Software Development

Computer Science NEA

Joshua Heffernan

Contents

Annotated Listings

Name of Variable	Type	Scope	Description	Function(s) in which it appears
Loginpage.password	String	Instance Variable	This is used to store the password that the user has entered when logging in	Loginpage.passwordinput, Loginpage.clear, Loginpage.outputpassword, Loginpage.login
x	Integer	Local Variable	This is used to store the x coordinate of the button that is about to be outputted	Loginpage.printkeypadbuttons
y	Integer	Local Variable	This is used to store the y coordinate of the button that is about to be outputted	Loginpage.printkeypadbuttons
text	String	Local Variable	This is used to store the text that will outputted onto the button	Loginpage.printkeypadbuttons
loginverification	Array	Local Variable	Used to store the result of the query 'loginverification'. Used to determine whether the user/password exists.	Loginpage.login
Loginpage.accountid	Integer	Instance Variable	Used to store the accounted of the logged in user. It is stored as an instance variable to access the accountid throughout the program	Loginpage.login, guestdetailswindow.createbooking
value	Integer	Local Variable	Used to store OrderID/index of the selected item in the table.	removeconfirmationscreen.remove_clicked
category	String	Local Variable	Used to store the category of the selected item in the table.	removeconfirmationscreen.remove_clicked

itemname	String	Local Variable	Used to store the name of the item that the user wants to add/alter/remove	AlterConfirmation.alteritem, AddItemScreen.addconfirmation, RemoveItemScreen.getselectedItem, OrderScreen.alteritem, AlterConfirmation.fillinputs, OrderScreen.getselecteditem, RemoveItemScreen.remove_clicked, AddConfirmation.addconfirm_clicked, AddItemScreen.itemnamevalidation, OrderScreen.getOrderID
category	Integer	Local Variable	Used to store the index of the selected category of the item that the user wants to alter/add	Addconfirmationscreen.addconfirm_clicked, alterconfirmation.fillinputs, alterconfirmation.alteritem, additemwindow.addconfirmation, orderwindow.numofitemsincategory, orderwindow.getmenuitems, starterspage.hideitembuttons, starterspage.printitembuttons, removeconfirmationscreen.remove_clicked
price	Float	Local Variable	Used to store the price of the menu item that the user wants to use/alter/add	Addconfirmationscreen.addconfirm_clicked, alterconfirmation.fillinputs, alterconfirmation.alteritem, additemwindow.addconfirmation, orderwindow.getitemprice, orderwindow.addrowtotable, orderwindow.addtoorderlist
orderid	Integer	Local Variable	Used to the store the OrderID of the menu item that is going to be used in a process, such as adding an item, removing an item, altering an item, or using the OrderID in a query.	Addconfirmationscreen.addconfirm_clicked, Addconfirmationscreen.getOrderID, alterconfirmation.alteritem, orderwindow.getitemorderid, orderwindow.getOrderID, orderwindow.addtoorderlist,

				changequantitywindow.change_clicked, changequantitywindow.clearitemrecords
orderidquery	String	Local Variable	Used to store the query that is used to obtain the largest OrderID in the system.	Addconfirmationscreen.getOrderID
query	String	Local Variable	Used to store the query that is going to be executed on the database	Alteritemwindow.fillitemmenutable, removeconfirmationscreen.remove_clicked, additemwindow.fillmenutable, removeitemwindow.fillmenutable, orderwindow.gettotalprice, orderwindow.getordereditems, orderwindow.getitemquantity, orderwindow.getitemorderid, orderwindow.getTableOrderID, orderwindow.getSittingID, orderwindow.getOrderID, orderwindow.numofitemsincategory, tableselectionwindow.unavailabletables, tableselectionwindow.gettablecapacity, guestdetailswindow.customercheck, seatingwindow.setactivetableimages, seatingwindow.settableimages, calendarwindow.reservationquery, calendarwindow.executequery, orderwindow.addToOrder, Loginpage.login, changequantitywindow.change_clicked, changequantity.clearitemrecords, AddConfirmation.addconfirm_clicked, OrderScreen.getitemprice, OrderScreen.getSittingID
rows	Array	Local Variable	Used to store the rows of records that will be outputted into the table	TableSelection.unavailabletables, AddItemScreen.fillmenutable, Statistics.filltable,

				AlterMenuItem.fillitemmenutable, Calendar.executequery, Staff.filltable, RemoveItemScreen.fillmenutable
updateitem	String	Local Variable	Used to store the query that is used to alter a menu item	alterconfirmation.alteritem
rowindex	String	Local Variable	Used to store the index of the selected row from an input widget	Removeitemwindow.getselectedItem, timeselectionwindow.gettime
numofitemsincategory	Integer	Local Variable	Used to store the number of menu items that have been allocated to a specific category	orderwindow.numofitemsincategory, starterspage.hideitembuttons, starterspage.printitembuttons
menuitemsincategory	Array (Elements are string type)	Local Variable	Used to store a list of all the menu items in a certain category	StartersPage.printitembuttons, OrderScreen.getmenuitems, StartersPage.hideitembuttons
menuitemsprice	Array (Elements are real/float type)	Local Variable	Used to store the prices of the selected menu items	orderwindow.gettotalprice
ordereditems	Array(Elements are string)	Local Variable	Used to store the ordered items and its features as a 2D array	Orderwindow.getordereditems, orderwindow.outputordereditems
itemquantity	Integer	Local Variable	Used to store the quantity of an ordered item	Orderwindow.getitemquantity, orderwindow.outputordereditems, orderwindow.addrowtotable, changequantitywindow.change_clicked
Orderwindow.orderlist	Array (Elements are string)	Instance Variable	Used to store an array of the menu items that the user would like to order	Orderwindow.gettableid, orderwindow.addtoorderlist, orderwindow.addToOrder, orderwindow.outputordereditems
Orderwindow.tableid	Integer	Instance Variable	Used to store the TableID of the table that the user wants to add/remove food onto	Closeconfirmation.remove_clicked, orderwindow.alteritem, orderwindow.closetable, orderwindow.gettotalprice, orderwindow.getordereditems,

				orderwindow.getitemquantity, orderwindow.gettableid, orderwindow.addToOrder, orderwindow.getSittingID, Changequantitywindow.change_clicked, changequantitywindow.clearitemrecords
ordereditem	String	Local Variable	Used to store the name of the Menu Item that is going to be ordered	Orderwindow.outputordereditems
ordereditems	Array (Elements are string)	Local Variable	Used to store a list of menu items that have been ordered in the form of an array	orderwindow.outputordereditems, orderwindow.getordereditems
rowPosition	Integer	Local Variable	Used to store the index of the row that the user would like to modify	orderwindow.addrowtotable
largesttableorderid	Integer	Local Variable	Used to store the largest tableorderID that is currently stored in the database	Orderwindow.getTableOrderID
activesittingid	Integer	Local Variable	Used to store the SittingID that is linked to the selected table and that is set to the 'Active' (in use) state	Orderwindow.getSittingID
sittingid	Integer	Local Variable	Used to store the SittingID that is going to be used in processes (such as queries, etc)	Orderwindow.getSittingID, changequantitywindow.change_clicked, Changequantitywindow.clearitemrecords, orderwindow.addToOrder,
today	Date	Local Variable	Used to store the current date in Py format	Dateselectionwindow.setminimumdate, Dateselectionwindow.setmaximumdate, Calendar.setmaximumdate
date	Date	Local Variable	Used to store the date that is going to processed (whether is being for creating bookings	Dateselectionwindow.getdate, tableselectionwindow.unavailabletables, guestdetailswindow.createbooking,

			or searching for bookings, etc)	bookingconfirmation.insertbookingdetails, calendarwindow.runreservationquery
partysize	Integer	Local Variable	Used to store the number of guests that will be attending to a reservation	TableSelection.enableguestbutton, TableSelection.updateguestsleft, GuestDetails.displayreservationdetails, PartySizeSelection.getpartysize, GuestDetails.createbooking, TableSelection.sizeverification
bookingtime	String	Local Variable	Used to store the time that the user has selected for their reservation (in the form HH:MM)	Timeselectionwindow.gettime, timeselectionwindow.converttime, tableselectionwindow.unavailabletables, guestdetailswindow.createbooking, bookingconfirmation.insertbookingdetails
convertedtime	TimeDelta	Local Variable	Used to store the time that the user has selected for their reservation in TimeDelta Format/Data Type	timeselectionwindow.converttime
tablecapacity	Integer	Local Variable	Used to store the capacity of a specified table	Tableselectionwindow.enableguestbutton, tableselectionwindow.gettablecapacity, tableselectionwindow.getselectedtables, tableselectionwindow.sizeverification, closetableconfirmation.remove_clicked
Mainwindow.currenttab	String	Instance Variable	Used the store the most recently pressed pushbutton in the taskbar	MainWindow.__init__, MainWindow.toggleselectedtab
password	String	Local Variable	Used to store the password that the user has entered in the input field.	Staff.passwordvalidation, Staff.alterdetails, Staff.fillinputs
manager	Integer	Local Variable	Used to store the value of the combo box ‘cbManager’, which stores whether the record(staff member) is a manager or not.	Staff.fillinputs, Staff.alterdetails, LoginPage.managercheck

accountid	Integer	Local Variable	Used to store the AccountID of the staff member that will have their details altered	Staff.alterdetails
name	String	Local Variable	Used to store either the name of the customer or the staff member	Calendar.removebooking, Staff.alterdetails
usernamevalid	Boolean	Local Variable	Used to store the result whether the entered username is valid or not	Staff.alterdetails
passwordvalid	Boolean	Local Variable	Used to store the result whether the entered password is valid or not	Staff.alterdetails
regex	String	Local Variable	Stores the regular expression used to validate certain fields	GuestDetails.lastnamevalidation, GuestDetails.emailvalidation, GuestDetails.firstnamevalidation, Staff.usernamevalidation, AddItemScreen.itemnamevalidation
username	String	Local Variable	Stores the username (Name of staff) that's record is going to be altered	Staff.fillinputs, Staff.usernamevalidation
data	String	Local Variable	Stores the data that is going to be inputted into the cell in a table widget	AddItemScreen.fillmenutable, Statistics.filltable, AlterMenuItem.fillitemmenutable, Calendar.executequery, Staff.filltable, RemoveItemScreen.fillmenutable
namevalidation	Boolean	Local Variable	Stores whether the item name inputted is valid or not	AlterConfirmation.fieldvalidation, AddItemScreen.fieldvalidation
categoryvalidation	Boolean	Local Variable	Stores whether the category selected is valid or not	AddItemScreen.fieldvalidation
stylesheet	String	Local Variable	Stores the stylesheet that is going to be applied onto a widget	Seating.settableimages, CloseTableConfirmation.remove_clicked, Seating.setactivetableimages

