POST-PROTOTYPE REFINEMENT OF DESIGN

FEEDBACK FROM THIRD PARTIES AND CHANGES REQUIRED TO DESIGN

Here I will outline the feedback given to me from my peers and where possible my end user.

# THE FEEDBACK

* Double booking is a major flaw in the current system as 2 or more people can book the same room at the same time.
* The user interface is not particularly appealing to customers and staff members.
* The system had no pictures of rooms. Therefore, customers do not know what they are paying for.
* The system did not have a particular description of the room. Therefore, customers do not know what they are paying for.
* The system was not particularly user-friendly as someone could easily change someone else’s details. This is because the customer ID is displayed on the customer page which is used to change details and view bookings.

# CHANGES REQUIRED

based on the feedback I will be implementing the following changes to the system.

* Double booking is a major flaw in the current system as 2 or more people can book the same room at the same time. This will lead to angry customers as other customers may be in that room with them. As a result, customers would request a refund and it will impact the brand image/ reputation of the hotel and this could lead to business failure. Therefore, I will implement this change.
* The user interface is not particularly appealing to customers. This is because this is a hotel booking system and if the system is not appealing to customers, customers may avoid using it. Furthermore, Aesthetics is one of the components of the design mix. As a result, customers may go to rival competitors within the same market with better aesthetics. Therefore, I will need to implement this change.
* The system had no pictures of rooms. This is because not having pictures impacts the aesthetics and the value of the item. As a result, customers do not know what they are paying for. Therefore, if the customer experience is below their expectations it can lead to a bad brand image/ reputation. As a result, the customer could further ask for a refund due to the Customer Protection Act. Therefore, I will need to implement it.

* The system did not have a particular description of the room. This is because not having a particular description of the room impacts the value of the item. As a result, customers do not know what they are paying for. Therefore, if the customer experience is below their expectations it can lead to a bad brand image/ reputation. As a result, the customer could further ask for a refund due to the Customer Protection Act. Therefore, I will need to implement it.
* The system was not particularly user-friendly as someone could easily change someone else’s details. This is because the customer ID is displayed on the customer page which is used to change details and view bookings. As a result, someone could change the username or password, and that person would lose their account as they may not know their new username and password. Therefore, I will need to implement this in the system.

**POST-PROTOTYPE INPUTS, OUTPUTS, AND USER INTERFACE DESIGN**

**Changes to form Booking.**

**A screenshot of a grid

Description automatically generated**

Here I added the “btnCheck” to help with the collision objective. I also added Label8 to help calculate the total price “lblPrice”.

**POST-PROTOTYPE PROCESING STAGE**

* **The system should be user-friendly. (PP)**

This will be in the post prototype because if customers don’t know how to use the system there will be staff members who will be trained to use the system. Therefore, I will not include this objective in the prototype but in the post prototype.

To make the system more user friendly I added different colour notation meaning different colours for each function. To make this easier to understand I will be using “frmAddRooms” as an examples.

Here are the prefixes used in “frmAddRoom”;

Textboxes = “txt”, button = “btn”, combox = “cbm” , checkbox = “rich textbox = “rtb” and data grid view = “dgv”.

A screenshot of a computer

Description automatically generated

I will display the items in orange as being labels that guide through a process. This is because the vibrant Orange will pop out of the screen making easier to distinguish normal labels and orange labels being important. Therefore, I will be displaying labels that are important in orange.

I will also display the items that you must fill in as a light orange. This is because it will make the system more aesthetically pleasing. Furthermore, it will also give the same sensation of importance as the vibrant orange.

Therefore, I will be using light orange to display items that need to be filled in.

I will display normal labels as a silvery grey tone. This is because those labels may not be as important as the orange. Therefore, customers can be able to distinguish them.

I will also be displaying buttons in a light grey tone with white borders. This is because it will distinguish between labels, textboxes, combo boxes etc.. Therefore, improving the customer experience.

Finally I will display picture boxes in a light red tone. This is because it will make it easier to distinguish them between other procedures. Therefore improving the overall experience of the user.

**Pseudocode** – There ist really a pseudocode for making the system “more user friendly” therefore I will not be including it.

* **There should be pictures of the room in the system. (PP)**

This will be in the post prototype because pictures are not a must-have when booking as you can book a room without seeing pictures. Therefore, I will not include this objective in the prototype but in the post-prototype.

Pictures play a huge role when purchasing. This is before it gives the customer a sensation of what they could expect. Rooms in this instance are the main selling point. Therefore, I must make sure I incorporate this in the project.

Here are the prefixes used in “frmAddRoom”;

Textboxes = “txt”, button = “btn”, combox = “cbm” , checkbox = “rich textbox = “rtb” and data grid view = “dgv”.

A screenshot of a computer

Description automatically generated

In this form , staff members can add pictures but before that the staff members should have added rooms in the system to work. I will be explaining 3 different scenarios for adding rooms.

The first situation is when creating a new room. Firstly, for this to happen, the staff members must input the data into the com boxes, textboxes, and rich text boxes. Afterward, the staff members would need to press the “btnAddRoom” button to add the room to the database. Once the room has been added, the staff member should press on the “pictureboxroom” picture box to add the image using the base64 method. Once that’s done, if the data is validated, the following message box should show (“You have successfully added the picture to the database”). Otherwise, this message box should show up (“Sorry, there was an issue when adding this picture”).

