29.4.2023

Reservation system

Alex Mecklin 907666, Automation and robotics

Table of Contents

1. General description ... 2

2. Instructions to the user ... 2

3. External libraries ... 2

4. Structure of the program ... 3

5. Algorithms ... 4

6. Data structures ... 4

7. Files ... 4

8. Testing ... 5

9. Known shortcomings and flaws ... 5

10. 3 best and 3 worst areas ... 6

Best areas ... 6

Areas that could be improved ... 7

11. Changes to the original plan ... 7

12. Realized order and scheduled ... 7

13. Assessment of the final result ... 7

14. References ... 8

15. Attachments ... 8

1. General description

The project is reservation system for a hotel, where guests can book three different kinds of rooms for a length of their choosing. The customer can add optional comments when they are booking. The reservation system has a graphical user-interface, a unique feature and some extra design features that were not described in the instructions.

The reservation system can be used to access all of the hotels and guests reservations, so the system is meant to be used by hotel personnel, not by customers directly.

The project meets all of the criteria for the hard difficulty.

2. Instructions to the user

The program is run by running the main.py file. After running the program a small window with options on what to do pops up for the user. The user can use the program to make reservations in the hotel, print all of the hotels reservations within a time-interval, print all of a guests reservations and remove reservations from the system.

After running the program, the program tells you what you can do and what to do next, in order to make the system easier to understand.

3. External libraries

I have used the datetime library in addition to the Pyqt6 library. The reason for this is datetime is more familiar to me than the QDate class of Pyqt6, and when I started the project I was not aware that the QDate class existed, so instead I used the datetime library.

As QDate and datetime is similarly implemented It doesn’t affect the program or the difficulty of coding the program.