totalprice	Float	Local Variable	Stores the total price spent on a table	OrderScreen.gettotalprice
tablenum	Integer	Local Variable	Stores table number of selected table	Seating.displayorderscreen, Seating.outputtablebuttons
tableorderid	Integer	Local Variable	Stores the TableOrderID of the record that is going to be added/altered	ChangeQuantity.change_clicked, OrderScreen.addToOrder, OrderScreen.getTableOrderID
xcoordinate	Integer	Local Variable	Stores the x coordinate of the button that is going to be printed	StartersPage.printitembuttons
maximum	Date	Local Variable	Stores the maximum date that can be stored	DateSelection.setmaximumdate, Calendar.setmaximumdate
Tableselectionwindow.seated	Integer	Instance Variable	Stores the number of seats that have been selected when making reservations	TableSelection.enableguestbutton, TableSelection.updateguestsleft, TableSelection.getselectedtables, TableSelection.cleartables, TableSelection.sizeverification
Tableselectionwindow.selectedtables	Array (Elements stored as integers)	Instance Variable	Stores an array of the TableIDs that the user has selected	TableSelection.cleartables, TableSelection.getselectedtables, GuestDetails.createbooking, GuestDetails.displayreservationdetails
unavailabletables	Array (Elements stored as integers)	Local Variable	Stores an array of the TableIDs that the are unavailable to select at that time	TableSelection.disabletables
columncoordinate	Integer	Local Variable	Stores column coordinate of the button that is going to be added	TableSelection.outputtables, Seating.outputtablebuttons
verificationresult	Boolean	Local Variable	Stores whether the selected tables have passed the verification process	TableSelection.getselectedtables

firstname	String	Local Variable	Stores First name of the customer that is entered in the line edit	GuestDetails.firstnamevalidation, GuestDetails.createbooking
lastname	String	Local Variable	Stores Last name of the customer that is entered in the line edit	GuestDetails.lastnamevalidation, GuestDetails.createbooking
email	String	Local Variable	Stores email of the customer that is entered in the line edit	GuestDetails.customercheck, GuestDetails.createbooking
fnamevalid	Boolean	Local Variable	Stores whether the first name entered by the user is valid or not	GuestDetails.validatefields
lnamevalid	Boolean	Local Variable	Stores whether the last name entered by the user is valid or not	GuestDetails.validatefields
emailvalid	Boolean	Local Variable	Stores whether the email entered by the user is valid or not	GuestDetails.validatefields
customerid	Integer	Local Variable	Stores the customerid of the customer that is going to create a reservation	GuestDetails.getcustomerid, GuestDetails.createbooking
reservationid	Integer	Local Variable	Stores the ReservationID of a reservation	GuestDetails.getreservationid, RemoveBooking.remove_clicked, Calendar.removebooking, GuestDetails.createbooking
customerquery	String	Local Variable	Stores the query that will input data into the customer table	GuestDetails.createbooking
reservationquery	String	Local Variable	Stores the query that will input data into the reservation table	GuestDetails.createbooking
tablereservationquery	String	Local Variable	Stores the query that will input data into the TableReservation table	GuestDetails.createbooking

fullname	String	Local Variable	Stores the full name of a customer	BookingConfirmation.insertbookingdetails
numbergoing	Integer	Local Variable	Stores the number of people going in a reservation	BookingConfirmation.insertbookingdetails