On the other hand, in the second situation, the staff member should have already added a room. The staff member should select the “index” on the left of each room, for the room the staff member wants to add the image to. After clicking on the “index” the details of the room selected should show up. Once the details show up, the staff member should click on the “picutureboxroom” picture box and select the image for the room using base64. Once that’s done, if the data is validated, the following message box should show (“You have successfully added the picture to the database”). Otherwise, this message box should show up (“Sorry, there was an issue when adding this picture”).

Finally, if the staff member decides he wants to update the room picture because of some adjustments to the room, he may do that by first the staff member should have already added a room. The staff member should select the “index” on the left of each room, for the room the staff member wants to add the image to. After clicking on the “index” the details of the room selected should show up. Once the details show up, the staff member should click on the “picutureboxroom” picture box and select the image for the room using base64. Once that’s done, if the data is validated, the following message box should show (“You have successfully added the picture to the database”). Otherwise, this message box should show up (“Sorry, there was an issue when adding this picture”). Lastly the staff member should press the “btnUpdate” button to update the details.

Once the picture has been added to the database. Customers may see the picture on the form where all the rooms are shown.

Here are the prefixes used in “frmRoom1”;

Label = “lbl”, rich textbox = “rtb”, list box = “lst”

A screenshot of a computer

Description automatically generated

Here, the customer may see the picture of the room in the “pictureboxroom” picture box but before that, the customer should select the room he wants to see the picture of, out of the rooms shown. This is shown in the “lstroom” listbox. As soon as the room is selected, the customer should see the details of the room the customer has selected along with its details.

**PSEUDOCODE OF DISPLAY IMAGE**

SQL ( Select RoomPicture

From RoomTbl

Where RoomNumberID = @LIST and RoomPicture = @Picture)

While Reading data

@LIST = lstRoom.Identifier

@Picture = PictureBoxRoom

End procedure

* **The system should display a brief description of the room with what the room comes with, for example, a coffee machine. (PP)**

This will be in the post-prototype because a description of the room is not necessary when you are booking a room. You can book a room without having a description of the room. Therefore, I will not include this objective in the prototype but in the post-prototype.

Having a description of the room highly increases the chances of purchasing a room. This is because there is more proof that the site/ program is not fake or a scam. Adding a description increases credibility and enhances the customer experience. Here are the prefixes used in “frmAddRoom”;

Textboxes = “txt”, button = “btn”, combox = “cbm”, checkbox = “rich textbox = “rtb” and data grid view = “dgv”.

A screenshot of a computer

Description automatically generated

Here, the staff member should first create a room to save the room content which I decided to represent as textboxes as I believe it will be an effective and easy-to-read option, especially at first glance. As for room description, I decided to go with text as that is what someone expects a description to look like. Then staff members should select all the checkboxes that the room contains and fill in the rest of the text boxes and com boxes and description. Afterward, the staff member should press the “btnAddRoom” button. If data is validated it should show the following message box (“You have successfully added the room”). Otherwise, it should show the following message box (“Sorry, please double check the values entered”).

To update the room description and content, the staff members should select the “index” on the left of each room, for the room the staff member wants to add the image to. After clicking on the “index” the details of the room selected should show up. Once the details show up, the staff member should click on the “btnUpdate” button to update the values.

Once the room is set up, for customers to see this data they should log in and head to the “frmRoom1” form. Here the customers can see all the details for the room the customer selects. Here are the prefixes used in “frmRoom1”;

Label = “lbl”, rich textbox = “rtb”, list box = “lst”

A screenshot of a computer

Description automatically generated

Here the customer should firstly select the room number. The room number can be selected using the “lstrooms” list. Once the customer selects the room number, it should show all the details of the room including the room description(lblDescription) and the room content on the checkboxes on the right.

**PSEUDOCODE OF DISPLAYING ROOM DECRIPTION**

SQL ( Select RoomDescription

From RoomTbl

Where RoomNumberID = @LIST)

While Reading data

@LIST = lstRoom.Identifier

End procedure

* **There should be no collision. (PP)**

This will be in the post-prototype because it is a complex algorithm. Furthermore, I am an A-level student with 2 other A levels to deal with so will not have a lot of time to implement complex features. In addition, it is just an addition to the program to make it more effective but does not affect the booking system itself. Therefore, I will not include this objective in the prototype but in the post-prototype.

The term “Collision” I will constantly refer to, in this case, it means that customers should not be able to book the same room at the same time. This is essential because the room that the first customer booked should be booked meaning other customers should not be able to book the same room during the time that the customer booked stays there for (“StartDate” to “EndDate”).

Here are all the prefixes used in “frmBooking”;

textboxes = “txt”, buttons = “btn”, datetimepicker = “dat”, datagridview = “dgv” and listboxes = “lst”. Below, there is the design for frmBooking;

**A screenshot of a grid

Description automatically generated**

To begin this process, first the customer should select the “StartDate” and “EndDate” in the datStart and datEnd respectively. This will be the start and end date of the stay. After selecting the dates, the customer should press “btnCheck”. This will check all the rooms and display all the rooms that are not booked within those dates in the “dgvRoom”. Furthermore the” cmbRoom” will also become restricted. Only booking the rooms that are available will show up. Finally, the customer can go back to usual booking.

If a new room is booked it will still work as all the data is stored in SQL to the database.

**PSEUDOCODE FOR DISPLAYING ROOMS AVAILABLE**

SQL (Select RoomNumberID

From ROOMTBL

Where RoomNumberID IS NOT

(Select RoomNumberID

From BookingTBL

Where StartDate => dateStart and EndDate <= dateEnd))