

# HOTEL RESERVATION SYSTEM PROJECT DOCUMENTATION

By: Ali Amaan

Student Number: 906955

# Contents

1. Personal Information
2. General Description
3. Instructions for the User
4. External Libraries
5. Structure of the Program
6. Algorithms
7. Data Structures
8. Files & File Formats
9. Testing
10. Known Shortcomings & Flaws
11. 3 Best & 3 Worst Aspects
12. Changes to the Original Plan
13. Realized Order and Schedule
14. Literature References & Links
15. Attachments

## Personal Information

Project's & Documentation's Author/Creator: Ali Amaan

Student Number: 906955

Course's Name: CS-A1121 Programming Y2

Study Programme: BSc. Digital Systems & Design

Study Year: 2021-22 (2<sup>nd</sup> Year)

Date: 6th May 2022

## General Description

This project plan is to create a reservation system for an imaginary hotel which in this case is set by the name of '**Y2 Hotel Reservation System**' whose logo is given in **Figure 1**.



*Figure 1: Y2 Hotel Reservation System's Logo*

The system contains information on the scheduling of room reservations' data, customers' data, and the related hotel resources/services reserved. A separate system for an administrator user has also been implemented that can view/delete the any of the hotel's users' or reservations' data.

The related resources/services, termed in the documentation as 'add-ons', includes (daily) meal, pick & drop (to and from the hotel), car (rental), grooming, and wellbeing services.

Reservations can be displayed for single day or multi-day reservations, as specified by the user. The program maintains a record of the hotel reservations on a (.csv format) database file and displays them on a calendar according to their reservations. Error handling has been implemented to take care of out-of-bound bookings, resolve reservation conflicts, and user registration & login system. A record of the users is also maintained on a (.csv format) database file.

A tentative list of unique aspects, above and beyond the specified conditions for the 'Hard' difficulty level that have been incorporated as features into the program are as follows:

- Functionality to add/remove add-ons on top of the hotel's room reservations system.
- Point-based customer loyalty system with applicable discounts as per the loyalty level.
- Separate Administrator portal that acts as a super-user to maintain user & reservation data.
- Also added a touch of personal branding with displaying a custom designed 'Y2 Hotel Reservation System' logo on the main reservation window.

This project is made to well accomplish the 'Hard' version of implementation, which includes the following requirements that have all been fulfilled:

- The program prevents concurrent reservations and 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 save information on customers (e.g. name, address, phone number and e-mail) and connect it to a reservation.
- The reservations don't disappear when the program is closed, meaning they are 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.
- Made the user interface as usable, clear, and informative as possible.
- Additional services can be attached to reservations ((daily) meal, pick & drop (to and from the hotel), car (rental), grooming, and wellbeing services).
- 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.
- Added other useful features, for example an admin portal.
- Unittests implemented for at least part of the program.

The project was originally planned to accomplish the 'Hard' level of difficulty. Now, it is implemented in a way to accomplish above and beyond the 'Hard' level with extra features, quality code and human-centred GUI design.

## Instructions for the User

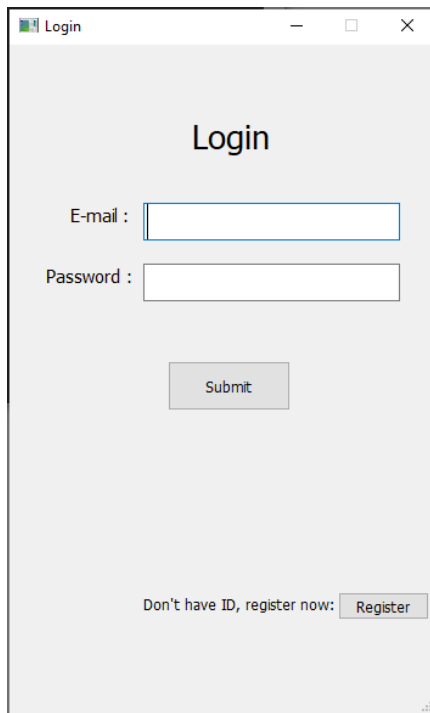
For executing this programming project, it is required that Python's version 3 is installed (preferably 3.9). The most reliable information source for Python is the page: <https://docs.python.org/3>

For inspecting the code and the programming files, it is recommended to use IntelliJ's PyCharm IDE given at the link: <https://www.jetbrains.com/pycharm/>

It is recommended that the OS's screen resolution is kept fixed at 1920x1080 pixels at 100% zoom/scale level for best performance and GUI view.

