Design

Computer Science NEA

21HeffernaJN42

Contents

Decomposition

Your system needs to be broken into a series of sub problems; each one should generate a different method. You need to justify each one, linking each to the objectives you agreed at the end of the investigation, even if it feels very obvious. These will then be used in each section of the design so you will keep referring to them.

Login Page

Allows user to log into system by inputting a passcode through the use of an onscreen keypad (which will be a 4-digit passcode) and comparing with the passcodes stored in the database. This will prohibit any unauthorised users from gaining access and altering the system.

If no passcode is found when compared to the database, a message will appear disclosing that the passcode is incorrect. This will ensure that the passcode needs to correct in order to gain access to the system.

Each passcode will be linked to a userID (which is the primary key). This will allow the system to have different access levels as once a user has logged in, a flag containing the userID will be present, and therefore apply the corresponding restrictions.

Overall, introducing this into the system has completed certain objectives. For example, allowing user access levels has enabled the system to grant certain permissions to the specified users. This is specifically done, as mentioned before, through the use of storing the UserID of the user. In addition to this, the objective of allowing staff members is completed, as each staff member has a unique userID and passcode.

Main Window

The objective of this window is the be the frame of the system. This means that all of the pages of the system will appear through the widgets placed in the window. There is a taskbar that contains buttons which will help toggle between the main pages of the system.

Once a button is clicked in the taskbar, a function which will set the current index of the widget to the page corresponding to the correct button.

In addition to this, there is a log out button that will lock the system and will require a passcode to continue.

Linking to the Criteria and Objectives, the system fulfils the objective to be able to navigate throughout the system through the use of buttons. This is particularly present in the use of the taskbar, as it allows the system to efficiently cycle between pages with a click of a button.

Bookings Page

Using this set of subpages, the user will be able to create bookings for the customer. Information that will be disclosed in this process include Booking date, Party size, Booking Time, and Guest details e.g. contact number, name, etc. The buttons at the top of the widget, which allow you to toggle between pages, will only enable once the previous page has been completed. This will help avoid the user to skip over essential details.

- **Booking Date**

  In this page, a calendar will appear with the current date as the current selection. All of the dates before the current date will be disabled, which ensures that the staff member cannot accidentally book reservations for the past. Once an appropriate date is selected, a function will run which will allow the user to select the party size of the reservation and return the date selected by the user.

- **Party Size**

  In this page, a list widget will appear with the selection of 1 to 8 guests. However, if a reservation were to have a guest list that is greater than 8, there is a check box in the corner which the user will select to indicate there will be a custom number of guests. Under the check box, there will a spin box which will allow the user to change the number of guests by clicking the up and down button.

  Once the number of guests is selected, the button allowing the user to select the reservation time will be enabled. In addition to this, a function will return the correct number of guests by getting the result of the selected item on the list widget or the spin box, depending on whether the custom amount check box is selected or not.

- **Reservation Time**

  Similarly, to the party size selection screen, a list widget is used to obtain the value inputted by the user. This list widget will include values ranging from 12:00 to 18:00 with 30-minute intervals. These fixed times allow the system to maintain structure and ensures that customers are correctly spaced out.

  In addition to this, if the restaurant is fully booked at a certain time, the corresponding button will be disabled. The system will be able to detect whether the restaurant is fully booked by executing a query which will total the number of customers booked for a certain time period.

  Once a time slot has been selected, the system will allocate a table to the reservation by executing a query that will search for available tables for that select time. Once that table has been found, the tableID will be stored and used in further operations.

- **Table Selection**

  In this page, a collection of buttons is outputted, which represent the tables in the restaurant. This is possible as a function is run during initialisation, which loops the creation of push buttons in an ordered fashion. Once you have reached this page, a party size is guaranteed to have been obtained.

  When pressing a button, the capacity of the table, which is obtained from executing an SQL query, is compared with the party size. If the tables capacity is less than the party size, or if the table capacity – 3 is still greater than the party size, the table the user is trying to select will deselect itself. This is because is it not good to book for a table with a lower capacity than the number of guests and it is bad to book a table that is too big for the number of guests coming.

- **Guest Details**

  In this page, there is a collection of line edits which are used to input the customer's details. The details that will be entered include First name, Last name, and email address. To ensure the data inputted is valid, validation will be present. Examples of validation will include a format check on the email address as emails require a specific format e.g., @gmail.com. In addition to this, a presence check will be needed as all of the line edits are essential to identify each customer and to avoid redundancy.

Also present on the Guest Details page is a proceed button. Once all inputted data passes the corresponding validation checks, the system will begin to create the booking. The system will create the booking by executing a combination of SQL queries. These queries include inserting the customer details, reservation time, party size, tableID, and booking date.

In order to ensure all of the data is fit to be inserted into the database, validation is essential. As a result of this, the system must have consistent validation throughout the system, which will assure the consistent validation objective is met. In particular, all of the input fields will include validation to ensure all of the inputs are sensible and fit for the database.

Calendar Page

The calendar page is used to easily view all of the bookings present for a select date. This page consists of a calendar widget and a list widget. The calendar widget is used as an input field where the user will click on a certain date, and through the use of functions, the system will be able to retrieve this date and use it for further functions.

Once a date is selected, the reservations will be outputted in rows in the list widget. The reservations outputted will include customer name, reservation time, table number and number of guests. These details are attained through the use of queries where the system will collect all of the records with the corresponding booking date.

Overall, by implementing these features into my system, the objective to allow staff to search for bookings on a certain day has been met. I believe that this objective has been met as the user can view all of the reservations for the day, while also being able to view the reservation details, such as customer name, reservation time, etc.

Table Layout Page

In this page, a collection of buttons representing the tables are present. There is a total of 15 buttons, which all run the same function when clicked. Once a button is clicked, a signal is sent, and the table number is stored. After this, the widget will set its index to the table order page.

Table Order Page

Once a table has been selected, the table order page will open. This page will consist of 4 subpages that contain a variety of buttons. These subpages are Starters page, Mains page, Desserts page, Drinks page. During the initialisation of the program, a function is run that generates all the menu item buttons and places them into the corresponding subpage. This is done by executing a query that selects all the records in the select category, e.g., starters, drinks, etc. Once this query has been executed, all the item names will be stored in array. The system will then add the first item as a button and repeat for the length of the array, moving up an index every loop.

Once a button in the subpage is clicked, a function will run which will add the item name to an array called OrderItems. As more and more buttons are clicked, the array will continue to grow. This array will be used later to add the ordered items to the database.

At the bottom of the page, there is a collection of buttons, the functions of these buttons are:

- **Confirm Order** – Once this button is clicked, a function will run which will add the items in the array OrderItems to the order of the table. This is achieved by executing a query that will add to the table 'TableOrder', the corresponding TableID and OrderID, which were obtained in previous functions.
  After this query has been executed another query will be run to display the table's orders on a list widget. In order to complete this, a query is run to select the item name and price of the ordered items for the table. The use of inner join will be required as the system will first need to obtain all of the OrderID from TableOrder with the corresponding TableID, the system will then find the item names and prices of all the OrderIDs previously obtained. This query is also run automatically when a table button is pressed in order to automatically update the list widget to the correct orders.
- **Close Table** – Once this button is clicked, a function will run which will execute a certain SQL statement. This statement will change the active status of the SittingID, turning it into False. This means that the SittingID is no longer in use, and the group sitting at the has finished.
- **Back** – Once this button is clicked, the current index of the main widget will return to the index of the seat layout screen.

Management Page

Once the management button is clicked, several buttons will appear. These buttons will be used

To toggle through the submenus/subpages.

- **Edit Menu**
  - **Add Item –** Once this button has been clicked, the widget will be set to another index. This page will contain multiple line edits. These line edits will be used to input all the key data needed to add items to the menu. These items include item name, item category, and item price. Once the confirm button has been clicked, the data in the line edits will be validated to ensure that they follow the systems guidelines. If all of the line edits inputs pass validation, a query will be run to add all of these items to the table MenuItems. The OrderID of the item will be calculated by using the count() function in SQL, which displays the number of records in a table. The OrderID will be +1 of this value. Once this has all been done, the widget will then reset all line edits and return to the edit menu index page.
  - **Remove Item –** Once the remove item button has been clicked, the widget will be set to the corresponding widget page. This page will consist of a table widget and 2 buttons (Back and Remove). The table is used to display all of records in the Table MenuItems. The table is filled by executing an SQL query that selects all the item names, category, and prices. These records are then inputted into the table row by row, meaning 1 record takes up 1 row.
    When the remove button is pressed, a function is run which can get any field value of the selected record, in this case it is the item name. The returned value of the function is

stored as the variable 'name'. Another dialog is then opened as a confirmation window for the removal.

- **Removal Confirmation –** This window only contains a label and 2 buttons. The label asks the user for confirmation if they want to remove the selected item selected in the previous window.
  The two buttons are:
  Remove Item – When this button is clicked it confirms to the system that the user wants to remove this item. The system then uses OOP to run the function in another class in order to get the OrderID of the selected item. The system will then run a query that will delete the record with the corresponding OrderID. After this, the table is then updated to display the altered menu.
  Cancel – Once this button is clicked, the removal confirmation window is simply closed.

Inputs, outputs and processes.

For each of your sub problems you need to identify the inputs, outputs and processes.