4. Structure of the program

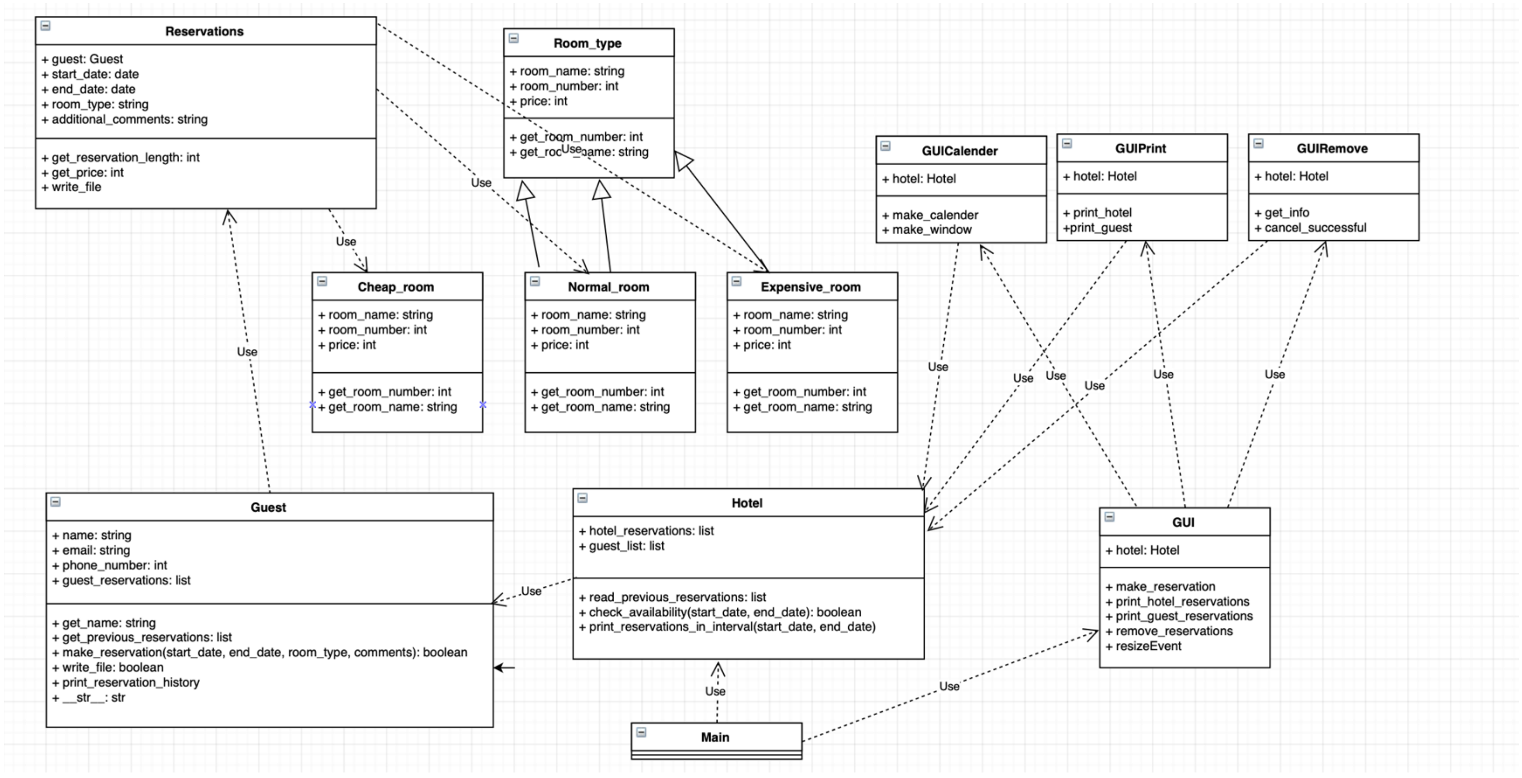


Figure 1

As can be seen from the UML diagram, the main function is used to run the program. The most central classes are the Hotel and the GUI class.

The GUI class is used to implement the first popup window, after which depending on what the user wants to do, the GUI class calls one of the other gui classes. The other gui classes are then used to make new windows with the specific task the user wants to do. Splitting the different functionalities of the gui into different classes like this makes the program easier to understand and debug.

The Hotel class is used for almost everything, and the methods for making reservations, checking availability, removing reservations, reading from the text file and writing to the text file are the most central methods of the class, and of the whole program. The Hotel class uses the Guest class and a new Guest object is created for every new guest that makes a reservation in the hotel. For every reservation that is made, a new reservation object is also created.

The Reservation class uses the different room classes when calculating prices for the reservation. The different room classes are child classes to the RoomType class.

In addition to my own classes I have also used classes from Pyqt6 and from the datetime library.

5. Algorithms

The most central algorithm in my reservations system is the algorithm of reading previous reservations that has been saved to a text file when the program has been previously run. All reservations are saved in a list when the program is running, but in order to save reservations when the program is closed, the reservations are written to a text file.

When the program is starting, the first thing the algorithm does is starting to read the text file. For every reservation it creates a new guest object if the guest is not already a customer of the hotel. After that it saves the guests info into a dictionary, and the guests reservation into a list with all of the other guests reservations.

Additionally, the algorithm also saves all of a guests own reservations into a guest specific list, so all of the guests have an own list with only their reservations in it.

This algorithm is the first thing the program executes. The algorithm is essential to the functionality of the program, and without it the program wouldn't be able to save data after being shut down.

6. Data structures

I used lists and dictionaries to save customer info and reservations info in my program. Dictionaries were a good choice for my program when saving a guests info because I could use the guests phone number as the key. As every guest has a unique phone number this worked better than using their name as the key for example, as guests could have the same names.

I used a list for storing all the reservations made. This was a good option because a list is easy to iterate through, and a dictionary couldn't be used since one guest can have multiple reservations, which means using their phone number as the key would be problematic. Additionally, there was no need to have any more complex data structure to store the reservations so a simple list was the best option.

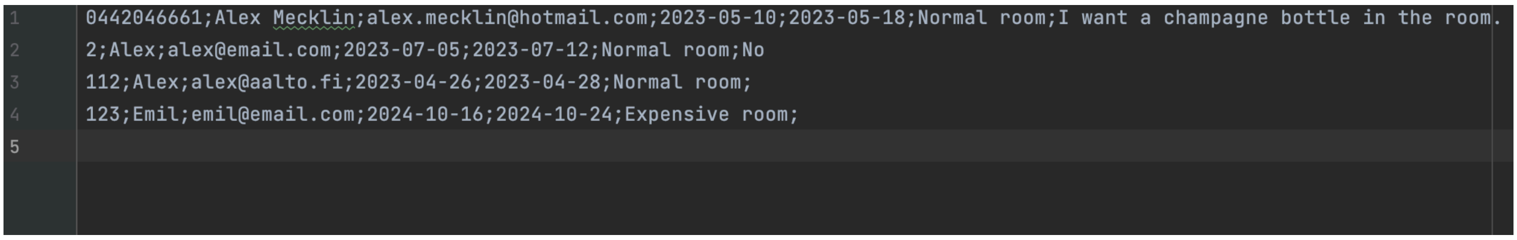
7. Files

The program uses a text file to save reservations when the program is shut down. The text file is in the format:

Phone_nr;Name;Email;check-in;check-out;Room-type;Comments\n

Every variable is a string. The variables are separated by a semicolon. Each reservation is one line in the file. The comments variable can be empty as comment are not mandatory. At the end of the file there needs to be one, and only one, empty line.

Below is an example of how the text file can look:



```
0442046661;Alex Mecklin;alex.mecklin@hotmail.com;2023-05-10;2023-05-18;Normal room;I want a champagne bottle in the room.
2;Alex;alex@email.com;2023-07-05;2023-07-12;Normal room;No
112;Alex;alex@aalto.fi;2023-04-26;2023-04-28;Normal room;
123;Emil;emil@email.com;2024-10-16;2024-10-24;Expensive room;
```

Figure 2

The program has three different kind of rooms, Cheap room, Normal room and Expensive room. The room names need to be exactly like this if the text file is changed manually, otherwise the program doesn't recognize them. If the program is run normally this isn't a problem, as you wouldn't need to change to text file manually, but the program does that for you. Changing the text file manually is only for testing purposes.

8. Testing

The most essential parts of the program were tested using unittests. These include testing availability, reading file, writing to file, calculating price, printing info and making a reservation. Most of the methods tested were from the Hotel class, as it is the most used and most important class of the program.

Different tests were implemented as the different methods were implemented. Different tests were also used to test different inputs to the methods. For example one test that tests the method then the reservation shouldn't go through, and another test that tests the same method when the reservation should go through.

In total the program has 11 unittests and it passes all of them. The test that tests the method for writing to the text file needs to be tested individually, as it disrupts the other tests when writing to the text file.

9. Known shortcomings and flaws

There are no major shortcomings or flaws in the project. The project handles error situations so it should never crash or make an inaccurate booking for example.

However, when taking info from a guest, the program does not check that the name, phone number or email is in the right format. All the variables are just strings, which means that for example the name could consist of special characters or numbers. The phone number could also consist of other characters than integers and the email does not have to be in a normal email format.

The program still works like this, but if I were to continue the development of the program, I would make it so the name should just consist of letters, the phone number must be just integers and in a known country format and the email address also has to be in a known format, like @hotmail.com or @gmail.com, etc.

Although these things could be improved, I still think the program handles these situations sufficiently, as the program prints a "confirm reservation" info section with all of the inputs, that the user reads before confirming their reservations. This makes mistakes that the user has made when inputting their info easier to spot.

10. 3 best and 3 worst areas

Best areas

1. The best part of the program, and the hardest to implement, is the highlight feature in the calendar. When the user chooses a check-in date and then the check-out date, those dates and also all of the dates in their interval becomes painted green, as to indicate that those are all the dates the room is reserved by the user.

   If the user chooses "wrong dates" like for example a check-in date that has already passed, or a check-out date before the check-in date, the dates are not highlighted, as to not cause confusion.

   Similarly, all of the reserved dates are highlighted red in the calendar.

2. Another good part of the program is its intuitive GUI and the fault-tolerant code. The GUI always instructs the user what to do and what to input. The GUI also has images to make it look better. The first pop-up window has a background image from a beach, and if the user scales or changes the window size, the background image changes with the window.

   The program is also very fault-tolerant, meaning the user can input or click on buttons in the wrong order, but the program does not crash or make any errors, instead the GUI informs the user what to do instead.

3. The third good part of the program is its scrollable windows. Every window except the smaller pop-up windows is completely or at least to some extent scrollable.

   When making a reservation the whole window is scrollable to make everything fit in the same window without making the buttons and info too small. The hotels and guests reservations are also printed in a scrollable window, to make sure all of the reservations can be the same size and still be visible, even if there are a lot of reservations.