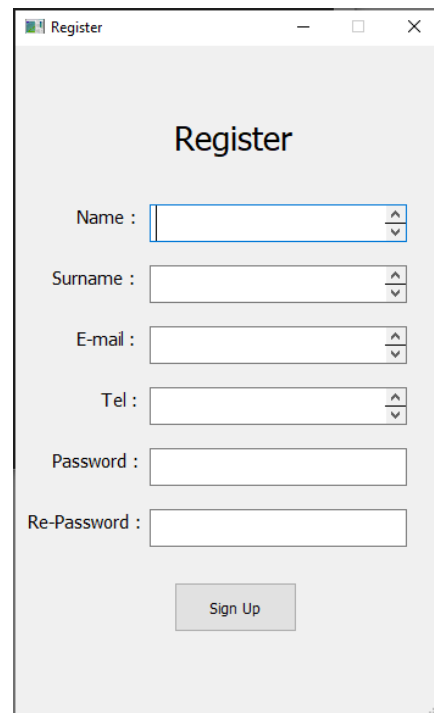
The program can be launched by simply clicking on the 'LoginMain.py' file.

From there a login/register window opens which can be used to either login by entering the email and password or register as a new user. The login window is shown in **Figure 2** and the registration window is shown in **Figure 3**.



The Login window has a title bar with 'Login' and standard window controls. The main area is light gray. At the top center is the title 'Login'. Below it are two input fields: 'E-mail : ' followed by a text box, and 'Password : ' followed by a text box. Below the password field is a 'Submit' button. At the bottom, there is a link 'Don't have ID, register now:' followed by a 'Register' button.

Figure 2: Log In Window (Program Start)



The Register window has a title bar with 'Register' and standard window controls. The main area is light gray. At the top center is the title 'Register'. Below it are six input fields: 'Name : ' (with a dropdown arrow), 'Surname : ' (with a dropdown arrow), 'E-mail : ' (with a dropdown arrow), 'Tel : ' (with a dropdown arrow), 'Password : ' (text box), and 'Re-Password : ' (text box). Below the Re-Password field is a 'Sign Up' button.

Figure 3: User Registration Window

As an example, the program already has 2 stored users and 1 admin with the following credentials which can be used to login or the user can register a new user. The credentials:

User Type	Email	Password
Normal User	alex@gmail.com	1234
Normal User	John.doe@gmail.com	Test
Administrator	admin	1234

For an administrator, login by simply entering their username ('admin') in place of the 'E-mail' and password ('1234').

After successfully logging in, the reservation window is opened through which a reservation can be made. **Figure 4** displays the different possible options the user can interact with. Moreover, the administrator dashboard is also displayed below in **Figure 5** alongside all its possibilities of interaction.

The program can be ultimately closed by logging out and then simply closing the 'Log In' window using the red close button on the top right of the window.

\*'Booking List' tab is to show a list of the past bookings made by this user. It can be used to delete bookings as well.

\*'About' tab is to display the user's stored information, such as the name, email, phone, loyalty level and points, etc.

\*to 'Log Out', go to the 'About' tab and click the 'Log Out' button.

\*select room from the dropdown. If that room is reserved on certain days, those dates on the calendar will be highlighted red.

\*select 'Check In' date by typing, using the buttoned arrows, or by selecting a day on the calendar.

\*select 'Check Out' date by typing or using the buttoned arrows.

\*clicking the 'Book Now!' button will also alert of any errors in the booking through a pop-up message box.

\*select the number of guests (Adults and Children) by typing or using the buttoned arrows.