Sub Problem Ref	Input	Process	Output
Booking Date	Click a date on calendar	Get the selected date using PyQt modules, then convert the date from PyQt form to PyDate. The conversion to PyDate allows python to manipulate (e.g. add days, months, etc.) the date that is selected.	Return variable 'date' in PyDate form. This variable will be passed into the createbooking() function, in order to obtain the reservation date when creating the booking.
Booking Date	Click a date on calendar	Execute command to enable the push button to select the party size	Push button party size is now enabled and able to press
Party Size	Party Size is selected by clicking the corresponding size in the List Widget	Execute command to enable the push button to allow the user to proceed and select the reservation time.	Push button Time is now enabled and able to press, which allows the user to proceed to the reservation time selection page.
Party Size	Check box for custom amount is selected/deselected	If the checkbox is set to a checked state, the system will enable the push button to proceed to the reservation time selection. If the check box is not checked and the user has not selected a party size in the List Widget, the system will	The Push Button used to proceed to the selection of the reservation time is either enabled or disabled based on the result of the processes stated.

		disable the push button to proceed to the reservation time selection.	
Reservation Time	Reservation time is selected by clicking the corresponding time/row in the List Widget	The index of the selected item is found from using .currentrow() feature in PyQt. The selected time is then obtained from getting the text in the index found previously. This can now be used in an array of functions.	Return the variable 'bookingtime'. This variable can now be passed to the function 'createbooking' in order to use the desired booking time of the customer.
Reservation Time	Reservation time is selected by clicking the corresponding time/row in the List Widget	Execute command to enable the Table Selection Push Button in order to allow user to select their desired tables.	Push Button in the taskbar is now enabled and able to press. The user can now click on this button and proceed to the table selection screen.
Table Selection	Select the Table Selection Push Button in the taskbar	Execute commands to loop the creation of check boxes, with the placement being altered each loop. When a check box's state is changed, the system will automatically run 2 separate functions.	The system will output a series of checkboxes in the form of tables. These will all be outputted in a specified structure.
Table Selection	Select the Table Selection Push Button in the taskbar	A function is run to clear all the check boxes and reset them all to an enabled state. Another function is run to find all of the tables that have been booked within 2 hours of the selected reservation time. These details are acquired through the use of a SQL query. Once these tableIDs have been returned, the system will loop though and disable all of the checkboxes corresponding with the unavailable tables.	If a table has been booked within 2 hours of the users desired booking time, the system will disable the checkbox, making it unable to select for the user. This is done to ensure there is no overbooking/double booking.
Table selection (Table size verification)	Select/Deselect any checkbox representing a table	A function, which is passed the table number of the selected checkbox, executes a SQL query which obtains the capacity of said table. This result is then compared to the party size that was selected before. If the party size is greater than the table capacity, the system will not	If the user selects a suitable table which passes verification, the checkbox will be selected. And if the table doesn't pass verification, it will instantly be deselected, making it unable for the user to select it.

		allow the table to be selected. If the party size is less than the table capacity - 3, the system will not allow the check box to be selected.	
Table Selection	Select any checkbox representing a table	Once the selected table has passed verification, a function will run which will execute the command to enable the push button to allow the user to proceed and enter the guest details.	The push button which will allow the user to proceed to enter the guest details can now be selected. This allows the user to finally add the guest's details such as name, contact number, etc.
Guest Details	The Push Button 'Proceed' is pressed	This function will execute an SQL query that selects the number of records in the Customer table. Once this number is found, the variable customerid will be stored as +1 of this number. This value will be returned.	The variable customerid is returned to function 'createbooking'. This allows the system to correctly assign the customerid to the reservation.
Guest Details	The Push Button 'Proceed' is pressed	This function will execute an SQL query that will obtain the number of records in the table Reservation. The variable reservationid will be stored as 1 more than this value.	The variable reservationid is returned to function 'createbooking'. This allows the system to correctly assign the reservationid to the reservation.
Guest Details	The Push Button 'Proceed' is pressed	A set of functions are run to obtain all of the reservation details selected in the previous pages. A series of SQL queries are then executed to insert the data obtained into the tables Customer, Reservation and TableReservation.	The functions 'getdate', 'getpartysize', 'gettime' are run. As a result of this, the function has obtained the values of the variables: date, partysize, bookingtime, and tables
Calendar Page	The Calendar button is selected in the taskbar	The current index of the widget will be set to the index of the calendar page.	The widget will now display the page containing the calendar to allow the user to view reservations.
Calendar Page	A date is selected in the Calendar Widget	A query is passed into the function. The headers in the widget are set to 'People', 'Table', 'Name', 'Time'. The query is then executed, and the results are then inserted into the table row by row.	The reservations that are booked for the date selected by the user will be displayed in the table. Each row will contain the number of people, table number(s), customer name, and reservation time.
Calendar Page	A date is clicked on the calendar	A function is run with a try statement. In the try statement, the date that is selected on the calendar is	The variable containing the query has been passed into the function which outputs the reservations into the table. This now allows the

		obtained and converted for QT format to PyDate format. This is done in order for SQL to interpret the date given. The query is then made with the selected date as a parameter. This query is then passed into the table fill function.	system to successfully output the reservations for the corresponding date.
Table Layout Page	The Seating button is selected in the taskbar	The current index of the widget will be set to the index of the seating page.	The widget will now display the page containing a series of pushbuttons to allow the user to make orders for tables.
Table Layout Page	The Seating button is selected in the taskbar	A function is run that will output a series of push buttons that represent the tables. A total of 15 tables are outputted. When any of the buttons are clicked, 2 different functions will run.	The system will output a series of push buttons which represent the tables. These can be selected in order to create orders for the tables.
Table Layout Page	A push button representing a table is clicked	The current index of the main window will be set to the order page for the tables.	The page will change to the order page for the Tables
Table Layout Page	A push button representing a table is clicked	The table number will be passed into the order page so that the TableID can be accessed from the other page.	A label in the corner of the screen will output the number of the table the user has selected. This is done to avoid any mistakes when ordering.
Table Order Page	Push button Exit is clicked	The current index of the main page will be set to the table layout page	The page will change to the previous page which displays the push buttons representing the tables.
Table Order Page	Push Button Main is clicked	The current index of the Menu widget will be set to the index of the Main menu page	The section of the page containing the menu will switch to the Main meals. This will allow the user to order the Main meals of the customers.
Table Order Page	Push Button Starters is clicked	The current index of the Menu widget will be set to the index of the Starter menu page	The section of the page containing the menu will switch to the Starters. This will allow the user to order the starters of the customers.
Table Order Page	Push Button Desserts is clicked	The current index of the Menu widget will be set to the index of the Dessert menu page	The section of the page containing the menu will switch to the Desserts. This will allow the user to order the desserts of the customers.
Table Order Page	Push Button Drinks is clicked	The current index of the Menu widget will be set to the index of the Drinks menu page	The section of the page containing the menu will switch to the Drinks. This will allow the user to order the drinks of the customers.
Table Order Page	Push button Confirm is clicked	An SQL query is executed to add the OrderID and TableID	Once the query has been complete, the screen will then return to the

		to the database. The current index of the widget will then be set to the seating page.	seating page, which displays all of the tables in the form of push buttons.
Table Order Page	A push button representing a table is clicked	A query that will output the Item name and price of all the orders made for the selected table. This data will then be displayed through the use of a list widget.	The orders that have been made previously to that table, will be displayed in a list widget. Each row will contain the item name, price, quantity, etc.
Table Order Page	Either Push Button Drinks, Desserts, Starters, Mains is clicked	A query is executed to get all of the menu item names in a select category (which is passed into when calling the function). This list of menu items is then returned.	Once the list of menu items has been obtained, the system will pass this array into another function, which will output a series of buttons which have each of the menu item names as the button name/text.
Management Page	Push button Edit Menu is clicked	The current index of the main window will be set to the index of the edit menu page	The main page will be set to the edit menu page. This will allow the user to make changes to the current Menu.
Edit Menu	Push button Add Item is clicked	The current index of the main window will be set to the index of the Add Item page	The main page will be set to the Add Item page. This page will allow the user to add food/beverages onto the menu.
Add Item	Push button confirm is clicked	The values entered in line edits are obtained. These values are then validated to ensure they follow the systems guidelines. If all values are valid, a query is run to add these values to the table Menu.	Once the system has obtained all of the Menu item details, and the OrderID of the item has been passed into the function, the system will execute an SQL Query to add this record into the database.
Add Item	Push button confirm is clicked	A query is run that obtains the number of records in the Menu table. The orderID will be stored as +1 of the value returned from the query.	The variable orderID will be passed into the function that will add the Menu Item into the database. This ensures the orderID is correct for the new Menu Item.
Add Item	Push Button Confirm is clicked	Once the query to add the menu item has been completed all of the line edits will be reset and the main widget will change its current index to the management page.	All of the input fields on the add menu page will be reset to its default values. The main page will then be set to the management page.
Add Item	Push Button Back is clicked	The current index of the main window will be set the index of Edit Menu page	The main page will be set to the Edit menu page
Edit Menu	Push button Remove Item is clicked	The current index of the main window will be set to the index of the Remove Item page	The main page will be set to the Remove Item page. This page will allow the user to remove menu items from the database

Edit Menu	Push Button Remove Item is clicked	Set table headers to ‘OrderID’, ‘ItemName’, ‘’Category’, ‘Price’ Execute SQL query to obtain all of these fields. Then fill the table with 1 record equalling 1 row.	All of the Items on the Menu will be outputted into the table. Each row will contain the Menu Items OrderID, Name, Category, And Price.
Remove Item	Push Button Remove is clicked	If the user has selected a record on the table, the system will get the item name of the selected record. It will then output a confirmation window	If a user has selected a record on the table and pressed confirm, a confirmation window will appear to ensure the user is making the correct decision. However, if the user has not selected a record in the table, a error message will appear
Removal Confirmation	Push Button Cancel is clicked	If the user clicks the Push Button Cancel, the system will run a command which will close the confirmation window that has appeared.	The Confirmation Window will close and the system will return to the remove menu item page
Removal Confirmation	Push Button Remove is clicked	The system will execute a function in another class which will get the OrderID of the record previously selected. It will then execute a query to remove this item from the Table MenuItems. The table will then be updated with the removed item.	The Confirmation Window will close and the system will return to the remove menu item page with the table containing the updated database changes