Programming Code

```
import pyodbc

import re

from datetime import datetime, timedelta, date

from PyQt5.QtWidgets import *

from PyQt5.QtGui import *

from PyQt5.QtCore import *

import time

from Bookings import *

from Calendar import *

from Seating import *

from DateSelection import *

from PartySizeSelection import *

from TimeSelection import *

from TableSelection import *

from GuestDetails import *

from MainWindow import *

from OrderScreen import *

from StartersPage import *
```

```
from MainsPage import *
from DessertsPage import *
from DrinksPage import *
from ManagementScreen import *
from EditMenu import *
from RemoveMenuItem import *
from RemoveConfirmation import *
from AddMenuItem import *
from CloseTableConfirmation import *
from AddMenuItem import *
from AddConfirmation import *
from ChangeQuantity import *
from AlterMenuItem import *
from AlterItemConfirmation import *
from LoginPage import *
from BookingConfirmation import *
from QRC import *
from Statistics import *
from RemoveBooking import *
from Staff import *
```

```
cs = (
   "Driver={SQL Server};"
   "Server=svr-cmp-01;"
   "Database=21HeffernaJN42;"
   "Trusted_Connection=yes;"
   "UID=COLLYERS\21HeffernaJN42;"
   "pwd=SY219842"
)
##            "Driver={SQL Server};"
##            "Server=DESKTOP-MDD6DGU\SQLEXPRESS01;"
##            "Database=Restaurant;"
##            "Trusted_Connection=yes;"
##            )



class MainWindow(QDialog):
   def __init__(self):
      super(QDialog, self).__init__()
      self.ui = Ui_MainWindow()
```

```
        self.ui.setupUi(self)
        self.currenttab = "pbBookings"
        self.toggleselectedtab("pbBookings")
        self.ui.DisplayWidget.setCurrentIndex(0)
        self.ui.pbBookings.clicked.connect(self.bookings_clicked)
        self.ui.pbCalendar.clicked.connect(self.calendar_clicked)
        self.ui.pbSeating.clicked.connect(self.seating_clicked)
        self.ui.pbManagement.clicked.connect(self.management_clicked)
        self.ui.pbLogout.clicked.connect(self.logout_clicked)

    def logout_clicked(self):
        Mainwindow.close()
        loginpage.show()

    def bookings_clicked(self):
        self.toggleselectedtab("pbBookings")
        self.ui.DisplayWidget.setCurrentIndex(0)
        bookingwindow.datepage()
        self.clearbookingdetails()
```

```
    def calendar_clicked(self):
        self.toggleselectedtab("pbCalendar")
        self.ui.DisplayWidget.setCurrentIndex(1)

    def seating_clicked(self):
        self.toggleselectedtab("pbSeating")
        seatingwindow.settableimages("seatingwindow", "pushButton")
        seatingwindow.setactivetableimages()
        self.ui.DisplayWidget.setCurrentIndex(2)

    def management_clicked(self):
        self.toggleselectedtab("pbManagement")
        self.ui.DisplayWidget.setCurrentIndex(4)

    def clearbookingdetails(self):
        bookingwindow.resetbuttons()
        dateselectionwindow.setminimumdate()
        partysizewindow.clearsizeselection()
        timeselectionwindow.cleartimeselection()
        tableselectionwindow.cleartables()
```

```
        guestdetailswindow.cleardetails()


    def toggleselectedtab(self, button):
        exec(f'self.ui.{self.currenttab}.setStyleSheet("background-color: #474747;")')
        exec(f'self.ui.{button}.setStyleSheet("background-color: #383838;")')
        self.currenttab = button



################################################
class LoginPage(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_LoginPage()
        self.ui.setupUi(self)
        self.printkeypadbuttons()
        self.password = ""

    def printkeypadbuttons(self):
        y = 210
        count = 0
```

```
        for i in range(0, 3):
            x = 270
            for j in range(0, 3):
                count += 1
                exec("self.pushButton{0} = QtWidgets.QPushButton(self)".format(count))
                exec(
                    "self.pushButton{0}.setGeometry(QtCore.QRect({1}, {2}, 71, 61))".format(
                        count, x, y
                    )
                )
                exec("self.pushButton{0}.setEnabled(True)".format(count))
                exec("self.pushButton{0}.setText('{0}')".format(count))
                exec("self.pushButton{0}.setIconSize(QtCore.QSize(9, 8))".format(count))
                exec("self.pushButton{0}.setObjectName('Button{0}')".format(count))
                exec(
                    "self.pushButton{0}.clicked.connect(lambda: loginpage.passwordinput(str({0})))".format(
                        count
                    )
                )
                exec("self.pushButton{0}.show()".format(count))
```

```
            x += 70
        y += 60
    x = 270
    for k in range(0, 3):
        if k == 0:
            text = "Clear"
        elif k == 1:
            text = 0
        else:
            text = "Confirm"
        exec("self.pushButton{0} = QtWidgets.QPushButton(self)".format(text))
        exec(
            "self.pushButton{0}.setGeometry(QtCore.QRect({1}, {2}, 71, 61))".format(
                text, x, 390
            )
        )
        exec("self.pushButton{0}.setEnabled(True)".format(text))
        exec("self.pushButton{0}.setText('{0}')".format(text))
        exec("self.pushButton{0}.setIconSize(QtCore.QSize(9, 8))".format(text))
        exec("self.pushButton{0}.setObjectName('Button{0}')".format(text))
```

```
        if text == 0:
            exec(
                "self.pushButton{0}.clicked.connect(lambda: loginpage.passwordinput(str({0})))".format(
                    text
                )
            )
        if text == "Confirm":
            exec(
                "self.pushButton{0}.clicked.connect(lambda: loginpage.login())".format(
                    text
                )
            )
        if text == "Clear":
            exec(
                "self.pushButton{0}.clicked.connect(lambda: loginpage.clear())".format(
                    text
                )
            )
        exec("self.pushButton{0}.show()".format(text))
        x += 70
```

```
    def passwordinput(self, clicked):
        self.password += f"{clicked}"
        self.outputpassword()

    def clear(self):
        self.ui.lblError.clear()
        self.password = ""
        self.outputpassword()

    def outputpassword(self):
        self.ui.lePassword.setText(self.password)

    def managercheck(self, manager):
        print(manager)
        if manager == 0:
            Mainwindow.ui.pbManagement.setEnabled(False)
        else:
            Mainwindow.ui.pbManagement.setEnabled(True)
```

```
    def login(self):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = "SELECT AccountID, Manager FROM Management WHERE (Password = '{0}');".format(
            self.password
        )
        cursor.execute(query)
        loginverification = cursor.fetchone()
        print(loginverification)
        if len(self.password) > 6 or len(self.password) == 0:
            self.ui.lblError.setText("Please Enter a Valid Passcode")
        elif loginverification == None:
            self.ui.lblError.setText("Incorrect Password Entered")
        else:
            self.clear()
            self.accountid = loginverification[0]
            Mainwindow.bookings_clicked()
            self.managercheck(loginverification[1])
            Mainwindow.show()
            loginpage.close()
```

```
################################################
class RemoveConfirmation(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Confirmation()
        self.ui.setupUi(self)
        self.ui.pbRemove.clicked.connect(self.remove_clicked)
        self.ui.pbCancel.clicked.connect(self.cancel_clicked)

    def cancel_clicked(self):
        removeconfirmationscreen.close()

    def remove_clicked(self):
        value = removeitemwindow.getselectedItem(0, "twMenuItems", "removeitemwindow")
        category = removeitemwindow.getselectedItem(
            2, "twMenuItems", "removeitemwindow"
        )
        if category == "Starter":
```

```
        starterspage.printitembuttons(1, "starterspage")
        starterspage.hideitembuttons(1, "starterspage")
    if category == "Main":
        starterspage.printitembuttons(2, "mainspage")
        starterspage.hideitembuttons(2, "mainspage")
    if category == "Desserts":
        starterspage.printitembuttons(3, "dessertspage")
        starterspage.hideitembuttons(3, "dessertspage")
    else:
        starterspage.printitembuttons(4, "drinkspage")
        starterspage.hideitembuttons(4, "drinkspage")
    query = "DELETE FROM MenuItems WHERE OrderID={0}".format(value)
    cnxn = pyodbc.connect(cs)
    cursor = cnxn.cursor()
    cursor.execute(query)
    cursor.commit()
    cnxn.close()
    removeconfirmationscreen.close()
    removeitemwindow.fillmenutable()
```

```
################################################
class Staff(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Staff()
        self.ui.setupUi(self)
        self.ui.twStaff.doubleClicked.connect(self.fillinputs)
        self.ui.pbConfirm.clicked.connect(self.alterdetails)

    def clearinputs(self):
        self.ui.lblError.setText("")
        self.ui.leUserName.clear()
        self.ui.lblErrorUserName.setText("")
        self.ui.lePassword.clear()
        self.ui.lblErrorPassword.setText("")
        self.ui.cbManager.setCurrentIndex(0)

    def alterdetails(self):
        name = self.ui.leUserName.text()
```

```
        password = self.ui.lePassword.text()
        manager = self.ui.cbManager.currentIndex()
        self.ui.lblError.setText("")
        if self.ui.twStaff.selectedItems() != []:
            usernamevalid = self.usernamevalidation()
            passwordvalid = self.passwordvalidation()
            if usernamevalid == True and passwordvalid == True:
                accountid = removeitemwindow.getselectedItem(
                    0, "twStaff", "staffwindow"
                )
                query = f"UPDATE Management SET Name = '{name}', Password = '{password}', Manager = '{manager}' WHERE AccountID = {accountid}"
                cnxn = pyodbc.connect(cs)
                cursor = cnxn.cursor()
                cursor.execute(query)
                cursor.commit()
                cnxn.close()
                self.filltable()
                self.clearinputs()
        else:
            self.ui.lblError.setText("Please Select a User to alter")
```

```
    def passwordvalidation(self):
        self.ui.lePassword.setStyleSheet("border: 1px solid #2E2E2E")
        self.ui.lblErrorPassword.setText("")
        password = self.ui.lePassword.text()
        if password.isspace() or password == "":
            self.ui.lePassword.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorPassword.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorPassword.setText("Please Enter a Password")
        elif password.isnumeric() == False:
            self.ui.lePassword.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorPassword.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorPassword.setText("Only Numbers Allowed")
        elif len(password) > 6:
            self.ui.lePassword.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorPassword.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorPassword.setText("Maximum of 6 Digits Allowed")
        else:
            return True
```

```
    def usernamevalidation(self):
        self.ui.leUserName.setStyleSheet("border: 1px solid #2E2E2E")
        self.ui.lblErrorUserName.setText("")
        regex = re.compile("^[a-zA-Z]+([ \\-']{0,1}[a-zA-Z]+){0,2}[.]{0,1}$")
        username = self.ui.leUserName.text()
        if regex.match(username) and len(username) <= 64:
            return True
        if re.search(r"\d", username):
            self.ui.leUserName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorUserName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorUserName.setText("No Numbers Allowed")
        elif username.isspace() or username == "":
            self.ui.leUserName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorUserName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorUserName.setText("Please Enter First Name")
        elif len(username) > 64:
            self.ui.leUserName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorUserName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorUserName.setText("Name exceeded maximum character length")
        else:
```

```
            self.ui.leUserName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorUserName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorUserName.setText("No Special Characters Allowed")

    def fillinputs(self):
        username = removeitemwindow.getselectedItem(1, "twStaff", "staffwindow")
        password = removeitemwindow.getselectedItem(2, "twStaff", "staffwindow")
        manager = removeitemwindow.getselectedItem(3, "twStaff", "staffwindow")
        self.ui.leUserName.setText(f"{username}")
        self.ui.lePassword.setText(f"{password}")
        self.ui.cbManager.setCurrentIndex(int(manager))

    def filltable(self):
        try:
            self.ui.twStaff.setRowCount(10)
            self.ui.twStaff.setColumnCount(4)
            self.ui.twStaff.horizontalHeader().setMinimumSectionSize(178)
            self.ui.twStaff.horizontalHeader().setMaximumSectionSize(178)
            self.ui.twStaff.clear()
            self.ui.twStaff.setHorizontalHeaderLabels(
```

```
            ["AccountID", "User Name", "Password", "Manager"]
        )
        query = "SELECT * FROM Management"
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        cursor.execute(query)
        rows = cursor.fetchall()
        noRow = 0
        for tuple in rows:
            noCol = 0
            for column in tuple:
                data = QTableWidgetItem(str(column))
                self.ui.twStaff.setItem(noRow, noCol, data)
                noCol += 1
            noRow += 1
        self.ui.twStaff.setRowCount(noRow)
        self.ui.twStaff.setColumnCount(noCol)
        cnxn.close()
    except:
        pass
```

```
################################################
class ManagementScreen(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Management()
        self.ui.setupUi(self)
        self.ui.pbEditMenu.clicked.connect(self.editmenu_clicked)
        self.ui.pbStatistics.clicked.connect(self.statistics_clicked)
        self.ui.pbStaff.clicked.connect(self.staff_clicked)

    def editmenu_clicked(self):
        Mainwindow.ui.DisplayWidget.setCurrentIndex(5)

    def statistics_clicked(self):
        Mainwindow.ui.DisplayWidget.setCurrentIndex(10)
        statisticswindow.filltable()

    def staff_clicked(self):
```

```
        Mainwindow.ui.DisplayWidget.setCurrentIndex(11)
        staffwindow.clearinputs()
        staffwindow.filltable()



################################################
class Statistics(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Statistics()
        self.ui.setupUi(self)
        self.ui.cbSort.currentIndexChanged.connect(self.filltable)


    def filltable(self):
        try:
            self.ui.twStatistics.setRowCount(50)
            self.ui.twStatistics.setColumnCount(3)
            self.ui.twStatistics.horizontalHeader().setMinimumSectionSize(332)
            self.ui.twStatistics.horizontalHeader().setMaximumSectionSize(332)
            if self.ui.cbSort.currentIndex() == 0:
```

```
        query = "SELECT MenuItems.OrderID, MenuItems.ItemName, COUNT(TableOrder.OrderID) FROM MenuItems INNER JOIN TableOrder ON MenuItems.OrderID=TableOrder.OrderID GROUP BY MenuItems.OrderID, MenuItems.ItemName ORDER BY COUNT(TableOrder.OrderID) DESC"
    if self.ui.cbSort.currentIndex() == 1:
        query = "SELECT MenuItems.OrderID, MenuItems.ItemName, COUNT(TableOrder.OrderID) FROM MenuItems INNER JOIN TableOrder ON MenuItems.OrderID=TableOrder.OrderID GROUP BY MenuItems.OrderID, MenuItems.ItemName ORDER BY COUNT(TableOrder.OrderID) ASC"

    self.ui.twStatistics.clear()
    self.ui.twStatistics.setHorizontalHeaderLabels(
        ["OrderID", "Item Name", "Amount of Orders"]
    )
    cnxn = pyodbc.connect(cs)
    cursor = cnxn.cursor()
    cursor.execute(query)
    rows = cursor.fetchall()
    noRow = 0
    for tuple in rows:
        noCol = 0
        for column in tuple:
            data = QTableWidgetItem(str(column))
            self.ui.twStatistics.setItem(noRow, noCol, data)
            noCol += 1
```

```
            noRow += 1
        self.ui.twStatistics.setRowCount(noRow)
        self.ui.twStatistics.setColumnCount(noCol)
        cnxn.close()
    except:
        pass



################################################
class EditMenuScreen(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_EditMenu()
        self.ui.setupUi(self)
        self.ui.pbRemoveItem.clicked.connect(self.remove_clicked)
        self.ui.pbAddItem.clicked.connect(self.add_clicked)
        self.ui.pbAlterItem.clicked.connect(self.alter_clicked)

    def remove_clicked(self):
        removeitemwindow.ui.twMenuItems.clearSelection()
```

```
        removeitemwindow.ui.lblError.setText("")
        Mainwindow.ui.DisplayWidget.setCurrentIndex(6)

    def add_clicked(self):
        Mainwindow.ui.DisplayWidget.setCurrentIndex(7)
        additemwindow.fillmenutable()

    def alter_clicked(self):
        Mainwindow.ui.DisplayWidget.setCurrentIndex(8)



################################################
class AddConfirmation(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_AddConfirmation()
        self.ui.setupUi(self)
        self.ui.pbAdd.clicked.connect(self.addconfirm_clicked)
        self.ui.pbCancel.clicked.connect(self.cancel_clicked)
```

```
    def cancel_clicked(self):
        addconfirmationscreen.close()


    def addconfirm_clicked(self):
        itemname = additemwindow.ui.leItemname.text()
        category = additemwindow.ui.cbCategory.currentIndex()
        price = additemwindow.ui.sbPrice.value()
        orderid = self.getOrderID()
        if category == 1:
            starterspage.printitembuttons(category, "starterspage")
            starterspage.hideitembuttons(category, "starterspage")
        elif category == 2:
            starterspage.printitembuttons(category, "mainspage")
            starterspage.hideitembuttons(category, "mainspage")
        elif category == 3:
            starterspage.printitembuttons(category, "dessertspage")
            starterspage.hideitembuttons(category, "dessertspage")
        elif category == 4:
            starterspage.printitembuttons(category, "drinkspage")
            starterspage.hideitembuttons(category, "drinkspage")
```

```
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = f"INSERT INTO MenuItems (MenuItems.OrderID,MenuItems.ItemName, MenuItems.CategoryID, MenuItems.Price) VALUES ({orderid},'{itemname}',{category},{price});"
        cursor.execute(query)
        cursor.commit()
        cursor.close()
        addconfirmationscreen.close()
        additemwindow.fillmenutable()
        additemwindow.clearinputs()


    def getOrderID(self):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        orderidquery = "SELECT MAX(OrderID) FROM MenuItems;"
        cursor.execute(orderidquery)
        orderid = cursor.fetchone()
        if orderid[0] == None:
            orderid = 1
        else:
```

```
        orderid = orderid[0] + 1
    return orderid



################################################
class AlterMenuItem(QDialog):
  def __init__(self):
    super(QDialog, self).__init__()
    self.ui = Ui_AlterMenuItem()
    self.ui.setupUi(self)
    self.ui.pbBack.clicked.connect(additemwindow.back_clicked)
    self.fillitemmenutable()
    self.ui.twMenuItems.doubleClicked.connect(self.alteritemdetails)

  def alteritemdetails(self):
    alterconfirmation.close()
    alterconfirmation.fillinputs()
    alterconfirmation.show()

  def fillitemmenutable(self):
```

```
        try:
            self.ui.twMenuItems.setRowCount(50)
            self.ui.twMenuItems.setColumnCount(4)
            self.ui.twMenuItems.horizontalHeader().setMinimumSectionSize(251)
            self.ui.twMenuItems.horizontalHeader().setMaximumSectionSize(251)


            self.ui.twMenuItems.clear()
            self.ui.twMenuItems.setHorizontalHeaderLabels(
                ["OrderID", "Item Name", "Category", "Price"]
            )
            cnxn = pyodbc.connect(cs)
            cursor = cnxn.cursor()
            query = "SELECT MenuItems.OrderID, MenuItems.ItemName, Category.CategoryName, FORMAT(MenuItems.Price, 'N2') FROM MenuItems INNER
JOIN Category ON MenuItems.CategoryID = Category.CategoryID"
            cursor.execute(query)
            rows = cursor.fetchall()
            noRow = 0
            for tuple in rows:
                noCol = 0
                for column in tuple:
```

```
                data = QTableWidgetItem(str(column))
                self.ui.twMenuItems.setItem(noRow, noCol, data)
                noCol += 1
            noRow += 1
        self.ui.twMenuItems.setRowCount(noRow)
        self.ui.twMenuItems.setColumnCount(noCol)
        cnxn.close()
    except:
        pass



################################################
class AlterConfirmation(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_AlterConfirmation()
        self.ui.setupUi(self)
        self.ui.pbAlter.clicked.connect(self.fieldvalidation)
        self.ui.pbCancel.clicked.connect(self.cancel_clicked)
```

```
    def fillinputs(self):
        itemname = removeitemwindow.getselectedItem(1, "twMenuItems", "alteritemwindow")
        category = removeitemwindow.getselectedItem(2, "twMenuItems", "alteritemwindow")
        if category == "Starter":
            category = 0
        elif category == "Main":
            category = 1
        elif category == "Dessert":
            category = 2
        else:
            category = 3
        price = removeitemwindow.getselectedItem(3, "twMenuItems", "alteritemwindow")
        self.ui.leItemname.setText(itemname)
        self.ui.cbCategory.setCurrentIndex(category)
        self.ui.sbPrice.setValue(float(price))

    def fieldvalidation(self):
        namevalidation = additemwindow.itemnamevalidation("alterconfirmation")
        if namevalidation:
            self.alteritem()
```

```
    def alteritem(self):
        orderid = removeitemwindow.getselectedItem(0, "twMenuItems", "alteritemwindow")
        itemname = self.ui.leItemname.text()
        category = self.ui.cbCategory.currentIndex()
        price = self.ui.sbPrice.value()
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        updateitem = f"UPDATE MenuItems SET ItemName = '{itemname}', CategoryID = {category+1}, Price = {price} WHERE OrderID = {orderid};"
        cursor.execute(updateitem)
        cursor.commit()
        cnxn.close()
        alteritemwindow.fillitemmenutable()
        alterconfirmation.close()

    def cancel_clicked(self):
        alterconfirmation.close()
```