\*select 'Additional Services', if any, through the checkboxes. Hover over the services to know more about them and their pricing.

\*click the 'Book Now!' button once everything is done to display price and confirm or cancel booking.

Figure 4: Reservation Window (Main Program Window)

\*'User\_data' tab is to view the database list of all the users registered in the system. It can be used to delete any of the user using their ID.

\*'Reservation\_data' tab is to view the database list of all the reservations made by the users. It can be used to delete any of the reservations using their ID.

ID	Room_number	User_ID	Check In	Check Out	Adults	Children	Services	Price
1 4	311	3	Tue Apr 26 2022	Wed Apr 27 2022	1	0	Meal Pick and Drop Car Grooming Wellbeing	93
2 6	202	3	Sun May 1 2022	Tue May 10 2022	1	0	Meal Pick and Drop Grooming Wellbeing	381.65
3 7	401	3	Wed May 11 2022	Fri May 20 2022	1	0	Pick and Drop Car Grooming	476.85
4 8	311	3	Sun May 1 2022	Wed May 4 2022	1	0	Pick and Drop Grooming Wellbeing	124.1

\*the data can be sorted according to their ID, which, in fact, is according to the order they were entered into the system.

\*the admin can log out of the Admin Dashboard by clicking the 'Log Out' button.

Figure 5: Admin Dashboard Window

## External Libraries

The following external libraries are used in the project other than the **PyQt5** library. The libraries:

- **datetime**: allows us to handle date and time objects within the program.
- **csv**: allows us to retrieve and store reservation and user data in .csv format database files.

## Structure of the Program

The main function of this program is given inside the 'LogInMain.py' file. The program always starts with the Log In window and then proceeds from there to the rest of the windows as per the interaction of the user.

The program uses four classes for the different user interfaces (login window, user registration window, reservation window, administrator window) displayed through inherited methods of PyQt5 library as per the definition in the class definitions code.

The program uses two classes for handling users' and reservations' data. These classes define the information stored for each user or reservation.

This approach of separately defining each window's class and different classes of users and reservations aided in several aspects of the code, in terms of understanding, clarity, navigation, debugging, and readability. A more specialised vocabulary for such an approach is the term 'modular approach' in which a larger more complex problem is divided into smaller more simpler chunks.

Alternative approaches include not separating the different classes and having just a single window for all the functionalities; however, it would have added to the complexity of coding as well as possibly clustered the code and the user interface. Additionally, had the graphical user interface not been implemented, the text-based user interface could have been coded in only the simple data handling classes with looped scripts for the users.

A UML diagram is shown below in **Figure 6** displaying the structure of how the program is implemented.