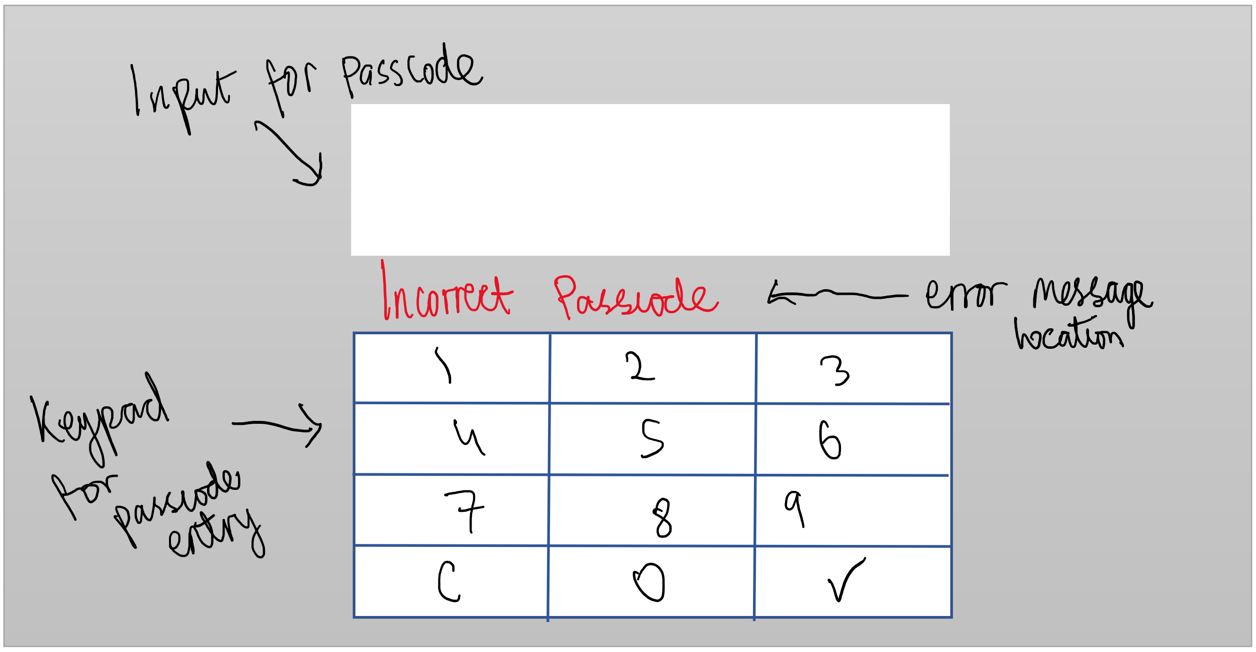
Joshua Heffernan
7865

The College of Richard Collyer
Centre: 65131

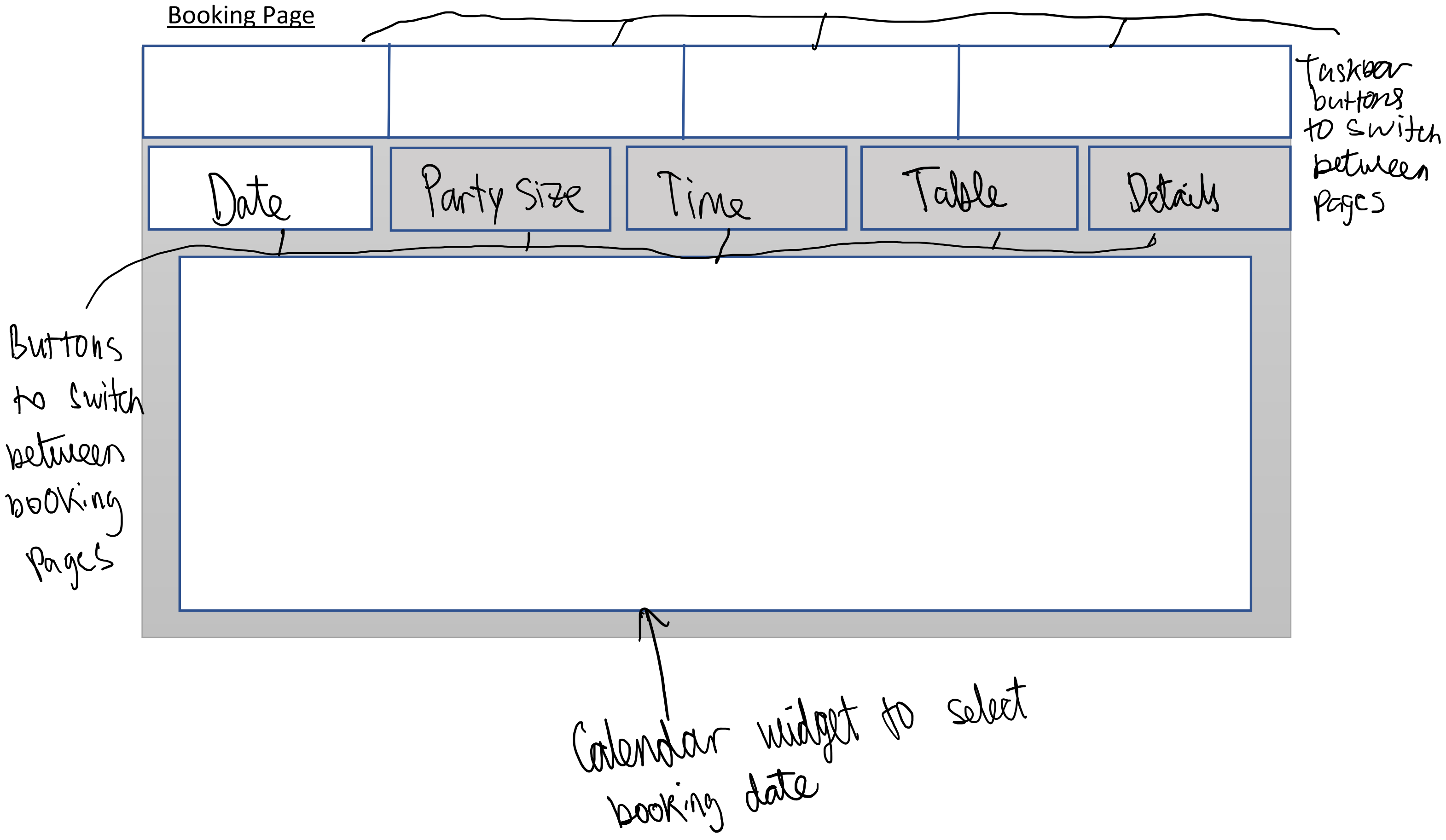
User Interface

Don't be deceived by the title, you need to design every form you will be creating, including error messages. These need to be fully annotated, you will be showing these to your peers, they should have enough detail to be able to pick up your idea and program it. You should link these with the objectives in the design or your decomposition tasks.

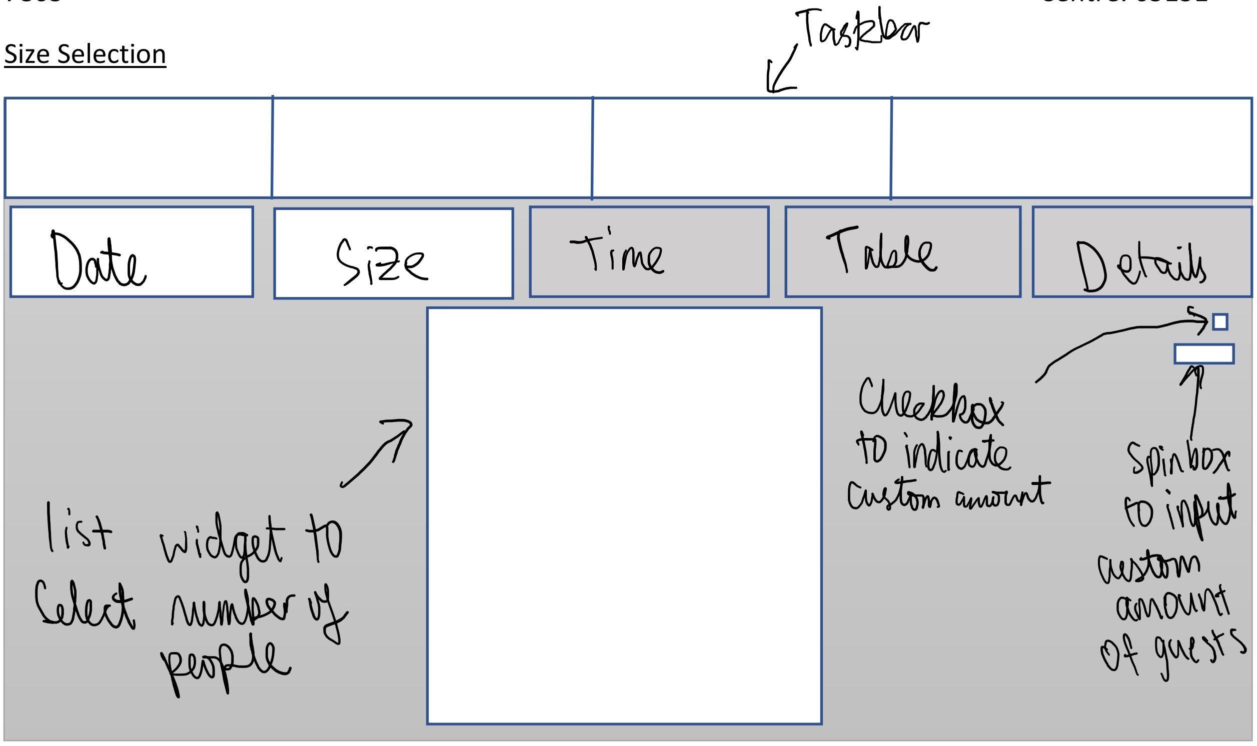
Login Page



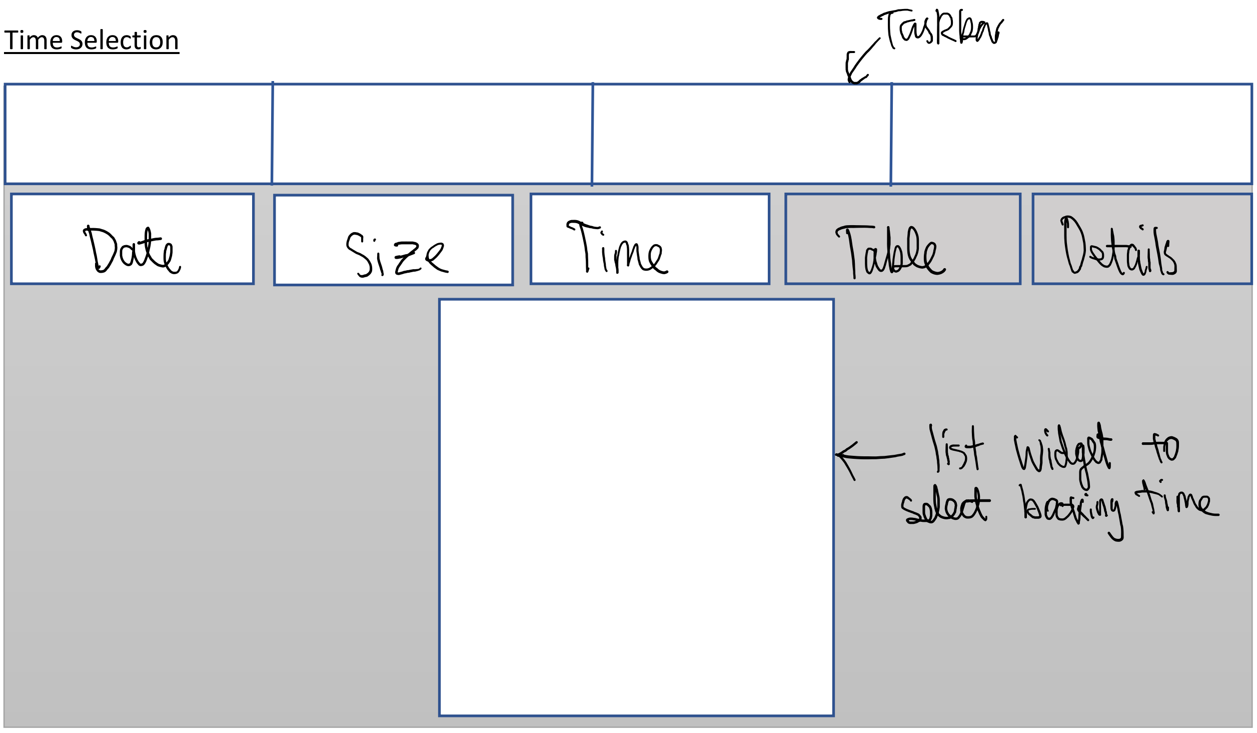
Booking Page


Taskbar buttons to switch between pages
Date
Party Size
Time
Table
Details
Buttons to switch between booking pages
Calendar widget to select booking date