##

```
class AddItemScreen(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_AddMenuItem()
        self.ui.setupUi(self)
        self.ui.pbConfirm.clicked.connect(self.fieldvalidation)
        self.ui.pbBack.clicked.connect(self.back_clicked)

    def back_clicked(self):
        Mainwindow.ui.DisplayWidget.setCurrentIndex(5)

    def clearinputs(self):
        self.ui.leItemname.clear()
        self.ui.cbCategory.setCurrentIndex(0)
        self.ui.sbPrice.setValue(9.99)

    def itemnamevalidation(self, window):
        itemname = eval(f"{window}.ui.leItemname.text()")
        exec(f'{window}.ui.leItemname.setStyleSheet("border: 1px solid #2e2e2e;")')
        exec(f'{window}.ui.lblErrorItemName.setText("")')
```

```
    regex = re.compile("^[A-Za-z0-9 _]*[A-Za-z0-9][A-Za-z0-9 _]*$")
    if itemname == "" or itemname.isspace():
        exec(
            f'{window}.ui.leItemname.setStyleSheet("border: 1px solid rgb(170, 0, 0);")'
        )
        exec(f'{window}.ui.lblErrorItemName.setStyleSheet("color:rgb(170, 0, 0);")')
        exec(f'{window}.ui.lblErrorItemName.setText("Please Enter an Item Name")')
    elif regex.match(itemname) and len(itemname) <= 64:
        return True
    elif len(itemname) > 64:
        exec(
            f'{window}.ui.leItemname.setStyleSheet("border: 1px solid rgb(170, 0, 0);")'
        )
        exec(f'{window}.ui.lblErrorItemName.setStyleSheet("color:rgb(170, 0, 0);")')
        exec(
            f'{window}.ui.lblErrorItemName.setText("Item Name exceeded maximum character length")'
        )
    else:
        exec(
            f'{window}.ui.leItemname.setStyleSheet("border: 1px solid rgb(170, 0, 0);")'
```

```
            )
            exec(f'{window}.ui.lblErrorItemName.setStyleSheet("color:rgb(170, 0, 0);")')
            exec(
                f'{window}.ui.lblErrorItemName.setText("No Special Characters Allowed")'
            )


    def categoryvalidation(self, window):
        exec(f'{window}.ui.cbCategory.setStyleSheet("border: 1px solid #2e2e2e;")')
        exec(f'{window}.ui.lblErrorCategory.setText("")')
        category = eval(f"{window}.ui.cbCategory.currentText()")
        if category == "Category:":
            exec(
                f'{window}.ui.cbCategory.setStyleSheet("border: 1px solid rgb(170, 0, 0);")'
            )
            exec(f'{window}.ui.lblErrorCategory.setStyleSheet("color:rgb(170, 0, 0);")')
            exec(f'{window}.ui.lblErrorCategory.setText("Please select a Category")')
        else:
            return True


    def fieldvalidation(self):
```

```
        namevalidation = self.itemnamevalidation("additemwindow")
        categoryvalidation = self.categoryvalidation("additemwindow")
        if namevalidation == True and categoryvalidation == True:
            self.addconfirmation()


    def addconfirmation(self):
        itemname = self.ui.leItemname.text()
        category = self.ui.cbCategory.currentText()
        price = self.ui.sbPrice.value()
        addconfirmationscreen.ui.lblConfirmation.setText(
            f"""Are you sure you want to add the item:
Item Name: {itemname}
Category: {category}
Price: £{price}"""
        )
        addconfirmationscreen.show()


    def fillmenutable(self):
        try:
            self.ui.twMenuItems.setRowCount(50)
```

```
        self.ui.twMenuItems.setColumnCount(4)
        self.ui.twMenuItems.horizontalHeader().setMinimumSectionSize(183)
        self.ui.twMenuItems.horizontalHeader().setMaximumSectionSize(183)
        self.ui.twMenuItems.verticalHeader().setMinimumSectionSize(5)
        self.ui.twMenuItems.verticalHeader().setMaximumSectionSize(5)


        self.ui.twMenuItems.clear()
        self.ui.twMenuItems.setHorizontalHeaderLabels(
            ["OrderID", "Item Name", "Category", "Price"]
        )
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = "SELECT MenuItems.OrderID, MenuItems.ItemName, Category.CategoryName, FORMAT(MenuItems.Price, 'N2') FROM MenuItems INNER
JOIN Category ON MenuItems.CategoryID = Category.CategoryID"
        cursor.execute(query)
        rows = cursor.fetchall()
        noRow = 0
        for tuple in rows:
            noCol = 0
            for column in tuple:
```

```
                data = QTableWidgetItem(str(column))
                self.ui.twMenuItems.setItem(noRow, noCol, data)
                noCol += 1
            noRow += 1
        self.ui.twMenuItems.setRowCount(noRow)
        self.ui.twMenuItems.setColumnCount(noCol)
        cnxn.close()
    except:
        pass



################################################
class RemoveItemScreen(QDialog):
  def __init__(self):
    super(QDialog, self).__init__()
    self.ui = Ui_RemoveMenuItem()
    self.ui.setupUi(self)
    self.fillmenutable()
    self.ui.pbRemoveItem.clicked.connect(self.remove_clicked)
    self.ui.pbBack.clicked.connect(self.back_clicked)
```

```
    def back_clicked(self):
        Mainwindow.ui.DisplayWidget.setCurrentIndex(5)


    def remove_clicked(self):
        self.ui.lblError.setText("")
        if self.ui.twMenuItems.selectedItems() != []:


            itemname = self.getselectedItem(1, "twMenuItems", "removeitemwindow")
            removeconfirmationscreen.ui.lblConfirmation.setText(
                "Are you sure you want to delete the item '{0}'?".format(itemname)
            )
            removeconfirmationscreen.show()
        else:
            self.ui.lblError.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblError.setText("Please select an Item to Remove")


    def getselectedItem(self, header, table, window):
        rowindex = eval(f"{window}.ui.{table}.selectionModel().currentIndex()")
        itemname = rowindex.sibling(rowindex.row(), header).data()
```

```
        return itemname


    def fillmenutable(self):
        try:
            self.ui.twMenuItems.setRowCount(50)
            self.ui.twMenuItems.setColumnCount(4)
            self.ui.twMenuItems.horizontalHeader().setMinimumSectionSize(251)
            self.ui.twMenuItems.horizontalHeader().setMaximumSectionSize(251)


            self.ui.twMenuItems.clear()
            self.ui.twMenuItems.setHorizontalHeaderLabels(
                ["OrderID", "Item Name", "Category", "Price"]
            )
            cnxn = pyodbc.connect(cs)
            cursor = cnxn.cursor()
            query = "SELECT MenuItems.OrderID, MenuItems.ItemName, Category.CategoryName, FORMAT(MenuItems.Price, 'N2') FROM MenuItems INNER
JOIN Category ON MenuItems.CategoryID = Category.CategoryID"
            cursor.execute(query)
            rows = cursor.fetchall()
            noRow = 0
```

```
        for tuple in rows:
            noCol = 0
            for column in tuple:
                data = QTableWidgetItem(str(column))
                self.ui.twMenuItems.setItem(noRow, noCol, data)
                noCol += 1
            noRow += 1
        self.ui.twMenuItems.setRowCount(noRow)
        self.ui.twMenuItems.setColumnCount(noCol)
        cnxn.close()
    except:
        pass



################################################
class CloseTableConfirmation(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_CloseConfirmation()
        self.ui.setupUi(self)
```

```
        self.ui.pbClose.clicked.connect(self.remove_clicked)
        self.ui.pbCancel.clicked.connect(self.cancel_clicked)


    def cancel_clicked(self):
        closeconfirmation.close()


    def remove_clicked(self):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = f"UPDATE Sitting SET Active = 'FALSE' FROM Sitting INNER JOIN TableOrder ON Sitting.SittingID=TableOrder.SittingID WHERE TableOrder.TableID = {orderwindow.tableid} AND Sitting.Active = 'TRUE';"
        cursor.execute(query)
        cursor.commit()
        cnxn.close()
        tablecapacity = tableselectionwindow.gettablecapacity(orderwindow.tableid)
        stylesheet = f"background-image: url(:/Tables/TableIcon{tablecapacity}Unavailable.png);background-position: center; background-repeat: no-repeat; background-color: transparent;border: 1px solid transparent;"
        exec(
            f'seatingwindow.pushButton{orderwindow.tableid}.setStyleSheet("{stylesheet}")'
        )
        Mainwindow.ui.DisplayWidget.setCurrentIndex(2)
```

```
        closeconfirmation.close()



################################################
class OrderScreen(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_OrderScreen()
        self.ui.setupUi(self)
        self.ui.pbExit.clicked.connect(self.exitorderscreen)
        self.ui.pbMain.clicked.connect(self.mainmenuscreen)
        self.ui.pbDesserts.clicked.connect(self.dessertmenuscreen)
        self.ui.pbDrinks.clicked.connect(self.drinkmenuscreen)
        self.ui.pbStarter.clicked.connect(self.startermenuscreen)
        self.ui.pbConfirm.clicked.connect(self.addToOrder)
        self.ui.pbConfirm.clicked.connect(self.outputordereditems)
        self.ui.pbClose.clicked.connect(self.closetable)
        self.ui.twOrders.doubleClicked.connect(self.alteritem)

    def alteritem(self):
```

```
        changequantitywindow.closewindow()
        itemname = self.getselecteditem()
        changequantitywindow.ui.lblItem.setText(
            f"Enter the quantity of {itemname} for Table {self.tableid}:"
        )
        changequantitywindow.show()
        self.getOrderID()

    def getselecteditem(self):
        itemname = removeitemwindow.getselectedItem(0, "twOrders", "orderwindow")
        return itemname

    def getOrderID(self):
        itemname = self.getselecteditem()
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = f"SELECT OrderID FROM MenuItems WHERE ItemName = '{itemname}';"
        cursor.execute(query)
        OrderID = cursor.fetchone()
        OrderID = OrderID[0]
```

```
        cnxn.close()
        return OrderID

    def closetable(self):
        closeconfirmation.ui.lblConfirmation.setText(
            f"Are you sure you would like to close Table {self.tableid}?"
        )
        closeconfirmation.show()

    def exitorderscreen(self):
        seatingwindow.setactivetableimages()
        Mainwindow.ui.DisplayWidget.setCurrentIndex(2)

    def numofitemsincategory(self, category):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = "SELECT COUNT(*) FROM MenuItems WHERE CategoryID = '{0}';".format(
            category
        )
        cursor.execute(query)
```

```
        numofitemsincategory = cursor.fetchone()
        numofitemsincategory = numofitemsincategory[0]
        cnxn.close()
        return numofitemsincategory

    def getmenuitems(self, category):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = "SELECT ItemName FROM MenuItems WHERE CategoryID = '{0}';".format(
            category
        )
        cursor.execute(query)
        menuitemsincategory = cursor.fetchall()
        return menuitemsincategory

    def gettotalprice(self):
        totalprice = 0
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
```

```
        query = f"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'"
        cursor.execute(query)
        menuitemsprice = cursor.fetchall()
        for i in range(0, len(menuitemsprice)):
            totalprice = totalprice + menuitemsprice[i][0]
        self.ui.lblTotal.setText(f"Total: £{round(totalprice,2)}")
        cnxn.close()

    def getordereditems(self):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = f"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'"
        cursor.execute(query)
        ordereditems = cursor.fetchall()
        cnxn.close()
        return ordereditems

    def getitemquantity(self, item):
```

```
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = f"SELECT COUNT(*) FROM TableOrder INNER JOIN MenuItems ON MenuItems.OrderID=TableOrder.OrderID INNER JOIN Sitting ON Sitting.SittingID=TableOrder.SittingID WHERE TableOrder.TableID = {self.tableid} AND Sitting.Active='TRUE' AND MenuItems.ItemName ='{item}'"
        cursor.execute(query)
        itemquantity = cursor.fetchone()
        cnxn.close()
        return itemquantity

    def getitemprice(self, item):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = f"SELECT Price FROM MenuItems WHERE ItemName = '{item}';"
        cursor.execute(query)
        price = cursor.fetchone()[0]
        cnxn.close()
        return price

    def outputordereditems(self):
        self.gettotalprice()
```

```
        self.clearorderwidget()
        self.orderlist = []
        ordereditems = self.getordereditems()
        for i in range(0, len(ordereditems)):
            itemquantity = (self.getitemquantity(ordereditems[i][0]))[0]
            ordereditem = ordereditems[i][0]
            price = round(ordereditems[i][1], 2)
            self.addrowtotable(ordereditem, itemquantity, price)

