HOTEL RESERVATION SYSTEM PROJECT PLAN

By: Ali Amaan

Student Number: 906955

Study Year: 2022

Date: 25th February 2022

Contents

General Description

This project plan is to create a reservation system for an imaginary hotel.

The system contains information on the scheduling of room bookings, customers' data, and the related hotel resources/services. The related resources/services, termed in this plan as 'add-ons', can include, but are not strictly limited to, valet services, wellbeing services, meal services, grooming services, car rental services, etc.

The program maintains a record of the hotel resources on a calendar according to their reservations. Error handling will be implemented to take of out-of-bound bookings, resolve reservation conflicts, and wherever needed. A record of the customer may also be maintained.

Reservations can be displayed for single day or an interval of days, as specified by the user.

A tentative list of unique aspects, above and beyond the specified conditions for difficulty level in the following section, that can potentially be incorporated as features into the program is as follows:

- A functionality to export the report of the reservations in a '.pdf' format.
- A functionality to edit reservations only to 'add/remove' add-ons from the reservation.
- Customer Loyalty system with applicable discount's depending on the Loyalty level.

Difficulty Level

This project is aimed to accomplish the '**Hard**' version of implementation, which includes the following requirements:

- The program prevents concurrent reservations and, if needed, informs the user about the resource already being reserved at the given point of time.
- The program marks the chosen resource as reserved for the given points/intervals of time.
- The program can be used to save information on customers (e.g. name, address, phone number and e-mail) and to connect it to a reservation.
- The reservations don't disappear when the program is closed, meaning they can be saved into and loaded from separate files.
- A graphical user interface which includes a calendar view that shows reservation times of the resources and can be used to make new reservations.
- Make the user interface as usable, clear and informative as possible.
- Additional services can be attached to reservations (e.g. daily meal service, grooming service(s), pick & drop service from the airport, etc.).
- The program saves history data on customers/services provided (e.g. basic information and past hotel reservations of the user), which can be searched and viewed.
- The reservations can be of different lengths.
- The program uses the length of the time slot and the reserved services to calculate a price and prints it out.
- Add other useful features, for example statistics or automatic reminders.
- Unittests for at least part of the program.

Use Case Description & Draft of the User Interface (UI)

The program communicates with the user using a ‘Graphical User Interface’ (GUI). The program get’s its inputs from the prompts and subsequent text boxes displayed on the interface. Outputs are either displayed on the same GUI window, pop-up window, or a generated report file.

The GUI will be implemented using an initial login system. So that administrator and customer options are differently displayed. However, for both the user types will have a menu bar on the top side or left side of the window, enabling them to select relevant options pertaining to the different functionalities. Such as, a menu option could be for ‘Reservations’ with sub-options that could be ‘Make Reservation’, ‘Edit Reservations’, and ‘Past Reservations’. Another menu option could be for ‘Settings’ where a user can edit their personal information and an administrator can manage display option’s or change password.

The main window section of the GUI can then display relevant options, features, buttons, info-graphics, text, textboxes, etc. as per the relevant option selected from the menu.

A typical use case of the program is when a user can make a hotel room reservation in a successful manner by selecting the ‘Make Reservation’ menu option and then specifying the rest of the required information (dates, add-ons, etc.) on the main window and clicking the ‘Book/Reserve’ button.

A typical use case of the program for an administrator can be to generate a report of the reservations made by the users by specifying the interval of days in the main window after selecting the sub-menu option of ‘Reports’ from the ‘Settings’ menu.

Program’s Structure Plan

Following is a UML class diagram that is planned to be implemented.