Size Selection


Taskbar
Date
Size
Time
Table
Details
list widget to select number of people
Checkbox to indicate custom amount
Spinbox to input custom amount of guests

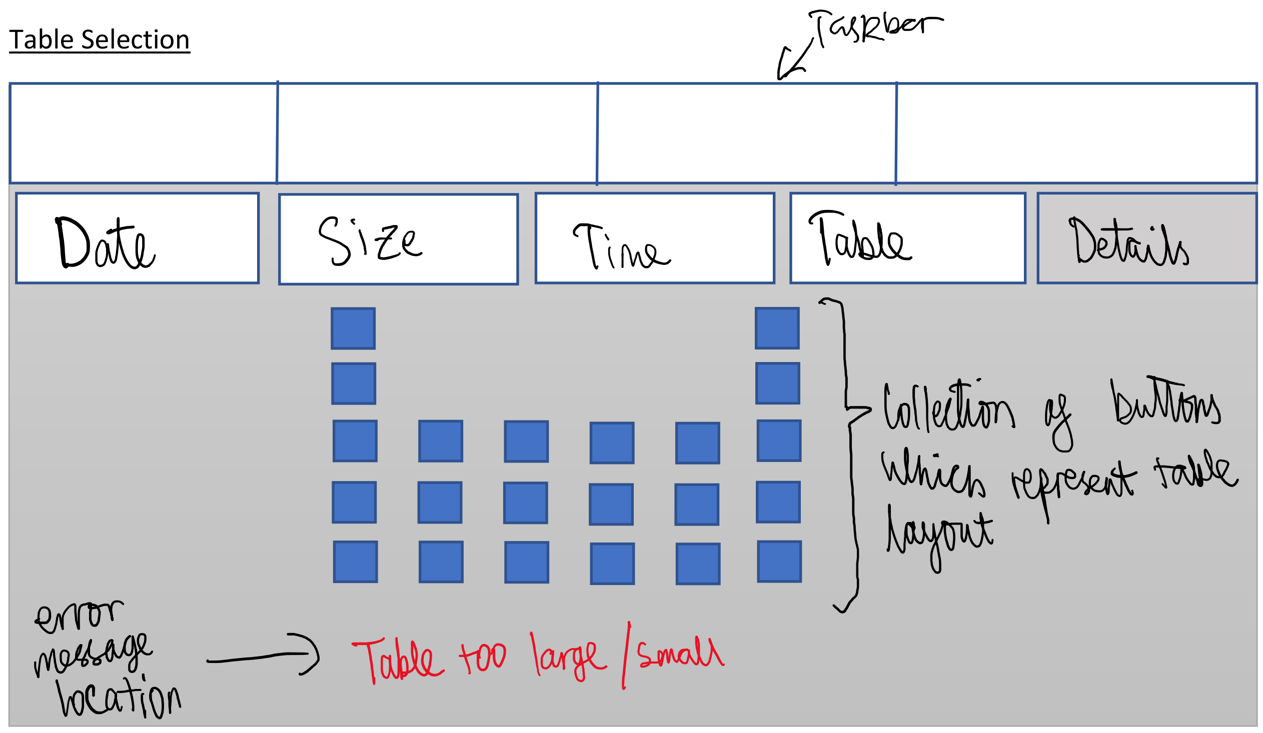
Time Selection



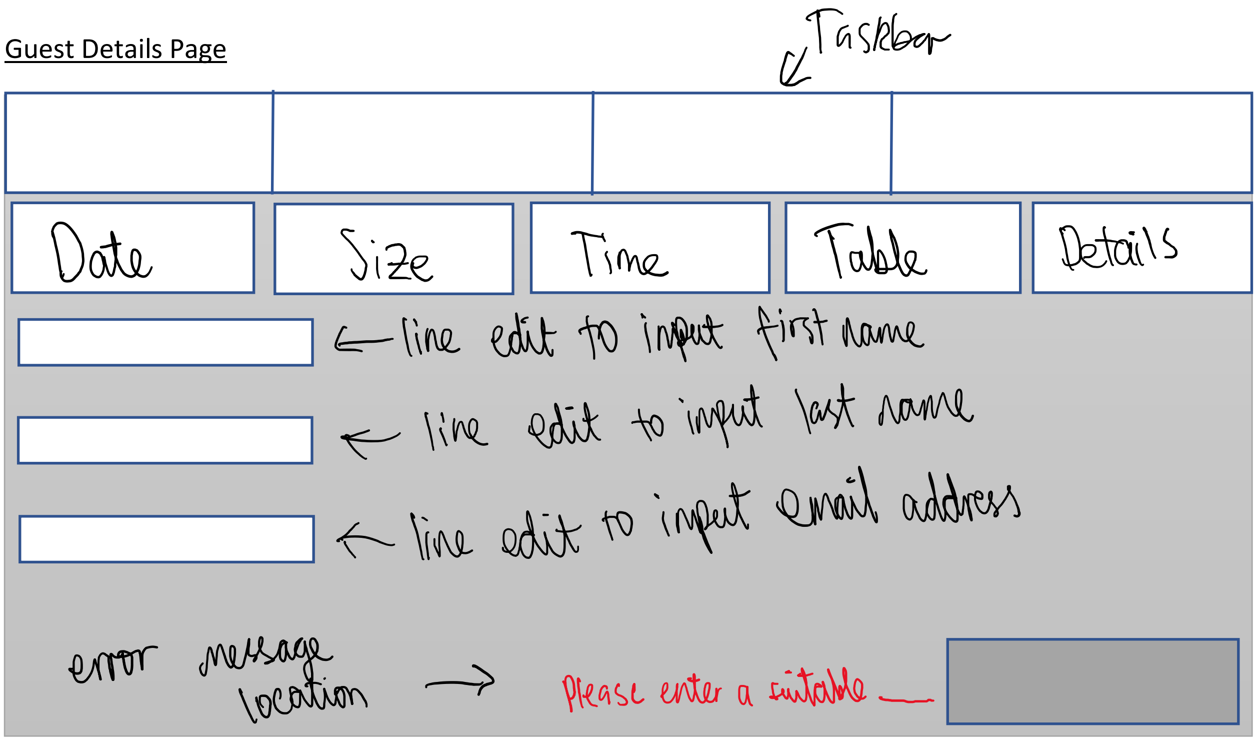
Joshua Heffernan
7865

The College of Richard Collyer
Centre: 65131

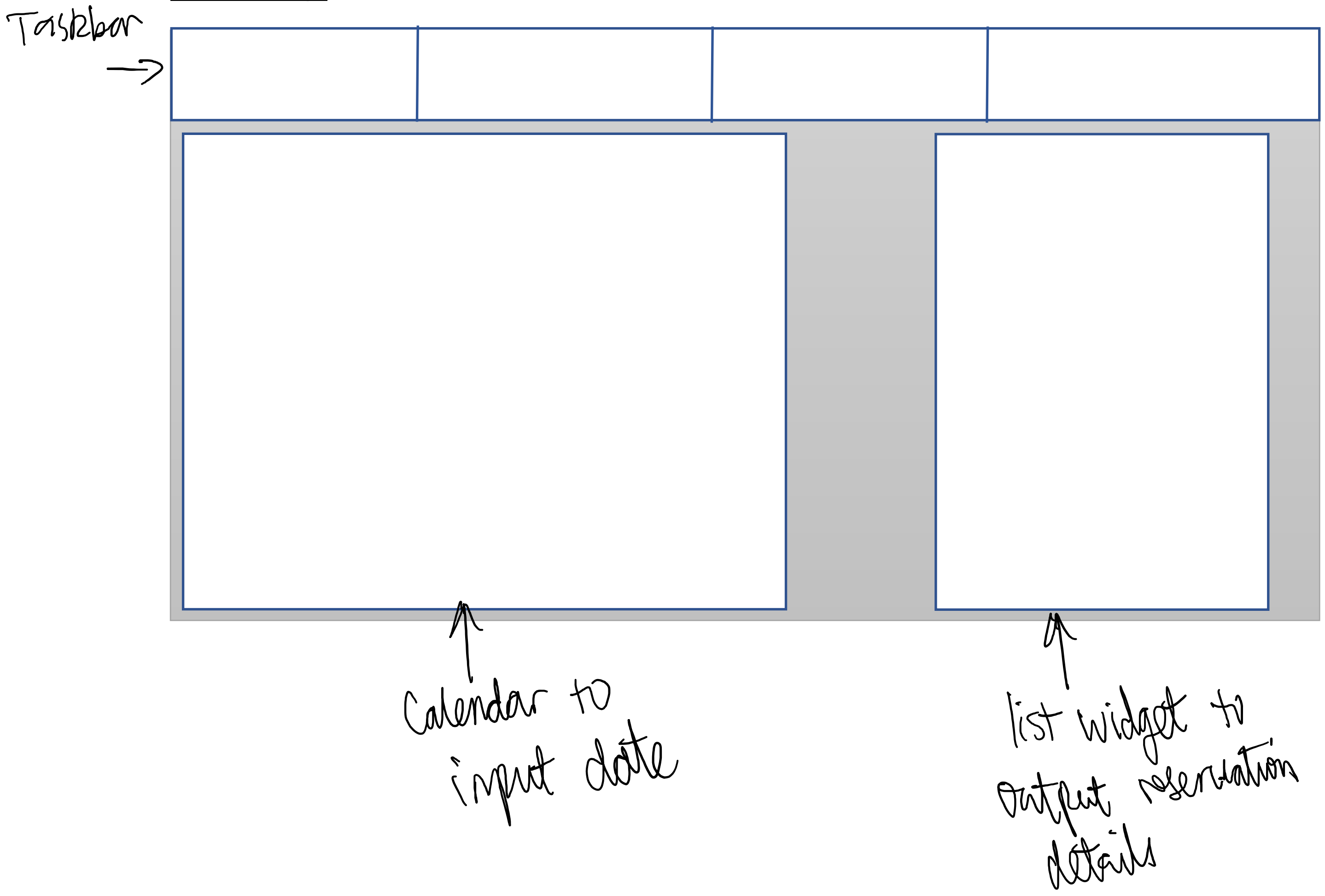
Table Selection



<u>Guest Details Page</u>


Taskbar
Date
Size
Time
Table
Details
line edit to input first name
line edit to input last name
line edit to input email address
error message location
Please enter a suitable ___