    def addrowtotable(self, item, itemquantity, price):
        self.ui.twOrders.setColumnWidth(0, 180)
        rowPosition = self.ui.twOrders.rowCount()
        self.ui.twOrders.insertRow(rowPosition)
        self.ui.twOrders.setItem(rowPosition, 0, QTableWidgetItem(f"{item}"))
        self.ui.twOrders.setItem(rowPosition, 1, QTableWidgetItem(f"{itemquantity}"))
        self.ui.twOrders.setItem(
            rowPosition, 2, QTableWidgetItem(f"£ {(price*itemquantity):.2f}")
        )

    def clearorderwidget(self):
```

```
        self.ui.twOrders.setRowCount(0)


    def getitemorderid(self, item):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = f"SELECT OrderID FROM MenuItems WHERE ItemName = '{item}'"
        cursor.execute(query)
        OrderID = cursor.fetchone()
        OrderID = OrderID[0]
        cnxn.close()
        return OrderID


    def gettableid(self, tablenum):
        self.orderlist = []
        self.tableid = tablenum


    def addtoorderlist(self, item):
        orderid = self.getitemorderid(item)
        self.orderlist.append(orderid)
        price = round(self.getitemprice(item), 2)
```

```
        self.addrowtotable(item, 1, price)


    def addToOrder(self):
        sittingid = self.getSittingID()
        for i in range(0, len(self.orderlist)):
            tableorderid = self.getTableOrderID()
            cnxn = pyodbc.connect(cs)
            cursor = cnxn.cursor()
            query = f"INSERT INTO TableOrder (TableOrder.TableOrderID, TableOrder.TableID, TableOrder.OrderID, TableOrder.SittingID) VALUES
({tableorderid},{self.tableid},{self.orderlist[i]},{sittingid});"
            cursor.execute(query)
            cursor.commit()
            cnxn.close()


    def getTableOrderID(self):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = f"SELECT MAX(TableOrderID) FROM TableOrder;"
        cursor.execute(query)
        largesttableorderid = cursor.fetchone()[0]
```

```
        if largesttableorderid == None:
            tableorderid = 1
        else:
            tableorderid = largesttableorderid + 1
        return tableorderid


    def getSittingID(self):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = f"SELECT Sitting.SittingID FROM TableOrder INNER JOIN Sitting ON TableOrder.SittingID=Sitting.SittingID WHERE Sitting.Active = 'TRUE' AND TableOrder.TableID = {self.tableid};"
        cursor.execute(query)
        activesittingid = cursor.fetchone()
        if activesittingid == None:
            query = "SELECT MAX(SittingID) FROM Sitting;"
            cursor.execute(query)
            sittingid = cursor.fetchone()
            if sittingid[0] == None:
                sittingid = 1
            else:
```

```
            sittingid = sittingid[0] + 1
        query = f"INSERT INTO Sitting (Sitting.SittingID,Sitting.Active) VALUES ({sittingid},'TRUE');"
        cursor.execute(query)
        cursor.commit()
    else:
        sittingid = activesittingid[0]
    cnxn.close()
    return sittingid

def mainmenuscreen(self):
    orderwindow.ui.orderwidget.setCurrentIndex(1)
    starterspage.printitembuttons(2, "mainspage")

def startermenuscreen(self):
    orderwindow.ui.orderwidget.setCurrentIndex(0)
    starterspage.printitembuttons(1, "starterspage")

def dessertmenuscreen(self):
    orderwindow.ui.orderwidget.setCurrentIndex(2)
    starterspage.printitembuttons(3, "dessertspage")
```

```
    def drinkmenuscreen(self):
        orderwindow.ui.orderwidget.setCurrentIndex(3)
        starterspage.printitembuttons(4, "drinkspage")



################################################
class ChangeQuantity(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_ChangeQuantity()
        self.ui.setupUi(self)
        self.ui.pbChange.clicked.connect(self.change_clicked)
        self.ui.pbCancel.clicked.connect(self.closewindow)

    def closewindow(self):
        changequantitywindow.close()

    def change_clicked(self):
        self.clearitemrecords()
```

```
        sittingid = orderwindow.getSittingID()
        orderid = orderwindow.getOrderID()
        itemquantity = self.ui.sbQuantity.value()
        for i in range(0, itemquantity):
            tableorderid = orderwindow.getTableOrderID()
            cnxn = pyodbc.connect(cs)
            cursor = cnxn.cursor()
            query = f"INSERT INTO TableOrder (TableOrder.TableOrderID, TableOrder.TableID, TableOrder.OrderID, TableOrder.SittingID) VALUES
({tableorderid},{orderwindow.tableid},{orderid},{sittingid});"
            cursor.execute(query)
            cursor.commit()
            cnxn.close()
        orderwindow.outputordereditems()
        self.closewindow()

    def clearitemrecords(self):
        sittingid = orderwindow.getSittingID()
        orderid = orderwindow.getOrderID()
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
```

```
    query = f"DELETE FROM TableOrder WHERE TableOrder.TableID = {orderwindow.tableid} AND TableOrder.SittingID = {sittingid} AND TableOrder.OrderID = {orderid}"

    cursor.execute(query)

    cursor.commit()

    cnxn.close()



################################################
class StartersPage(QDialog):
  def __init__(self):
    super(QDialog, self).__init__()
    self.ui = Ui_StartersPage()
    self.ui.setupUi(self)


  def hideitembuttons(self, category, window):
    numofitemsincategory = orderwindow.numofitemsincategory(category)
    menuitemsincategory = orderwindow.getmenuitems(category)
    for i in range(1, numofitemsincategory + 1):
      exec("{1}.pushButton{0}.hide()".format(i, window))
```

```
    def printitembuttons(self, category, window):
        try:
            self.hideitembuttons(category, window)
        except:
            pass
        numofitemsincategory = orderwindow.numofitemsincategory(category)
        menuitemsincategory = orderwindow.getmenuitems(category)
        xcoordinate = 20
        for i in range(1, numofitemsincategory + 1):
            if i % 4 == 1:
                xcoordinate = 20
            y = 0 + 51 * ((i - 1) // 4)
            exec("{1}.pushButton{0} = QtWidgets.QPushButton({1})".format(i, window))
            exec(
                "{2}.pushButton{0}.setGeometry(QtCore.QRect({1}, {3}, 151, 51))".format(
                    i, xcoordinate, window, y
                )
            )
            exec("{1}.pushButton{0}.setEnabled(True)".format(i, window))
            exec(
```

```
            "{2}.pushButton{0}.setText('{1}')".format(
                i, menuitemsincategory[i - 1][0], window
            )
        )
        exec("{1}.pushButton{0}.setIconSize(QtCore.QSize(9, 8))".format(i, window))
        exec("{1}.pushButton{0}.setObjectName('Button{0}')".format(i, window))
        exec(
            "{2}.pushButton{0}.clicked.connect(lambda: orderwindow.addtoorderlist(str('{1}')))".format(
                i, menuitemsincategory[i - 1][0], window
            )
        )
        exec("{1}.pushButton{0}.show()".format(i, window))
        xcoordinate += 151



################################################
class MainsPage(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_MainsPage()
```

```
        self.ui.setupUi(self)



################################################
class DessertsPage(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_DessertsPage()
        self.ui.setupUi(self)



################################################
class DrinksPage(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_DrinksPage()
        self.ui.setupUi(self)



################################################
```

```
class DateSelection(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Date()
        self.ui.setupUi(self)
        self.ui.cwCalendar.clicked.connect(self.getdate)
        self.ui.cwCalendar.clicked.connect(self.enablesizebutton)
        self.setminimumdate()
        self.setmaximumdate()

    def setminimumdate(self):
        today = date.today()
        self.ui.cwCalendar.setMinimumDate(today)
        self.ui.cwCalendar.setSelectedDate(today)

    def setmaximumdate(self):
        today = date.today()
        maximum = today.replace(year=today.year + 1)
        self.ui.cwCalendar.setMaximumDate(maximum)
```

```
    def enablesizebutton(self):
        exec("bookingwindow.ui.pbSize.setEnabled(True)")

    def getdate(self):
        date = self.ui.cwCalendar.selectedDate()
        date = date.toPyDate()
        return date

################################################
class PartySizeSelection(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_PartySize()
        self.ui.setupUi(self)
        self.ui.lwSize.itemClicked.connect(self.enabletimebutton)
        self.ui.cbCustom.stateChanged.connect(self.checkstate)

    def enabletimebutton(self):
        tableselectionwindow.cleartables()
```

```
        exec("bookingwindow.ui.pbTime.setEnabled(True)")


    def checkstate(self):
        if self.ui.cbCustom.isChecked():
            self.enabletimebutton()
        else:
            if self.ui.lwSize.currentRow() == -1:
                exec("bookingwindow.ui.pbTime.setEnabled(False)")


    def getpartysize(self):
        if self.ui.cbCustom.isChecked():
            partysize = self.ui.sbCustom.value()
        else:
            row = self.ui.lwSize.currentRow()
            partysize = self.ui.lwSize.item(row).text()
            partysize = int(re.search(r"\d+", partysize).group())
        tableselectionwindow.partysize = partysize
        return partysize


    def clearsizeselection(self):
```

```
        self.ui.cbCustom.setCheckState(0)
        self.ui.lwSize.setCurrentRow(-1)



################################################
class TimeSelection(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Time()
        self.ui.setupUi(self)
        self.ui.lwTime.itemClicked.connect(self.gettime)
        self.ui.lwTime.itemClicked.connect(self.enabletablebutton)

    def enabletablebutton(self):
        exec("bookingwindow.ui.pbTable.setEnabled(True)")

    def gettime(self):
        row = self.ui.lwTime.currentRow()
        bookingtime = self.ui.lwTime.item(row).text()
        return bookingtime
```

```
    def converttime(self):
        bookingtime = self.gettime()
        (hours, minutes) = bookingtime.split(":")
        convertedtime = timedelta(hours=int(hours), minutes=int(minutes))
        return convertedtime

    def cleartimeselection(self):
        self.ui.lwTime.setCurrentRow(-1)



################################################
class TableSelection(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Table()
        self.ui.setupUi(self)
        self.outputtables()

    def enableguestbutton(self, tablenum):
```

```
        tablecapacity = int(self.gettablecapacity(tablenum))
        if self.seated < self.partysize:
            exec("bookingwindow.ui.pbGuest.setEnabled(False)")
        else:
            exec("bookingwindow.ui.pbGuest.setEnabled(True)")


    def unavailabletables(self):
        date = dateselectionwindow.getdate()
        bookingtime = timeselectionwindow.converttime()
        unavailabletables = []
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        for i in range(0, 5):
            query = "SELECT TableReservation.TableID FROM TableReservation INNER JOIN Reservation ON
TableReservation.ReservationID=Reservation.ReservationID WHERE (Reservation.Time = '{0}') AND (Reservation.Date = '{1}')".format(
                bookingtime - timedelta(minutes=(i * 30)), date
            )
            cnxn = pyodbc.connect(cs)
            cursor = cnxn.cursor()
            cursor.execute(query)
```

```
            rows = cursor.fetchall()
            cnxn.close()
            for i in range(0, len(rows)):
                unavailabletables.append(rows[i][0])
        for i in range(1, 3):
            query = "SELECT TableReservation.TableID FROM TableReservation INNER JOIN Reservation ON
TableReservation.ReservationID=Reservation.ReservationID WHERE (Reservation.Time = '{0}') AND (Reservation.Date = '{1}')".format(
                bookingtime + timedelta(minutes=(i * 30)), date
            )
            cnxn = pyodbc.connect(cs)
            cursor = cnxn.cursor()
            cursor.execute(query)
            rows = cursor.fetchall()
            cnxn.close()
            for i in range(0, len(rows)):
                unavailabletables.append(rows[i][0])
        return unavailabletables


    def cleartables(self):
        exec("bookingwindow.ui.pbGuest.setEnabled(False)")
```

```
        self.seated = 0
        self.selectedtables = []
        for i in range(1, 16):
            exec("self.checkBox{0}.setCheckState(0)".format(i))
            exec("self.checkBox{0}.setEnabled(True)".format(i))


    def disabletables(self):
        self.cleartables()