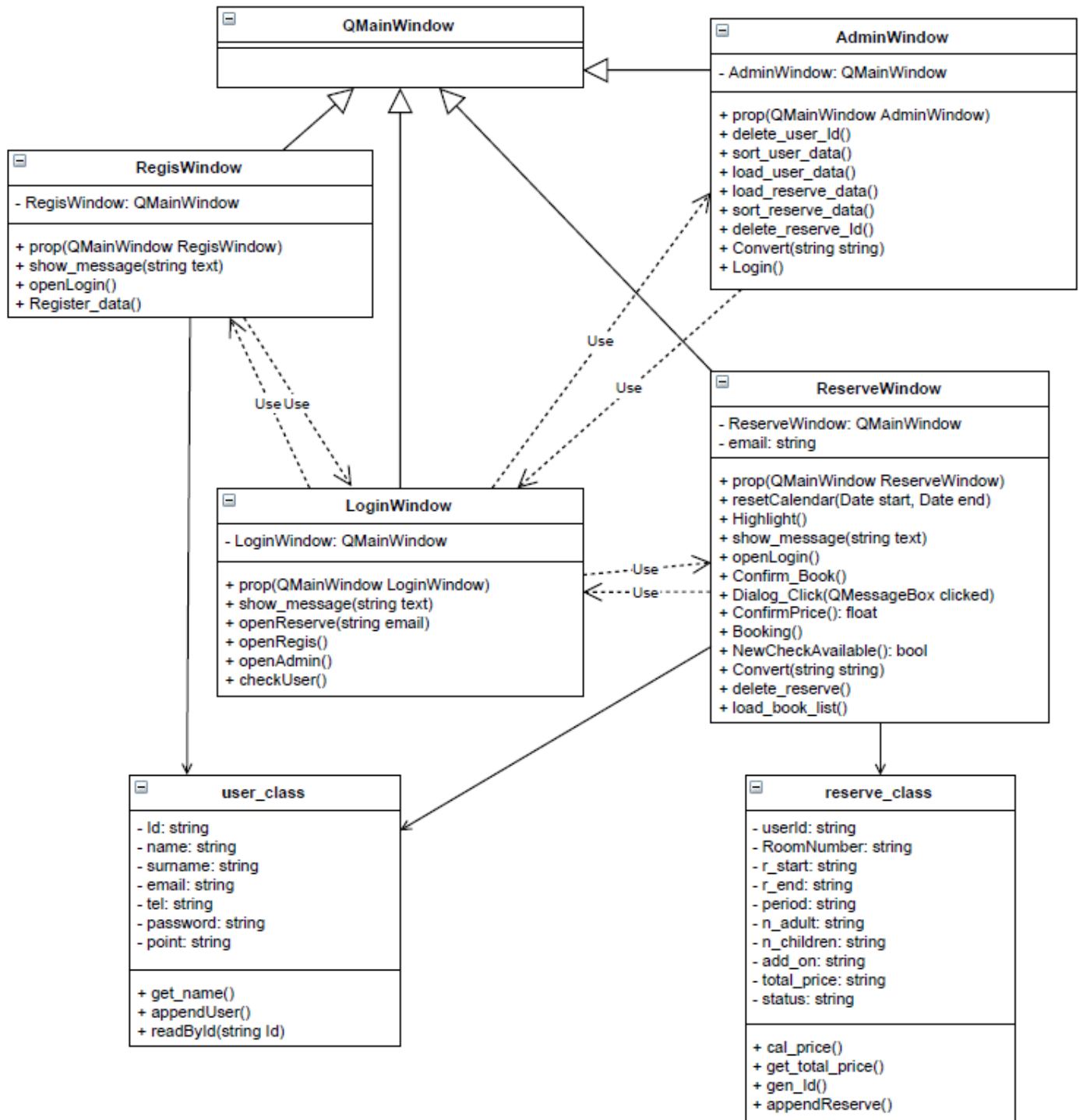


Figure 6: Program's UML Diagram

## 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. For registering a new user, the registration option can be used, upon whose successful registration, the user will be prompted to log in using their registered credentials, i.e., a combination of an email and a password.

Once logged in, there will be a reservation window showing the calendar view as well as the option to select the intended room and services for the reservation. The calendar also



shows already booked days of the specific rooms highlighted in red. Similarly, the user can also view their past reservations and their personal information using this window until they log out. This will take them back to the log in window

Similarly, if the admin has logged in, the respective options for it will be displayed showing the list of users and reservations that the admin can view or delete. When the admin logs out, the log in window appears again.

To fully close the program, simply close the program window using the red buttoned 'X' sign from top right side of the window.

Important algorithms implemented as part of this project include resolution of reservation conflicts, sorting of data in display tables, calculation of pricing, implementing the point-based loyalty system, and showing of prompts in case of errors, confirmation requests, and information alerts.

In case of reservation conflicts, when a user tries to book a certain room on date's it is already booked, the reservation will not register and a message prompt showing the booked schedule of the room will be displayed. A user can only book a certain room from the exact date a reservation is being ended at, e.g., if a room reservation is set to end at the check out date of 3<sup>rd</sup> of May 2022, then a newer reservation can be made by anyone with the check in date set at the same date, 3<sup>rd</sup> of May 2022. In this case, however, a reservation will not succeed if the newer reservation's check in date is before the previous reservation's check out date. Additionally, the check in date and check out date can never be set identical because the shortest booking that can be done in a real life hotel is also of checking in at a day and checking out only at the day after.