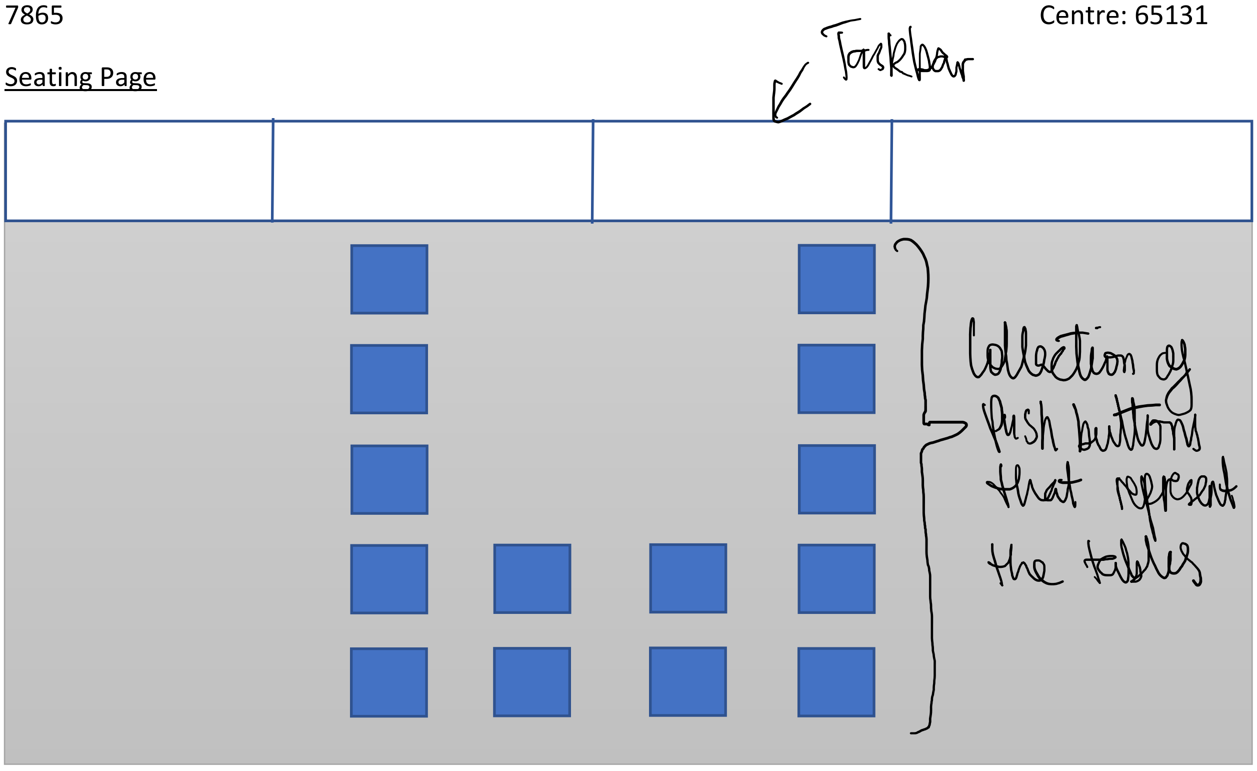
<u>Calendar Page</u>



Joshua Heffernan
7865

The College of Richard Collyer
Centre: 65131

Seating Page



Management Page

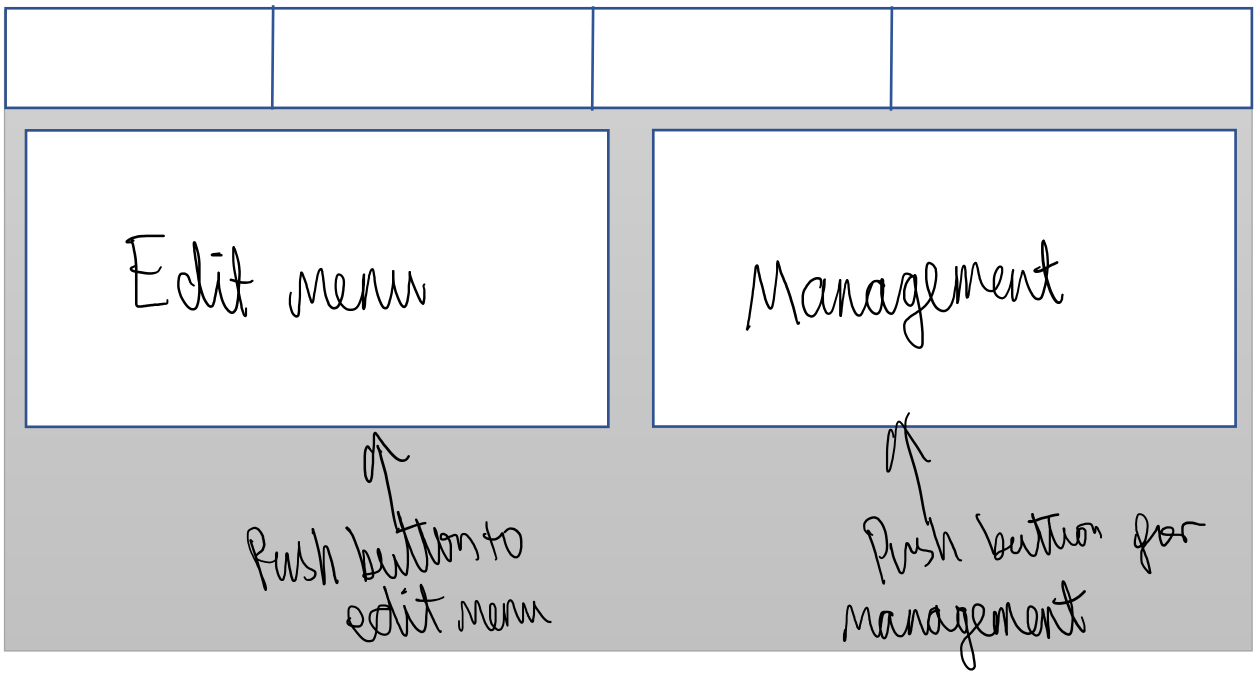
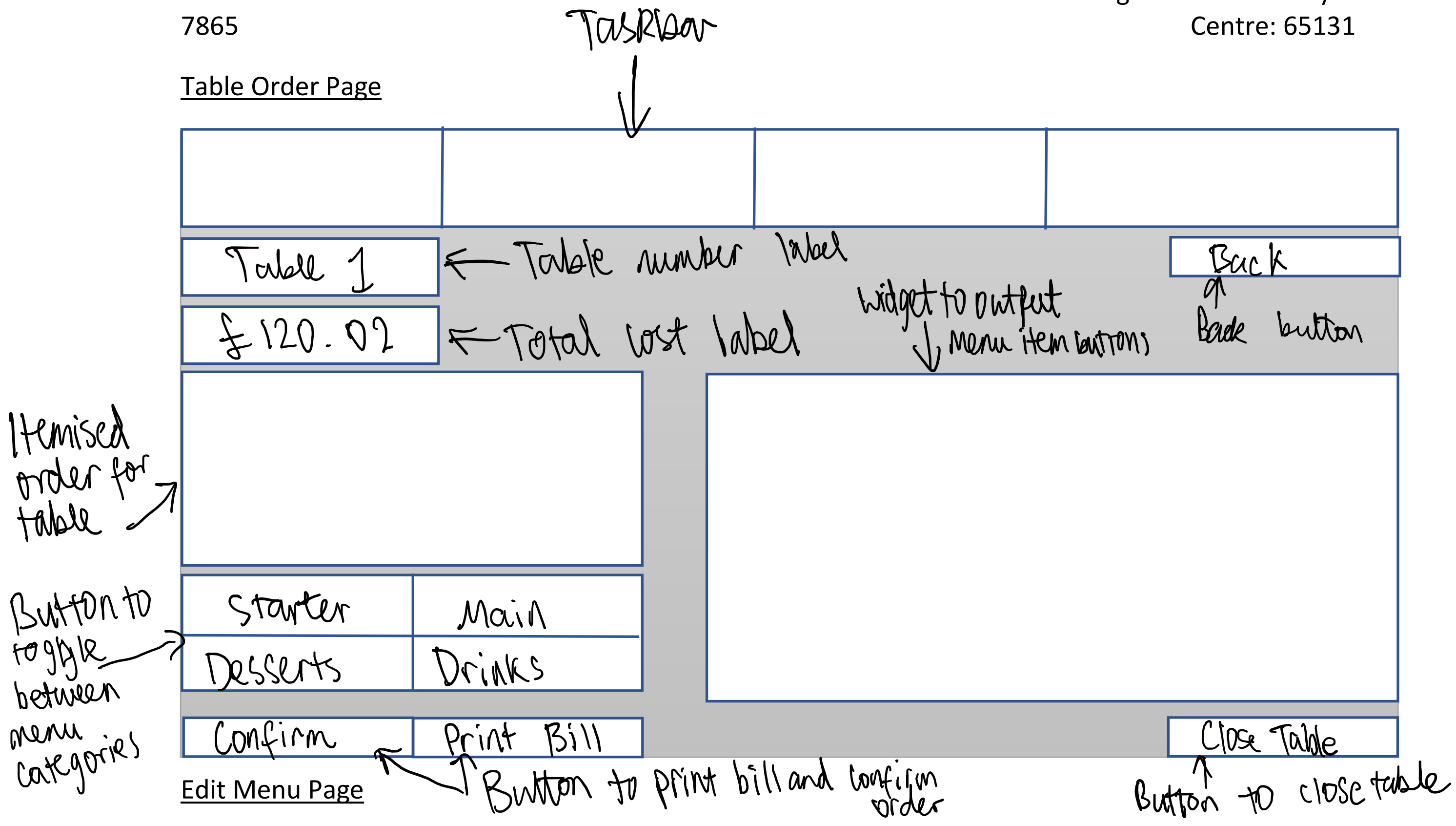
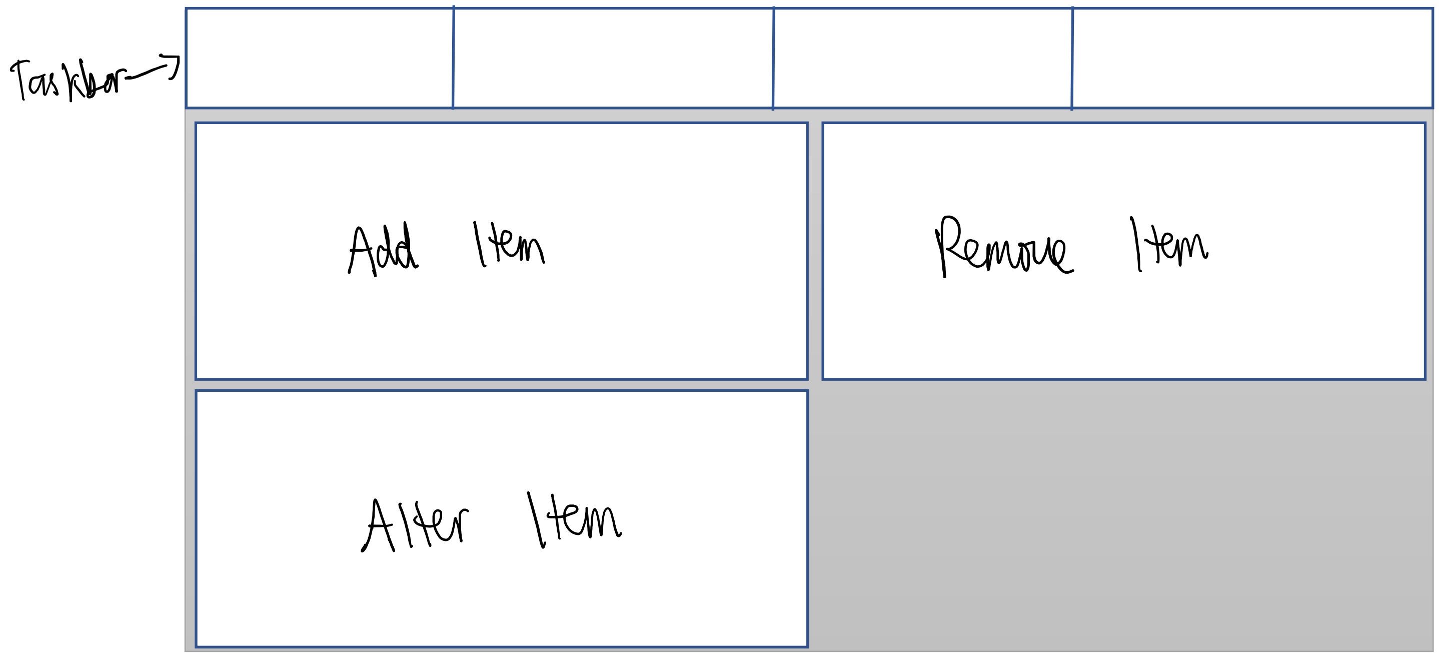


