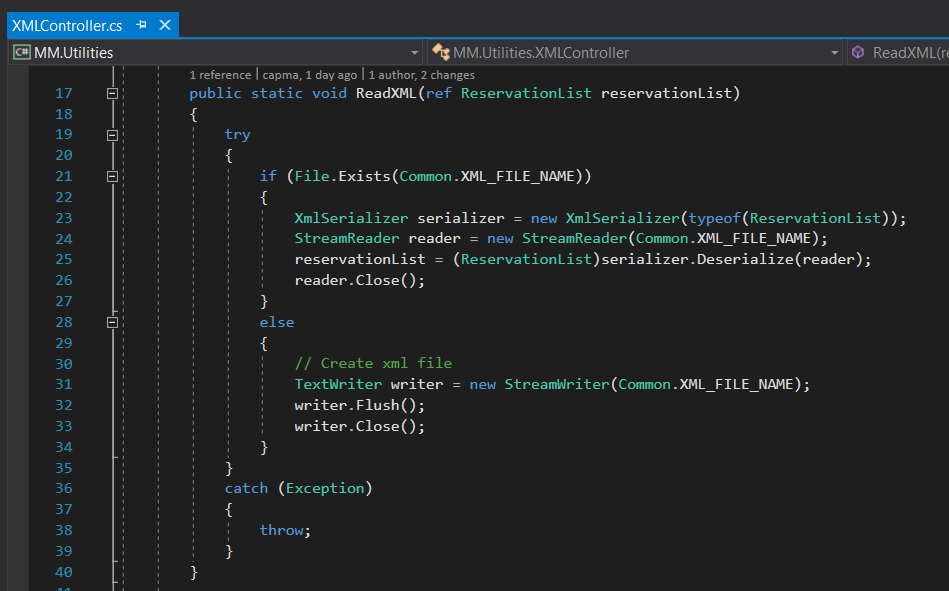




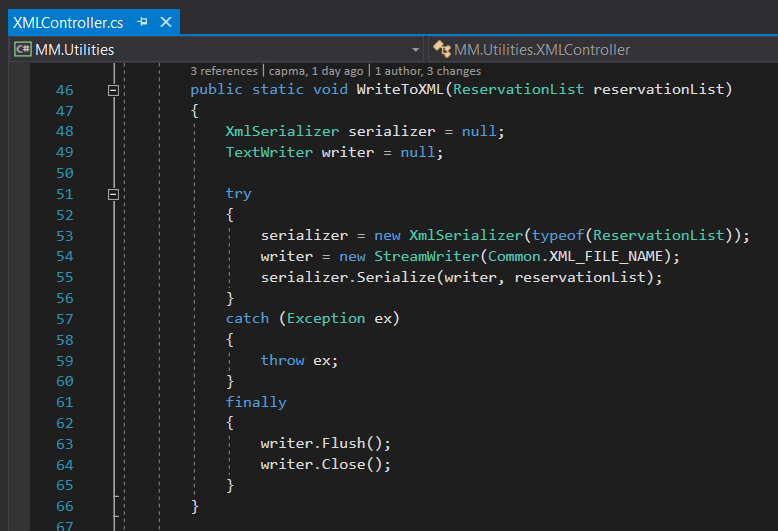




Read XML

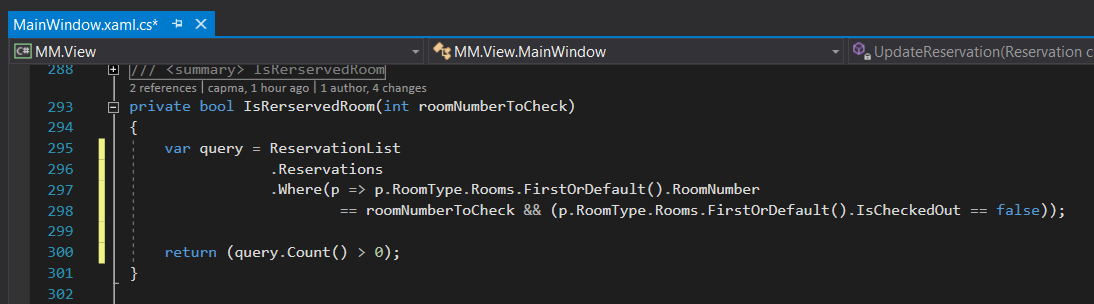


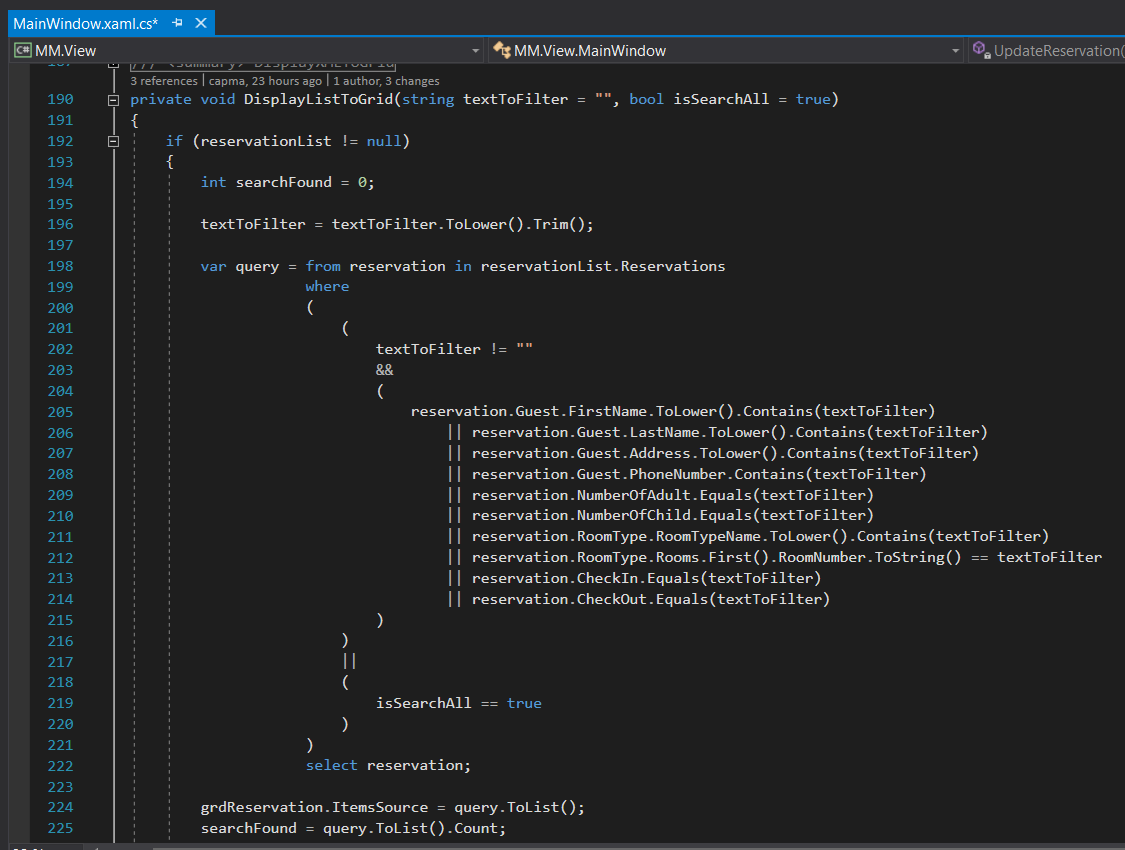
Write XML



**LINQ – Queries**

LINQ is used on the data collections. The use of LINQ is fully justified





**Documentation**

Please refer to the word file: [MotelManagement\_Guideline.docx]

**General Errors**

The code is compiled and running properly