In case of sorting the data in tables, for displaying in user's past reservations or in the admin view, the data is sorted and displayed using the reservation ID's and ID's are regular integers in the order the reservations are made in or users are created in.

Calculation of pricing is done as per the following tabled pricing matrix.

Description	Pricing Type	Price (\$)
Room Rent	Per Day	40
Add-on: Meal	Per Day	7
Add-on: Pick & Drop	One Time	12
Add-on: Rent-a-Car	Per Day	20
Add-on: Grooming	One Time	9
Add-on: Wellbeing	One Time	5

Every pricing type of 'per day' is summed into the final price by multiplying the duration of stay with its respective price whereas every pricing type of 'one time' is only summed once into the final price.

An example booking can be a 2 day stay with add-on services of car rental, grooming, and meals. Its final pricing would then be =  $(40 * 2) + (20 * 2) + (9) + (7 * 2) = \$143.00$ .

The loyalty system is based on points earned through making reservations at the hotel and with each loyalty level, a discount on the final price of the reservation is applied. It's discount matrix is tabled below. Note that a reservation price of \$1 is equal to 1 point earned.

Loyalty Level	Points Interval	Discount Applicable (%)
Basic	0-100	0
Bronze	101-250	5
Silver	251-500	10
Gold	500 & above	15

Continuing the previous example, if this user had a silver loyalty level with 315 points before making the reservation. The final price of the user's current reservation would now be =  $143 * (1 - 0.10) = \$128.70$ . If the user confirms this reservation, the updated points of the user would now be =  $315 + 129 = 444$ .

Every time a user tries to make a reservation, they will be showed a prompt displaying the price breakdown and asking to proceed with confirmation of the reservation.

## Data Structures

Other than Python's predefined data structures that are planned to be used, as per Figure 6, two custom classes are defined in the program. They are as follows:

- **user\_class**: The class for defining, storing, retrieving necessary information about a 'user' of the hotel reservation system. The structure includes user's ID, name, surname, email, telephone number, password, and loyalty points.
- **reserve\_class**: The class for defining reservations made by the user(s). This includes all the booked services and the duration of the reservation alongside other necessary information. The structure includes reservation's user's ID, room number, check-in date, check-out date, duration/period, no. of adults, no. of children, additional services, total price, and status.

This list of data structure is implemented in the context presented in Figure 6. Both classes can be found in the '**Classes**' directory.

## Files

Files are stored in .csv format, where features of a class instance are separated by commas and class instances are separated by newline character(s) between them. They can be accessed, viewed, and edited even without the hotel reservation plan using Microsoft Excel.

Two files that are used in this project are '**reservation\_data.csv**' and '**user\_data.csv**' for storing reservation and user data, respectively, according to their class definitions. They both can be found in the '**DataCSV**' directory.

The 'Static' directory stores the Y2\_Logo.png file to display the logo on the main reservation window.

## Testing

The program was tested using partial Unittests, black box, and white box testing.

According to the plan, Unittests were implemented for methods of the custom user and reservation classes. Unittests can be found in the Unittest.py file.

Other than this the program has been rigorously tested with many extreme/borderline inputs and parameter values as well as straight correct or erroneous values. Peers were asked to use the reservation system and evaluate important aspects of the reservation system and the usability of the GUI based selection windows. This system was minorly edited iteratively in response to the feedback received to improve the program towards a human-centred design. The program was made to be used and tested by both, individuals who could see how the code was implemented (white-box) and those who could not (black-box).

It is pertinent to note that the testers skills in python could vary and hence the feedback could largely be entirely relative to the tester's understanding of coding and/or program logic. All the testers were students with relatively similar or basic programming skills.

Additionally, I also used 'pytest' to implement further tests for the PyQt5 GUI parts of the code, above and beyond the requirement. The documentation of the pytest-qt can be found at: <https://pytest-qt.readthedocs.io/en/latest/intro.html>

These tests, using 'pytest' have been implemented in the '**Test**' directory. To run these tests kindly follow the following steps.