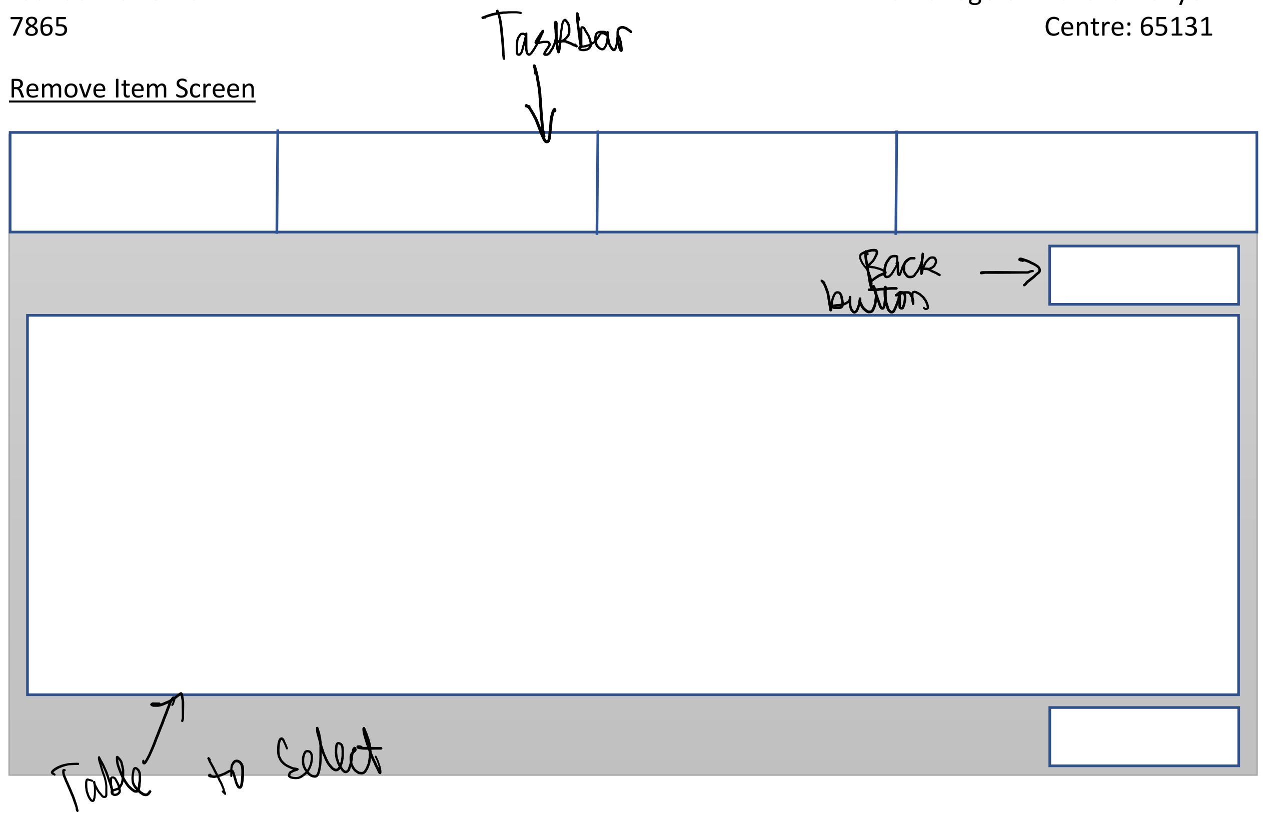
Table Order Page



Edit Menu Page



Remove Item Screen



Data structure

Describe why you will be storing the data in a SQL database, what features it has that suit your project etc

Ease of retrieval of data

In SQL, you can retrieve data through the use of queries. These queries are very easy to understand as they are in English. In addition to this, it is easy to link tables as you can simply identify the Primary Keys in each table. Also, you can link tables using the INNER JOIN feature which allows the user to execute a query that covers multiple tables. This is particularly useful in this system as a lot of the data is spread across multiple tables.

Inclusivity in Python

In Python, there are modules/libraries which allows the user to execute queries within the program. This is very good for the system, as all commands can be executed within one application. In addition to this, this allows the system to obtain the query results with a simple line of code. This is crucial for the system as it relies on the reservation details provided.

Methods of Access

You need to identify the methods of access for each of your purposes. These can be:

- Direct – searching directly using a primary key, highly efficient as it will only produce a single result, ie a customer number
- Indirect – searching using a variable that can be duplicated in a file, ie a customer name.

Direct Access

- “SELECT AccountID FROM Management WHERE (Password = ?)”
  - As every staff member has a unique password, this query will only produce one result.
- "SELECT MenuItems.OrderID, MenuItems.ItemName, Category.CategoryName, FORMAT(MenuItems.Price, 'N2') FROM MenuItems INNER JOIN Category ON MenuItems.CategoryID = Category.CategoryID"
  - As MenuItems.OrderID is a unique identifier for the table Menu Items, there will only be one record with this OrderID.
- "SELECT MenuItems.Price FROM MenuItems INNER JOIN TableOrder ON MenuItems.OrderID=TableOrder.OrderID INNER JOIN Sitting ON Sitting.SittingID=TableOrder.SittingID WHERE TableOrder.TableID = {self.tableid} AND Sitting.Active='TRUE'"
  - As there is only 1 record that is ‘active’ per table, this query will only select the values that are correspondent to intended table.
- "SELECT DISTINCT MenuItems.ItemName, MenuItems.Price FROM MenuItems INNER JOIN TableOrder ON MenuItems.OrderID=TableOrder.OrderID INNER JOIN Sitting ON Sitting.SittingID=TableOrder.SittingID WHERE TableOrder.TableID = {self.tableid} AND Sitting.Active='TRUE'"
  - As stated above, there is only 1 record that is active per table, which will cause only the intended items to be selected.
- "SELECT MAX(TableOrderID) FROM TableOrder;"
  - As this query will only select the maximum value of TableOrderID, there will only be one record that is selected every time this query is run.
- "DELETE FROM TableOrder WHERE TableOrder.TableID = {orderwindow.tableid} AND TableOrder.SittingID = {sittingid} AND TableOrder.OrderID = {orderid}"
  - As there is only one record with the same TableId, SittingID, and OrderID, this query is guaranteed to only perform the delete action on the intended record.
- "SELECT Capacity FROM [Table] WHERE TableID = '{0}';"
  - As TableID is the Primary key for the table ‘Table’, there is only 1 record per unique primary key value, making it direct access.
- "SELECT * from Customer WHERE EmailAddress = '{email}';"
  - As email addresses are unique to every customer, this query will only select one record.

Indirect Access

- "SELECT COUNT(*) FROM MenuItems WHERE CategoryID = '{0}';"
  - As CategoryID is a foreign key for the table MenuItems, multiple records can be selected from a query. This makes it indirect access.
- "SELECT ItemName FROM MenuItems WHERE CategoryID = '{0}';"
  - As stated above, CategoryID is a foreign key for MenuItems, which will allow multiple records to be selected from a query.
- "SELECT Price FROM MenuItems WHERE ItemName = '{item}';"
  - As some foods in different cultures may have the same name but be different entirely, searching using the itemname may cause multiple records to be selected.