Areas that could be improved

1. As discussed in the previous section, 9. Known shortcomings and flaws, the format checking on the user input could be improved. This is further discussed in that section.

11. Changes to the original plan

The classes and functions and how they communicate remained similar to what was described in the plan. Small changes were made to some functions and how the reservations are stored in the program, and also the format of the text file was slightly changed. Some new methods were added.

The design of the GUI changed drastically from the original plan, as when I wrote the plan I had no idea how to code a GUI and what was even possible to add to the GUI.

The implementation order and the time scheduled for the project was according to the plan.

12. Realized order and scheduled

The back end of the program was first implemented before the first checkpoint. Tests were added along the way to ensure that the added parts functioned as they should.

Before the second checkpoint a working GUI was made and connected to the back end of the program. After that just small changes and design changes were made.

13. Assessment of the final result

I think the project is done in a way that fulfills the requirement of the hard difficulty. The program has a GUI with multiple windows, that are easy to use for the user. The program handles all error situations and has all of the features described in the project description. In addition to this the program also has some extra features that were not described in the project description, such as the reservation date highlight feature of the calendar for example. The program also has a unique feature, which is that you are able to remove existing reservations. The program has no major flaw that would affect its usability.

If I were to continue developing the program I would improve the input sections, such that the users input has to be in a known format. I would also add more images and such to make the design stand out more.

The class division is good, with every bigger part of the program being handled by its own class. The data structures used in the program could have been used more efficiently. Although the current solution works, with the reservations being stored in lists in a list, it could probably have been implemented in a more intuitive way.

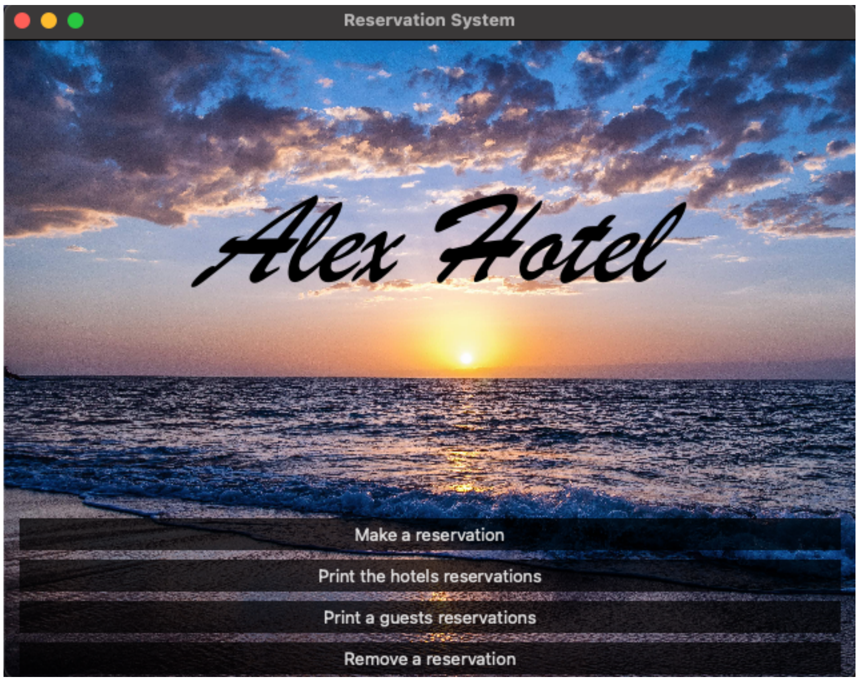
As the program uses constants and variables that are easy to change, the program should be easily expandable and changeable. Additionally, the program has a good class division without repeating code, with each method handling just one specific task.