1. As a pre-requisite, kindly install 'pytest-qt', if it has not been installed already. It can be installed by running the following command on a python terminal:

```
pip install pytest-qt
```

2. After installation, change directory to the '**Test**' directory of this project. It can be done using the '`cd`' command on a python terminal.
3. Then, simply run the test python files, one by one using the following command on a python terminal:

```
pytest <filename>
```

As an example, we can test the 'ReserveWindow' from the ReserveMain.py file using the predefined '**reserve\_test.py**' using the following command on a python terminal:

```
pytest reserve_test.py
```

Right now, all tests are working correctly and therefore indicate the correct working of the program.

## Known Shortcomings and Flaws

Some areas of weakness or potential improvements of the program are listed below:

- ability to change window sizes and support various other screen resolutions as well. It can be implemented by using methods of the PyQt5 library.

- more options of sorting the various fields of tables where reservation or user data are displayed could be present. Or a functionality could be added to just sort the rows just by pressing the table's header cells. Requires further in-depth coding using the table widget.
- ability to edit user's personal information by user themselves could be added. This could be coded in the 'About' tab with text boxes where one could edit the information and then using a save button, save it.
- ability to select entire time interval of reservation from the Calendar view (widget) in the reservation window instead of just selecting one date only can be implemented. Requires further in-depth coding using the calendar widget.
- alert prompts on progressing through the loyalty levels can be displayed. It could be added using a simple message box prompt.
- ability for the administrator to edit the user and reservations data can be added. This could be coded where one could edit the information directly in the table view and then using a save button, save it. Would require effort with the table widget.
- more validation checks on reservation inputs by the user can be added, e.g., maximum value of adults or children allowed, maximum duration a room can be booked for, etc.
- more stricter text validation for registration of a user can be added, e.g., setting minimum password length to be 8 alphanumeric characters, etc.
- ability for the user to delete reservation specifically, e.g., by a reservation ID instead of current implementation where upon entering the room number the latest respective booking gets deleted.
- addition of tooltips upon hovering the mouse over certain or all buttons could improve understandability., just like it is already implemented on the 'Additional Services'.
- option to return to the log in window without signing up can be an option that could be added by having a 'cancel' registration button on the registration window.
- ability to add/delete admin user or change admin's information and password can be coded.
- ability to change password when the user forgot their password could have been implemented.
- ability to individually increase/decrease prices of specific rooms in case of peak season, etc.
- ability for the admin to change charges and prices of rooms and services could have enhanced admin control.

### 3 Best & 3 Worst Areas

Overall, the 3 best aspects of this implementation of the programming project are:

- **Efficiently Programmed:** This program is coded in such a way that does not require running of endless infinite loop cycles which is an alternative implementation technique. Hence, it has a relatively improved time complexity. The program is also

efficient in terms of size in a way that the total size of the programming project is even less than 200 KB which is relatively quiet less. Contributing to this is conscious design choices of .csv files, compressed logo design, etc.

- **Usability & Re-usability:** the modular approach of programming using class structures allows supporting of object-oriented programming and its inherent benefits in term of usability and reusability of the code. This conscious design choice has many pros. The modular GUI implementation allows for transitioning back-and-forth between different windows quite smooth and according to functionality which makes the window's less cluttered and more function specific instead of a single window program implementation technique. Additionally, separate classes structure of user and reservation data allow for easier future extensions, such as added reservation features, additional user information, etc. Moreover, this reservation system can also be used across several different industry segments other than hotels with little cosmetic modifications (changing option display labels, variable names, etc.) required, e.g., in hospitals, schools, service-based industries, etc. Additionally, the project is well commented out as well which makes understanding the project very easy for any other programmer as well.
- **Human-Centred GUI Design:** the multi-window and multi-tab design of the program is intentionally kept after repeatedly getting feedback from peers and then incorporating them into the program's code. This is the reason a single window programming solution was not implemented which would have cluttered all the information at once in front of the user making the program largely dysfunctional.