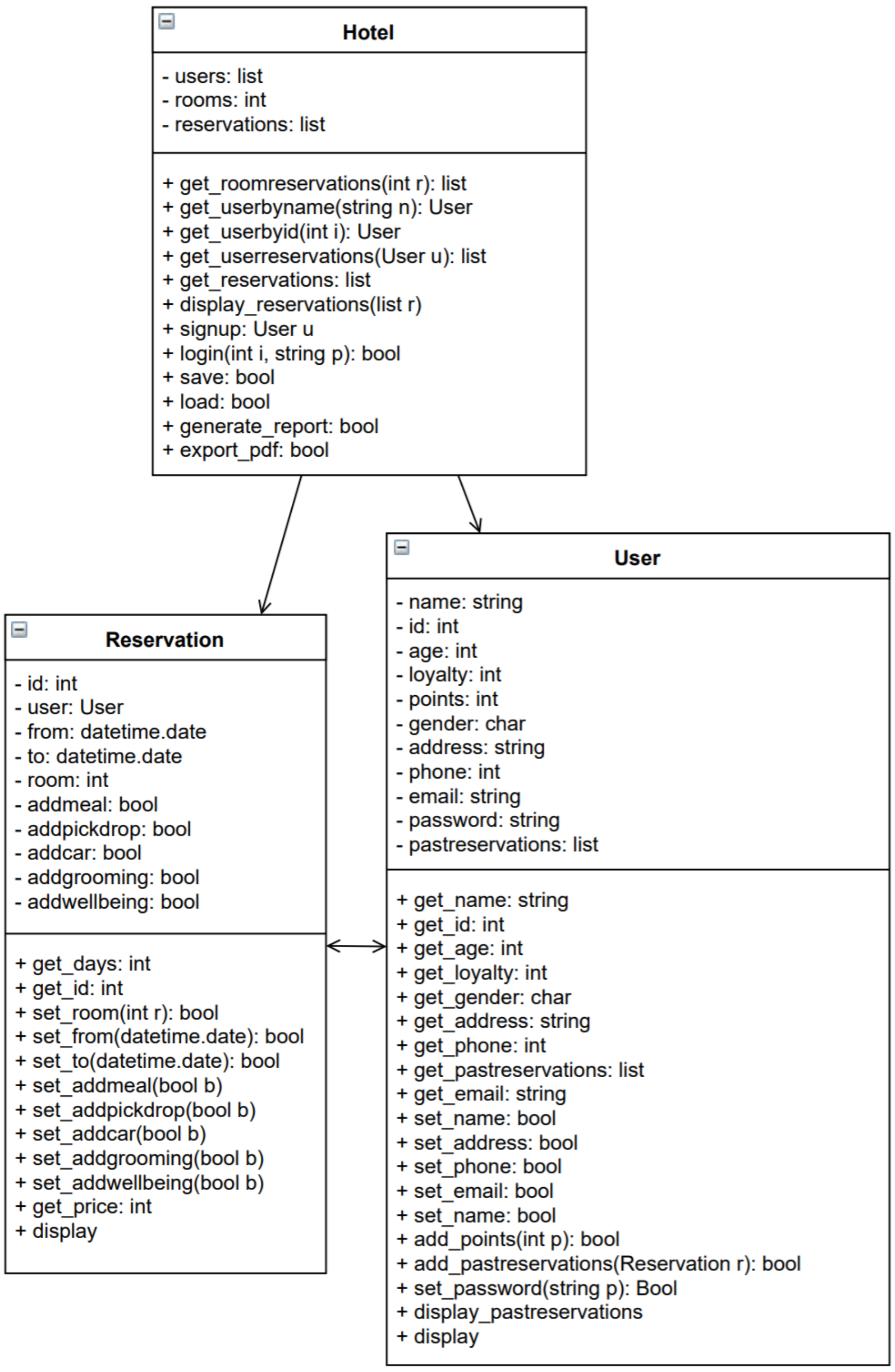

FIGURE 1: UML CLASS DIAGRAM

Data Structures

Other than Python's predefined data structures that are planned to be used as per Figure 1, a few custom classes are also planned to be defined. They are as follows:

1. 'Hotel' Class: The main class of a hotel with necessary lists and details of the hotel.
2. 'User' Class: The class for defining and storing necessary information about a single 'customer' of a hotel.
3. 'Reservation' Class: The class for defining reservations made by the 'User'. This includes all the booked services and the duration of the reservation alongside other necessary information.

This list of data structure is to be implemented in the context presented in Figure 1.

Kindly note that this list is not exhaustive and reflects a tentative plan for implementation.

Files & File Formats

The files that are intended to be used in this project are simple '.csv' text files to maintain the records of the lists of class objects defined above to be loaded, worked upon, saved, and reloaded later, according to the usage of the program.

Algorithms

The idea of the algorithm for this project is as follows.

Basically, upon commencement of the program, the user will be prompted to login. Once, logged in, there will be a welcome screen with upcoming reservations, if any, on the GUI window alongside any necessary information. Then, the user could select the option from the menu displayed to select and perform certain actions, such as making a reservation, edit upcoming reservations, view past reservation, etc. As soon as any of the option is selected, a window for that particular option opens for the user to make the required edits/inputs and/or display the requested output. After the user is done with the intended use, the user can simply choose to log off and the program will exit.

Similarly if the admin has logged in, the respective options for it will be displayed and executed after which the program can be exited.

No particularly programs are yet intended to be implemented. However, maybe for making sorting and then inserting efficient for maintaining data objects, a particularly known program for sorting maybe used. Additionally, for searching and retrieving object data, 'binary search' algorithm might be used.

Testing Plan

Testing will be done by implementing Unittests. These will be implemented for some of the individual methods of the main classes, such as display_reservations() in the case where there are none, are a few, are many, etc. The tests will also be implemented for the main (system testing) and subsequent programs (unittesting) to test the core functionality to reserve calendar day slots and perform subsequent user operations with them correctly and in other cases to test for error handling as well.

Additionally, for making it a more user-centric GUI and functionality, feedback would be collected for external black box testers and implemented.

Libraries & Other Tools

Library	**Reason of Use**
PyQt	To design and implement the GUI of the program.
Datetime	To get the dates and times for reservation processes.

**This list is not exhaustive. Other libraries can be added, subject to approval.*

Schedule

Checkpoint 1 (25.03.2022)	- Text-based implementation of major program functionality along with the subsequent error handling. Implementation of the specified classes. - Tentative features that may **not** completed by then: saving & loading from .csv files, exporting, basic GUI implementation. - Other than this a functional program would be ready by then.
Checkpoint 2 (15.04.2022)	- Completed GUI with all the functionalities. - Error handling with all the functionalities. - Unittests implemented. - Feedback on improvements and possible implementation.

Literature References and Links

As a given, the python documentation for the above specified libraries will be used and are as follows:

- https://doc.qt.io/qtforpython/
- https://docs.python.org/3/library/datetime.html

**This list is not exhaustive. Other links may be added later as they may come in to use during the project.*