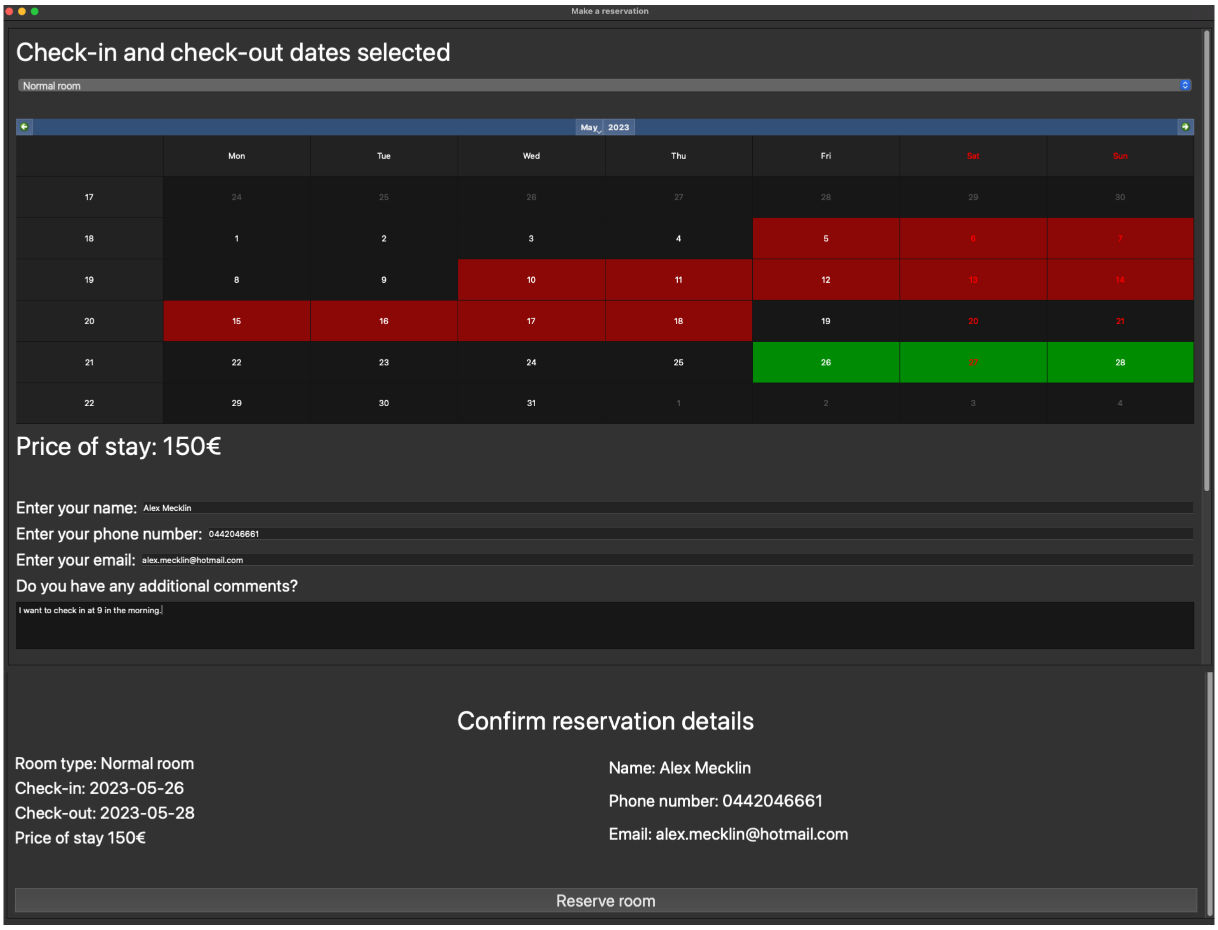
14. References

When making my project I used the textbook from A+ and a website with descriptions of all PyQt6 classes (https://doc.qt.io/qt-6/classes.html).

15. Attachments

Below are some screenshots from every window in the program, showing all of the program's functionalities:





Make a reservation

Check-in and check-out dates selected

Normal room

May 2023

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
17	24	25	26	27	28	29	30
18	1	2	3	4	5	6	7
19	8	9	10	11	12	13	14
20	15	16	17	18	19	20	21
21	22	23	24	25	26	27	28
22	29	30	31	1	2	3	4

Price of stay: 150€

Enter your name: Alex Mecklin

Enter your phone number: 0442046661

Enter your email: alex.mecklin@hotmail.com

Do you have any additional comments?

I want to check in at 9 in the morning.