        unavailabletables = self.unavailabletables()
        for i in range(0, len(unavailabletables)):
            exec("self.checkBox{0}.setEnabled(False)".format(unavailabletables[i]))


    def outputtables(self):
        tablenum = 0
        t = 20
        for n in range(1, 4):
            columncoordinate = 270
            for i in range(1, 3):
                tablenum += 1
                exec("self.checkBox{0} = QtWidgets.QCheckBox(self)".format(tablenum))
```

```
            exec(
                "self.checkBox{0}.setGeometry(QtCore.QRect({1}, {2}, 71, 71))".format(
                    tablenum, columncoordinate, t
                )
            )
            exec("self.checkBox{0}.setEnabled(True)".format(tablenum))
            exec("self.checkBox{0}.setText('{0}')".format(tablenum))
            exec(
                "self.checkBox{0}.setIconSize(QtCore.QSize(9, 8))".format(tablenum)
            )
            exec("self.checkBox{0}.setObjectName('Table{0}')".format(tablenum))
            exec(
                "self.checkBox{0}.stateChanged.connect(lambda: tableselectionwindow.getselectedtables(str({0})))".format(
                    tablenum
                )
            )
            exec(
                "self.checkBox{0}.stateChanged.connect(lambda: tableselectionwindow.enableguestbutton(str({0})))".format(
                    tablenum
                )
```

```
            )


            columncoordinate = 730
        t = t + 75
    for k in range(1, 4):
        columncoordinate = 270
        for j in range(1, 4):
            tablenum += 1
            exec("self.checkBox{0} = QtWidgets.QCheckBox(self)".format(tablenum))
            exec(
                "self.checkBox{0}.setGeometry(QtCore.QRect({1}, {2}, 71, 71))".format(
                    tablenum, columncoordinate, t
                )
            )
            exec("self.checkBox{0}.setEnabled(True)".format(tablenum))
            exec("self.checkBox{0}.setText('{0}')".format(tablenum))
            exec(
                "self.checkBox{0}.setIconSize(QtCore.QSize(9, 8))".format(tablenum)
            )
            exec("self.checkBox{0}.setObjectName('Table{0}')".format(tablenum))
```

```
            exec(
                "self.checkBox{0}.stateChanged.connect(lambda: tableselectionwindow.getselectedtables(str({0})))".format(
                    tablenum
                )
            )
            exec(
                "self.checkBox{0}.stateChanged.connect(lambda: tableselectionwindow.enableguestbutton(str({0})))".format(
                    tablenum
                )
            )
            columncoordinate = columncoordinate + 230
        t = t + 75


    def gettablecapacity(self, tablenum):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = "SELECT Capacity FROM [Table] WHERE TableID = '{0}';".format(tablenum)
        cursor.execute(query)
        tablecapacity = cursor.fetchone()[0]
        cnxn.close()
```

```
        return tablecapacity


    def updateguestsleft(self):
        if self.seated > self.partysize:
            self.ui.lblGuestsLeft.setText(
                f"{self.partysize} Guests Seated out of {self.partysize}"
            )
        else:
            self.ui.lblGuestsLeft.setText(
                f"{self.seated} Guests Seated out of {self.partysize}"
            )


    def getselectedtables(self, tablenum):
        tablecapacity = int(self.gettablecapacity(tablenum))
        if eval("self.checkBox{0}.checkState()".format(tablenum)) == 2:
            verificationresult = self.sizeverification(tablenum)
            if verificationresult:
                self.selectedtables.append(tablenum)
                self.seated = self.seated + tablecapacity
            else:
```

```
            exec("self.checkBox{0}.setCheckState(0)".format(tablenum))
        if (
            eval("self.checkBox{0}.checkState()".format(tablenum)) == 0
            and self.selectedtables.count(tablenum) == 1
        ):
            self.selectedtables.remove(tablenum)
            self.seated = self.seated - tablecapacity
        else:
            pass
        self.updateguestsleft()


    def sizeverification(self, tablenum):
        tablecapacity = int(self.gettablecapacity(tablenum))
        if self.seated < self.partysize:
            if (tablecapacity - 2) > (self.partysize - self.seated):
                exec("self.checkBox{0}.setCheckState(0)".format(tablenum))
            elif (self.partysize < 6) and (self.partysize > tablecapacity):
                exec("self.checkBox{0}.setCheckState(0)".format(tablenum))
            else:
                return True
```

```
        else:
            exec("self.checkBox{0}.setCheckState(0)".format(tablenum))



################################################
class GuestDetails(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Guest()
        self.ui.setupUi(self)
        self.ui.pbProceed.clicked.connect(self.validatefields)

    def displayreservationdetails(self):
        date = dateselectionwindow.getdate()
        partysize = partysizewindow.getpartysize()
        time = timeselectionwindow.gettime()
        tables = tableselectionwindow.selectedtables
        self.ui.lblDateInsert.setText(f"{date}")
        self.ui.lblPartyInsert.setText(f"{partysize}")
        self.ui.lblTimeInsert.setText(f"{time}")
```

```
        self.ui.lblTableInsert.setText(f"{', '.join(tables)}")


    def firstnamevalidation(self):
        self.ui.leFName.setStyleSheet("border: 1px solid #2E2E2E")
        self.ui.lblErrorFName.setText("")
        regex = re.compile("^[a-zA-Z]+([ \\-']{0,1}[a-zA-Z]+){0,2}[.]{0,1}$")
        firstname = self.ui.leFName.text()
        print(len(firstname))
        if regex.match(firstname) and len(firstname) <= 64:
            return True
        if re.search(r"\d", firstname):
            self.ui.leFName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorFName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorFName.setText("No Numbers Allowed")
        elif firstname.isspace() or firstname == "":
            self.ui.leFName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorFName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorFName.setText("Please Enter First Name")
        elif len(firstname) > 64:
            self.ui.leFName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
```

```
            self.ui.lblErrorFName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorFName.setText(
                "First Name exceeded maximum character length"
            )
        else:
            self.ui.leFName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorFName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorFName.setText("No Special Characters Allowed")


    def lastnamevalidation(self):
        self.ui.leLName.setStyleSheet("border: 1px solid #2E2E2E")
        self.ui.lblErrorLName.setText("")
        regex = re.compile("^[a-zA-Z]+([ \\-']{0,1}[a-zA-Z]+){0,2}[.]{0,1}$")
        lastname = self.ui.leLName.text()
        if regex.match(lastname) and len(lastname) <= 64:
            return True
        if re.search(r"\d", lastname):
            self.ui.leLName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorLName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorLName.setText("No Numbers Allowed")
```

```
        elif lastname.isspace() or lastname == "":
            self.ui.leLName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorLName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorLName.setText("Please Enter Last Name")
        elif len(lastname) > 64:
            self.ui.leLName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorLName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorLName.setText("Last Name exceeded maximum character length")
        else:
            self.ui.leLName.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorLName.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorLName.setText("No Special Characters Allowed")


    def emailvalidation(self):
        self.ui.leEmail.setStyleSheet("border: 1px solid #2E2E2E")
        self.ui.lblErrorEmail.setText("")
        regex = re.compile(
            r"([A-Za-z0-9]+[.-_])*[A-Za-z0-9]+@[A-Za-z0-9-]+(\.[A-Z|a-z]{2,})+"
        )
        email = self.ui.leEmail.text()
```

```
        if regex.fullmatch(email):
            return True
        elif email.isspace() or email == "":
            self.ui.leEmail.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorEmail.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorEmail.setText("Please Enter an Email Address")

        else:
            self.ui.leEmail.setStyleSheet("border: 1px solid rgb(170, 0, 0);")
            self.ui.lblErrorEmail.setStyleSheet("color:rgb(170, 0, 0);")
            self.ui.lblErrorEmail.setText("Please Enter a Valid Email Address")

    def validatefields(self):
        fnamevalid = self.firstnamevalidation()
        lnamevalid = self.lastnamevalidation()
        emailvalid = self.emailvalidation()
        if fnamevalid == True and lnamevalid == True and emailvalid == True:
            self.createbooking()

    def getcustomerid(self):
```

```
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        if self.customercheck((self.ui.leEmail.text()).lower()) == True:
            indexcust = f"SELECT CustomerID FROM Customer WHERE EmailAddress = '{(self.ui.leEmail.text()).lower()}';"
            cursor.execute(indexcust)
            numofcustomer = cursor.fetchone()
            customerid = numofcustomer[0]
            cnxn.close()
            return customerid
        else:
            indexcust = "SELECT MAX(CustomerID) FROM Customer;"
            cursor.execute(indexcust)
            numofcustomer = cursor.fetchone()
            customerid = numofcustomer[0] + 1
            cnxn.close()
            return customerid


    def createbooking(self):
        date = dateselectionwindow.getdate()
        partysize = partysizewindow.getpartysize()
```

```
        bookingtime = timeselectionwindow.gettime()

        customerid = self.getcustomerid()

        reservationid = self.getreservationid()

        firstname = self.ui.leFName.text()

        lastname = self.ui.leLName.text()

        email = (self.ui.leEmail.text()).lower()

        accountid = loginpage.accountid

        tables = tableselectionwindow.selectedtables


        customerquery = "INSERT INTO Customer (Customer.CustomerID, Customer.FirstName, Customer.LastName, Customer.EmailAddress) VALUES
({0},'{1}','{2}','{3}');".format(

            customerid, firstname, lastname, email

        )


        reservationquery = "INSERT INTO Reservation (Reservation.ReservationID, Reservation.CustomerID, Reservation.AccountID, Reservation.Time,
Reservation.Date, Reservation.NumberofPeople) VALUES ({0},{1},{2},'{3}','{4}',{5});".format(

            reservationid, customerid, accountid, bookingtime, date, partysize

        )


        cnxn = pyodbc.connect(cs)

        cursor = cnxn.cursor()
```

```
        if self.customercheck(email) == False:
            cursor.execute(customerquery)
        cursor.execute(reservationquery)
        for i in range(0, len(tables)):
            tablereservationquery = "INSERT INTO TableReservation (TableReservation.ReservationID,TableReservation.TableID) VALUES ({0},{1});".format(
                reservationid, tables[i]
            )
            cursor.execute(tablereservationquery)
        cursor.commit()
        cnxn.close()
        bookingconfirmation.insertbookingdetails()
        Mainwindow.ui.DisplayWidget.setCurrentIndex(9)

    def customercheck(self, email):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = f"SELECT * from Customer WHERE EmailAddress = '{email}';"
        cursor.execute(query)
        emailpresent = cursor.fetchone()
        if emailpresent:
```

```
            return True
        else:
            return False


    def getreservationid(self):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        indexreservations = "SELECT COUNT(*) FROM Reservation;"
        cursor.execute(indexreservations)
        numofreservations = cursor.fetchone()
        reservationid = numofreservations[0] + 1
        cnxn.close()
        return reservationid


    def cleardetails(self):
        self.ui.leFName.clear()
        self.ui.leLName.clear()
        self.ui.leEmail.clear()
```

```
################################################
class Bookings(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Booking()
        self.ui.setupUi(self)
        self.datepage()
        self.ui.pbDate.clicked.connect(self.datepage)
        self.ui.pbSize.clicked.connect(self.sizepage)
        self.ui.pbTime.clicked.connect(self.timepage)