The 3 worst aspects of this implementation of the programming project are:

- **Security Deficiencies:** the highly limited validation checks implemented at the registration window allows for very vulnerable security of users of this program. Additionally, no hashing/cyphering the passwords while storing in the .csv database files allows for another security loophole.
- **Reservation Validations:** the validation checks in place for making reservations in the reserve window are also highly limited. This allows for hypothetically infinitely long durations of reservations as well as unlimited number of guests (adults and children) in a single room booking. Moreover, dates validation is also weak in the sense that no concept of today's date is incorporated which allows for bookings and cancellations in the past as well.
- **Sorting & Deletion Functions:** the inability to select a specific reservation for a single room for deletion by the user in the 'Booking List' tab makes cancelling a specific reservation only according to the order they were booked in. When in fact it needs to be according to some ID and/or selected by the user. Additionally, sorting by clicking the table widget's header is another thing that is another not implemented aspect that is not the best of the program.

## Changes to the Original Plan

Majorly the programming was implemented according to the original plan except for changes to some methods/classes and the addition of objects due to the implementation of the GUI.

Moreover, for rapid implementation, I also implemented an earlier version of the GUI using QT Designer, however, all work from it was later removed and implemented through PyQt5 library, which also delayed the project a bit around the second checkpoint.

An additional difference was also in realising the GUI of the system which was previously thought to be easily implemented without using any clear UI/UX design mock-ups. Contrary to that, mock-ups of the intended windows and designs had to be drawn using pen and paper. Similarly, design inspirations also were needed to be taken from online materials. This all was an aspect that was very weakly or not at all planned earlier on.

Moreover, another additional aspect added later into the program were tests for the PyQt parts of the code using pytest-qt tests on top of the few unittests.

## Realized Order and Schedule

The major difference from the plan was in terms of the timeline which could not be accurately realized due to external reasons (excessive course load). This had caused a delay in meeting the checkpoint 1's targets as per the plan, however, at checkpoint 2, the planned targets were completed, albeit with a bit of delay due to forbidden use of QT Designer which was soon rectified. Afterwards, the focus remained on debugging and enhancements.

Here, the intended checkpoint 1 refers to 25<sup>th</sup> of March 2022 and checkpoint 2 refers to 15<sup>th</sup> of April 2022. Unintentionally, the realized checkpoint 1 then happened at 31<sup>st</sup> of March 2022 and checkpoint 2 happened at 28<sup>th</sup> of April 2022.

## Assessment of the Final Result

All in all, I am very content with the progress and result of my Y2 programming project. It was a challenging journey to curate this final project result, as is any programming project, which is part of the learning process. For me, there were many learnings in this project, such as realising the significance of the project plan, design mock ups, iterating structure of the program to suit the efficiency of the program's functioning, implementing tests, etc. I would also hit roadblocks in the implementation, such as during the time I had to implement the calendar highlight function for reserved days, however, the best remedy that worked for me was taking a break and thinking afresh. Additionally, programming continuously when a focus gets maintained also worked really well for me. In retrospect, this project was a major advantage from a coding standpoint as my final programming project result is much better organized, coded, and designed than my initial version and initial mind map.

As an assessment of the programming project, I worked and implemented it to fulfil all and above the specified 'Hard' level of the implementation. On top of the many hours spent to accomplish Hard level of implementation, I also went above and beyond to add extra features to the project, a list of which is mentioned in the General Description and detailed in the sections that follow. Much effort was also put into crafting this document to reflect the project in its true light, hence, it was only after carefully and critically assessment of the programming that this document was curated with all the pros and cons listed in this. The flaws and shortcomings of the project are majorly trivial

enhancements and not major functionality bugs. The reason for those is mainly the time constraint I faced this semester because of several reasons: this was the first semester I had moved to Finland from my home country of Pakistan, hence, had to carry out major moving tasks; had to take up around 40 credits (8 courses) this semester which added to the study load; etc. Moreover, I had already went very overboard with the recommended hours to be put in this 5 ECTS course and hence I think it the effort is justified.

I also think that there could be minor ways in which the project's coding or structure could be made better, however, I have not come across an implementation technique majorly different from mine that is better in functionality. Hence, I have little idea on how the structure, algorithm, or code can be drastically improved further. Therefore, I would love to be informed if there are ways in which this application could be implemented.