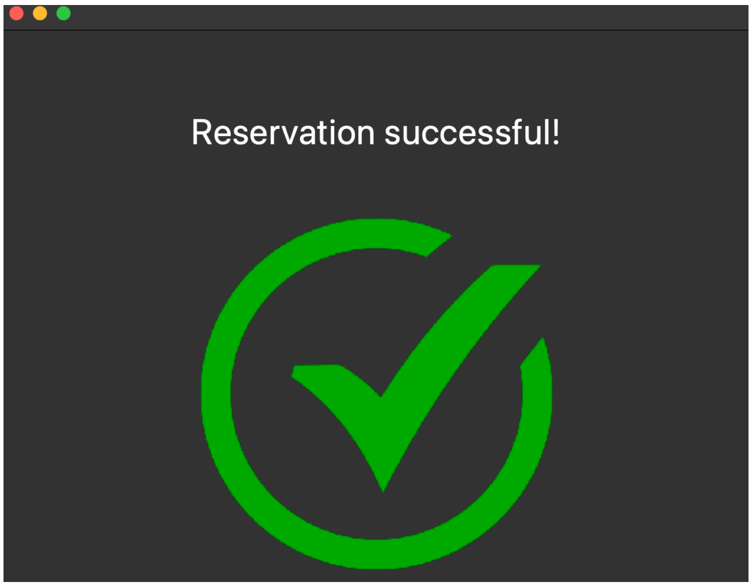
Confirm reservation details

Room type: Normal room
Check-in: 2023-05-26
Check-out: 2023-05-28
Price of stay 150€

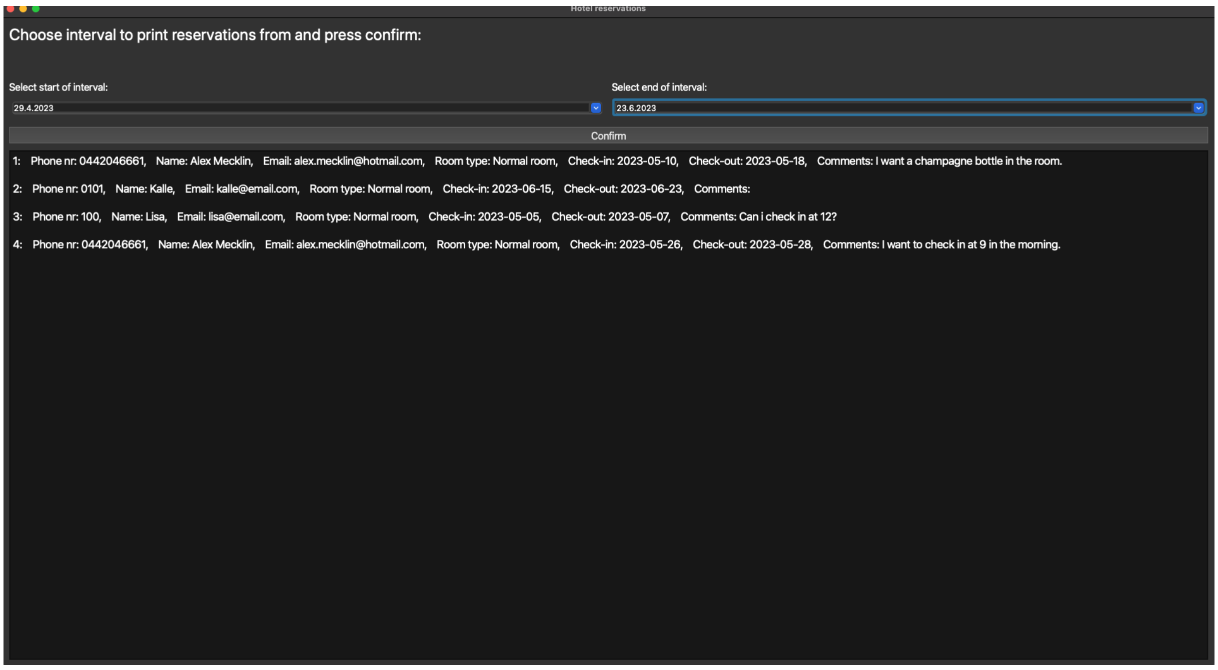
Name: Alex Mecklin

Phone number: 0442046661

Email: alex.mecklin@hotmail.com



Reservation successful!