        self.ui.pbTable.clicked.connect(self.tablepage)
        self.ui.pbGuest.clicked.connect(self.guestpage)

    def datepage(self):
        self.ui.BookingWidget.setCurrentIndex(0)

    def sizepage(self):
        self.ui.BookingWidget.setCurrentIndex(1)

    def timepage(self):
```

```
        self.ui.BookingWidget.setCurrentIndex(2)


    def tablepage(self):
        seatingwindow.settableimages("tableselectionwindow", "checkBox")
        self.ui.BookingWidget.setCurrentIndex(3)
        partysizewindow.getpartysize()
        tableselectionwindow.ui.lblGuestsLeft.setText(
            f"0 Guests Seated out of {tableselectionwindow.partysize}"
        )
        tableselectionwindow.disabletables()


    def guestpage(self):
        guestdetailswindow.displayreservationdetails()
        self.ui.BookingWidget.setCurrentIndex(4)


    def resetbuttons(self):
        exec("bookingwindow.ui.pbGuest.setEnabled(False)")
        exec("bookingwindow.ui.pbTable.setEnabled(False)")
        exec("bookingwindow.ui.pbTime.setEnabled(False)")
        exec("bookingwindow.ui.pbSize.setEnabled(False)")
```

```
################################################
class BookingConfirmation(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_BookingConfirmation()
        self.ui.setupUi(self)
        self.ui.pbHome.clicked.connect(Mainwindow.bookings_clicked)


    def insertbookingdetails(self):
        fullname = (
            str(guestdetailswindow.ui.leFName.text())
            + " "
            + str(guestdetailswindow.ui.leLName.text())
        )
        numbergoing = str(partysizewindow.getpartysize())
        date = str(dateselectionwindow.getdate())
        bookingtime = str(timeselectionwindow.gettime())
        tables = ", ".join(str(x) for x in tableselectionwindow.selectedtables)
```

```
        self.ui.lblNameInsert.setText(fullname)
        self.ui.lblDateinsert.setText(date)
        self.ui.lblTimeInsert.setText(bookingtime)
        self.ui.lblGoinginsert.setText(numbergoing)
        self.ui.lblTableInsert.setText(tables)



################################################
class Seating(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Seating()
        self.ui.setupUi(self)
        self.outputtablebuttons()


    def setactivetableimages(self):
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        query = """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'"""
```

```
        cursor.execute(query)
        tables = cursor.fetchall()
        for i in range(1, len(tables) + 1):
            stylesheet = f"background-image: url(:/Tables/TableIcon{tables[i-1][1]}Selected.png);background-position: center; background-repeat: no-repeat; background-color: transparent;border: 1px solid transparent;"
            exec(f"self.pushButton{tables[i-1][0]}.setStyleSheet(stylesheet)")
        cnxn.close()


    def outputtablebuttons(self):
        tablenum = 0
        t = 70
        for n in range(1, 4):  # loop to make 10 rows
            columncoordinate = 270
            for i in range(1, 3):  # loop to print 20 seats per row
                tablenum += 1
                exec(
                    "self.pushButton{0} = QtWidgets.QPushButton(self)".format(tablenum)
                )
                exec(
                    "self.pushButton{0}.setGeometry(QtCore.QRect({1}, {2}, 51, 51))".format(
```

```
            tablenum, columncoordinate, t
        )
    )
    exec("self.pushButton{0}.setEnabled(True)".format(tablenum))
    exec("self.pushButton{0}.setText('{0}')".format(tablenum))
    exec(
        "self.pushButton{0}.setIconSize(QtCore.QSize(9, 8))".format(
            tablenum
        )
    )
    exec("self.pushButton{0}.setObjectName('Table{0}')".format(tablenum))
    exec(
        "self.pushButton{0}.clicked.connect(lambda: seatingwindow.displayorderscreen(str({0})))".format(
            tablenum
        )
    )
    exec(
        "self.pushButton{0}.clicked.connect(lambda: orderwindow.gettableid(str({0})))".format(
            tablenum
        )
```

```
            )
            exec(
               "self.pushButton{0}.clicked.connect(lambda: orderwindow.outputorderedditems())".format(
                  tablenum
               )
            )
            exec(
               "self.pushButton{0}.clicked.connect(lambda: orderwindow.startermenuscreen())".format(
                  tablenum
               )
            )
            columncoordinate = 730
         t = t + 75
      for k in range(1, 4):  # loop to make 10 rows
         columncoordinate = 270
         for j in range(1, 4):  # loop to print 20 seats per row
            tablenum += 1
            exec(
               "self.pushButton{0} = QtWidgets.QPushButton(self)".format(tablenum)
            )
```

```
            exec(
                "self.pushButton{0}.setGeometry(QtCore.QRect({1}, {2}, 51, 51))".format(
                    tablenum, columncoordinate, t
                )
            )
            exec("self.pushButton{0}.setEnabled(True)".format(tablenum))
            exec(f"self.pushButton{tablenum}.setText('{tablenum}')")
            exec(
                "self.pushButton{0}.setIconSize(QtCore.QSize(9, 8))".format(
                    tablenum
                )
            )
            exec("self.pushButton{0}.setObjectName('Table{0}')".format(tablenum))
            exec(
                "self.pushButton{0}.clicked.connect(lambda: seatingwindow.displayorderscreen(str({0})))".format(
                    tablenum
                )
            )
            exec(
                "self.pushButton{0}.clicked.connect(lambda: orderwindow.gettableid(str({0})))".format(
```

```
                tablenum
            )
        )
        exec(
            "self.pushButton{0}.clicked.connect(lambda: orderwindow.outputordereditems())".format(
                tablenum
            )
        )
        exec(
            "self.pushButton{0}.clicked.connect(lambda: orderwindow.startermenuscreen())".format(
                tablenum
            )
        )
        columncoordinate = columncoordinate + 230
    t = t + 75


def settableimages(self, window, widget):
    cnxn = pyodbc.connect(cs)
    cursor = cnxn.cursor()
    query = 'SELECT * FROM "TABLE"'
```

```
        cursor.execute(query)
        tables = cursor.fetchall()
        for i in range(1, len(tables) + 1):
            if widget == "pushButton":
                stylesheet = f"background-image: url(:/Tables/TableIcon{tables[i-1][1]}Unavailable.png);background-position: center; background-repeat: no-repeat; background-color: transparent;border: 1px solid transparent;"
            elif widget == "checkBox":
                stylesheet = (
                    "QCheckBox::indicator:unchecked {image: url(:/Tables/TableIcon"
                    + "{0}".format(tables[i - 1][1])
                    + "Available.png);} QCheckBox::indicator:checked {image: url(:/Tables/TableIcon"
                    + "{0}".format(tables[i - 1][1])
                    + "Selected.png);}"
                    + "QCheckBox::indicator:disabled {image: url(:/Tables/TableIcon"
                    + "{0}".format(tables[i - 1][1])
                    + "Unavailable.png);}"
                )
            exec(f'{window}.{widget}{i}.setStyleSheet("{stylesheet}")')

    def displayorderscreen(self, tablenum):
```

```
        Mainwindow.ui.DisplayWidget.setCurrentIndex(3)
        orderwindow.ui.lblTable.setText(f"Table {tablenum}")



################################################
class RemoveBooking(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_RemoveBooking()
        self.ui.setupUi(self)
        self.ui.pbRemove.clicked.connect(self.remove_clicked)
        self.ui.pbCancel.clicked.connect(self.cancel_clicked)

    def cancel_clicked(self):
        removebooking.close()

    def remove_clicked(self):
        reservationid = removeitemwindow.getselectedItem(
            0, "twReservations", "calendarwindow"
        )
```

```
        query = f"DELETE FROM Reservation WHERE ReservationID = {reservationid}"
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        cursor.execute(query)
        cursor.commit()
        cnxn.close()
        calendarwindow.runreservationquery()
        removebooking.close()



################################################
class Calendar(QDialog):
    def __init__(self):
        super(QDialog, self).__init__()
        self.ui = Ui_Calendar()
        self.ui.setupUi(self)
        self.ui.cwCalendar.clicked.connect(self.runreservationquery)
        self.setmaximumdate()
        self.ui.twReservations.doubleClicked.connect(self.removebooking)
```

```
    def setmaximumdate(self):
        today = date.today()
        maximum = today.replace(year=today.year + 1)
        self.ui.cwCalendar.setMaximumDate(maximum)


    def removebooking(self):
        reservationid = removeitemwindow.getselectedItem(
            0, "twReservations", "calendarwindow"
        )
        query = f"""SELECT DISTINCT Reservation.NumberofPeople, CONCAT(Customer.FirstName,' ',Customer.LastName), SUBSTRING(convert(varchar, Reservation.Time,108),1,5), Reservation.Date FROM Customer INNER JOIN Reservation ON Customer.CustomerID=Reservation.CustomerID INNER JOIN TableReservation ON TableReservation.ReservationID=Reservation.ReservationID WHERE Reservation.ReservationID = '{reservationid}';"""
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        cursor.execute(query)
        queryresults = cursor.fetchone()
        cnxn.close()
        numofpeople = queryresults[0]
        name = queryresults[1]
        time = queryresults[2]
        date = queryresults[3]
```

```
        removebooking.ui.lblConfirmation.setText(
            f"""Are you sure you want to remove this booking:
Name: {name}
Number of Guests: {numofpeople}
Date: {date}
Time: {time}"""
        )
        removebooking.show()


    def runreservationquery(self):
        date = self.ui.cwCalendar.selectedDate()
        date = date.toPyDate()
        query = """SELECT Reservation.ReservationID, 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);""".format(
            date
        )
        try:
            self.executequery(query)
```

```
        except:
            pass


    def executequery(self, query):
        self.ui.twReservations.setRowCount(10)
        self.ui.twReservations.setColumnCount(5)
        header = self.ui.twReservations.horizontalHeader()
        header.setSectionResizeMode(2, QtWidgets.QHeaderView.ResizeToContents)
        self.ui.twReservations.clear()
        self.ui.twReservations.setHorizontalHeaderLabels(
            ["ReservationID", "People", "Table", "Name", "Time"]
        )
        cnxn = pyodbc.connect(cs)
        cursor = cnxn.cursor()
        cursor.execute(query)
        rows = cursor.fetchall()
        noRow = 0
        for tuple in rows:
            noCol = 0
            for column in tuple:
```

```
            data = QTableWidgetItem(str(column))
            self.ui.twReservations.setItem(noRow, noCol, data)
            noCol += 1
        noRow += 1
    self.ui.twReservations.setRowCount(noRow)
    self.ui.twReservations.setColumnCount(noCol)
    cnxn.close()



################################################
################################################
if __name__ == "__main__":
    import sys

    sys._excepthook = sys.excepthook

    def exception_hook(exctype, value, traceback):

        print(exctype, value, traceback)
        sys._excepthook(exctype, value, traceback)
```

```
        sys.exit(1)


    sys.excepthook = exception_hook
    app = QApplication(sys.argv)


    Mainwindow = MainWindow()


    bookingwindow = Bookings()
    calendarwindow = Calendar()
    seatingwindow = Seating()
    orderwindow = OrderScreen()
    managementwindow = ManagementScreen()
    bookingconfirmation = BookingConfirmation()


    statisticswindow = Statistics()
    staffwindow = Staff()
    editmenuwindow = EditMenuScreen()
    removeitemwindow = RemoveItemScreen()
    additemwindow = AddItemScreen()
    alteritemwindow = AlterMenuItem()
```

```
    starterspage = StartersPage()
    mainspage = MainsPage()
    dessertspage = DessertsPage()
    drinkspage = DrinksPage()
    closeconfirmation = CloseTableConfirmation()
    changequantitywindow = ChangeQuantity()

    loginpage = LoginPage()

    dateselectionwindow = DateSelection()
    partysizewindow = PartySizeSelection()
    timeselectionwindow = TimeSelection()
    tableselectionwindow = TableSelection()
    guestdetailswindow = GuestDetails()

    removeconfirmationscreen = RemoveConfirmation()
    addconfirmationscreen = AddConfirmation()
    alterconfirmation = AlterConfirmation()
    removebooking = RemoveBooking()
```