- "SELECT TableReservation.TableID FROM TableReservation INNER JOIN Reservation ON TableReservation.ReservationID=Reservation.ReservationID WHERE (Reservation.Time = '{0}') AND (Reservation.Date = '{1}')"
    - ➢ As multiple reservations can happen at the same time and day, this query may result in multiple records being selected. Therefore this query causes indirect access.
- "SELECT "Table".TableID,"Table".Capacity FROM "TABLE" INNER JOIN TableOrder ON "Table".TableID=TableOrder.TableID INNER JOIN Sitting ON TableOrder.SittingID=Sitting.SittingID WHERE Sitting.Active ='TRUE'"
    - ➢ Since there can be many tables that are active at the same time, this query will result in multiple records being selected.
- "SELECT Reservation.NumberofPeople, TableIDs = STRING_AGG(TableReservation.TableID,','), CONCAT(Customer.FirstName,' ',Customer.LastName), SUBSTRING(convert(varchar, Reservation.Time,108),1,5) FROM Customer INNER JOIN Reservation ON Customer.CustomerID=Reservation.CustomerID INNER JOIN TableReservation ON TableReservation.ReservationID=Reservation.ReservationID WHERE Date = '{0}' GROUP BY Reservation.ReservationID,Reservation.NumberofPeople,Customer.FirstName,Customer.LastName, SUBSTRING(convert(varchar, Reservation.Time,108),1,5);"
    - ➢ Since there can be multiple bookings within one day, this query will result in multiple records being selected, making it indirect access.

<u>Customer</u>

Field Name	Data Type	Key	Method of Access
CustomerID	Integer	Primary Key	Direct
First Name	String		Indirect
Last Name	String		Indirect
Email Address	String		Direct

<u>Management</u>

Field Name	Data Type	Key	Method of Access
AccountID	Integer	Primary Key	Direct
Name	String		Indirect
Password	String		Direct

<u>Reservation</u>

Field Name	Data Type	Key	Method of Access
ReservationID	Integer	Primary Key	Direct
CustomerID	Integer	Foreign Key	Indirect (As customers may have multiple bookings)
AccountID	Integer	Foreign Key	Indirect (As staff members will have multiple bookings)

Reservation Time	Time		Indirect
Reservation Date	Date		Indirect
Number of People	Integer		Indirect

TableReservation

Field Name	Data Type	Key	Method of Access
ReservationID	Integer	Composite Key	Direct
TableID	Integer	Composite Key	Direct

Table

Field Name	Data Type	Key	Method of Access
TableID	Integer	Primary Key	Direct
Capacity	Integer		Indirect

Menu Items

Field Name	Data Type	Key	Method of Access
OrderID	Integer	Primary Key	Direct
CategoryID	Integer	Foreign Key	Indirect
Item Name	String		Indirect
Price	Float		Indirect

TableOrder

Field Name	Data Type	Key	Method of Access
TableOrderID	Integer	Primary Key	Direct
TableID	Integer	Foreign Key	Indirect
OrderID	Integer	Foreign Key	Indirect
SittingID	Integer	Foreign Key	Indirect

Sitting

Field Name	Data Type	Key	Method of Access
SittingID	Integer	Primary Key	Direct
Active	Boolean		Indirect

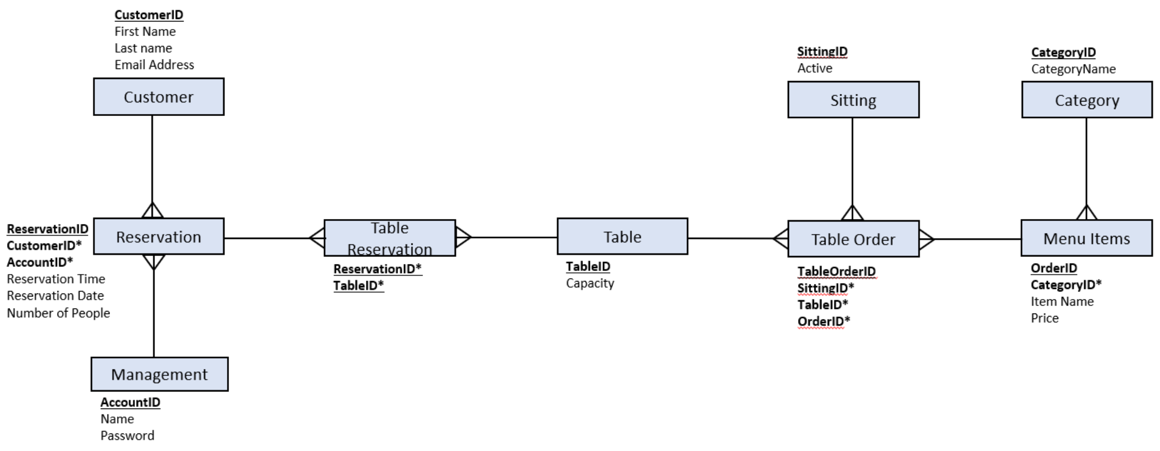
Category

Field Name	Data Type	Key	Method of Access
CategoryID	Integer	Primary Key	Direct
CategoryName	String		Indirect

Database design - ERD

Please make sure you indicate the relationships in a recognised convention, crows feet for Many side of the relationship or any symbol using UML software. You can do this using the following.

- Using Word Shapes as an image
- Using UML software to draw the diagram.
- Hand drawing the diagram and scanning in as an image.



Data Dictionary.

EVERY table you have identified in your ERD MUST have its own Data Dictionary!! Each one should have its own separate table.

Customer

Field Name	Data Type	Description	Key
CustomerID	Integer	This will be the unique identifier for the table Customer	Primary Key
First Name	String	This is the first name of the customer making the reservation.	
Last Name	String	This is the last name of the customer making the reservation.	
Email Address	String	This is the email address provided by the customer making the reservation.	

Management

Field Name	Data Type	Description	Key
AccountID	Integer	This will be the unique identifier for the table Management. In the program, this will be used as an identifier for different access levels between staff members.	Primary Key
Name	String	This field stores the name of the server in order to make it easier to identify which account belongs to who.	
Password	String	This is the passcode that will be needed to enter the system.	

		By inputting the correct passcode, the system will be able to link which account has logged in.	

Reservation

Field Name	Data Type	Description	Key
ReservationID	Integer	This will be the unique identifier for the table Reservation.	Primary Key
CustomerID	Integer	This is the primary key of the table ‘Customer’. This helps link the tables, Reservation and Customer.	Foreign Key
AccountID	Integer	This field is used to identify who has logged in. This also helps assign different access levels in the system. This links the table ‘Management’ to the table ‘Reservation’.	Foreign Key
Reservation Time	Time	This is the allocated time for the reservation. This is also important as it is used in the system to block certain tables from being booked.	
Reservation Date	Date	This is the allocated date for the reservation. This is also important as it is used in the system to block certain tables	

		from being booked.	
Number of People	Integer	This is the number of people that are allocated to the booking. This is used to determine if a party size will fit on a certain table etc.	

<u>TableReservation</u>

Field Name	Data Type	Description	Key
ReservationID	Integer	This will be the unique identifier for the table Reservation. This links both tables 'Reservation' and 'Table'	Composite Key
TableID	Integer	This field is the unique identifier for the table 'Table'. This represents the table number of a certain table. This links both tables 'Reservation' and 'Table'	Composite Key

<u>Table</u>

Field Name	Data Type	Description	Key
TableID	Integer	This field is the unique identifier for the table 'Table'. This represents the table number of a certain table.	Primary Key
Capacity	Integer	This field represents how much people can be sat at the table. This is	

		useful as it indicates whether a party size is too big or too small	

Menu Items

Field Name	Data Type	Description	Key
OrderID	Integer	This field is the unique identifier for the table 'MenuItems'. This represents the identifier of a certain table.	Primary Key
CategoryID	Integer	This is used to uniquely identify each record in the table 'Category' and identify which category an item is in. This links the table 'Category' to the table 'MenuItems'	Foreign Key
Item Name	String	This field is the name of the item on the menu. This will be used to output the buttons on the system.	
Price	Float	The field represents the price of the item. This is used to calculate the total of the customer's order.	

TableOrder

Field Name	Data Type	Description	Key
TableOrderID	Integer	This field is the unique identifier for the table 'TableOrder'.This	Primary Key

		is used to uniquely identify each record.	
TableID	Integer	This represents the table number of a certain table. This field links the table 'Table' to the table 'TableOrder'	Foreign Key
OrderID	Integer	This field is the unique identifier for each menu item. This field links the tables 'MenuItems' to the table 'TableOrder'	Foreign Key
Active	Boolean	This field represents whether a table is currently sitting in the restaurant.	
SittingID	Integer	This field represents whether a group is sitting together.	Foreign Key

<u>Sitting</u>

Field Name	Data Type	Description	Key
SittingID	Integer	This field represents whether a group is sitting together.	Primary Key
Active	Boolean	This field represents whether a table is currently sitting in the restaurant.	

Field Name	Data Type	Description	Key
CategoryID	Integer	This field is the unique identifier for the table 'Category'. This is	Primary Key

		used to uniquely identify each record and identify which category an item is in.	
CategoryName	String	This field represents which category an item is in, e.g., mains, starters, drinks, etc. This is used to separate the items on the ordering screen on the system.	

Validation

Probably the most important aspect of the project. This is looked at to judge the level of the project. You need to explain the different validation you will use. You should discuss each validation technique you will use from above and any try except statements or additional python validation you will include.