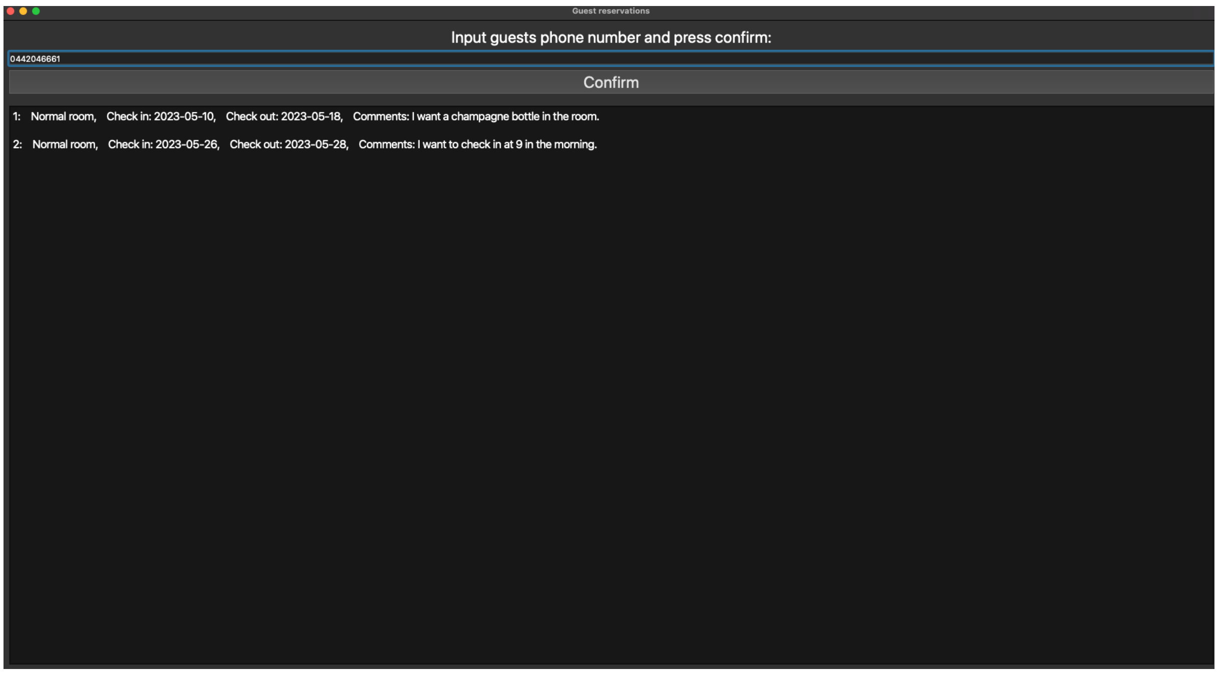


Choose interval to print reservations from and press confirm:

Select start of interval: 29.4.2023

Select end of interval: 23.6.2023

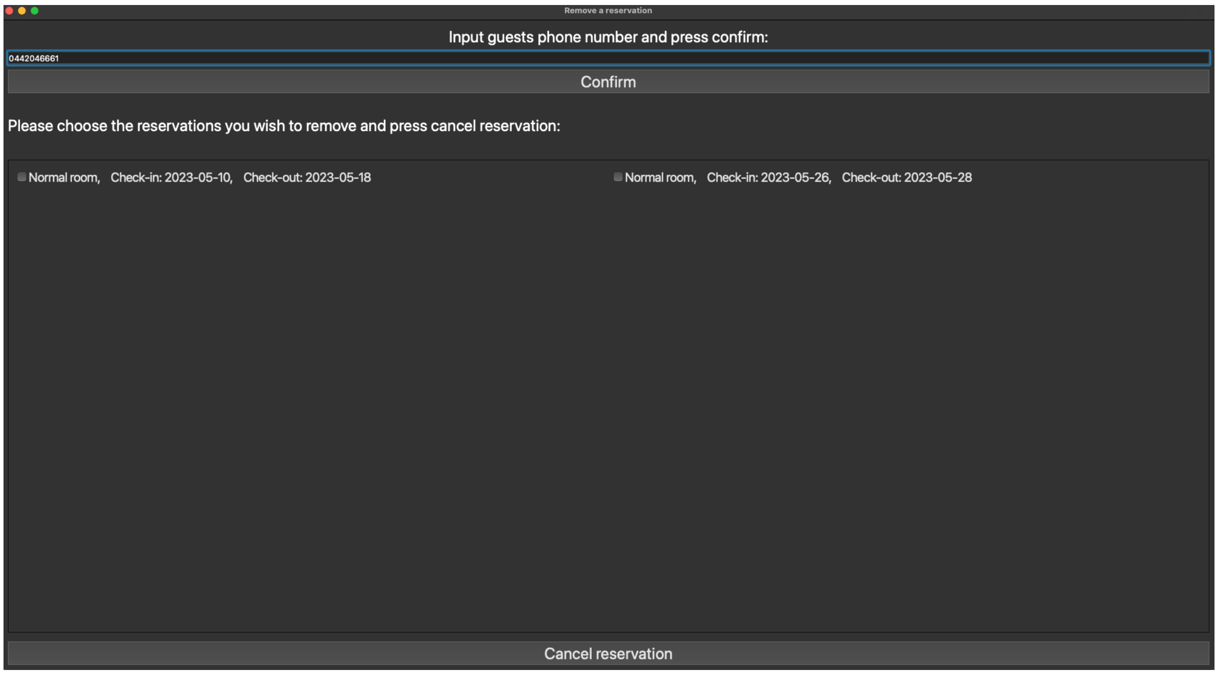
```
1:   Phone nr: 0442046661,   Name: Alex Mecklin,   Email: alex.mecklin@hotmail.com,   Room type: Normal room,   Check-in: 2023-05-10,   Check-out: 2023-05-18,   Comments: I want a champagne bottle in the room.
2:   Phone nr: 0101,   Name: Kalle,   Email: kalle@email.com,   Room type: Normal room,   Check-in: 2023-06-15,   Check-out: 2023-06-23,   Comments:
3:   Phone nr: 100,   Name: Lisa,   Email: lisa@email.com,   Room type: Normal room,   Check-in: 2023-05-05,   Check-out: 2023-05-07,   Comments: Can i check in at 12?
4:   Phone nr: 0442046661,   Name: Alex Mecklin,   Email: alex.mecklin@hotmail.com,   Room type: Normal room,   Check-in: 2023-05-26,   Check-out: 2023-05-28,   Comments: I want to check in at 9 in the morning.
```



Input guests phone number and press confirm:

0442046661

```
1:   Normal room,   Check in: 2023-05-10,   Check out: 2023-05-18,   Comments: I want a champagne bottle in the room.
2:   Normal room,   Check in: 2023-05-26,   Check out: 2023-05-28,   Comments: I want to check in at 9 in the morning.
```



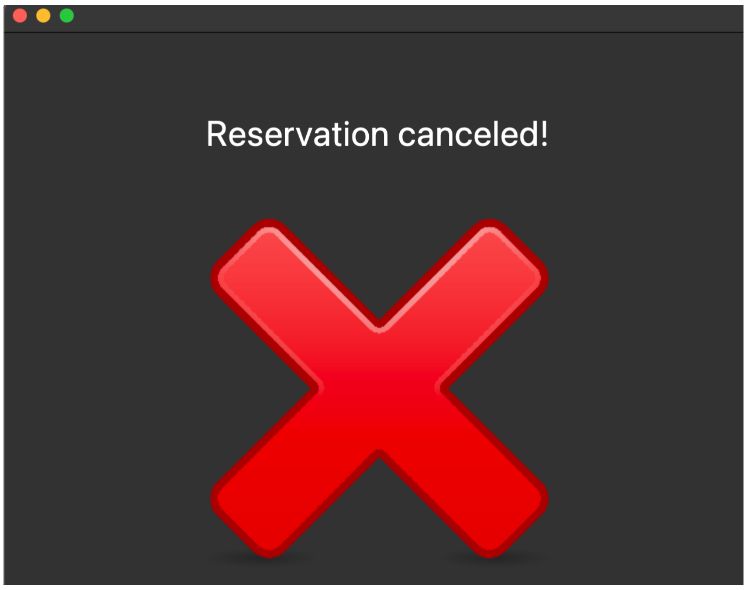
Input guests phone number and press confirm:

0442046661

Please choose the reservations you wish to remove and press cancel reservation:

Normal room, Check-in: 2023-05-10, Check-out: 2023-05-18

Normal room, Check-in: 2023-05-26, Check-out: 2023-05-28



Reservation canceled!