Mainwindow.ui.DisplayWidget.addWidget(bookingwindow)

Mainwindow.ui.DisplayWidget.addWidget(calendarwindow)

Mainwindow.ui.DisplayWidget.addWidget(seatingwindow)

Mainwindow.ui.DisplayWidget.addWidget(orderwindow)

Mainwindow.ui.DisplayWidget.addWidget(managementwindow)

Mainwindow.ui.DisplayWidget.addWidget(editmenuwindow)

Mainwindow.ui.DisplayWidget.addWidget(removeitemwindow)

Mainwindow.ui.DisplayWidget.addWidget(additemwindow)

Mainwindow.ui.DisplayWidget.addWidget(alteritemwindow)

Mainwindow.ui.DisplayWidget.addWidget(bookingconfirmation)

Mainwindow.ui.DisplayWidget.addWidget(statisticswindow)

Mainwindow.ui.DisplayWidget.addWidget(staffwindow)

bookingwindow.ui.BookingWidget.addWidget(dateselectionwindow)

bookingwindow.ui.BookingWidget.addWidget(partysizewindow)

bookingwindow.ui.BookingWidget.addWidget(timeselectionwindow)

bookingwindow.ui.BookingWidget.addWidget(tableselectionwindow)

bookingwindow.ui.BookingWidget.addWidget(guestdetailswindow)

```
    orderwindow.ui.orderwidget.addWidget(starterspage)
    orderwindow.ui.orderwidget.addWidget(mainspage)
    orderwindow.ui.orderwidget.addWidget(dessertspage)
    orderwindow.ui.orderwidget.addWidget(drinkspage)

    style = """
    QWidget{
        background: #212121;
        color: white;
        font-family: 'Roboto Medium';
    }
    QLabel{
        background-color: transparent;
    }

    QLineEdit{
        padding: 1px;
        background: #2E2E2E;
        border: 1px solid #2E2E2E;
        border-radius: 4px;
```

```
}
QPushButton{
    padding: 1px;
    background-color: #383838;
    border: 1px solid #2E2E2E solid;
    border-radius: 4px;
    transition-duration: 2000ms;
}
QTableWidget {
     background-color: #474747;
     padding: 10px;
     border-radius: 5px;
     gridline-color: #2E2E2E;
     border-bottom: 1px solid #2E2E2E;
}
QHeaderView::section:horizontal {
   border: 1px solid #2E2E2E;
   background-color: #383838;
   color: #FFF;
}
```

```
QScrollBar:vertical {
border: 10px;
background: rgb(52, 59, 72);
height: 20px;
margin: 5px 21px 10 21px;
border-radius: 10px;
}
QScrollBar::handle:horizontal {
    background: rgb(57, 65, 80);
    min-width: 25px;
    border-radius: 4px
}
#PartySize .QCheckBox::indicator {
        width: 15px;
        height: 15px;
        background-color: #474747;
        border-radius: 5px;
        border-style: solid;
        border-width: 1px;
        border-color: #474747;
```

```
    }
    #PartySize .QCheckBox::indicator:checked {
        background-color:#939393;
        border-color:#939393;
    }

    #MainWindow .QFrame .QPushButton{
        padding: 1px;
        background-color: #474747;
        border: 1px solid #2E2E2E solid;
        border-radius: 2px;
        transition-duration: 2000ms;
    }
    #MainWindow .QPushButton:disabled{
        padding: 1px;
        color: #474747;
        background-color: #272727;
        border: 1px solid #2E2E2E solid;
        border-radius: 2px;
        transition-duration: 2000ms;
```

```
    }
  QCalendarWidget QAbstractItemView:enabled
  {
        font-size:13px;
        color: white;
        background-color: #383838;
        alternate-background-color: #383838;
        selection-background-color: #474747;
        selection-color: white;
        padding: 2px;
        border-radius: 5px;
  }
  QCalendarWidget QAbstractItemView :disabled
  {
    background-color: #474747;
    alternate-background-color: #eee;
    selection-background-color: #0F4A8C;
    selection-color:#fff;
  }
  QCalendarWidget QWidget#qt_calendar_navigationbar{
```

```
    background-color: #C092FE;
    padding: 2px;
        border-radius: 5px;
}
QCalendarWidget QToolButton {
        color: white;
        background-color: #C092FE;
}
#Booking .QPushButton:enabled{
    background: #474747;
    padding: 1px;
    border: 1px solid #2E2E2E solid;
    border-radius: 4px;
    border-left: 1px solid #2E2E2E solid;
    border-right: 1px solid #2E2E2E solid;
    border-bottom: 5px solid #C092FE solid;
    transition-duration: 2000ms;
  }
#Booking .QPushButton:disabled{
    background: #474747;
```

```
        padding: 1px;
        color: white;
        border: 1px solid #2E2E2E solid;
        border-radius: 4px;
        border-left: 1px solid #2E2E2E solid;
        border-right: 1px solid #2E2E2E solid;
        border-bottom: 5px solid #272727 solid;
        transition-duration: 2000ms;
    }
  QListWidget {
         background-color: #474747;
         padding: 10px;
         border-radius: 5px;
         gridline-color: rgb(242,242,247);
         border-bottom: 1px solid rgb(44, 49, 60);
  }
  QSpinBox{
    padding: 1px;
    border: 2px solid #474747;
    border-radius: 4px;
```

```
    background-color: #474747;
}
QComboBox{
    padding: 1px;
    color: #8b8b8b;
    border: 2px solid #2E2E2E;
    border-radius: 4px;
    background-color: #2E2E2E;
}
#AlterConfirmation .QLabel{
    background-color: transparent;
}
#AlterConfirmation .QLineEdit{
    padding: 1px;
    background: #2E2E2E;
    border: 1px solid #2E2E2E;
    border-radius: 4px;
}
#AlterConfirmation .QComboBox{
    padding: 1px;
```

```
        color: #8b8b8b;
        border: 2px solid #2E2E2E;
        border-radius: 4px;
        background-color: #2E2E2E;
    }
    #Seating .QPushButton{
            color: black;
            fontsize: 15px;
        }
    """
    Mainwindow.setStyleSheet(style)
    loginpage.setStyleSheet(style)
    closeconfirmation.setStyleSheet(style)
    removeconfirmationscreen.setStyleSheet(style)
    addconfirmationscreen.setStyleSheet(style)
    alterconfirmation.setStyleSheet(style)
    removebooking.setStyleSheet(style)
    changequantitywindow.setStyleSheet(style)
    loginpage.show()
    sys.exit(app.exec())
```

Joshua Heffernan
7865

The College of Richard Collyer
Centre: 65131

Joshua Heffernan
7865

The College of Richard Collyer
Centre: 65131