I believe the program well goes beyond and above the scope and requirements of the application and with minor tweaks, correcting the shortcomings, and a few other enhancements, can even be deployed to test as a product in a more real-life scenario as well. Overall, this project helped me put a lot of my python knowledge into practice and good use to summarize all my learnings as part of Y2 and all online self-learning during this course.

## References

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

- <https://doc.qt.io/qtforpython/>
- <https://docs.python.org/3/library/datetime.html>
- <https://docs.python.org/3/library/csv.html>
- <https://pytest-qt.readthedocs.io/en/latest/intro.html>

## Attachments

Login

E-mail : alex@gmail.com

Password : ●●●●

Submit

Don't have ID, register now: Register

Reservation System

User: Alex Xander

Y2 HOTEL  
RESERVATION SYSTEM

Reserve Booking List About

May, 2022

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

311

Check In : 6/5/2022

Check Out : 7/5/2022

Adults 0

Children 0

Additional Services

☐ Meal

☐ Pick and Drop

☐ Car

☐ Grooming

☐ Wellbeing

Book Now!



User: Alex Xander



Reserve Booking List About

May, 2022

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

Confirmation of Reserva... X

PRICE BREAKDOWN  
 Duration (Days): 3  
 Room Charge: \$120  
 Meal: \$21  
 Pick & Drop: \$0  
 Rent a Car: \$0  
 Groom & Care: \$9  
 Wellbeing: \$0  
 Discount %: 15  
 Discount: \$22.5  
 Total Price: 127.5

OK Cancel

311

Check In : 17/5/2022

Check Out : 20/5/2022

Adults Children

3 0

## Additional Services

- ☒ Meal  
☐ Pick and Drop  
☐ Car  
☒ Grooming  
☐ Wellbeing

Book Now!

User: Alex Xander



Reserve Booking List About

	Room Number	Check In	Check Out	Services	Total Price
1	311	Tue Apr 26 2022	Wed Apr 27 2022	Meal Pick and Drop Car Grooming Wellbeing	93
2	202	Sun May 1 2022	Tue May 10 2022	Meal Pick and Drop Grooming Wellbeing	381.65
3	401	Wed May 11 2022	Fri May 20 2022	Pick and Drop Car Grooming	476.85
4	311	Sun May 1 2022	Wed May 4 2022	Pick and Drop Grooming Wellbeing	124.1

Cancel Reservation : Type Room Number

Select

User: Alex Xander



Reserve Booking List About

Name : Alex Xander  
E-mail : alex@gmail.com  
Tel. : 12345678  
Member Level : Gold (1076) points

Log Out

## Register

Name : John

Surname : Doe

E-mail : john.doe@gma

Tel : 358333111888

Password : ●●●●

Re-Password : ●●●●

Sign Up

## Login

E-mail : admin

Password : ●●●●

Submit

Don't have ID, register now:

Register

User\_data

Reservation\_data

	User_ID	Name	Surname	E-mail	Tel.	Password	Point
1	3	Alex	Xander	alex@gmail.com	12345678	1234	1076
2	4	John	Doe	john.doe@gmail.com	358333111888	test	0

Sort By Latest

Delete By User\_ID

Submit

Log out

User\_data

Reservation\_data

	ID	Room_number	User_ID	Check In	Check Out	Adults	Children	Services	Price
1	4	311	3	Tue Apr 26 2022	Wed Apr 27 2022	1	0	Meal Pick and Drop Car Grooming Wellbeing	93
2	6	202	3	Sun May 1 2022	Tue May 10 2022	1	0	Meal Pick and Drop Grooming Wellbeing	381.65
3	7	401	3	Wed May 11 2022	Fri May 20 2022	1	0	Pick and Drop Car Grooming	476.85
4	8	311	3	Sun May 1 2022	Wed May 4 2022	1	0	Pick and Drop Grooming Wellbeing	124.1

Sort By Latest

Delete By ID

Submit

Log out