Table: Customer		
Field Name	**Validation**	**Expected Output**
CustomerID	Presence check – This field is the unique identifier. This means that it is needed in order to make it unique from other records.	
CustomerID	Data Type Check – The data type of the primary key must be an integer as integers are much easier to index compared to other data types	
FirstName	Presence Check	‘Please enter first name’
FirstName	Data type check – The contents of this field must be of string data type. This field should not contain any special characters or numbers. This is needed to make sure the inputs are sensible. For example, a customer is not going to have the name ‘986’.	‘Please enter a suitable first name’
LastName	Presence Check	‘Please enter a last name’

LastName	Data type check – This is needed to make sure the inputs are suitable and correct. For example, a customer is not going to have the surname '223'.	'Please enter a suitable last name'
Email Address	Presence Check – This field is very important for the reservation as it is the only form of contact between customer and restaurant.	'Please enter an email address'
Email Address	Format Check – This is also very important as emails follow a set structure. Therefore, if the input does not follow this structure, the system will not be able to send the confirmation email to the customer.	'Please enter the correct email format'

Table: Management		
Field Name	**Validation**	**Expected Output**
AccountID	Presence check – As this field is the unique identifier for the table 'Management', it is needed to be present in order to make it unique from other records.	
AccountID	Data Type Check – The data type of the primary key must be an integer as integers are much easier to index compared to other data types	
Name	Presence check – This is needed as it will help locate which account belongs to who.	'Please enter a name'
Name	Data Type Check – This is done to ensure the correct data type has been inputted as names are a string, not integers, etc.	'Please enter a suitable name'
Password	Presence Check – This is needed as a passcode is critical in order for the user to log in and to determine the access levels.	'Please enter a password'

Table: Reservation		
Field Name	**Validation**	**Expected Output**

ReservationID	Presence check – As this field is the unique identifier for the table 'Reservation', it is needed to be present in order to make it unique from other records.	
ReservationID	Data Type Check – The data type of the primary key must be an integer as integers are much easier to index compared to other data types	
CustomerID	Presence check – This field is the unique identifier. This means that it is needed in order to make it unique from other records	
CustomerID	Data Type Check – The data type of the primary key must be an integer as integers are much easier to index compared to other data types	
AccountID	Presence check – As this field is the unique identifier for the table 'Management', it is needed to be present in order to make it unique from other records.	
AccountID	Data Type Check – The data type of the primary key must be an integer as integers are much easier to index compared to other data types	
ReservationTime	Presence Check – This field is very important for the reservation as it confirms to the user what time each reservation is, which can help with preparation.	'Please select a reservation time'
ReservationTime	Format Check – This is also very important as the system will not work if the input is not in time format	'Please input the correct format for the reservation time'
ReservationDate	Presence Check - This field is important as it is used to block certain tables to prevent overbooking.	'Please select a reservation date'
NumberofPeople	Presence check – This is needed to determine whether a party can fit on the table allocated to them.	'Please select the number of people attending'

Table: Table Reservation		
Field Name	**Validation**	**Expected Output**
ReservationID	Presence check – As this field is the unique identifier for the table 'Reservation', it is needed to be present in order to make it unique from other records.	
ReservationID	Data Type Check – The data type of the primary key must be an integer as integers are much easier to index compared to other data types	
TableID	Presence check – This is crucial as it identifies which table is which in the restaurant.	
TableID	Data Type Check – As all of the table numbers are integers, the data type of this field must be integers.	
TableID	Range Check – As there is a limited amount of tables, a range check is necessary to ensure the TableID the restrictions.	

Table: Table		
Field Name	**Validation**	**Expected Output**
TableID	Presence check – This is crucial as it identifies which table is which in the restaurant.	
TableID	Data Type Check – As all of the table numbers are integers, the data type of this field must be integers.	
Capacity	Presence check – This is crucial as it identifies how many people can sit at a table in the restaurant.	
Capacity	Data Type Check – As all of the table capacities are integers, the data type of this field must be integers.	

Table: MenuItems		
Field Name	**Validation**	**Expected Output**
OrderID	Presence check – This is crucial as it uniquely identifies each item in the menu.	

OrderID	Data Type Check – The data type of this field is an integer as indexing is easier on integers.	
CategoryID	Presence check – This is important as it uniquely identifies each record in the table 'Category'	
ItemName	Presence check – This is important as it is used in the system to output the buttons when ordering. This will make the ordering experience easier for staff.	'Please enter item name'
Price	Presence check – This field is needed as it is used to calculate the total price of the customer's order.	'Please enter item price'
Price	Data type check – The data type of the input is also important as if the input were to be a string for example, the system would not be able to add a string to a float.	'Please enter a suitable price'

Table: TableOrder		
Field Name	**Validation**	**Expected Output**
TableOrderID	Presence check – This is important as it uniquely identifies each record.	
TableOrderID	Data Type Check – As there will be a large amounts of orders per table, integers will be the most efficient in indexing.	
TableID	Presence check – This is crucial as it identifies which table is which in the restaurant.	
TableID	Data Type Check – As all of the table numbers are integers, the data type of this field must be integers.	
OrderID	Presence check – This is crucial as it identifies which table is which in the restaurant.	
OrderID	Data Type Check – The data type of this field is an integer as indexing is easier on integers.	

Table: Sitting		
Field Name	**Validation**	**Expected Output**
SittingID	Presence check – This field is used to determine which orders belong to which group; therefore, it is needed to be present for SQL queries.	
SittingID	Data Type Check – As there will be a large amounts of sittings, integers will be the most efficient in indexing as there is always +1 of a previous number.	
Active	Presence check – This field needs to be present as it will be used in a lot of queries to gain the table orders.	

Table: Category		
Field Name	**Validation**	**Expected Output**
CategoryID	Presence check – This is important as it uniquely identifies each record.	
CategoryName	Presence check – This field needs to be included as the system will not be able to allocate the item to a certain group on the order screen.	

Algorithms

Each sub problem you have identified needs an algorithm. I don't need to see any user interface algorithms, but I need to see everything else. Please make sure you use suitable pseudocode, it can't look too 'pythony'

Function to get the party size:

```
DEFINE SUBPROCEDURE getpartysize():

    IF the CheckBox Custom is checked:

        SET partysize = value in spinbox
    ELSE:
      SET partysize = value selected in ListWidget

  RETURN partysize
END
```

Function to get the Reservation time:

```
DEFINE SUBPROCEDURE gettime:
   SET bookingtime = value selected in ListWidget

RETURN bookingtime
```

Function to get unavailable tables:

```
DEFINE SUBPROCEDURE unavailabletables:
    SET date = value returned from running function getdate
    SET bookingtime = value returned from running function gettime
    SET unavailabletables = array[]
    FOR i = 0 to 5:
        Execute query that selects TableID where reservation time = (bookingtime - i*30) and
        reservation date = date
        Append values returned to the array unavailabletables[]
    FOR i = 1 to 3:
        Execute query that selects TableID where reservation time = (bookingtime + i*30) and
        reservation date = date
Append values returned to the array unavailabletables[]
RETURN unavailabletables[]
END
```

Function to get capacity of table:

```
DEFINE SUBPROCEDURE gettablecapacity(tablenum):
    Execute query to select capacity from Table where the TableID = tablenum(the parameter pass
in the function)
    Capacity = value returned from query
    RETURN capacity
```

```
END
```

Function to verify size of party and table:

```
DEFINE SUBPROCEDURE sizeverification(tablenum):
    SET size = value returned from running function getpartysize()
    SET tablecapacity = value returned from running function gettablecapacity(tablenum)
    IF size > tablecapacity:
        Deselect the table the user is trying to select
    ELSEIF (tablecapacity-3)> size:
        Deselect the table the user is trying to select
    ELSE:
      Run function enableguestbutton()
END
```

Function to get customerID of new record in database:

```
DEFINE SUBPROCEDURE getCustomerID():
        Execute query to select the maximum value of customerid in the table Customer
        SET customerID = result of query +1
        RETURN customerid
END
```

Function to create a reservation:

```
DEFINE SUBPROCEDURE createbooking():
        SET date = value returned from running function getdate()
        SET partysize = value returned from running function getpartysize()
        SET bookingtime = value returned from running function gettime()
        SET customerid = value returned from running function getCustomerID()
        SET ReservationID = value returned from running function getReservationID()
        SET firstname = value inputted in the line edit leFName
        SET lastname = value inputted in the line edit leLName
        SET email = value inputted in the line edit leEmail
        SET accountID = value returned from running function getAccountID
        SET tableid = value returned from getTableID
        Execute query that will insert the values customerid, firstname, lastname, and email into the
```

```
    table Customer.

    Execute query that will insert the values Reservationid,  customerid, AccountID,

    bookingtime, date, and partysize into the table Reservation.

    Execute query that will insert the values Reservationid and tableid into the table
    TableReservation.

END
```

Function to output reservations for the day:

```
DEFINE SUBPROCEDURE outputReservations():

    TRY:

        SET date = Selected date in calendar

        SET date = date converted into PyDate format

        Execute query to Select NumberofPeople, TableID, First name and last name, and
        time, from tables Customer, Reservation and TableReservation.

    EXCEPT:

        PASS

END
```

Function to get total price due for a table:

```
DEFINE SUBPROCEDURE gettotalprice():

    SET totalprice = 0

    Execute query to obtain the prices of the ordered items for a table in the form of an array

    SET menuitemsprice = Array returned from running the query

    For i = 0 to length of menuitemsprice:

        Totalprice = totalprice + menuitemsprice[i]

END
```

Function to get the items ordered by a table:

```
DEFINE SUBPROCDEURE getordereditems():

    Execute query to select the distinct Item Names for the selected table

    SET ordereditems = array returned from the SQL query

    return value of ordereditems

END
```

Function to get the SittingID when making an order to a table:

```
DEFINE SUBPROCEDURE getSittingID():
    Execute query that selects the SittingID where the TableID is the same as the selected table
    and the Active field is set to 'TRUE'
    SET activesittingid = value returned from executing query
    IF activesittingid has no value (None):
        Execute query to select the maximum SittingID in the table Sitting
        SET sittingid = value returned from the query
        IF sittingid has no value:
        Sittingid = 1
        ELSE:
            Sittingid = sittingid + 1
    ELSE:
        Sittingid = activesittingid
    Return value of sittingid
END
```

Function to change the quantity of an ordered menu item:

```
DEFINE SUBPROCEDURE change_clicked():
    SET sittingid = value returned from running function getSittingID()
    SET orderid = calue returned from running function getOrderID()
    SET itemquantity = value assigned to the spinbox on the page
    For i = 0 to itemquantity:
        SET tableorderid = value returned from running function getTableOrderID()
        Execute query to insert the TableOrderID, TableID, OrderID, and SittingID to the
        table TableOrder
    Close the change quantity window
END
```

Function to log user into system:

```
DEFINE SUBPROCEDURE login():
    Execute query to select the AccountID where the password is equal to the password
    inputted by the user
    SET loginverification = result returned from running the query
    If loginverification has no value (None):
```

```
                Output error message saying incorrect password
        Else:
                SET self.accountid = loginverification
                Display the home page by changing the current index of the main display widget
END
```